به نام خداوند بخشنده‌ی مهربان

مفاهیم پیشرفته علوم کامپیوتر

ولادستون فریرا فیلیو

موتو پیکتت

مترجم: علی ناصراسدی

علوم کامپیوتر شباهت‌های زیادی با فیزیک دارد. هر دو در مورد نحوه‌ی عملکرد جهان در یک سطح نسبتاً بنیادین بحث می‌کنند. تفاوت آن‌ها در این است که در فیزیک شما قرار است بفهمید جهان چگونه ساخته شده، ولی در علوم کامپیوتر شما جهان را خلق می‌کنید. در ریاضیات، مانند برنامه‌نویسی، هر چیز تا زمانی پابرجا است که خودسازگار باشد. شما می‌توانید مجموعه‌ای از معادلات داشته باشید که در آن سه به علاوه‌ی سه برابر با دو باشد. هر کاری که بخواهید می‌توانید انجام دهید.

– لینوس توروالدز[1]

در حال توضیح این که عشق او به کامپیوتر از کجا سرچشمه می‌گیرد.

[1] Linus Torvalds (-1969): مهندس نرم‌افزار فنلاندی-آمریکایی، توسعه‌دهنده‌ی هسته‌ی لینوکس و نرم‌افزار گیت. م.

پیشگفتار

من هرگز اصطلاح «علوم کامپیوتر» را دوست نداشتم. دلیل اصلی که من آن را دوست ندارم این است که چنین چیزی وجود ندارد. علوم کامپیوتر آش شله‌قلمکاری از حوزه‌های مرتبط است که در اثر یک حادثه‌ی تاریخی، مانند یوگسلاوی، کنار هم قرار گرفته‌اند.

– پل گراهام[1]

بیشتر پیشرفت‌های فناوری عصر ما در دنیای دیجیتال جدیدی که توسط برنامه‌نویسان ایجاد شده است، اتفاق می‌افتند. دانشمندان کامپیوتر زمینه‌های مطالعاتی مختلف را به منظور توانمندسازی این دنیای جدید ترکیب می‌کنند. این کتاب به بررسی مبانی برخی از این زمینه‌ها از جمله شبکه، رمزنگاری و علم داده می‌پردازد.

ما با داستان چگونگی ایجاد پیوند بین دو کامپیوتر برای اشتراک‌گذاری اطلاعات شروع کرده و شما را به سمت ماجرای ظهور ایمیل و وب هدایت می‌کنیم. رمزنگاری را بررسی خواهیم کرد و خواهیم فهمید که چگونه اینترنت و سایر سیستم‌هایی که با داده‌های خصوصی سروکار دارند، ایمن می‌شوند. سپس، یاد خواهیم گرفت که چگونه می‌توان دانش را از داده‌های خام به دست آورد و چگونه می‌توان ماشین ها را برای پیش‌بینی آینده آموزش داد.

امیدواریم این داستان‌ها شما را با مفاهیم مهمی آشنا کنند که می‌توانند به یک اندازه برای برنامه‌نویسان و علاقه‌مندان به فناوری مفید باشند. هدف ما پوشش دادن آن چیزهایی است که مبتدیان برای درک سریع شبکه، امنیت و علم داده به آن‌ها نیاز دارند، البته بدون سختگیری‌های مرسوم دانشگاهی که گاهی این موضوعات را غیرقابل تحمل می‌کنند.

نوشتن این کتاب به واسطه‌ی حمایت از کتاب قبلی ما، *مبانی و مفاهیم علوم کامپیوتر*، مقدور شد. ما اولین کتاب خود را برای توضیح اصول اساسی علوم کامپیوتر نوشته بودیم. خوانندگان مشتاق ما موارد بیشتری خواستند، بنابراین ما به کار برگشتیم! این بار، هسته‌ی رشته خود را بررسی نمی‌کنیم، بلکه جهان‌های جدیدی را که ما را قادر به ایجاد آن‌ها کرده است، بررسی خواهیم کرد.

[1] Paul Graham (1964-): دانشمند انگلیسی در علوم کامپیوتر و از توسعه‌دهندگان زبان لیسپ و وب‌اپلیکیشن. م.

شکل ۱: «داده‌ها نفت جدید هستند»، اثر آمیت درنگل و ایوانو نارداکیونه[۱].

آیا این کتاب برای من است؟

اگر یک برنامه‌نویس تازه کار هستید، این کتاب برای شما نوشته شده است. این کتاب نیازی به تجربه‌ی برنامه‌نویسی ندارد، زیرا ایده‌ها و مکانیزم‌هایی را به صورت پایه‌ای ارائه می‌دهد: ما می‌خواهیم شما یاد بگیرید که چیزهای جالب چگونه کار می‌کنند. اگر کنجکاو هستید و می‌خواهید بدانید اینترنت چگونه ساخته می‌شود، چگونه هکرها به سیستم‌های کامپیوتری حمله می‌کنند، یا چرا داده‌ها طلای قرن بیست‌ویکم هستند، این کتاب را ارزشمند خواهید یافت. برای کسانی که قبلاً علوم کامپیوتر خوانده‌اند، این کتاب یک جمع‌بندی عالی برای تثبیت دانش و تخصص آن‌ها است.

تقدیر و تشکر

ما عمیقاً از همه‌ی کسانی که از تلاش چند ساله‌ی ما برای نوشتن این کتاب حمایت کردند، سپاسگزاریم. به ویژه از آبنر مارسیانو، آندره لمبرت، کایو ماگنو، کارلوتا فابریس، دامیان هیرش، دانیل استوری، ادواردو باربوسا، گابریل پیکتت، گیلرمه ماتار، ژاکلین ویلسون، لئوناردو کونگوندس، لوید کلارک، مایکل اولمان، رافائل آلمه و روحان بیدارت تشکر می‌کنیم. در نهایت، ما از کلر مارتین، مصحح، و پدرو نتو، تصویرگر، سپاسگزاریم که این کتاب را بسیار بهتر کردند.

باشد که جهان‌های بسیاری بسازید،

والد و موتو

[۱] این کاریکاتور به عبارت Data Driven به معنای داده‌محور و نزدیکی آن به Drive به معنی راندن اشاره دارد. م.

فصل ۱

اتصالات

> این یک سیستم کاملاً توزیع شده است، هیچ کنترل مرکزی وجود ندارد. تنها دلیلی که کار می‌کند این است که همه تصمیم گرفتند از مجموعه پروتکل‌های مشابهی استفاده کنند.
>
> – وینتون سرف[۱]

انسان‌ها مشتاق ارتباطات هستند و ظهور انقلاب دیجیتال به ما این قدرت را داده است که بیش از هر زمان دیگری به هم متصل باشیم. اینترنت آزادی اقتصادی و سیاسی بی‌سابقه و همچنین ابزارهای قدرتمندی را برای کنترل و سلطه بر روی میلیاردها انسان فراهم کرده است. با این حال، اکثریت قریب به اتفاق ما از عملکرد درونی آن غافل هستیم.

افراد ماهری که می‌توانند کامپیوترها را برای استفاده از اینترنت برنامه‌ریزی کنند، پیشتازان انقلاب دیجیتال هستند. این فصل به شما نحوه‌ی کار اینترنت را آموزش می‌دهد، بنابراین می‌توانید به این گروه برگزیده بپیوندید.

شما یاد خواهید گرفت که:

- کامپیوترها را برای ساختن یک شبکه به هم **پیوند** دهید،
- شبکه‌ها را با استفاده از پروتکل **اینترنت** با هم ترکیب کنید،
- مکان یک گیرنده را بر اساس **آدرس** اینترنتی وی مشخص کنید،
- یک **مسیر** را بر روی اینترنت به آن مکان بیابید،
- داده‌ها را بین برنامه‌های کابردی راه‌دور **منتقل** کنید.

[۱] Vinton Cerf (1943-): دانشمند آمریکایی در علوم کامپیوتر و یکی از پایه‌گذاران پروتکل TCP/IP. م.

پیش از ظهور اینترنت، ارتباط راه دور بین دو طرف نیاز به یک پیوند فیزیکی مستقیم داشت. در دهه ۱۹۵۰، هر تلفن دارای سیمی بود که مستقیماً به ایستگاه مرکزی منتهی می‌شد. برای برقراری تماس، یک اپراتور مجبور بود سیم دو تلفن را به صورت فیزیکی به هم وصل کند. برای تماس‌های راه دور، سیم‌ها بین ایستگاه‌های دور قرار می‌گرفت و چندین اپراتور در مکان‌های مختلف مجبور بودند زنجیره‌ای از سیم‌ها را که دو تلفن را به هم وصل می‌کردند، به صورت فیزیکی به هم متصل کنند.

اینترنت این مشکل را از حل کرد. برای ایجاد پیوندهای مستقیم و انحصاری سیم‌ها به صورت مکرر دستکاری نمی‌شوند. درعوض، اطلاعات گام به گام از طریق زنجیره‌ای از دستگاه‌های متصل به هم مکرراً جابجا می‌شوند تا زمانی که به مقصد برسند. این امر نیاز به اپراتورهای سیم و هماهنگی مرکزی را از بین می‌برد. همچنین، سیم‌ها دیگر محدود به ارائه‌ی یک اتصال واحد نیستند و اتصالات بسیاری به صورت همزمان می‌توانند سیم یکسانی را به اشتراک بگذارند. این پدیده اجازه می‌دهد تا ارتباطات جهانی سریع، ارزان و در دسترس باشند.

با این حال، فناوری مدرن شبکه‌ها پیچیده‌تر از تلفن اولیه است. این فناوری دارای چندین لایه‌ی متوالی است که هر یک بر روی لایه‌ی قبلی قرار گرفته است. بیایید ببینیم اتصالات چگونه در این سطوح مختلف به وجود می‌آیند و برای این کار از ابتدایی ترین لایه شروع می‌کنیم.

۱-۱- پیوندها

ارتباط مستقیم بین دو کامپیوتر از طریق یک **رسانه‌ی انتقال**[۱] حاصل می‌شود: یک کانال فیزیکی که در آن سیگنال‌ها می توانند جریان داشته باشند. این کانال ممکن است یک سیم مسی حامل جریان الکتریکی، یک کابل فیبر نوری هدایت‌کننده‌ی نور، یا هوای میزبان امواج رادیویی باشد. هر کامپیوتر متصل دارای یک **رابط شبکه**[۲] برای ارسال و دریافت سیگنال در رسانه انتقال است. به عنوان مثال، تلفن‌های همراه دارای یک تراشه‌ی رادیویی و آنتن برای کنترل سیگنال‌های رادیویی جاری در هوا هستند.

به منظور برقراری ارتباط، رابط‌های شبکه باید در مورد قوانینی که هنگام ارسال و دریافت سیگنال‌ها باید رعایت شوند، توافق کنند. این مجموعه‌ی قوانین **لایه‌ی پیوند**[۳] نامیده می‌شوند.

هنگامی که یک رسانه *به طور انحصاری* دو کامپیوتر را به هم متصل می‌کند، می‌گوییم که آن‌ها یک اتصال نقطه به نقطه را حفظ می‌کنند، و لایه‌ی پیوند آن‌ها به ابتدایی‌ترین مجموعه‌ی قوانین متکی است:

---

۱ Transmission Medium
۲ Network Interface
۳ Link Layer

پروتکل نقطه به نقطه[۱]. این پروتکل تضمین می‌کند که دو کامپیوتر می‌توانند یکدیگر را شناسایی کرده و داده‌ها را به طور دقیق مبادله کنند.

با این حال، کامپیوترهای متصل همیشه نمی‌توانند از داشتن چنین پیوند انحصاری لذت ببرند. اغلب، آن‌ها باید رسانه‌ی انتقال را با چندین کامپیوتر دیگر به اشتراک بگذارند.

شکل ۱-۱: اگر دو رابط شبکه در یک رسانه‌ی انتقال مشترک باشند و بر روی قوانین ارتباط توافق کنند، یک **پیوند** بین دو رابط شبکه برقرار می‌شود.

پیوندهای اشتراکی

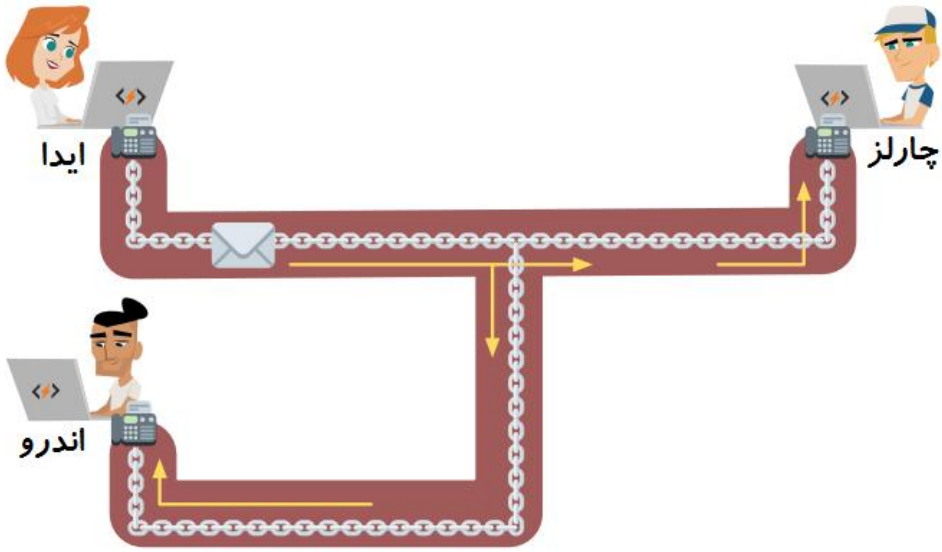
یکی از راه‌های پیوند دادن کامپیوترها در یک دفتر، وصل کردن هر یک از آنها به یک هاب با سیم است. هاب به طور فیزیکی تمام سیم‌هایی را که به آن می‌رسند به هم متصل می‌کند، بنابراین سیگنال ارسال شده توسط یک کامپیوتر توسط *بقیه* شناسایی می‌شود! این پدیده در وای‌فای خانه‌ی شما نیز اتفاق می‌افتد، زیرا فرکانس رادیویی یکسانی توسط همه‌ی دستگاه‌های متصل استفاده می‌شود. اگر همه‌ی آن‌ها به طور همزمان از رسانه استفاده کنند، ارتباطات ممکن است دچار آشفتگی شود.

لایه‌ی پیوند شامل مجموعه‌ای از قوانین برای تعریف نحوه‌ی اشتراک‌گذاری رسانه‌های ارتباطی توسط کامپیوترها است که که به طور مناسبی کنترل دسترسی به رسانه[۲] نامیده می‌شود. این قوانین دو چالش اصلی را حل می‌کنند:

---

[۱] Point to Point Protocol یا **PPP**

[۲] Medium Access Control یا **MAC**



شکل ۱-۲: پیام ارسال شده در پیوند اشتراکی توسط همه شناسایی خواهد شد.

تصادم[۱]: اگر دو کامپیوتر همزمان سیگنالی را از طریق یک رسانه ارسال کنند، تداخل حاصل هر دو انتقال را مختل می‌کند. به چنین رویدادهایی **تصادم** می‌گویند. مشکل مشابه زمانی رخ می‌دهد که گروه دوستان یا خانواده‌ی شما با یکدیگر صحبت می‌کنند و هیچ صدایی به وضوح شنیده نمی‌شود.

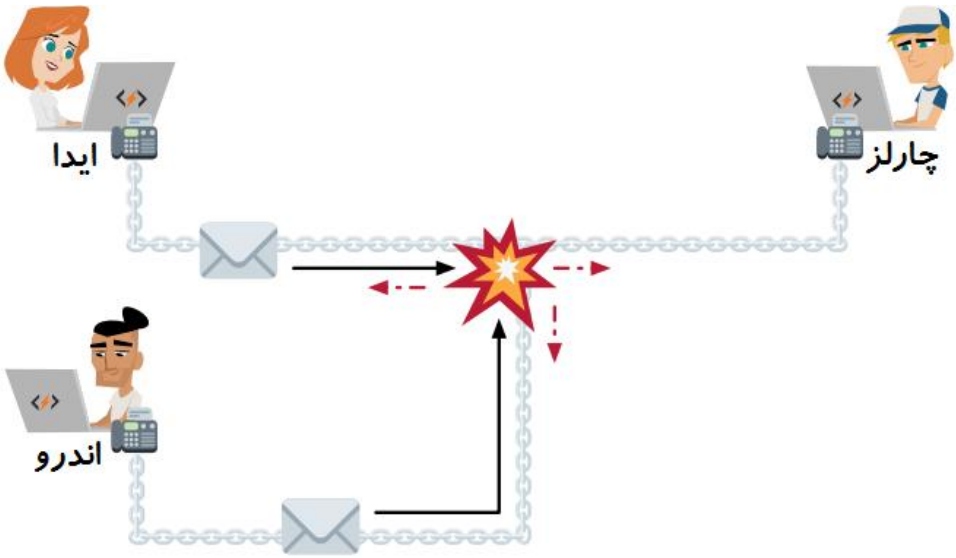
روش‌هایی برای جلوگیری از تصادم‌ها وجود دارد. اول، فقط زمانی شروع به انتقال سیگنال کنید که هیچ کامپیوتر دیگری سیگنال ارسال نمی‌کند. دوم، نظارت کردن بر ارتباطات، اگر تصادمی رخ داد، قبل از تلاش برای ارسال مجدد، برای مدت کوتاهی که تصادفی انتخاب می‌شود، منتظر بمانید.

این روش‌ها محدودیت‌هایی دارند. هنگامی که تلاش‌های زیادی برای انتقال از طریق یک رسانه انجام می‌شود، تصادم‌ها بی‌وقفه رخ می‌دهند. در این شرایط و هنگامی که تصادم‌های بیش از حد ارتباطات را خراب می‌کنند، می‌گوییم پیوند **اشباع**[۲] می‌شود. آیا تا به حال در یک مکان شلوغ به علت عدم ارسال پیامک یا برقراری تماس با تلفن همراه‌تان، ناامید شده‌اید؟ این پدیده ممکن است در صورتی اتفاق بیفتد که تعداد زیادی از تلفن‌ها به طور همزمان سعی در برقراری ارتباط داشته باشند و پیوند سلولی اشباع شود.

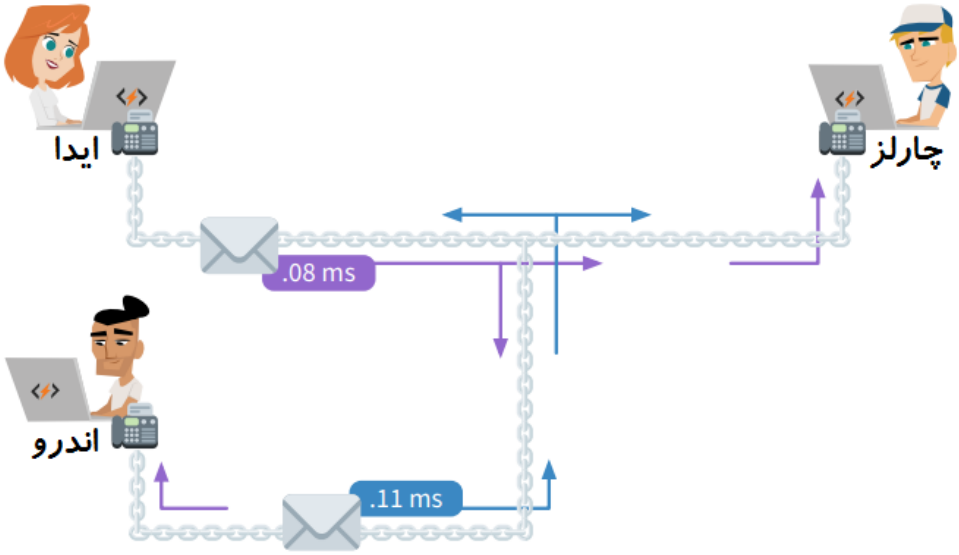
---

۱ Collision

۲ Saturated



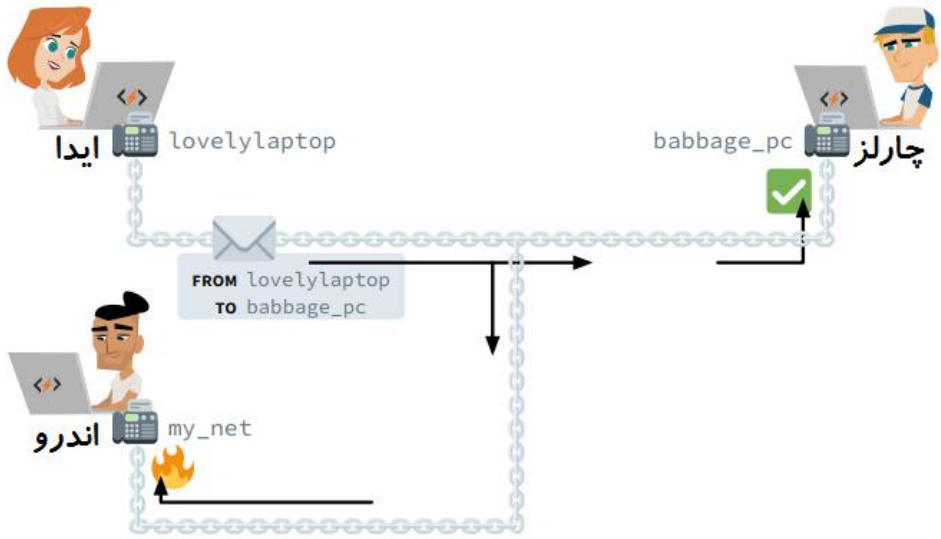
شکل ۱-۳: تصادم بین ایدا و اندرو.



شکل ۱-۴: ایدا و اندرو هر دو و بعد از مدت زمانی تصادفی مجدداً ارسال می‌کنند.

آدرس‌دهی فیزیکی: ایدا و چارلز پیوند مستقیمی بین کامپیوترهای خود دارند. ایدا می‌خواهد با چارلز صحبت کند، بنابراین سیگنالی را با پیام خود از طریق رسانه ارسال می‌کند. با این حال، رسانه به اشتراک

گذاشته شده است، بنابراین هر کسی که به رسانه متصل باشد، پیام را دریافت می‌کند. چگونه کامپیوترهای دیگر می‌توانند بفهمند سیگنالی که دریافت کرده‌اند برای آن‌ها نبوده است؟



شکل ۱-۵: رابط شبکه‌ی اندرو پیام را نادیده می‌گیرد.

رابط شبکه‌ی هر کامپیوتر دارای یک شناسه است که به عنوان **آدرس فیزیکی** یا **آدرس سخت‌افزاری** آن شناخته می‌شود. انتقال در یک رسانه‌ی اشتراکی باید با دو آدرس از این نوع آغاز شود: آدرس گیرنده و آدرس فرستنده. به محض دریافت یک انتقال، کامپیوتر می‌داند که آیا باید نادیده گرفته شود یا آن را دریافت کند و به کدام آدرس باید پاسخ دهد.

این فرایند تنها در صورتی درست کار می‌کند که آدرس‌های فیزیکی منحصربه‌فرد باشند: اگر دو کامپیوتر از آدرس `my_netinterface` استفاده کنند، به نقطه‌ی اول برمی‌گردیم. به همین دلیل، تقریباً تمام رابط‌های شبکه از یک طرح نام‌گذاری که در قوانین کنترل دسترسی رسانه تعریف شده است، پیروی می‌کنند. این آدرس‌های فیزیکی استاندارد، **آدرس های MAC** نامیده می‌شوند.

آدرس‌دهی MAC

کامپیوترها، تلفن‌های هوشمند، ساعت‌های هوشمند و تلویزیون‌های هوشمند هر کدام می‌توانند دارای رابط شبکه‌ی وای‌فای، بلوتوث و اترنت باشند. هر رابط شبکه دارای آدرس MAC منحصر به فرد خود است که در زمان تولید در سخت افزار مشخص شده است. شما نباید نگران اختصاص یک آدرس MAC

به کامپیوتر خود باشید: همیشه می‌توانید از آدرسی که همراه با رابط شبکه‌ی آن ارائه شده است استفاده کنید.

از آنجایی که آدرس‌های MAC صرفاً اعداد تصادفی بزرگی هستند، سازندگان رابط شبکه در سرتاسر جهان باید برای جلوگیری از تخصیص تصادفی یک عدد به دو دستگاه مختلف، هماهنگی لازم را انجام دهند. برای این منظور، آن‌ها به موسسه‌ی مهندسین برق و الکترونیک[۱] یا IEEE متکی هستند که به هر یک از آن‌ها محدودهی متفاوتی از آدرس‌های MAC را اختصاص می‌دهد.

یک آدرس MAC به صورت شش جفت هگزادسیمال[۲] که با دو نقطه از هم جدا شده‌اند بیان می‌شود. نیمه‌ی اول آدرس یک شناسه است که توسط IEEE به یک سازنده منحصر به فرد اختصاص داده شده است. سپس این سازنده برای هر رابط شبکه، یک نیمه‌ی دوم منحصر به فرد را انتخاب می‌کند.

```
60:8B:0E:C0:62:DE
```

در اینجا، 608B0E شماره‌ی تولیدکننده است. این شماره‌ی خاص توسط IEEE به شرکت اپل اختصاص داده شده است، بنابراین این آدرس MAC باید متعلق به یک دستگاه اپل باشد[۳]. آدرس MAC دستگاه اغلب روی برچسب چسبانده شده بر روی بسته بندی یا روی خود دستگاه، در کنار شماره سریال، نوشته می‌شود.

یک آدرس ویژه برای ارسال به همه‌ی کامپیوترها در یک رسانه رزرو شده است. به آن **آدرس پخش**[۴] می‌گویند که به صورت FF:FF:FF:FF:FF است. زمانی که سعی می‌کنید به یک دستگاه ناشناس متصل شوید از آن استفاده می‌کنید. به عنوان مثال، زمانی که کارت وای‌فای گوشی هوشمندتان غیرفعال نیست، دائماً برای پیدا کردن یک نقطه‌ی دسترسی به FF:FF:FF:FF:FF سیگنال ارسال می‌کند. نقاط دسترسی قابل کشف با آدرس MAC خود پاسخ می‌دهند تا بتوانید یک پیوند ایجاد کنید.

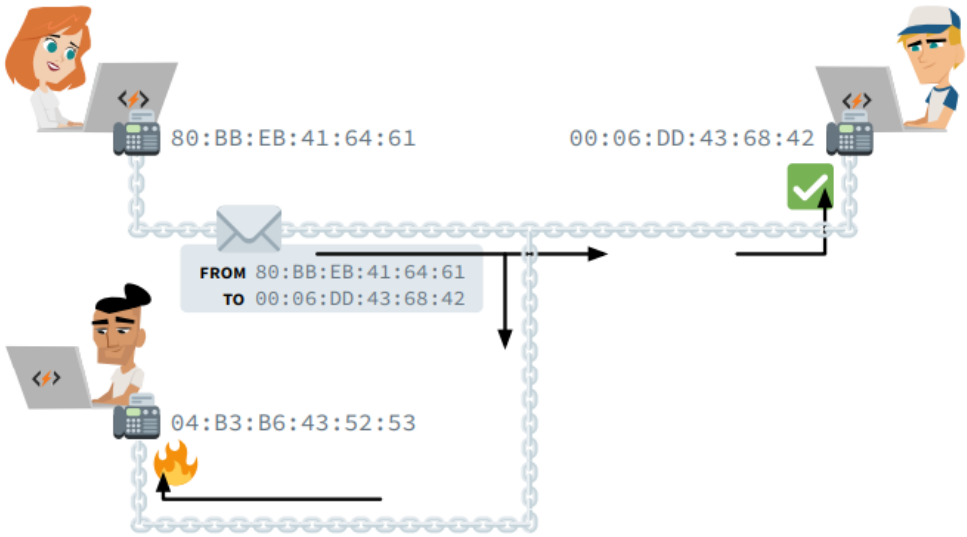
---

[۱] Institute of Electrical and Electronics Engineers یا **IEEE**: یک سازمان بین‌المللی و غیرانتفاعی که هدف آن پیشبرد فناوری به طور عام و حوزه‌های وابسته به مهندسی برق و مهندسی کامپیوتر به طور خاص است. این سازمان در سال ۱۹۶۳ بنا نهاده شده و بیش از ۴۲۳ هزار عضو در ۱۶۰ کشور دنیا دارد. م.

[۲] در زندگی روزمره، ما تقریباً همیشه اعداد را به صورت ده‌دهی بیان می‌کنیم، که در آن هر رقم یکی از این ده کاراکتر است: ۰، ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹. از سوی دیگر، دانشمندان علوم کامپیوتر علاقمند به بیان اعداد به صورت هگزادسیمال (شانزده‌شانزدهی) هستند، که در آن هر رقم می‌تواند یکی از این شانزده کاراکتر است: ، ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹،a، b، c، d، e، f.برای کسب اطلاعات بیشتر در مورد مبناهای اعداد، به پیوست I مراجعه کنید.

[۳] می‌توانید با وارد کردن شش رقم اول آدرس **MAC** یک دستگاه در **http://code.energy/mac-lookup**، بفهمید که چه شرکتی آن را تولید کرده است.

[۴] Broadcast Address



شکل ۱-۶: هر آدرس MAC منحصربه‌فرد است.

چنین پخش‌هایی که برای کشف دستگاه‌ها انجام می‌شوند، مانند سایر انتقال‌ها، حاوی آدرس MAC فرستنده هستند. بنابراین راه رفتن در خیابان با گوشی هوشمند می‌تواند مانند راه رفتن با بلندگویی باشد که نام شما را بی‌وقفه فریاد می‌زند، فقط از امواج رادیویی به جای صدا و آدرس MAC به جای نام خود استفاده می‌کنید. در سال ۲۰۱۳، ادوارد اسنودن فاش کرد که آژانس امنیت ملی[۱] با شنود انتقال اطلاعات وای‌فای در شهرهای بزرگ، حرکات افراد را زیر نظر می‌گیرد و سوابقی از محل مشاهده هر آدرس MAC را ذخیره می‌کند.

همچنین می‌توانید رابط شبکه‌ی خود را روی **حالت بی‌وقفه**[۲] تنظیم و تمام ارسال‌ها را بدون در نظر گرفتن گیرنده مورد نظر دریافت کنید. انجام این کار به شما امکان می‌دهد شبکه‌های وای‌فای پنهان را کشف کنید، آدرس‌های MAC را در منطقه‌تان فهرست کنید، و گاهی اوقات حتی محتویات ارسال‌های دیگران را بخوانید. بنابراین، مرور اینترنت از طریق یک شبکه وای‌فای ناامن می‌تواند ناامن باشد: هر کسی که در محدوده‌ی آن شبکه باشد می‌تواند آنچه را که شما پخش می‌کنید بشنود. به همین دلیل است که رمزگذاری[۳] برای لایه‌ی پیوند وای‌فای مهم است.

مراقب باشید: فرایند انتقال یک رابط شبکه را می‌توان به گونه‌ای پیکره‌بندی کرد که با هر آدرس MAC برای فرستنده یا گیرنده کار کند. هیچ چیز نمی‌تواند از جعل آدرس فیزیکی شما توسط یک

---

[۱] National Security Agency یا NSA یک سازمان جاسوسی دولت آمریکا است.
[۲] Promiscuous Mode
[۳] رمزگذاری به پیام‌ها اجازه می‌دهد تا برای استراق‌سمع‌کنندگان مخدوش به نظر برسند.

جاعل در ارسال‌های وی ممانعت به عمل آورد. این نوع حمله به عنوان **جعل MAC** [۱] شناخته می‌شود. هنگامی که لایه‌ی پیوند برای اولین بار توسعه یافت، امنیت یک نگرانی جدی برای توسعه‌دهندگان نبود. پروتکل‌ها برای ایمن‌تر شدن و خنثی کردن چنین حملاتی در حال تکامل هستند، اما این فرایند همواره در جریان است.

قاب‌ها

گاهی اوقات، یک انتقال باید حاوی داده‌های زیادی باشد و ارسال یک پیام تک و بزرگ غیرعملی است. رابط‌های شبکه و کامپیوترها همگی قادر به سرعت انتقال یکسان نیستند. علاوه بر این، اگر یک تصادم در وسط انتقال رخ دهد، چه اتفاقی می‌افتد؟ کل انتقال باید کنار گذاشته شود، زیرا برای فرستنده و گیرنده دشوار است که دقیقاً تعیین کنند کدام بخش از پیام دریافت شده و کدام قسمت دریافت نشده است.

برای حل این مشکلات، پیام‌های طولانی همیشه به قسمت‌های کوچک تقسیم و هر کدام به عنوان یک انتقال مستقل ارسال می‌شوند. زمان بین انتقال‌ها می‌تواند با توجه به قابلیت‌های هر دو کامپیوتر متفاوت باشد: دستگاه‌های کندتر نیاز به فواصل طولانی‌تری دارند. اگر خطایی رخ دهد، فقط باید انتقال کوچکی را که ناموفق بود دور انداخته و دوباره ارسال کرد.

هر انتقال مستقل یک **قاب** [۲] نامیده می‌شود. پروتکل‌های استاندارد وای‌فای اندازه‌ی قاب‌ها را به ۲۳۴۶ بایت محدود می‌کنند. سی‌وچهار بایت برای آدرس‌های MAC و کدهای تشخیص خطا مورد نیاز است. بنابراین، یک قاب وای‌فای در نهایت می‌تواند تا ۲۳۱۲ بایت داده را حمل کند که به آن ظرفیت بار [۳] می‌گویند [۴]. در شبکه های سیمی، حداکثر اندازه‌ی قاب معمولاً ۱۵۲۶ بایت است و ظرفیت باری معادل ۱۵۰۰ بایت وجود دارد.

در برخی موارد نادر، اختلالات رسانه با انتقال تداخل می‌کنند و گیرنده سیگنال‌هایی را دریافت می‌کند که دقیقاً شامل اطلاعات کدگذاری شده‌ی مورد نظر فرستنده نیستند. بیایید فیلد خاصی را که برای رفع این مشکلی اضافه شده است، با هم بررسی کنیم.

---

[۱] MAC Spoofing

[۲] Frame

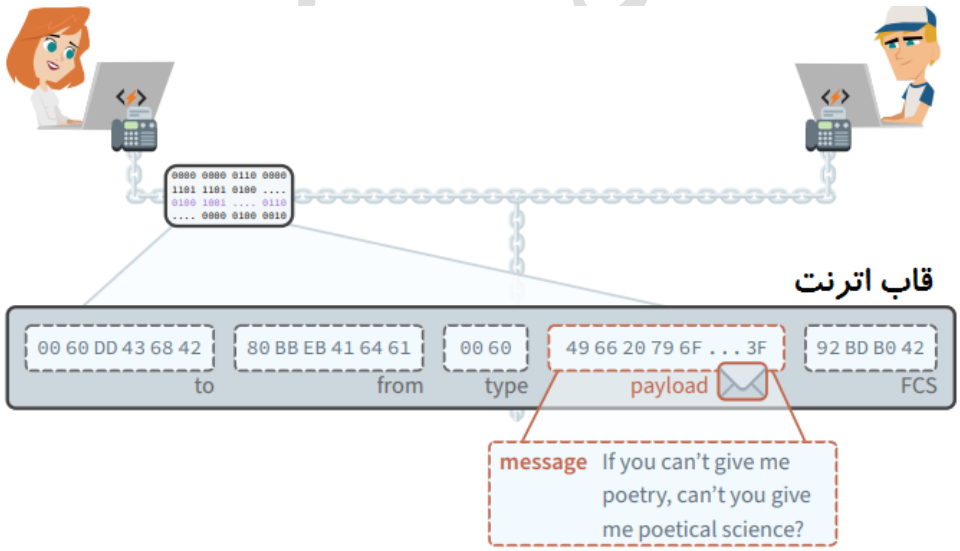
[۳] Payload

[۴] اگر به ازای هر کاراکتر یک بایت کدگذاری کنیم، یک قاب وای‌فای حدود ۵۰۰ کلمه جا دارد که برای پر کردن یک صفحه‌ی متن کافی است.

دنباله‌ی بررسی قاب: آخرین قسمت قاب **دنباله‌ی بررسی قاب**[۱] یا **FCS** است و از انتقال دقیق اطلاعات اطمینان حاصل می‌کند. در حقیقت، FCS اطلاعات جدیدی را به انتقال اضافه نمی‌کند: این بخش فقط نتیجه‌ی یک محاسبه با استفاده از محتویات تمام فیلدهای دیگر است. تغییر هر محتوای قبل از FCS باید باعث شود که شماره FCS نیز تغییر کند.

یک کامپیوتر به محض دریافت یک قاب، عدد FCS مورد انتظار را از اطلاعات دریافتی محاسبه کرده و آن را با FCS دریافتی مقایسه می‌کند. اگر مطابقت نداشته باشند، قاب کنار گذاشته می‌شود. اگر مطابقت داشته باشند، می‌دانیم که پیام مخدوش نشده است و اطمینان داریم که بار دریافتی بدون خطا است.

نوع: قاب نشان داده شده در شکل ۱–۷ فیلدی دارد که در مورد آن صحبت نکرده‌ایم: **نوع** بار[۲]. این فیلد به گیرنده می‌گوید که برای تفسیر داده‌های بار قاب باید از چه قوانینی پیروی کرد. در بخش بعدی، متداول‌ترین مجموعه از این قوانین را بررسی خواهیم کرد.



شکل ۱–۷: یک قاب اترنت هنگامی که در یک سیم مسی منتقل می‌شود، به یک سری سیگنال‌های الکتریکی تبدیل می‌شود که یک عدد را کدگذاری می‌کنند. پروتکل اترنت نحوه‌ی تفسیر این عدد را مشخص می‌کند. به عنوان مثال، ۱۲ رقم اول عدد، آدرس MAC مقصد را کدگذاری می‌کنند.

---

[۱] Frame Check Sequence
[۲] Payload Type

۲-۱- اینترنت

دیدیم که لایه‌ی پیوند کامپیوترهای متصل مستقیم را قادر می‌سازد تا پیام‌ها را درون قاب‌ها تبادل کنند. **لایه‌ی اینترنت**[۱] که با نام **لایه‌ی شبکه**[۲] نیز شناخته می‌شود، نحوه‌ی انتقال این پیام‌ها را بین کامپیوترهایی که مستقیماً به یکدیگر متصل *نیستند* مشخص می‌کند.

روال کار این گونه است که برخی از کامپیوترها، به نام **مسیریاب**[۳]، را به چندین رابط شبکه مجهز کنید. سپس تمام کامپیوترهای موجود در یک شبکه حداقل به یک مسیریاب و همه‌ی مسیریاب‌ها حداقل به یک مسیریاب دیگر متصل می‌شوند. هنگامی که مسیریاب پیامی را در یکی از رابط‌های شبکه‌ی خود دریافت می‌کند، می‌تواند آن را از طریق یک رابط شبکه‌ی دیگر به مسیریاب دیگری ارسال کند.

شبکه‌های محلی: می‌توانیم از مسیریابی که با آن مرتبط هستیم بخواهیم پیامی را به کامپیوتری که با آن مرتبط نیستیم ارسال کند. فرض کنید یک شبکه‌ی سیمی در خانه خود دارید که یک مسیریاب و یک کامپیوتر رومیزی را به هم متصل می‌کند. فرض کنید مسیریاب نیز مستقیماً به یک تلفن هوشمند در یک شبکه‌ی بی‌سیم متفاوت متصل است.

حتی اگر کامپیوتر رومیزی و تلفن هوشمند مستقیماً به یک شبکه متصل نباشند، می‌توانند با استفاده از مسیریاب به عنوان واسطه، پیام‌هایی را برای یکدیگر ارسال کنند. کامپیوترهایی از شبکه‌های مختلف در محدوده‌ی نزدیک که می‌توانند از طریق مسیریاب‌ها با یکدیگر صحبت کنند، شبکه‌ی بزرگ‌تری را تشکیل می‌دهند که به آن شبکه‌ی محلی[۴] یا LAN می‌گویند.

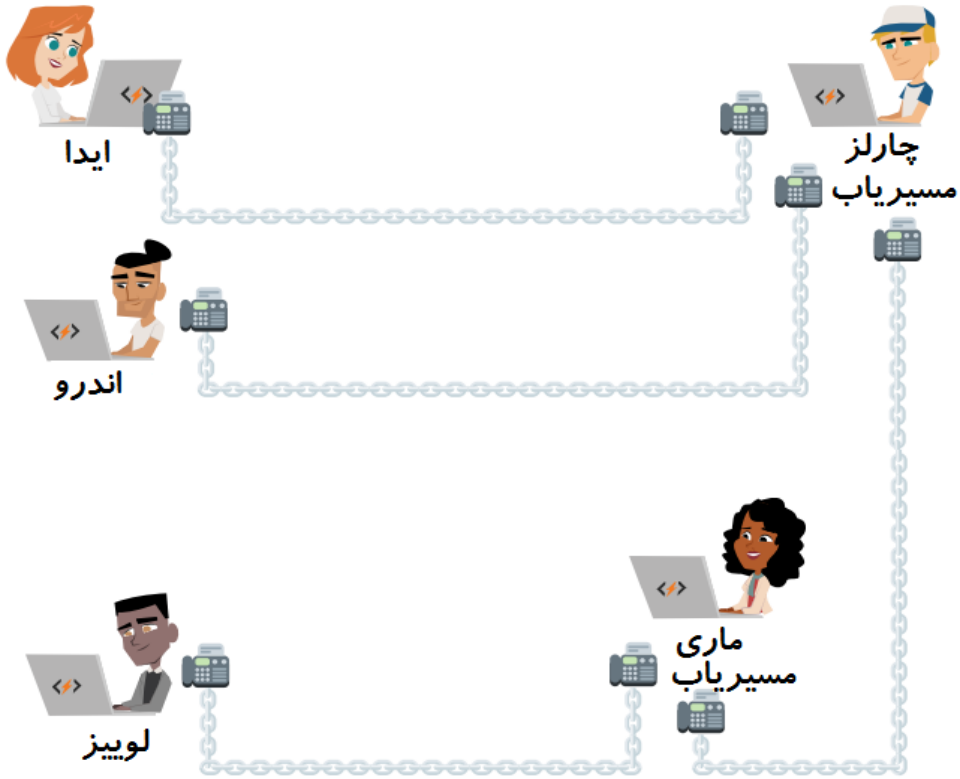
در یک خانه یا دفتر کوچک، یک مسیریاب برای اتصال تمام شبکه‌های کامپیوتری در آن محدوده کافی است. در هنگام آماده‌سازی یک LAN که یک سازمان بزرگ مانند یک دانشگاه یا بیمارستان را پوشش می‌دهد، ممکن است نیاز به مسیریاب‌های زیادی باشد تا همه‌ی شبکه‌های کامپیوتری مختلف را کاملاً به یکدیگر متصل کنند.

شبکه‌های گسترده: اما چرا در باید این‌جا توقف کرد؟ اگر مسیریاب شما با یک مسیریاب در خارج از خانه مرتبط است، که آن نیز به نوبه‌ی خود با یک مسیریاب در دانشگاه مرتبط است، می‌توانید درخواست کنید که پیام شما به کامپیوترهای موجود در شبکه‌ی LAN دانشگاه ارسال شود. وقتی که شبکه‌های محلی دور به یکدیگر متصل می‌شوند، یک شبکه‌ی گسترده[۵] یا WAN را تشکیل می‌دهند.

---

۱ Internet Layer
۲ Network Layer
۳ Router
۴ Local Area Network یا LAN
۵ Wide Area Network یا WAN

شکل ۱–۸: در این شبکه‌ی محلی کوچک، ایدا و اندرو می‌توانند از طریق مسیریاب یعنی چارلز با هم به تبادل پیام بپردازند.



شکل ۱–۹: چارلز به یک مسیریاب راه‌دور یعنی ماری متصل است و هر دوی آن‌ها پیام‌ها را در شبکه‌ی گسترده جابجا می‌کنند.

یک شبکه‌ی گسترده می‌تواند با اتصال شبکه‌های محلی بیشتر به آن، بزرگ‌تر شود. شبکه‌های WAN مختلف نیز می‌توانند به یکدیگر متصل شوند تا یک WAN بسیار بزرگتر را تشکیل دهند. بزرگترین شبکه‌ی WAN در جهان مجموعه‌ای از هزاران شبکه‌ی به هم متصل است که ما آن را **اینترنت**[۱] می‌نامیم. این همان شبکه‌ای است که ما هر روز برای ارسال ایمیل و مرور وب از آن استفاده می‌کنیم؛ و تا سال ۲۰۲۰، بیش از یک میلیارد کامپیوتر داشت. بیایید ببینیم همه‌ی آن‌ها چگونه به هم متصل شده‌اند.

ارتباط متقابل

ساده‌ترین راه برای اتصال مسیریاب خود به اینترنت، پرداخت هزینه‌ی آن است. برخی از سازمان‌های موجود در اینترنت یکی از مسیریاب‌های خود را به مسیریاب شما پیوند داده و به پیام‌های ورودی و خروجی شبکه‌ی شما اجازه می‌دهند به کمک این پیوند از طریق شبکه‌ی آنها عبور کنند. این سرویس پولی **ترانزیت**[۲] نامیده می‌شود، زیرا همه‌ی پیام‌های شما قبل از رسیدن به مسیریاب خاص مورد نظرتان، از طریق شبکه‌ی آن‌ها جابجا می‌شوند.

با این حال، ترانزیت از طریق یک شبکه‌ی شخص ثالث همیشه برای اتصال به مسیریاب دیگری از اینترنت ضروری نیست. مثلاً فرض کنید که دو دانشگاه مجاور بسیار با هم ارتباط برقرار می‌کنند. آن‌ها می‌توانند مسیریاب‌های خود را به منظور انتقال مستقیم پیام‌ها بین شبکه‌هایشان پیوند دهند. این کار می‌تواند در هزینه صرفه‌جویی کند، زیرا در غیر این صورت این پیام‌ها باید از طریق یک اتصال پولی منتقل شوند. به تبادل رایگان پیام‌ها بین شبکه‌های سازمان‌های مختلف، **همتاسازی**[۳] می‌گویند.

مسیریابی

هر کامپیوتری که به یک مسیریاب اینترنت متصل است می‌تواند درخواست کند که پیام‌هایش توسط مسیریاب‌ها دیگر ارسال شوند. پیام‌ها را می‌توان در مسافت‌های زیاد هدایت کرد. به عنوان مثال، سیستمی از کابل‌های زیردریایی وجود دارد که مسیریاب‌ها را در بسیاری از شهرهای ساحلی به هم وصل می‌کند (شکل ۱-۱۰).

هیچ پیوند مستقیمی بین مسیریاب‌ها میامی و بوئنوس‌آیرس وجود ندارد. با این حال، میامی با پورتوریکو مرتبط است، که آن نیز با فورتالزا مرتبط است، که با ریودوژانیرو و در نهایت با بوئنوس‌آیرس پیوند خورده است. اگر مسیریاب‌ها در طول مسیر پیام‌ها را جابجا کنند، میامی و بوئنوس‌آیرس می‌توانند

---

[۱] **Inter**connected **Net**works یا Internet

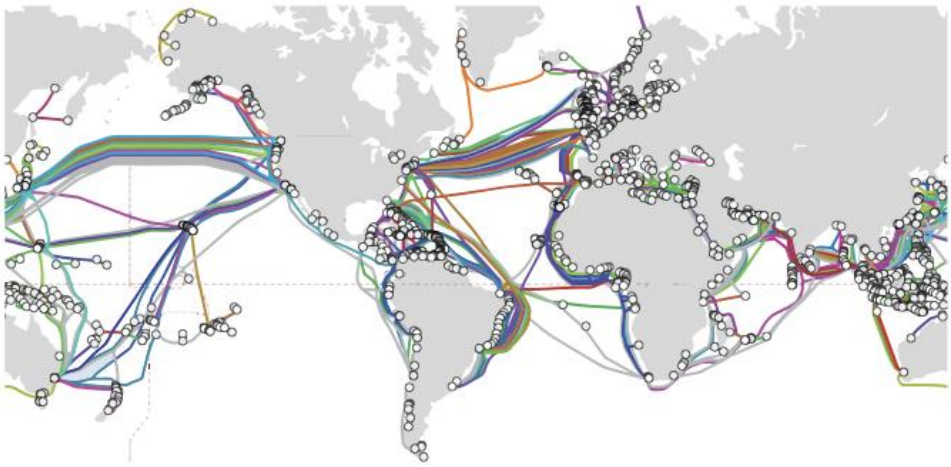
[۲] Transit

[۳] Peering

پیام‌ها را از طریق این کابل‌ها مبادله کنند. امروزه کابل‌های زیردریایی صدها مسیریاب شهرهای ساحلی را در سراسر جهان به هم متصل می‌کنند (شکل ۱-۱۱).

تقریباً هر شهر دیگری روی زمین به طور مستقیم یا غیرمستقیم به این شهرهای ساحلی و اغلب از طریق کابل‌های موجود در زمین مرتبط است. همچنین ماهواره‌های ارتباطی دارای مسیریاب‌هایی برای ایجاد پیوندهای بی‌سیم به مکان‌های دوردست هستند. همه‌ی مسیریاب‌ها می‌توانند پیام ها را جابجا کنند، بنابراین پیامی که در اینترنت ارسال می‌کنید می‌تواند به هر کامپیوتر دیگری در اینترنت هدایت شود. البته اگر راهی برای آن پیدا شود.

شکل ۱-۱۰: سیستم Sam-1 مسیریاب‌های ۱۶ شهر در ۱۱ کشور مختلف را با استفاده از بیش از ۱۵ هزار مایل کابل زیردریایی به هم متصل می‌کند.



شکل ۱-۱۱: کابل‌های زیردریایی فیبر نوری که در هم اکنون در حال استفاده هستند.

آدرس‌دهی مکان

در لایه‌ی پیوند، کامپیوترها با یک آدرس فیزیکی شناسایی می‌شوند. آدرس‌های فیزیکی کامپیوترها را به‌طور منحصربه‌فردی شناسایی می‌کنند، اما هیچ اشاره‌ای درباره‌ی مکان اتصال کامپیوتر و نحوه‌ی دسترسی به آن نمی‌دهند. اگر کامپیوتر به آن سوی دنیا برود، آدرس فیزیکی خود را حفظ خواهد کرد!

فرض کنید بسته‌ای را از طریق پست به همراه عکس لوییز به جای آدرسش برای او ارسال کرده‌اید. این بسته دارای یک مقصد مشخص است. با این حال، یک سرویس پستی بین‌المللی هیچ راهی برای دانستن اینکه بسته باید به کدام سمت ارسال شود تا آن را به لوییز تحویل دهد، ندارد.

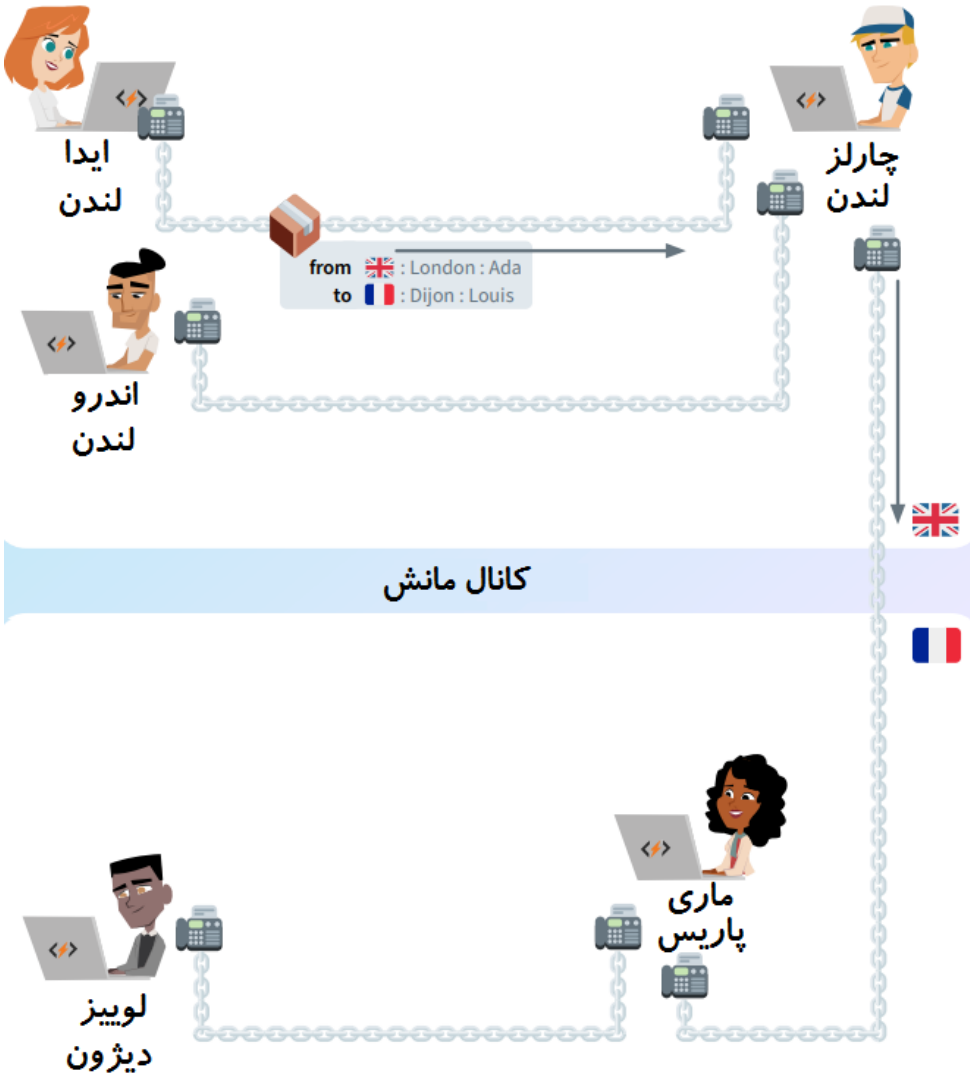
دفاتر پست ابتدا باید بدانند که بسته باید به کدام کشور برود. پس از آن اولین اداره پست در آن کشور باید بداند که به کدام استان یا ایالت مراجعه کند. اداره‌ی پست بعدی باید شهر را بشناسد و پست آخر آدرس خیابان را. آدرسی که تمام این اطلاعات را در خود دارد **آدرس سلسله‌مراتبی**[۱] می‌نامیم. شبیه به دفاتر پست، مسیریاب‌ها به بسته‌هایی نیاز دارند تا آدرس سلسله‌مراتبی محل گیرنده خود را داشته باشند (شکل ۱-۱۲).

برای اینکه این مکانیزم در مقیاس جهانی کار کند، همه کامپیوترهای درگیر باید از مجموعه قوانین یکسانی برای ایجاد و رسیدگی به درخواست‌های ارسال بسته‌ها پیروی کنند. یک کامپیوتر در چین باید

---

[۱] Heirarchical Address

درخواست یک کامپیوتر در نیجریه را درک کند، حتی اگر این دو کامپیوتر از زبان‌ها، سیستم‌عامل‌ها و سخت‌افزارهای متفاوتی استفاده کنند.



شکل ۱-۱۲: ایدا می‌خواهد بسته‌ای را برای لوییز بفرستد، بنابراین از مسیریابش یعنی چارلز می‌خواهد آن را جابجا کند. او روی بسته یک آدرس سلسله‌مراتبی از لوییز می نویسد. حال چارلز می‌داند که باید بسته را به فرانسه بفرستد، بنابراین آن را به مسیریاب فرانسوی که با آن مرتبط است می‌فرستد: یعنی ماری.

شکل ۱-۱۳: «قبل از اینترنت» دریافت شده از http://xkcd.com.

پروتکل اینترنت

دیدیم که یک کامپیوتر برای برقرای پیوند با کامپیوتر دیگر، باید از قوانین کنترل دسترسی رسانه پیروی کند. به طور مشابه، باید از **پروتکل اینترنت**[۱] یا IP [۲] نیز پیروی کند تا از مسیریاب‌ها بخواهد پیام‌ها را به کامپیوترهای دیگر در شبکه‌ی محلی شما یا اینترنت ارسال کنند.

درخواست ارسال پیام که از قوانین IP پیروی می‌کند **بسته‌ی IP**[۳] نامیده می‌شود. بسته‌ی IP اساساً یک عدد بزرگ است که در آن ارقام در موقعیت‌های خاص اطلاعات کلیدی را کدگذاری می‌کنند. تقریباً هر کامپیوتری که در چند دهه‌ی گذشته تولید شده است، بسته‌های IP را درک می‌کند و می‌تواند آن‌ها را ارسال کند. این امر باعث می‌شود بسته‌ی IP تا زمان رسیدن به مقصد، به راحتی از یک کامپیوتر به کامپیوتر دیگر منتقل شود.

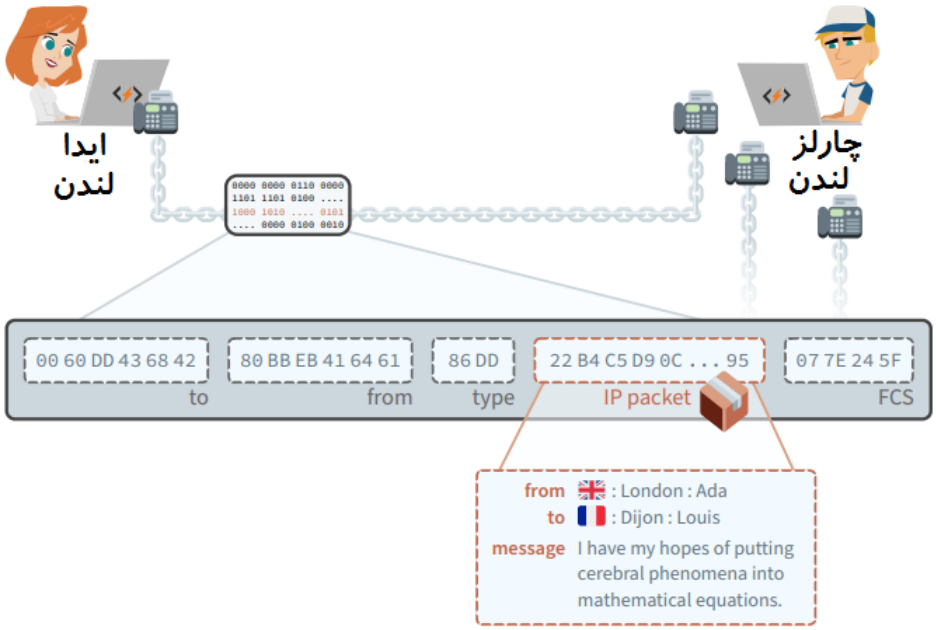
یک بسته‌ی IP حاوی آدرس‌های مکان فرستنده و گیرنده‌ی خود و به دنبال آن هر داده‌ای است که آن‌ها می‌خواهند تبادل کنند. برای ارسال یک بسته‌ی IP، قابی را ارسال می‌کنیم که بار آن شامل بسته‌ی IP و نوع قاب آن 86DD است. هنگامی که یک مسیریاب قابی را از این نوع دریافت می‌کند، بسته‌ی IP در قاب دیگری به کامپیوتر بعدی در مسیر مقصد بسته مجدداً ارسال می‌شود.

---

[۱] Internet Protocol

[۲] منظور از IP آخرین نسخه‌ی آن یعنی IPv6 است. یک نسخه‌ی قدیمی از پروتکل، IPv4، علی‌رغم انتشار در سال ۱۹۸۱، هنوز مورد استفاده قرار می‌گیرد. نسخه‌ی IPv4 تنها می‌تواند حدود ۳ میلیارد کامپیوتر را پشتیبانی کند. نسخه‌ی IPv6 که در سال ۲۰۱۲ راه اندازی شد، می‌تواند تقریباً تعداد نامحدودی از کامپیوترها را پشتیبانی کند. در سال ۲۰۲۰، یک سوم کامپیوترهای اینترنت از IPv6 استفاده می‌کنند.

[۳] IP Packet



شکل ۱-۱۴: ایدا یک قاب اترنت به مسیریاب خود یعنی چارلز می‌فرستد که حاوی یک بسته‌ی IP برای لوییز است. بنابراین قاب اترنت حاوی آدرس فیزیکی چارلز و بسته حاوی آدرس مکان لوییز است. سپس چارلز بسته را در یک قاب جدید از طرف خودش که حاوی آدرس فیزیکی شخصی در فرانسه است، ارسال می‌کند.

برای اینکه بسته‌های IP در سرتسر جهان ارسال شوند، همه باید بر روی استانداردی برای آدرس‌دهی مکان توافق کنند. دیدیم که چگونه آدرس‌های فیزیکی طبق قوانین کنترل دسترسی رسانه توسط سازندگان تخصیص داده می‌شوند. اکنون باید بیاموزیم که چگونه پروتکل اینترنت این کار را برای آدرس‌های مکان انجام می‌دهد. سپس خواهیم دید که چگونه پروتکل اینترنت قوانین مسیریابی را بر اساس این آدرس‌ها تعریف می‌کند.

۳-۱- آدرس‌دهی IP

پروتکل اینترنت قوانینی را در مورد نحوه‌ی عملکرد آدرس‌های مکان تعیین می‌کند، لذا به آن‌ها **آدرس IP** می‌گویند. کامپیوترها فقط می‌توانند بسته‌های IP را پس از دریافت آدرس IP ارسال یا دریافت کنند. اجازه‌ی استفاده از گروهی از آدرس‌های IP ابتدا به یک سازمان داده می‌شود. سپس این آدرس‌ها به کامپیوترهایی که به طور مستقیم یا غیرمستقیم با سازمان مرتبط هستند، اختصاص داده می‌شود.

برای توضیح نحوه‌ی عملکرد این فرآیند، اجازه دهید مشخص کنیم که آدرس IP چیست و چگونه نوشته می‌شود[۱]. یک آدرس IP یک عدد ۱۲۸ بیتی است[۲]. این آدرس IP سرور فیس‌بوک است:

```
2a03:2880:f003:0c07:face:b00c:0000:0002
```

آدرس‌های IP را می‌توان با حذف صفرهای ابتدایی هر بلوک چهار رقمی کوتاه کرد:

```
2a03:2880:f003:c07:face:b00c::2
```

مانند یک آدرس پستی با کشور، شهر و خیابان، آدرس‌های IP سلسله مراتبی هستند تا مسیریابی امکان‌پذیر باشد. در حالی که وسیع‌ترین بخش یک آدرس پستی کشور است، وسیع‌ترین بخش یک آدرس IP **پیشوند مسیریابی**[۳] است:

```
2a03:2880:f003:c07:face:b00c::2
```
پیشوند مسیریابی

پیشوند به عنوان اولین ارقام یک آدرس IP نشان داده می‌شود. هنگامی که به یک سازمان چنین پیشوندی داده می‌شود، این حق را دارد که هر آدرس IP را که با آن پیشوند شروع می‌شود به کامپیوترهای خود اختصاص دهد. این پیشوند دارای طول متغیر است: به سازمان‌هایی که کامپیوترهای بیشتری برای مدیریت دارند، پیشوندهای کوتاه‌تری داده می‌شود. حتی به برخی سازمان‌ها پیشوندهای متعددی داده می‌شود.

به عنوان مثال، می‌دانیم که تمام آدرس‌هایی که با پیشوند `2a03:2880` شروع می‌شوند به کامپیوترهای داخل شبکه‌ی فیس‌بوک اختصاص داده می‌شوند. آن‌هایی که با `2c0f:fb50:4002` شروع می‌شوند در شبکه‌ی گوگل در کنیا هستند. گوگل برای مرکز داده‌ی خود در سنگاپور، پیشوند `2404:6800` را دریافت کرده است.

برای اهداف مسیریابی، شبکه‌های محلی و گسترده‌ای که پیشوند یکسانی دارند در شبکه‌های کوچکی به نام **زیرشبکه**[۴] سازماندهی می‌شوند. ارقام بعد از پیشوند مسیریابی و تا وسط یک آدرس IP نشان می‌دهند که یک کامپیوتر در کدام زیرشبکه می‌تواند پیدا شود.

---

[۱] ما آدرس‌های IP را همانطور که در آخرین نسخه‌ی IP تعریف شده است ارائه خواهیم داد. آدرس‌های IPv4 قدیمی هنوز استفاده می‌شوند. آن‌ها به صورت چهار گروه از اعداد اعشاری تا سه رقمی نوشته می‌شوند که با نقطه از هم جدا شده‌اند، به عنوان مثال `192.168.0.1`.

[۲] برای نوشتن یک عدد به ۱۲۸ صفر و یک نیاز است. این بدان معنی است که عددی بین ۰ و ۳۴۰، ۲۸۲، ۳۶۶، ۹۲۰، ۹۳۸، ۴۶۳، ۴۶۳، ۳۷۴، ۶۰۷، ۴۳۱، ۷۶۸، ۲۱۱، ۴۵۶ است.

[۳] Routing Prefix

[۴] Subnet

2a03:2880:**f003:c07**:face:b00c::2
زیرشبکه

این بدان معناست که شبکه‌ای در فیسبوک وجود دارد که در آن همه کامپیوتر آدرس‌های IP دارند که با 2a03:2880:f003:c07 شروع می‌شوند. پیشوند مسیریابی و زیرشبکه در کنار هم **شناسه‌ی شبکه**[۱] یک آدرس IP را تشکیل می‌دهند. شناسه‌ی شبکه همیشه ۱۶ رقمی (شامل صفرهای حذف شده) است. این بدان معناست که سازمانی با پیشوند مسیریابی طولانی‌تر می‌تواند زیرشبکه‌های کمتری در خود داشته باشد.

در نهایت، ۱۶ رقم بعدی یک آدرس IP، **شناسه‌ی رابط**[۲] نامیده می‌شوند، زیرا آن‌ها یک رابط خاص شبکه را در یک زیرشبکه مشخص می‌کنند. بسیاری از مدیران شبکه به سادگی این قسمت از آدرس IP را با آدرس MAC دستگاه پر می‌کنند. این ارقام تا زمانی که فقط یک بار در هر زیرشبکه استفاده شوند، می‌توانند هر عددی باشند.

2a03:2880:**f003:c07**:face:b00c::2
شناسه‌ی شبکه شناسه‌ی رابط

برای اینکه این سیستم آدرس‌دهی به صورت جهانی کار کند، باید مکانیزمی وجود داشته باشد تا اطمینان حاصل شود که هیچ دو سازمانی از پیشوند مسیریابی یکسانی استفاده نمی‌کنند. همانند مورد آدرس‌های MAC، مهندسان این مشکل را از طریق برخی هماهنگی‌های بین المللی حل کرده‌اند.

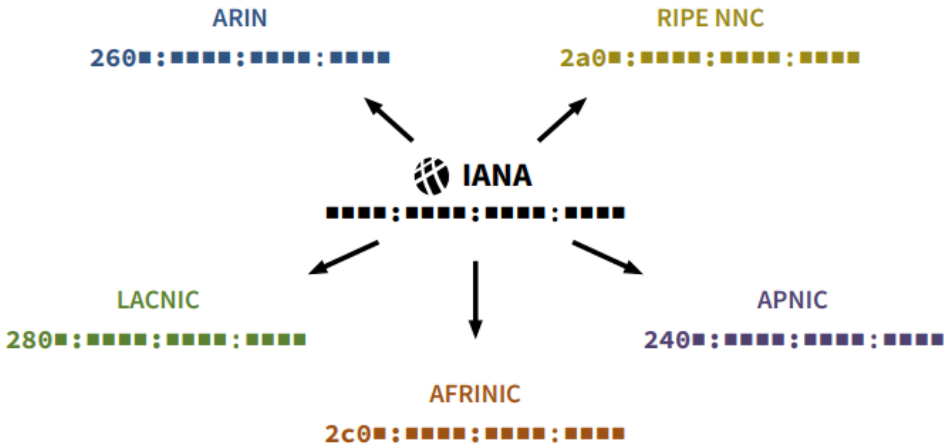
شکل ۱–۱۵: هیچ وقت آدرس محل زندگی رئیستان را نپرسید.

---

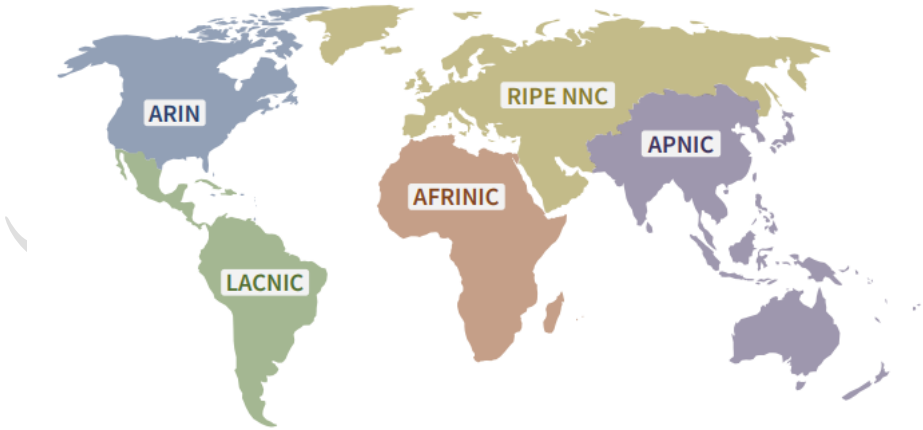
[۱] Network ID
[۲] Interface ID

IANA

مهندسان در سراسر جهان توافق کرده‌اند که یک سازمان غیرانتفاعی آمریکایی، IANA[۱]، تصمیم بگیرد که چه کسی بر پیشوندهای مسیریابی IP کنترل داشته باشد. در عمل، IANA بیشتر قدرت خود را به پنج سازمان غیرانتفاعی به نام RIR[۲] واگذار می‌کند. برای انجام این کار، به هر RIR ترکیبات هگز کوتاهی را واگذار می‌کند که آن‌ها می‌توانند به عنوان اولین ارقام پیشوندهای مسیریابی که اختصاص می‌دهند استفاده کنند.



شکل ۱-۱۶: مثال‌هایی از تخصیص شماره به هر RIR.



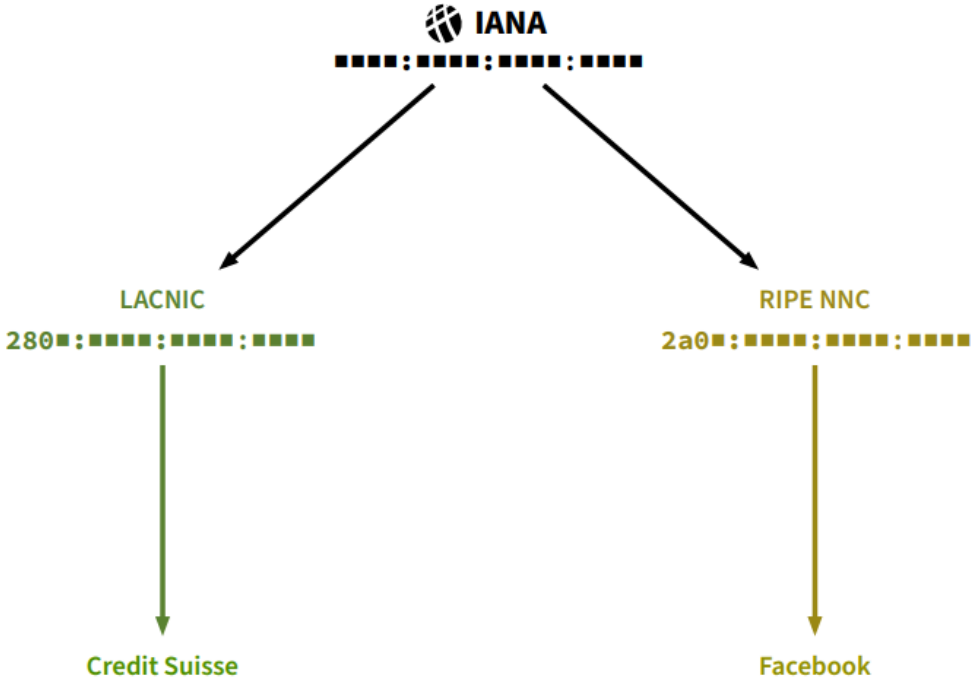
شکل ۱-۱۷: IANA فرایند آدرس‌دهی IP خود را به صورت جغرافیایی تفویض می‌کند: هر RIR مسئول منطقه‌ی متفاوتی است.

---

[۱] Internet Assigned Numbers Authority
[۲] Regional Internet Registries

برای به دست آوردن یک پیشوند مسیریابی برای سازمان خود، باید به RIR منطقه‌ای که مسیریاب‌های شما در آن قرار دارند، درخواست دهید. سپس آن RIR پیشوندی را به شما اختصاص می‌دهد که با یکی از ترکیب‌های ارقام هگز که IANA به آن اختصاص داده، شروع می‌شود.

به عنوان مثال، فیسبوک که دفتر مرکزی آن در ایرلند است، پیشوند مسیریابی خود را توسط RIPE NCC دریافت کرده است. به همین ترتیب، بانک سوئیسی Credit Suisse یک شعبه در آمریکای لاتین دارد که پیشوند مسیریابی آن توسط LACNIC تعریف شده است:



شکل ۱-۱۸: زنجیره‌ی اختصاص IP برای دو شرکت.

این بدان معناست که کامپیوترهای شعبه‌های Credit Suisse در آمریکای لاتین ممکن است آدرس‌های IP را به شرح زیر اختصاص دهند:

2801:80:1380:■■■■ :----:----:----:----

مدیران شبکه در بانک ترکیبی منحصربه‌فرد از ارقام هگزا را به هر یک از زیرشبکه‌های خود اختصاص می‌دهند به طوری که این ارقام فضای مشخص شده به صورت ■■■■ را پر می‌کنند. از آنجایی که هر رقم هگز می‌تواند ۱۶ مقدار مختلف داشته باشد، بانک فضای کافی برای ۶۵،۵۳۶ = 16^4 زیرشبکه‌ی مختلف دارد. فیس‌بوک که یک سازمان بزرگتر است، پیشوندی با فضای بیش از ۴ میلیارد زیرشبکه دریافت کرده است!

دیدیم که مدیران شبکه می‌توانند نحوه‌ی پر شدن شانزده جای خالی شناسه رابط را برای دستگاه‌های متفاوت انتخاب کنند. این دستگاه‌ها تا زمانی که مسیریاب آنها اتصال داشته باشد می‌توانند بسته‌های IP را از اینترنت دریافت یا به آن ارسال کنند.

ارائه‌دهندگان خدمات اینترنتی

بیشتر افراد و سازمان‌های کوچک مستقیماً با RIRها سروکار ندارند، و همچنین پیوندهای همتا با دیگر شبکه‌های کامپیوتری نیز ندارند. در عوض، آن‌ها اتصال اینترنت را از شرکت‌های تخصصی خریداری می‌کنند که به آنها ارائه‌دهندگان خدمات اینترنتی[۱] یا ISP می‌گویند. شرکت‌های ISP مسیریاب‌هایی را نزدیک به مشتریان خود نصب میکنند. به این ترتیب، آن‌ها می‌توانند به راحتی یکی از مسیریاب‌های خود را به یک مسیریاب در محل هر مشتری متصل کنند. آن‌ها همچنین یک پیشوند مسیریابی برای هر یک از مشتریان خود اختصاص می‌دهند.

بیایید ببینیم این فرایند در عمل چگونه کار می‌کند. در بریتانیا، یک ISP به نام Sky پیشوند مسیریابی 2a02:0c7f را دریافت کرده است. شرکت Sky در بسیاری از شهرهای بریتانیا کار می‌کند، بنابراین این پیشوند بین پایگاه‌های منطقه‌ای آن‌ها تقسیم می‌شود. به عنوان مثال، آن‌ها 2a02:c7f:48 را به شبکه‌ی میلتون کینز خود و 2a02:c7f:7e را به شبکه‌ی رامفورد اختصاص می‌دهند[۲].

فرض کنید ایدا در رامفورد زندگی می‌ند و می‌خواهد شبکه‌ای را در خانه‌اش راه‌اندازی کند. او یک کامپیوتر رومیزی و یک چاپگر دارد که می‌خواهد با استفاده از سیم اترنت به آن وصل شود. او همچنین شبکه وای‌فای خودش را می‌خواهد تا تلفن هوشمند، تبلت و لپ‌تاپ خود را به هم متصل کند.

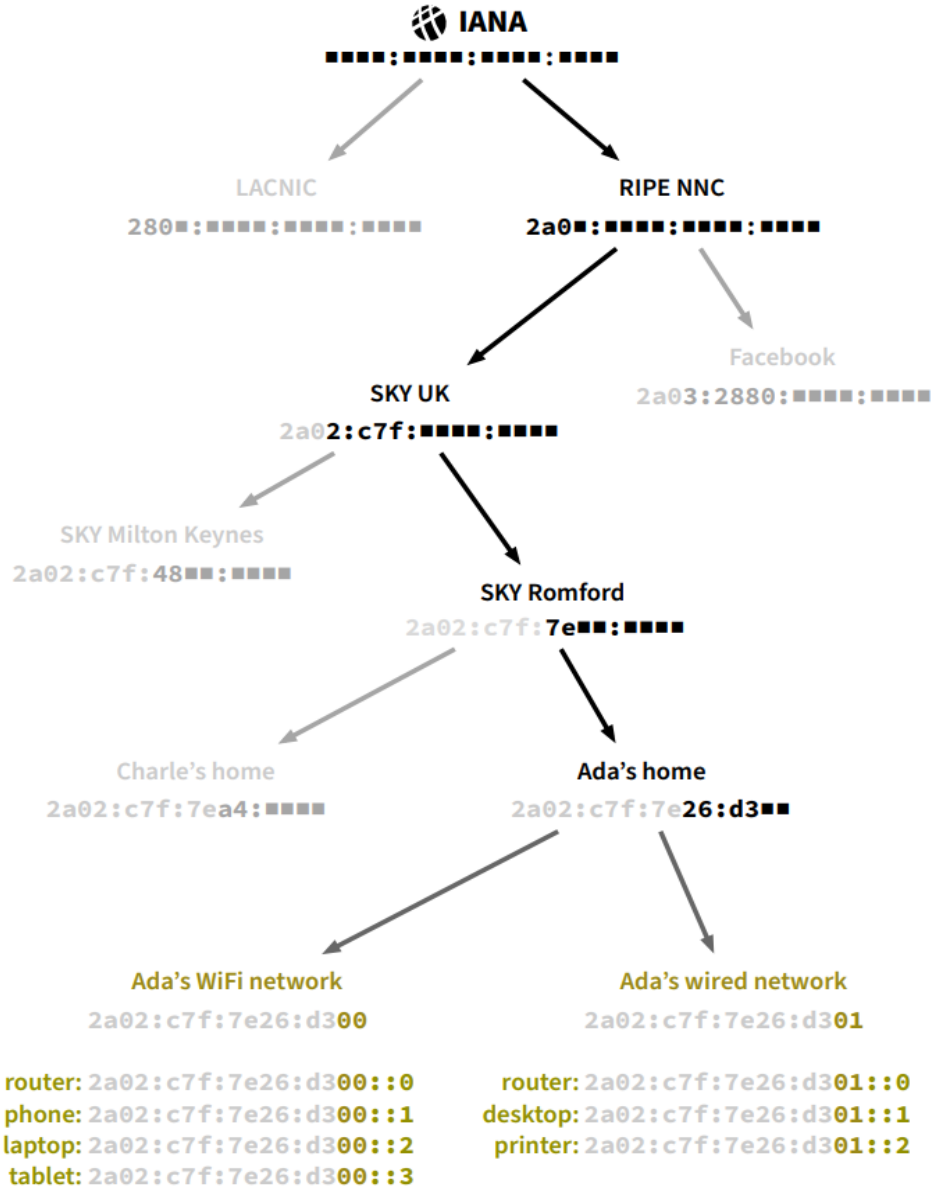
ایدا شرکت Sky را به کار می‌گیرد و آن‌ها مسیریاب رامفورد خود را به یک مسیریاب در خانه‌ی او متصل می‌کنند. شرکت Sky به مسیریاب ایدا یک پیشوند مسیریابی ۱۴ رقمی بر اساس پیشوند پایگاه رامفورد خود اختصاص می‌دهد. به هر شبکه در خانه‌ی ایدا (سیمی و بی‌سیم) یک زیرشبکه بر اساس پیشوند مسیریابی که Sky به ایدا اختصاص داده است، داده می‌شود. شکل ۱-۱۹ در صفحه‌ی بعد مسیر تخصیص آدرس IP کامل از IANA به هر یک از دستگاه های ایدا را نشان می‌دهد.

---

[۱] Internet Service Provider یا **IPS**

[۲] این اطلاعات عمومی هستند، می‌توانید موقعیت شبکه هر پیشوند مسیریابی را جستجو کنید. این روش **مکـان‌یابی IP** (یا IP Geolocation) نامیده می‌شود و به این صورت است که وب‌سایت‌ها کشور و شهری را که از آن به اینترنت وصل شده‌اید حدس می‌زنند.

مسیریاب ایدا بسته‌های IP را از چندین کامپیوتر مختلف دریافت می‌کند، با این حال برای مسیریاب او آسان است تصمیم بگیرد که از کدام پیوند برای ارسال هر بسته‌ی دریافت شده، استفاده کند. بسته‌های متعلق به کامپیوتر در یکی از زیرشبکه‌های ایدا می‌توانند مستقیماً تحویل داده شوند. تمام بسته‌های IP دیگری که مسیریاب دریافت می‌کند از طریق پیوند به ISP ارسال می‌شوند.

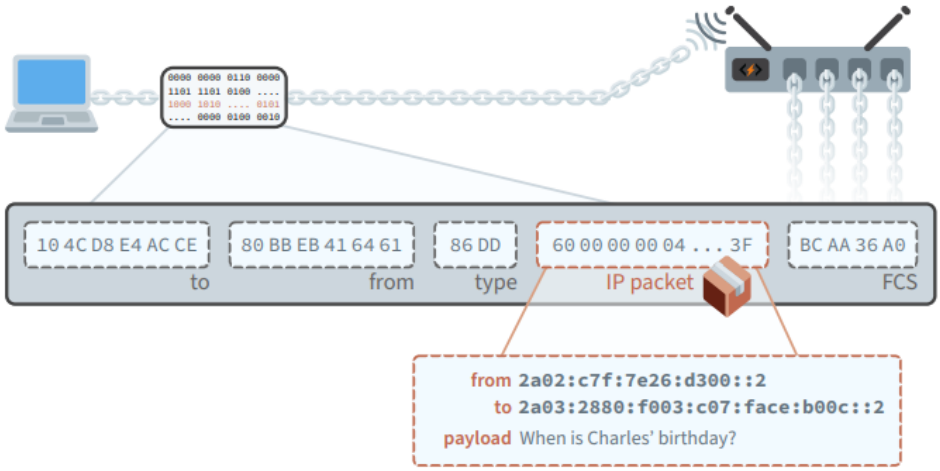


شکل ۱-۱۹: تخصیص آدرس IP از IANA به دستگاه‌های ایدا. مسیریاب او از زیرشبکه‌های مختلفی برای شبکه‌های بی‌سیم و سیمی خود استفاده می‌کند و بنابراین آدرس IP متفاوتی برای هر کدام دارد.

برای مسیریاب‌هایی که به یک ISP متکی نیستند، این کار چندان آسان نیست: آن‌ها اتصال را از پیوندهایی با چندین مسیریاب از چندین شبکه کامپیوتری به دست می‌آورند. اما چگونه تصمیم می‌گیرند که یک بسته‌ی IP از طریق کدام پیوند باید ارسال کنند؟ و حتی پس از آن، چگونه می‌توانند مطمئن شوند که آن را به مسیریاب نزدیکتر به مقصد نهایی خود ارسال می‌کنند؟

۴-۱- مسیریابی IP

فرض کنید ایدا می‌خواهد از لپ‌تاپ خود پیامی به فیس‌بوک بفرستد. او از پروتکل اینترنت استفاده خواهد کرد، بنابراین با ایجاد یک بسته‌ی IP که شامل آدرس IP خود، آدرس IP فیس‌بوک و پیام او به عنوان بار است، شروع می‌کند. سپس بسته را در یک قاب وی‌فای از لپ‌تاپ خود به مسیریاب خانگی خود منتقل می‌کند:



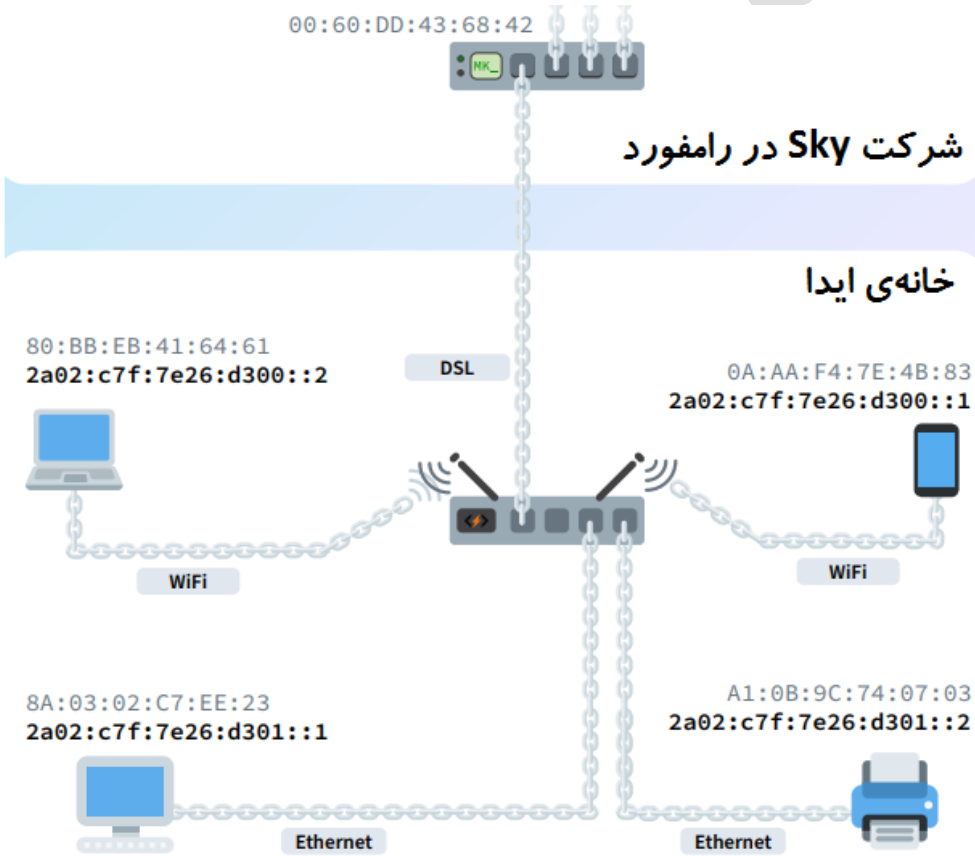
شکل ۱-۲۰: یک بسته‌ی IP منتقل شده با استفاده از وای‌فای[۱].

چندین مسیریاب، که با مسیریاب خانه‌ی ایدا شروع می‌شوند، بسته را مکرراً ارسال می‌کنند تا به فیس‌بوک برسد. در طول مسیر، هر یک از آن مسیریاب‌ها باید انتخاب کند که بسته در کدام جهت باید منتقل شود تا به مسیریاب بعدی برسد. سپس آخرین مسیریاب بسته را به سمت کامپیوتر مقصد نهایی خود ارسال می‌کند.

[۱] ما فیلدهای قاب وای‌فای را که در فریم‌های اترنت نیز وجود دارند، اضافه کرده‌ایم. یک قاب وای‌فای دارای فیلدهای بیشتری است که برای سادگی پنهان شده بودند.

جدول‌های آدرس‌ها

مسیریاب‌ها انتقال بعدی بسته را بر اساس آدرس IP مقصد انتخاب می‌کنند. برای انجام این کار، آن‌ها با یک جدول پر از آدرس‌های IP مجهز شده‌اند. ردیف‌های چنین جدولی آدرس‌های ممکنی را که مسیریاب برای شناسایی آن‌ها پیکره‌بندی شده است، فهرست می‌کنند. برای هر آدرس، جدول نشان می‌دهد که کدام کامپیوتر باید مقصد انتقال بعدی بسته‌ای باشد که به آن آدرس می‌رسد. هر مسیریاب یک جدول منحصربه‌فرد دارد که نحوه‌ی اتصال آن را نشان می‌دهد. برای مثال، در اینجا نحوه‌ی پیوند مسیریاب ایدا آمده است:



شکل ۱-۲۱: مسیریاب ایدا به یک کامپیوتر رومیزی و یک چاپگر از طریق اترنت، به یک نوت‌بوک و یک گوشی هوشمند از طریق وای‌فای و به ISP از طریق DSL (یا Digital Subscriber Line، فناوری که به داده‌های دیجیتال اجازه می‌دهد از طریق کابل‌های تلفن قدیمی جریان یابند) به متصل می‌شود.

Destination IP address	Interface	MAC address
2a02:c7f:7e26:d301::1	Ethernet	8A:03:02:C7:EE:23
2a02:c7f:7e26:d301::2	Ethernet	A1:0B:9C:74:07:03
2a02:c7f:7e26:d300::1	WiFi	0A:AA:F4:7E:4B:83
2a02:c7f:7e26:d300::2	WiFi	80:BB:EB:41:64:61
default	DSL	00:60:DD:43:68:42

شکل ۱-۲۲: جدولی که مسیریاب ایدا را برای ارسال صحیح بسته‌های IP به کامپیوترهای نشان داده شده در شکل ۱-۲۱ راهنمایی می‌کند.

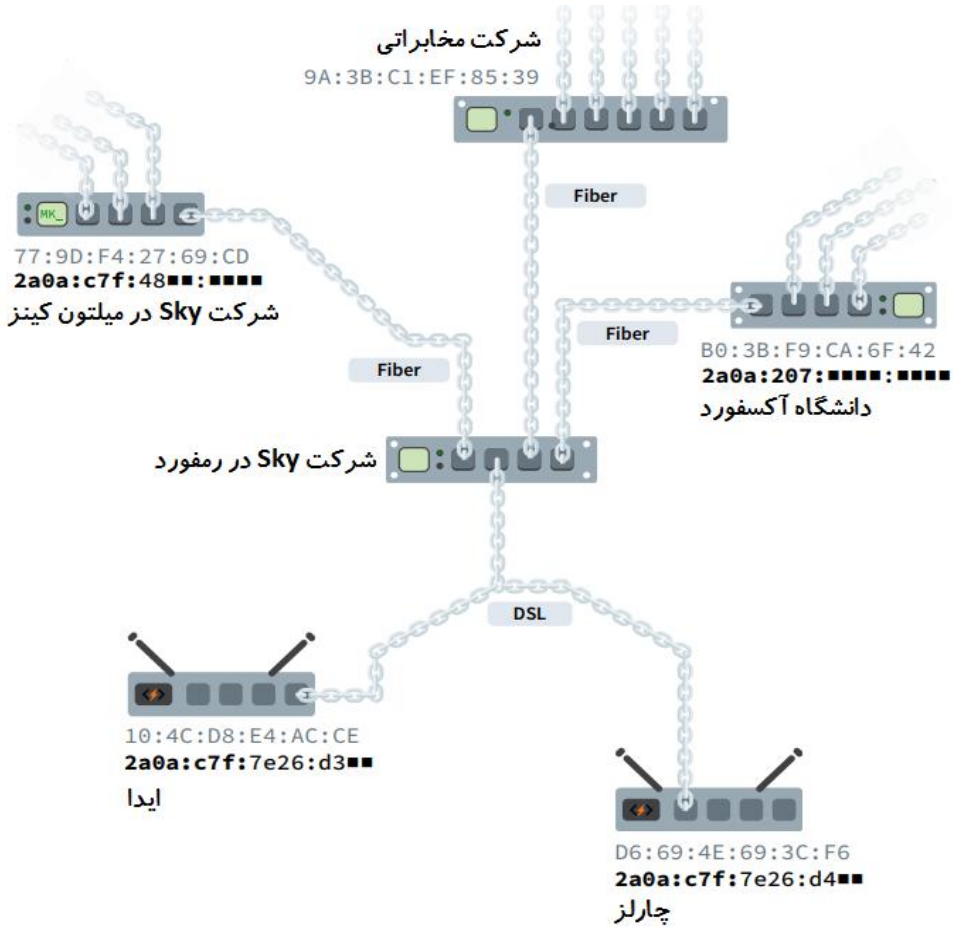
اگر مسیریاب بسته‌ای را دریافت کند که آدرس IP مقصد آن با هیچ ردیفی در جدول مطابقت نداشته باشد، بسته به **مسیر پیش‌فرض**[۱] ارسال می‌شود. برای مسیریاب ایدا مسیریابی ساده است: یک بسته یا مستقیماً به کامپیوتری در خانه‌ی او تحویل داده شده یا به شرکت Sky در رمفورد یعنی ISP او ارسال می‌شود.

مسیریابی برای مسیریاب ISP پیچیده‌تر است. این مسیریاب علاوه بر پیوندهای همتا و ترانزیت، بسته‌هایی را از مشتریان مختلف دریافت می‌کند. برای سادگی، فرض کنید مسیریاب شرکت Sky در رمفورد تنها به دو مشتری خدمات می‌دهد و دو پیوند همتا دارد: یکی به مسیریاب Sky در میلتون کینز، و دیگری به دانشگاه آکسفورد. در نهایت، بیایید تصور کنیم که یک پیوند ترانزیت با یک شرکت مخابراتی بزرگتر دارد:

اینجاست که سلسله مراتب آدرس‌دهی IP به کار می‌آید. در جدول ارسال بسته که در شکل ۱-۲۴ آمده است، آدرس‌های IP بر اساس پیشوند مسیریابی گروه‌بندی می‌شوند. این فرایند به این دلیل درست کار می‌کند که تمام آدرس‌های IP که با 2a0a:207 شروع می‌شوند از کامپیوترهای دانشگاه آکسفورد هستند و همه‌ی آدرس‌های IP که با 2a02:c7f:48 شروع می‌شوند از کامپیوترهایی هستند که از شرکت Sky در میلتون کینز سرویس می‌گیرند.

---

[۱] Default Route



شکل ۱-۲۳: نقشه‌ی پیوندهای شرکت Sky در رمفورد. ایدا، چارلز و دانشگاه آکسفورد می‌توانند فقط با استفاده از زیرساخت‌های محلی شرکت Sky با هم صحبت کنند.

Destination IP address	Interface	MAC address
2a0a:c7f:7e26:d3■■:...	DSL	10:4C:D8:E4:AC:CE
2a0a:c7f:7e26:d4■■:...	DSL	D6:69:4E:69:3C:F6
2a0a:c7f:48■■:■■■■:...	Fiber	77:9D:F4:27:69:CD
2a0a:207:■■■■:■■■■:...	Fiber	B0:3B:F9:CA:6F:42
default	Fiber	9A:3B:C1:EF:85:39

شکل ۱-۲۴: جدولی که مسیریاب شرکت Sky در رمفورد را برای ارسال صحیح بسته‌های IP به شبکه های نشان داده شده در شکل ۱-۲۳ راهنمایی می‌کند.

نقاط تبادل اینترنت

مدیران شبکه به منظور افزایش ظرفیت و سرعت، اغلب پیوندهای همتا را با بسیاری از سازمان‌های دیگر راه‌اندازی می‌کنند. ارزان‌ترین راه برای انجام این کار از طریق مکان‌هایی به نام نقاط تبادل اینترنت[۱] یا IXP است. سازمان‌ها با سیم‌کشی مسیریاب‌های خود به ساختمان IXP به یک IXP می‌پیوندند. سپس هر سازمان شرکت‌کننده می‌تواند پیوندهای همتای شخصی با سایر سازمان‌های متصل به ساختمان ایجاد کند[۲].

در شکل ۱-۲۳، فقط دو پیوند همتا به وضوح نشان داده شده‌اند. یک ISP معمولی در واقع دارای تعداد زیادی پیوند همتا به ازای هر IXP متصل شده است. علاوه بر این، در شهرهای بزرگ برای شرکت‌های اینترنتی مانند نتفلیکس و گوگل برقراری پیوند همتا با هر ISP امری معمولی است زیرا به آن‌ها امکان اتصال کوتاه‌تر و سریع‌تر به بسیاری از مشتریان خود را می‌دهند.

ستون فقرات اینترنت

شرکت‌های ISP و سایر شرکت‌های مخابراتی معمولاً با ایجاد پیوندهای همتا در هر کجا که می‌توانند، اتصالات خود را تا آنجا که ممکن است گسترش می‌دهند. با این حال، برای دسترسی به شبکه‌هایی که نمی‌توانند با آنها همتا شوند، باید از اپراتورهای دیگر ترانزیت بخرند.

تعداد انگشت‌شماری از شرکت‌ها در جهان وجود دارند که برای ترانزیت به کسی پولی پرداخت نمی‌کنند. این شرکت‌ها شبکه‌های عظیمی را اداره می‌کنند که همگی با یکدیگر همتا هستند و به شرکت‌های ISP منطقه‌ای اجازه می‌دهند در سطح جهانی به یکدیگر متصل شوند. این شبکه‌های *عظیم* **شبکه‌های سطح ۱**[۳] نامیده می‌شوند و ستون فقرات اینترنت را تشکیل می‌دهند. برخی از این شبکه‌ها توسط AT&T، Verizon و Lumen اداره می‌شوند[۴].

مسیریابی پویا

حتی اگر برخی از پیوندهای ترانزیت یا همتای آن‌ها خراب شود، شرکت‌های مخابراتی بزرگ باید اتصال خود را حفظ کنند. این بدان معنی است که آن‌ها نمی‌توانند برای هر پیشوند مسیریابی در جدول

---

[۱] Internet Exchange Points یا **IXP**

[۲] نقاط تبادل اینترنت برای اتصال خوب و ارزان بودن اینترنت بسیار مهم هستند. این ویدیو دلیل این امر را توضیح می‌دهد: http://code.energy/IXP.

[۳] Tier-1 Networks

[۴] برای درک اینکه چقدر این شبکه ها عظیم هستند باید بدانید Lumen به تنهایی ۷۵۰۰۰۰ مایل کابل فیبر نوری را مدیریت و استفاده می‌کند. این اندازه، سه برابر مسافت زمین تا ماه است!

آدرس‌های خود به یک پیوند تکیه کنند. در واقع، آن‌ها **مسیریاب‌های پویا**[۱] دارند که نحوه‌ی اتصال شبکه‌های دیگر را به منظور انتخاب مسیرهایی که در جدول‌هایشان اولویت دارند، مشخص می‌کنند.

مسیریاب‌های پویا به صورت دوره‌ای اطلاعات را با سایر مسیریاب‌های پویا که به آنها متصل هستند مبادله می‌کنند. آن‌ها به یکدیگر می‌گویند که کدام پیشوند شبکه از طریق هر یک از پیوندهای آن‌ها قابل دسترسی است. این امر به آن‌ها اجازه می‌دهد تعیین کنند که هر پیوند از هر پیشوند مسیریابی چقدر فاصله دارد و چگونه به آن متصل می‌شود. سپس مسیریاب‌های پویا می‌توانند بهترین مسیر را برای هر پیشوند بر اساس معیارهایی مانند مسافت و سرعت تعیین کنند[۲].

با این اطلاعات، مسیریاب‌های پویا جدولی می‌سازند که تمام پیشوندهای مسیریابی را پوشش می‌دهد. برای هر پیشوند، جدول نشان می دهد که در بهترین مسیر به سمت مقصد نهایی، کدام انتقال باید متعاقباً انجام شود. هنگامی که یک پیوند برقرار می‌شود یا یک پیوند از بین می‌رود، مسیریاب‌های پویا به همتایان خود اطلاع می‌دهند. با انتشار اطلاعات، همه‌ی آن‌ها جداول خود را به روز می‌کنند تا بسته‌ها را به سمت بهترین مسیرها ارسال کنند.

هیچ نهاد مرکزی برای هماهنگی تبادل این اطلاعات وجود ندارد: مسیریاب‌ها جزئیات پیوند را آزادانه و داوطلبانه با همتایان خود به اشتراک می‌گذارند. در نتیجه، مشکلات مسیریابی اغلب وجود دارند.

حلقه‌ی مسیریابی

مسیریاب‌های دارای پیکره‌بندی نادرست می‌توانند باعث ایجاد خطا شوند. مهم‌تر از همه، جدول‌های آدرس دارای خطا می‌توانند یک بسته را چند مرحله به عقب برگردانند و آن را در یک حلقه‌ی بی‌پایان عذاب‌آور گرفتار کنند (شکل ۱-۲۵).

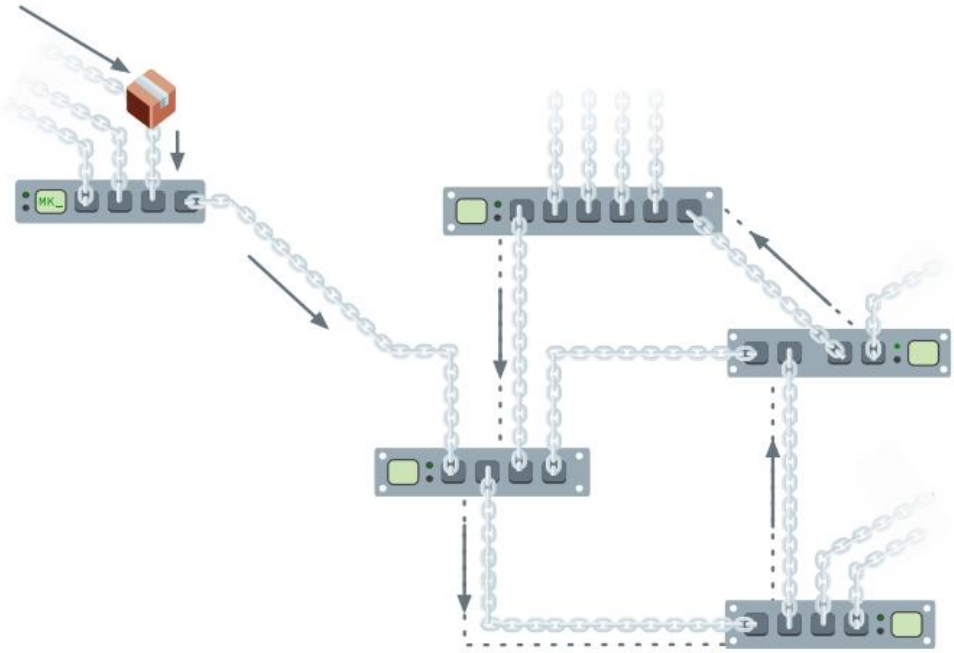
اگر جدول‌ها تصحیح نشوند، بسته‌های بیشتری با همان مقصد مورد نظر به طور بی‌پایان در حلقه‌ها ارسال می‌شوند. تعداد زیاد بسته‌ها حتی می‌تواند پیوندها را اشباع و مسدود کند. این پدیده به عنوان یک **مشکل حلقه‌ی مسیریابی**[۳] شناخته می‌شود. خوشبختانه، پروتکل اینترنت راهی برای شناسایی مشکل در هنگام بروز آن فراهم می‌کند.

---

[۱] Dynamic Router

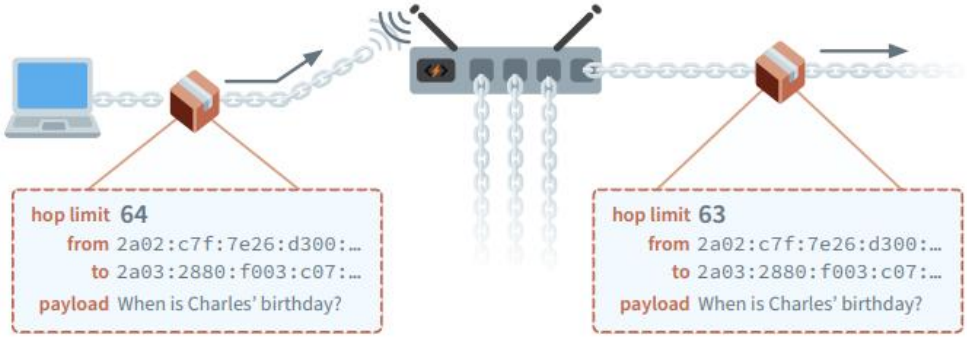
[۲] هر پنج RIR دائماً اطلاعات مربوط به تمام پیشوندهای مسیریابی را که به آنها واگذار می‌کنند، منتشر می‌کنند. مسیریاب‌های پویا به دقت این اعلامیه ها را ردیابی می‌کنند، بنابراین می‌توانند اطمینان حاصل کنند که جداول آن‌ها دارای یک ردیف برای هر پیشوند مسیریابی موجود است.

[۳] Routing Loop Problem



شکل ۱–۲۵: جدول‌های دارای خطا یک بسته را مکرراً در یک حلقه ارسال می‌کنند.

حد انتقال: برای قطع حلقه‌های مسیریابی دائمی، همه‌ی بسته‌های IP دارای یک حد انتقال[۱] بین ۰ تا ۲۵۵ هستند. این نشان‌دهنده تعداد دفعاتی است که بسته می‌تواند توسط مسیریاب‌ها ارسال شود. به طور معمول، بسته‌ها با حد انتقال ۶۴ ایجاد می‌شوند. هر زمان که مسیریاب یک بسته را ارسال می‌کند، حد انتقال را یک واحد کاهش می‌دهد:



شکل ۱–۲۶: حد انتقال بسته تنها عنصر بسته‌ی IP است که مسیریاب‌ها در حین ارسال آن را تغییر می‌دهند.

---

[۱] Hop Limit

اگر بسته‌ای به صورت حلقه‌ای در گردش باشد، حد انتقال آن در نهایت به صفر خواهد رسید. اگر مسیریاب یک بسته‌ی IP با حد انتقال صفر دریافت کند، می‌تواند آن را دور بیندازد. سپس یک بسته‌ی IP حاوی یک پیام خطا باید توسط آخرین مسیریاب به فرستنده بازگردانده شود که بیان می‌کند بسته نمی‌تواند تحویل داده شود زیرا به حد انتقال آن رسیده است.

بازخورد از طریق چنین پیام‌های خطایی به مدیران شبکه کمک می‌کند تا خطاهای مهم را برطرف کنند؛ البته حلقه‌های مسیریابی تنها مشکلات موجود نیستند. در واقع، پروتکل اینترنت نحوه‌ی مقابله با انواع مشکلات مسیریابی را پوشش می‌دهد.

خطایابی

مسیریاب‌ها بسته‌های IP را که قادر به مدیریت آن‌ها نیستند دور می‌اندازند. هنگامی که این اتفاق می‌افتد، آن‌ها یک پیام اطلاعاتی در مورد حادثه به فرستنده‌ی بسته ارسال می‌کنند. پروتکل اینترنت مشخص می‌کند که مسیریاب‌ها چگونه باید چنین پیام‌هایی را قالب‌بندی کنند تا اطمینان حاصل شود که آن‌ها برای هر کامپیوتری قابل درک هستند. این قوانین زیرمجموعه‌ای از پروتکل اینترنت به نام **پروتکل پیام کنترل اینترنت**[۱] یا ICMP هستند.

پروتکل ICMP کدهای خطا را به رایج‌ترین مشکلات مسیریابی اختصاص می‌دهد. برای گزارش یک مشکل، مسیریاب یک بسته‌ی IP حاوی کد خطا را به عنوان متن پیام ارسال می‌کند که بر اساس قوانین ICMP قالب‌بندی شده است. بیایید برخی از مشکلات رایج را که می‌توان با استفاده از ICMP گزارش کرد با هم ببینیم. با مشکل حلقه‌ی مسیریابی شروع می‌کنیم.

اتمام زمان: اگر مسیریاب یک بسته‌ی IP با حد انتقال صفر دریافت کند، زمان انتقال آن تمام شده است. بسته یا در یک حلقه‌ی مسیریابی گیر کرده یا فرستنده حد انتقال ناکافی به آن داده است. در چنین مواردی، یک پیام ICMP با کد خطای **اتمام زمان**[۲] بازگردانده می‌شود. پیام ICMP شامل اولین بایت‌های بسته‌ی حذف شده است تا به فرستنده اصلی اجازه دهد بداند کدام بسته به مقصد خود نرسیده است.

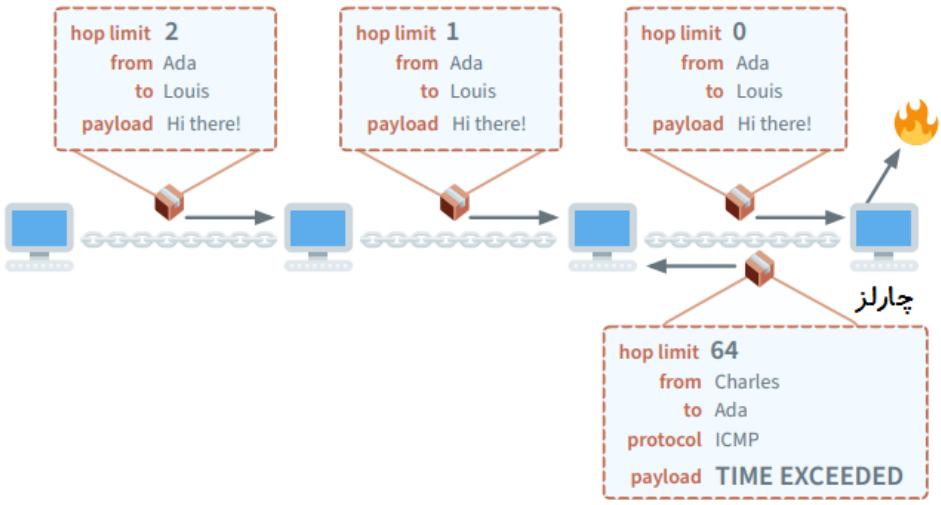
توجه داشته باشید که در شکل۱-۲۷، بسته‌ی IP که چارلز آن را برگردانده است شامل یک فیلد پروتکل است. این فیلد یک عدد هگز دو رقمی است که مشخص می‌کند بار بسته چگونه باید تفسیر شود. آخرین نسخه‌ی ICMP با دارای شماره پروتکل 0x3A است. تمام بسته‌های IP باید دارای یک شماره

---

[۱] Internet Control Message Protocol یا **ICMP**

[۲] Time Exceeded

پروتکل باشند. در بخش بعدی، در این مورد بیشتر خواهیم آموخت. در حال حاضر، بیایید سایر مشکلات رایج مسیریابی را بررسی کنیم.



شکل ۱-۲۷: هنگامی که مسیریابی بسته‌ای با حد انتقال صفر دریافت می‌کند، آن را دور انداخته و یک پیام خطای ICMP برای فرستنده‌ی بسته ارسال می‌کند.

مقصد دست‌نیافتنی: گاهی اوقات، مسیریاب جایی برای ارسال بسته ندارد. این پدیده ممکن است به دلایل مختلفی اتفاق بیفتد، برای مثال اگر آدرس IP در جدول آدرس‌های مسیریاب نباشد و جدول انتقال بعدی پیش‌فرض را نیز پیشنهاد نکند. گاهی اوقات، مقصد انتقال بعدی آفلاین است.

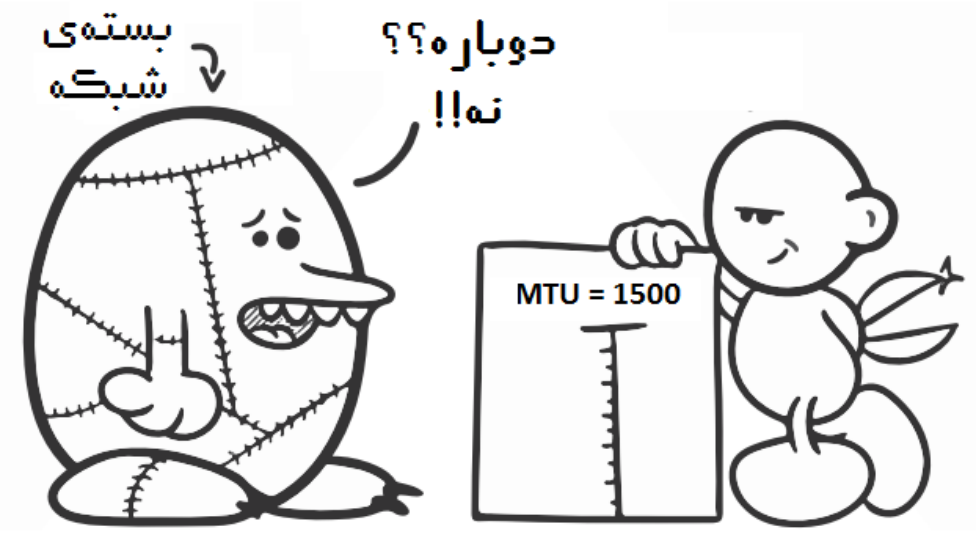
وقتی مسیریاب نمی‌داند بسته را کجا ارسال کند، یک پیام ICMP با کد خطای **مقصد دست‌نیافتنی**[۱] را همراه با اولین بایت‌های محتوای بسته‌ی حذف شده برمی‌گرداند.

بسته‌ی خیلی بزرگ: دیدیم که پروتکل‌های لایه‌ی پیوند، اندازه‌ی داده‌ای را که می‌توان در یک قاب ارسال کرد، محدود می‌کنند. قاب‌های انواع مختلف پیوندهای شبکه می‌توانند بارهایی با اندازه‌های مختلف جابجا کنند. حداکثر تعداد بایت‌های بار قابل حمل در یک قاب را **حداکثر واحد انتقال**[۲] یا MTU

[۱] Destination Unreachable
[۲] Maximum Transmission Unit یا **MTU**

می‌گویند. پروتکل‌های مختلف لایه‌ی پیوند دارای مقادیرMTU متفاوتی هستند. برای قاب‌های اترنت مقدار MTU برابر با ۱۵۰۰ بایت است. این مقدار برای قابهای وای‌فای ۲۳۰۵ بایت است.

اگر مسیریاب بسته‌ای بزرگ‌تر از آنچه که انتقال بعدی می‌تواند جابجا کند دریافت کند، نمی‌توان آن را همان‌طور که هست ارسال کرد. در عوض، مسیریاب یک پیام ICMP با کد خطای **بسته‌ی خیلی بزرگ**[۱]، همراه با اولین بایت‌های بسته‌ی مشکل‌دار وMTU انتقال بعدی را برمی‌گرداند. به این ترتیب فرستنده آگاه شده و می‌تواند قبل از تلاش مجدد، پیام اصلی را برش داده یا به بسته‌های کوچکتر تقسیم کند.



شکل ۱-۲۸: حداکثر واحد انتقال (توسط دنیل ستوری، دریافت شده از http://turnoff.us).

مشکل پارامتر: یک بسته‌یIP حاوی اطلاعات اضافی زیادی در کنار بار آن است. دیدیم که این بسته شامل آدرس‌هایIP، یک حد انتقال و یک شماره‌ی پروتکل است. همچنین این بسته شامل یک فیلد است که اندازه‌ی بار را نشان می‌دهد و فیلد دیگری که نسخه‌ی پروتکل اینترنت مربوطه را مشخص می‌کند. فیلدهای اضافی نیز برای کمک به مسیریاب‌ها در اولویت‌بندی بسته‌های مهم نیز وجود دارند.

همه‌ی این فیلدها باید طبق قوانین سختگیرانه مرتب و قالب‌بندی شوند. هنگامی که مسیریاب بسته‌ای را دریافت می‌کند که با پروتکل مطابقت ندارد، یک پیامICMP با کد خطای **مشکل پارامتر**[۲] و مکانی در

---

[۱] Packet Too Big
[۲] Paremeter Problem

بسته که تداخل پیدا شده است، برمی‌گرداند. طبق معمول، پیامICMP حاوی چند بایت از بسته‌ی دور انداخته شده برای اهداف شناسایی نیز است.

پیام‌های اطلاعاتی: گزارش‌های خطا تنها پیام‌هایی نیستند که ICMP برای بازرسی و تشخیص شبکه‌های کامپیوتری معیوب تعریف می‌کند. مهم‌تر از همه، جفت پیام اطلاعاتی **درخواست اکو**[۱] و **پاسخ اکو**[۲] به طور گسترده مورد استفاده قرار می‌گیرند. هنگامی که یک کامپیوتر یک درخواست اکوICMP دریافت می‌کند، یک بسته‌ی حاوی یک پاسخ اکو ICMP را برمی‌گرداند.

این کار برای بررسی آنلاین بودن کامپیوتر مفید است. برنامه ای به نام ping وجود دارد که یک پیام درخواست اکو ICMP ارسال و مدت زمانی را که طول می‌کشد تا پاسخ به شما برسد اندازه گیری می‌کند[۳]. علاوه بر این، با ارسال درخواست‌های اکوICMP با حد انتقال اولیه‌ی متفاوت، می‌توانید بسته‌های مسیر را تا رسیدن به مقصدشان دنبال کنید[۴].

۵-۱- انتقال

دیدیم که کامپیوترهای موجود در اینترنت می‌توانند پیام‌ها را در قالب بسته‌هایIP مبادله کنند. به عنوان مثال، آن‌ها می توانند پیام‌هایICMP را مبادله کنند. با این حال، قدرت واقعی اینترنت زمانی آشکار می‌شود که *برنامه‌ها*، نه کامپیوترها، شروع به ارسال داده‌ها به یکدیگر در قالب بسته‌هایIP کنند.

کامپیوترهای موجود در یک شبکه اغلب **میزبان**[۵] نامیده می‌شوند زیرا آن‌ها صرفاً میزبان برنامه‌هایی هستند که از شبکه استفاده می‌کنند. به عنوان مثال، یک گوشی هوشمند می‌تواند میزبان برنامه‌هایی برای پخش همزمان موسیقی، گشت‌وگذار در وب و دریافت ایمیل باشد.

به طور کلی، یک برنامه، میزبان خود را با برنامه‌های دیگر به اشتراک می‌گذارد، اما نمی‌خواهد از طریق تمام بسته‌هایIP ورودی عبور کند. برای حل این مشکل، برنامه‌ها از میزبان خود می‌خواهند که بسته ها را از طرف آن‌ها ارسال و دریافت کند. به این ترتیب، میزبان‌ها، بار هر بسته را به سمت برنامه‌ی مناسب هدایت می‌کنند و هر برنامه فقط داده‌هایی را که باید بخواند، دریافت می‌کند.

---

[۱] Echo Request

[۲] Echo Reply

[۳] شما می‌توانید بسته‌های ICMPرا اینجا ارسال کنید: http://code.energy/ping.

[۴] توضیحی را در مورد نحوه‌ی استفاده از ICMP برای ردیابی مسیرهایی که بسته‌هایIP از طریق آن‌ها عبور می کنند می‌توان در http://code.energy/traceroute یافت.

[۵] Host

این راه حل مستلزم آن است که داده‌های ارسال شده توسط یک برنامه با اطلاعات اضافی همراه باشند. **لایه‌ی انتقال**[۱] نحوه قالب‌بندی و تفسیر این اطلاعات اضافی را در بسته‌ی IP مشخص می‌کند.

پروتکل دیتاگرام کاربر

ساده‌ترین پروتکل لایه‌ی انتقال، **پروتکل دیتاگرام کاربر**[۲] یا UDP است. به برنامه‌هایی که از UDP استفاده می‌کنند توسط میزبان آن‌ها شماره پورت[۳] اختصاص می‌یابد. پیام‌ها همراه با شماره‌ی پورت برنامه‌های ارسال‌کننده و دریافت‌کننده مبادله می‌شوند.

برای برقراری ارتباط از طریق UDP، یک برنامه ابتدا یک **سوکت**[۴] ایجاد می‌کند. سوکت یک کانال ارتباطی بین برنامه و میزبان آن است. فرض کنید سرور فیس‌بوک در حال اجرای یک برنامه تقویم با استفاده از پورت UDP شماره‌ی ۱۸ است. برنامه‌ای که روی میزبان ایدا اجرا می‌شود می‌تواند پیامی را به این برنامه‌ی فرضی فیس‌بوک ارسال کند:

```
socket ← SOCKET.new(IP, UDP)
message ← "When is Charles' birthday?"
facebook_address ← 2a03:2880:f003:c07:face:b00c::2
app_port ← 18
socket.sendto(message, facebook_address, app_port)
```

هنگام ارسال این پیام، میزبان ایدا به طور خودکار یک شماره پورت استفاده نشده را به سوکت تازه ایجاد شده اختصاص می‌دهد. فرض کنید شماره پورت ۵۴۳۲۱ را انتخاب می‌کند. بسته‌ی IP که هنگام فراخوانی `sendto()` ارسال می‌شود به شکل ۱-۲۹ خواهد بود.

یک میزبان می‌تواند هزاران سوکت مختلف را برای برنامه‌های خود مدیریت کند. یک برنامه‌ی کاربردی حتی می‌تواند چندین سوکت ایجاد کند تا چندین کانال ارتباطی را به صورت موازی استفاده کند. هر سوکت شماره پورت خود را از میزبان خود دریافت می‌کند.

برنامه‌ی تقویم فرضی ما را در میزبان فیس‌بوک در نظر بگیرید. پس از دریافت تماس اولیه‌ی ایدا، یک دیتاگرام با پورت مقصد ۵۴۳۲۱ در بسته‌ای برای آدرس IP ایدا برمی‌گرداند. برنامه‌ی ایدا با فراخوانی یک متد سوکت دیگر منتظر این دیتاگرام می‌ماند:

```
received ← socket.recv()
```

---

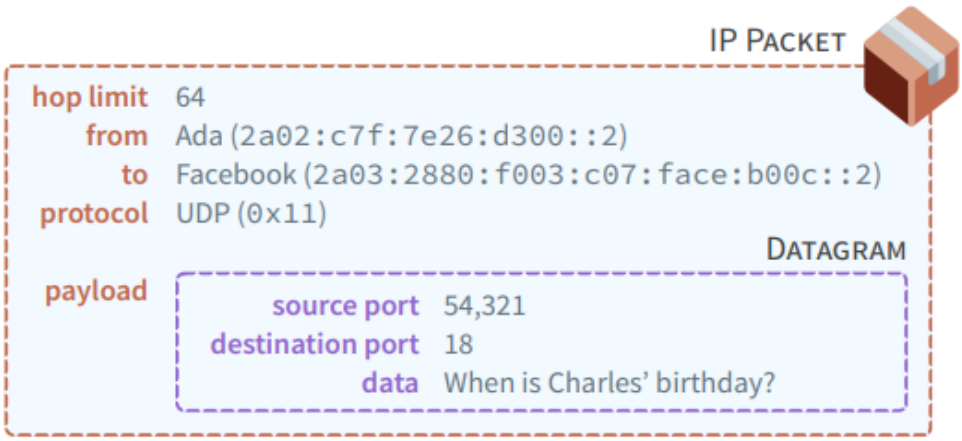
۱ Transform Layer

۲ **U**ser **D**atagram **P**rotocol یا UDP

۳ Port Number

۴ Socket

این متد برنامه‌ی ایدا را تا زمانی که دیتاگرام به پورت ۵۴۳۲۱ برسد متوقف می‌کند. هنگامی که این اتفاق می‌افتد، داده‌های دریافتی در `received` ذخیره و برنامه از سر گرفته می‌شود.



شکل ۱-۲۹: زمانی که بار بسته‌ی IP با قالب UDP مطابقت داشته باشد، **دیتاگرام** نامیده می‌شود. بسته‌های حامل دیتاگرام دارای شماره پروتکل `0x11` هستند.

کلاینت و سرور: برای برقراری ارتباط دو برنامه، یکی باید منتظر تماس باشد و دیگری باید آن را شروع کند. اولی را **سرور**[۱] یا **خدمات‌دهنده** و دومی را **کلاینت**[۲] یا **مشتری** می‌نامیم. در مثال ما، برنامه‌ی فیس‌بوک سرور و برنامه‌ی ایدا کلاینت است. با این حال، برنامه‌ی ایدا می‌تواند به عنوان یک سرور نیز عمل کند. باید به سادگی یک سوکت ایجاد، آن را به پورتی که در آن انتظار دریافت دیتاگرام را دارد متصل کرده و منتظر تماس باشد:

```
socket ← Socket.new(IP, UDP)
socket.bind(17)
received ← socket.recv()
```

شما شماره‌ی پورت سرور خود را انتخاب می‌کنید. اگر برنامه‌ی دیگری در میزبان قبلاً یک سوکت متصل به آن پورت داشته باشد، `bind()` یک خطا ایجاد می‌کند. هنگامی که `recv()` فراخوانی می‌شود، برنامه متوقف شده تا زمانی که دیتاگرام حاوی شماره‌ی پورت شما بیاید. همراه با داده‌های دیتاگرام، آدرس IP فرستنده و شماره پورت منبع در متغیر دریافتی ذخیره می‌شود. به این ترتیب، برنامه می‌تواند در صورت نیاز پاسخ دهد:

---

۱ Server
۲ Client

```
received ← socket.recv()
x ← process_data(received.data)
socket.sendto(x, received.src_ip, received.src_port)
```

جمع‌آزمای UPD: دیدیم که هر کامپیوتر و مسیریاب در یک شبکه می‌تواند جامعیت قاب‌های اترنت و وای‌فای را با استفاده از فیلد FCS که دریافت می‌کند بررسی کند. یک دیتاگرام در داخل یک بسته‌ی IP برای هر پیوند روی یک قاب متفاوت حمل می‌شود، بنابراین FCS در هر انتقال تغییر می‌کند. متأسفانه، این بدان معناست که هیچ راهی برای میزبانی که دیتاگرام را دریافت می‌کند وجود ندارد تا بداند آیا همه‌ی این فیلدهای FCS به درستی تولید و تأیید شده‌اند یا خیر.

برای رفع این مشکل، هر دیتاگرام دارای بررسی‌کننده جامعیت مخصوص به خود به نام **جمع‌آزمای UPD**[1] است. این جمع‌آزما فقط یک بار تولید می‌شود و یک بار به ترتیب توسط میزبان‌های ارسال‌کننده و دریافت‌کننده‌ی آن تأیید می‌شود. دیتاگرام‌های خراب به طور خودکار توسط میزبان دریافت‌کننده دور ریخته می‌شوند، بنابراین برنامه‌های کاربردی آن میزبان می‌توانند مطمئن باشند که داده‌های دریافتی به طور تصادفی آسیب ندیده‌اند.

محدودیت‌های UPD: یک دیتاگرام باید به قدری کوتاه باشد که بتواند درون بار یک بسته‌ی IP قرار گیرد. از آنجایی که MTU برای بخش عمده‌ی اینترنت ۱۵۰۰ بایت است، اکثر دیتاگرام‌ها برای رعایت این محدودیت طراحی شده‌اند. یک پیام سنگین، مانند یک عکس بزرگ، باید از طریق چندین بسته‌ی IP ارسال شود.

به یاد داشته باشید که هر بسته‌ی IP ممکن است به مسیریاب شلوغ در مسیر خود برخورد کند و رها شود. گاهی اوقات، میزبان‌ها حتی در صورت ناپدید شدن بسته‌های آن‌ها مطلع نمی‌شوند. علاوه بر این، انتقال بسته‌ها توسط مسیریاب‌ها می‌تواند باعث بی‌نظمی در دریافت یا تکرار آن‌ها شود. این امر بازیابی داده‌هایی را که در چندین دیتاگرام تقسیم شده‌اند دشوار می‌کند.

پروتکل UDP برای برنامه‌هایی که درخواست‌ها و پاسخ‌های آن‌ها هر کدام دقیقاً در یک دیتاگرام قرار می‌گیرند، مناسب‌تر است. در حالت ایده‌آل، یک برنامه باید هر بار یک درخواست ارسال و فقط زمانی که پاسخ آن را دریافت کرد، دیتاگرام بعدی را ارسال کند. اگر دریافت پاسخ بیش از حد طول بکشد، برنامه باید یا منصرف شده یا سعی کند دوباره همان دیتاگرام را ارسال کند.

---

[1] UDP Checksum

دیتاگرام‌ها همچنین برای انتقال جریان‌های داده‌ای مناسب هستند که در آن‌ها از دست دادن گاه به گاه داده قابل قبول است، مانند هنگام انتقال یک تماس تلفنی زنده، که در آن اشکالات صوتی کوچک اهمیت زیادی ندارند.

پروتکل کنترل انتقال

لایه‌ی انتقال می‌تواند به غیر از تطبیق داده‌های بسته‌های IP با سوکت‌های برنامه‌های کاربردی مربوط به آن‌ها، کارهای بیشتر دیگری نیز انجام دهد. در زمان استفاده از **پروتکل کنترل انتقال**[۱] یا TCP به جای UDP، میزبان‌ها اطلاعات بیشتری را به بسته‌های IP اضافه می‌کنند. این امر به TCP اجازه می‌دهد عملکردهایی را ارائه دهد که قابلیت اطمینان ارتباطات بین برنامه‌ها را افزایش می‌دهند.

در زمان استفاده از TCP، برنامه‌ها داده‌ها را به گونه‌ای مبادله می‌کنند که گویی یک خط ارتباطی مستقیم با برنامه‌ی دیگر وجود دارد، حتی اگر بسته‌های IP اندازه‌ی محدودی داشته باشند و نتوان به مسیریاب‌ها برای تحویل آن‌ها اعتماد کرد. در پشت صحنه، میزبان‌ها داده‌ها را به قطعات کوتاه‌تر تقسیم می‌کنند و با بسته‌های گم شده، تکراری و بی نظم سروکار دارند. برنامه فقط باید از میزبان خود بخواهد که با برنامه‌ی دیگری ارتباط برقرار کند و سپس درخواست ارسال یا دریافت داده‌ها را بدهد.

می‌توانید سوکت‌های UDP را به‌عنوان صندوق‌های پستی تصور کنید که پیام‌های کوتاه و تک قسمتی را، بدون هیچ تضمینی برای تحویل، به صندوق‌های دیگر ارسال و دریافت می‌کنند. سوکت‌های TCP متفاوت هستند. یک سوکت TCP فعال را به عنوان یک لوله‌ی مجازی به سوکت دیگر تصور کنید. تمام داده‌های ارسال شده از طریق یک سوکت TCP می‌توانند به صورت دست نخورده تحویل داده شوند. بیایید ببینیم چگونه می‌توان این کار را در بر روی سیستم غیرقابل اعتماد تحویل بسته‌ی IP انجام داد.

سگمنت‌های TCP

در زمان استفاده از TCP، داده‌های برنامه تقسیم شده و از طریق بارهایی به نام **سگمنت**[۲] ارتباط برقرار می‌کنند. همانند دیتاگرام‌های UDP، سگمنت‌های TCP داده‌ها را با برخی اطلاعات اضافی بسته‌بندی می‌کنند. هر یک از این فیلدها برای TCP به منظور تضمین جامعیت داده‌های کلی و دوباره مونتاژ شده ضروری هستند:

---

[۱] **T**ransmission **C**ontrol **P**rotocol یا **TCP**

[۲] TCP Segment

شکل ۱-۳۰: یک نمایش ساده از یک سگمنت. بسته‌های حامل سگمنت‌ها دارای شماره پروتکل برابر با 0x6 است.

شماره‌های پورت: یک میزبان معمولاً چندین کانال TCP را برای برنامه‌های مختلف خود مدیریت می‌کند. هنگامی که یک سگمنت دریافت می‌شود، باید راهی برای میزبانی وجود داشته باشد تا آن را با کانال ارتباطی که به آن تعلق دارد مطابقت دهد. برای این منظور، سگمنت‌ها شماره‌ی پورت مبدا و مقصد خود را دارند. یک کانال ارتباطیTCP را می‌توان به طور منحصربه‌فرد با چهار عدد شناسایی کرد: آدرس‌هایIP میزبان‌ها، و دو شماره پورت که هریک توسط یک میزبان انتخاب می‌شوند.

شماره‌ی توالی: میزبان ارسال‌کننده داده‌های برنامه را به یک دنباله از قطعه‌های کوچک تقسیم می‌کند. به هر قطعه یک **شماره‌ی توالی**[۱] اختصاص می‌دهد تا نشان دهد چگونه باید آن‌ها را مرتب کرد. هر قطعه در یک سگمنت با شماره‌ی توالی مربوطه جا می‌گیرد و سگمنت‌ها یکی یکی با استفاده از بسته‌هایIP مجزا منتقل می‌شوند. اگر سگمنت‌ها نامرتب برسند، میزبان دریافت‌کننده با استفاده از

---

[۱] Sequence Number

شماره‌های توالی آن‌ها را دوباره مرتب می‌کند. اگر یک سگمنت دو بار دریافت شود، میزبان متوجه شماره توالی تکراری شده و داده‌های تکراری را دور می‌اندازد.

شماره‌ی تایید: ارتباط با استفاده از TCP دو طرفه است: برنامه‌های راه دور باید به طور *همزمان* سگمنتها را مبادله کنند. به منظور پیگیری سگمنت‌هایی که هر میزبان دریافت کرده است، هر کدام از سگمنت‌ها شامل یک **شماره‌ی تایید**[۱] هستند. این شماره مربوط به شماره‌ی توالی سگمنت بعدی است که میزبان انتظار دارد از همتای خود دریافت کند. به عنوان مثال، اگر چارلز سگمنتی را با شماره‌ی تأیید ۴۴ برای ایدا ارسال کند، او تایید می‌کند که تمام سگمنت‌های ایدا را تا شماره‌ی توالی ۴۳ دریافت کرده است. اگر چارلز داده‌ای برای ارسال به ایدا نداشته باشد، همچنان باید سگمنت‌هایی را بدون داده برای برآورده کردن اهداف تأییدیه بفرستد.

پس از ارسال یک بخش غیر تهی، میزبان فرستنده یک زمان‌سنج برای دریافت تاییدیه‌های خود راه می‌اندازد. اگر دریافت یک تاییدیه خیلی طول بکشد، فرستنده فرض می‌کند بسته گم شده و در نتیجه سگمنت تایید نشده را دوباره ارسال و زمان‌سنج را بازنشانی می‌کند.

میزبان‌ها بر زمان لازم برای رسیدن تاییدیه‌ها نظارت می‌کنند. وقتی زمان افزایش می‌یابد، نشانه‌ی این است که شبکه در حال شلوغ شدن است، بنابراین میزبان‌ها سرعت ارسال سگمنت‌ها را کاهش می‌دهند. هنگامی که زمان تایید کاهش می‌یابد، نشانه‌ی آن است که پهنای باند بیشتری در دسترس است، بنابراین میزبان‌ها سگمنت‌ها را با نرخ بیشتری ارسال می‌کنند. این فرایند **کنترل تراکم**[۲] نامیده می‌شود و تضمین می‌کند که از پهنای باند شبکه به درستی استفاده می‌شود.

اندازه‌ی پنجره: اگر یک میزبان سگمنت‌های زیادی را به طور همزمان دریافت کند، منابع محاسباتی آن ممکن است بسیار درگیر شوند. میزبان‌ها برای اطمینان از این‌که همیشه قادر به پردازش داده‌های دریافتی هستند، یک اندازه‌ی پنجره[۳] را در کنار شماره‌ی تایید قرار می‌دهند. این عدد نشان می‌دهد میزبان بعد از آخرین سگمنتی که تایید کرده مایل است چند بایت داده دریافت کند.

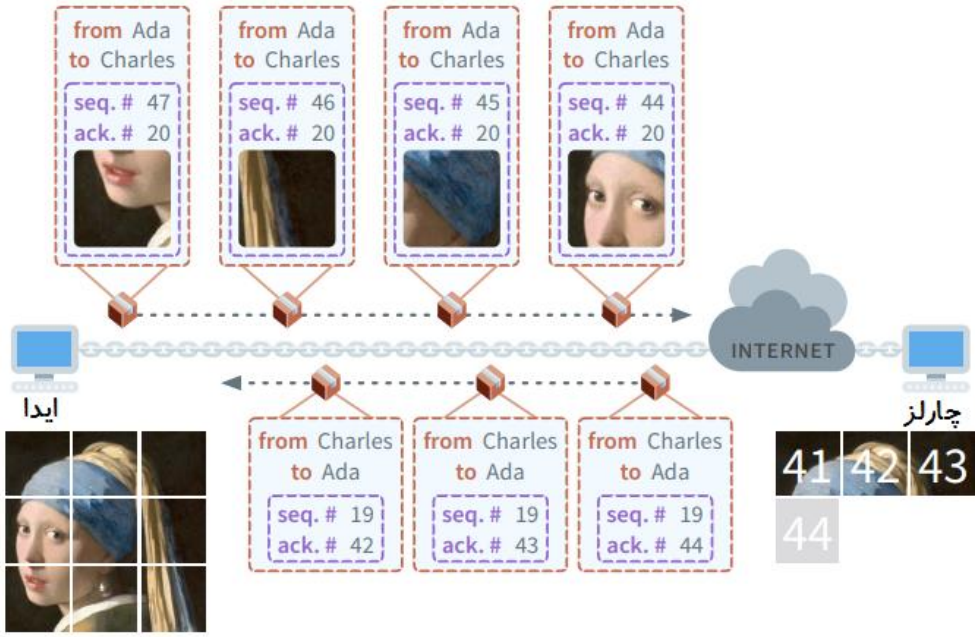
فرستنده‌ها هر زمان که اندازه‌ی کل اطلاعات ارسالی آن‌ها به اندازه‌ی پنجره‌ی همتای‌شان برسد، انتقال بخش‌های تاییدنشده را متوقف می‌کنند. این فرایند **کنترل جریان**[۴] نامیده می‌شود و تضمین می‌کند که از قابلیت‌های محاسباتی هر میزبان تجاوز نشود. با استفاده ازTCP، برنامه‌ها نیاز نیست در مورد کنترل تراکم و کنترل جریان نگران باشند، زیرا میزبان آن‌ها از این موارد مراقبت می‌کنند. از سوی دیگر،

---

۱ Acknowledgment Number
۲ Congestion Control
۳ Window Size
۴ Flow Control

برنامه‌های کاربردی که از UDP استفاده می‌کنند، خود وظیفه‌ی تطبیق نرخ ارسال با پهنای باند شبکه و منابع محاسباتی همتایان خود را بر عهده دارند.

جمع‌آزمای TCP: همانند دیتاگرام‌های UDP، بخش‌های TCP شامل یک جمع‌آزما هستند. این جمع‌آزما توسط میزبان ارسال‌کننده اضافه شده و توسط میزبان دریافت‌کننده تایید می‌شود. اگر یک سگمنت خراب دریافت شود، بدون تایید حذف می‌شود. این امر تضمین می‌کند که تمام قطعه‌های داده در نهایت بدون خطا به مقصد می‌رسند.



شکل ۱-۳۱: چارلز تصویری از ایدا دریافت می‌کند. او هیچ داده‌ای برای ارسال به ایدا ندارد، بنابراین سگمنت‌های تاییدیه خالی را ارسال می‌کند. توجه داشته باشید که شماره‌های توالی سگمنت‌های او افزایش نمی‌یابند. ایدا در حین ارسال داده های خود، اطلاع می‌دهد که آماده‌ی دریافت سگمنت شماره‌ی ۲۰ چارلز است.

اتصال TCP

همانطور که در شکل ۱-۳۱ مشاهده می‌شود، ارتباطاتTCP همیشه از طریق یک جفت رشته سگمنت به نام **اتصال TCP**[۱] برقرار می‌شوند. قبل از اینکه دو برنامه بتوانند تبادل داده را از طریق یک اتصالTCP آغاز کنند، میزبان آن‌ها باید بتواند بخش‌های آن اتصال را شناسایی و آماده کند. آن‌ها این کار را با یادآوری اینکه کدام شماره پورت با کدام رشته مطابقت دارد انجام می‌دهند. هر میزبان همچنین یک شماره‌ی توالی اولیه را برای شروع اتصال سمت خود انتخاب و شماره‌ی توالی اولیه همتای خود را تأیید می‌کند.

به این دلایل، ایجاد یک اتصالTCP نیازمند تبادل سه سگمنت بین کلاینت و سرور است. دو سگمنت اول حامل یک پرچم[۲] SYN ویژه هستند که هدف آن همگام‌سازی[۳] توالی و شماره‌های تایید رشته‌ها است. علاوه بر این، یک پرچم ACK توسط سگمنت‌هایی حمل می‌شود که شماره‌های تأیید[۴] آن‌ها معتبر و همگام‌سازی شده است. حال سگمنت سوم می‌تواند شروع به حمل داده‌های برنامه کند.

قبل از اینکه تبادل سگمنت در شکل ۱-۳۲ انجام شود، چارلز باید از میزبان خود بخواهد که انتظار درخواست‌های اتصال TCP ورودی را در پورت شماره ۲۰ داشته باشد. هنگامی که ایدا از میزبان خود درخواست می‌کند که یک اتصال جدید با چارلز در پورت ۲۰ شروع کند، میزبان ایدا یک شماره پورت را به صورت تصادفی انتخاب می‌کند تا به اتصال تخصیص دهد. در اینجا، پورت ۶۱۳۱۱ انتخاب شده است.

تا زمانی که یک اتصال فعال است، میزبان‌ها شماره‌های توالی، شماره‌ی تایید و اندازه‌ی پنجره را حفظ می کنند. آن‌ها به تأییدیه‌های همتایان خود زمان اختصاص می‌دهند و تصمیم می‌گیرند که چه زمانی سگمنت‌های بعدی را ارسال کنند. در این شرایط، برنامه‌ی کاربردی در کمال آرامش به سر می‌برد و از همه‌ی این کارها بی‌خبر است، زیرا فقط متدهای سوکتTCP را برای اتصال، ارسال و دریافت فراخوانی می‌کند.

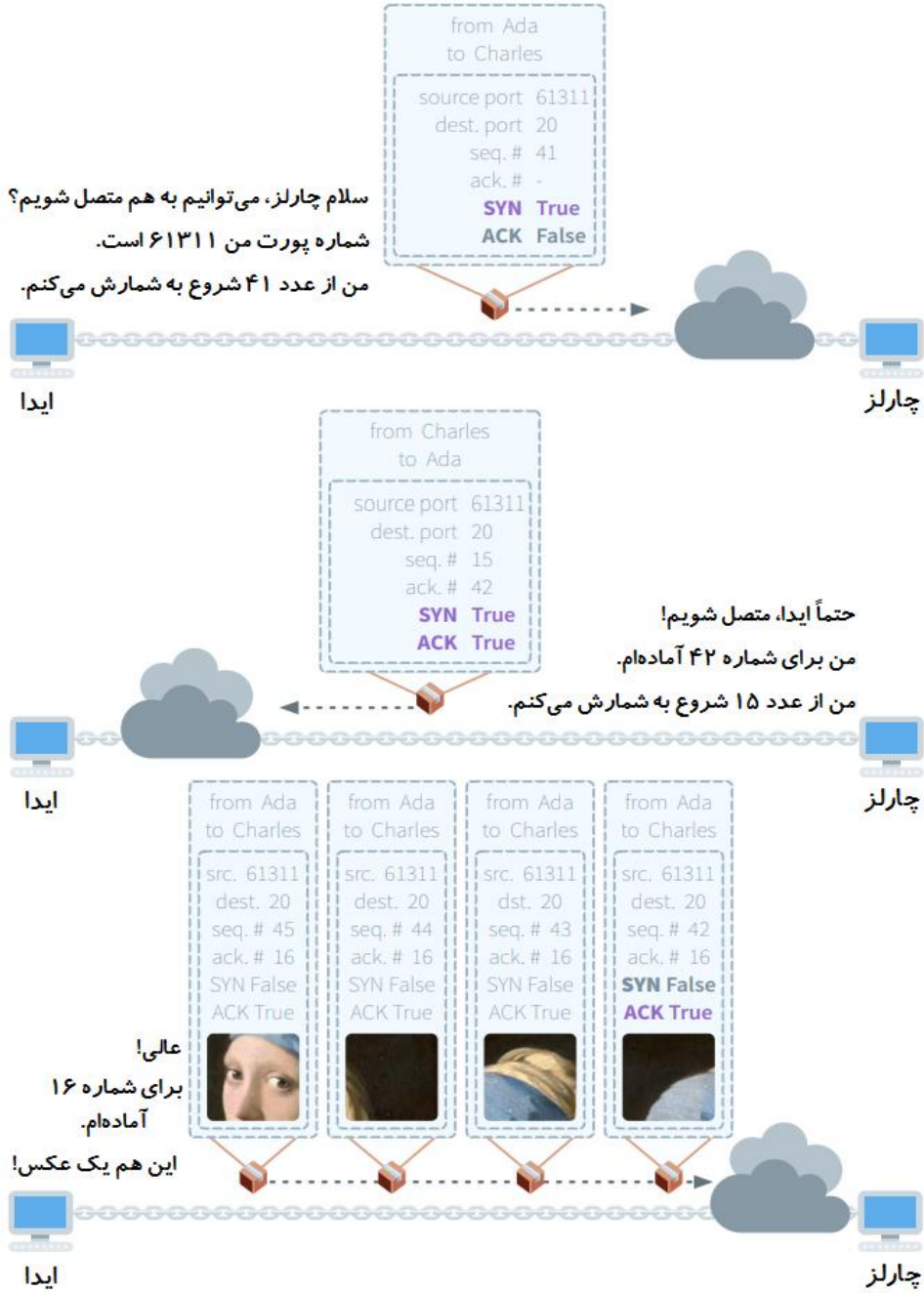
---

[۱] TCP Connection

[۲] فیلدهایی که با یک رقم صفر یا یک کدگذاری می‌شوند، پرچم یا Flag نامیده می‌شوند.

[۳] **SYN**chronize

[۴] **ACK**nowledgement Number



شکل ۱-۳۲: میزبان‌ها اتصالات TCP را با شماره‌های توالی تصادفی آغاز می‌کنند. از لحظه‌ای که سگمنت سوم دریافت می‌شود، از نظر ما اتصال به طور کامل برقرار است، زیرا هر دو میزبان تأییدیه و شماره‌ی توالی خود را به دست آورده‌اند.

سوکت‌های TCP

سوکت‌های TCP فعال یا غیرفعال هستند. فقط سوکت‌های فعال می‌توانند به یک اتصال مرتبط شوند. هنگامی که سوکت‌های TCP ایجاد می‌شوند، به طور پیش‌فرض فعال هستند. در اینجا نحوه‌ی شروع یک اتصال توسط کلاینت آمده است:

```
active_socket ← Socket.new(IP, TCP)
port_number ← 80
server_ip ← 2a03:2880:f003:c07:face:b00c::2
active_socket.connect(server_ip, port_number)
active_socket.send("When is Charles' birthday?")
```

کلاینت فقط باید آدرسIP سرور میزبان و شماره‌ی پورتی را که سرور به آن گوش می‌دهد مشخص کند. تمام مراحل شروع و حفظ اتصالTCP توسط میزبان و در پشت صحنه انجام می‌شوند. پس از اجرای موفقیت‌آمیز ()connect، متدهای ()send و ()recv می‌توانند برای تبادل داده‌ها فراخوانی شوند.

به منظور انتظار و پذیرش درخواست‌های اتصال در یک شماره‌ی پورت معین، باید از یک سوکت غیرفعال استفاده شود. یک سوکت پس از مرتبط شدن به یک شماره‌ی پورت و فراخوانی متد ()listen آن، غیرفعال می‌شود. در اینجا نحوه‌ی انتظار سرور برای اتصال در پورت ۸۰ آمده است:

```
server_socket ← Socket.new(IP, TCP)
server_socket.bind(80)
server_socket.listen()
active_socket ← server_socket.accept()
```

سرور، شماره‌ی پورت خود را انتخاب می‌کند. اگر برنامه‌ی دیگری قبلاً یک سوکت مرتبط به همان پورت داشته باشد، متد ()bind یک خطا می‌دهد.

هنگامی که متد ()accept فراخوانی می‌شود، سرور متوقف می‌شود تا زمانی که یک درخواست اتصال جدید وارد شود، در این مرحله یک اتصالTCP جدید ایجاد می‌شود. اتصال جدید به یک سوکت فعال تازه ایجاد شده مرتبط است. سوکت غیرفعال اصلی را می‌توان برای پذیرش اتصالات بیشتر به میزبان‌های دیگر استفاده کرد. سوکت فعال جدید ایجاد شده را می‌توان برای برقراری ارتباط با مشتری با فراخوانی ()send و ()recv بر روی آن استفاده کرد.

اغلب، سرورها برای برقراری بسیاری از اتصالاتTCP با چندین کلاینت و برقراری ارتباط با همه‌ی آن‌ها به طور همزمان، به اجرای ()accept در یک حلقه ادامه می‌دهند. به این ترتیب است که یک وب سرور می‌تواند هزاران یا حتی میلیون‌ها برنامه‌ی مختلف کلاینت را به طور همزمان ارائه دهد.

این یک نمای کلی بسیار ساده از نحوه‌ی کار اتصالات TCP بود. موارد حاشیه‌ای زیادی وجود دارند که در مورد آن‌ها صحبت نکرده‌ایم. سگمنت‌ها دارای پرچم‌های اضافی برای نشان دادن فوریت، حالت‌های خطا، ازدحام شبکه، خاتمه‌ی اتصال و موارد دیگر هستند. اگرچهTCP در ابتدا و در مدت کوتاهی پس از IP در سال ۱۹۷۴ طراحی شد، اما عملکرد درونی آن تا به امروز در حال تغییر است؛ به عنوان مثال برای بهبود مدیریت ازدحام و بهینه‌سازی استفاده از پهنای باند.

اگر در شبکه‌سازی تخصص ندارید، سوکت‌هایTCP به شما این امکان را می‌دهند که از همه‌ی این جزئیات بی‌خبر بمانید. آن‌ها نمونه‌ای از مهندسی صحیح سیستم هستند: زیر آن‌ها یک رابط ساده برای انجام عملیات پیچیده با حداقل دانش لازم در مورد پیچیدگی‌های اساسی ارائه می‌دهند.

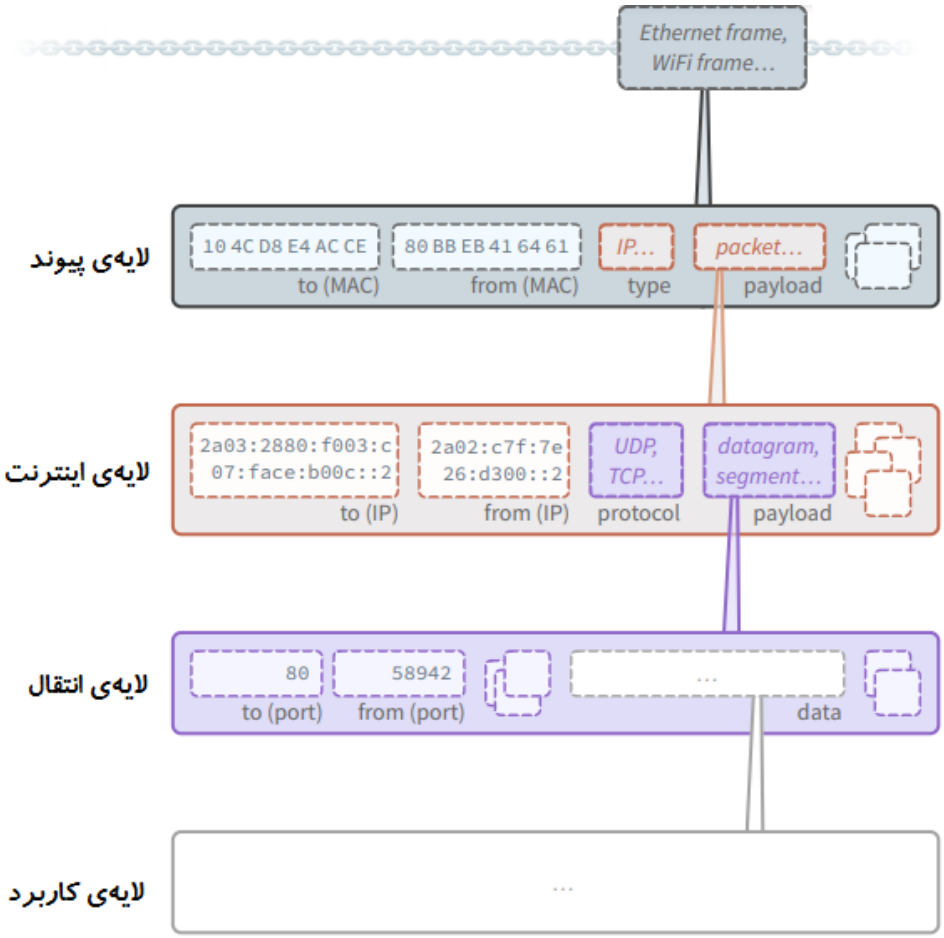
نتیجه‌گیری

اینترنت یکی از بزرگ‌ترین دستاوردهای فناورانه‌ی بشر و نتیجه‌ی حجم عظیمی از مهندسی و همکاری‌های بین المللی است. برنامه‌های کامپیوتری تقریباً در هر نقطه‌ی از جهان می‌توانند بدون نقص و بدون زحمت و بدون نیاز به هماهنگی مرکزی با یکدیگر ارتباط برقرار کنند.

چنین ارتباطاتی از اتصالات فیزیکی و مجازی در سطوح مختلف سخت‌افزار و نرم‌افزار استفاده می‌کنند. هنگامی که برنامه‌ها با هم ارتباط برقرار می‌کنند، طبیعی است که داده‌های آن‌ها در چندین لایه‌ی بسته‌بندی محصور شود.

لایه‌ی پیوند: آنتن‌ها، کابل‌ها و سایر تجهیزات برای دستکاری امواج الکترومغناطیسی به زوج کامپیوترها اجازه می‌دهند تا سیگنال‌ها را به یکدیگر ارسال کنند. پروتکل‌های لایه‌ی پیوند به این زوج کامپیوترها اجازه می‌دهند تا از این سیگنال‌ها برای تبادل داده‌ها در قاب‌ها استفاده کنند. در کنار سایر موارد دیگر، هر قاب حاوی آدرس‌هایMAC فرستنده و گیرنده‌ی خود، یک بار و یک فیلد است که قوانین مدیریت بار را در لایه‌ی اینترنت نشان می‌دهد.

لایه‌ی اینترنت: کامپیوترهای خاصی به نام مسیریاب می‌توانند شبکه‌های مختلف را به شبکه‌های بزرگ‌تر مانند اینترنت متصل کنند. به لطف پروتکل اینترنت، بارهای بسته‌بندی‌شده در قالب بسته‌هایIP می‌توانند از یک قاب به قاب دیگر منتقل شوند و بنابراین بین کامپیوترهایی که پیوند فیزیکی مستقیمی با هم ندارند حرکت کنند. در این بین، هر بسته حاوی آدرس‌هایIP مبدأ و گیرنده‌ی نهایی، یک بار و یک فیلد است که قوانین مربوط به مدیریت آن بار را در لایه‌ی انتقال نشان می‌دهد.



شکل ۱-۳۳: لایه‌های مختلف ارتباط.

لایه‌ی انتقال: کامپیوترهای واقع در فاصله‌های دور میزبان برنامه‌هایی هستند که مایل به برقراری ارتباط با یکدیگرند. به لطف پروتکل‌هایی مانند UDP و TCP، بارهای متعددی که به صورت دیتاگرام یا سگمنت نمایش داده می‌شوند، می‌توانند جریان‌هایی از داده‌ها را بین برنامه‌های راه دور تشکیل دهند. در کنار سایر موارد، هر دیتاگرام یا سگمنت شامل قطعه‌ای از داده‌های برنامه به همراه پورت‌های مبدا و مقصد است که آن داده‌ها از طریق آن‌ها جریان دارند.

هر دو پروتکل TCP و UDP بسته به ماهیت ارتباطات بین برنامه‌ها به روش‌های مختلفی استفاده می‌شوند. در فصل بعدی، **لایه‌ی کاربرد**[۱] را پوشش خواهیم داد: چگونه IP، TCP و UDP را برای

[۱] Application Layer

ساخت بسیاری از خدمات اینترنتی مدرنی که هر روز استفاده می‌کنیم، مانند ایمیل و شبکه‌ی جهانی وب، به کار می‌گیریم.

این معماری لایه‌ای برای شبکه مدت‌ها است که داشته است. در سال ۱۹۷۵، بسته‌هایIP حامل بارهای TCP در سطح بین‌المللی بین کامپیوترهای استنفورد و لندن در حال حرکت بودند. پروتکل‌ها تکامل می‌یابند؛ IP در سال ۲۰۱۲ ارتقا پیدا کرد وTCP بارها بهینه‌سازی شده است اما مکانیزم اساسی اینترنت برای چندین دهه تغییر نکرده است.

در این فصل، مفاهیمی اساسی از شبکه را پوشش دادیم که احتمالاً ما را برای سال‌های بسیار بیشتری متصل نگه می‌دارند. بیایید اکنون بیاموزیم که چگونه می‌توانیم از اتصال اینترنت برای تقویت خدمات دیجیتالی مانند پست الکترونیکی و شبکه‌ی جهانی وب استفاده کنیم.

منابع و مراجع

- Warriors of the Net
  - http://code.energy/net-warriors
- Introduction to Networking, by Charles Severance
  - http://code.energy/severance

فصل ۲

ارتباط

اینترنت درست کار می‌کند زیرا افراد زیادی برای
انجام کارها با هم همکاری می‌کنند.

– جان پاستل[۱]

صدای ما به ما امکان می‌دهد اصوات را برای دیگران ارسال کنیم، اما این برای برقراری ارتباط کافی نیست. برای اینکه دو نفر از طریق گفتار به تبادل افکار بپردازند، باید زبان مشترکی داشته باشند. به همین ترتیب، پروتکل‌های انتقال، داده‌ها را قادر می‌سازند بین برنامه‌ها جریان داشته باشند، اما این برای درک برنامه‌ها و کار کردن با یکدیگر کافی نیست. علاوه بر اتصالات، برنامه‌ها به قوانین ارتباطی نیز نیاز دارند.

بسیاری از قوانین ارتباطی که در روزهای اولیه اینترنت پدیدار شدند، امروزه هنوز به طور گسترده توسط برنامه‌های کاربردی مورد استفاده قرار می‌گیرند. اگر می‌خواهید برنامه‌های کاربردی شما در اینترنت شرکت کنند، باید با برخی از آن‌ها آشنا شوید. در این فصل، شما یاد خواهید گرفت که:

به آدرس‌های آنلاین **نام** اختصاص دهید،


ساعت‌ها را با **زمان** جهانی هماهنگ کنید،


به کامپیوترهای راه‌دور **دسترسی** یافته و از آن‌ها استفاده کنید،


نامه‌های الکترونیک را ارسال و دریافت کنید،


به مرور اسناد و خدمات **وب** بپردازید.

هر مجموعه‌ی مستقل از قوانین ارتباطی یک پروتکل را تشکیل می‌دهند و مجموعه‌ی همه‌ی این پروتکل‌ها لایه‌ی کاربرد را ایجاد می‌کنند. اولین پروتکل لایه‌ی کاربرد که مشاهده خواهیم کرد مانند یک دفترچه‌ی آدرس عمل می‌کند که تقریباً هر برنامه‌ای که به اینترنت متصل است به طور مستقیم یا غیرمستقیم با آن ارتباط دارد.

---

[۱] Jon Postel (1943-1998): دانشمند آمریکایی در فناوری اطلاعات و یکی از همکاران اولیه‌ی پروژه‌ی آرپانت. م.

۱-۲- نام‌ها

فقط در صورتی می‌توانیم یک بسته‌ی IP را برای یک میزبان ارسال کنیم که آدرس IP آن را بدانیم. با این حال، ما هرگز حتی آدرس‌های IP میزبان‌های اصلی مانند گوگل، آمازون یا فیس‌بوک را حفظ نمی‌کنیم.

مهندسان متوجه شدند که ارجاع به میزبان‌ها با استفاده از اعداد بزرگ غیرعملی است، بنابراین **سیستم نام دامنه**[۱] یا DNS را ایجاد کردند: راهی برای مرتبط کردن نام‌های خوانا به آدرس‌های IP. به لطف DNS، ما می‌توانیم با استفاده از نام‌هایی مانند google.com، amazon.com یا facebook.com به میزبان‌ها متصل شویم.

وقتی عبارت facebook.com را در یک مرورگر وب تایپ می‌کنید، از DNS برای پیدا کردن آدرس IP مرتبط با آن نام استفاده می‌کند. سپس می‌تواند با ارسال بسته‌های IP به آن آدرس، وب سایت را درخواست کند. اگر آدرس IP فیس‌بوک را مستقیماً در مرورگر تایپ کرده بودید، رجیستری DNS مورد بررسی قرار نمی‌گرفت. با این حال همان صفحه وب همچنان نشان داده می‌شود[۲].

سیستم نام دامنه یا DNS به همکاری بسیاری از کامپیوترها در سراسر اینترنت وابسته است. این کامپیوترها طبق پروتکل DNS برای حفظ رجیستری نام، ارتباط برقرار می‌کنند. بیشتر برنامه‌های کاربردی اینترنتی، مانند مرورگرهای وب و کلاینت‌های ایمیل، از DNS برای یافتن آدرس‌های IP میزبان‌هایی که باید با آن‌ها ارتباط برقرار کنند، استفاده می‌کنند. اما قبل از اینکه بتوانیم قوانین این پروتکل را بررسی کنیم، بیایید برخی از مفاهیم زیربنایی را بیاموزیم. به این منظور از ساختار سلسله‌مراتبی نام‌ها شروع می‌کنیم.

دامنه‌ها

نام ها در DNS **نام دامنه**[۳] یا **دامنه** نامیده می‌شوند. آن‌ها از برچسب‌های متنی ساخته شده‌اند که با نقطه از هم جدا می‌شوند. برچسب‌ها می‌توانند از ۲۶ حرف الفبا از a تا z، اعداد از 0 تا 9 و خط فاصله (-) استفاده کنند. به عنوان مثال، book-2.code.energy یک دامنه با سه برچسب است.

نقطه‌ها زیر دامنه‌های سلسله‌مراتبی را نشان می‌دهند. دامنه‌های فرعی دامنه‌هایی هستند که توسط یک دامنه‌ی اصلی کوتاه‌تر مشخص می‌شوند: book-2.code.energy یک زیردامنه از

---

[۱] Domain Name System یا **DNS**

[۲] خودتان ببینید، از وب‌سایت http://[2a03:2880:f003:c07:face:b00c::2] بازدید کنید.

[۳] Domain Name

code.energy است. به همین ترتیب، code.energy یک زیردامنه از دامنه‌ی تک برچسبی energy است. توجه داشته باشید که دامنه‌ها به حروف بزرگ و کوچک حساس نیستند: code.energy و coDE.enERgy یکسان هستند.

هر دامنه یک مالک دارد. مالکان می‌توانند زیردامنه‌های دامنه‌های خود را ایجاد کنند. مالکیت یک زیردامنه در هر زمان توسط مالک دامنه اصلی قابل انتقال است. به عنوان مثال، Verisign یک شرکت آمریکایی است که مالک هر دو دامنه‌ی com و net است. در سال ۱۹۹۷، به درخواست یک تیم کوچک از افراد بسیار علاقمند، شرکت Verisign، دامنه‌ی google.com را ایجاد و مالکیت آن را به آن‌ها اعطا کرد.

دامنه‌های تک برچسبی مانند com، net و energy، **دامنه‌های سطح بالا**[۱] یا TLD نامیده می‌شوند. سیستم DNS با توجه به نحوه‌ی طراحی آن، زمانی که تعداد محدودی از TLD وجود دارد کارآمدتر است. به همین دلیل، ایجاد TLD جدید با این طراحی دشوار است. معمولاً وقتی افراد و سازمان‌ها دامنه می‌خواهند، از صاحب یک TLD موجود، به ویژه com، یک زیر دامنه درخواست می‌کنند[۲].

با این وجود، بسیاری از سازمان‌های قدرتمند دارای دامنه‌های سطح بالای خاص خود هستند. در سال ۲۰۱۴ دامنه‌ی سطح بالای google به گوگل اعطا شد[۳]. در سال ۲۰۱۶ موتورولا دامنه‌ی سطح بالای moto را دریافت کرد. مهم‌تر از آن، آمازون در تلاش است تا دامنه‌ی سطح بالای amazon را به دست آورد، اما دولت‌های برزیل و پرو با این کار مخالفت می‌کنند، زیرا ادعا دارند که این نام در درجه‌ی اول متعلق به جنگل‌های بارانی آن‌ها است.

ICANN

برای اینکه نام‌های دامنه به طور جهانی قابل درک باشند، باید راهی وجود داشته باشد که همه در مورد اینکه چه کسی مالکیت هر TLD را دارد، توافق کنند. این امر مستلزم فرآیندهای توافقی برای ایجاد TLD و برای حل و فصل تضادها بر سر مالکیت آن‌ها است. برای این منظور، تصمیم گرفته شد که یک سازمان غیرانتفاعی آمریکایی یعنی **شرکت اینترنتی برای نام‌ها و شماره‌های واگذارشده**[۴] یا **ICANN** اختیار اصلی را در مورد مدیریت دامنه‌های سطح بالا داشته باشد.

---

[۱] Top Level Domains یا **TLD**

[۲] می‌توانید فهرستی از دامنه‌های سطح بالا را در http://code.energy/tlds ببینید.

[۳] خودتان می‌توانید وب سایت http://about.google را ببینید.

[۴] Internet Corporation for Assigned Names and Numbers یا **ICANN**

هنگامی که ICANN اجازه‌ی ایجاد یک TLD را می‌دهد، اغلب از مالک جدید می‌خواهد که به عموم مردم اجازه دهد تا زیردامنه‌ها را ثبت کنند. این مورد برای اکثر دامنه‌های سطح بالای موجود، از جمله com، net و energy صادق است. برای اینکه Verisign مالکیت com خود را حفظ کند، ICANN از آن‌ها خواسته است که درخواست‌های ایجاد زیردامنه‌های com. را از هر کسی بپذیرند[۱].

گاهی اوقات، ICANN به مالکان TLD اجازه می‌دهد تا از آن برای منافع انحصاری استفاده کنند. به عنوان مثال، دامنه‌های gov و mil توسط دولت ایالات متحده کنترل می‌شوند. دولت آمریکا فقط برای نهادهای دولتی آمریکا از این دامنه‌ها زیردامنه ایجاد می‌کنند، مانند nasa.gov و spaceforce.mil. به همین ترتیب، مالک edu فقط به مؤسسات آموزشی آمریکایی اجازه می‌دهد دامنه‌های edu. مانند mit.edu و caltech.edu داشته باشند. و شرکت گوگل از TLD خود منحصراً برای امور شرکت مانند grow.google و blog.google استفاده می‌کند.

علاوه بر دامنه‌های سطح بالای عمومی و دامنه‌های سطح بالای با استفاده‌ی انحصاری، ICANN دامنه‌های سطح بالای دو حرفی با کد هر کشور را تخصیص می‌دهد و کنترل آن‌ها را به دولت‌های مربوطه واگذار می‌کند. کلمبیا دارای دامنه‌ی co، ایالات متحده دارای us، و قلمرو اقیانوس هند بریتانیا دارای io است[۲]. گاهی اوقات، ICANN برای مناطق جغرافیایی معروف TLD ایجاد می‌کند. به عنوان مثال، دامنه‌ی rio به شورای شهر ریودوژانیرو داده شده است.

سرورهای نام

رکوردهای DNS داده‌ها را به دامنه‌ها پیوند می‌دهند. یک دامنه می‌تواند رکوردهای زیادی داشته باشد که هر کدام قطعه‌ای از داده‌ها را به دامنه مرتبط می‌کنند. همه‌ی رکوردها دارای کدی هستند که نوع اطلاعاتی را که دارند نشان می دهد. به عنوان مثال، رکوردهای AAAA یک دامنه را به یک آدرس IP پیوند می‌دهند[۳]:

```
sprint.net.          AAAA 2600::
```

---

[۱] موسسه‌ی ICANN کنترل IANA یعنی سازمان اختصاص‌دهنده‌ی پیشوندهای آدرس IP را بر عهده دارد.

[۲] دامنه‌های سطح بالا با استفاده از کدهای کشورها مطابق استاندارد ISO 3166-1 alpha-2 اختصاص داده می‌شوند، به استثنای چند مورد مانند بریتانیا که کد کشور آن‌ها GB است اما دامنه‌ی UK به آن‌ها داده شده است.

[۳] به دامنه‌ی .sprint.net توجه کنید. در رکوردهای رسمی، تمام دامنه‌ها با یک نقطه ختم می‌شوند. در استفاده‌ی روزمره، برای راحتی بیشتر، نقطه‌ی نهایی حذف می‌شود.

تمام رکوردهای DNS باید در **سرورهای نام**[۱] ذخیره شوند: میزبان‌های ویژه‌ای که رکوردهای خود را در هر زمان با هر کسی در اینترنت به اشتراک می‌گذارند. مالک یک دامنه باید حداقل به دو سرور نام اجازه‌ی ذخیره رکوردها DNS خود را بدهد. مالکان دامنه می‌توانند این سرورها را خودشان تهیه کرده یا اشخاص ثالث را برای این کار استخدام کنند.

مالکیت دامنه‌ها از طریق رکوردهای DNS اعمال می‌شود. به عنوان مثال، Verisign مالک com است. در اصل، سرورهای نام Verisign باید شامل تمام رکوردهای دامنه‌های com. از جمله google.com باشند. با این حال، اینطور نیست، زیرا Verisign رکوردهای زیر را در سرورهای نام خود نگه می‌دارد:

```
google.com.          NS      ns1.google.com.
google.com.          NS      ns2.google.com.
ns1.google.com.      AAAA    2001:4860:4802:32::a
ns2.google.com.      AAAA    2001:4860:4802:34::a
```

یک رکورد NS مشخص می‌کند که سرور نام داده‌شده دارای اختیارات کنترل دامنه است. با رکوردهای بالا، Verisign مسئولیت ذخیره‌ی تمامی رکوردهای مرتبط با google.com را به عهده می‌گیرد. در واقع کنترل دامنه را در اختیار گوگل قرار می‌دهد که میزبان سرورهای نام ذکر شده است. این امر به گوگل این اختیار را می‌دهد که رکوردهایDNS را برای google.com در سرورهای نام خود تنظیم کند.

سرورهای ریشه: برخی از سرورهای نام خاص، که سرورهای ریشه[۲] نامیده می‌شوند، توسط ICANN برای ذخیره رکوردهای DNS دامنه‌های سطح بالا تعیین شده‌اند. دامنه‌های سطح بالای جدید زمانی ایجاد می‌شوند که ICANN از رکوردهای NS می‌خواهد که به این سرورها اضافه شوند. سیزده سرور ریشه وجود دارد که ICANN آدرس IP آن‌ها را ارائه کرده است. دانشگاه مریلند، ناسا و ارتش ایالات متحده از جمله نهادهای میزبان سرور ریشه هستند.

پرس‌وجو کردن

سرورهای نام دو عملکرد دارند: ذخیره رکوردها و پاسخگویی به پرس‌وجوها. اکثر برنامه‌های اینترنتی به آن‌ها متکی هستند. به عنوان مثال، مرورگر وب شما تنها در صورتی کار می‌کند که راهی برای کشف آدرس‌های IP وب‌سایت‌هایی که می‌خواهید به آن‌ها دسترسی داشته باشید، داشته باشد.

---

[۱] Name Servers
[۲] Root Servers

سرورهای نام انتظار پرس‌وجوهایی را در پورت ۵۳ پروتکل UDP دارند. به طور معمول، یک پیام پرس‌وجوی DNS و پاسخ آن در یک دیتاگرام قرار می‌گیرند. برای نشان دادن نحوه‌ی عملکرد آن، اجازه دهید سرورهای نام را با استفاده از برنامه dig مورد پرس‌وجو قرار دهیم[۱]. این برنامه یک دیتاگرام حاوی یک پرس‌وجوی منطبق بر پروتکل DNS را به آدرس IP که ما مشخص کرده‌ایم ارسال می‌کند. دستور Dig از خط فرمان[۲] به صورت زیر فراخوانی می‌شود:

```
dig @[address] [domain] [type]
```

این دستور یک دیتاگرام را به آدرس داده‌شده در پورت ۵۳ پروتکل UDP ارسال می‌کند و در مورد یک نوع خاص رکورد مرتبط با دامنه‌ی داده‌شده سوال می‌کند. هنگامی که یک دیتاگرام در پاسخ برمی‌گردد، dig آن را رمزگشایی می‌کند و رکوردها را به صورت متن ساده نمایش می‌دهد.

فرض کنید می‌خواهیم به icmc.usp.br وصل شویم، اما فقط آدرس IP سرور ریشه‌ی ناسا را میدانیم، یعنی 2001:500:a8::e. بیایید از ناسا بپرسیم که سرور نام مسئول icmc.usp.br کدام است:

```
dig @2001:500:a8::e icmc.usp.br. NS
```

سرور ریشه با رکوردهای زیر پاسخ می‌دهد:

```
br.             NS      a.dns.br.
a.dns.br.       AAAA    2001:12f8:6::10
```

بلافاصله می‌توانیم بفهمیم که ناسا رکوردهای NS برای icmc.usp.br را ندارد. در واقع، حتی نباید انتظار داشت که سرورهای ریشه این رکوردها را ذخیره کنند: br یک TLD با کد کشور است که تحت اختیار دولت برزیل است. سرور ریشه با نشان دادن آدرس IP سرور برزیلی مسئول br به ما کمک می‌کند. بیایید این توصیه را بپذیریم و درخواست خود را به آنجا ارسال کنیم:

```
dig @2001:12f8:6::10 icmc.usp.br. NS
```

این بار، رکوردها خاصی را که خواسته‌ایم دریافت می‌کنیم:

```
icmc.usp.br.        NS      c.dns.usp.br.
c.dns.usp.br.       AAAA    2001:12d0::8
```

---

[۱] برنامه‌ی dig به صورت آماده و از قبل بر روی شبیه‌سازهای ترمینال لینوکس و MacOS نصب شده است. اگر به رابط خط فرمان دسترسی ندارید، از http://code.energy/dig استفاده کنید. اگر نمی‌دانید خط فرمان چیست، نگران نباشید! در ادامه‌ی این فصل در مورد آن بیشتر خواهیم آموخت.

[۲] Command Line

اکنون می‌توانیم رکورد AAAA مرتبط با icmc.usp.br را بخواهیم:

```
dig @2001:12d0::8 icmc.usp.br. AAAA
```

و در نهایت، آدرس IP مورد نظر خود را دریافت می‌کنیم:

```
icmc.usp.br.        AAAA        2001:12d0:2080::231:6
```

اگر پرس‌وجو اصلی ما یک زیردامنه از icmc.usp.br بود، ممکن بود نیاز داشته باشیم که یک بار دیگر درخواست خود را برای یک رکورد NS تکرار کنیم. این کار **پرس‌وجوی تکراری**[۱] نامیده می‌شود و به شما امکان می‌دهد هر نوعی از اطلاعات DNS را که با آدرس IP هر سرور ریشه شروع می‌شود، جستجو کنید.

پرس‌وجوی بازگشتی

در سال ۲۰۲۰، تخمین زده می‌شود که در هر ثانیه چند میلیون پرس‌وجوی DNS انجام می‌شود. اگر میزبان‌ها فقط پرس‌وجوی تکراری را انجام دهند، رسیدگی به تمام این درخواست‌ها برای سرورهای نام بسیار دشوار خواهد بود. خوشبختانه، اکثر پرس‌وجوهای DNS به روشی متفاوت انجام می‌شوند؛ راهی که بار روی سرورهای نام را به حداقل می‌رساند.

برخی از سرورهای نام، که اغلب **سرورهای DNS**[۲] نامیده می‌شوند، به شما امکان می‌دهند از طریق یک پرس‌وجو به هر رکورد DNS دسترسی پیدا کنید. این سرورها معمولاً توسط سازمان‌های بزرگ با توان عملیاتی زیاد مانند ارائه‌دهندگان خدمات اینترنتی نگهداری می‌شوند. یک ISP تقریباً همیشه مسیریاب‌های کلاینت را از آدرس IP سرور DNS خود مطلع می‌کند. مسیریاب‌های مشتریان اغلب از آن سرور برای تمام پرس‌وجوهای DNS خود استفاده می‌کنند.

امروزه اکثر مسیریاب‌های خانگی قادر به اجرای سرور DNS مخصوص خود هستند. به طور معمول، کامپیوترهای شخصی به گونه‌ای پیکربندی شده‌اند که از مسیریاب خود به عنوان سرور DNS پیش‌فرض خود استفاده کنند. اگر دستور dig را بدون آرگومان @ فراخوانی کنید، احتمالاً کامپیوتر شما از مسیریابتان سؤال می‌کند. ، مسیریاب شما نیز به نوبه‌ی خود احتمالاً درخواست را به سرور DNS در ISP ارسال می‌کند. با دستور زیر آن را امتحان کنید:

```
dig icmc.usp.br. AAAA
```

---

[۱] Iterative Querying
[۲] DNS Server

این کار بسیار سریع‌تر است: سرور DNS شما از هر سرور نام دیگری به شما نزدیکتر است. علاوه بر این، با یک پرس‌وجو، رکوردهای DNS مورد نظرتان را دریافت می‌کنید. به عنوان کلاینت، شما حتی نمی‌دانید (یا اهمیت نمی‌دهید) که آیا سرور DNS شما از سرور DNS دیگری پرس‌وجو کرده یا اینکه رکوردها را مستقیماً از سرورهای نامی که تحت مدیریت دارد، واکشی کرده است.

ما این فرآیند را **پرس‌وجوی بازگشتی**[۱] می‌نامیم، زیرا پرس‌وجوی واحد کلاینت زنجیره‌ای از سرورهای DNS را تا زمانی که پاسخ پیدا شود جستجو می‌کند. مزیت این نوع پرس‌وجو در **حافظه‌ی نهان**[۲] نهفته است: سرورها نتایج پرس‌وجوهای گذشته را در حافظه‌ی محلی یا حافظه‌ی نهان خود نگه می‌دارند. به عنوان مثال، اولین باری که یک سرور DNS در مورد هر دامنه `io.` پرس‌وجو می‌شود، از سرور ریشه رکوردهای NS مربوط به `io` را درخواست می‌کند. با این حال، بار بعدی که یک پرس‌وجو برای یک دامنه`io.` دریافت می‌شود، به جای اتلاف وقت سرور ریشه، رکوردها را از حافظه‌ی نهان خود بازیابی می‌کند.

از آنجایی که ICANN تعداد دامنه‌های سطح بالا را محدود کرده است، برای ISPها آسان است که همه‌ی آن‌ها را در سرورهای DNS خود ذخیره و بنابراین فقط به طور پراکنده از سرورهای ریشه پرس‌وجو کنند. این طراحی تضمین می‌کند که سرورهای ریشه بیش از حد، بار دریافت نمی‌کنند.

در روی دیگر سکه، حافظه‌ی نهان انتشار تغییرات در سراسر شبکه را کند می‌کند: سرورهای DNS فقط رکوردهای NS ذخیره‌شده در حافظه‌ی نهان خود را هر از چند گاهی به‌روز می‌کنند. به طور خاص، مقدار دوره‌ی حیات[۳] یا TTL یک رکورد نشان می‌دهد که چه مدت ممکن است در حافظه‌ی نهان ذخیره شود.

در مثال‌های قبلی، ما مقادیر TTL را برای سادگی حذف کرده‌ایم، اما هر رکورد یک مقدار TTL دارد که معمولاً قبل از نوع رکورد نمایش داده می‌شود:

```
br.             149836      NS      a.dns.br.
```

این رکورد دارای مقدارTTL برابر با ۱۴۹۸۳۶ است که نشان می‌دهد رکورد می‌تواند برای این مدت (به ثانیه) در حافظه نهان بماند. حدود ۴۱ ساعت پس از بازیابی این رکورد توسط یک سرور DNS، باید از حافظه‌ی نهان آن حذف شود. تنظیم یک مقدار TTL پایین باعث می‌شود تغییرات آتی در رکورد سریع‌تر منتشر شوند. با این حال، مقادیر بالاتر TTL به سرورها اجازه می‌دهد تا منابع کمتری را برای تازه‌سازی حافظه‌ی نهان هدر دهند.

---

[۱] Recursive Querying
[۲] Caching
[۳] Time To Live یا **TTL**

انواع رکوردها

ما دو نوع رکورد DNS را تا کنون دیده‌ایم: AAAA که آدرس‌های IP را به دامنه‌ها پیوند می‌دهد و NS که به سرورهای نام قدرت کنترل بر دامنه‌ها می‌دهد. بسیاری از انواع رکوردهای دیگر وجود دارند. بیایید رایج ترین آن‌ها را ببینیم.

آدرس: هنگامی که DNS در سال ۱۹۸۳ اختراع شد، نوع رکورد A برای آدرس‌های IP استفاده شد. در سال ۲۰۱۲، IP ارتقاء یافت و آدرس‌ها چهار برابر شد. نوع رکورد AAAA این آدرس‌های جدید را برای IP قدیمی ذخیره می‌کند تا با ارتقای شبکه‌ها بدون تأثیر منفی کار کند. دامنه‌ها می‌توانند هر دو نوع رکورد را داشته باشند، به عنوان مثال:

```
one.one.one.one.        AAAA      2606:4700:4700::1111
one.one.one.one.           A      1.1.1.1
```

اگر یک اتصال قدیمی دارید که فقط از IPv4 پشتیبانی می‌کند، می‌توانید رکوردهای A را جستجو کنید و بسته‌های IPv4 را به جای بسته‌های IPv6 به آدرس بازیابی شده ارسال کنید.

دامنه‌های سطح بالا نمی‌توانند رکوردهای A یا AAAA داشته باشند. به همین دلیل است که هیچ وب‌سایتی به آدرس http://com/ یا http://google/ وجود ندارد. نام‌های تک برچسبی فقط برای آدرس‌دهی در شبکه‌های محلی استفاده می‌شوند. رایج‌ترین آنها localhost است و همیشه به رابط شبکه‌ی خود کامپیوتر اشاره می‌کند.

تبادل ایمیل: اگر می‌خواهید درباره‌ی این کتاب بازخوردی بدهید، می‌توانید یک ایمیل به hi@code.energy بفرستید (حتماً این کار را انجام دهید!). اما چگونه سیستم ایمیل شما می‌داند که ایمیل شما را به کجا ارسال کند تا به سرورهای ما برسد؟

مانند وب، ایمیل نیز به DNS وابسته است. یک رکورد MX میزبانی را مشخص می‌کند که مسئول دریافت ایمیل برای یک دامنه است. برای ارسال ایمیل به maria@icmc.usp.br، اولین قدم جستجوی رکوردهای MX برای icmc.usp.br است. بیایید با dig آن را بررسی کنیم:

```
dig  icmc.usp.br.  MX
```

هر رکورد MX شامل یک نام میزبان و یک شماره اولویت است:

```
icmc.usp.br.          MX    1    aspmx.l.google.com.
icmc.usp.br.          MX    5    alt1.aspmx.l.google.com.
icmc.usp.br.          MX    5    alt2.aspmx.l.google.com.
icmc.usp.br.          MX   10    alt3.aspmx.l.google.com.
```

این رکوردها نشان می‌دهند که مالک icmc.usp.br برای دریافت ایمیل google را از طرف خود انتخاب کرده است. برای ارسال ایمیل به icmc.usp.br می‌توانید به هر یک از این میزبان‌ها وصل شوید، اما باید از میزبانی با کمترین اولویت استفاده کنید.

همانند رکوردهای A و AAAA، دامنه‌های سطح بالا نمی‌توانند رکوردهای MX داشته باشند. به همین دلیل است که هیچ آدرس ایمیلی مانند ada@io یا larry@google وجود ندارد.

نام متعارف[۱]: در روزهای اولیه‌ی وب، مردم شروع به پیکره‌بندی سرورها در سازمان خود به منظور پردازش درخواست‌ها برای صفحات وب کردند. معمولاً نام چنین سروری "www" بود، بنابراین http://www.example.com/ به آدرس وب مورد انتظار برای example.com تبدیل شد. اخیراً، بسیاری قسمت "www" را از آدرس‌های وب خود حذف می‌کنند. یک رکورد ویژه‌ی CNAME به ما امکان می‌دهد زیردامنه‌ای را تعریف کنیم که صرفاً یک نام مستعار است:

```
www.code.energy.        CNAME        code.energy.
```

این رکورد به هر کسی که سعی در بازیابی رکوردی برای www.code.energy دارد می‌دهد دستور می‌دهد تا در عوض رکوردهای code.energy را در نظر بگیرد.

متن: پیوند متن دلخواه به یک دامنه با استفاده از رکوردهای TXT امکان‌پذیر است. این نوع رکورد اغلب برای اثبات مالکیت یک دامنه استفاده می‌شود، اما شما می‌توانید از آن برای هر چیز دیگری استفاده کنید. ما یک پیام ویژه برای شما در رکوردهای TXT مرتبط با enigma.code.energy گذاشته‌ایم. آیا می‌توانید این پیام را پیدا کنید؟ به یاد داشته باشید، می‌توانید از dig برای پرس‌وجو کردن هر نوع رکوردی که به هر دامنه‌ای مرتبط است استفاده کنید.

سرور نام دامنه‌ی معکوس

سیستم DNS بیشتر برای دریافت آدرس IP از یک نام دامنه‌ی خاص استفاده می‌شود. با این حال، این سیستم به صورت معکوس نیز کار می‌کند[۲]: می‌تواند مشخص کند که کدام دامنه با یک آدرس IP معین مرتبط است. روال کار این‌گونه است که آدرس IP را به عنوان یک زیر دامنه از دامنه‌ی سطح بالای ICANN یعنی arpa بیان کنید، مثلاً:

---

[۱] Canonical Name
[۲] Reverse DNS

```
icmc.usp.br.          AAAA   2001:12d0:2080::231:6
```

که تبدیل می‌شود به:

```
.0.8.2.0.d.2.1.1.0.0.2.ip6.arpa.          PTR   icmc.usp.br.
```

آدرس IP معکوس (۳۲ رقم هگز)

به لطف ICANN، یک زیردامنه‌ی منحصر به فرد arpa به ازای هر آدرس IP وجود دارد. می‌توان یک رکورد PTR برای آن زیردامنه تنظیم کرد تا آن را با نام دامنه مرتبط کند.

مدیریت بر arpa متعلق به IANA است. هنگامی که یک بلوک آدرس IP به یک سازمان داده می‌شود، سازمان همچنین اختیارات مربوط به زیردامنه‌های arpa را نیز دریافت می‌کند. هنگامی که به کامپیوتر یک آدرس IP و یک نام دامنه اختصاص می‌یابد، تمرین خوبی است که این زوج را در یک رکورد PTR با استفاده از زیردامنه‌ی arpa مربوط به آدرس IP ثبت کرد.

با استفاده از dig، می‌توانید رکوردهای PTR مربوط به هر آدرس IP را با استفاده از آرگـومان x- پرس‌وجو کنید. امتحان کنید:

```
dig  -x  2600::
```

سیستم DNS معکوس برای تأیید اینکه یک بسته‌ی IP از کجا آمده، مفید است. به عنوان مثال، فرض کنید یک بسته‌ی IP حاوی پیامی از apple.com دریافت می‌کنید. اگر DNS معکوس آدرس IP فرستنده، یک دامنه‌ی اپل نباشد، یک جای کار می‌لنگد!

دلیل این است کسانی که اختیار تنظیم رکوردهای DNS برای زیردامنه‌های arpa را دارند مراقب هستند. اکثر ISPها حتی به مشتریان خود اجازه نمی‌دهند تا رکوردهایی را در زیردامنه‌های arpa که با آدرس‌های IP آن‌ها مطابقت دارد تنظیم کنند.

ثبت دامنه

سیستمDNS فقط اعداد بزرگ را با نام‌ها جایگزین نمی‌کند، بلکه به افراد امکان می‌دهد تا شبکه‌های خود را مجدداً پیکره‌بندی کنند. کامپیوتری را در نظر بگیرید که میزبان یک وب سایت است. این کامپیوتر در صورت جابجایی، یک آدرسIP جدید دریافت می‌کند. رکوردهایDNS وب سایت را می‌توان به آدرس جدید به روز کرد تا بازدیدکنندگان وب سایت متوجه تغییر نشوند!

این کار نیاز به DNS جهانی دارد. داشتن دامنه برای هر کسی باید آسان، ایمن و مقرون به صرفه باشد. چندین قانون وجود دارد که این قانون را تضمین می‌کنند. ICANN با استفاده‌ی انحصاری از TLDها مشکلی ندارد: گوگل و آی‌بی‌ام هر طور که بخواهند از google و ibm استفاده می‌کنند. همچنین، ICANN به هر دولت ملی اجازه می‌دهد تا قوانینی را برای TLD کد کشور خود تنظیم کند. با این حال، ICANN قوانین سختگیرانه‌ای را بر TLDهای عمومی که برای عموم در نظر گرفته شده است، مانند com، org، rocks و energy اعمال می‌کند.

موسسه‌یICANN نهادی را که کنترل یک یا چند TLD را در اختیار دارد، رجیستری یا **مرکز اطلاعات شبکه**[۱] یا NIC می‌نامد. همانطور که دیدیم، رجیستری برای com شرکت Verisign است. موسسه‌ی ICANN رجیستری‌های مسئول TLDهای عمومی را ملزم به پذیرش درخواست همه‌ی افراد کرده است. همچنین ICANN تصریح می‌کند که یک رجیستری نمی‌تواند مستقیماً هیچ درخواست ثبت دامنه‌ای را دریافت کند.

در تلاش برای تقویت رقابت و به حداکثر رساندن دسترسی DNS، ICANN تصریح می‌کند که درخواست‌های ثبت دامنه برای TLDهای عمومی باید توسط شرکت‌هایی به نام **ثبت‌کننده**[۲] پردازش شوند. در سال ۲۰۲۰، بزرگترین ثبت‌کننده GoDaddy است که ده‌ها میلیون ثبت دامنه را پردازش کرده است.

ثبت‌کنندگان باید توسط ICANN تأیید شوند. به همین دلیل ICANN حداکثر و حداقل هزینه‌هایی را که رجیستری‌ها و ثبت‌کنندگان می‌توانند برای ثبت دامنه دریافت کنند، تعیین می‌کند. هر بار که دامنه‌ای ثبت می‌شود، بخشی از هزینه به ICANN پرداخت می‌شود[۳]. از نظر فنی، دامنه همچنان در مالکیت ICANN است و حداکثر برای مدت ۱۰ سال اجاره داده می‌شود. با این حال، از آنجایی که اجاره‌نامه قابل تمدید است، معمولاً مستاجر را مالک می‌دانیم

WHOIS: از همان ابتدا ICANN ثبت‌کنندگان را موظف کرد که نام و اطلاعات تماس صاحبان دامنه را در یک فهرست عمومی به نام WHOIS منتشر کنند. این فهرست از طریق یک پروتکل اینترنتی متفاوت و مستقل از DNS کار می‌کند. این فهرست را می‌توان با استفاده از برنامه‌ی whois در خط فرمان مشاهده کرد[۴]:

---

[۱] Network Information Center یا NIC

[۲] Registrar

[۳] در سال ۲۰۲۰، ICANN به ازای هر دامنه ۰.۱۸ دلار در هر سال دریافت می‌کند.

[۴] اگر به رابط خط فرمان دسترسی ندارید، از این سرویس استفاده کنید: http://code.energy/whois.

```
whois code.energy
```

به لطف پیشرفت روش‌ها و مقررات حفاظت از داده‌ها، ICANN اکنون به ثبت‌کنندگان اجازه می‌دهد تا هویت صاحبان دامنه را اصلاح کنند. در سال ۲۰۱۲، ICANN قول داد که WHOIS را دوباره بازسازی کند. اما در سال ۲۰۲۰، این روند همچنان ادامه دارد.

میزبانی WHOIS: از لحاظ نظری، هنگام ثبت دامنه، باید نام و آدرس IP سرورهای نام خود را ارائه دهید. این اطلاعات از ثبت‌کننده‌ی شما به رجیستری برای درج در سرورهای نام TLD ارسال می‌شود. با این حال، اکثر مردم حتی نمی‌دانند سرور نام چیست. برای آسان‌تر کردن کارها، بسیاری از ثبت‌کننده‌ها یک سرویس میزبانی DNS را ارائه می‌کنند: آن‌ها از سرورهای نام خود از طرف شما استفاده می‌کنند و به شما اجازه می‌دهند رکوردهای DNS خود را از طریق یک رابط وب مدیریت کنید.

بازار نام‌ها: ما در دوران رونق اقتصاد آنلاین زندگی می‌کنیم. نام‌های دامنه به عنوان املاک مجازی در نظر گرفته شده و گاهی اوقات به قیمت میلیون‌ها دلار فروخته می‌شوند. به عنوان مثال، در سال ۲۰۱۹، GoDaddy فروش voice.com را به قیمت ۳۰ میلیون دلار واسطه‌گری کرد. نام های دامنه com. به قدری ارزشمند هستند که تمام ترکیبات چهار حرفی ممکن گرفته شده‌اند.

اکنون که می‌دانید DNS چگونه کار می‌کند، به میزبان‌هایی که از نام دامنه‌ی آن‌ها استفاده می‌کنند اشاره می‌کنیم. هر زمان نام دامنه‌ای را می‌بینید که به عنوان آدرس یک میزبان استفاده می‌شود، به یاد داشته باشید که برای یافتن آدرس IP و ارسال بسته‌ی IP یک پرس‌وجوی DNS نیاز است.

۲-۲- زمان

هزاران سال پیش، انسان ها شروع به کشف این موضوع کردند که چگونه نجوم و مکانیک می‌توانند به تعیین کمیت زمان کمک کنند. تمدن‌های باستان تقویم‌های قمری و خورشیدی مختلفی را برای دنبال کردن فصول ایجاد کردند و دستگاه‌هایی مانند ساعت‌های آفتابی و ساعت‌های آبی را برای پیگیری زمان در روز یا شب اختراع کردند[۱].

با حرفه‌ای شدن لژیون‌های رومی در قرن اول قبل از میلاد، آن‌ها به طور فزاینده‌ای زمان را برای استفاده از پتانسیل کامل نظامی خود نگه می‌داشتند. زمان‌سنجی به آن‌ها اجازه می‌داد تا مانورهای دقیق را *هماهنگ* کرده و ضربات ویرانگری را به دشمنان خود وارد کنند. آن‌ها همچنین به طور منظم زمان

[۱] ساعت‌های آفتابی وقتی که در معرض نور خورشید قرار می‌گیرند، با انداختن سایه بر روی سطح مشخص شده زمان را نشان می‌دهند. ساعت‌های آبی زمان سپری شده را بر اساس جریان تدریجی آب از یک ظرف به ظرف دیگر ردیابی می‌کنند.

جمع‌آوری اطلاعات توسط پیشاهنگان و پیام‌رسان‌ها را ثبت می‌کردند تا بتوانند رویدادها را *به هم مرتبط* کنند و اطلاعاتی در مورد تاکتیک‌های دشمن به دست آورند.

امروزه ما به شدت به زمان‌سنجی تکیه می‌کنیم زیرا کامیون‌ها، قطارها، کشتی‌ها و هواپیماها باید برای حمل مقادیر غیرقابل اندازه‌ای کالا در سراسر جهان *هماهنگ* شوند. ما گاهی اوقات هر بار که یک بسته‌ی پستی دست‌به‌دست می‌شود، گزارش‌های مربوطه را نگه می‌داریم. اگر همه در مورد زمان به توافق برسند، می‌توانیم رویدادها را *به هم مرتبط* کنیم تا بتوانیم حوادث و عواقب آن‌ها را زیر نظر بگیریم و اگر افراد اشتباهی انجام دهند پاسخگو باشند.

به روشی مشابه، زمان‌سنجی به کامپیوترهای به هم متصل اجازه می‌دهد تا تیم‌های قدرتمندی را تشکیل دهند. اگر کامپیوترهای دور از هم بتوانند در مورد زمان به توافق برسند، می‌توانند اقدامات خود را با سرعتی *هماهنگ* کنند که انسان نمی‌تواند به آن دست یابد. به عنوان مثال، تراکنش‌های مالی بین طرفین در قاره‌های مختلف را می‌توان در عرض چند ثانیه تایید کرد.

کامپیوترها همچنین گزارش‌های مربوط به هر یک از اقدامات خود را با رکوردی از زمان دقیق انجام آن ذخیره می‌کنند که به آن **برچسب زمانی**[۱] گفته می‌شود. توافق در مورد زمان به کامپیوترهای متصل به هم این امکان را میدهد که اقدامات، سوابق و پیام‌های گذشته را به ترتیب زمانی ترتیب دهند و بنابراین رویدادها را *به هم مرتبط* کنند. این امر بیشتر به برنامه‌نویسان کمک می‌کند تا اشکالات را پیدا کنند، اما می‌تواند به عنوان مثال به متخصصان امنیتی نیز کمک کند تا فعالیت‌های مخرب را شناسایی و ردیابی کنند.

به اشتراک‌گذاری زمان امکان هماهنگی و همبستگی رویداد را در مقیاس‌های جهانی فراهم می‌کند. حال این سؤال پیش‌می‌آید: انسان‌ها و کامپیوترهای به هم پیوسته چگونه حتی در مورد این که الان ساعت چند است، توافق می‌کنند؟

ساعت هماهنگ جهانی

در دوران عصر اکتشاف، ملوانان اروپایی تکنیک‌های ناوبری خود را بر اساس مشاهدات آسمانی و زمان‌سنجی تکمیل کردند. در قرن هجدهم، آن‌ها توانستند با استفاده از جداول منتشر شده توسط رصدخانه‌ی رویال گرینویچ در لندن[۲]، مکان خود را در دریا دقیقاً مشخص کنند. این فرایند تا زمانی درست کار می‌کرد که آن‌ها دقیقاً می‌دانستند ساعت چند ساعت است، بنابراین کشتی‌ها ساعت‌های

---

[۱] ع Timestamp

[۲] Royal Greenwich Observatory in London

مکانیکی[۱] خود را با گرینویچ هماهنگ می‌کردند. همانطور که این منابع قابل اعتماد زمان در سراسر جهان در حال حرکت در دریاها بودند، زمان گرینویچ به تدریج به استاندارد جهانی زمان تبدیل شد.

چند قرن بعد، برای دقت بیشتر، از ساعت‌های مکانیکی به ساعت‌های کوارتز و ساعت‌های اتمی روی آوردیم[۲]. امروزه صدها ساعت اتمی زمان جهانی را نگهداری می‌کنند. ساعت هماهنگ جهانی[۳] یا UTC، میانگین اندازه‌گیری‌های این ساعت‌ها و در حال حاضر بهترین استاندارد ما برای زمان است[۴]. به لطف پیشرفت‌های نجومی و علوم سیاره‌ای، حتی می‌توانیم دلیل کند شدن چرخش زمین را نیز توضیح دهیم: ما گاهی اوقات ثانیه‌های کبیسه را به UTC اضافه می‌کنیم تا با زمان خورشیدی در لندن هماهنگ بماند.

در سراسر جهان، زمان رسمی با اضافه کردن یک مقدار اختلاف به UTC به دست می‌آید. اختلاف زمانی می‌تواند به صورت فصلی تغییر کند. به عنوان مثال، انگلستان در نیمی از سال به دلیل ساعت تابستانی یک ساعت جلوتر از UTC است. توکیو در تمام طول سال ۹ ساعت جلوتر از UTC است.

مناطق زمانی: نواحی که از یک ساعت رسمی با اختلاف یکسان از UTC تبعیت می‌کنند، یک منطقه‌ی زمانی تشکیل می‌دهند. موسسه‌ی IANA مناطق زمانی در حال تغییر و اختلاف آن‌ها را پیگیری و این داده‌ها را در پایگاه‌داده‌ی `tz` منتشر می‌کند. به عنوان مثال، بخش بزرگی از شرق ایالات متحده، از میامی تا نیویورک، از یک اختلاف UTC تبعیت می‌کنند. در پایگاه‌داده‌ی `tz`، این منطقه‌ی زمانی دارای کد `America/New_York` است. بریتانیا به طور کامل در منطقه‌ی زمانی `Europe/London` و ژاپن در `Asia/Tokyo` قرار دارد.

کامپیوترها زمان را در UTC ردیابی می‌کنند، اما معمولاً آن را بر اساس منطقه‌ی زمانی خود نمایش می‌دهند. برای مثال، ایمیلی که از نیویورک به توکیو ارسال می‌شود، دارای برچسب زمانی ایجاد در UTC

---

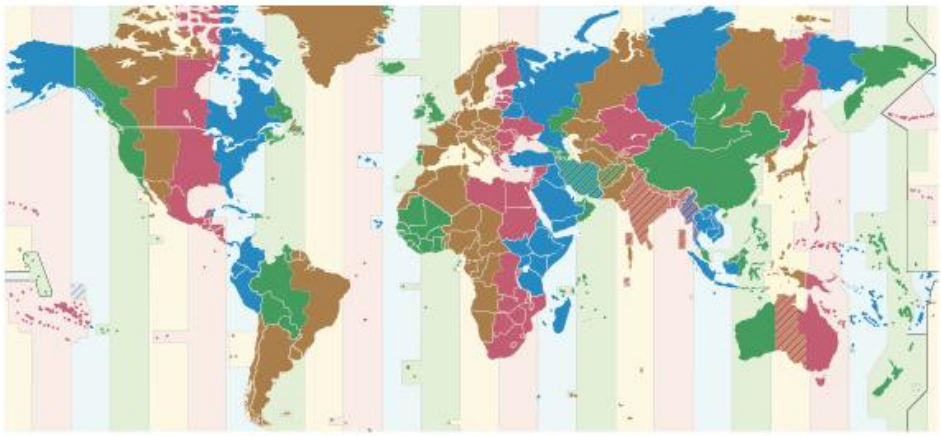
[۱] ساعت‌های مکانیکی معمولاً انرژی را با وزنه یا فنر ذخیره می‌کنند و مکانیزم هوشمندانه‌ای دارند که اجازه می‌دهد انرژی در فواصل زمانی ثابت خارج شود.

[۲] ساعت های کوارتز از وسایل الکترونیکی هوشمند و یک کریستال خاص برای ایجاد ارتعاش با فرکانس دقیق و از پیش تعیین شده استفاده می‌کنند. ساعت‌های اتمی شبیه به ساعت‌های کوارتز هستند، اما پیوسته ارتعاشات را با پدیده‌های فیزیک اتمی همگام می‌کنند.

[۳] **C**oordinated **U**niversal **T**ime یا **CUT**

[۴] کشورهای انگلیسی زبان در ابتدا نام اختصاری CUT را پیشنهاد کردند، در حالی که کشورهای فرانسوی زبان TUC را برای Temps Universel Coordonné ترجیح دادند. عبارت UTC که به صورت بین‌المللی به رسمیت شناخته شده یک سازش بین این دو است.

است. این برچسب زمانی برای فرستنده و گیرنده، با توجه به پیکره‌بندی منطقه‌ی زمانی کامپیوترهایشان، متفاوت نمایش داده می‌شود. از این به بعد، وقتی به زمان اشاره می‌کنیم، منظورمان UTC است.



شکل ۲-۱: طرح کلی مناطق زمانی جهان تا سال ۲۰۲۰.

پروتکل زمان شبکه

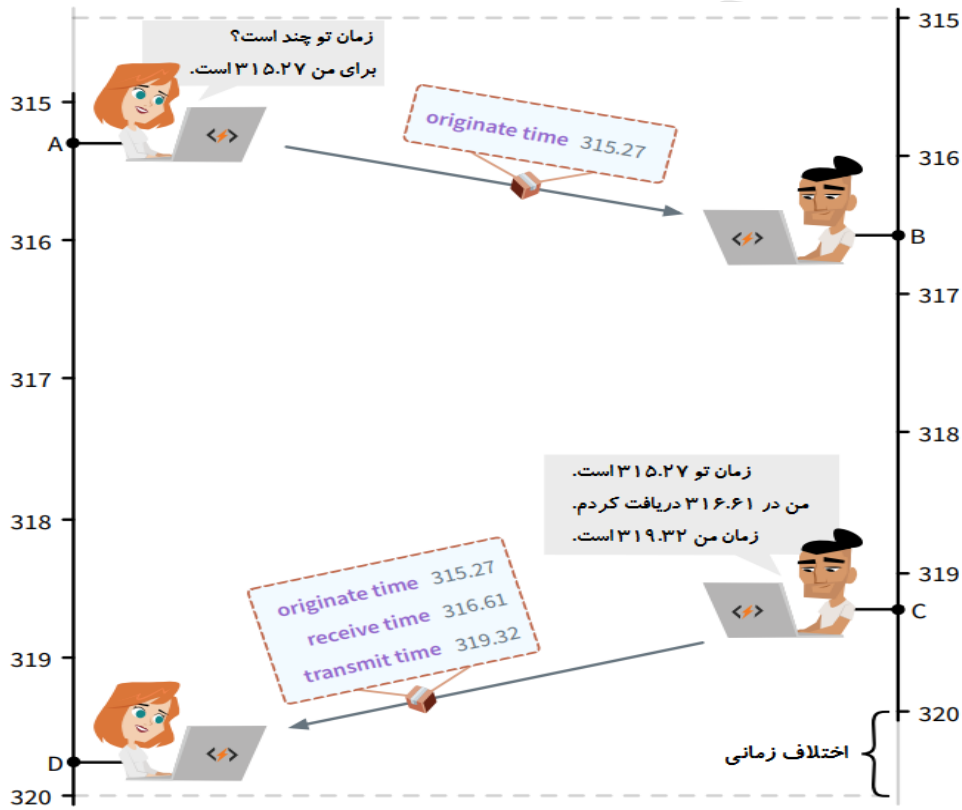
اگر ساعت انسان‌ها در حد ثانیه دقیق باشد، می‌توانند مانند انگلیسی‌ها وقت‌شناس باشند. اما کامپیوترها سریعتر کار می‌کنند و ساعت آن‌ها به دقت بیشتری نیاز دارد. به عنوان مثال، گروهی از کامپیوترها می‌توانند صدها رکورد را در هر ثانیه ثبت کنند. برای اینکه این رکوردها را به ترتیب زمانی نگه داریم، باید ساعت‌ها را تا میلی‌ثانیه همگام کنیم.

تنظیم ساعت در چنین مقیاس‌های زمانی کار آسانی نیست. یک رویکرد ساده‌لوحانه ارسال پیامی در اینترنت و درخواست زمان از کسی که آن را می‌داند و تنظیم ساعت بر اساس برچسب‌زمانی دریافت شده در پاسخ است. با این حال، این رویکرد جواب نمی‌دهد: بسته های IP برای مدت زمان نامشخصی منتقل می‌شوند. زمانی که از یک کامپیوتر از راه دور بسته‌ای را دریافت می‌کنیم، ناگزیر با تاخیر به ما رسیده است.

با این حال، با برخی داده‌های اضافی، می‌توانیم تخمین بزنیم که چقدر طول می‌کشد یک بسته از طریق اینترنت به مقصد برسد. سپس می‌توان این مدت زمان را قبل از این‌که از آن برای تنظیم ساعت خود

استفاده کنیم، به برچسب زمانی مشخص شده اضافه کرد. **پروتکل زمان شبکه**[۱] یا NTP استاندارد پرکاربردی برای همگام‌سازی ساعت‌ها با استفاده از این اصل است.

برچسب‌های زمانی NTP بر حسب ثانیه از نیمه شب ۱ ژانویه‌ی ۱۹۰۰ بیان می‌شوند. برای مثال، نیمه شب ۱ ژانویه‌ی سال ۲۰۰۰ دارای برچسب زمانی ۳,۱۵۵,۶۷۳,۶۰۰ است. این عدد می‌تواند یک بخش اعشاری نیز داشته باشد، بنابراین می‌تواند زمان را با دقت بسیار خوبی ثبت کند. سرورهای NTP پیام‌های دریافتی را در پورت شماره‌ی ۱۲۳ پروتکل UDP شنود می‌کنند. تبادل پیام NTP به صورت زیر عمل می‌کند:



شکل ۲-۲: یک کلاینت با استفاده از NTP از سرور زمان را می‌پرسد. برای سادگی، بیایید وانمود کنیم که در سال ۱۹۰۰ هستیم، به طوری که برچسب‌های زمانی اعداد کوچکی هستند. زمان نگهداری شده توسط هر یک از کامپیوترها از بالا به پایین اجرا می‌شود.

مبادله زمانی شروع می‌شود که کلاینت درخواست زمان خود را ارسال کند. این پیام حاوی زمان ارسال آن، مطابق با ساعت کلاینت (زمان A) است. سرور زمان رسیدن پیام را با توجه به ساعت سرور

[۱] Network Time Protocol یا NTP

(زمان B) ثبت می‌کند. هنگامی که سرور پاسخ می‌دهد، یک پیام حاوی سه زمان ارسال می‌کند: زمان A، زمان B و زمان پاسخ آن (C). کلاینت این زمان‌ها را دریافت و تأیید می‌کند که زمان A همان است که قبلاً ارسال کرده بود. کلاینت همچنین زمان دریافت را مطابق با ساعت خودش (زمان D) یادداشت می‌کند.

در شکل ۲.۲، زمان D را ۳۱۹.۴۲ فرض می‌کنیم. با دانستن زمان‌های A، B، C و D، مشتری می‌تواند زمان سفر بسته‌های IP از طریق اینترنت را محاسبه کند:

زمان پردازش سرور − زمان رفت و برگشت کلاینت = زمان جابجایی

$$= D - A - (C - B) = 319.42 - 315.27 - (319.32 - 316.61)$$
$$= 4.15 - 2.71 = 1.44.$$

حدوداً ۱.۴۴ ثانیه طول کشید تا پیام‌ها از کلاینت به سرور بروند و برگردند. ما تخمین می‌زنیم که نیمی از این زمان طول کشید تا بسته از سرور به کلاینت برسد: ۰.۷۲ ثانیه. کلاینت باید ۰.۷۲ ثانیه پس از زمان C، در ۳۲۰.۰۴ = ۰.۷۲ + ۳۱۹.۳۲ پاسخ را دریافت کرده باشد. با این حال، کلاینت پیام را زمانی دریافت کرد که زمان آن ۳۱۹.۴۲ بود. این بدان معناست که ساعت کلاینت در مقایسه با سرور حدود ۰.۶۲ = ۳۱۹.۴۲ − ۳۲۰.۰۴ ثانیه دیرتر اجرا می‌شود.

پروتکل NTP محافظه‌کارانه است: کلاینت‌ها را موظف می‌کند ساعت خود را تنها با نصف اختلاف زمانی تخمین‌زده‌شده تنظیم کنند. بنابراین کلاینت ما ساعت خود را ۰.۳۱ ثانیه جلو می‌برد. هر ده دقیقه، کلاینت ما یک درخواست جدید ارسال و ساعت خود را بر اساس آن تنظیم می‌کند. اگر زمان جابجایی بسته بین کلاینت و سرور در هر دو جهت یکسان باشد، ساعت کلاینت در نهایت با ساعت سرور همگام خواهد شد. در پیوندهای اینترنتی معمولی، NTP ساعت‌ها را با خطاهایی در حدود ده میلی‌ثانیه همگام‌سازی می‌کند.

سرورهای زمان

ممکن است تعجب کنید که سرورهای NTP زمان خود را از کجا می‌گیرند. اکثر سرورهای NTP در واقع با سایر سرورهای NTP در اینترنت همگام می‌شوند: هیچ مشکلی در اجرای برنامه‌های NTP سرور و کلاینت به طور همزمان وجود ندارد.

با این حال، این امر می‌تواند منجر به بروز مشکلاتی شود. فرض کنید ایدا زمان خود را از اندرو می‌گیرد و اندرو زمان خود را از چارلز می‌گیرد و چارلز زمان خود را از ایدا می‌گیرد. هیچ یک از این سه

نفر منبع زمانی کالیبره شده با UTC ندارند. حتی اگر تمام ساعت‌های آن‌ها در ابتدا درست باشد، عدم دقت در طول زمان ایجاد می‌شود و باعث می‌گردد که از زمان جهانی دور شوند.

برای جلوگیری از این مشکل، سرورهای NTP در سلسله‌مراتبی به نام **طبقه‌های ساعت**[۱] سازماندهی می‌شوند. به سروری که مستقیماً به یک منبع زمانی قابل اعتماد، مانند ساعت اتمی یا گیرنده‌ی GPS مرتبط است، طبقه‌ی[۲] ۱ گفته می‌شود. به این ترتیب، کامپیوترهایی که با سرور طبقه‌ی ۱ همگام می‌شوند، طبقه‌ی ۲ هستند. اگر کامپیوتری زمان را از سرور طبقه‌ی ۲ دریافت کند، به طبقه‌ی ۳ تبدیل می‌شود و الی آخر. بر طبق قانون NTP سرورها باید طبقه‌ی خود را هنگام پاسخگویی به پرس‌وجوهای زمانی گزارش کنند و پایین‌ترین طبقه‌ی سلسله‌مراتب طبقه‌ی ۱۵ است.

این روال به کامپیوترها اجازه می‌دهد از هماهنگ‌سازی زمان به صورت حلقه‌ای اجتناب کنند: اگر چارلز طبقه‌ی ۲ باشد و زمان را به اندرو بگوید، اندرو تبدیل به طبقه‌ی ۳ می‌شود. اگر اندرو آن را به ایدا بگوید، او تبدیل به طبقه‌ی ۴ می‌شود. حتی اگر روزی ایدا زمان را به چارلز بگوید، چارلز آن را نادیده می‌گیرد، زیرا او طبقه‌ی ۲ است و ایدا طبقه‌ی ۴ است. چارلز باید یک سرور طبقه‌ی ۱ برای همگام‌سازی پیدا کند.

سرورهای NTP عمومی بسیاری وجود دارند که به پرس‌وجوهای هر کسی پاسخ می‌دهند. کامپیوترهای اپل برای همگام‌سازی ساعت خود با time.apple.com که یک سرور طبقه‌ی ۲ است، از قبل پیکره‌بندی شده‌اند. دولت‌ها و سازمان‌های بزرگ اغلب سرورهای عمومی NTP را ارائه می‌دهند. به عنوان مثال، دولت ایالات متحده time.nist.gov، یک سرور عمومی طبقه‌ی ۱، را اداره می‌کند.

پروتکل NTP در سال ۱۹۸۵ ایجاد شد و یکی از قدیمی‌ترین پروتکل‌های اینترنت است. از آن‌جایی که تقریباً هر کامپیوتری یک کلاینت NTP را اجرا می‌کند، مردم اغلب این را بدیهی فرض می‌کنند که کامپیوترهای آن‌ها به طور طبیعی زمان را بدانند. می‌توانید ساعت خود را در http://time.gov/ بررسی کنید. البته احتمال دارد که این سیستم برای کمتر از یک ثانیه در دسترس نباشد. در ادامه، بیایید یک پروتکل اینترنتی را بررسی کنیم که حتی از NTP نیز قدیمی‌تر است.

---

[۱] Clock Strata
[۲] Stratum

۳-۲- دسترسی

در طی قرن نوزدهم، اپراتورهای انسانی مجبور بودند سیگنال‌های الکتریکی تلگراف را به زبان طبیعی رمزگشایی کنند و بالعکس. در آغاز قرن جدید، کار آن‌ها توسط دستگاه جدیدی به نام **تله‌تایپ**[۱] آسان شد. این ماشین‌ها به سیم‌های تلگراف متصل و یک ماشین تحریر مکانیکی را در خود جای داده بودند که می‌توانست کاراکترهای دریافتی تلگراف را به‌طور خودکار چاپ کند.

دستگاه‌های تله‌تایپ دارای صفحه کلید نیز بود، بنابراین کاراکترهایی که توسط اپراتور آن‌ها تایپ می‌شد به طور خودکار از طریق سیم تلگراف می‌شد. این به افراد اجازه می‌داد تا در فواصل طولانی گفتگو کنند: پیام‌هایی که در یک دستگاه تایپ می‌شوند توسط دستگاه دوم در طرف دیگر خط چاپ می‌شوند.

ترمینال‌ها

هنگامی که کامپیوترهای الکترونیکی برای اولین بار در دهه ۱۹۵۰ تجاری شدند، هیچ صفحه‌ی نمایشی نداشتند. مردم برای دریافت بازخورد از ماشین‌های جدیدشان، مجبور بودند سیم‌های ورودی/خروجی خود را به تله‌تایپ‌ها متصل کنند. به این ترتیب، آن‌ها می‌توانستند داده‌ها و دستورالعمل‌ها را با تایپ کردن به کامپیوتر برسانند و پاسخ کامپیوتر بلافاصله چاپ شود. این تله‌تایپ‌ها که به ما اجازه می‌دهند از طریق متن با کامپیوترها ارتباط برقرار کنیم، **ترمیال**[۲] یا **پایانه** نامیده می‌شوند.

برنامه‌ی کامپیوتری که اطلاعات و جریان دستورالعمل‌ها به ترمینال و از آن را پردازش می‌کند، **شل**[۳] یا **پوسته** نامیده می‌شود. شل به منظور برقراری ارتباط با انسان‌ها، از یک رابط خط فرمان[۴] یا CLI استفاده می‌کند. همزمان با تایپ دستورات در CLI توسط کاربر، پوسته از ترمینال می‌خواهد هر کاراکتر را چاپ کند. با انجام این کار، کاربر همیشه می‌تواند آن‌چه را که در حال تایپ کردن است، بلافاصله مشاهده کند. کاربر می‌تواند یک **کاراکتر غیرچاپی**[۵] یا NPC به نام **سرخط** را وارد کند تا شل کل خط را به عنوان یک فرمان اجرا، در صورت درخواست کاربر خروجی را بر روی ترمینال چاپ کرده و برای دستور بعدی منتظر بماند.

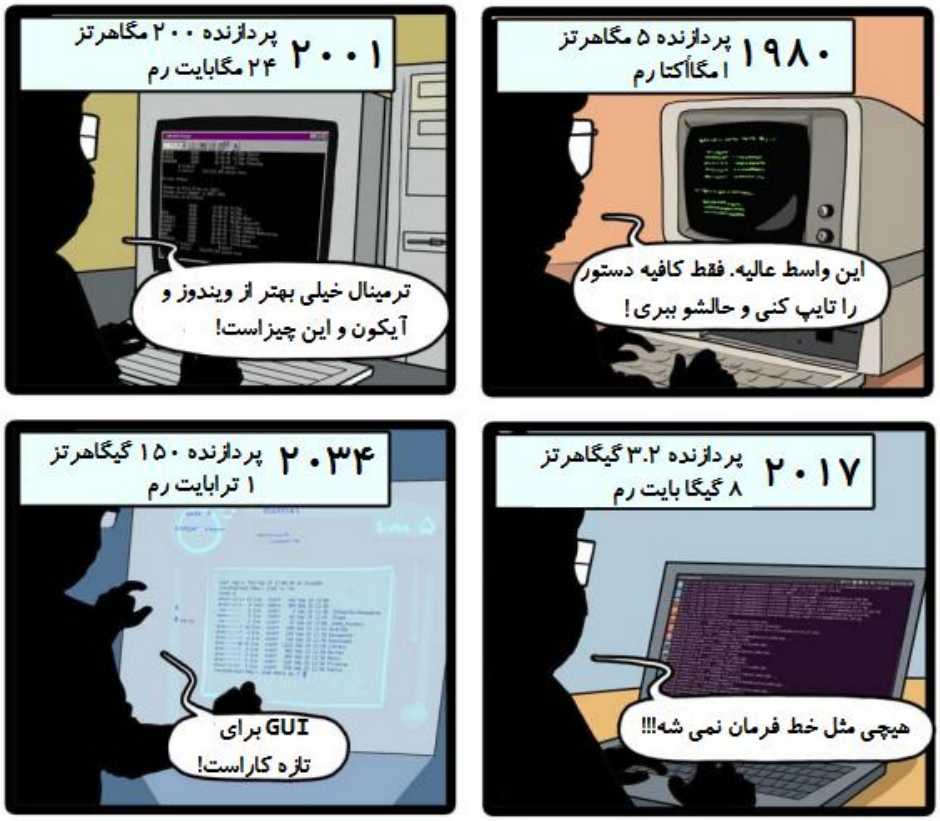
در دهه‌ی ۱۹۶۰، **تله‌تایپ‌های شیشه‌ای** ظهور کردند. آن‌ها دقیقاً مانند تله‌تایپ‌های قدیمی کار می‌کردند، با این تفاوت که کاراکترها را به جای چاپ روی کاغذ، روی صفحه نمایش می‌دادند. به همین دلیل است که تا به امروز، نمایش متن در صفحه‌ی ترمینال به عنوان چاپ شناخته می‌شود!

---

۱ Teletype
۲ Terminal
۳ Shell
۴ **C**ommand-**L**ine **I**nterface یا **CLI**
۵ **N**on-Printing **C**haracter یا **NPC**

صفحه‌ها مجاز به ایجاد **مکان‌نما**[۱] هستند؛ یک نشانگر متحرک که مکان تعامل بین شل و صفحه را مشخص می‌کند. کاربران با استفاده از کاراکترهای غیرچاپی می‌توانند از شل بخواهند مکان‌نما را به اطراف حرکت دهد. مهم‌تر از همه، مکان‌نماها اجازه می‌دهند متن چاپ شده قبلی جایگزین یا حذف شود و به این ترتیب تعامل با کامپیوتر کارآمدتر می‌شود. این امر تغییر زیادی در توسعه‌ی برنامه‌های کامپیوتری با واسط‌های تعاملی مبتنی بر متن ایجاد کرد.

مانیتورهای ویدیویی در اواخر دهه‌ی ۱۹۶۰ معرفی شدند و اولین کامپیوتر تجاری با **واسط کاربری گرافیکی**[۲] یا GUI در سال ۱۹۷۹ ظاهر شد. امروزه، تقریباً تمام کامپیوترهای شخصی مانند لپ‌تاپ و تلفن‌ها یک شل را اجرا می‌کنند که دارای واسط کاربری گرافیکی است: ما با آیکون‌ها و پنجره‌ها با استفاده از موس، تاچ‌پد وصفحه‌ی نمایش لمسی تعامل داریم.



شکل ۲-۳: «خط فرمان»، دریافت شده از http://commitstrip.com[۳].

---

[۱] Cursor

[۲] Graphical User Interface یا GUI

[۳] در اینجا باید توضیح داده شود که اُکتا واحدی قدیمی و پیش از بایت برای اندازه‌گیری حافظه بوده که هم‌اکنون منسوخ شده است. هر اُکتا نیز همانند بایت برابر با ۸ بیت است. م.

امروزه هم‌چنان بسیاری از متخصصان فناوری اطلاعات به تعامل با کامپیوترها از طریق CLI ادامه می‌دهند. این امر می‌تواند بنا به ضرورت، مثلاً برای راه‌اندازی سرورهای شرکتی بدون واسط کاربری گرافیکی، و همچنین به دلیل علاقه‌ی شخصی باشد. بسیاری از طرفداران فناوری از واسط کاربری گرافیکی خود فقط برای باز کردن یک **برابرساز** یا **امولاتور ترمینال**[۱] استفاده می‌کنند: برنامه‌ای که به شل CLI متصل شده و صفحه‌ی نمایش و صفحه‌کلید را به یک تله‌تایپ شیشه‌ای تبدیل می‌کند.

امولاتورهای ترمینال رابط‌های قدرتمندی هستند[۲]. از طریق CLI، می‌توان تقریباً هر کار مبتنی بر متن، مانند سازماندهی پوشه‌ها، ارسال ایمیل، نوشتن کد و کامپایل نرم افزار را انجام داد. ما نیز واقعاً این کتاب را بر روی امولاتور ترمینال نوشتیم و ویرایش کردیم!

```
ada@lovelylaptop:~$ dig babbage.code.energy. AAAA

; <<>> DiG 9.10.6 <<>> babbage.code.energy. AAAA
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35327
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;babbage.code.energy.           IN      AAAA

;; ANSWER SECTION:
babbage.code.energy.    516505  IN      AAAA    2620:0:862:ed1a::1

;; Query time: 24 msec
;; SERVER: 10.1.0.1#53(10.1.0.1)
;; WHEN: Wed Mar 03 09:53:40 CET 2021
;; MSG SIZE  rcvd: 76

ada@lovelylaptop:~$ _
```

شکل ۲-۴: اجرای dig بر روی یک امولاتور ترمینال لینوکس یا MacOS.

تلنت

در دهه‌های ۱۹۵۰ و ۱۹۶۰، یک ترمینال باید مستقیماً به کامپیوتر متصل می‌شد تا با شل آن ارتباط برقرار کند. با این حال، این امر در سال ۱۹۶۹ تغییر کرد، زمانی که مهندسان راهی برای اتصال ترمینال‌ها به هر کامپیوتری در یک شبکه ابداع کردند. آن‌ها آن را **شبکه‌ی تله‌تایپ** یا **تلنت**[۳] نامیدند.

---

[۱] Terminal Emulator

[۲] برخی از علاقمندان دوآتشه‌ی فناوری اطلاعات این موضوع را با نمایش جنگ ستارگان بر روی یک ترمینال در http://code.energy/terminal-movie به شما نشان داده‌اند.

[۳] **Tel**etype **Net**work یا **TelNet**

امولاتورهای ترمینال می‌توانند از تلنت برای دسترسی به شل کامپیوترهای دور از طریق اینترنت استفاده کنند. خط فرمان شل یک کامپیوتر دور باید بسیار قابل اعتماد باشد زیرا کاراکترهای نامنظم یا گم شده می‌تواند باعث اجرای دستورات اشتباه و بالقوه مخرب شوند. به همین دلیل، پروتکل لایه‌ی انتقال ترجیحی برای تلنت، TCP است تا UDP. سرورهای تلنت معمولاً انتظار اتصالات بر روی پورت ۲۱ پروتکل TCP را دارند.

هنگامی که یک اتصال TCP در پورت ۲۱ برقرار می‌شود، سرور تلنت یک شل را راه‌اندازی کرده و تمام داده‌های دریافتی از اتصال را به شل ارسال می‌کند. خروجی شل برای کلاینت ارسال می‌شود. در سمت کلاینت، تمام کاراکترهای تایپ شده توسط کاربر به صورت بلادرنگ، کاراکتر به کاراکتر به سرور منتقل می‌شوند و هنگامی که کلاینت یک یا چند کاراکتر دریافت می‌کند، آن‌ها را در امولاتور ترمینال خود نمایش می‌دهد.

معمولاً پس از برقراری اتصال تلنت، شل کامپیوتر دور از شما نام کاربری و رمز عبور را برای احراز هویت می‌خواهد. پس از ورود به سیستم، کلاینت تلنت به عنوان ترمینالی که مستقیماً به سرور متصل می‌شود، کار می‌کند. اگر کامپیوتری یک سرور شبکه‌ی راه دور را اجرا کند، هر کامپیوتری که به اینترنت متصل است می‌تواند به عنوان ترمینال آن سرور عمل کند[1]. افراد متخصص می‌توانند با یک کامپیوتر در قاره‌ای دیگر طوری کار کنند که گویی دقیقاً در کنار آن با صفحه‌ی نمایش و صفحه‌کلید نشسته‌اند!

پروتکل‌های ارتباطی مبتنی بر اینترنت منحصراً توسط افراد متخصص استفاده نمی‌شوند. بیایید اکنون نگاهی به محبوب‌ترین برنامه‌های کاربردی اینترنت بیندازیم و با شناخته‌شده‌ترین روش مبادله‌ی پیام‌های نوشتاری شروع می‌کنیم.

۲-۴- میل

در دهه‌ی ۱۹۶۰، تعداد کمی کامپیوتر موجود توسط بسیاری از کاربران به اشتراک گذاشته می‌شدند. در آن زمان، برنامه‌های «میل» قبلاً محبوب بودند، اما آن‌ها فقط به کاربر اجازه می‌دادند برای همتای خود که روی همان کامپیوتر کار می‌کرد، پیام بگذارد. این برنامه‌ها فقط متن را به فایل‌هایی که نقش صندوق پستی داشتند، اضافه می‌کردند. یک فایل صندوق پستی به هر کاربر نسبت داده می‌شد. هنگامی که

[1] قرار گرفتن گسترده در اینترنت باعث خطرات امنیتی می‌شود و از این رو تلنت ناامن تلقی می‌گرد. امروزه از پروتکل مشابهی به نام شل یا پوسته‌ی امن یا SSH (Secure Shel) استفاده می‌کنیم. تجربه‌ی کاربر نهایی تقریباً شبیه تلنت است، اما از رمزنگاری برای جلوگیری از حملات هک استفاده می‌کند. ما آن را در فصل بعد توضیح می‌دهیم، پس منتظر بمانید!

کاربران وارد کامپیوتر خود می‌شدند، در صورتی که صندوق پستی آن‌ها حاوی پیام‌های جدیدی بود، به ایشان اطلاع داده می‌شد.

در طی دهه‌ی ۱۹۷۰، شبکه‌های کامپیوتری رشد کردند و فناوری دسترسی به فایل‌ها در کامپیوترهای دور ایجاد شد. برنامه‌های میل برای افزودن متن به صندوق‌های پستی کامپیوترهای دور تکامل یافتند. این امر به کاربران کامپیوترهای دور اجازه می‌داد تا پیام‌ها را به گونه‌ای مبادله کنند که گویی در همان ماشین هستند. با گذشت زمان، برنامه‌نویسان کارایی این سیستم‌ها را با اتخاذ تدریجی یک قالب پیام معمول بهبود بخشیدند که به شکل زیر بود:

```
From: White at SRI-ARC
Date: 24 JUL 1973 1527-PDT
Subject: Multi-Site Journal Meeting Announcement
NIC: 17996

At 10 AM Wednesday 25-JULY there will be a meeting to
discuss a Multi-Site Journal in the context of the
Utility. Y'all be here.
```

پیام با **هدرها** یا **سرصفحه**[۱] شروع می‌شود که حاوی اطلاعات کلی درباره‌ی پیام مانند فرستنده، گیرنده و موضوع آن است. هر هدر در یک خط جا می‌گیرد و علامت دونقطه نام هدر را از مقدار آن جدا می‌کند. بعد از هدرها، یک خط خالی شروع متن پیام را نشان می‌دهد. این پیام‌ها به‌طور گسترده توسط متخصصان کامپیوتر مورد استفاده قرار می‌گرفتند به همین دلیل آن‌ها به آن **ایمیل**[۲] گفتند. تا به امروز، ایمیل‌هایی که ما رد و بدل می‌کنیم همچنان از این قالب پیروی می‌کنند!

در سال ۱۹۷۳، می‌شد به ایمیل فوق با الحاق یک پیام جدید به فایل صندوق پستی White در کامپیوتر SRI-ARC پاسخ داد. در آن زمان، TCP/IP و DNS وجود نداشتند و کامپیوترها با چنین نام‌هایی نامیده می‌شدند. بعدها، @ جای at را گرفت و آن را منسوخ کرد، بنابراین صندوق پستی در مثال فوق white@SRI-ARC نوشته می‌شود. محض اطلاع باید بدانید، SRI مخفف موسسه‌ی تحقیقاتی استنفورد[۳]، یکی از سازمان‌هایی است که به ایجاد و استانداردسازی این قالب پیام کمک کرد.

سرورهای میل

---

[۱] Header

[۲] Electronic **Mail** یا **EMail**

[۳] Stanford Research Institute

کامپیوترها به تدریج ارزان‌تر شدند و در دهه‌ی ۱۹۸۰، کار کردن بر روی چندین ماشین غیرمعمول نبود. با این حال، اگر هر فرد یک آدرس صندوق پستی اصلی در یک مکان واحد داشته باشد، دسترسی به افراد از طریق ایمیل در مقیاس‌های بزرگ آسان‌تر خواهد بود. بسیاری از گروه‌های کاربران تصمیم گرفتند صندوق پستی اصلی خود را روی یک میزبان قرار دهند، گویی که همه هنوز روی یک کامپیوتر کار می‌نند. سازمان‌ها معمولاً با تعیین یکی از کامپیوترهای خود به عنوان سرور میل، این کار را تسهیل می‌کنند.

در آن زمان، دانشجویان دانشگاه و مهندسان فناوری بدون در نظر گرفتن اینکه از کدام کامپیوتر استفاده می‌کردند، یک حساب در سرور ایمیل سازمان خود داشتند. آدرس ایمیل اصلی همه به این سرور ایمیل اشاره می‌کند. افراد از کامپیوترهای مختلف خود از تلنت برای ورود به سرور ایمیل و خواندن ایمیل‌های ذخیره شده در فایل‌های صندوق پستی فردی خود استفاده می‌کنند.

در سال ۱۹۸۵، سیستم DNS راه اندازی شد و سیستم ایمیل را بسیار بهبود بخشید. سازمان‌ها شروع به استفاده از نام‌های دامنه و ایجاد رکوردهای MX برای اعلام عمومی سرورهای ایمیل خود کردند. نام دامنه به طرح آدرس‌دهی استاندارد برای صندوق‌های پستی تبدیل شد. به عنوان مثال، صندوق پستی با نام White در سرور ایمیل SRI اکنون با آدرس white@sri.com نشان داده می‌شود.

وقتی این آدرس ایمیل داده می‌شود، هر کسی می‌تواند از DNS برای کشف سرور ایمیلی که فایل صندوق پستی آن را ذخیره می‌کند کمک گرفته و بسته‌های IP را به آن کامپیوتر ارسال کند. با این حال، برای ذخیره کردن یک پیام جدید در سرور یک نفر دیگر، ابتدا باید به فایل‌های صندوق پستی آن اجازه دسترسی داده می‌شد. این کار دست و پا گیر بود، بنابراین مهندسان شروع به اختراع روش‌هایی برای دریافت ایمیل‌ها بدون دستکاری خارجی هیچ فایلی بر روی سرور دریافت‌کننده، کردند.

این پدیده، ظهور پروتکل‌های ایمیل بود. همانطور که NTP چارچوبی را برای برنامه‌ها به منظور تبادل زمان فراهم می‌کند، پروتکل‌های میل به برنامه‌ها امکان تبادل ایمیل را می‌دهند. پروتکل‌های مختلفی توسعه یافتند و خیل زود هر سرور ایمیل یکی از آن‌ها را رعایت کرد. اکنون می‌توان ایمیل‌ها را تقریباً به هر مکانی که بسته‌های IP می‌توانند برسند، ارسال کرد.

چون هر سازمانی که به اینترنت متصل است می‌تواند از ایمیل استفاده کند، لذا این پدیده خیلی سریع مشهور شد. چیزی که با یادداشت‌های ساده‌ای که روی میز یک دوست باقی می‌ماند شروع شد، به تدریج به اندازه‌ی مکاتبات مکتوب سنتی اهمیت پیدا کرد.

در دهه‌ی ۱۹۹۰، ارائه دهندگان خدمات اینترنت شروع به وارد کردن اینترنت به خانه‌های مردم کردند. شرکت‌های ISP به عنوان یکی از اصلی‌ترین نقاط فروش خود، اغلب مشتریان را به باز کردن

حساب‌های ایمیل در سرورهای خود دعوت می‌کردند. در آن سال‌ها، عرضه‌ی کامپیوترهای شخصی با برنامه‌های ایمیل از پیش نصب شده نیز شروع شد. مردم بیشتر از این کامپیوترها برای اتصال به ISP یا سرور ایمیل کارفرمای خود استفاده می‌کردند.

ایمیل مانند آتش‌سوزی سریعاً گسترش یافت. افراد عادی به صورت دسته‌جمعی به اینترنت هجوم می‌آورند و گاهی اوقات این کار صرفاً برای استفاده از ایمیل بود. یک پروتکل به عنوان فعال‌کننده‌ی اصلی این انقلاب ایمیل نقشی برجسته داشت. بیایید اکنون یاد بگیریم که چگونه با استفاده از آن پروتکل پیام‌ها را بین میزبان‌ها و سرورهای ایمیل منتقل کنیم.

پروتکل ساده‌ی انتقال میل

پرکاربردترین پروتکل ایمیل، **پروتکل ساده‌ی انتقال میل**[۱] یا SMTP است که قوانین مکالمه بین سرور ایمیل و کامپیوتر کلاینت را تعریف می‌کند. همانطور که خواهیم دید، SMTP پیچیده‌تر از پروتکل‌های *درخواست - پاسخ*[۲] مانند DNS و NTP است[۳].

ابتدا باید بدانیم که ایمیل‌ها ممکن است حاوی پیام‌هایی باشند که در یک بسته‌ی IP جا نمی‌گیرند. هنگامی که یک ایمیل توسط تعداد زیادی بسته‌ی IP حمل می‌شود، مهم است که بارهای آن‌ها به ترتیب در کنار هم قرار گیرند. به همین دلیل، SMTP به جای UDP بر روی TCP اجرا می‌شود.

سرورهای ایمیل انتظار بر روی پورت شماره ۲۵ پروتکل TCP منتظر برقراری اتصال هستند. وقتی اتصال برقرار شد، سرور مکالمه را شروع می‌کند و برای شناسایی خود پیامی می‌فرستد:

```
220 mail-server.example.com
```

تمام پیام‌های ارسال شده توسط سرور با یک عدد سه رقمی شروع می‌شوند که به آن **کد بازگشت**[۴] می‌گویند. پروتکل SMTP کدهای مختلفی را تعریف می‌کند که هر کدام معنای خاص خود را دارند. به طور خاص، کد ۲۲۰ اعلام می‌کند که سرور آماده‌ی دریافت دستورالعمل است. بعد از کد، سرور نام خود را بیان می‌کند. در پاسخ، مشتری باید یک فرمان HELO برای شناسایی خود ارسال کند:

```
HELO client.code.energy
```

---

[۱] **S**imple **M**ail **T**ransfer **P**rotocol یا **SMTP**

[۲] Request-Response

[۳] پروتکل‌های درخواست-پاسخ دقیقاً همان چیزی هستند که به نظر می‌رسند: یک کلاینت درخواستی را به سرور ارسال می‌کند و سپس منتظر پاسخ می‌ماند.

[۴] Return Code

در این مرحله، سرور بر اساس هویت مورد ادعای کلاینت تصمیم می‌گیرد که آیا می‌خواهد مکالمه را ادامه دهد یا خیر. برخی از سرورهای ایمیل از DNS معکوس استفاده می‌کنند و نام گزارش شده‌ی کلاینت را با نامی که به آدرس IP آن پیوند داده شده، مقایسه می‌کنند. اگر سرور ایمیل تصمیم به ادامه کار داشته باشد، یک کد پاسخ ۲۵۰ ارسال می‌کند، به این معنی که اقدام درخواست شده توسط کلاینت پذیرفته شده است:

```
250 mail-server.example.com
```

با این تبادل، هم کلاینت و هم سرور تأیید کرده‌اند که در شرف انتقال ایمیل هستند. هر دوی آن‌ها همچنین بررسی کرده‌اند که با همتای مورد نظر خود در ارتباط هستند. در مرحله‌ی بعد، از کلاینت انتظار می‌رود که آدرس بازگشت ایمیلی که باید ارسال شود را ارائه دهد. اگر سرور نتواند ایمیل کلاینت را تحویل دهد، ایمیل دیگری را به آدرس بازگشت ارسال می‌کند و موضوع را به او اطلاع می‌دهد:

```
MAIL FROM: <ada@code.energy>
```

مجدداً سرور این امکان را دارد که این را بپذیرد یا رد کند. برخی از سرورهای ایمیل طوری پیکره‌بندی شده اند که فقط آدرس‌های ایمیل خاصی را بپذیرند. اگر آدرس پذیرفته شود، سرور کد ۲۵۰ را برمی‌گرداند:

```
250 Ok
```

در مرحله‌ی بعد، کلاینت باید صندوق پستی مقصد را به سرور اطلاع دهد:

```
RCPT TO: <charles@example.com>
```

دوباره سرور وضعیت پذیرش این ایمیل مقصرد را بررسی می‌کند. اگر سرور پیام دریافتی را بپذیرد، از کد ۲۵۰ مشابه استفاده می‌کند تا به کلاینت اطلاع دهد کار را دنبال کند:

```
250 Ok
```

اکنون کلاینت می‌تواند دستور DATA را ارسال کند. این دستور از سرور می خواهد تا فرایند ارسال پیام ایمیل را آغاز کند:

```
DATA
```

سرور تأیید می‌کند و به کلاینت دستور می‌دهد تا پایان انتقال را با خطی که فقط حاوی یک نقطه است علامت دهد. کد بازگشت ۳۵۴ به کلاینت اطلاع می‌دهد که سرور کاراکترهای دریافتی بعدی را به عنوان بخشی از پیام ایمیل در نظر می‌گیرد[1]:

```
354 End data with <CR><LF>.<CR><LF>
```

[1] کاراکترهای غیرچاپی <CR><LF> یک سرخط و به دنبال آن یک خط جدید هستند. این ترکیب پایان یک خط را نشان می دهد.

سپس کلاینت می تواند پیام را منتقل کند. هدرهای From، To و Date اجباری هستند و باید در هر ایمیل وجود داشته باشند. انواع دیگری از هدرها، مانند SUBJECT نیز معمولاً استفاده می‌شوند، اما الزامی نیستند. متن پیام نیز اختیاری است. ارسال ایمیل بدون متن مشابه ارسال یک پاکت خالی است. این نمونه‌ای از ایمیلی است که کلاینت می‌تواند ارسال کند:

```
From: <ada@code.energy>
Date: Wed, 27 Nov 2002 15:30:34 +0100
To: <charles@example.com>

That brain of mine is something more than
merely mortal; as time will show.
.
```

پس از تحویل <CR><LF>.<CR><LF> پایانی سرور بررسی می‌کند که داده‌های دریافتی معتبر هستند یا خیر؛ و در صورت تایید پیام، آن را به طور رسمی می‌پذیرد. بسیاری از سرورها نیز شناسه‌ی داخلی را که به ایمیل دریافتی اختصاص داده‌اند به کلاینت اطلاع می‌دهند:

```
250 Ok: queued as 1079212633C
```

در این مرحله، کلاینت می تواند مطمئن باشد که سرور یا پیام را تحویل می دهد یا آن را با یک پیام خطا[1] برمی‌گرداند. کلاینت ممکن است با ارسال یک فرمان MAIL FROM دیگر، کار را برای ارسال ایمیل‌های بیشتر ادامه دهد. در غیر این صورت می‌تواند با ارسال یک فرمان QUIT، مودبانه بگوید که فرایند تمام شده است:

```
QUIT
```

سرور پس از دریافت این پیام، باید خداحافظی کند و اتصال TCP را ببندد:

```
221 Bye
```

هر بار که ایمیلی ارسال می‌کنید، مکالمه‌ای شبیه به این مکالمه انجام می‌شود که نرم‌افزار ایمیل شما آن را در پشت صحنه انجام می‌دهد. در ابتدا، هیچ احراز هویتی در SMTP وجود نداشت: سرورها به صورت کورکورانه به مشتریان اعتماد داشتند که ایمیل‌های قانونی ارسال می‌کنند. متأسفانه، هنگامی که ایمیل محبوب شد، کاربران بی‌ملاحظه‌ی اینترنت شروع به ارسال ایمیل‌های انبوه ناخواسته به هر آدرس

[1] پروتکل SMTP به گونه‌ای طراحی شده است که قابل اعتماد باشد، بنابراین سرورهای ایمیل یک ایمیل تایید شده را بدون اطلاع قبلی رد نمی‌کنند. به پیامی که رد می‌شود، برگشتی یا Bounce گفته می‌شود زیرا محتوای آن به فرستنده بازگردانده می‌شود.

ایمیلی که داشتند، کردند. مخرب‌ترین آن‌ها حتی ایمیل‌های جعلی را با فیلد From غیرقانونی ارسال می‌کنند.

ارسال ایمیل‌ها

ایمیل در ابتدا به عنوان یک سیستم باز ایجاد شد که هر کسی می‌توانست بدون درخواست مجوز از هیچ مرجع مرکزی از آن استفاده کند. پیشگامان آن امیدوار بودند که همه‌ی شرکت‌کنندگان با حسن نیت عمل کنند. متاسفانه، به محض اینکه ایمیل به طور گسترده پذیرفته شد، این امید از بین رفت. با این وجود، مهندسان به کار خود ادامه دادند تا بدون تغیر در ماهیت باز ایمیل، از آن در مقابل کاربران بد محافظت کنند.

در ابتدا، مهندسان شروع به نگه‌داشتن **لیست‌های سیاه**[۱] عمومی کردند که اغلب با نام و آدرس IP سوءاستفاده‌کنندگان شناخته‌شده‌ی ایمیل به روز می‌شد. مدیران نیز سرورهای خود را به گونه‌ای پیکره‌بندی می‌کردند که به طور خودکار اتصالات کاربران موجود در لیست سیاه را رد کنند. این امر مشکل را کمی تعدیل کرد، اما آن را برطرف نکرد: سوءاستفاده‌کنندگان یاد گرفتند که از آدرس‌های IP جدید و نام‌های دامنه برای ارسال ایمیل‌های ناخواسته استفاده کنند.

تا به امروز، اکثر ایمیل‌های بد از اتصالات اینترنتی ارائه شده توسط شرکت‌های ISP ارسال می‌شوند. بیشتر سازمان‌ها به‌منظور حفظ شهرت خود، استفاده از شبکه‌های خود را به دقت کنترل می‌کنند. با این حال، شرکت‌های ISP اغلب به میلیون‌ها کاربر خدمات ارائه می‌دهند، و غربالگری جداگانه‌ی هر یک امکان‌پذیر نیست.

خوشبختانه، بررسی اینکه آیا یک آدرس IP متعلق به یک سازمان دارای کنترل است یا یک اتصال خانگی، آسان است: باید به رکورد DNS معکوس آن مراجعه کنید. شرکت‌های ISP به مشتریان خانگی اجازه نمی‌دهند این رکوردها را تنظیم کنند، اما همه‌ی سرورهای ایمیل معتبر DNS معکوس مناسبی برای آدرس‌های IP خود خواهند داشت. بنابراین سرورهای ایمیل شروع به محدود کردن آدرس‌های IP خانگی کرده‌اند؛ که این امر به شدت سوءاستفاده از ایمیل را کاهش می‌دهد.

به منظور اجرای این سیاست، SMTP به نسخه‌ی توسعه‌یافته‌ی ESMTP[۲] با پشتیبانی از احراز هویت از طریق نام کاربری و رمز عبور ارتقا یافته است. برای همه‌ی کاربران خانگی، دستورات MAIL FROM تنها پس از احراز هویت کاربر از طریق دستور AUTH پذیرفته می‌شوند.

---

[۱] Blacklist

[۲] Extended SMTP

امروزه ارسال ایمیل به این صورت است: شما یک ایمیل می‌نویسید و کامپیوتر شما از SMTP برای ارسال آن به سرور ایمیل‌تان استفاده می‌کند. سپس سرور ایمیل شما به سرور ایمیل مرتبط با صندوق پستی مقصد متصل می‌شود و از SMTP برای انتقال ایمیل شما استفاده می‌کند. از آنجایی که آدرس IP سرور ایمیل شما دارای یک رکورد DNS معکوس است، می‌تواند بدون احراز هویت با تمام سرورهای ایمیل دیگر ارتباط برقرار کند.

بنابراین ترافیک ایمیل به دو بخش تقسیم می‌شود: کاربرانی که پیام‌ها را به سرور ایمیل خود ارسال می‌کنند و سرورهای ایمیل که پیام‌ها را به یکدیگر منتقل می‌کنند. امروزه پورت شماره ۵۸۷ در پروتکل TCP برای ارسال ایمیل و پورت ۲۵ برای رله‌ی ایمیل رزرو شده است. کاربران ایمیل خانگی دیگر از پورت شماره ۲۵ در پروتکل TCP استفاده نمی‌کنند.

بر این اساس، اکثر شرکت‌های ISP حتی بسته‌های IP خانگی حاوی سگمنت‌هایی به مقصد پورت ۲۵ را رد کرده و به طور موثری کاربران اینترنت خانگی را از انجام رله‌ی ایمیل منع می‌کنند. سرورهای ایمیل انتظار اتصالات ورودی بر روی دو پورت TCP را دارند:

- پورت ۲۵ برای دریافت ایمیل از سایر سرورهای میل،
- پورت ۴۶۵ برای دریافت ایمیل از کاربران احراز هویت شده.

همه‌ی این کارها هنوز برای کنترل ارسال‌کنندگان ایمیل‌های مخرب کافی نیستند. نبرد بین مهندسان و فرستندگان هرزنامه‌ها همچنان ادامه دارد. در برخی موارد، مهندسان با استفاده از مکانیزم‌های مرتب‌سازی پیچیده حدس می‌زنند کدام ایمیل‌ها ناخواسته هستند. در برخی موارد دیگر، آن‌ها از امضاهای مبتنی بر رمزنگاری برای تأیید آدرس ایمیل فرستنده استفاده می‌کنند.

بازیابی ایمیل‌ها

اوایل، افراد ایمیل‌های خود را با باز کردن مستقیم فایل‌های صندوق پستی می‌خوانند. در بیشتر موارد، ایشان جلوی سرورهای ایمیل خود نمی‌نشستند، بلکه از تلنت برای باز کردن فایل‌های صندوق پستی خود از راه دور استفاده می‌کردند. مهندسان متوجه شدند که اگر افراد بتوانند ایمیل‌های خود را در کامپیوتر شخصی خود دانلود کنند و آن‌ها را به صورت محلی بخوانند، کار آسان‌تر خواهد بود. این کار باعث می‌شود هر بار که نیاز به خواندن یک ایمیل داشتند، از دردسر استفاده از تلنت دوری کنند.

از آنجایی که SMTP روش مناسبی برای سرور به منظور ارسال ایمیل به کاربر نهایی نبود، استراتژی دیگری مورد نیاز بود. از این گذشته، اکثر کاربران کامپیوترهای خود را به صورت شبانه‌روزی آنلاین ندارند و آماده‌ی دریافت اتصالات SMTP ورودی نیستند. بنابراین، پروتکل‌های جدیدی توسعه داده

شدند. این پروتکل‌ها به کاربران اجازه می‌دادند تا با سرور ایمیل ارتباط برقرار کرده، فهرستی از ایمیل‌ها را در صندوق پستی دریافت و انتخاب نموده که کدام‌یک را از فهرست دانلود کنند.

رایج ترین پروتکل مورد استفاده، **پروتکل دسترسی به پیام‌های اینترنتی**[۱] یا IMAP است. اصل کار این پروتکل شبیه به SMTP است، سرور همچنان با استفاده از کدهای دستوری و متنی ساده با کلاینت مکالمه می‌کند. دستوراتی برای کلاینت‌ها وجود دارد تا بتوانند ایمیل‌ها را در صندوق پستی خود فهرست کرده و درخواست ارسال ایمیل خاصی را ارائه دهند.

امروزه تقریباً نیمی از ایمیل‌های دنیا از طریق خدمات **وبمیل**[۲] مانند جی‌میل ارسال و دریافت می‌شوند. این خدمات بدون دردسرترین راه دسترسی را به ایمیل ارائه می‌کنند، زیرا نیازی به نصب یا پیکره‌بندی یک برنامه کلاینت بر روی کامپیوتر شما ندارند. بیایید اکنون **وب**[۳] را بررسی کنیم: مجموعه‌ای زیبا از مکانیزم‌هایی که این کار و خیلی کارهای دیگر را ممکن کرده است.

۵-۲- وب

اگر یک سند متنی حاوی پیوندهایی باشد که به اسناد دیگر ارجاع می‌دهند، و این پیوندها به شما امکان می‌دهند در چندین سند به جلو و عقب بپرید، به این متن **ابرمتن**[۴] گفته می‌شود: پیوندها یک بعد اضافی در فضای متن ایجاد می‌کنند. به عنوان مثال، *ویکی‌پدیا* فرامتن است، اما یک دایره‌المعارف فیزیکی اینطور نیست. لازم نیست که ابرمتن مانند متن معمولی به صورت خطی خوانده شود؛ می‌توان آن را با پیمایش از طریق پیوندها کاوش کرد.

با گسترش واسط‌های کاربری گرافیکی در دهه‌ی ۱۹۸۰، دانش‌ورزان[۵] دریافتند که فرامتن می‌تواند کار آن‌ها را آسان‌تر و کارآمدتر کند. در حالی که برنامه‌های اولیه‌ی خواندن و نوشتن اسناد ابرمتن محبوبیت پیدا کردند، هنوز ایجاد پیوند بین اسناد ابرمتن موجود در کامپیوترهای مختلف غیرممکن بود. اسناد ابرمتن تا حد زیادی از یکدیگر جدا بودند.

با ظهور اینترنت، سیستم ابرمتن جدیدی برای غلبه بر این محدودیت ایجاد شد. اسناد ایجاد و ذخیره‌شده در کامپیوترهای مختلف در نهایت توانستند به یک وب جهانی از اسناد مرتبط شوند. این

---

[۱] **I**nternet **M**essage **A**ccess **P**rotocol یا **IMAP**

[۲] Webmail

[۳] Web

[۴] Hypertext

[۵] Knowledge Workers

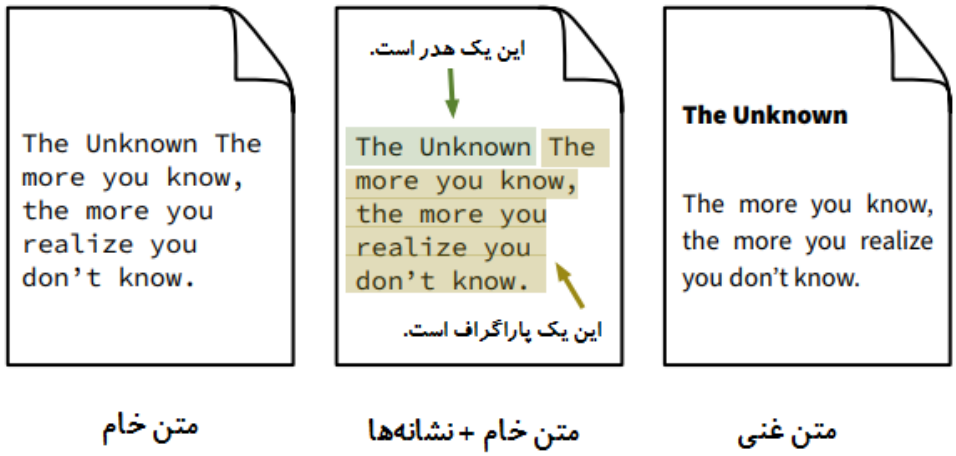
سیستم **وب جهانی**[۱] یا **WWW** نام داشت و اسناد ابرمتن آن **صفحات وب**[۲] نامگذاری شدند. برای اینکه این سیستم کار کند، کامپیوترهای مختلف در سراسر جهان باید سه مولفه‌ی اساسی را به اشتراک بگذارند:

- یک زبان برای نوشتن صفحات وب در قالب فایل‌ها،
- یک راه برای پیوند دادن صفحات وب به یکدیگر،
- یک پروتکل برای انتقال فایل‌ها بین کامپیوترها

برنامه‌ای که از این سه مولفه استفاده می‌کند **مرورگر وب**[۳] نامیده می‌شود. امروزه تقریباً هر مرورگر وب بر روی مجموعه‌ای اساسی از استانداردها توافق دارد. بیایید بررسی کنیم که مرورگرهای وب چگونه کار می‌کنند.

زبان نشانه‌گذاری ابرمتن

ایجاد صفحات وب باید آسان و در عین حال برای همه کاربران اینترنت قابل درک باشد. برای دستیابی به این اهداف، تصمیم گرفته شد که صفحات وب از متن خام ولی غنی‌شده با برچسب‌های ویژه برای مشخص کردن ساختار و نحوه‌ی ارائه، تشکیل شوند.



شکل ۲-۵: اسناد غنی را می‌توان با نشانه‌گذاری متن خام تولید کرد.

---

[۱] World Wide Web یا WWW

[۲] Web Page

[۳] Web Browser

از برچسب‌ها می‌توان برای نشانه‌گذاری یک پاراگراف، تنظیم یک عبارت به عنوان هدر، تاکید بر کلمه و موارد دیگر استفاده کرد. مجموعه‌ای از برچسب‌ها که می‌توانند با متن ترکیب شوند تا آن را گسترش دهند، یک **زبان نشانه‌گذاری**[۱] را تشکیل می‌دهند. زبان‌های نشانه‌گذاری متعددی وجود دارند. زبانی که برای وب ایجاد شده است، زبان نشانه‌گذاری ابرمتن[۲] یا HTML نام دارد. در HTML، هر متنی که توسط کاراکترهای <...> محصور شده باشد یک برچسب[۳] است. به عنوان مثال، در متن زیر یک هدر و به دنبال آن یک پاراگراف آورده شده‌اند:

```
<h1>The Unknown</h1>
<p>The more you know, the more you realize you don't
know.</p>
```

این مثال شامل دو زوج برچسب HTML است: <h1>...</h1> و ... . برچسب‌های h1 نشان می‌دهند که عبارت The Unknown یک هدر است. برچسب‌های p یک پاراگراف را مشخص می‌کنند. بسیاری از برچسب‌های دیگر غیر از این‌ها نیز وجود دارند و بیشتر آن‌ها برای تعریف *ساختار* سند استفاده می‌شوند[۴].

معمولاً، برچسب‌های HTML به صورت زوج باز/بسته آورده می‌شوند: هر زوج بر روی بخشی از سند اعمال می‌شود و برچسب بسته شامل یک اسلش است. با این حال، چند برچسب، که **برچسب تهی**[۵] نامیده می‌شوند، زوج نیستند. به عنوان مثال،
 برای درج شکسته خط در متن و <img> برای درج یک تصویر استفاده می‌شود.

صفحات وب کامل HTML باید دارای دو قسمت باشند: **هد**[۶] و **بادی**[۷]. هد حاوی اطلاعاتی درباره‌ی سند است که مرورگرهای وب نباید نمایش دهند. از سوی دیگر، بادی قرار است توسط مرورگرهای وب بر روی صفحه نمایش داده شود. در اینجا یک مثال از یک صفحه‌ی وب ساده اما کامل HTML آورده شده است:

```
<html>
  <head>
      <title>Daily quote</title>
  </head>
  <body>
```

---

۱ Markup Language
۲ HyperText Markup Language یا HTML
۳ Tag
۴ زبان دیگری به نام Cascading Style Sheets یا CSS بعدها برای سفارشی‌کردن بیشتر جنبه‌های نحوه‌ی ارائه‌ی یک سند HTML و ساختار آن ایجاد شد: فونت‌ها، طرح‌بندی، رنگ‌ها و غیره.
۵ Empty Tag
۶ Head
۷ Body

```
        <h1>The Unknown</h1>
        <p>The more you know, the more you realize
        you don't know.</p>
    </body>
</html>
```

همانطور که می‌بینید، هد و بادی هر کدام در یک زوج برچسب با نام مربوطه‌ی خود قرار گرفته‌اند و هر دو در یک زوج برچسب <html> قرار دارند. سعی کنید یک فایل متنی با این محتوا ایجاد، آن را به عنوان یک فایل .html ذخیره و در یک مرورگر وب باز کنید! در یک فایل HTML، فاصله و شکست خط توسط مرورگر نادیده گرفته می‌شود. این موارد فقط به منظور تسهیل خواندن فایل برای انسان استفاده می‌شوند.

بسیاری از برچسب‌ها می‌توانند دارای ویژگی‌هایی باشند که آن‌ها را گسترش داده یا تکمیل کنند. به عنوان مثال، برچسب <html> می‌تواند حاوی یک ویژگی باشد که زبان سند را مشخص می‌کند: <html lang="en">. برخی از برچسب‌ها دارای یک ویژگی اجباری هستند. برای مثال، تگ <img> باید مشخص کند که کدام تصویر نمایش داده شود:

```
<img scr="cat.gif">
```

مهمترین برچسب HTML برچسب <a> یا **انکور**[۱] یا **لنگر** است. این برچسب بخشی از سند را به پیوندی به مکانی در یک ابرمتن دیگر تبدیل می‌کند. به مثال زیر توجه کنید:

```
Everybody loves <a href="cats.html">them</a>.
```

مرورگرهای وب این خط را به صورت زیر بر روی صفحه نمایش می‌دهند:

Everybody loves them.

با کلیک کردن بر روی کلمه‌ی زیرخط‌دار، سند فوراً به مکان دیگری که توسط ویژگی مرجع ابرمتن[۲] یعنی href مشخص شده است، جابجا می‌شود. اکنون بیایید بیاموزیم که چگونه این مکان‌ها را بیان کنیم تا مرورگرهای وب آن‌ها را پیدا کنند.

مکان‌یاب منبع یکسان

---

۱ Anchor
۲ **H**ypertext **REF**rence

پارامتر href انکور ارجاع به سندی است که پس از کلیک روی انکور نشان داده می‌شود. این پارامتر از یک قالب ساده پیروی می‌کند: دو اسلش، به دنبال آن نام میزبانی که سند در آن قرار دارد، به اضافه‌ی مسیر سند. برای مثال، در اینجا پیوندی به سند "pets/cat.html/"، واقع در میزبان zoo.org وجود دارد:

```
<a href="//zoo.org/pets/cat.html">kitten</a>
```

مرورگرها مسیر سند را به عنوان محل یک فایل در سیستم فایل میزبان تعبیر می‌کنند. به عنوان مثال، سندی که توسط انکور بالا به آن ارجاع داده شده است، یک فایل با نام "cat.html" در پوشه‌ی "pets" فرض می‌شود.

مراجع نسبی: اگر یک مرجع با علامت اسلش شروع نشود، به عنوان یک مرجع مرتبط با پوشه‌ی جاری در نظر گرفته می‌شود. به عنوان مثال، فرض کنید این پیوند در سند //zoo.org/pets/cat.html وجود دارد:

```
Are <a href="dog.html">they</a> our best friends?
```

در اینجا، پارامتر href به فایلی در همان پوشه‌ی فایل حاضر (cat.html) اشاره دارد. از این رو، مرورگر این مرجع را به عنوان //zoo.org/pets/dog.html تفسیر می‌کند. همچنین مراجعی که تنها با یک اسلش شروع نیز وجود دارند که مربوط به همان میزبان هستند:

```
<a href="/bugs/ant.html">Ants</a> are small.
```

این مرجع به عنوان //zoo.org/bugs/ant.html تفسیر می‌شود.

مرجع بخشی: راهی برای ارجاع به بخش یا قطعه‌ی خاصی از یک سند HTML وجود دارد. فرض کنید فایل dog.html به شکل زیر است:

```
<h1 id="bulldog">Bulldog</h1>
<p>…</p>
<h1 id="poodle">Poodle</h1>
<p>…</p>
<h1 id="golden">Golden Retriever</h1>
<p>…</p>
```

می‌توانیم یک انکور ایجاد کنیم که مستقیماً به هدر مربوط به سگ رتریور طلایی در سند dog.html پیوند می‌دهد، مانند این:

```
<a href="dogs.html#golden">These dogs</a> rock!
```

کاراکتر # می‌تواند برای پرش به مکان دیگری *در سندی که در حال دسترسی* است نیز استفاده شود. به عنوان مثال، این ترفند در *کادر محتویات* صفحات ویکی‌پدیا استفاده می‌شود تا به شما اجازه دهد مستقیماً به بخشی که به آن علاقه دارید بروید.

از مراجع مشابه در هنگام ارجاع به انواع دیگر منابع رسانه‌ای مانند تصاویر استفاده می‌شود. به عنوان مثال، اگر تصویری از یک سگ رتریور طلایی در میزبان zoo.org وجود داشته باشد، می‌توان آن را به این روش در سند گنجاند:

```
<h1>Golden Retriever</h1>
<img src="/pics/golden-retriever.jpg">
<p>As you can see in this picture…</p>
```

طرح: اطلاعات مهم دیگری وجود دارد که می‌توانیم به مراجع وب خود اضافه کنیم: *روش دسترسی به منبع*، که به عنوان **طرح**[۱] شناخته می‌شود. یکی از مواردی که ممکن است بلافاصله تشخیص دهید http است، پرکاربردترین پروتکل برای انتقال فایل‌های صفحات وب بین کامپیوترها. این طرح در ابتدای یک مرجع ظاهر می‌شود و پس از آن یک علامت دونقطه قرار می‌گیرد:

```
<a href="http://zoo.org/pets/cat.html">kitten</a>
```

یک ارجاع کامل به یک منبع وب، که شامل یک طرح، یک میزبان و یک مسیر سند است، یک **مکان‌یاب منبع یکسان**[۲] یا URL نامیده می‌شود. مردم آن‌ها را **آدرس وب** نیز می‌نامند. مرورگرهای وب معمولاً URL صفحه‌ای را که در حال رندر کردن هستند در نواری در بالای صفحه نمایش می‌دهند. حال بیایید ببینیم که طرح http چگونه کار می‌کند.

پروتکل انتقال ابرمتن

---

۱ Scheme

۲ Uniform Resource Locator یا URL

متداول‌ترین اقدامی که هنگام مرور وب انجام می‌دهیم این است که روی پیوندی کلیک می‌کنیم تا از صفحه‌ی وب در حال مشاهده را به صفحه‌ی بعدی تغییر دهیم. هر بار که روی پیوندی کلیک می‌شود، مرورگر وب باید با کامپیوتر میزبان سند ارجاع شده تماس بگیرد و یک کپی از آن فایل را بازیابی کند. **پروتکل انتقال ابرمتن**[۱] یا HTTP واسطه‌ی این انتقال اسناد است.

از آنجایی که اسناد منتقل‌شده می‌توانند بزرگ‌تر از MTU باشند، HTTP به جای UDP به TCP متکی است. این پروتکل یک طراحی کلاینت–سرور دارد: یک مرورگر وب برنامه‌ی کلاینت است، و وب سرور انتظار اتصالات در پورت ۸۰ را دارد. پس از برقراری ارتباط، کلاینت درخواستی را ارسال می‌کند. هنگامی که سرور آن را پردازش می‌کند، یک پاسخ ارسال می‌شود.

در سال ۲۰۱۵، HTTP ارتقاء زیادی یافت[۲]، اما نسخه‌ی مربوط به دهه‌ی ۱۹۹۰ هنوز به طور گسترده مورد استفاده قرار می‌گیرد. درخواست‌هایی که از نسخه‌ی قبلی HTTP پیروی می‌کنند به صورت متن ساده هستند. خط اول نوع درخواست، مسیر سند و نسخه‌ی پروتکل را نشان می‌دهد. خطوط بعدی مانند پیام‌های ایمیل برای هدرها استفاده می‌شوند. در نهایت یک خط خالی پایان درخواست را نشان می‌دهد:

```
GET /pets/cat.html HTTP/1.0
User-Agent: Mozilla/5.0 (Macintosh)
Accept-Language: en-us
```

مثال فوق رایج‌ترین نوع درخواست را نشان می‌دهد: GET. این درخواست نشان می‌دهد که کلاینت می‌خواهد یک سند را بازیابی کند. هدرها اختیاری هستند، اما تقریباً همیشه گنجانده می‌شوند. مثال فوق، دو مورد از رایج‌ترین هدرها را نشان می‌دهد. هدر Accept-Language مشخص می‌کند که نسخه‌ی انگلیسی سند ترجیح داده می‌شود، و User-Agent می‌گوید کدام نرم‌افزار مرورگر وب و کدام سیستم عامل کامپیوتر، درخواست را ایجاد کرده است. یک پاسخ معمولی از یک سرور به چنین درخواستی به صورت زیر است:

```
HTTP/1.0 200 OK
Server: nginx/1.15.8
Date: Wed, 19 Aug 2020 20:46:53 GMT
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
Content-Length: 49
Content-Type: text/html
```

---

[۱] **H**yper**T**ext **T**ransfer **P**rotocol یا **HTTP**

[۲] نسخه‌ی ۲۰۱۵ پروتکل HTTP/2 نامیده می‌شود که چندین بهبود عملکرد نسبت به نسخه‌ی قبلی دارد، اما ساختار کلی اطلاعات ارسال شده را تغییر نمی‌دهد.

```
<html><body><h1>Cats are cute!</h1></body></html>
```

خط اول نسخه‌ی پروتکل را نشان می‌دهد و به دنبال آن یک کد وضعیت سه رقمی، مشابه آنچه در SMTP استفاده می‌شود، می‌آید. در اینجا، ۲۰۰ به این معنی است که درخواست بدون مشکل انجام شده است. کدهای بسیار دیگری نیز وجود دارند. به عنوان مثال، ۴۰۴ نشان می‌دهد که سند درخواستی یافت نمی‌شود. کدها بر اساس رقم اول گروه‌بندی می‌شوند. یعنی، آن‌هایی که با ۲ شروع می‌شوند، درخواست‌های موفقیت‌آمیز را نشان می‌دهند، و آن‌هایی که با ۴ شروع می‌شوند خطای ناشی از درخواست‌های نامناسب را نشان می‌دهند.

بعد از خط اول، هدرهای اختیاری گنجانده می‌شوند. در مثال ما، هدرهایی وجود دارند که نرم افزار سرور، تاریخ ارسال پیام، آخرین تغییر سند درخواستی، نوع سند حمل شده و طول کل آن را به کلاینت اطلاع می‌دهند. بسیاری از هدرهای دیگر به طور گسترده برای احراز هویت، ردیابی، کش کردن و غیره استفاده می‌شوند.

پروتکل HTTP فوق‌العاده موفق بوده است و در خارج از شبکه‌ی جهانی وب نیز مورد استفاده قرار می‌گیرد. به عنوان مثال، یک برنامه در تلفن شما ممکن است یک درخواست HTTP برای به دست آوردن اطلاعات خام آب و هوا ارسال کند. سپس داده‌های بازیابی شده را می‌توان برای نمایش شرایط آب‌وهوایی توسط یک ویجت در واسط تلفن، به جای صفحه‌ی وب ارائه‌شده توسط سرور، استفاده کرد.

برنامه‌های کاربردی وب

یک درخواست HTTP ممکن است دارای مسیری باشد که به یک برنامه ارجاع می‌دهد. در چنین مواقعی سرور برنامه را اجرا می‌کند و با خروجی آن پاسخ می‌دهد. به عنوان مثال، درخواست برای /random در code.energy به برنامه‌ای اشاره دارد که صفحه‌ای حاوی یک عدد تصادفی تولید می‌کند. سرور به جای صفحه‌ی ایستا که در هر درخواست یکسان به نظر می‌رسد، یک صفحه‌ی پویا را برمی‌گرداند. از این صفحه در یک مرورگر بازدید کرده و آن را چند بار بازخوانی کنید تا خودتان ببینید:

```
http://code.energy/random
```

صفحات پویا می‌توانند ورودی‌هایی را نیز از کاربر به عنوان پارامتر دریافت کنند. به عنوان مثال، google.com دارای صفحه‌ی پویا search/ است که پارامتری به نام q را به عنوان یک عبارت جستجو می‌گیرد. پارامتر به صورت زیر بعد از مسیر سند، اضافه می‌شود:

```
http://google.com/search?q=energy
```

برای بارگیری این صفحه وب، مرورگرها می‌توانند این درخواست را ارسال کنند:

```
GET /search?q=energy HTTP/1.0
```

با دریافت این درخواست، وب سرور برنامه مرتبط با search/ را فراخوانی می‌کند. هر پارامتر به شکل name=value? پس از مسیر، به برنامه ارسال می‌شود. پارامترهای ورودی در این فرم، **رشته‌های پرس‌وجو**[۱] نامیده می‌شوند.

چون کاراکترهایی وجود دارند که آدرس‌هایURL معتبر نمی‌توانند حاوی آن‌ها باشند، یک راه استاندارد برای کدگذاری آن‌ها با استفاده از علامت درصد وجود دارد. برای مثال، فضای خالی به 20% وعلامت ! به 2F% تبدیل می‌شوند[۲]. رشته‌های پرس‌وجو می‌توانند شامل پارامترهای متعددی باشند که با امپرسند یا همان علامت & از هم جدا شده‌اند. بازدید از URL زیر عبارت code.energy را جستجو و یک پارامتر عددی اضافی را برای لیست کردن تنها پنج نتیجه ارسال می‌کند:

```
http://google.com/search?q=code%20energy&num=5
```

رشته‌های پرس‌وجو به خوبی با HTML یکپارچه شده‌اند. برخی از برچسب‌ها، از جمله <form> و <input>را می‌توان به صفحات وب اضافه کرد تا کاربر به راحتی داده‌ها را با استفاده از رشته‌های پرس‌وجو وارد و ارسال کند. کد HTML زیر یک کادر ورودی برای وارد کردن کلمات کلیدی و یک دکمه برای ارسال درخواست ایجاد می‌کند:

```
<form action="//google.com/search">
   <input name="q">
   <button>Search Google</button>
</form>
```

---

[۱] Query Strings

[۲] در رشته‌های پرس‌وجو، فضای خالی می‌تواند با 20% یا به سادگی با علامت مثبت + کدگذاری شود. با این حال، این روال در جاهای دیگر URL جواب نمی‌دهد. کاراکترهایی که دارای توابع خاص هستند (مانند +، %، ?، =، و &) نیز باید برای دور زدن آن توابع، کدگذاری شوند.

هنگامی که دکمه فشار داده می‌شود، یک درخواست HTTP از نوع GET، متن تایپ شده در ورودی را به عنوان یک رشته‌ی پرس‌وجو به سرور ارسال می‌کند. در سمت سرور، یک برنامه پیوندهایی را که برای کاربر مفیدتر می‌داند پیدا کرده و آن‌ها را در یک سند صفحه‌ی وب می‌فرستد.

به طور کلی، برنامه‌هایی که به درخواست‌های HTTP از نوع GET پاسخ می‌دهند، نباید هیچ اقدام دیگری جز بازیابی اطلاعات انجام دهند. به عنوان مثال، درخواست GET نباید به کاربر اجازه دهد تا پیام وضعیت را در رسانه‌های اجتماعی پست کند، عکسی را از درایو ابری حذف کند یا در یک فروشگاه آنلاین سفارش دهد. این بدان معناست که کاربران می‌توانند درخواست‌های GET را بدون نگرانی در مورد عواقبی غیر از آنچه در صفحه‌ی وب آن‌ها نشان داده می شود، صادر کنند.

برای انجام این نوع کارها، چندین نوع درخواست HTTP دیگر وجود دارند. به عنوان مثال، اگر کار مورد نظر ایجاد چیزی است، باید از نوع POST استفاده شود. یک فرم HTML را می‌توان برای ارسال درخواست‌های POST با تنظیم پارامتر متد در برچسب <form>تغییر داد:

```
<form action="/send-message" method="POST">
<p>Subject: <input name="subject"></p>
<p>Message: <input name="message"></p>
<button>Send message</button>
</form>
```

در درخواست‌های POST، داده‌های ورودی در بدنه‌ی درخواست به جای رشته‌ی پرس‌وجو ارسال می‌شوند. با این حال، داده‌ها همچنان به همان روش کدگذاری می‌شوند. این یک درخواست POST است که می‌تواند با ارسال فرم از مثال بالا ایجاد شود:

```
POST /send-message HTTP/1.0
subject=Greetings&message=Hello%20World%2B
```

به این ترتیب می‌توان سیستم‌های پیچیده‌ای ایجاد کرد که به کمک مرورگرهای وب کار کنند. نمونه بارز آن سیستم‌های ایمیل مانند Hotmail و Gmail هستند. این وب سایت‌ها به افراد اجازه می‌دهند از طریق مرورگر خود از ایمیل استفاده کنند. وب سرور وظیفه‌ی مکالمه با سرورهای ایمیل را بر عهده دارد و به کاربران این امکان را می‌دهد که بدون برقراری مستقیم هیچ‌گونه اتصال SMTP از یک کلاینت وب، ایمیل ارسال کنند.

برای تسهیل این امر، صفحات HTML می‌توانند شامل کد جاوا اسکریپت باشند. در حقیقت، می‌توانیم صفحات وب را با همان قابلیت‌های برنامه‌های گرافیکی دیگر برنامه‌نویسی کنیم! صفحات وب به طور فزاینده‌ای از اسناد ابرمتن ساده به برنامه‌های کامل تبدیل شده‌اند.

نتیجه‌گیری

در این فصل در مورد برخی از برجسته‌ترین برنامه‌های کاربردی اینترنتی و پروتکل‌های آن‌ها مطالبی را یاد گرفتیم، اما بسیاری از برنامه‌های کاربردی دیگر نیز وجود دارند که آن‌ها را بررسی نکرده‌ایم. به عنوان مثال، در فصل ۱ دیدیم که مسیریاب‌ها باید به طور منظم اطلاعات اتصال را مبادله کنند. برای این منظور، آن‌ها برنامه‌هایی را اجرا می‌کنند که با استفاده از **پروتکل دروازه‌ای مرزی**[۱] یا BGP در پورت شماره‌ی ۱۷۹ پروتکل TCP ارتباط برقرار می‌کنند. اگر می‌خواهید در مورد BGP و دیگر پروتکل‌های جالب به طور عمیق اطلاعاتی کسب کنید، مراجع پایان فصل را بررسی کنید.

شماره‌های پورت

دیدیم که چگونه هر پروتکل به یک شماره پورت خاص TCP یا UDP مرتبط است. برنامه‌هایی که از پروتکل یکسانی استفاده می‌کنند تقریباً همیشه انتظار دارند که با شماره‌ی پورت یکسانی ارتباط برقرار کنند. به عنوان مثال، برای بازیابی یک صفحه‌ی وب از یک میزبان، نیازی به دانستن شماره‌ی پورت ندارید؛ وب سرورها ب ر روی پورت ۸۰ پروتکل TCP منتظر اتصالات هستند. سازمان IANA سازمانی است که تصمیم می‌گیرد کدام شماره پورت برای هر پروتکل استاندارد باشد. هنگامی که یک پروتکل جدید ایجاد می‌شود، توسعه‌دهندگان آن برنامه‌ای را برای رزرو شماره‌ی پورت به IANA ارسال می‌کنند[۲].

به جای توسعه‌ی یک پروتکل جدید و درخواست برای شماره‌ی پورت، بسیاری از توسعه‌دهندگان برنامه از پروتکل‌های عمومی موجود برای برقراری ارتباط از طریق اینترنت استفاده می‌کنند. به عنوان مثال، بسیاری از برنامه‌ها از طریق اینترنت با یکپارچه‌سازی با یک وب سرور قابل دسترسی هستند. در چنین مواردی، از HTTP برای انتقال داده‌های خام (به جای صفحات وب) به و از برنامه‌ها (به جای مرورگرهای وب) استفاده می‌شود.

همتا به همتا

---

[۱] Border Gateway Protocol یا **BGP**

[۲] پورت های ۰ تا ۴۹۱۵۱ فقط باید توسط پروتکل‌های ثبت شده در IANA استفاده شوند. پورت های ۴۹۱۵۲ تا ۶۵۵۳۵ هرگز به هیچ پروتکلی اختصاص داده نمی‌شوند و می‌توانند برای هر منظوری مورد استفاده قرار گیرند. می‌توانید تخصیص شماره پورت فعلی را در http://code.energy/ports بررسی کنید.

در این فصل، تمام پروتکل‌های لایه‌ی کاربرد که بررسی کرده‌ایم از معماری کلاینت-سرور پیروی می‌کنند. با این حال، یک معماری جایگزین به نام همتا به همتا[۱] یا P2P وجود دارد که در آن هر برنامه هم به عنوان سرور و هم به عنوان کلاینت عمل می‌کند. نمونه‌های قابل توجه پروتکل‌های P2P عبارتند از بیت‌تورنت، پروتکل به‌اشتراک‌گذاری فایل، و همچنین بیت‌کوین و اتریوم، دو شبکه‌ی ارزهای دیجیتال بزرگ.

سرویس‌های همتا به همتا، واسطه‌های بین کاربران را حذف می‌کنند و نقطه‌ی مدیریت مرکزی ندارند. بیت تورنت، بیت‌کوین و اتریوم را نمی‌توان تعطیل کرد. ارتش ایالات متحده از فناوری P2P به طور گسترده برای سربازان استفاده می‌کند تا بدون هماهنگی مرکزی و بدون تکیه بر زیرساخت‌های متمرکز مانند دکل‌های تلفن همراه عمل کنند.

امنیت

یکی دیگر از جنبه‌های مهم اینترنت که ما به آن اشاره نکرده‌ایم، عدم امنیت آن است. هر کسی که از یک مسیریاب استفاده می‌کند می‌تواند داده‌ها را از هر بسته‌ای که به آن می‌رسد بخواند و تغییر دهد. دیدیم که بسته‌های IP هنگام جابجایی از طریق اینترنت، توسط مسیریاب‌های ناشناخته بسیاری مدیریت می‌شوند. بنابراین، باید تمام داده‌هایی را که روی بسته‌های IP جابجا می‌شوند به‌عنوان اطلاعات عمومی بالقوه در نظر بگیریم، و نمی‌توانیم کاملاً اعتماد داشته باشیم که داده‌های دریافتی از طریق اینترنت در راه تغییر نکرده‌اند. برای کاهش این نقص، پروتکل‌های لایه کاربرد اغلب شامل طرح‌های رمزگذاری می‌شوند تا داده‌ها توسط واسطه‌ها خوانده یا دستکاری نشوند. . در فصل بعد، نحوه‌ی عملکرد آن را خواهیم دید.

منابع و مراجع

- Computer Networking: A Top-Down Approach, by Kurose
  - http://code.energy/kurose
- Computer Networks, by Tanembaum
  - http://code.energy/tanenbaum

[۱] Peer-**to**-Peer یا **P2P**

فصل ۳

امنیت

سیستم‌های کامپیوتری امن‌تر نمی‌شوند. من امیدوارم که در آینده، پیشرفت‌هایی که در زمینه‌ی رمزنگاری به وجود می‌آوریم، بر وضعیت نسبتاً بد امنیت مجازی تأثیر بگذارند.

– ادی شامیر[۱]

وظیفه‌ی حفاظت از داده‌های حساس مانند پیام‌های خصوصی، تراکنش‌های بانکی، سوابق پزشکی و غیره به برنامه‌نویسان سپرده شده است. این وظیفه‌ی شماست که از این داده‌ها در برابر هکرها محافظت کنید. شما باید اطمینان حاصل کنید فقط افرادی که دارای اعتبارنامه معتبر هستند می‌توانند به سیستم شما دسترسی داشته باشند و اطلاعات محرمانه حتی اگر فاش شوند قابل خواندن نباشند.

عمل ایمن‌سازی داده‌ها در برابر حملات اشخاص غیرمجاز، **رمزنگاری**[۲] نامیده می‌شود. الگوریتمی که به صورت برگشت‌پذیر داده‌ها را به شکلی نامفهوم رمزگذاری می‌کند، **رمز**[۳] نامیده می‌شود. بیشتر سیستم‌ها را می‌توان با استفاده از کتابخانه‌های رمزنگاری که توسط متخصصان بررسی می‌شوند، ایمن کرد. در این فصل، اصول اساسی رمزها و سایر ابزارهایی را که کتابخانه‌های رمزنگاری ارائه می‌کنند، بررسی خواهیم کرد. شما یاد خواهید گرفت که:

شما یاد خواهید گرفت که:


با رمزهای **قدیمی** سرگرم‌کننده اما ناامن، کار کنید،


آن‌ها را برای به‌دست آوردن رمزهای **متقارن** قوی بهبود دهید،


با استفاده از رمزهای **نامتقارن** برای افراد غریبه پیام محرمانه ارسال کنید،

---

[۱] Adi Shamir (1952-): رمزنگار و یکی از مخترعان الگوریتم رمزنگاری RSA و بسیاری دیگر از ابدعات مرتبط با رمزنگاری. م.

[۲] Cryptography

[۳] Cipher


هر مقدار داده را در قالب یک اثرانگشت دیجیتال **درهم‌ریزی** کنید،


پروتکل‌های شبکه‌ی خود را برای داشتن یک اینترنت امن‌تر ایمن کنید،


سیستم‌های خود را قبل از آدم‌های بد **هک** کنید تا آسیب‌پذیری‌ها را پیدا کنید.

اگر مهاجمان با موفقیت داده‌های رمزگذاری شده را رهگیری کنند، این احتمال وجود دارد که بفهمند چگونه داده‌ها را از هم جدا کرده و اسرار آن‌ها را فاش کنند. اگر این اتفاق بیفتد، می‌گوییم رمز مورد استفاده برای رمزگذاری داده‌ها **شکسته شده** است. بیایید با کاوش رمزهایی شروع کنیم که در گذشته محبوب بودند و امروزه به راحتی شکسته می‌شوند. این امر به ما کمک می‌کند بفهمیم چرا شکستن رمزهای مدرن دشوار است.

۱-۳- رمزهای قدیمی

یکی از قدیمی‌ترین رمزها توسط ژولیوس سزار بیش از دو هزار سال پیش برای ارسال نامه‌های مخفی به ژنرال‌های خود استفاده شد. پیام‌های رمزگذاری‌شده‌ی او به این صورت بودند[۱]:

```
GR QRW EULQJ DQB ERGB RI PHQ DFURVV WKH UKLQH
```

اگر نامه‌ی محرمانه‌ای از سزار به دست دشمنان می‌افتاد، اطلاعات موجود در نامه محرمانه باقی می‌ماندند. دشمنان سزار نمی‌توانستند پیام‌های رمزگذاری‌شده را بفهمند، اما ژنرال‌های او به راحتی می‌توانستند آن‌ها را بخوانند. برای رمزگذاری پیام‌ها، رومی‌ها از قبل توافق کرده بودند که هر حرف از پیام اصلی را در بر اساس حروف الفبا سه حرف به جلو منتقل کرده و حروف را به صورت زیر جایگزین کنند:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

در ادامه، پیام‌ها با جابجایی حروف به سمت عقب رمزگشایی می‌شدند. این نوع رمز به عنوان **رمز جابجایی**[۲] شناخته می‌شود و از آن‌جایی که دشمنان سزار هرگز نحوه‌ی عملکرد آن را کشف نکردند، امنیت کافی را فراهم می‌کرد. با این حال، هر کسی که در مورد این رمز چیزی بداند، می‌تواند به راحتی

---

[۱] در واقعیت، پیام‌ها کمی متفاوت بودند، زیرا سزار فقط ۲۳ حرف می‌دانست... و انگلیسی هم صحبت نمی‌کرد.

[۲] Shift Cipher

آن را بشکند، حتی اگر حروف بیش از سه حرف جابجا شوند: فقط ۲۵ جابجایی ممکن برای الفبای لاتین وجود دارد. مهاجم می‌تواند رمز را با امتحان کردن یک به یک این حالات تا زمانی که پیام معنی پیدا کند، بشکند.

بیایید نگاهی به چند اصطلاح مفید بیندازیم. یک پیام رمزگذاری‌شده **متن رمز**[۱] نامیده می‌شود. برای رمزگشایی یک متن رمز، باید بدانیم از کدام رمز استفاده شده است، و کدام **کلید رمزگذاری**[۲] با آن به کار گرفته شده است. برای رمز جابجایی، کلید تعداد حروفی در الفبا است که هر حرف جابجا می‌شوند. با دانستن رمز و کلید، می‌توانیم رمزگذاری را خنثی و داده‌های اصلی که **متن ساده**[۳] نامیده می‌شوند را بازیابی کنیم.

> **کد مخفی**: با ورق زدن صفحات یک کتاب قدیمی از کتابخانه‌ی مادربزرگتان، با این پاورقی دست نویس روبرو می‌شوید:
>
> MAXI KBVX HYLX VNKB MRBL XMXK GTEO BZBE TGVX VTKX EXLL VHFF NGBV TMBH GLVH LMEB OXL
>
> او به شما می‌گوید در زمان نوجوانی با رمز جابجایی سرگرم بوده است، اما نمی‌تواند به یاد بیاورد که این نوشته‌ی رمزآلود چه معنایی دارد. آیا می‌توانید متن ساده را بازیابی کنید؟

ساده‌ترین راه برای شکستن رمز جابجایی این است که کلیدهای مختلف را بر روی متن رمز آزمایش کنید و ببینید آیا خروجی منطقی است یا خیر. فقط ۲۵ امکان وجود دارد! می توانید راه حل را در پیوست II بیابید.

رمز زیگزاگی

راهی برای برهم زدن متن ساده وجود دارد که نیازی به جایگزینی حروف ندارد. در عوض، حروف به روشی از پیش تعیین‌شده به هم‌ریخته می‌شوند. به عنوان مثال، پیام زیر را در نظر بگیرید:

We were found. Flee at once.

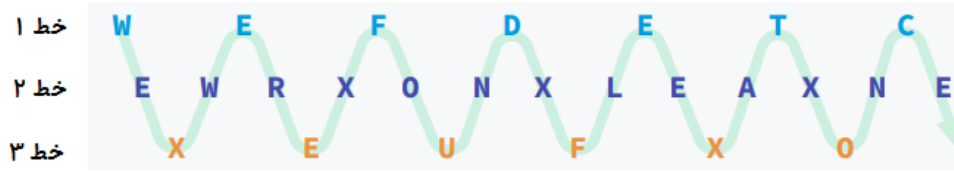
---

[۱] Ciphertext
[۲] Encryption Key
[۳] Plaintext

بیایید ابتدا همه‌ی قالب‌بندی‌های غیرضروری را حذف کنیم، زیرا می‌توانند سرنخ‌هایی درباره‌ی رمز ارائه دهند. به طور معمول، تمام حروف به صورت بزرگ نوشته می‌شوند، علائم نگارشی حذف می‌شوند و فاصله‌ها یا حذف شده یا با X جایگزین می‌شوند:

```
WEXWEREXFOUNDXFLEEXATXONCE
```

اکنون هر حرف را روی خطی متفاوت از حرف قبلی می‌نویسیم. برای مثال، بیایید یک الگوی زیگزاگی را در سه خط دنبال کنیم:



متن رمز نهایی با به هم پیوستن سه خط به هم به دست می‌آید:

```
WEFDETC
EWRXONXLEAXNE → WEFDETCEWRXONXLEAXNEXEUFXO
XEUFXO
```

این **رمز زیگزاگی**[۱] نامیده می‌شود و کلید رمزگذاری آن تعداد خطوط استفاده شده توسط الگو است. این رمز نیز مشابه رمز جابجایی شکننده است: فقط تعداد محدودی کلید امکان‌پذیر وجود دارد. با آزمایش روند معکوس برای هر تعداد ممکن از خطوط، می‌توان به راحتی رمز را شکست.

رمز جابجایی

بیایید به رمزهایی برگردیم که حروف را در متن ساده جایگزین می‌کنند. رمزی که قانون متفاوتی برای جایگزینی هر حرف دارد از رمز جابجایی ایمن‌تر است. به عنوان مثال، می‌توانید با یک دوست ملاقات کرده و در مورد نگاشت زیر برای جایگزینی حروف به توافق برسید.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
V H I E R P X N D J F T G L B W O Q K Z M U C S Y A
```

---

[۱] Zigzag Cipher

این یک **رمز جایگزینی ساده**[۱] است. در این رمز عبارت ENERGY به صورت RLRQXY رمزگذاری می‌شود. کلید رمزگذاری نگاشت بالا است که نحوه‌ی جایگزینی حروف را مشخص می‌کند. در این نوع رمز 26! راه برای برهم زدن الفبا و ایجاد کلیدهای مختلف وجود دارد[۲]. این بدان معناست که کلیدهای متمایز این نوع رمز از تعداد قطرات آب اقیانوس‌ها بیشتر هستند. امتحان کردن یک به یک همه‌ی این کلیدها برای یک نفر غیرعملی است.

با این وجود، این نوع رمز الگوهایی را در متن رمزگذاری‌شده به‌جا می‌گذارد که قابل تجزیه و تحلیل هستند. با کمی سعی و خطا، شکستن این رمز به‌طور شگفت‌انگیزی آسان است، به‌ویژه زمانی که کامپیوتر می‌تواند به ما در شمارش حروف و تشخیص کلمات فرهنگ لغت کمک کند. به عنوان مثال، در متون معمولی انگلیسی، E پرتکرارترین حرف است. اگر پرتکرارترین حرف در یک متن رمز R باشد، به احتمال زیاد E → R در کلید وجود دارد. علاوه بر این، TH زوج حرفی است که اغلب با هم در انگلیسی ظاهر می‌شوند. اگر ZN متداول‌ترین زوج حـــرف متن رمز باشد، می‌تـــوانیم حدس بزنیم که T → Z و H → N.

این کار **تحلیل تکرار**[۳] نامیده می‌ود و نقطه‌ی شروع خوبی برای آزمایش کلیدهای رمزگذاری مختلف است. سعی کنید از آن برای حل مشکل زیر استفاده کنید:

> **سکه‌ی توخالی**: یک سکه‌ی قدیمی پیدا کرده‌اید که بسیار سبک به نظر می‌رسید. آن را روی زمین انداختید، باز شد و یک تکه‌ی کاغذ کوچک نمایان شد. با ذره بین، یادداشت زیر را خوانده‌اید:
>
> DUA KVYBVHA PVJ OAZQMASAO DW CWES PQLA KASJWFVZZC. AMASCDUQFH QJ VZZ SQHUD PQDU DUA LVGQZC. PA PQJU CWE JEYYAJJ. HSAADQFHJ LSWG DUA YWGSVOAJ.
>
> آیا می‌توانید رمز را بشکنید؟

این کار در ابتدا می‌تواند دلهره‌آور به نظر برسد، اما با یک قلم، کاغذ و کمی صبر، امکان پذیر است. از آن‌جایی که متن ساده انگلیسی فرض می‌شود، با یافتن حروف رمزگذاری‌شده که به احتمال زیاد

---

[۱] Simple Substitution Cipher

[۲] علامت تعجب نشان‌دهنده‌ی فاکتوریل است: $26! = 26 \times 25 \times \cdots \times 2 \times 1$.

[۳] Frequency Analysis

جایگزین E و TH شده‌اند، شروع می‌کنیم. گام به گام، کلمات نیمه‌حل شده را با کلمات رایج انگلیسی مقایسه می‌کنیم تا ببینیم چگونه جایگزینی‌های مختلف ممکن بر بقیه‌ی متن تأثیر می‌گذارند. یکی از راه‌های ممکن برای حل این مسئله در پیوست III آورده شده است.

جایگزینی‌های پیشرفته: رمزهای جایگزینی قوی‌تر ممکن است یک حرف متن ساده را به نمادهای مختلفی تبدیل کنند. به عنوان مثال، متن ساده‌ی E را می توان با R، $، یا * جایگزین کرد. با داشتن گزینه‌های جایگزینی بیشتر برای پرتکرارترین حروف انگلیسی، تجزیه و تحلیل تکرار دشوارتر می‌شود. با این حال، کلمات رایج و زوج حروف همچنان الگوهای ظریفی را در متن رمزی باقی می‌گذارند. نمادهای بیشتری را می‌توان برای جایگزینی زوج حروف و کلمات رایج ابداع کرد تا تحلیل تکرار را سخت‌تر کنند، اما هرگز آن را شکست نمی‌دهند.

رمزهای ترکیبی

به استفاده از ترکیبی از رمزها را **رمز ترکیبی**[۱] می‌گویند. این نوع رمز وقتی که از ترکیب رمزهای جابجایی و جایگزینی به دست می‌آید، کارایی بیشتری دارد. به عنوان مثال، یک متن ساده می‌تواند از طریق رمز زیگزاگی و سپس از طریق رمز جایگزینی ساده تغییر کند. رمز ترکیبی به دست آمده از هر یک از اجزای آن قوی‌تر است.

فرض کنید همانطور که برای متون انگلیسی انتظار می‌رود، TH پرتکرارترین زوج حرف متن ساده است، اگر TH با ZN جایگزین شود، متداول‌ترین زوج حرف متن رمز شده، ZN نخواهد بود، زیرا به‌دلیل استفاده از رمز زیگزاگی این زوج دیگر در کنار هم قرار نمی‌گیرند. با این حال، هر حرف رمزگذاری‌شده همیشه از همان حرف متن ساده به دست می‌آید، بنابراین تجزیه و تحلیل تکرار هنوز امکان‌پذیر است.

در طی جنگ جهانی اول، آلمانی‌ها از رمز ترکیبی برای برقراری ارتباط از طریق رادیو استفاده می‌کردند. رمز دارای یک مرحله‌ی جایگزینی و به دنبال آن یک مرحله‌ی درهم‌ریزی بود. آلمانی‌ها معتقد بودند که این رمز نشکن است. با این حال، فرانسوی‌ها موفق شدند آن را بشکنند و دشمنان خود را استراق سمع کنند. متفقین با اطلاع از برنامه‌های آلمان از قبل، قادر به پیش‌بینی حملات و مدیریت بهتر منابع خود بودند.

---

[۱] Product Cipher

رمز ویژنر

تا به حال، رمزهایی را دیده‌ایم که بر اساس یک نگاشت، بین متن رمز و حروف متن ساده جایگزینی انجام می‌دهند. رمزهای جایگزینی قوی‌تر از نگاشت‌های *متعدد* استفاده می‌کنند. یکی از راه‌های انجام این کار، تکرار فهرستی از رمزهای جابجایی برای هر دنباله از حروف متوالی از متن ساده است. به عنوان مثال، با استفاده از کلید 2-0-1-1-0-6-4، می‌توانیم حروف را به صورت زیر جایگزین کنیم[۱]:

T H E R E F O	R E T H E Y L	O A T H E D T	H E F T
2-0-1-1-0-6-4	2-0—1-1-0-6—4	2-0-1-1-0-6—4	2-0-1-1-
V H F S E L S	T E U I E E P	Q A U I E J X	J E G U

به این نوع رمز، **رمز ویژنر**[۲] می‌گویند. توجه داشته باشید که حرف E در متن ساده را می‌توان به صورت F، E، G، K یا I رمزگذاری کرد. با استفاده از کلیدهای طولانی‌تر، هر حرف متن ساده جایگزین‌های احتمالی بیشتری دارد. در نتیجه، رمز ویژنر می‌تواند تهدید تجزیه و تحلیل تکرار را کاهش دهد. این رمز در قرن شانزدهم اختراع شد و بیش از ۳۰۰ سال شکسته نشد. حتی بسیاری معتقد بودند که غیرقابل شکستن است.

امروزه رمز ویژنر را می‌توان به راحتی توسط کامپیوتر شکست[۳]. با مشاهده‌ی توالی‌های مکرر حروف در متن رمزی، می‌توان طول کلید رمزگذاری را حدس زد. به عنوان مثال، توجه کنید که در مثال ما، الگوی UIE تکرار شده است. رخداد دوم در هفت موقعیت بعد از مورد اول ظاهر می‌شود:

V H F S E L S T E **U I E** E P Q A **U I E** J X J E G U
1 2 3 4 5 6 7

این احتمال وجود دارد الگویی که دو بار در یک متن رمز ویژنر ظاهر می‌شود، با الگویی مطابقت داشته باشد که دو بار در متن ساده‌ی اصلی نیز ظاهر شده است. با این حال، این پدیده تنها در صورتی اتفاق می‌افتد که فاصله‌ی بین دو رخداد برابر با مضربی از طول کلید رمزگذاری باشد. از آن‌جایی که دو

---

[۱] کلیدهای ویژنر اغلب با حروف به جای اعداد بیان می‌شوند. این امر اجازه می‌دهد کلیدهای کوتاه‌تری داشته باشید، زیرا اعداد ۰ تا ۲۵ با استفاده از یک کاراکتر A تا Z نوشته می‌شوند. به عنوان مثال، کلید 2-14-19-15-24-17 را می‌توان به صورت RYPTOC نوشت.

[۲] Vigenère Cipher

[۳] رمز ویژنر اولین بار در سال ۱۸۴۵ توسط چارلز بابیج شکسته شد، همان شخصی که اولین کامپیوتر قابل برنامه‌ریزی را در سال ۱۸۳۷ طراحی کرده بود.

دنباله‌ی UIE در فاصله‌ی هفت موقعیت از هم ظاهر می‌شوند، احتمال زیادی وجود دارد که کلید دارای هفت عدد باشد. اگر مثلاً پانزده موقعیت از هم فاصله داشتند، نشانه‌ی این بود که کلید سه، پنج یا پانزده عدد دارد.

زمانی که طول کلید را حدس زدیم، می‌توان از تحلیل تکرار استفاده کرد. اگر طول کلید هفت باشد، با بازیابی دنباله‌ی حروف هفتم در متن رمز شروع می‌کنیم. در گروه حاصل از این حروف، پرتکرارترین حروف را پیدا می‌کنیم: احتمالاً این حرف جایگزین حرف E در متن ساده شده است.

اگر متن به اندازه‌ی کافی طولانی باشد، تکرار این فرآیند با شش حرف ابتدایی دیگر برای شکستن رمز کافی است. حتی اگر متن رمز مانند مثال ما کوتاه باشد، تحلیل تکرار محتمل‌ترین کلیدها را برای آزمایش به ما می‌گوید و راه حل را می‌توان با قدرت محاسباتی کمی با مقیاس‌های امروزی پیدا کرد[۱].

رمز ورنام

زمانی که طول کلید رمزگذاری به اندازه‌ی تعداد حروف موجود در متن ساده باشد، رمز ویژنر امن‌ترین رمز است. ما این نسخه را **رمز ورنام**[۲] می‌نامیم. ریاضیدانان ثابت کرده‌اند که شکستن رمز ورنام تا زمانی که کلید به طور تصادفی انتخاب شده و فقط یک بار استفاده شود، غیرممکن است. در غیر این صورت، الگوها ظاهر می‌شوند.

دو متن رمز VSXOCQZQCLGHQGBFHTJK و NHPJEXGPTDDBFCFOYKRA را در نظر بگیرید. اگر هر دو با همان کلید رمزگذاری ورنام رمزگذاری شده باشند، ترفندی وجود دارد که می‌توانیم اعمال کنیم. ابتدا یک کلمه رایج را انتخاب می‌کنیم که احتمالاً در متن ساده وجود دارد و به آن **کریب**[۳] می‌گوییم. فرض می‌کنیم کریب در یکی از متن‌های ساده قرار دارد و بررسی می‌کنیم که چگونه روی دیگری تأثیر می‌گذارد. بیایید عبارت tomorrow را به عنوان کریب امتحان کنیم:

متن رمز ۱	V	S	X	O	C	Q	Z	Q	C	L	G	H	Q	G	B	F	H	T	J	K
	T	O	M	O	R	R	O	W												
	2	4	11	0	11	25	11	20												
متن رمز ۲	N	H	P	J	E	X	G	P	T	D	D	B	F	C	F	O	Y	K	R	A
	L	D	E	J	T	Y	V	V												

شکل ۳-۱: اگر حرف اول متن ساده ۱ برابر با T باشد، اولین عدد کلید باید ۲ باشد، به طـوری که T→V. با این فرض، حرف اول متن ساده ۲ باید L باشد، به طوری که L→N.

---

[۱] ابزاری برای شکستن متن رمز ویژنر در http://code.energy/vigenere وجود دارد.

[۲] Vernam Cipher

[۳] Crib

اگر tomorrow در ابتدای متن اول بود، متن ساده‌ی دوم باید با ldejtyvv شروع می‌شد. با فرض اینکه هیچ یک از متن‌های ساده حاوی حرف‌های بیهوده نباشد، کلمه‌ی tomorrow نمی‌تواند در ابتدای متن ساده‌ی اول باشد. در ادامه، کریب را در موقعیت‌های دیگر امتحان می‌کنیم. هنگام امتحان کردن موقعیت ۷، یک موفقیت به دست می‌آوریم:

متن رمز ۱	V	S	X	O	C	Q	Z	Q	C	L	G	H	Q	G	B	F	H	T	J	K
							T	O	M	O	R	R	O	W						
							6	2	16	23	15	16	2	10						
متن رمز ۲	N	H	P	J	E	X	G	P	T	D	D	B	F	C	F	O	Y	K	R	A
							A	N	D	G	O	L	D	S						

شکل ۳-۲: احسنت! با آزمایش tomorrow در موقعیت هفتم اولین متن ساده، یک بخش از کلید را استخراج می‌کنیم که متن رمز دیگر را به متنی قابل فهم رمزگشایی می‌کند و نه حروف بی‌معنی. این امر تایید می‌کند که حدس درست بوده است.

به این ترفند **کشیدن کریب**[۱] می‌گویند. برای اینکه رمز ورنام در برابر آن مصون بماند، هرگز نباید از یک کلید رمزگذاری روی دو متن ساده‌ی متفاوت استفاده کنیم. به همین دلیل، ما معمولاً کلید رمزگذاری ورنام را یک **کلید یکبار مصرف**[۲] می‌نامیم.

در دهه‌ی ۱۹۴۰، ارتش شوروی گاهی اوقات از کلیدهای یکبار مصرف برای ارسال متون رمزی ورنام دو بار استفاده می‌کرد. سازمان اطلاعات آمریکا موفق شد مخابره‌های رادیویی آن‌ها را رهگیری کند و در نهایت با کشیدن کریب کد را شکست. آن‌ها از جمله کشف کردند که جاسوسان شوروی در برنامه‌ی تسلیحات هسته‌ای آن‌ها نفوذ کرده‌اند.

تا به امروز، رمز ورنام استاندارد طلایی برای ارتباطات مخفی است. این تنها رمزی است که ثابت شده است در صورت استفاده‌ی صحیح غیرقابل شکستن است. اما این کار چندان عملی نیست: طرفین قبل از این‌که بتوانند پیام‌های رمزگذاری شده را مبادله کنند، باید یک دنباله‌ی یکسان و بزرگ از اعداد تصادفی مخفی را به اشتراک بگذارند. یک رمز امن که با کلیدی کوتاه‌تر کار کند، بسیار بهتر خواهد بود.

---

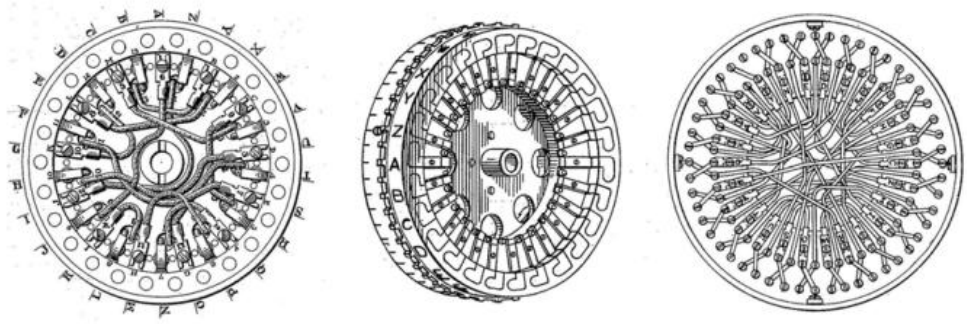
[۱] Crib-Dragging
[۲] One-Time Pad

ماشین‌های رمز

در دهه‌های ۱۹۲۰ و ۱۹۳۰، قدرت‌های نظامی در سرتاسر جهان نیاز داشتند که پیام‌ها را بدون دردسر به اشتراک گذاشتن کلیدهای بزرگ رمزگذاری، رمزگذاری و رمزگشایی کنند. بنابراین، آن‌ها ماشین‌هایی طراحی کردند که شکل‌های کوچک‌تری از رمزهای مشترک را به‌عنوان جانشین کلیدهای بزرگ، در میان طرف‌های درگیر در ارتباط اجرا می‌کردند. دستگاه‌های جدید از ترفندهایی استفاده می‌کردند که اجازه می‌داد رمزهای کوتاه به اشتراک گذاشته شده به طور مداوم به کلیدهای رمزگذاری بزرگ‌تر و بزرگ‌تر به نام **رشته‌کلید**[۱] گسترش یابند.

این ماشین‌های رمز اولیه از چرخ‌های متوالی ساخته می‌شدند که هر کدام سیم‌کشی داخلی پیچیده‌ی خود را داشتند (شکل ۳-۳). قبل از رمزگذاری یا رمزگشایی هر حرف، چرخ‌ها به طور هماهنگ به موقعیت جدیدی جابجا می‌شدند. در هر موقعیت، سیم‌کشی‌های آن‌ها یک مدار کاملاً متفاوت را تشکیل داده و رشته‌لید را با یک نگاشت *به ظاهر تصادفی* جدید گسترش می‌دادند.

به نظر می‌رسید پیام‌ها با استفاده از کلیدهای رمزگذاری تصادفی بی‌نهایت طولانی رمزگذاری شده‌اند. در واقع، رشته‌کلید تصادفی نبود، بلکه **شبه‌تصادفی**[۲] بود: همان رشته‌کلید را می‌توان در هر زمان در یک نسخه‌ی متفاوت از دستگاه تکثیر کرد. تنها چیزی که مورد نیاز بود آگاهی از رمز مشترک بود: موقعیت اولیه‌ی چرخ ها[۳].



شکل ۳-۳: چرخ های ماشین رمز. همانطور که این چرخ‌ها می‌چرخند، مدارهای ظاهراً آشفته‌ای را به روشی قطعی و قابل‌تکرار تشکیل می‌دهند.

---

[۱] Keystreams

[۲] Pseudorandom

[۳] ما اغلب از اصطلاح کلید رمزگذاری برای اشاره به رمز مشترک کوتاه به جای رشته‌کلیدی که تولید می‌کند استفاده می‌کنیم.

ماشین‌های رمز به طور گسترده در جنگ جهانی دوم مورد استفاده قرار گرفتند و به پیام‌ها اجازه می‌دادند سریعاً رمزگذاری و رمزگشایی شوند و از رمزهای مشترک کوتاه به جای کلیدهای یک‌بار مصرف طولانی استفاده کنند. با این حال، رشته‌کلیدهای آن‌ها الگوهای ظریفی را نشان می‌داد که توسط کدشکن‌ها در هر دو طرف مورد سوءاستفاده قرار می‌گرفت. تیم‌هایی از ریاضی‌دانان و مهندسان شبانه روز برای شکستن متون رمزی رهگیری شده کار می‌کردند.

تمام ماشین‌های رمزنگاری آلمانی قبل از پایان جنگ شکسته شدند، حتی مدلی که توسط بخش فرماندهی عالی هیتلر استفاده می‌شد. در همین حال، قوی‌ترین ماشین رمزگذاری آمریکایی به نام سیگابا[۱] هرگز شکسته نشد. در سال‌های اخیر، ثابت شده است که الگوها همیشه از رشته‌کلیدی که توسط چرخ‌های چرخان تولید می‌شود پدیدار می‌شوند؛ به این معنی که حتی سیگابا نیز آسیب‌پذیر بوده است.

خوشبختانه، کامپیوتر به ما اجازه می‌دهند تا رمزهای قوی‌تری را اجرا کنیم. اکنون بیایید بررسی کنیم که چگونه کامپیوترها می‌توانند یک قدم از این ماشین‌های رمزگذاری آنالوگ فراتر بروند.

۲-۳- تقارن

امروزه تقریباً تمام فرایند رمزگذاری ما به کامپیوترها متکی است. از آن‌جایی که کامپیوترهای ما بر روی داده‌های دودویی کار می‌کنند، باید رمزهای خود را برای الفبای دو نمادی طراحی کنیم: 0 و 1. برای مثال، رمز ورنام می‌تواند با یک الفبای دودویی به همان اندازه مؤثر کار کند که با یک الفبای ۲۶ حرفی کار می‌کرد:

متن ساده	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0	1	1	1	0	0
کلید	1	0	1	0	1	1	1	1	0	1	0	1	1	1	0	0	1	0	1	0
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
متن رمز	1	0	0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	1	0

شکل ۳-۴: جابجایی متن ساده‌ی دودویی با یک کلید یکبار مصرف دودویی. هنگامی که رقم 1 به جلو منتقل می‌شود، حلقه می‌زند و به 0 برمی‌گردد[۲].

هر اطلاعاتی را که در کامپیوتر ذخیره می‌شود می‌توان کاملاً محرمانه و با استفاده از رمز ورنام رمزگذاری کرد. با این حال، مشکل کلیدهای یکبار مصرف همچنان باقی است. برای رمزگذاری یک

---

[۱] SIGABA

[۲] این فرآیند معادل عملیات بولی متوالی XOR است. می‌توانید در کتاب اول ما، *مبانی و مفاهیم علوم کامپیوتر*، درباره‌ی عملیات بولی اطلاعات بیشتری کسب کنید.

گیگابایت ویدیو، یک گیگابایت اعداد تصادفی مخفی مورد نیاز است؛ و به یاد داشته باشید، کلیدهای یکبار مصرف را نمی‌توان دوباره استفاده کرد: کشیدن کریب به صورت باینری نیز کار می‌کند!

خوشبختانه، رمزهای امنی وجود دارند که می‌توانند حجم زیادی از داده‌ها را با استفاده از یک رمز مشترک نسبتاً کوچک رمزگذاری کنند. این رمزها با توجه به مکانیزم کار زیربنایی خود به دو دسته تقسیم می‌شوند. بیایید ابتدا ساده‌ترین روش را در نظر بگیریم.

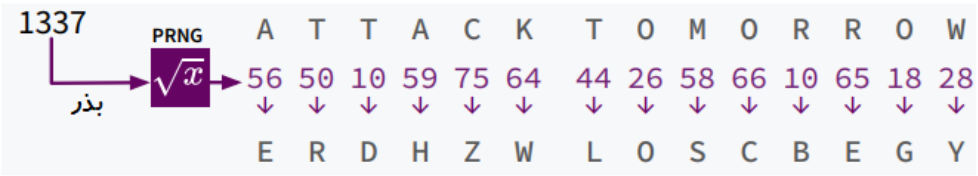
رمزهای رشته‌ای

اولین رویکرد این است که از کلیدهای یکبار مصرف طولانی به روشی مشابه ماشین‌های رمز با چرخ‌های چرخنده اجتناب کنید. با این حال، این بار، یک برنامه‌ی کامپیوتری می‌نویسیم که رشته‌ای بی‌پایان از اعداد شبه‌تصادفی تولید می‌کند. به عنوان مثال، با توجه به عدد غیر مربعی ۱۳۳۷، رشته‌ای از ارقام به ظاهر غیرقابل پیش‌بینی با محاسبه‌ی جذر آن ظاهر می‌شود:

$$\sqrt{1337} = 36.56\ 50\ 10\ 59\ 75\ 64\ 44\ 26\ 58\ 66\ 10\ 65\ 18\ 28\ \ldots$$

این یک نمونه‌ی ابتدایی از **مولد اعداد شبه‌تصادفی**[۱] یا PRNG است. یک PRNG به یک مقدار اولیه یا **بذر**[۲] نیاز دارد تا به عنوان نقطه‌ی شروع آن عمل کند، مشابه رمز مشترکی که موقعیت اولیه‌ی چرخ‌های ماشین رمزگذاری قدیمی را توصیف می‌کند. در مثال ما، بذر عدد ۱۳۳۷ است.

رمزهای رشته‌ای[۳] برای تولید رشته‌کلید به PRNG متکی هستند. بیایید مثال خود را در نظر بگیریم و ارقام بعد از ممیز اعشار را به صورت دوتایی گروه‌بندی کرده تا جریانی از اعداد شبه‌تصادفی بین ۰ تا ۹۹ ایجاد کنیم:



شکل ۳-۵: رمز ورنام در حال اجرا بر روی یک مولد اعداد شبه‌تصادفی. رمز مشترک بین طرفین فقط همان بذر است[۴].

---

[۱] PseudoRandom Number Generator یا PRNG

[۲] Seed

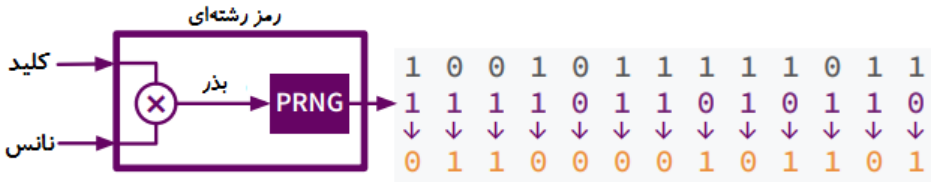
[۳] Stream Ciphers

[۴] در اینجا همچنین توافق شد که ۲۶ مربوط به جابجایی موقعیت صفر، ۲۷ مربوط به جابجایی ۱ و غیره است.

امروزه رمزهای رشته‌ای معمولاً از مولدهایی استفاده می‌کنند که مستقیماً رشته‌ای از ارقام دودویی را تولید می‌کنند. الگوریتم PRNG پایه و عنصر تعیین‌کننده‌ی یک رمز رشته‌ای است. یک رمز رشته‌ای تنها در صورتی ایمن است که PRNG آن اعداد شبه‌تصادفی را بدون الگوهای قابل تشخیص تولید کند. حتی اگر دیگران بدانند از کدام PRNG استفاده می‌شود، بدون دانستن رمز مشترک، تشخیص خروجی PRNG از دنباله‌ی حاصل از پرتاب واقعاً تصادفی سکه‌ها برای ایشان غیرممکن است.

تا به امروز، ما نمی‌دانیم که آیا یک PRNG آسیب‌ناپذیر وجود دارد یا خیر. آسیب‌پذیری‌هایی در بسیاری از مولدهای پیشنهاد شده توسط کارشناسان در گذشته یافت شده است. قبل از انتخاب رمز رشته‌ای، بررسی کنید که کدام یک توسط متخصصان بررسی شده و برای کدام یک نقطه‌ی ضعفی گزارش نشده است[۱]. طراحی یک PRNG خوب بسیار چالش‌برانگیز است، بنابراین توصیه می‌کنیم به موارد موجود پایبند باشید، مگر اینکه خودتان بخواهید متخصص رمزنگاری شوید.

نانس: از آنجایی که رشته‌کلیدها همان هدفی را دارند که کلیدهای یکبار مصرف دارند، هرگز نباید بیش از یک بار از آن‌ها استفاده کرد. یک PRNG زمانی که بذر یکسانی به آن داده شود، همیشه همان رشته‌کلید را تولید می‌کند، بنابراین یک بذر هرگز نباید دوباره استفاده شود. اگر رمز مشترک به‌طور مستقیم به‌عنوان بذر مورد استفاده قرار گیرد، طرف‌های ارتباطی باید بتوانند قبل از هر ارتباط بر سر رمز جدیدی به توافق برسند. این کار ممکن است غیر عملی باشد، بنابراین رمزنگاران ترفندی ابداع کردند که به ما امکان می دهد از یک رمز مشترک استفاده‌ی مجدد کنیم. این ترفند از یک **نانس**[۲] استفاده می‌کند: یک عدد دلخواه، یکبار مصرف و غیر مخفی که با رمز مشترک ترکیب می‌شود تا یک بذر تولید کند.



شکل ۳-۶: رمزهای رشته‌ای معمولاً به یک رمز مشترک (که به آن کلید گفته می‌شود) و یک نانس نیاز دارند. بر خلاف کلید، نانس لازم نیست مخفی باشد. اگر از نانس‌ها هرگز دوباره استفاده نشود، می‌توان از یک کلید برای رمزگذاری چندین متن ساده استفاده کرد.

---

[۱] رمزهای پیشنهادی را در اینجا بررسی کنید: http://code.energy/stream.

[۲] Nonce

قبل از اینکه هر چیزی را با رمز رشته‌ای رمزگذاری کنید، باید یک نانس انتخاب کنید. اگر یک عدد تصادفی انتخاب می‌کنید، مطمئن شوید که بعید است در آینده دوباره آن را انتخاب کنید. به عنوان مثال، می‌توانید یک عدد تصادفی ۶۴ بیتی ایجاد کنید: از آنجایی که بیش از *صد میلیون تریلیون* احتمال مختلف وجود دارد، شانس دوباره انتخاب یک نانس یکسان تقریباً صفر است.

هنگام ارسال پیام رمزگذاری شده با رمز رشته‌ای، نانس رمزگذاری‌نشده را همراه با متن رمز ارسال کنید. گیرنده ابتدا نانس ورودی را با کلید ترکیب کرده، سپس کلید یکبار مصرف مصنوعی را دوباره ایجاد و در نهایت متن رمز را برای به دست آوردن متن ساده به عقب برمی‌گرداند. حتی اگر هکرها تعداد زیادی متن رمز را رهگیری کنند و نانس هر یک را نیز بدانند، نمی‌توانند هیچ الگویی بین آن‌ها شناسایی کنند زیرا هر کدام با رشته‌کلید خود رمزگذاری شده‌اند!

انتخاب کلید: مهاجمی که متن رمز شما را رهگیری می‌کند، می‌تواند سعی کند آن را با حمله‌ی همه‌جانبه[۱] بشکند: تمام کلیدهای ممکن را تا زمانی که یک متن ساده قابل فهم پیدا شود، امتحان کند. بدبین باشید: فرض کنید مهاجم شما می‌داند از کدام رمز استفاده می‌کنید و چندین برابر کل قدرت محاسباتی جهان را در اختیار دارد. اگر یک کلید رمزگذاری تصادفی به اندازه‌ی کافی طولانی انتخاب کنید، خود را در برابر هرگونه حمله‌ی همه‌جانبه احتمالی بیمه خواهید کرد. به عنوان مثال، یک کلید تصادفی ۱۲۰ بیتی، هیچ شانسی برای مهاجم قدرتمند باقی نمی‌گذارد که متون رمز شما را بشکند[۲].

حمله‌ی جعل: فرض کنید ایدا یک پیام رمزگذاری شده به بانک خود ارسال کرده و درخواست انتقال ۱۰۰ دلار به حساب اندرو می‌کند؛ و فرض کنید پیام از یک فرمت استاندارد پیروی می‌کند، به طوری که بیت های ۵ تا ۹ شماره حساب مقصد را رمزگذاری می‌کنند. فرض کنید چارلز همه‌ی این‌ها را می‌داند و مسئول انتقال پیام رمزگذاری شده از ایدا به بانک است. اگر چارلز بداند که شماره حساب بانکی اندرو ۱۰۰۱ است، می‌تواند پیام ایدا را تغییر دهد تا واریز به شماره حساب خودش، یعنی حساب ۱۱۱۰۰ انجام شود:

					0	1	0	0	1	شماره حساب اندرو						
متن رمز اصلی	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0
					0	0	1	1	0	بخشی از کلید						
					1	1	1	0	0	شماره حساب چارلز						
متن رمز جعلی	0	1	1	1	**1**	**1**	**0**	**1**	**0**	1	0	0	0	1	1	1	0

[۱] Brute-Force

[۲] برای داشتن یک شانس ۰.۱ درصدی برای حدس زدن یک کلید تصادفی ۱۲۰ بیتی، مهاجم باید ۱۰۶ میلیارد کلید در ثانیه را برای ۴۰۰ تریلیون سال امتحان کند، تقریباً سی هزار برابر سن جهان.

اگر مهاجم بتواند بخش‌هایی از متن رمز را تنها با دانستن قسمت‌های متناظر متن اصلی جعل کند، می‌گوییم که رمز قابل جعل است. همانطور که در ادامه خواهیم دید، می‌توانیم رمزهایی ابداع کنیم که قابل جعل نیستند.

رمزهای قطعه‌ای

در رمزهای رشته‌ای، همیشه یک رابطه‌ی واضح بین متن ساده و متن رمز وجود دارد: اگر دهمین بیت متن ساده را برگردانید (تغییر آن به 1 اگر 0 باشد و به 0 اگر 1 باشد)، باعث برگردانده‌شدن بیت دهم از متن رمز می‌شود. وجود این رابطه مطلوب نیست. در حقیقت، جعل رمزهای رشته‌ای نتیجه‌ی این رابطه است. یک رمز در صورتی امن‌تر است که هیچ رابطه‌ی آشکاری بین متن رمز و متن ساده ایجاد نکند.

رمز ترکیبی قدیمی را به یاد بیاورید که رمزهای زیگزاگی و جایگزینی ساده را ترکیب می‌کرد. این رمز قوی‌تر بود زیرا درهم‌آمیزی زیگزاگی رابطه‌ی بین متن رمز و متن ساده را نامفهوم می‌کرد. اما رمز زیگزاگی به اندازه‌ی کافی خوب نیست: در حالت مطلوب، رابطه‌ی بین متن ساده و متن رمز باید چنان نامفهوم باشد که تغییر یک بیت از متن ساده باعث تغییر کل متن رمز شود.

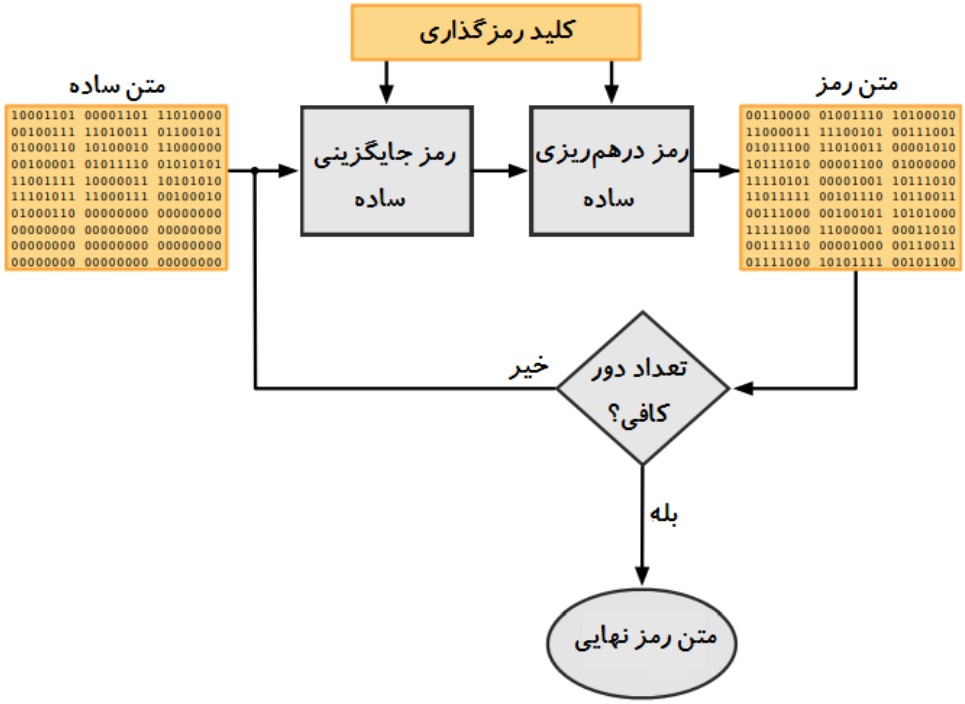
در هنگام استفاده از رمزهای رشته‌ای، تغییر یک بیت از کلید منجر به متن رمز کاملاً متفاوتی می‌شود. از زمان جنگ جهانی دوم، رمزنگارها در حال توسعه‌ی رمزهایی بوده‌اند که تغییر یک بیت از کلید یا متن ساده، منجر دگرگونی کل متن رمز شود.

این هدف را می‌توان با استفاده از رمزگذاری ترکیبی که عملیات جایگزینی و به‌هم‌ریختن را ترکیب می‌کند، به دست آورد. با این حال، حذف همه‌ی جعل‌های ممکن و قابل مشاده در بخش‌های مختلف متن رمز، ساده نیست: رمز ترکیبی باید چندین بار اعمال شود، و عملیات جایگزینی و به‌هم‌ریختن آن باید با دقت انتخاب شوند. در واقع، فرایند رمزگذاری ترکیبی مناسب تنها در شرایطی ممکن است که متن ساده همیشه اندازه‌ی ثابت و یکسانی داشته باشد.

رمزهایی که بر اساس این اصل عمل می‌کنند، **رمزهای قطعه‌ای**[1] نامیده می‌شوند، زیرا متن ساده باید در قطعاتی با اندازه‌ی ثابت رمزگذاری شود. اگر متن ساده از اندازه قطعه طولانی‌تر باشد، باید به چند بخش تقسیم شود. اگر یک متن ساده به اندازه‌ی کافی طولانی نباشد که یک قطعه‌ی کامل را پر کند، معمولاً با افزدون تعداد مناسبی بیت صفر پر می شود. در حال حاضر، رایج‌ترین رمزهای قطعه‌ای مورد استفاده با قطعات ۱۲۸ یا ۲۵۶ بیتی کار می‌کنند.

---

[1] Block Cipher



شکل ۳-۷: طرح ساده‌شده‌ی رمز قطعه‌ای. به طور معمول، رمزهای قطعه‌ای باید رمز ترکیبی اصلی خود را حدود ده بار تکرار کنند.

رمزهای قطعه‌ای قابل جعل نیستند. اگر یک بیت در یک قطعه از متن رمز تغییر کند، کل پیام بعد از رمزگشایی به داده‌های مخدوش تبدیل می‌شود. به همین دلیل، حتی اگر بیت‌هایی از متن ساده معلوم باشند، مهاجم نمی‌تواند متن رمز را تغییر دهد و یک پیام جعلی ایجاد کند.

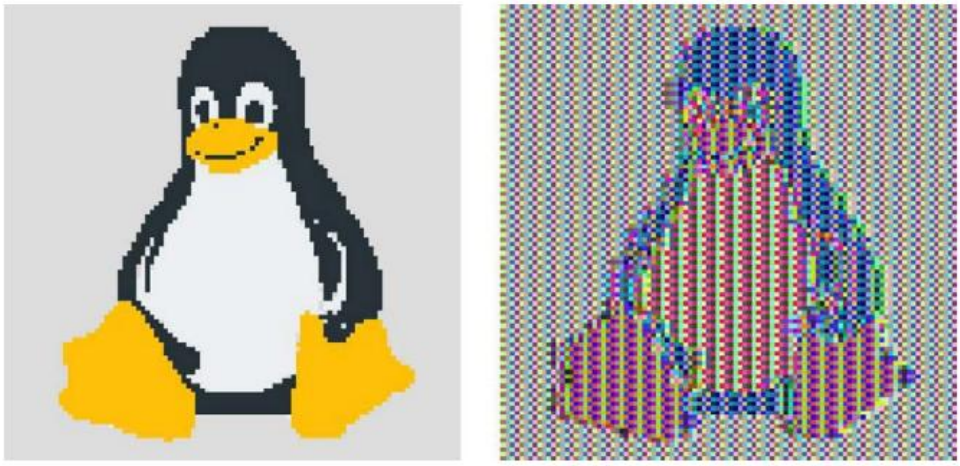
روش‌های مختلفی برای تقسیم و رمزگذاری متن‌های ساده‌ی طولانی‌تر از اندازه‌ی قطعه وجود دارد. ساده‌ترین راه برای انجام این کار آن است که متن ساده را به قطعاتی با اندازه‌ی مناسب تقسیم کرده سپس رمز قطعه‌ای را مستقیماً روی هر یک از آن‌ها اعمال کنید. در این شرایط می‌گوییم که متن رمز به دست آمده در حالت **کتاب کد الکترونیکی**[۱] یا ECB رمزگذاری شده است.

متأسفانه، ECB یک مشکل دارد: یک مهاجم می‌تواند بفهمد آیا یک قطعه‌ی متن ساده دو بار با استفاده از یک کلید، رمزگذاری شده است یا خیر؛ زیرا به یک قطعه‌ی متن رمز مشابه منجر می‌شود. به عنوان مثال، اگر یک مهاجم بداند که یک قطعه از متن رمز به بانک می‌گوید ۱۰۰ دلار به حسابش واریز کند، می‌تواند هر بار که ۱۰۰ دلار اضافی بخواهد، آن پیام را دوباره ارسال کند.

---

[۱] Electronic CodeBook یا **ECB**

مراقب باشید: این آسیب‌پذیری همچنین می‌تواند به مهاجم اجازه دهد اطلاعاتی در مورد یک متن ساده‌ی طولانی رمزگذاری شده در حالت ECB به دست آورد. به عنوان مثال، فرض کنید می‌خواهیم یک فایل تصویری را رمزگذاری کنیم. وقتی بخش‌هایی از فایل تصویر را به قطعاتی از متن ساده تقسیم می‌کنیم، بخش‌هایی با سایه و رنگ یکسان، بلوک‌های یکسانی از متن رمز را ایجاد می‌کنند:



شکل ۳-۸: تصویری که با استفاده از رمز قطعه‌ای در حالت ECB رمزگذاری شده است. الگوهای متن ساده (سمت چپ) تکرار می‌شوند، و نحوه‌ی قرارگیری آن قطعات تکرارشده در متن رمز (راست) ماهیت محتوای آن را آشکار می‌کند.

برای رفع این مشکل، متن ساده را تغییرشکل می‌دهیم تا مطمئن شویم هر قطعه‌ی متن ساده دارای یک مقدار منحصر به فرد است. راه‌های بسیاری برای انجام این کار وجود دارند. یکی از این راه‌ها آن است که از یک نانس به بزرگی یک قطعه استفاده کنید که به آن **بردار مقداردهی اولیه**[۱] یا IV می‌گویند:

- با تغییرشکل اولین قطعه‌ی متن ساده با اعمال نانس به عنوان یک کلید بار مصرف (همانطور که در ابتدای این بخش نشان داده شد) شروع می‌کنیم. سپس این متن ساده‌ی تبدیل‌شده با کلید رمز قطعه‌ای به درستی رمزگذاری می‌شود.
- برای هر بلوک بعدی، از بلوک قبلی متن رمز به عنوان کلید مصرف برای تغییر شکل متن ساده استفاده می‌کنیم. سپس هر متن ساده‌ی تبدیل‌شده را می‌توان به طور ایمن با همان کلید رمز قطعه‌ای رمزگذاری کرد.

---

[۱] Initialization Vector یا IV

هنگامی که از یک رمز قطعه‌ای به این روش استفاده می‌شود، می گوییم که در حالت زنجیره‌ی رمز قطعه[۱] یا CBC کار می‌کند. شما نیازی به کدنویسی هیچ یک از این تبدیلات متن ساده ندارید. در عوض، می‌توانید کتابخانه‌ی رمزنگاری خود را طوری پیکره‌بندی کنید که رمز قطعه‌ای شما با استفاده از حالتی که می‌خواهید عمل کند. حالت‌های دیگری نیز وجود دارند که از تله‌های حالت ECB جلوگیری می‌کنند، اما اگر نمی‌دانید در حال انجام چه کاری هستید، توصیه می‌کنیم به حالت CBC پایبند باشید!

همانند رمزهای رشته‌ای، آسیب‌پذیری‌هایی در بسیاری از رمزهای قطعه‌ای پیشنهاد شده توسط کارشناسان تاکنون یافت شده است. قبل از اینکه رمز قطعه‌ای را برای استفاده انتخاب کنید، بررسی کنید که کدام یک در حال حاضر توسط رمزنگاران توصیه می‌شود[۲].

رمز قطعه‌ای در مقابل رمز رشته‌ای: رمزهای قطعه‌ای به قدرت محاسباتی بیشتری نسبت به رمزهای رشته‌ای نیاز دارند. هنگامی که امنیت اضافی رمزهای قطعه‌ای مورد نیاز نیست، و متن ساده به صورت پویا ایجاد می‌شود (مثلاً رمزگذاری همزمان یک تماس تلفنی)، بهتر است یک رمز رشته‌ای انتخاب کنید. رمزهای رشته‌ای اجازه می‌دهند رمزنگاری متن ساده و ارسال متن رمزشده همزمان انجام شوند، در حالی که رمزهای قطعه‌ای بخش‌های بزرگ‌تری از داده‌ها را باید قبل از ارسال رمزگذاری کنند.

رمزهای رشته‌ای و رمزهای قطعه‌ای بخشی از یک دسته‌ی بزرگ‌تر به نام **رمزهای متقارن**[۳] هستند، زیرا در این رمزها هر دو طرف ارتباط باید یک نسخه از یک کلید واحد را داشته باشند. به عبارت دیگر، تنها در صورتی می‌توانید ارتباطی را با رمز متقارن ایمن کنید که قبلاً بر روی یک رمز مشترک با همتای خود توافق کرده باشید.

۳-۳- عدم تقارن

قبل از اینکه بتوانیم با استفاده از یک رمز متقارن ارتباط برقرار کنیم، باید راهی امن برای به اشتراک‌گذاری یک کلید رمزگذاری مخفی با همتای خود پیدا کنیم. در گذشته، افراد مجبور بودند شخصاً با یکدیگر ملاقات کنند یا کلیدهای خود را از طریق یک پیک مورد اعتماد ارسال کنند. امروزه راه‌هایی برای دو نفری که هرگز با یکدیگر ملاقات نکرده‌اند، وجود دارد تا *بدون* استفاده از یک پیوند ارتباطی امن، یک کلید مخفی را به اشتراک بگذارند.

---

[۱] **C**ipher **B**lock **C**haining یا **CBC**

[۲] در حال حاضر، بهترین رمزهای قطعه‌ای توصیه شده AES و Twofish با اندازه قطعه‌ی ۲۵۶ بیت هستند.

[۳] Symmetric Ciphers

تبادل کلید دیفی- هلمن

فرض کنید ایدا می‌خواهد یک پیام مخفی برای چارلز بفرستد، اما آن‌ها هنوز در مورد یک رمز مشترک تصمیم نگرفته اند. روشی به نام **تبادل کلید دیفی – هلمن**[۱] وجود دارد که به ایدا و چارلز اجازه می‌دهد مشترکاً یک عدد مخفی را انتخاب کنند. فقط لازم است که آن‌ها پیامی به یکدیگر بفرستند که حتی *لازم نیست مخفی باشد*. برای شخص ثالث غیرممکن است با استفاده از این پیام‌های عمومی بداند ایدا و چارلز کدام عدد مخفی را انتخاب کرده‌اند. بیایید ببینیم این روش چگونه کار می‌کند:

۱. ایدا یک عدد اول a و یک عدد دلخواه دیگر مانند g را انتخاب می‌کند. سپس یک عدد مخفی مانند a را انتخاب می‌کند که آن را به اشتراک نمی‌گذارد.

۲. ایدا پیامی به چارلز می‌فرستد که شامل p، g و عدد سوم A است که با استفاده از حساب *هم‌نهشتی* محاسبه می‌شود[۲]:

$$A = g^a \ mod\ p$$

۳. چارلز یک عدد مخفی c را انتخاب می‌کند و پیامی حاوی عدد C به ایدا می‌رستد که به صورت زیر محاسبه می‌شود:

$$C = g^c \ mod\ p$$

۴. ایدا $C^a \ mod\ p$ و چارلز $A^c \ mod\ p$ را محاسبه می‌کنند. بر اساس چند ویژگی شگفت‌انگیز ریاضی، هر دو عدد یکسانی را محاسبه کرده‌اند! به عبارت دیگر، رمز مشترک آن‌ها این است:

$$C^a \ mod\ p = \ A^c \ mod\ p$$

به طوری که ایدا تنها کسی است که a و چارلز تنها کسی است که c را می‌داند.

برای ایمن بودن این طرح، اعداد باید بزرگ باشند: a، c، g و p باید طوری انتخاب شوند که طول هر عدد چند صد رقم باشد. همچنین اعداد مخفی a و c باید به صورت تصادفی انتخاب شوند. اگر این شرایط برآورده شوند، هیچ روش قابل قبولی برای مهاجمان وجود ندارد که بتوانند کلید مخفی مشترک را بدون اطلاع از a یا c کشف کنند.

با این حال، هنوز یک خطر امنیتی وجود دارد که توسط تبادل کلید دیفی- هلمن پوشش داده نمی‌شود: ایدا و چارلز چگونه می‌توانند مطمئن باشند که واقعاً با یکدیگر تعامل دارند؟ شاید آدا C را نه از

---

[۱] Diffie-Hellman Key Exchange

[۲] باید بدانید $a \ mod\ b$ برابر با باقیمانده‌ی $a \div b$ است. به عنوان مثال $27 \ mod\ 10 = 7$ زیرا وقتی 27 را بر 10 تقسیم می‌کنیم، باقیمانده برابر با 7 است.

چارلز، بلکه از مهاجمی که خود را به جای چارلز جا زده است، دریافت کند. خوشبختانه، رمزنگاران ابزارهایی را توسعه دادند که می‌توانند این مشکل را برطرف کند.

رمزهای کلید عمومی

رمزهایی که تاکنون دیده‌ایم از یک کلید برای رمزگذاری و رمزگشایی استفاده می‌کنند. رمزنگاران با الهام از دیفی و هلمن، نوع دیگری از رمز را توسعه دادند که در آن کلید دوم از کلید اصلی مشتق شده است. هرکس کلید دوم را بداند می‌تواند عملیات رمزگذاری را انجام دهد. با این حال، رمزگشایی به کلید اصلی نیاز دارد. رمزهایی که از کلیدهای مختلف برای رمزگذاری و رمزگشایی استفاده می‌کنند، **رمزهای نامتقارن**[۱] هستند.

در رمزهای متقارن تنها یک کلید وجود دارد که همه‌ی نهادهای درگیر در مکالمه‌ی مخفی آن را می‌دانند. کلید فقط برای آن دسته از افراد مفید است و همیشه باید یک *رمز مشترک و مخفی* باشد. در رمزهای نامتقارن، دو کلید وجود دارد. کلید اصلی **کلید خصوصی**[۲] است و تنها برای یک شخص (یا نهاد) شناخته شده است. کلید دوم **کلید عمومی**[۳] است و در حالت مطلوب برای همه در دسترس است. یک زوج کلید عمومی وخصوصی می‌تواند زمینه‌های ارتباطی مختلفی را فراهم کند.

فرض کنید چارلز می خواهد با ایدا ارتباط برقرار کند. اگر چارلز کلید عمومی ایدا را بداند، می‌تواند از آن برای رمزگذاری پیام متن ساده‌ی خود استفاده کند. سپس متن رمز را به ایدا ارسال می‌کند و ایدا می‌تواند از کلید خصوصی خود برای رمزگشایی آن استفاده کند. اگر اندرو نیز بخواهد با ایدا ارتباط برقرار کند، می‌تواند همین روند را دنبال کند و به طور موثر از یک جفت کلید در یک زمینه‌ی متفاوت استفاده کند! با این حال، اگر ایدا بخواهد به هر دوی آن‌ها پاسخ دهد، باید از کلید عمومی چارلز برای رمزگذاری پیام مخصوص چارلز و کلید عمومی اندرو برای رمزگذاری پیام مخصوص اندرو استفاده کند.

هنوز مشکلاتی در هنگام استفاده از رمزهای کلید عمومی وجود دارد. برای مثال، وقتی ایدا متن رمزی را از چارلز دریافت می‌کند، چگونه می‌تواند مطمئن باشد که پیام واقعاً از طرف چارلز آمده است و نه یک شیاد؟ کلید عمومی ایدا *عمومی* است، بنابراین هر کسی که ادعا می کند چارلز است، می‌تواند متن رمزی را برای او ارسال کند. ثانیاً، اگر چارلز قادر به ملاقات فیزیکی با ایدا نباشد، چگونه می‌تواند یک نسخه از کلید او را بازیابی کند و از معتبر بودن کلید اطمینان حاصل کند؟

---

۱ Asymmetric Ciphers
۲ Private Key
۳ Public Key

امضاهای دیجیتال

علاوه بر رمزگذاری و رمزگشایی، دو عملیات دیگر نیز مبتنی بر رمزنگاری نامتقارن ایجاد شده‌اند: **امضا**[۱] و **تأیید**[۲]. عملیات امضا یک متن ساده و یک کلید خصوصی را به عنوان ورودی گرفته و یک عدد بزرگ به نام **امضای دیجیتال**[۳] را به عنوان خروجی برمی‌گرداند. عملیات تأیید یک کلید عمومی، یک متن ساده و یک امضای دیجیتال را به عنوان ورودی می‌گیرد و بسته به اعتبار امضای دیجیتال، `True` یا `False` را به عنوان خروجی برمی‌گرداند.

تنها راه برای تولید یک امضای دیجیتال قانونی برای متن ساده و کلید عمومی داده شده، انجام عملیات امضا با کلید خصوصی منطبق است. به یاد داشته باشید، در رمزنگاری نامتقارن، کلیدهای خصوصی با یک فرد مرتبط هستند. استفاده از کلید خصوصی برای تولید یک امضای دیجیتال برای یک متن ساده، معادل دیجیتال نوشتن امضای شما در کنار متن ساده است. تنها تفاوت این است که یک امضای سنتی همیشه یکسان است و یک امضای دیجیتال برای هر متن جدید کاملاً متفاوت به نظر می رسد، حتی اگر یک فقط یک حرف تغییر کند.

یک امضای دیجیتال را محاسبه و آن را همراه با پیام منتشر کنید، به این ترتیب هر کسی که کلید عمومی شما را بداند، می‌تواند عملیات تأیید را انجام دهد و تأیید کند که شما آن امضا را برای آن پیام ایجاد کرده‌اید یا خیر. در حالی که کلاهبرداران به طور معمول امضاهای فیزیکی را جعل می‌کنند، هیچ کس نتوانسته است بفهمد که چگونه یک امضای دیجیتال ساخته شده با رمزنگاری قوی را جعل کند.

امضای دیجیتال ما را قادر می‌سازد تا به طور ایمن با هر کسی ارتباط برقرار کنیم، حتی اگر مهاجمان در پیوندهای ارتباطی دخالت کنند. تبادل کلید دیفی - هلمن را به یاد بیاورید: تنها مشکل این بود که شرکت‌کنندگان نمی‌توانستند مطمئن شوند شماره‌هایی که به یکدیگر ارسال می‌کنند در حین انتقال توسط مهاجمان تغییر نکرده است. هنگامی که پیام‌ها در مبادله‌ی کلید با امضای دیجیتال همراه می‌شوند، هیچ راهی وجود ندارد که مهاجم بتواند این فرآیند را تضعیف کند. اگر مهاجمی در پیام‌ها مداخله کند، امضاهای دیجیتال با هم مطابقت نداشته و افراد می‌توانند قبل از تبادل اطلاعات بدانند که فریب خورده‌اند.

رمزهای RSA و ECDSA: در سال ۲۰۲۰، طرح‌های رمزنگاری نامتقارنی که بیشتر مورد استفاده قرار می‌گیرند RSA و ECDSA هستند[۴]. رمز RSA پرکاربردتر است: زوج کلیدهای آن را می‌توان هم

---

[۱] `Signing`
[۲] `Verifying`
[۳] `Digital Signature`
[۴] عبارت RSA مخفف `Rivest-Shamir-Adleman` (نام سه مبدع این طرح رمزنگاری) و عبارت ECDSA مخفف `Elliptic Curve Digital Signature Algorithm` (الگوریتم امضای دیجیتال خم بیضوی) است.

برای رمزگذاری/رمزگشایی و هم برای امضا/تأیید استفاده کرد. الگوریتم ECDSA از نظر محاسباتی ارزان‌تر است، اما فقط می‌تواند برای امضا/تأیید استفاده شود. مکانیزم‌های کاری هر دو طرح بر پایه‌ی ریاضیات پیشرفته بنا شده است. اگر دستورالعمل‌های خاص را برای اجرای هر یک از این طرح‌ها دنبال کنید، ایمن خواهید بود. به عنوان مثال، هنگام امضا با ECDSA، همیشه باید یک نانس ارائه دهید. اگر از یک نانس دوباره استفاده کنید، امنیت شما تضعیف خواهد شد!

گواهی‌های دیجیتال

تصور کنید چارلز می‌خواهد یک پیام مخفی برای ایدا ارسال کند، اما کلید عمومی او را نمی‌داند. اگر ایدا کلید خود را از طریق یک پیوند ناامن برای چارلز بفرستد، مهاجمان می‌توانند پیام را رهگیری کرده و آن را تغییر دهند تا چارلز در عوض کلید آن‌ها را دریافت کند. اگر او به دام بیفتد، ارتباطات آن‌ها برای مهاجمان قابل رمزگشایی خواهد بود.

فرض کنید چارلز کلید عمومی دوست مورد اعتمادش لوییز را می‌داند و لوییز یک نسخه از کلید عمومی ایدا را دارد. لوییز می‌تواند با انتشار پیامی شامل کلید ایدا و به دنبال آن این جمله به چارلز کمک کند: «من، لوییز، گواهی می‌دهم که این کلید عمومی متعلق به ایدا است». سپس به صورت دیجیتالی این پیام را امضا می‌کند و پیام و امضا را عمومی می‌کند. اکنون، هرکسی که به لوییز اعتماد داشته و یک نسخه از کلید عمومی او داشته باشد، می‌تواند پیام را بازیابی و امضا را تأیید کند. اگر امضا معتبر باشد، آن‌گاه می‌تواند به معتبر بودن نسخه‌ی کلید ایدا در پیام او، اعتماد کند؛ حتی در شرایطی که هرگز او را ملاقات نکرده باشد!

پیامی که بیان می‌کند یک کلید عمومی مشخص به یک فرد یا سازمان معین تعلق دارد، یک **گواهی دیجیتال**[۱] نامیده می‌شود. یک نهاد کاملاً قابل اعتماد که گواهی‌های دیجیتال را ایجاد و امضا می‌کند، مرجع صدور گواهی[۲] یا CA نامیده می‌شود. شرکت Verisign یکی از بهترین مراجع معتبر صدور گواهی است. در واقع، اکثر کامپیوترها با کلید عمومی Verisign که در سیستم عامل ثبت شده است، از جعبه خارج می‌شوند. اگر از Verisign بخواهید یک گواهی دیجیتال برای کلید عمومی شما ارائه دهد، می‌توانید هر دو (گواهی و کلید) را از طریق یک پیوند ناامن تقریباً برای هر کسی ارسال کنید و آن‌ها می توانند صحت آن را تأیید کنند.

رمزهای نامتقارن هزاران بار بیشتر از رمزهای متقارن به محاسبات نیاز دارند. به همین دلیل، افراد از این رمزها بیشتر برای اعتبارسنجی کلید عمومی یکدیگر با استفاده از گواهی‌های دیجیتال و ایجاد کلیدهای

---

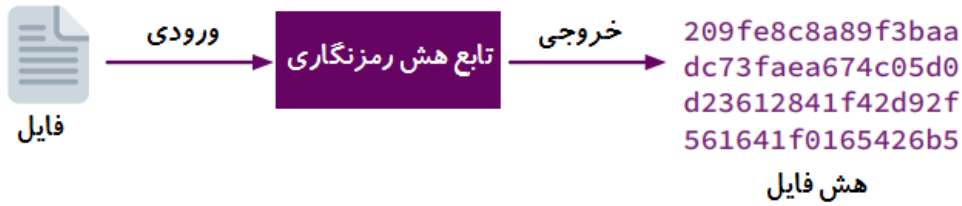
[۱] Digital Certificate

[۲] Certificate Authority یا CA

رمزنگاری متقارن با استفاده از پیام‌های امضا شده‌ی دیفی – هلمن بهره می‌گیرند. در ادامه، بیایید ببینیم چگونه می‌توانیم نیازهای محاسباتی فرایندهای امضا و تأیید پیام‌های طولانی را کاهش دهیم.

۴-۳- درهم‌ریزی یا هش کردن

در فصل اول، در مورد جمع‌آزماها صحبت کردیم: توابعی که با دریافت هر ورودی، یک عدد خروجی با طول ثابت تولید می‌کنند. اگر یک بیت از ورودی تغییر کند، جمع‌آزما خروجی متفاوتی را محاسبه می‌کند. با کمک رمزنگاری، می‌توان توابع جمع‌آزمای ویژه را به‌گونه‌ای ایجاد کرد که یافتن ورودی متناظر با خروجی خاص تولید شده توسط تابع، غیرممکن باشد.. این روش‌های خاص **توابع درهم‌ریزی رمزنگاری** یا **توابع هش رمزنگاری**[۱] نامیده می‌شوند و خروجی چنین تابعی معمولاً **درهم‌ریزی** یا **هش**[۲] نامیده می‌شود.



شکل ۳-۹: هش کردن یک فایل نمونه. این تابع هش یک عدد دودویی با ۲۵۶ بیت به عنوان خروجی تولید می‌کند که به صورت هگزادسیمال نمایش داده شده است. یافتن ورودی متفاوتی که خروجی مشابهی تولید کند غیرممکن است.

توابع هش از نظر محاسباتی ارزان هستند و این ویژگی می‌تواند اجرای امضای دیجیتال را تسهیل کند. به جای انجام محاسبات پرهزینه‌ی امضای یک فایل بزرگ، می‌توانیم به سرعت آن را از طریق یک تابع هش اجرا و خروجی را امضا کنیم. این امضا امن است زیرا مهاجمان بالقوه نمی‌توانند فایل دیگری با همان هش ایجاد کنند. امروزه تقریباً هر امضا با استفاده از فرایند هش کردن داده‌های امضا شده محاسبه می‌شود. چندین روش دیگر برای استفاده از توابع هش وجود دارد. بیایید چند مورد را بررسی کنیم.

---

۱ Cryptographic Hash Functions
۲ Hash

شناسایی تغییرات مخرب

جمع‌آزماهای ساده برای تشخیص اینکه آیا تغییرات تصادفی در یک بخش از داده‌ها رخ داده است یا خیر، استفاده می‌شوند. با این حال، آن‌ها به کشف تغییرات مخرب ایجاد شده توسط مهاجمان باتجربه کمکی نمی‌کنند: با کمی تلاش، مهاجمان می‌توانند داده‌هایی را جعل کنند که دارای جمع‌آزمای مشابه با داده‌های اصلی باشد.

> **امنیت رصدخانه:** ابرکامپیوتر آزمایشگاه شما داده‌های مشاهدات آسمانی را ذخیره می‌کند. شما مشکوک هستید که شخصی در اواخر شب داده‌های شما را تغییر می‌دهد تا تحقیقات شما را خراب کند. فایل‌ها بزرگ هستند: نمی‌توانید از هر فایل یک کپی خصوصی تهیه کنید. چگونه می‌توانید اطمینان حاصل کنید که داده‌های شما به عمد و برای تخریب، تغییر نمی‌کنند؟

راه حل ساده است: یک تابع هش را انتخاب و هش هر فایل را محاسبه کنید. همه‌ی هش‌ها را در یک دستگاه ذخیره‌سازی قابل حمل که به خانه می برید، ذخیره کنید. اگر شک کردید که فایل‌ها تغییر کرده‌اند، دوباره هش آن‌ها را محاسبه کنید. اگر هش جدید برابر با موارد موجود در دستگاه ذخیره‌سازی قابل حمل شما هستند، می‌توانید مطمئن باشید که حتی یک بیت هم تغییر نکرده است؛ مگر اینکه مهاجم موفق به شکستن امنیت تابع هش شده باشد. در ادامه خواهیم دید که چه چیزی یک تابع هش را ایمن می‌کند.

کد احراز هویت پیام

کد **احراز هویت پیام**[۱] یا MAC راهی است برای اثبات اینکه یک پیام معین توسط شخصی که از یک کلید خاص اطلاع دارد ایجاد شده است. اگر چه MAC را می‌توان با امضای دیجیتال پیاده‌سازی کرد، پیاده سازی آن با استفاده از توابع هش از نظر محاسباتی سریعتر است.

> **نظرسنجی امن:** شما در حال انجام یک نظرسنجی از میلیون‌ها پاسخ‌دهنده هستید. هم‌چنین قصد دارید یک کد نظرسنجی منحصربه‌فرد برای هر پاسخ‌دهنده

---

[۱] Message Authentication Code یا **MAC**: این عبارت را با MAC در شبکه‌ها یعنی Medium Access Control (به معنی کنترل دسترسی به رسانه) اشتباه نگیرید (بخش ۱-۱ را ببینید).

ارسال کنید و چیزی شبیه به این بگویید: «این شخص واجد شرایط پاسخگویی به نظرسنجی با کد ■ است» است، که ■ مطابق با نام یا شناسه‌ی هر پاسخ‌دهنده پر می‌شود. چگونه می‌توانید فقط به افرادی که کد نظرسنجی قانونی دریافت کرده‌اند اجازه دهید به نظرسنجی شما پاسخ دهند؟

اگر فقط پیام ساده‌ی ذکر شده در بالا را ارسال کنید، مهاجمان می‌توانند پیام را تغییر داده و هر داده‌ای را که می‌خواهند وارد کنند! ساده‌ترین راه برای مهار این حملات این است که یک عدد تصادفی بزرگ به هر کد نظرسنجی اضافه کنید و تمام کدهایی را که ارسال می‌کنید، ثبت کنید. هنگامی که یک غریبه کد نظرسنجی را با عدد تصادفی بزرگ ارائه می‌دهد، شما در پایگاه داده خود پرس‌وجو می‌کنید تا مشخص کنید آیا واقعاً آن کد نظرسنجی را صادر کرده‌اید یا خیر. این راه حل دست‌وپا گیر است زیرا باید میلیون‌ها کد را پیگیری کنید.

با استفاده از رمزگذاری نامتقارن، می‌توانید از امضای دیجیتال کمک گرفته و برای هر پاسخ‌دهنده یک کد نظرسنجی امضا شده ارسال کنید. به این ترتیب، نیازی به نگهداری هیچ‌گونه رکوردی ندارید. وقتی کدی همراه با امضای آن ارائه می‌شود، اعتبار امضای خود را بررسی می‌کنید تا مشخص شود آیا آن کد نظرسنجی را خودتان ایجاد کرده‌اید یا خیر.

راه دیگری نیز برای رسیدن به همان نتیجه وجود دارد، اما با استفاده از یک تابع هش. ابتدا یک کلید مخفی تصادفی تولید می‌کنید. آن را با دقت ذخیره کرده و با کسی به اشتراک نمی‌گذارید. برای هر پاسخ‌دهنده، از کلید مخفی برای محاسبه موارد زیر استفاده می‌کنید[1]:

```
msg ← "Eligible to answer as " + respondent_name
mac ← hash(secret_key + msg)
```

کد نظرسنجی را می‌توان با الحاق `msg` و `mac` ایجاد کرد. وقتی کسی کدی را ارائه می‌دهد، بخش پیام را استخراج می‌کنید و از کلید مخفی خود برای محاسبه‌ی مجدد `MAC` استفاده می‌کنید. اگر مقدار `MAC` محاسبه‌شده دقیقاً مشابه همان مقداری باشد که در کد نظرسنجی ارائه شده، کد معتبر است[2].

مقادیر `MAC` برای افرادی که از طریق رمزهای رشته‌ای ارتباط برقرار می‌کنند مفید هستند. به یاد داشته باشید: پیام‌های رمزگذاری‌شده با هر رمز رشته‌ای در برابر حملات جعل آسیب‌پذیر هستند. اگر یک

---

[1] در اینجا، علامت + عملیات الحاق را نشان می‌دهد: `"code" + "energy" = "codeenergy"`.

[2] بسته به تابع هش، `MAC` ممکن است به صورت `hash(key + hash(key + msg))` به جای `hash(key + msg)` محاسبه شود. از یک کتابخانه‌ی رمزنگاری برای تولید `MAC` و اطمینان از محاسبه‌ی صحیح آن استفاده کنید.

مقدار MAC با استفاده از کلید رمزگذاری به عنوان رمز مخفی، در انتهای پیام اضافه شود، گیرنده می‌تواند MAC را دوباره محاسبه کرده و تأیید کند که پیام دستکاری نشده است.

مدیریت کلمه‌ی عبور

اکثر سیستم‌های کامپیوتری با درخواست از کاربران برای وارد کردن کلمه‌ی عبور، آن‌ها را احراز هویت می‌کنند. این بدان معنی است که آن‌ها باید رکوردهای تمام کلمه‌های عبور را ذخیره کنند. اما نگه‌داشتن نسخه‌های متنی ساده از کلمه‌های عبور عمل وحشتناکی است: اگر سیستم هک شود، کلمه‌ی عبور هر کاربر به طور بالقوه برای هکرها شناخته می‌شود. علاوه بر این، بسیاری از افراد عادت بد استفاده‌ی مجدد از رمزهای عبور خود را دارند، بنابراین چنین نقص امنیتی در یک سیستم می‌تواند منجر به نقص در سیستم‌های دیگر شود.

ذخیره کردن کلمه‌های عبور به صورت رمزگذاری‌شده نیز مشکل را حل نمی‌کند. با دسترسی کامل به یک سیستم، مهاجمان به طور بالقوه می‌توانند کلید را پیدا کنند و بفهمند که سیستم چگونه به طور خودکار کلمه‌های عبور را در هنگام تأیید درخواست کاربر برای ورود به سیستم، رمزگذاری یا رمزگشایی می‌کند. در این مرحله، کلمه‌ی عبور رمزگذاری‌شده هر فرد به طور بالقوه می‌تواند رمزگشایی شود!

به همین دلیل، بهتر است هش هر کلمه‌ی عبور را به جای یک کپی رمزگذاری شده ذخیره کنید: بر خلاف رمزگذاری، توابع هش برگشت‌پذیر نیستند. با این حال، همچنان می‌توانید کلمه‌ی عبور ارائه شده در هنگام ورود را با هش کردن و مقایسه‌ی آن با مقدار هش ذخیره‌شده، بررسی کنید. اگر هش‌ها یکسان باشند، کلمه‌ی عبور صحیح است[۱]. با این حال، حتی اگر یک سیستم کامپیوتری فقط هش کلمه‌ی عبور را ذخیره کند، هنوز راهی برای مهاجمان وجود دارد که کلمه‌ی عبور افراد را بدانند.

حمله‌ی دیکشنری: مهاجمان می‌توانند هش تریلیون‌ها کلمه‌ی عبور دلخواه را محاسبه و نوعی دیکشنری ایجاد کنند که هش‌ها را به کلمه‌های عبور متن ساده نگاشت می‌کند. سپس هش‌های کلمه‌های عبور دزدیده شده را می‌توان در این دیکشنری جستجو کرد. **حمله‌ی دیکشنری**[۲] یک نوع حمله همه‌جانبه است، اما کار باید فقط یک بار انجام شود. این روش کارآمد است زیرا اکثر مردم از کلمه‌های عبور ساده

---

[۱] به همین دلیل است که سیستم‌های کامپیوتری دارای امنیت مناسب هرگز کلمه‌ی عبور شما را در طول فرآیند بازیابی، ایمیل نمی‌کنند: آنها حتی نمی‌دانند که کلمه‌ی عبور شما چیست!

[۲] Dictionary Attack

استفاده می‌کنند. در سال ۲۰۱۲، یک هکر به لینکدین حمله کرد و هش کلمه‌ی عبور بیش از ۶ میلیون نفر را فاش کرد؛ به طوری که حمله‌ی دیکشنری می‌تواند بیشتر آن‌ها را آشکار کند. ذخیره‌سازی هش کلمه‌ی عبور می‌تواند تقریباً به اندازه‌ی ذخیره‌سازی نسخه‌های ساده‌ی کلمه‌ی عبور بد باشد.

سالتینگ: خوشبختانه راهی برای جلوگیری از حملات دیکشنری وجود دارد. به جای ذخیره کردن نتیجه‌ی `hash(password)` در پایگاه‌داده، سیستم می‌تواند یک عدد تصادفی به نام **سالت**[1] تولید و آن را به صورت متن ساده به همراه نتیجه‌ی `hash(password+salt)` ذخیره کند. از آن‌جایی که هر کاربر سالت متفاوتی دارد، این امر مهاجمان را مجبور می‌کند تا هش هر کلمه‌ی عبور را به صورت جداگانه محاسبه کنند. هر بار که آن‌ها سعی می‌کنند کلمه‌ی عبور یک کاربر خاص را پیدا کنند، حتی نمی‌دانند که آن کاربر کلمه‌ی عبور ساده‌ای دارد که پیدا کردن آن چند دقیقه طول می‌کشد یا رمز پیچیده‌ای که ممکن است قرن‌ها طول بکشد.

هش کردن در چند دور: راهی برای سخت‌تر کردن کار مهاجمانی وجود دارد که سعی می‌کنند هش‌های کلمه‌های عبور را با حملات همه‌جانبه به دست آورند. به جای محاسبه و ذخیره‌سازی هش کلمه‌ی عبور، *هش آن هش* را محاسبه و ذخیره می‌کنیم. به عبارت دیگر، ما نتیجه‌ی `hash(hash(password+salt))` را ذخیره می‌کنیم. به این ترتیب، یک مهاجم حدود دو برابر زمان نیاز دارد تا کلمه‌ی عبور صحیح را حدس بزند. ما می‌توانیم هر تعداد دور هش را به دلخواه خود اضافه کنیم. از آنجایی که محاسبه‌ی هش سریع است، برخی از سیستم‌ها هزاران دور هش را انجام می‌دهند. به یاد داشته باشید: اگر هش کلمه‌ی عبور را ذخیره می‌کنید، مطمئن شوید که از سالت استفاده می‌کنید و چندین دور هش را انجام می‌دهید.

اثبات وجود

از توابع هش می‌توان برای اثبات وجود برخی اطلاعات استفاده کرد، بدون اینکه هیچ بخشی از خود اطلاعات آشکار شود.

> **ستاره‌ی پنهان:** شما آهنگ شگفت‌انگیزی ساخته‌اید که فکر می‌کنید «هتل کالیفرنیا»ی بعدی خواهد بود و می‌خواهید موسیقی را مخفی نگه دارید و آن را فقط در آلبوم بعدی خود منتشر کنید، اما می‌ترسید که کسی آهنگ را بدزدد. به

---

[1] Salt

همین دلیل می‌خواهید آهنگ خود را ثبت کنید، اما ترجیح می‌دهید آهنگ را حتی برای دفتر اسناد رسمی نیز فاش نکنید. چگونه می‌توانید ثابت کنید که آهنگی ساخته‌اید بدون اینکه آن را فاش کنید؟

اولین قدم این است که آهنگ را در یک فایل کامپیوتری ضبط کنید. سپس هش فایل را محاسبه کرده و از یک دفتر اسناد رسمی بخواهید آن شماره‌ی هش را ثبت کند. اگر مالکیت آهنگ مورد بحث است، می‌توانید شماره‌ی هش رسمی خود را به همراه ضبط آهنگ اعلام کنید. تنها راهی که می‌توانستید از یک دفتر اسناد رسمی برای تأیید آن شماره‌ی هش خاص درخواست کنید، داشتن فایل ضبط شده است.

سایت‌های قمار آنلاین نیز از هش استفاده می‌کنند تا به مشتریان خود ثابت کنند که هر بازی منصفانه است. یک بازی ساده را تصور کنید که در آن فرد روی نتیجه‌ی پرتاب سکه شرط‌بندی می‌کند. اگر پرتاب را به درستی حدس بزنید، سهام خود را دو برابر می‌کنید. اگر این کار را نکنید، همه چیز را از دست می‌دهید. ابتدا، کازینوی آنلاین از شما یک عدد می‌خواهد و به طور تصادفی یک سکه پرتاب می‌کند. فرض کنید شیر آمده است. پس کازینو مقدار زیر را محاسبه می‌کند:

```
hash("HEADS" + user_salt + house_salt)
```

مقدار `house_salt` توسط کازینو و `user_salt` توسط شما انتخاب می‌شوند. کازینو مقدارهش را به شما ارائه می‌دهد و سپس منتظر حدس شما می‌ماند. اگر باختید، کازینر می‌تواند با افشای `house_salt` خود ثابت کند که شما را فریب نداده است. شما می‌توانید هش را دوباره محاسبه کرده و بررسی کنید که با هش ارائه شده توسط کازینو قبل از حدس زدن شما، برابر باشد. کازینو نمی‌تواند تقلب کند! برای بررسی این‌که آیا این اصل را فهمیده‌اید، سعی‌کنید به این سوال پاسخ دهید: چرا کازینو نمی‌تواند *قبل* از حدس زدن، شماره‌ی `house_salt` خود را به شما بگوید[1]؟

اثبات کار

اگر به طور مداوم مقادیر هش را برای ورودی‌های مختلف محاسبه کنید، در نهایت یک خروجی خواهید داشت که با هر الگوی خاص مطابقت دارد. به عنوان مثال، اگر به دنبال یک هش دودویی هستید که با چهار صفر شروع می‌شود، می‌توانید به صورت زیر تکرار کنید:

---

[1] اگر هر دو سالت را از قبل بدانید، می‌توانید به راحتی قبل از حدس زدن، هش را با حمله‌ی همه‌جانبه به دست آورید؛ زیرا فقط دو هش وجود دارد که باید محاسبه کنید: یکی برای شیر و دیگری برای خط!

```
hash("heads1")→10111100000101101000011010010000 1…
hash("heads2")→11011001100111011000011110111000 0…
hash("heads3")→01110000011010011111110000101111 0…
hash("heads4")→11000101000110000011010111010000 0…
hash("heads5")→11101110000011110111011111100011…
hash("heads6")→11001001011010101011010011001001 1…
hash("heads7")→11000001001100101101101010010010 1…
hash("heads8")→01001111100001010110100111111000 0…
hash("heads9")→00111011011110101100010101010111 1…
hash("heads10")→11101111000011111110100010110100…
hash("heads11")→10101011110111001100000111111001…
hash("heads12")→11101000110010001110100001001001…
hash("heads13")→00100100001011011101101101000100…
hash("heads14")→01111100010000100101101011000000…
hash("heads15")→00101110000011100011001001100000…
hash("heads16")→01010001000001011111011000000000…
hash("heads17")→11110011000111011100001001101111…
hash("heads18")→00111010001011001001100100110110…
hash("heads19")→00100111100101110100011010000110…
hash("heads20")→10111101011010100010011111110001…
hash("heads21")→10110010100010110001000111000010…
hash("heads22")→01111000100001111111010010001000…
hash("heads23")→10101111001001011100100001110000…
hash("heads24")→01110011110010011101111001101011…
hash("heads25")→00100010101101110000111000110110…
hash("heads26")→00001110001101011100100110011111…
```

پس از ۲۶ محاسبه هش مختلف، با هش heads26 مواجه می‌شویم. این عدد که به صورت دودویی نوشته شده با چهار صفر شروع می‌شود! حالا تصور کنید که شخصی مورد زیر را به شما ارائه کرده است:

```
hash("heads750")→00000000101110000101101110110000…
```

در اینجا، هش با هشت صفر شروع می‌شود! می‌توانید فرض کنید هر کسی که heads750 را به شما ارائه می‌دهد باید حداقل منابع محاسباتی را برای محاسبه‌ی تعداد زیادی هش به کار می‌گرفت تا ورودی مناسب را برای تولید چنین الگویی پیدا کند. به این فرایند **اثبات کار**[1] می‌گوییم. احتمالاً کار بیشتری برای یافتن ورودی زیر صرف شده است، زیرا هش آن با ۱۲ صفر شروع می‌شود:

---

[1] Proof of Work

```
hash("heads11759")→00000000000001110100001001001…
```

از اثبات کار می‌توان برای دشوارتر کردن حملات استفاده کرد. برای مثال، با درخواست از کاربران برای ارائه‌ی اثبات کار قبل از درخواست، سرور می‌تواند هزینه‌ی ارسال درخواست‌های جعلی را برای مهاجمان افزایش دهد. مهاجمان باید منابع محاسباتی زیادی را صرف ایجاد انبوهی از درخواست‌ها کنند که سرور برای پردازش آن‌ها را بپذیرد.

بلاک‌چین: همراه با امضای دیجیتال و هش کردن، اثبات کار یک فناوری را به وجود آورد که قرار است جهان را تغییر دهد: بلاک‌چین[۱]! ما معتقدیم طی چند سال آینده این فناوری می‌تواند با حذف واسطه‌ها و افزودن شفافیت، حریم خصوصی و مسئولیت‌پذیری بیشتر به جامعه داده و برابری اقتصادی را برای میلیاردها نفر فراهم کند. همراه با بلاک‌چین، ابزارهای رمزنگاری کاملاً جدیدی با نام‌های فانتزی در حال توسعه هستند، مانند «اثبات دانش صفر[۲]» و «رمزگذاری هم‌ریختی[۳]». در حال حاضر، ما فقط می‌توانیم رویای قدرتی را داشته باشیم که در آینده توسط این فناوری‌ها به وجود خواهد آمد!

توابع هش ناامن

هنگامی که یک تابع هش به ازای دو ورودی مختلف یک خروجی یکسان تولید کند، می‌گوییم **تصادم** به وجود آمده است. در توابع هش امن، تنها راه برای به دست آوردن یک تصادم روش همه‌جانبه است: محاسبه هش ورودی‌های مختلف تا زمانی که یک تصادم ظاهر شود.

در شرایطی که توابع هش خروجی‌های بیش از ۲۰۰ بیت تولید کنند، یافتن تصادم به روش همه‌جانبه غیرممکن است[۴]. برای هیچ یک از توابع هش که در حال حاضر امن محسوب می‌شوند، هیچ تصادم شناخته‌شده‌ای وجود ندارد. هر چند وقت یکبار، وضعیت یک تابع هش تغییر می‌کند زیرا رمزنگاران روشی را برای یافتن تصادم‌ها کشف می‌کنند که صرفاً به روش همه‌جانبه متکی نیست. به محض اینکه چنین روشی برای یک تابع هش کشف شد، بلافاصله آن را ناامن در نظر می‌گیریم.

---

[۱] Blockchain

[۲] Zero-Knowledge Proof

[۳] Homomorphic Encryption

[۴] اگر یک تابع هش اعداد ۲۰۰ بیتی را تولید کند، 2^{200} مقدار هش متفاوت وجود دارد. اگر تعداد تمام دانه‌های شن روی زمین را در نظر بگیرید و آن را چند تریلیون ضرب کنید، حتی 2^{200} نزدیک نیز نخواهید شد. تصور کنید در چنین فضای جستجویی تلاش می‌کنید یک دانه شن خاص پیدا کنید.

این اتفاق برای توابع هش MD5 و SHA1 رخ داد، در حالی که در دهه‌ی ۱۹۹۰ امن تلقی می‌شدند. یک دهه بعد، روش‌هایی برای کاهش عملیات مورد نیاز برای یافتن تصادم برای هر دو کشف شد. در حال حاضر، تنها تعداد انگشت‌شماری از توابع هش وجود دارند که توسط رمزنگاران امن تلقی می‌شوند. در سال ۲۰۲۱، پرکاربردترین آن‌ها SHA2 است که برای دو دهه بدون شکست باقی مانده است. با این حال، به خاطر داشته باشید که این وضعیت ممکن است در آینده تغییر کند. اگر از یک تابع هش برای چیزی بالقوه حساس استفاده می‌کنید، در جریان باشید تا در صورت کشف روش حمله‌ی جدید، بتوانید زودتر واکنش نشان دهید.

۵-۳- پروتکل‌ها

همانطور که از فصل قبل به خاطر می‌آورید، تمام داده‌هایی که درون بسته‌های IP جابجا می‌شوند به صورت بالقوه اطلاعات عمومی هستند. هکرهایی که به شرکت‌های مخابراتی نفوذ می‌کنند، می‌توانند بسته‌های IP را که بین مبدا و مقصد خود رفت و آمد می‌کنند، تغییر داده تا ارتباطات آن‌ها را تضعیف کنند[۱]. ما می‌توانیم با استفاده از رمزنگاری از خود در برابر این خطرات محافظت کنیم. به طور معمول، ارتباط امن از طریق اینترنت شامل مراحل زیر است:

۱. کلید عمومی معتبر همتای خود را دریافت کنید.
۲. با استفاده از دیفی – هلمن یک کلید مخفی مشترک با همتای خود ایجاد کنید. با استفاده از کلید عمومی مرحله‌ی ۱، مطمئن شوید پیام‌هایی که دریافت می‌کنید معتبر هستند.
۳. تمام ارتباطات بعدی را با یک رمز متقارن با استفاده از کلید مخفی مشترک مرحله‌ی ۲ رمزگذاری کنید.

هنگامی که طرفین کلید عمومی یکدیگر را می‌دانند، امکان برقراری ارتباط امن با استفاده از رمز نامتقارن به صورت مستقیم و بدون استفاده از تبادل کلید دیفی – هلمن وجود دارد. با این حال، انجام این کار یک نقطه‌ضعف ایجاد می‌کند: مهاجمی که یک کلید خصوصی را کشف می‌کند می‌تواند تمام پیام‌های قبلی دریافت‌شده توسط قربانی حتی با افراد دیگر را نیز رمزگشایی کند. با ایجاد یک رمز جدید دیفی – هلمن برای هر نشست ارتباطی، مهاجمی که یک کلید خصوصی را دزدیده است قادر به رمزگشایی متون رمز نشست‌های قبلی نخواهد بود. گفته می‌شود که یک طرح ارتباطی با این ویژگی دارای **محرمانگی پیشرو**[۲] است.

---

[۱] در سال ۲۰۱۳، ادوارد اسنودن فاش کرد که NSA قبلاً چنین حملاتی را حتی بر علیه رهبران کشورهای متحد آمریکا نیز انجام داده است.

[۲] Forward Secrecy

جزئیات بسیاری باید رعایت شوند تا بتوانیم ارتباطات خود را به درستی با محرمانگی پیشرو رمزگذاری کنیم. به عنوان مثال، اگر یک نانس به طور ناخواسته مورد استفاده‌ی مجدد قرار گیرد یا اعداد شبه‌تصادفی بدون دقت تولید شوند، کل ارتباط در برابر حمله آسیب‌پذیر می‌شود. برای جلوگیری از این مشکلات، چندین پروتکل امنیتی به طور گسترده تعریف شده‌اند. این پروتکل‌ها فرآیندهایی را برای افراد به منظور برقراری ارتباط و انجام مراحل رمزنگاری لازم در یک فرآیند سخت‌گیرانه تعریف کرده که خطر اشتباهات را کاهش می‌دهند.

دسترسی امن

در روزهای ابتدایی اینترنت، میزبان‌هایی که یک سرور تلنت را اجرا می‌کردند، معمولاً اتصالات افراد غریبه را می‌پذیرفتند و فقط قبل از ایجاد امکان دسترسی از راه دور به شل، یک کلمه‌ی عبور درخواست می‌کردند. این روال تقریباً هیچ سطحی از امنیت را تضمین نمی‌کرد: کلمه‌ی عبور از طریق اینترنت به صورت متن ساده منتقل می‌شد.

در سال ۱۹۹۵ پروتکلی به نام شل امن[۱] یا SSH برای جایگزینی تلنت و رفع نواقص آن ایجاد شد. هر سروری که اتصالات SSH را می‌پذیرد باید دارای یک زوج کلید عمومی- خصوصی باشد. به منظور تضمین امنیت، کامپیوتر کلاینت باید از قبل کلید عمومی سروری را که می‌خواهد به آن متصل شود، بداند[۲].

به صورت پیش‌فرض، سرورهای SSH انتظار اتصالات را بر روی پورت شماره ۲۲ پروتکل TCP دارند. پس از اینکه یک کلاینت یک اتصال برقرار کرد، سرور کلید عمومی خود را به همراه ابزارهای رمزنگاری که برای رمزگذاری، هش کردن و کد احراز هویت پیام (MAC) استفاده می‌کند، ارسال می‌کند. کلاینت کلید عمومی که به تازگی دریافت کرده است با کلیدی که انتظارش را داشت مقایسه می‌کند. عدم تطابق کلیدهای عمومی یا به این معنی است که سرور مجدداً پیکره‌بندی شده یا اینکه شخصی در حال دستکاری بسته‌های IP در حین جابجایی بین طرفین است. در این شرایط کلاینت باید تلاش برای اتصال را متوقف کرده و بررسی کند که علت این امر چه بوده است.

---

[۱] **Secure Shell** یا **SSH**

[۲] در بسیاری از موارد، شخص یا سازمانی که کامپیوتر کلاینت را اداره می‌کند، صاحب سرور نیز هست، بنابراین این کار به آسانی کپی کردن کلید سرور در یک فضای ذخیره‌سازی قابل حمل است. در موارد دیگر، کاربر کلاینت باید کلید عمومی را از یک منبع قابل اعتماد دریافت کند و ریسک‌های امنیتی آن همانند هر سیستم رمزگذاری کلید عمومی دیگر است.

اگر کلید عمومی درست باشد، کلاینت در دسترس بودن ابزارهای رمزنگاری ارسال شده توسط سرور را بررسی کرده و انتخاب می کند که از کدام یک استفاده کند. سپس کلاینت انتخاب خود را همراه با اولین پیام دیفی – هلمن به سرور ارسال می‌کند. سرور با یک پاسخ دیفی – هلمن امضا شده پاسخ می‌دهد. اگر امضا معتبر باشد، کلاینت و سرور یک کلید مخفی مشترک را محاسبه کرده و با استفاده از رمز متقارن انتخاب شده، ارتباط را آغاز می‌کنند. در این مرحله، کلاینت می‌تواند به صورت امن کلمه‌ی عبور رمزگذاری‌شده را ارسال کند. پس از بسته شدن اتصال، کلاینت و سرور کلید مشترک خود را فراموش می‌کنند تا محرمانگی پیشرو حفظ شود.

انتقال امن

کاربران اولیه‌ی اینترنت به مرورگرهای وب خود برای مقابله با مواردی که به سطوح بالایی از امنیت نیاز داشت، مانند بانکداری آنلاین، اعتماد نداشتند. نگرانی آن‌ها موجه بود: در HTTP، اطلاعات درون بسته‌های IP به صورت متن ساده منتقل می‌شد. بنابراین، مهندسان تصمیم گرفتند طرح‌های رمزگذاری را در HTTP ادغام کنند تا افراد بتوانند با خیال راحت اطلاعات حساسی مانند شماره‌ی کارت اعتباری و پیام‌های خصوصی را از طریق مرورگرهای وب خود ارسال کنند.

آن‌ها یک پروتکل عمومی به نام **امنیت لایه‌ی انتقال**[1] یا TLS ایجاد کردند که می‌تواند هر *اتصال* TCP را امن کند[2]. مکانیزم کار آن شبیه SSH است. ابتدا، کلاینت فهرستی از ابزارهای رمزنگاری که پشتیبانی می‌کند و یک عدد تصادفی ارائه می‌کند. در پاسخ، سرور کلید عمومی خود، ابزار انتخاب‌شده برای استفاده و یک عدد تصادفی دیگر را ارسال می‌کند. کلاینت بررسی می‌کند که آیا کلید عمومی معتبر است یا خیر. کلاینت و سرور یک کلید مخفی مشترک را محاسبه کرده و با استفاده از رمز انتخاب شده توسط سرور، شروع به ارسال متون رمز برای یکدیگر می‌کنند.

برخلاف SSH، انتظار نمی‌رود مشتری از قبل کلید عمومی سرور را بداند. سرور کلید عمومی خود را به همراه یک گواهی دیجیتال ارسال می‌کند. به عنوان مثال، اگر سرور در `example.com` میزبانی می‌شود، باید یک کلید عمومی، به علاوه یک گواهی دیجیتال از یک مرجع صدور گواهی (CA) که تأیید می‌کند `example.com` از آن کلید عمومی استفاده می‌کند، ارسال کند.

---

[1] **T**ransport **L**ayer **S**ecurity یا **TLS**

[2] این پروتکل در ابتدا در سال ۱۹۹۵، لایه‌ی سوکت‌های امن یا SSL (**S**ecure **S**ockets **L**ayer) نام داشت. با ارتقای پروتکل، نام آن به امنیت لایه‌ی انتقال (TLS) تغییر یافت، اما بسیاری هنوز آن را با نام قدیمی‌تر یا SSL/TLS یاد می‌کنند.

پس از این مراحل، پیام‌های HTTP رمزگذاری شده را می‌توان بین کلاینت و سرور رد و بدل کرد. هنگامی که TLS با HTTP استفاده می‌شود، پروتکل حاصل را **HTTPS** می‌نامیم. در حالی که HTTP به طور پیش‌فرض از پورت ۸۰ استفاده می‌کند، HTTPS به طور پیش‌فرض پورت ۴۴۳ را مورد استفاده قرار می‌دهد. سرورهای وب که پورت۴۴۳ پروتکل TCP شنود می‌کنند، انتظار دارند کلاینت اولین پیام TLS را بلافاصله پس از برقراری اتصال ارسال کند.

پروتکل TLS علاوه بر HTTPS در چندین پروتکل دیگر از جمله SMTP استفاده می‌شود. کتابخانه‌های برنامه‌نویسی وجود دارند که استفاده از TCP با TLS را برای کدنویس‌ها آسان می‌کنند. به جای ایجاد یک سوکت TCP معمولی و امن کردن دستی آن، کدنویس از توابع کتابخانه برای ایجاد مستقیم یک سوکت TLS استفاده می‌کند. در پشت صحنه، کتابخانه تمام کارهای رمزگذاری را انجام می‌دهد. برنامه‌نویسان می‌توانند با سوکت‌های TLS تقریباً مانند سوکت‌های TCP کار کنند.

سایر پروتکل‌ها

پروتکل‌های امنیتی بسیار بیشتری وجود دارند. به عنوان مثال، IPsec یک بسط پروتکل است که امنیت ارتباطات مبتنی بر IP را فراهم می‌کند. برخلاف بسته‌های IP، بسته‌های IPsec را نمی‌توان در حین جابجایی خواند یا تغییر داد. پروتکل IPsec اغلب برای ایجاد **شبکه‌های خصوصی مجازی**[۱] یا VPN استفاده می‌شود، که در آن یک اتصال اینترنتی به عنوان یک پیوند فیزیکی عمل می‌کند تا یک کامپیوتر بتواند به یک شبکه‌ی محلی راه دور بپیوندد به طوری که گویی در آنجا قرار دارد.

سیستم DNS نیز دارای بسط‌های امن خود به نام DNSSEC است که رکوردهای DNS را با امضای دیجیتال گسترش می‌دهد. این فرایند به هر کسی امکان می‌دهد تأیید کند که یک رکورد DNS امضا شده، توسط مالک آن ایجاد شده است. بدون DNSSEC، یک سرور DNS می‌تواند رکوردهای DNS جعلی را برای شما ارسال تا شما را به سمت یک میزبان مخرب هدایت کند. در سال ۲۰۲۰، DNSSEC به طور گسترده مورد استفاده قرار نمی‌گیرد، اما میزان پذیرش آن به طور پیوسته در حال افزایش است.

پروتکل‌هایی نیز برای ایمن‌سازی ارتباطات بی‌سیم به طور گسترده مورد استفاده قرار می‌گیرند. به عنوان مثال، این پروتکل‌ها از خواندن محتویات بسته‌های IP شما که از طریق وای‌فای امن منتقل می‌شوند، توسط افراد دارای شنود رادیویی جلوگیری می‌کنند. آن‌ها همچنین به شما این امکان را می‌دهند تا بدون خطر ضبط فرایند استفاده از صفحه‌کلید توسط دستگاه غیرمجاز، روی صفحه‌کلید بلوتوث بی‌سیم تایپ کنید.

---

[۱] **V**irtual **P**rivate **N**etworks یا **VPN**

امروزه نقص در امنیت ایمیل یک نگرانی بزرگ است. ایمیل‌ها را می‌توان با استفاده از SMTP با TLS به طور ایمن منتقل کرد، اما اکثریت قریب به اتفاق سرورهای ایمیل ایمیل‌ها را به صورت متن ساده ذخیره می‌کنند. ارائه‌دهندگان ایمیل بزرگ مانند Gmail یا Hotmail در صورت تمایل می‌توانند ایمیل‌های کاربران خود را بخوانند. اکثر کارشناسان امنیتی بر این نکته اتفاق نظر دارند که یک پیام فقط باید توسط فرستنده و گیرنده‌ی آن قابل خواندن باشد. گفته می‌شود سیستم‌های ارتباطی که به این اصل پایبند هستند از **رمزگذاری سرتاسری**[۱] استفاده می‌کنند. برخی از برنامه‌های پیام‌رسان مانند سیگنال، تری‌ما و واتس‌اپ[۲] از رمزگذاری سرتاسری استفاده می‌کنند. پروتکل‌های جدید به طور فعال در حال توسعه هستند تا رمزگذاری سرتاسری را به صورت گسترده برای ایمیل در دسترس قرار دهند.

۶-۳- هک کردن

هکر فردی با مهارت‌های کامپیوتری قوی است که قادر به دستیابی به اهداف خاصی از را های غیرمجاز است. در فرهنگ عامه، هکرها اغلب به عنوان جادوگران اینترنتی به تصویر کشیده می‌شوند که می‌توانند با فشردن چند کلید به سیستم‌های کامپیوتری محرمانه‌ی نظامی نفوذ کنند. اگرچه فیلم‌ها همیشه واقع‌گرایانه نیستند، اما این ایده که هکرها یک نوع نیروی نخبه هستند، حقیقت دارد. هکرها اغلب در محاسبات سطح پایین متخصص هستند و هر مرحله‌ای را که کامپیوتر برای اجرای یک عمل ساده انجام می‌دهد با جزئیات درک می‌کنند.

بیشتر اوقات، هکرها متخصصان امنیت فناوری اطلاعات هستند که درک عمیقی از پروتکل‌های شبکه دارند و می‌توانند تک تک بیت‌ها را در بسته‌های شبکه دستکاری کنند. آن‌ها از مهارت‌های خود برای کشف حفره‌هایی استفاده می‌کنند که به آن‌ها اجازه می‌دهد کنترل یک ماشین را به دست گرفته و بدون اجازه کارهایی را بر روی آن انجام دهند[۳].

این بدان معنا نیست که همه‌ی هکرها افراد بدی هستند. **هکرهای کلاه سفید**[۴] سخت تلاش می‌کنند تا مطمئن شوند آسیب‌پذیری‌هایی که کشف می‌کنند آسیبی ایجاد نکنند. آن‌ها بیشتر به دنبال

---

[۱] End-To-End Encryption

[۲] Signal, Threema and WhatsApp

[۳] در این شرایط از عبارت Pwning استفاده می‌شود. این عبارتی عامیانه در فضای مجازی است که معادل کلمه‌ی owning است: شکست دادن یا نابود کردن کامل یک دشمن. چنین عبارتی اغلب به هک موفقیت‌آمیز یک سیستم یا نشت داده‌های حساس اشاره دارد. به عنوان مرجع، وب سایت Have I Been Pwned? بررسی می‌کند که آیا داده‌های شخصی مرتبط با یک آدرس ایمیل معین تا کنون عمومی شده‌اند یا خیر. آن را در http://haveibeenpwned.com ببینید.

[۴] White Hat Hackers

آسیب‌پذیری‌ها می‌گردند تا قبل از اینکه آسیبی وارد شود، آن‌ها را برطرف کنند. در مقابل، **هکرهای کلاه سیاه**[۱] معمولاً از آسیب‌پذیری‌ها برای منافع شخصی بدون توجه به ضررهایی که در این فرآیند ایجاد می‌شود، سوء استفاده می‌کنند.

به طور معمول، یک هکر به دنبال راه‌هایی برای *دور زدن* (به جای شکستن) طرح رمزنگاری است که از یک سیستم کامپیوتری محافظت می‌کند. طراحی یک سیستم کامپیوتری مانند ساختن یک قلعه‌ی قرون وسطایی است. شما باید پیش بینی کنید که چگونه مهاجمان مختلف ممکن است تلاش کنند تا به قلعه‌ی شما نفوذ کنند و اقدامات متقابلی را برای هر روش حمله انجام دهید. دانستن نحوه‌ی عملکرد هکرها به شما در توسعه‌ی این اقدامات متقابل کمک می‌کند. با کمال تعجب، پرکاربردترین روش‌های هک تقریباً هیچ مهارت فنی نیاز ندارند.

مهندسی اجتماعی

حملات هکری بیشتر از نقایص انسان‌ها استفاده می‌کنند تا کامپیوترها. فریب کسی که به سیستم کامپیوتری دسترسی دارد می‌تواند بسیار ساده‌تر از دور زدن سیستم دفاعی آن سیستم باشد. هکرهای مخرب معمولاً این کار را با جعل هویت دیگران از طریق ایمیل، تماس‌های تلفنی و پیام‌های متنی انجام می‌دهند. مهاجمان پیشرفته‌تر گاهی اوقات کل وب‌سایت‌ها را جعل می‌کنند. برخی دیگر حتی به صورت فیزیکی در یک مرکز داده ظاهر می‌شوند و وانمود می‌کنند که یک کارمند هستند! به این روش‌ها، حملات **مهندسی اجتماعی**[۲] می‌گویند.

مهندسان اجتماعی مانند کلاهبرداران بسیار حرفه‌ای عمل می‌کنند: آن‌ها می توانند طرح‌های پیچیده‌ای را برای جلب اعتماد و دسترسی ایجاد کنند. رایج ترین ترفند آن‌ها **فیشینگ**[۳] نام دارد. این ترفند زمانی استفاده می‌شود که مهاجم ایمیلی را جعل می‌کند که به نظر می‌رسد از یک منبع قابل اعتماد است. ایمیل‌های فیشینگ معمولاً به یک وب‌سایت تقلبی پیوند می‌خورند که اطلاعات محرمانه مانند کلمه‌ی عبور یا شماره کارت اعتباری را درخواست می‌کند. موارد پیچیده‌تر حاوی نرم‌افزارهای مخربی هستند که در یک پیوست پنهان شده‌اند.

فیشینگ کمیته‌ی ملی دموکرات : در زمان انتخابات ریاست جمهوری ایالات متحده در سال ۲۰۱۶، هکرهای متخاصم یک ایمیل فیشینگ ارسال کردند که برای یکی از اعضای کمیته‌ی ملی دموکرات ارسال شده بود[۴]. این ایمیل درباره فعالیت مشکوک حساب گوگل هشدار داده و از کاربر خواسته بود

---

[۱] Black Hat Hackers
[۲] Social Engineering Attack
[۳] Phishing
[۴] کمیته ملی دموکرات یا DNC، هیئت حاکمه‌ی حزب دموکرات، یکی از دو حزب سیاسی اصلی ایالات متحده است.

کلمه‌ی عبور خود را تغییر دهد. هم‌چنین این ایمیل شامل پیوندی به یک صفحه‌ی وب جعلی گوگل بود که جزئیات ورود کاربر را درخواست می‌کرد. به محض اینکه کاربر کلمه‌ی عبور را وارد کرد، مهاجمان وارد شده و هزاران ایمیل حساس را دانلود کردند. بسیاری از آن‌ها به صورت عمومی منتشر شده و باعث آسیب سیاسی قابل توجه و استعفای چندین سیاستمدار کلیدی شدند.

شکل ۳-۱۰: دریافت شده از http://smbc-comics.com.

حملاتی از این دست بارها و بارها تکرار شده‌اند. تخمین زده می‌شود که حدود ۹۰ درصد از نشت داده‌ها از یک حمله‌ی فیشینگ موفقیت‌آمیز سرچشمه می‌گیرد. نوع دیگری از این حملات به نام **فیشینگ صوتی** یا **ویشینگ**[۱] نیز وجود دارد: هکر برای به دست آوردن اطلاعات ممتاز یا دسترسی به سیستم‌های کامپیوتر، هویت شخصی را در تلفن جعل می‌کند.

ویشینگ سازمان اطلاعات مرکزی آمریکا: در سال ۲۰۱۵، یک هکر ۱۵ ساله‌ی انگلیسی با Verizon تماس گرفت و وانمود کرد که یک کارمند است. او موفق شد اطلاعات کلیدی را در مورد یک مشتری ویژه‌ی شرکت به دست آورد: مدیر سازمان اطلاعات مرکزی آمریکا[۲]. با استفاده از این اطلاعات، هکر توانست در تماسی که با پشتیبانی فنی AOL داشت، هویت مدیر را جعل کند. او به درستی به تمام سوالات امنیتی پاسخ داد و کلمه‌ی عبور ایمیل مدیر را تغییر داد. در نهایت، هکر جوان به اسناد کلیدی نظامی و اطلاعاتی در مورد عملیات سازمان اطلاعات مرکزی آمریکا در عراق و افغانستان دسترسی پیدا کرد.

مهندسی اجتماعی را می‌توان در درجه‌ی اول با آموزش کاربران سیستم در مورد اهمیت بررسی صحت ایمیل‌ها و صفحات وب، قبل از افشای هرگونه اطلاعات خصوصی کاهش داد. همچنین هنگام تغییر کلمه‌ی عبور یا به‌روزرسانی برخی تنظیمات امنیتی دیگر، اجرای فرایند تأیید هویت دقیق برای هر کاربر در سیستم بسیار مهم است. اما این اقدامات احتیاطی کافی نیستند: در حملات پیچیده‌تر، قربانی فقط باید روی پیوند وب کلیک یا یک سند پیوست را باز کند تا هکر کاملاً به سیستم دسترسی یابد.

آسیب‌پذیری‌های نرم‌افزاری

برنامه‌نویسان می‌دانند که بخش‌هایی از کد آن‌ها همیشه دقیقاً آن‌گونه که باید کار نمی‌کند. هر چقدر که نرم‌افزار پیچیده‌تر می‌شود، وضعیت‌های مختلفی که باید آن‌ها را مدیریت کند به طور تصاعدی افزایش می‌یابند؛ و به همین ترتیب خطر وقوع یک وضعیت غیرمنتظره که در آن ترکیبی از ورودی‌ها منجر به رفتار ناخواسته می‌شود نیز افزایش می‌یابد.

این رفتارهای ناخواسته ممکن است باعث از کار افتادن سیستم شوند. چنین رفتارهایی ممکن است باعث افشای اطلاعات محرمانه شوند. در بدترین حالت، ممکن است به یک نفوذگر اجازه داده شود هر

---

[۱] Vishing

[۲] سازمان اطلاعات مرکزی یا CIA، نهادی در ایالات متحده است که به کمک جاسوسان، اطلاعات خارجی را برای کمک به رئیس جمهور در امور امنیت ملی جمع آوری می‌کند.

کدی را اجرا کند. توالی ورودی‌هایی که منجر به چنین رفتارهای ناخواسته‌ای می‌شوند را **آسیب‌پذیری**[۱] می‌نامیم. بیایید اکنون برخی از انواع آسیب‌پذیری‌های رایج را ببینیم.

کنترل دسترسی ناقص: این پدیده زمانی اتفاق می‌افتد که سیستم یک عمل بالقوه خطرناک را بدون بررسی اینکه آیا کاربر مجوز انجام آن را دارد یا خیر، انجام می‌دهد؛ برای مثال زمانی که توسعه‌دهندگان فراموش می‌کنند بررسی‌های مجوز را به کد خود اضافه کنند یا زمانی که یک نرم‌افزار به اشتباه پیکربندی شده است چنین وضعیتی ممکن است رخ دهد. در سال ۲۰۱۶، یک شرکت فناوری که با داده‌های رأی‌دهندگان کار می‌کرد، پایگاه‌داده‌ی خود را بدون رمز عبور به‌درستی پیکره‌بندی‌شده به صورت آنلاین منتشر کرد. در نتیجه، داده‌های خصوصی ۱۵۴ میلیون رای‌دهنده‌ی ایالات متحده افشا شد. دفترچه‌های راهنمای تمام نرم‌افزارهایی که برای استفاده انتخاب می‌کنید را بخوانید و آن‌ها را به درستی پیکره‌بندی کرده تا دسترسی را محدود کنید.

تزریق SQL: این آسیب‌پذیری، رایج‌ترین نوع آسیب‌پذیری است و هکر را قادر می‌سازد تا هر کد SQL را در پایگاه داده اجرا کند[۲]، که اغلب به مهاجمان اجازه می‌دهد تا خودسرانه داده‌ها را بخوانند، بنویسند و حذف کنند. تصور کنید که یک مدیر مدرسه هستید و یک دانش‌آموز جدید به مدرسه‌ی شما آمده است. دفتردار مدرسه نام دانش‌آموز جدید را در کامپیوتر وارد می‌کند. نرم افزار مدیریت مدرسه از نام تایپ شده توسط دفتردار برای ارسال یک پرس‌وجوی SQL مانند زیر به پایگاه‌داده‌ی خود استفاده می‌کند:

```
INSERT INTO Students (name)
VALUES ('■');
```

در این مثال، مقدار ■ باید به‌وسیله‌ی نرم‌افزار با نامی که دفتردار در سیستم تایپ کرده است جایگزین شود. اگر دفتردار بخواهد نام زیر را ثبت کند چه اتفاقی می‌افتد؟

```
Robert'); DROP TABLE Students;--
```

اگر نرم‌افزار مدرسه در برابر تزریق‌های SQL آسیب‌پذیر باشد، کورکورانه هر ورودی را در پرس‌وجو کپی می‌کند. در نهایت این پرس‌وجویی است که اکنون توسط سیستم اجرا می‌شود:

```
INSERT INTO Students (name)
VALUES ('Robert'); DROP TABLE Students;--');
```

---

[۱] Vulnerability

[۲] زبان SQL یک زبان پایگاه‌داده برای استفاده، درج و اصلاح داده‌ها است. ما آن را در اولین کتاب خود، مبانی و مفاهیم علوم کامپیوتر، معرفی کرده‌ایم.

هنگام درج یک رکورد دانش‌آموز جدید با این نام مخرب، یک اثر جانبی رخ می‌دهد: کل جدول از پایگاه داده حذف می‌شود. برای دفاع در برابر این نوع حمله، تمام ورودی‌های خود را بررسی کنید و هر کاراکتری را که می‌تواند اثرات جانبی ایجاد کند، مانند علامت نقل قول، جایگزین کنید[۱].

حملات تزریق SQL بسیار رایج هستند. در سال ۲۰۱۲، هکرها به چندین وب‌سایت دولتی ایالات متحده از جمله ناسا، اف‌بی‌آی و پنتاگون حمله کردند. در نتیجه، اطلاعات شخصی بیش از یک میلیون کارمند و پیمانکار در اینترنت در دسترس عموم قرار گرفت.

حملات تزریق یکی از انواع آسیب‌پذیری‌های ناشی از مدیریت بد ورودی‌ها هستند. شما باید نه تنها محتوا، بلکه *اندازه‌ی* ورودی‌های بیرونی را که به سیستم خود می‌دهید نیز بررسی کنید.

سرریز بافر: قبل از اینکه برنامه‌ها بتوانند هر ورودی را پردازش کنند، ورودی باید در فضای حافظه داخلی به نام **بافر**[۲] کپی شود. اگر داده‌های ورودی بزرگتر از بافر باشند و هیچ‌گونه بررسی انجام نشود، داده‌ها همچنان در حافظه کپی می‌شوند و از انتهای بافر عبور می‌کنند. این بدان معناست که بخش‌هایی از ورودی در مکان‌های ناخواسته‌ی حافظه ختم می‌شود. این پدیده، **سرریز بافر**[۳] نامیده می‌شود و ممکن است باعث از کار افتادن سیستم یا اتفاقاتی بدتر از آن شود: اگر مهاجم موفق شود داده‌ها را در یک مکان خاص بنویسد، ممکن است کامپیوتر میزبان فریب بخورد و قسمت‌هایی از ورودی را به عنوان کد اجرا کند.

هنگامی که به هکرها این فرصت داده می‌شود تا کدهای مخرب را در یک ماشین از طریق سرریز بافر اجرا کنند، اغلب کنترل کاملی بر آن به دست خواهند آورد. یافتن این نوع آسیب‌پذیری برای هکرها دشوار است، اما در سیستم‌عامل‌های اصلی و نرم‌افزارها کاربردی رایج است. به عنوان مثال، در سال ۲۰۱۵، یک مورد در Adobe Reader یافت شد، یک برنامه‌ی معروف برای باز کردن فایل‌های PDF. قبل از اینکه Adobe یک به‌روزرسانی نرم‌افزاری برای رفع مشکل منتشر کند، هکرها توانستند یک فایل PDF حاوی کد مخرب ایجاد کرده که باعث سرریز بافر می‌شد. اگر کاربران سند را روی کامپیوتر خود باز می‌کردند، کد مخرب به صورت مخفیانه اجرا می‌شد!

---

[۱] علاوه بر تزریق SQL، انواع دیگری از تزریق وجود دارند که در آن‌ها یک ورودی به سیستم داده می‌شود که باعث عواقب ناخواسته می‌گردد. به عنوان مثال، هنگام کپی کردن ورودیهای کاربر در یک صفحه‌ی HTML، مطمئن شوید که کاراکترهای < و > جایگزین شده‌اند تا از تزریق برچسب‌های مخرب جلوگیری شود.

[۲] Buffer

[۳] Buffer Overflow

روز صفر: بسیاری از شرکت‌ها برای هکرهایی که به طور مسئولانه آسیب‌پذیری‌های امنیتی را گزارش می‌کنند، جوایز بزرگی ارائه می‌دهند. با این حال، بسیاری از هکرهای کلاه سیاه و آژانس‌های جاسوسی دولتی ترجیح می‌دهند آسیب‌پذیری‌هایی را که پیدا می‌کنند فاش نکنند، زیرا ممکن است بتوانند اطلاعات خود را به سایر هکرها فروخته یا از آن‌ها در راه اهدافی با سود زیاد استفاده کنند. آسیب‌پذیری‌هایی که به طور عمومی شناخته‌شده نیستند و توسط گروهی از هکرها استفاده می‌شوند، آسیب‌پذیری‌های روز صفر[۱] نامیده می‌شوند. تخمین زده می‌شود که هر نرم افزار یا سیستم عاملی که به طور گسترده مورد استفاده قرار می‌گیرد، چندین آسیب‌پذیری از نوع روز صفر دارد. این بدان معنی است که تقریباً هر کامپیوتری می‌تواند توسط سازمان‌های دولتی و گروه‌های هک نخبه هک شود[۲].

اکسپلویت

آسیب‌پذیری‌ها اغلب در نرم‌افزارهای رایج کشف می‌شوند. پس از افشای یک آسیب‌پذیری جدید، هکرها سریعاً یک توالی حمله‌ی عمومی را برای سیستم‌های حساس کدنویسی می‌کنند. به چنین کدی **اکسپلویت**[۳] می‌گویند. بسته به نوع آسیب‌پذیری، یک اکسپلویت می‌تواند یک ماشین آسیب‌پذیر را آفلاین کرده، اطلاعات خصوصی را در معرض دید قرار داده، یا بدتر از آن، کامپیوتر را وادار به اجرای کد مخرب کند. اگر یک اکسپلویت عمومی شود، هر کسی می‌تواند با کمترین تلاش از آن برای هک کردن سیستم‌های آسیب‌پذیر استفاده کند[۴].

هر روز بر تعداد آسیب‌پذیری‌های شناخته شده افزوده می‌شود و ابزارهای پیچیده‌ای به وجود می‌آیند که به هکرها کمک می‌کند تا از آنها سوء استفاده کنند. به عنوان مثال، `Metasploit` برنامه‌ای است که به طور خودکار یک سیستم را از نظر آسیب‌پذیری‌های شناخته شده غربال و اکسپلویت‌های مربوطه را اجرا می‌کند. برای انجام این کار، `Metasploit` از یک پایگاه‌داده‌ی بزرگ و مرتباً به روز شده از آسیب‌پذیری‌ها و اکسپلویت‌های مربوط به آن‌ها استفاده می‌کند. این ابزار همچنین می‌تواندکامپیوترها را از طریق شبکه اسکن و هر کدام را در رابطه با آسیب‌پذیری‌های احتمالی بررسی کند.

همانطور که هکرها می‌توانند از این ابزارها برای یافتن نقاط ضعف سیستم شما استفاده کنند، شما نیز می‌توانید! سازمان‌هایی که این ابزارهای هک را توسعه داده و نگهداری می‌کنند به *امنیت تهاجمی* اعتقاد

---

[۱] `Zero-Day`

[۲] به دلیل این تهدید، سرویس گارد فدرال روسیه در سال ۲۰۱۲ استفاده از کامپیوترها را برای ارتباطات خاص متوقف و به جای آن از ماشین‌های تحریر مکانیکی استفاده کرد.

[۳] `Exploit`

[۴] فرد آماتوری که اطلاعات کمی در مورد امنیت دارد و فقط به دانلود و اجرای اکسپلویت‌های ایجاد شده توسط هکرهای واقعی می‌پردازد، `Script Kiddie` یا `skid` نامیده می‌شود.

دارند: هک کردن آسیب‌پذیری‌های شناخته شده را آسان می‌کند تا افراد نیز بتوانند به راحتی بدانند که کدام بخش از سیستم‌هایشان ناامن است و نیاز به توجه دارد.

روت‌کیت و کی‌لاگر: کد مخربی که هکرها سعی می‌کنند روی کامپیوتر قربانی خود اجرا کنند، **پی‌لود**[۱] نامیده می‌شود. دو نوع اصلی پی‌لود وجود دارد. **روت‌کیت**[۲] به هکر اجازه می‌دهد به طور مخفیانه به یک شل در کامپیوتر میزبان دسترسی پیدا کند. **کی‌لاگر**[۳] همه چیزهایی را که روی صفحه‌کلید تایپ می‌شود ضبط می‌کنند تا بعداً توسط هکر بازیابی شوند. اگر هکر بتواند قربانیان خود را وادار کند که این پی‌لودهای مخرب را اجرا کنند، این ابزارها معمولاً پنهان می‌مانند، در پس‌زمینه بدون شناسایی اجرا و برای مدت زمان زیادی باعث آسیب می‌شوند.

نرم‌افزار آنتی‌ویروس: یک آنتی‌ویروس[۴] تمام داده‌های ذخیره شده در کامپیوتر و کدهایی را که در آن اجرا می‌شوند، بررسی می‌کند. این نرم‌فزار سعی می‌کند پی‌لودها را شناسایی و از اجرا یا انتشار بیشتر آن‌ها جلوگیری کند. ارائه‌دهندگان نرم‌افزار آنتی‌ویروس دائماً اینترنت را زیر نظر دارند و هر روت‌کیت یا کی‌لاگر را که پیدا کنند، فهرست می‌کنند. با این حال، هکرها برای فرار از شناسایی، بی‌وقفه پی‌لودها را جهش می‌دهند. در هر زمان معین، چندین نوع از پی‌لودها وجود دارند که هنوز توسط هیچ ارائه‌دهنده‌ی آنتی‌ویروسی کشف نشده‌اند. نرم افزار آنتی‌ویروس کمک می‌کند، اما در برابر حملات پیچیده دفاع نمی‌کند.

با توجه به سرعت انتشار اکسپلویت‌ها، انتظار می‌رود توسعه‌دهندگان به روزرسانی‌هایی را منتشر کنند که آسیب‌پذیری‌های نرم‌افزار را در سریع‌ترین زمان ممکن برطرف کنند. هنگامی که یک به روزرسانی وجود دارد که آسیب‌پذیری را برطرف می‌کند، گفته می شود که **پچ**[۵] شده است. افرادی که از نرم‌افزار آسیب‌پذیر استفاده می‌کنند، باید به‌روزرسانی‌های امنیتی را در سریع‌ترین زمان ممکن اعمال تا از آسیب‌پذیری جلوگیری کنند.

توسعه‌دهندگان باید همیشه در مورد جزئیات کتابخانه‌هایی که استفاده می‌کنند مطلع باشند تا مطمئن شوند نرم‌افزار خود را بر روی کدهای شخص ثالثی که دارای آسیب‌پذیری‌های اصلاح‌نشده است، نمی‌سازند. به همین ترتیب، مدیران سیستم باید از جدیدترین پچ‌های نرم‌افزاری که در حال استفاده از آن

---

۱ Payload
۲ Rootkit
۳ Keylogger
۴ Antivirus
۵ Patch

هستند، به ویژه برای زیرساخت‌های حیاتی مانند سرورهای وب، سرورهای ایمیل و سرورهای پایگاه‌داده مطلع باشند.

منبعی وجود دارد که می‌تواند به ما کمک کند تا از جزئیات این اطلاعات مطلع باشیم: **فهرست عمومی آسیب‌پذیری‌ها و مواجهه‌ها**[۱] یا CVE. به محض اینکه یک آسیب‌پذیری شناخته شد، به لیست CVE اضافه می‌شود و یک شماره به آن اختصاص می‌یابد. اگر توضیحات معمولی به‌روزرسانی نرم‌افزار را بررسی کنید، اغلب پچ‌های آسیب‌پذیری‌های امنیتی را می‌بینید که به شماره CVE آن‌ها ارجاع داده شده است. برای آگاهی از هر گونه آسیب‌پذیری شناخته شده در نرم افزاری که در حال حاضر استفاده می‌کنید، می‌توانید به فهرست CVE مراجعه کنید[۲].

بات‌نت: برخی از هکرها فرآیند اسکن اینترنت را برای یافتن کامپیوترهای آسیب‌پذیر و اجرای اکسپلویت‌ها به منظور تزریق روت‌کیت‌های خود خودکار کرده‌اند. پس از سال‌ها اجرای اکسپلویت‌های خودکار، برخی هکرها در نهایت ارتشی از کامپیوترهای راه دور به نام بات‌نت[۳] را کنترل می‌کنند. کارشناسان تخمین می‌زنند که امروزه بات‌نت‌هایی با صدها هزار کامپیوتر وجود دارند. هکرها اغلب این بات‌نت‌ها را به دیگران اجاره می‌دهند تا از آن‌ها به عنوان سکوی پرتاب برای حملات بیشتر استفاده کنند. اگر در مورد امنیت کامپیوتر خود محتاط نبوده‌اید، ممکن است به صورت مخفیانه بخشی از بات‌نت کسی باشد!

دیوارهای آتش: نوع دیگری از نرم‌افزار وجود دارد که می‌تواند به محافظت از کامپیوترها در برابر اکسپلویت‌های مضر کمک کند: **دیوار آتش** یا **فایروال**[۴]. این نرم‌افزار بسته‌های IP را که انتظار نمی‌رود در شبکه منتقل شوند مسدود می‌کند. به عنوان مثال، اگر انتظار نمی‌رود کامپیوتری منتظر اتصالات TCP باشد که از خارج شروع شده‌اند، می‌توان یک فایروال نصب کرد تا تمام بسته‌های IP خارجی را که تلاش می‌کنند اتصال TCP راه اندازی کنند مسدود کند. این امر ارتباط یک هکر با یک پی‌لود مخرب نصب شده از طریق یک اکسپلویت را بسیار دشوارتر می‌کند.

---

[۱] The Common Vulnerabilities and Exposures یا **CVE**

[۲] می توانید فهرست را در http://code.energy/cve جستجو کنید.

[۳] Botnet

[۴] Firewall

درب‌های پشتی: آسیب‌پذیری‌ها در سیستم‌های کامپیوتری همیشه به صورت تصادفی به وجود نمی‌آیند. آسیب ذیری که عمداً توسط یک برنامه‌نویس یا مهندس وارد سیستم شده است، **درب پشتی** یا **بک‌دور**[۱] نامیده می‌شود. گاهی اوقات، سازمان‌های مجری قانون یا ارتش از سازندگان سخت‌افزار یا ارائه‌دهندگان نرم‌افزار درخواست می‌کنند که درب‌های پشتی را در سیستم‌های خود قرار دهند. اگرچه ممکن است هدف درب‌های پشتی، استفاده از آن‌ها توسط افراد خوب باشد، ولی هکرهای کلاه سیاه نیز می‌توانند آن‌ها را کشف کرده و از آن‌ها سوءاستفاده کنند.

یک درب پشتی حتی می‌تواند در یک الگوریتم بنیادین وجود داشته باشد. چندین بار مشخص شده است که NSA برای تعیین استانداردهای رمزنگاری تغییریافته به صورت عمدی که قادر به شکستن توسط آن سازمان باشند، به افراد و سازمان‌ها فشار آورده است. در سال ۲۰۰۴، NSA حتی ۱۰ میلیون دلار به یک ارائه‌دهنده‌ی نرم‌افزار اصلی رشوه داد تا به طور پیش‌فرض از چنین الگوریتم‌های تغییریافته‌ای استفاده کند. وقتی نوبت به انتخاب رمزها و سایر ابزارهای رمزنگاری برای سیستم‌هایتان می‌رسد تحقیق کرده و مواردی را پیدا کنید که توسط کارشناسان دانشگاهی مستقلی که به آنها اعتماد دارید، توصیه می‌شود.

جنگ دیجیتال

سیستم‌های کامپیوتری برای عملکردهای یک کشور مدرن ضروری هستند. این سیستمها ارتباطات بخش دولتی و خصوصی را کنترل می‌کنند. آن‌ها نیروگاه‌ها و شبکه‌ی برق را اداره می‌کنند. آن‌ها ستون فقرات بانک‌ها و بازارها هستند. کنترل‌کننده‌های ترافیک هوایی و سیستم‌های راداری آن‌ها به این سیستم‌ها متکی هستند. شبکه‌های راه‌آهن و قطارهای آن نیز توسط کامپیوتر اداره می‌شوند.

اگر یک مهاجم بتواند همه‌ی این سیستم‌ها را به یکباره هک کند، فاجعه خواهد بود. خطوط ارتباطی ممکن است قطع شوند، حساب‌های بانکی و بازارها محو شوند، نیروگاه‌ها تعطیل شوند، رادارها خاموش شوند، قطارها از ریل خارج شوند و الی آخر. به طور خلاصه، آخرالزمان خواهد شد.

ما قبلاً شاهد حملات سایبری به اهداف نظامی برجسته بوده‌ایم. در سال ۲۰۱۰، هکرهای آمریکایی و اسرائیلی از آسیب‌پذیری‌های روز صفر برای نوشتن ویروسی استفاده کردند که آن را **استاکس‌نت**[۲] نامیدند. این برنامه برای نفوذ به کامپیوترهای داخل تأسیسات هسته‌ای ایران برنامه‌ریزی شده بود. این ویروس به آرامی گسترش یافت و در نهایت کامپیوترهای کنترل‌کننده‌ی سانتریفیوژهای غنی‌سازی اورانیوم را آلوده کرد. به سانتریفیوژها دستور داد که به طور متناوب با سرعت‌های مختلف بچرخند و به

---

[۱] Backdoor
[۲] Stuxnet

طور مکرر آن‌ها را تحت فشار قرار داد تا زمانی که از کار افتادند. این ویروس همچنین برای فرار از تشخیص، روال ابزار قرائت را جعل کرده بود.

سازمان اطلاعات ایران در نهایت ویروس را کشف کرد، اما اگر چند ماه بیشتر طول می‌کشید، ممکن بود آن را از دست می‌دادند: استاکس‌نت طوری برنامه‌ریزی شده بود که در نهایت خودش را حذف کند و ردی از خود باقی نگذارد. استاکس‌نت پیشرفته‌ترین مورد جنگ دیجیتالی است که می‌شناسیم، امافقط این مورد نبوده است. بسیاری از سرنخ‌های بیشتر نشان می‌دهند که عملیات جنگ دیجیتال[۱] در حال حاضر بین تیم‌های هکر کشورهای مختلف در حال انجام است.

نیروهای مسلح کشورهای قدرتمند سرمایه‌گذاری زیادی در **اکسپلویت کردن شبکه کامپیوتری**[۲] یا CNE: آموزش هکرها و توسعه‌ی اکسپلویت‌ها برای عملیات جاسوسی و شناسایی. آن‌ها همچنین در **حمله به شبکه کامپیوتری**[۳] یا CNA سرمایه گذاری می‌کنند: استقرار سلاح های دیجیتالی که می‌توانند سیستم‌های دشمن را خراب کنند، به عنوان مثال از بین بردن داده‌ها یا از کار انداختن ارتباطات.

یک مورد جالب CNA توسط ارتش اسرائیل در سال ۲۰۰۷ انجام شد. جت‌های جنگنده‌ی نیروی هوایی اسرائیل به اعماق حریم هوایی سوریه، *که گویی کاملاً بدون دفاع بود*، پرواز و تأسیسات هسته‌ای مشکوک را منهدم کردند. بعداً مشخص شد که اسرائیلی‌ها سیستم‌های کامپیوتری نیروی پدافند هوایی سوریه را هک کرده‌اند تا خوانش‌های راداری را جعل کنند. این امر به هواپیماهای اسرائیلی اجازه داد تا عملیات خود را با دقت انجام دهند و بدون شلیک حتی یک گلوله از سوی دشمنان محل را ترک کنند.

برای کشورهای پیشرفته از نظر فناوری، وجود گسترده‌ی آسیب‌پذیری‌های امنیتی جنبه مثبتی دارد. جان جاسوسان و سربازان آن‌ها روی زمین کمتر در خطر است. به جای اینکه جاسوسان جان خود را به خطر بیندازند و سال‌ها طول بکشد تا به صورت فیزیکی به مقر دشمن نفوذ کنند، اکنون می‌توانند مستقیماً کامپیوترهای دشمن را هک کرده و اسناد حساس را از راه دور به سرقت ببرند. آن‌ها می‌توانند به جای فرستادن نیرو یا پرتاب موشک‌های بالستیک برای نابودی تاسیسات هسته‌ای، به سرورها نفوذ و نرم‌افزارهای مخربی را برای تخریب این تاسیسات از داخل آن اجرا کنند.

در مقیاس کوچک‌تر، شرکت‌های خصوصی نیز گاهی با نفوذ به رقبای خود و شکار اسناد استراتژیک مانند گزارش‌های مالی یا طرح‌های اختراعات جدید سری، وارد جنگ دیجیتالی با یکدیگر می‌شوند. آن‌ها با الهام از ارتش، با تهدید حملات سایبری از طریق آموزش کارکنان خود به منظور تبدیل شدن به هکر، مقابله می‌کنند. این فرایند به اصطلاح تمرین تیم قرمز در مقابل تیم آبی نامیده می‌شود: تیم

---

۱ Digital Warfare

۲ Computer Network Exploitation یا **CNE**

۳ Computer Network Attack یا **CNA**

قرمز باید یک سیستم کامپیوتری مشخص را هک کند، در حالی که تیم آبی سعی می‌کند حمله را دفع کند. برخی از شرکت‌ها حتی هکرهای حرفه ای را برای تیم قرمز استخدام می‌کنند.

محاسبات کوانتوم: رمزنگاری و امنیت همیشه در حال پیشرفت هستند. امروزه ارتش و سرویس‌های اطلاعاتی کشورهای قدرتمند سرمایه‌گذاری زیادی در تحقیق و توسعه‌ی کامپیوترهای کوانتومی[۱] انجام می‌دهند. چنین کامپیوترهایی می‌توانند در این قرن به واقعیت تبدیل شوند و اولین کشوری که به آن‌ها دست یابد می‌تواند به طور بالقوه جهان را متحول کند. این کامپیوترها می‌توانند ر عرض میکروثانیه، مشکلاتی را که ابرکامپیوترهای فعلی ما حتی در یک تریلیون سال نمی‌توانند حل کنند، حل کنند.

هنگامی که کامپیوترهای کوانتومی کاملاً کاربردی ساخته شوند، تقریباً همه‌ی الگوریتم‌های رمزنگاری که امروزه استفاده می‌کنیم ناامن خواهند شد. با این حال، ذهن‌های درخشان در حال حاضر بی‌وقفه بر روی ایجاد رمزهای آینده کار می‌کنند که در زمان وقوع انقلاب محاسبات کوانتومی ایمن باقی می‌مانند.

چک‌لیست دفاع

هنگام شروع حمله به سیستم شما، هکرهای مخرب معمولاً برخی از مراحل زیر را دنبال می‌کنند:

۱. یک نسخه‌ی جعلی از سرویس شما ساخته و برای کاربران شما ایمیل می‌کنند و از آن‌ها می‌خواهند رمز عبور خود را وارد کنند.
۲. با کاربران شما تماس می‌گیرند و ادعا می‌کنند از تیم پشتیبانی هستند و آن‌ها را وادار می‌کنند که اعتبارنامه‌ها یا اطلاعات حساس را افشا کنند.
۳. دستورات مخفی SQL را در تمام ورودی‌های شما که احتمال دارد جزیی از یک پرس‌وجوی پایگاه‌داده باشند، قرار می‌دهند.
۴. سعی می‌کنند با روش‌هایی مانند تعریف کاربر جدید بدون داشتن مجوزهای لازم برای خود دسترسی ایجاد کنند.
۵. با ارسال قطعات بسیار بزرگ داده به هر ورودی ممکن که توسط سیستم شما مدیریت می‌شود، سعی می‌کنند سرریز بافر را به وجود بیاورند.
۶. اسکریپتی برای تعامل با نرم‌افزار احراز هویت شما ساخته و سعی می‌کنند کلمه‌های عبور را با روش همه‌جانبه بشکنند.

---

[۱] Quantom Computers

۷. شبکه‌ی شما را برای پیدا کدن آسیب‌پذیری‌های شناخته شده اسکن می‌کنند.

آماده سازی سیستم برای مقابله با تمام این تهدیدات اولین قدم برای بهبود امنیت سیستم شما است.

نتیجه‌گیری

رمزنگاری امنیت را در دنیای دیجیتال امکان پذیر می‌کند. این ابزار به ما امکان می‌دهد با داده‌های حساس حتی در زیرساخت‌های ناامن کار کنیم. امنیت دیجیتال به سه چیز بستگی دارد: ابزارهای رمزنگاری قوی، اصلاح آسیب‌پذیری نرم‌افزار، و آگاهی کاربر از تهدیدات مهندسی اجتماعی.

با مهمترین ابزارهای رمزنگاری آشنا شدیم. رمزهای متقارن برای انتقال و ذخیره‌ی مخفیانه‌ی داده‌ها استفاده می‌شود. رمزهای نامتقارن برای امضای دیجیتال، گواهی دیجیتال، و ایجاد کلیدهای مخفی مشترک بر روی اتصالات ناامن استفاده می‌شود. هش کردن به ما کمک می‌کند جامعیت داده‌ها را تضمین و تأیید و رمزهای عبور مخفی را مدیریت کنیم.

تاریخ اهمیت همراه ماندن با پیشرفت‌های رمزنگاری را به ما می‌آموزد، زیرا اکثریت قریب به اتفاق ابزارهای رمزنگاری ممکن است روزی شکسته شوند. همچنین به ما می‌آموزد که هکرها تقریباً همیشه با یافتن راه‌هایی برای دور زدن دفاع رمزنگاری، به جای شکستن آن‌ها، به سیستم‌ها حمله می‌کنند. این کار را می‌توان با بهره‌برداری از آسیب‌پذیری‌های نرم افزاری یا به سادگی با وادار کردن انسان‌ها به باز کردن دروازه‌ها انجام داد. این تکنیک‌ها به بخشی ضروری از جنگ مدرن تبدیل شده‌اند، جایی که سیستم‌های کامپیوتری حیاتی می‌توانند بی‌رحمانه توسط هکرهای تحت حمایت دولت دشمن مورد حمله قرار گیرند.

رمزنگاری امنیت را در دنیای دیجیتال امکان پذیر می کند. این به ما امکان می دهد با داده های حساس حتی در زیرساخت های ناامن کار کنیم. امنیت دیجیتال به سه چیز بستگی دارد: ابزارهای رمزنگاری قوی، اصلاح آسیب‌پذیری نرم‌افزار، و آگاهی کاربر از تهدیدات مهندسی اجتماعی.

با مهمترین ابزارهای رمزنگاری آشنا شدیم. رمزهای متقارن برای انتقال و ذخیره مخفیانه داده ها استفاده می شود. رمزهای نامتقارن برای امضای دیجیتال، گواهی دیجیتال، و ایجاد کلیدهای مخفی مشترک بر روی اتصالات ناامن استفاده می شود. هش کردن به ما کمک می کند یکپارچگی داده ها را تضمین و تأیید کنیم و رمزهای عبور مخفی را مدیریت کنیم.

این فصل مقدمه‌ای بر امنیت سایبری بود، جایی که ما اصول اولیه‌ای را ارائه کردیم که معتقدیم هر کسی که مسئول سیستم‌های بالقوه حساس است، باید بداند. اگر همه‌ی توسعه‌دهندگان با مفاهیم این فصل آشنا بودند، می‌شد از بسیاری از نقص‌های امنیتی مخرب در گذشته اخیر جلوگیری کرد.

دفاع از سیستم‌های کامپیوتری در برابر حملات پیچیده آسان نیست، و گاهی اوقات فقط می‌تواند توسط متخصصان امنیتی بسیار ماهر انجام شود. از این گذشته، یک تیم دفاعی موفق باید با همه‌ی انواع حمله‌ی احتمالی مقابله کند. یک مهاجم فقط باید یک آسیب‌پذیری را کشف کند تا سیستم را با مشکل مواجه کند. به همین دلیل، تیم‌های دفاعی به طور کلی چندین لایه اقدامات امنیتی را اجرا می‌کنند. هنگامی که این کار به خوبی انجام شود، بهره‌برداری از یک آسیب‌پذیری برای نفوذ به یک سیستم کافی نیست.

دیدیم که هکرهای خرابکار می‌توانند تا حد زیادی برای سرقت داده‌ها تلاش کنند. در برخی موارد، نشت داده‌های بزرگ حتی می‌تواند یک شرکت را ورشکست کند یا به یک دولت آسیب جدی برساند. اما چه چیزی باعث می‌شود حجم زیادی از داده‌ها تا این اندازه ارزشمند باشند؟ در فصل بعد، نحوه‌ی درک و پیمایش مقادیر زیادی از داده‌ها را بررسی خواهیم کرد، البته منظور، داده‌های به‌دست‌آمده از طریق قانونی است.

منابع و مراجع

- The Code Book, by Singh
  - http://code.energy/singh
- Serious Cryptography, by Aumasson
  - http://code.energy/aumasson
- Ghost in the Wires, by Mitnick
  - http://code.energy/mitnick
- Hands on Hacking, by Hickey and Arcuri
  - http://code.energy/hickey

فصل ۴

تحلیل

آینده‌ی تحلیل داده‌ها می‌تواند شامل پیشرفت‌های بزرگی باشد. آیا این‌طور خواهد شد؟ این به ما بستگی دارد، به تمایل ما برای در پیش گرفتن جاده‌ی سنگلاخ مشکلات واقعی، به جای جاده‌ی هموار فرضیات غیرواقعی.

– جان توکی[۱]

داده‌ها دانش را تقویت می‌کنند. چه در حال انجام یک نظرسنجی برای اندازه‌گیری رضایت مشتریان یا اجرای برخورددهنده‌ی هادرونی بزرگ[۲] برای پیشرفت فیزیک ذرات باشید، انتظار دارید از داده‌هایی که جمع‌آوری می‌کنید چیزهایی بیاموزید.

داده ها به راحتی ممکن است مورد سوءتفاهم یا سوءتعبیر قرار بگیرند. خوشبختانه، **تحلیل داده‌ها**[۳] در اینجا به ما کمک می‌کند تا دانش قابل اعتمادی تولید کنیم. دانشمندان مختلف رویکردهای متفاوتی برای تحلیل داده‌ها دارند که اغلب به ماهیت تحقیق و میزان دسترسی آن‌ها بستگی دارد. این فصل یک گردش‌کار را برای تحلیل داده‌ها توسط برنامه‌نویسان پیشنهاد می‌کند که به چهار مرحله تقسیم می‌شود: *جمع‌آوری*[۴]، *پردازش*[۵]، *کاوش*[۶] و *آزمایش*[۷].

شما یاد خواهید گرفت که:

---

[۱] John Tukey (1915-2000): ریاضی‌دان آمریکایی مبدع الگوریتم تبدیل فوریه سریع و نمودار جعبه‌ای. او اولین کسی بود که از کلمات بیت و نرم‌افزار استفاده کرد. تست محدوده‌ی توکی، توزع لامبدای توکی و تست افزایشی توکی به نام او نامگذاری شده‌اند. م.

[۲] Large Hadron Collider: بزرگترین برخورددهنده‌ی ذرات با بالاترین انرژی در جهان است که در تونلی به طول ۲۷ کیلومتر و در عمق ۱۷۵ متری زیر مرز سوئیس و فرانسه در نزدیکی ژنو قرار دارد. م.

[۳] Data Analysis

[۴] Collection

[۵] Processing

[۶] Exploration

[۷] Test

داده‌ها را به طور قابل اعتماد و جامع **جمع‌آوری** کنید،
آن‌ها را در یک پایگاه‌داده‌ی تمیز و قدرتمند **پردازش** کنید،
با خلاصه‌سازی مقادیر **کاوش** را شروع کنید،
از طریق مصورسازی به **کاوش** عمیق‌تر بپردازید،
با **آزمایش** شهود خود نتیجه‌گیری کنید.

حتی بهترین ما اغلب مراحل مهم در این فرآیند پیچیده را نادیده می‌گیریم. ما اندازه‌گیری‌های نادرست را ثبت می‌کنیم، اطلاعات مهم را نادیده می‌گیریم یا نتیجه‌گیری نادرست می‌کنیم. علاوه بر این، تحلیل داده‌ها فرایندی تکراری است: در هر مرحله، ممکن است متوجه شویم که مرحله‌ی قبلی می‌تواند بهبود یابد. چنین فرایندی می‌تواند به سرعت به یک فرآیند پیچیده تبدیل شود.

برای جلوگیری از ایجاد آشفتگی، باید روش‌مند باشیم. این کار خسته کننده، اما با ارزش است. تحلیل دقیق داده‌ها به چارلز داروین[۱] کمک کرد تا منشا گونه‌ها را کشف کند و ممکن است اسپیس ایکس[۲] را قادر سازد تا حیات را چند سیاره‌ای کند. آن را در بلندپروازانه‌ترین پروژه‌های خود به کار بگیرید.

هوش

تصور کنید صاحب یک قهوه‌خانه‌ی کوچک و پردردسر هستید. برای توسعه‌ی یک استراتژی سودآور، باید کسب و کار خود را بشناسید. مشتریان شما کدام نوشیدنی را ترجیح می‌دهند؟ چند قهوه در روز فروخته می‌شود؟ یک مشتری به طور متوسط چقدر هزینه می‌کند؟ بیشتر مشتریان در چه زمانی می‌آیند و می‌روند؟ آیا تامین‌کنندگان شما نسبت به دیگران مطلوب هستند؟ آیا این معیارها در چند ماه گذشته بهبود یافته‌اند؟

پاسخ به این سؤالات ارزشمند است، زیرا به شما امکان می‌دهد به طور مؤثر برنامه‌ریزی کنید و شاخص‌های کلیدی عملکرد را برای سنجش و پیگیری پیشرفت تعریف کنید. سازمان‌هایی که مدیران خوبی دارند، برای گردآوری چنین دانش ضروری، داده‌های خود را تحلیل کرده و گزارش‌های دوره‌ای

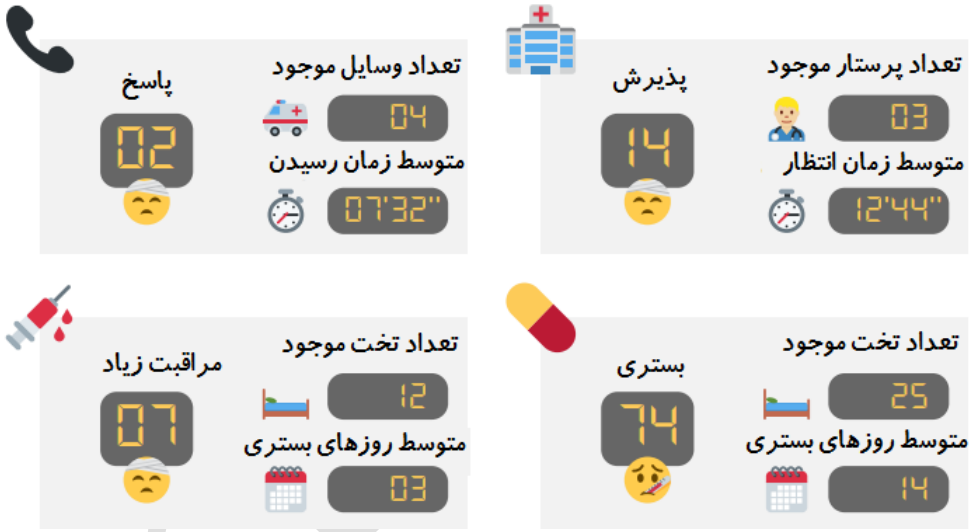
---

[۱] Charles Darwin (1809-1882): زیست‌شناس و زمین‌شناس انگلیسی و مشهور به دلیل نظریه‌ی فرگشت. م.

[۲] SpaceX: شرکت آمریکایی تولیدکننده‌ی محصولات هوافضا و ارائه‌دهنده‌ی خدمات اکتشاف فضایی. م.

تولید می‌کنند. کارمندان و مدیران می‌توانند به راحتی میزان پیشرفت را بررسی کرده و زمینه‌های بهبود را شناسایی کنند.

بسیاری از سازمان‌ها کاری بیش از ارائه‌ی گزارش در مورد داده‌های خود انجام می‌دهند؛ آن‌ها داده‌ها را به صورت بلادرنگ ردیابی می‌کنند، به طوری که می‌توانند پیامدهای اقدامات خود را در داشبوردهای مجازی ببینند. چنین فرآیندهایی از تصمیم‌گیری از طریق یادگیری و توزیع اطلاعات پشتیبانی می‌کنند و آنچه را که مردم **هوش تجاری**[1] می‌نامند شکل می‌دهند.



شکل ۴-۱: داشبورد بخش اورژانس بیمارستان، ردیابی بلادرنگ تعداد بیماران در واحدهای مختلف.

هوش تجاری به سازمان‌های تجاری محدود نمی‌شود. بخش اورژانس یک بیمارستان دولتی را در نظر بگیرید که در آن می‌توان از داشبورد اطلاعاتی «سازمان» برای ردیابی وضعیت بیماران استفاده کرد (شکل ۱-۴). سپس، برای مثال، اگر یک افزایش غیرمعمول در زمان انتظار وجود داشته باشد، به سرعت به کارکنان پذیرش بیمارستان هشدار داده می‌شود تا قبل از تشدید وضعیت به آن رسیدگی کنند.

در نتیجه‌ی این هوشمندی، بیمارستان نیز آمادگی بهتری برای مقابله با بحران‌ها دارد. اگر یک حادثه جدی در نزدیکی آن رخ دهد، بیمارستان قادر خواهد بود بلافاصله کاهش تعداد آمبولانس‌های موجود را مشاهده کند. با دانستن اینکه چه منابعی در دسترس است، کارکنان می‌توانند به سرعت حتی تصمیم بگیرند که آیا قبل از بازگشت آمبولانس‌های اعزام شده، با نیروهای کمکی تماس بگیرند یا خیر.

---

[1] Business Intelligence

هوش تجاری و موارد مشابه دیگر در زمینه‌ی تحلیل داده‌ها از شما می‌خواهد که داده‌ها را جمع‌آوری، خلاصه‌سازی و مصورسازی کرده و از آن‌ها یاد بگیرید. ما تمام این موضوعات را بررسی خواهیم کرد. اگر سازمان شما بر هوش تجاری و مدیریت دانش متمرکز نیست، الگوبرداری کنید. داده‌های مرتبط را بررسی کنید و یافته‌های خود را در یک گزارش قانع‌کننده یا در یک داشبورد ساده به اشتراک بگذارید. به این ترتیب ارزش بسیار زیادی ایجاد خواهید کرد.

۱-۴- جمع‌آوری

دانش مفید تنها از داده‌های مرتبط پدید می‌آید. در هنگام مواجهه با حجم عظیمی از اطلاعات نامرتبط، شناسایی داده‌های مهم می‌تواند دشوار باشد. با تعریف اهداف قبل از شروع، جستجوی خود را اصلاح کنید. چه اطلاعاتی با این اهداف مرتبط هستند؟

اگر هدف شما بهبود منوی قهوه‌خانه باشد، ویژگی‌های محصولات موفق و ناموفق اهمیت دارند. نظرات باریستاها و مشتریان خود را جمع‌آوری کنید. به همین ترتیب، داده‌های مربوط به مواد تشکیل دهنده، قیمت‌گذاری و فروش همگی به هدف شما مربوط می‌شوند. اگر هدف شما بهبود مراقبت از بیمار در بیمارستان باشد چه؟ ببینید چه عواملی برای تشخیص بیماری بیماران و پیگیری بهبودی آن‌ها اهمیت دارند. داده‌ها مرتبط با بیماران، بیماری‌ها و روش‌های درمان را جمع‌آوری کنید.

انواع داده‌ها

سعی کنید در هنگام بررسی و مطالعه، تصویر کاملی به دست بیاورید. اطلاعات را از تمام زوایای ممکن ضبط کنید، به این ترتیب کمتر احتمال دارد که جزئیات مهم را نادیده بگیرید. برای مثال، اگر در حال اندازه‌گیری دما هستید، دانستن زمان و مکان انجام این اندازه‌گیری‌ها و اینکه آیا هوا آفتابی، باد، مه‌گرفته یا بارانی بوده است، می‌تواند مفید باشد. به عنوان برنامه نویس، ما دوست داریم از دسته‌بندی زیر استفاده کنیم تا مطمئن شویم که داده‌های مربوطه را نادیده نگرفته‌ایم:

عددی	شمارش امتیازات، اندازه‌گیری فیزیکی و ...
دسته‌ای	ورزش‌های المپیک، ژانرهای فیلم، نژادهای سگ و ...
زمانی	تاریخ تولد، ساعت و ...
جغرافیایی	مکان، آدرس خانه، مرز و ...

غیرساخت‌یافته	ضبط صدا، بدنه‌ی ایمیل، فیلم وب‌کم، دستور غذا و ...

گفته می‌شود که چهار نوع اول داده‌ها، **ساخت‌یافته** [۱] هستند، زیرا به شیوه‌ای از پیش تعریف‌شده سازماندهی شده‌اند. به عنوان مثال، توان الکتریکی بر حسب وات اندازه‌گیری می‌شود، هر مدال در المپیک ۱۸۹۶ به یکی از ۹ ورزش موجود اعطا شد، تاریخ‌ها را می‌توان در یک تقویم سازماندهی کرد، و مرزهای بین المللی در امتداد مختصات تعریف می‌شوند.

کامپیوترها داده‌های ساخت‌یافته را خوب مدیریت می‌کنند. از سوی دیگر، درک داده‌های **غیرساخت‌یافته** [۲] برای آن‌ها سخت است. روش‌های خاصی برای استخراج داده‌های ساخت‌یافته از داده‌های غیرساخت‌یافته وجود دارد. به عنوان مثال، نرم‌افزار تشخیص چهره می‌تواند یک ویدیوی ورودی غیرساخت‌یافته را دریافت و هویت شخصی را به عنوان داده‌های دسته‌ای به عنوان خروجی تولید کند.

دریافت داده‌ها

بسیاری از فعالیت‌های روزانه‌ی ما توسط کامپیوترها تسهیل شده‌اند، بنابراین رفتار و اعمال ما اغلب ردپای دیجیتالی از خود به جا می‌گذارد. سیستم‌های کامپیوتری را برای ردیابی این داده‌ها به کار بگیرید. حسگرهای خود را نصب کنید تا داده‌های بیشتری جمع‌آوری کنید یا نحوه‌ی عملکرد شرکت خود را تغییر دهید. علاوه بر این، اغلب می‌توانید داده‌های مفیدی را از اشخاص ثالث دریافت کنید.

داده‌های موجود: کامپیوترهای بیمارستانی اغلب علائم حیاتی بیماران مانند ضربان قلب و دمای بدن را کنترل می‌کنند. کامپیوترهای رستوران سفارش داده شده در هر میز را ثبت می‌کنند. کامپیوترهای شما چه داده‌هایی را دستکاری می‌کنند؟ همه‌ی سیستم‌ها را برای پیدا کردن یک کپی از رکوردهای مفید جستجو کنید. به عنوان مثال، ماشین میزبان وب سایت شرکت را کاوش کنید و به احتمال زیاد فهرستی از تمام بازدیدکنندگان وب پیدا خواهید کرد.

داده‌های جدید: همیشه داده‌های جالبی وجود دارند که ثبت نشده‌اند. به عنوان مثال، بیشتر رستوران‌ها میزان رضایت مشتریان را پس از صرف غذا ثبت نمی‌کنند. برای رفع این مشکل، هر صورت‌حساب می‌تواند همراه با یک برگه باشد تا مشتری خدمات را در مقیاسی از یک تا ده رتبه‌بندی کند. تا زمانی که

---

[۱] Structured
[۲] Unstrutured

از مرزهای اخلاق تجاوز نمی‌کنید، همیشه به دنبال راه‌هایی برای گرفتن داده‌های بیشتر باشید. اگر سیستم صورت‌حساب رستوران مدت زمان نشستن مشتریان را ثبت نمی‌کند، آن را تغییر دهید.

شکل ۴-۲: به یاد داشته باشید که هنگام جمع‌آوری داده‌ها، از خود همدلی نشان دهید.

حسگرها: ما اغلب به حسگر[۱]هایی برای مطالعه‌ی پدیده‌های طبیعی فکر می‌کنیم. برای مثال، اقلیم‌شناسان به حسگرهایی برای دما، فشار هوا، رطوبت و غیره نیاز دارند. با این حال، حسگرها می‌توانند برای کسب‌وکارها نیز ارزشمند باشند: رستوران‌ها می‌توانند از صداسنج‌ها برای ردیابی نویز محیطی که مشتریانشان تجربه می‌کنند استفاده کنند، و مراکز خرید اغلب از حسگرهای حضور برای ثبت تعداد بازدیدکنندگان روزانه استفاده می‌کنند. علاوه بر این، برنامه‌های کاربردی وب اغلب از **حسگرهای مجازی**[۲] برای ردیابی رفتار کاربران، مانند زمانی که در صفحات جداگانه صرف می‌کنند، بهره می‌گیرند.

داده‌های بیرونی: شما می‌توانید از داده‌هایی استفاده کنید که توسط دیگران جمع‌آوری شده‌اند. به عنوان مثال، واسطه‌های املاک و مستغلات از داده‌های شخص ثالث برای اطلاع از تاریخچه‌ی قیمت املاک استفاده می‌کنند. دوستداران فیلم می‌توانند داده‌هایی را در مورد اکثر فیلم‌های تجاری بیابند و علاقه‌مندان به ورزش می‌توانند داده‌های مربوط به اکثر مسابقات حرفه‌ای را پیدا کنند. دولت‌ها معمولاً داده‌های سرشماری را ارائه می‌کنند که شاخص‌های مهم اجتماعی- اقتصادی ملی را نشان می‌دهند. از جستجو در مجموعه‌داده‌های گوگل[۳] برای پرس‌وجو از هزاران مجموعه‌داده از شرکتها، دانشگاه‌ها و سازمان‌های دولتی استفاده کنید. ممکن است داده‌های اضافی مرتبط با اهداف خودتان را پیدا کنید.

---

۱ Sensor

۲ Virtual Sensor

۳ مجوعه‌ی Google Dataset را در http://code.energy/google-data ببینید.

اسکرپینگ: اغلب اوقات، داده‌های مربوطه از اینترنت قابل دانلود نیستند؛ فقط در صفحات وب در دسترس هستند. به عنوان مثال، برخی از وب‌سایت‌ها نظرات مشتریان در مورد بارها و رستوران‌ها را جمع‌آوری می‌کنند. اگر به این داده‌ها نیاز دارید، اسکریپتی بنویسید که از آن صفحات بازدید کرده و بخش‌های مربوطه را در کامپیوتر شما کپی کند. به این فرایند **اسکرپینگ وب**[۱] می‌ویند، و کاری نسبتاً رایج است؛ حتی نرم افزار رایگانی وجود دارد که این فرآیند را خودکار می‌کند.

حریم خصوصی: مراقب باشید که اطلاعات هویتی شخصی افراد را بدون رضایت صریح آن‌ها جمع‌آوری نکنید. این کار کاملاً غیراخلاقی است. علاوه بر این، مردم را فریب ندهید تا سیاست‌های نامناسب حریم خصوصی را بپذیرند، همانطور که برخی از شرکت‌ها با ارائه‌ی شرایط و ضوابط طولانی که هیچ‌کس هرگز آن‌ها نمی‌خواند، این کار را انجام می‌دهند.

سوگیری انتخاب

تصور کنید صاحب رستورانی هستید و برگه‌های میزان رضایت مشتریان را جمع‌آوری می‌کنید. شما داده‌هایی بیشتر از آنچه می‌توانید پردازش کنید دریافت می‌کنید، بنابراین تصمیم می‌گیرید فقط از ۱۰ درصد مشتریان نظرسنجی کنید. اگر کارکنان انتخاب کنند که چه کسی مورد نظرسنجی قرار می‌گیرد، ممکن است ترجیح دهند مشتریانی را با روحیه‌ی خوب انتخاب کنند. در نتیجه، داده‌ها می‌توانند نشان دهند که رضایت مشتریان نسبت به وضعیتی که در آن از همه‌ی افراد نظرسنجی کنیم، بیشتر است.

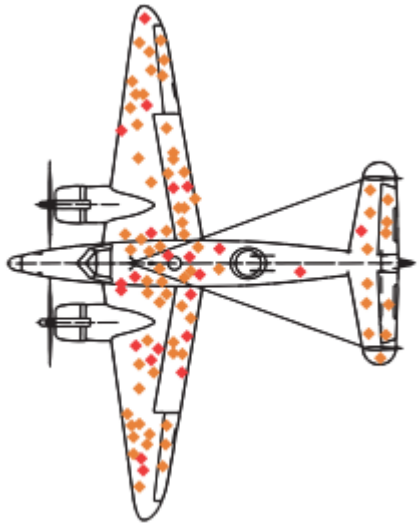
این مشکل **سوگیری انتخاب**[۲] نامیده می‌شود. همچنین اگر فقط به مشتریانی که پشت میزهای خاصی می‌نشینند، برگه بدهید، ممکن است همین مشکل مجدداً رخ دهد. آن‌ها ممکن است به دلیل نشستن در مکانی پر سر وصدا یا مکانی که نور خورشید به صورتشان می‌تابد، احساس متفاوتی داشته باشند این امر می‌تواند بر پاسخ‌ها تأثیر بگذارد و داده‌های شما واقعیت را به صورت دقیق منعکس نکنند. بیایید مثال دیگری را در نظر بگیریم:

هواپیمای زرهی: شما والد هستید، یک ریاضیدان در طی جنگ جهانی دوم. پدافند هوایی دشمن تلفاتی را به نیروی هوایی وارد می‌کند و شما تمام

---

[۱] Web Scraping
[۲] Selection Bias

عکس‌هایی را که از هواپیماهای بازگشتی گرفته شده در اختیار دارید. مهندسان به شما می‌گویند که باید وزن را محدود کنند، بنابراین آن‌ها فقط می‌توانند به یکی از سه مکان در یک هواپیما زره اضافه کنند: بال‌ها، بدنه یا موتورها. کدام را انتخاب می‌کنید؟



در ابتدا، به نظر می‌رسد که افزودن زره به بدنه یا بال‌ها بهترین کار است، زیرا بیشترین شلیک‌ها در این دو مکان رخ داده است. با این حال، داده‌ها سوگیری زیادی دارند: شما فقط عکس‌های هواپیماهای برگشتی را تهیه کرده‌اید. شما عکس‌هایی از هواپیماهایی که گم شده بودند ندارید! به نظر می‌رسد بسیاری از سقوط‌ها ناشی از آسیب موتور باشند. بهترین راه برای نجات خلبانان اضافه کردن زره به آنجا است.

تعصبات فردی ما اغلب به سختی قابل تشخیص است، زیرا بر اساس شهود ما به روشی عمل می‌کنند که انتظار نداریم، و سوگیری انتخاب نیز از این قاعده مستثنی نیست. همیشه با انتخاب تصادفی مطمئن شوید که هیچ چیز روی رکوردهایی که جمع‌آوری می‌شوند تأثیر نمی‌گذارد؛ و توجه داشته باشید که مانند والد، گاهی اوقات نمی‌توانید داده‌های بی‌طرفانه را جمع آوری کنید.

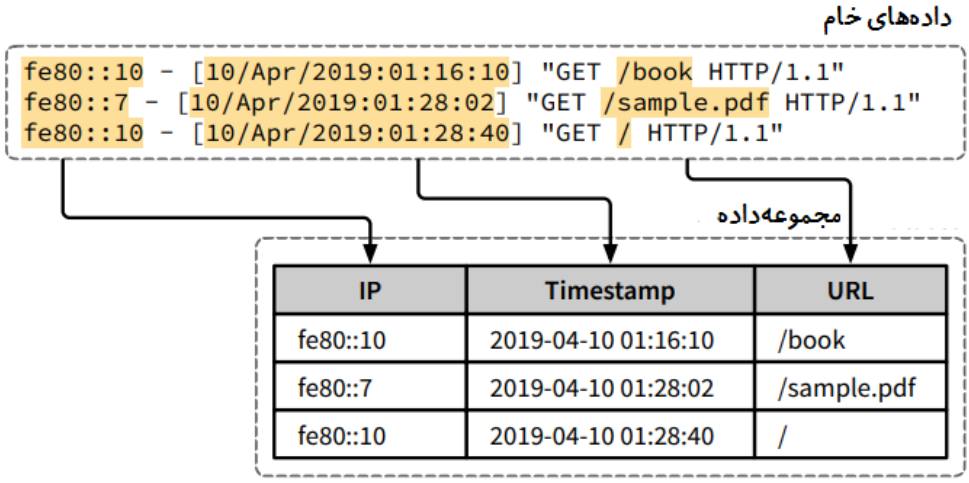
۲-۴- پردازش

جمع آوری داده‌های زیاد مانند پر کردن یک انبار با کالا است. می‌تواند وسوسه‌انگیز باشد که همه چیز را به طور تصادفی در قفسه‌ها رها کنیم و فقط به ذخیره‌سازی چیزهای جدید و جالب متمرکز شویم.

با این حال، هر چه انبار بهتر سازماندهی شود، کار آسان‌تر خواهد بود. داده‌های خود را به محض ورود سازماندهی کنید.

آماده‌سازی داده‌ها

فرآیند جمع‌آوری داده‌ها معمولاً فایل‌های بی‌شماری را ایجاد می‌کند: گزارش‌ها، داده‌های پایگاه‌داده SQL، صفحات گسترده، و غیره. به این موارد **منابع خام**[۱] می‌گویند. بخش‌های مرتبط از اطلاعات درون این فایل‌ها باید به گونه‌ای شناسایی، استخراج و سازماندهی شوند که دسترسی به آن‌ها را آسان کند. به عنوان مثال، استخراج داده‌ها از گزارش دسترسی به سرور وب می‌تواند به شکل زیر باشد:



شکل ۴-۳: استخراج داده‌ها از گزارش دسترسی به سرور وب.

فقط داده‌های مرتبط را از منابع خام انتخاب کنید. آن‌ها را استخراج کرده و با استفاده از قالبی که کامپیوترها به راحتی می‌توانند آن را درک کنند، آن‌ها را به فایل‌های جدید تبدیل کنید. ما این فایل‌های جدید را **مجموعه‌داده**[۲] می‌نامیم. مجموعه‌داده‌های خوب‌ساخته‌شده را می‌توان مستقیماً و بدون سازماندهی بیشتر تحلیل کرد. فرآیند تبدیل منابع خام به یک مجموعه‌داده را **آماده‌سازی داده‌ها**[۳] می‌نامند. بیایید مراحل مربوطه را بررسی کنیم.

---

۱ Raw Sources
۲ Dataset
۳ Data Wrangling

جدول‌سازی: جداول رایج‌ترین روش برای ساختاردهی داده‌ها هستند و ساختار پیش‌فرض برای پردازش داده‌ها به شمار می‌روند: بیشتر الگوریتم‌ها بر روی داده‌های جدولی بهترین عملکرد را دارند. هر جدول رکوردهایی از رویدادها یا اشیاء از یک نوع را ذخیره می‌کند. اولین گام از آماده‌سازی داده‌ها، تقسیم تمام داده‌های مرتبط در قالب جداول است.

در جداول، هر رکورد به یک ردیف تبدیل می‌شود. چیزهای مختلفی که در مورد این رکوردها می‌دانیم به ستون تبدیل می‌شوند. در زمینه‌ی تحلیل داده‌ها، ستون‌ها به عنوان متغیر[۱] نیز شناخته می‌شوند. شکل ۴-۳ یک منبع متنی خام را نشان می‌دهد که در یک جدول با سه متغیر کپی شده است: IP، Timestamp و URL.

معمولاً داده‌های جدولی در قالب CSV[۲] ذخیره می‌شوند. فایل‌های CSV را می‌توان به راحتی تقریباً در هر محیط برنامه‌نویسی وارد کرد. اگر داده‌های جدولی شما در یک پایگاه‌داده‌ی رابطه‌ای[۳] ذخیره می‌شوند، اغلب تلاش برای تبدیل داده‌ها به فایل‌های CSV ارزشمند است. این کار تضمین می‌کند که شما یک جدول برای هر موجودیت داده دارید و تحلیل را بسیار ساده‌تر می کند: برای تفسیر داده‌ها نیازی به حل و فصل روابط در جداول مختلف نیست.

نرمال‌سازی: همیشه بررسی کنید که سلول‌های یک ستون یکنواخت باشند. وقتی چندین راه برای بیان چیزی وجود دارد، باید داده‌ها را **نرمال‌سازی**[۴] کنید. به عنوان مثال، اگر یک ستون دما را نشان می‌دهد، فارنهایت یا سانتیگراد[۵] را انتخاب کرده و مقادیر دیگر را تبدیل کنید. سه نمونه از نرمال‌سازی در شکل‌های ۴-۴، ۴-۵ و ۴-۶ نشان داده شده است.

نرمال‌سازی همچنین می‌تواند شامل تقسیم یک ستون به دو یا چند ستون باشد. تصور کنید که پایگاه داده‌ی نقاشی‌های یک موزه دارای یک ستون «ابعاد» و حاوی متنی است که اندازه‌های بوم را نشان

---

[۱] Variable

[۲] مقادیر جدا شده با کاما (Comma-Separated Values) یا CSV ساده ترین فرمت برای ذخیره کردن داده‌های جدولی است. این داده‌ها به صورت یک متن ساده هستند که در آن کاما و خطوط جدید، سلول‌ها و ردیف‌ها را مشخص می‌کنند.

[۳] سیستم‌های پایگاه‌داده‌ی رابطه‌ای مانند MySQL و PostgreSQL از فرمت‌هایی استفاده می‌کنند که امکان تقسیم و ذخیره‌ی داده‌ها را در چندین جدول فراهم می‌کند. برنامه نویسان این کار را به منظور از بین بردن تکرار اطلاعات انجام می‌دهند. اولین کتاب ما، *مبانی و مفاهیم علوم کامپیوتر*، مزایا و معایب پایگاه‌داده‌های رابطه‌ای را توضیح می‌دهد.

[۴] Normalization

[۵] دانشمندان معمولاً داده‌ها را در قالب سیستم متریک نرمال‌سازی می‌کنند تا اعداد آن‌ها را راحت‌تر دستکاری و با داده‌های جهانی مقایسه کنند.

می‌دهد. منطقی است که این ستون را به ستون‌های ارتفاع و عرض تقسیم کنیم که هر کدام شامل اعداد با فرمت مناسب در یک واحد اندازه‌گیری یکنواخت هستند (شکل ۴–۷).

Temperature
98.6 °F
36,11 C
99.3778 f
lukewarm
30C

نرمال‌سازی

Temperature (°C)
37.0
36.1
37.4
NULL
30.0

شکل ۴–۴: نرمال‌سازی دما. در تعداد ارقام اعشاری و استفاده از کاما و نقطه یکنواخت عمل کنید. کاراکترهای اضافی F، C یا ° را حذف کرده و مقادیری را که عددی نیستند کنار گذاشته یا بازنویسی کنید.

Citizenship
British
GB
🇬🇧
United Kingdom
UK
Great Britain

نرمال‌سازی

Citizenship
GB
GB
GB
GB
GB
GB

شکل ۴–۵: نرمال‌سازی اطلاعات شهروندی. آماده رسیدگی به موارد غیرمنتظره، به خصوص با رکوردهای قدیمی باشید. آیا اهالی اتحاد جماهیر شوروی را باید روس دانست؟ یوگسلاوها چطور؟

تبدیل ارز

Year	Money
1970	$ 500
1977	£ 440,000
1988	¥ 65,000
1999	$ 500

Year	USD
1970	500.00
1977	502.42
1988	508.40
1999	500.00

معادل‌سازی بر اساس تورم

Year	USD/2019
1970	6,514.85
1977	2,095.70
1988	1,086.32
1999	758.63

شکل ۴-۶: نرمال‌سازی ارزها در سال‌های مختلف. به دلیل تورم، یک دلار در سال ۱۹۷۰ قدرت خریدی برابر با شش دلار در سال ۱۹۸۸ داشت. علاوه بر این، برخی از ارزها حتی دیگر وجود ندارند، مانند لیر ایتالیا! برای نرمال‌سازی چنین اعدادی به نرخ ارز و تورم در تاریخ مورد نظر نیاز داریم.

Name	Dimensions
No. 5, 1948	8 ft × 4 ft
Las Meninas	318 x 276 cm
The Scream	91 cm, 73.5 cm
Monalisa	30¼ by 21⅞ in

→

Name	H (cm)	W (cm)
No.5, 1948	244	122
Las Meninas	318	276
The Scream	91	74
Monalisa	77	53

شکل ۴-۷: نرمال‌سازی ابعاد نقاشی‌ها.

پاک‌سازی: همیشه بررسی کنید که مقادیر شما از نوع مورد نظر و منطقی باشد. سلول‌های حاوی داده‌های غیرمنطقی یا نادرست را پاک کنید. به این کار پاک‌سازی داده‌ها[۱] می‌گویند.

ضربان قلب: به عنوان مدیر سیستم‌های هوش تجاری یک بیمارستان، گزارشی از اندازه‌گیری ضربان قلب بسیاری از بیماران تهیه می‌کنید. در حین بررسی داده‌ها، با مقادیر شگفت‌انگیزی مانند ۸۱۹ ضربان در دقیقه مواجه می‌شوید. در مورد این مقادیر چه باید کرد؟

Time	Patient	Rate (bpm)
07:00	472	61
07:01	677	78
07:03	780	819
07:04	472	180

ضربان قلب هرگز نمی‌تواند منفی باشد. یک جست‌وجوی سریع در اینترنت به ما اطلاع می‌دهد که بالاترین ضربان قلب ثبت شده تا کنون حدود ۶۰۰ ضربان در دقیقه بوده است. رکوردهای خارج از

---

[۱] Data Cleansing

محدوده‌ی `(0,660)` را می‌توان با خیال راحت کنار گذاشت[۱]. سلولی که `819` را نشان می‌دهد باید روی مقدار تهی یا `NULL` تنظیم یا به طور کامل حذف شود.

بسیار مراقب باشید وقتی که یک سلول فاقد داده است، هرگز از صفر به جای `NULL` استفاده نکنید. در مثال ثبت دما (شکل ۴-۴)، نرمال‌سازی عبارت `lukewarm` (به معای ولرم) به درجه‌ی انجماد `0.0` سانتیگراد اشتباه است! متأسفانه، این بی‌دقتی بسیار رایج است. برای مثال، بسیاری از نرم‌افزارها زمانی که نمی‌توانند مختصات مکانی را به دست آورند، مختصات `(0°,0°)` را به عنوان خروجی ارائه می‌دهند[۲]. صرف نظر از نوع داده‌هایی که جمع‌آوری کرده‌اید، مقادیر صفر را در داده‌های خود بررسی کنید؛ آیا این مقادیر رکوردهای عددی دقیقی هستند یا باید `NULL` باشند؟

تکراری‌ها: اطمینان حاصل کنید که جداول شما حاوی داده‌های تکراری نیستند: هر ردیف باید چیزی منحصر به فرد را ثبت کند. فرض کنید یک جدول را برای عنوان، سال انتشار و ژانر فیلم‌های مختلف ذخیره می‌کند. این جدول باید فقط یک ردیف برای هر فیلم داشته باشد. اگر در دو ردیف فیلم `"Metropolis, 1927, Sci-Fi"` نوشته شده است، یکی از آن‌ها را حذف کنید.

هنگامی که داده‌ها هنوز نرمال‌سازی نشده‌اند، تشخیص موارد تکراری دشوارتر است. اگر یک رکورد دارای ژانر `Sci-Fi` و رکورد تکراری آن، `Science Fiction` باشد، یک جستجوی ساده مشکل را مشخص نمی‌کند. اسامی تکراری با اشتباهات املایی یا املاهای متفاوت کلمات اغلب بدون شناسایی باقی می‌مانند. همیشه داده‌های خود را نرماسازی و پاک‌سازی کرده و ردیف‌های مشابه را به دقت بررسی کنید تا موارد تکراری را از بین ببرید. اگر از یک سیستم مدیریت پایگاه‌داده استفاده می‌کنید، جستجو کنید که کدام ابزارها و توابع داخلی می‌توانند به شما در انجام این کار کمک کنند.

ناشناس‌سازی داده‌ها

برای داده‌هایی که جزئیات زندگی افراد را مشخص می‌کنند نهایت احترام را قائل باشید. سوابق مالی، داده‌های مراقبت‌های بهداشتی و پیام‌های خصوصی تنها چند نمونه از اطلاعات بسیار حساس هستند که هرگز نمی‌توانند عمومی شوند. در بسیاری از سازمان‌ها، همه یک کپی از داده‌های حساسی که با آن کار

---

[۱] ضربان قلب اکثر انسان‌ها در حالت استراحت ۴۰ تا ۱۰۰ ضربان در دقیقه است. با این وجود، بسته به اینکه داده‌ها را برای چه کاری تحلیل می کنیم، گاهی اوقات خوب است که فضایی برای داده‌های خارق‌العاده اما معتبر باقی بگذاریم. در اینجا، یک حاشیه‌ی ۱۰ درصد بیشتر از بالاترین رکورد به صورت دلخواه انتخاب شده است.

[۲] در حقیقت، طبق بسیاری از منابع داده‌ای پاک‌سازی نشده، آن نقطه‌ی خالی در وسط اقیانوس اطلس یکی از شلوغ‌ترین مکان‌های روی زمین است. علاقمندان به فناوری حتی نام آن را نیز `Null Island` گذاشته‌اند.

می‌کنند نگه می‌دارند. در چنین شرایطی نشت داده‌ها فاجعه‌بار است! داده‌های خصوصی را با احتیاط زیاد مدیریت و ذخیره کنید.

برای کاهش خطرات، داده‌ها باید به‌گونه‌ای تغییر داده شوند که میزان اطلاعات قابل شناسایی شخصی کاهش یابند. به این کار **ناشناس‌سازی داده‌ها**[۱] می‌گویند. بسیاری از کشورها در حال حاضر قوانینی دارند که ناشناس‌سازی داده‌ها را در موارد خاص الزامی می‌کنند[۲].

حذف داده‌ها: داده‌هایی را که فقط برای شناسایی افراد مفید هستند، دور بریزید. به عنوان مثال، نام و نام خانوادگی، شماره‌ی شناسایی مالیاتی و تامین اجتماعی، شماره تلفن و آدرس ایمیل تقریباً هرگز هیچ کاربرد آماری ندارند. داده‌های غیرمرتبط با جمعیت‌شناسی، اهداف سازمانی و الگوهای رفتاری را می‌توان به سرعت حذف کرد.

محو کردن داده‌ها: برخی از داده‌های قابل شناسایی شخصی نباید حذف شوند. به عنوان مثال، سن اطلاعات جمعیتی مهمی است که اغلب رفتار را توضیح می‌دهد؛ آن را حفظ کنید. با این حال، می‌توان از تاریخ‌های دقیق تولد برای یافتن افراد و به خطر انداختن ناشناس بودن استفاده کرد. فقط سال تولد را ذخیره کنید: این کار اطلاعات کافی از سن را حفظ می‌کند و به صورت شخصی قابل شناسایی نیست. فرآیند فدا کردن دقت به نفع ناشناس ماندن داده‌ها را **محو کردن داده‌ها**[۳] نامیده می‌شود.

به راه‌های ساده‌ای برای کاهش دقت داده‌های حساس فکر کنید. فرض کنید برای سازمانی کار می‌کنید که نیاز به پیگیری هزینه‌های مشتری دارد. آیا نگهداری سوابق به اندازه‌ی یک سنت[۴] به درک رفتار آن‌ها کمک می‌کند؟ سنت‌ها را رها کنید تا رکوردها ناشناس‌تر شوند. این پدیده در مورد آدرس خانه‌ی مشتریان نیز صدق می‌کند: کد پستی احتمالاً برای هدف قرار دادن افراد در یک منطقه‌ی خاص کافی است. دقت غیر ضروری را در داده‌ها محو کنید.

شناسایی مجدد داده‌ها: حتی داده‌هایی که با دقت ناشناس شده‌اند، گاهی اوقات می‌توانند از حالت «ناشناس» خارج شوند. این فرآیند **شناسایی مجدد داده‌ها**[۵] نامیده می‌شود: اگر اطلاعات کافی در مورد

---

۱ Data Anonymization

۲ برای مثال، به مقررات حفاظت از داده‌های عمومی اروپا (GDPR) مراجعه کنید.

۳ Data Blurring

۴ یک سنت معادل یک صدم دلار است. م.

۵ Data Re-Identification

شخصی دارید، می‌توانید رکوردهای ناشناس را فیلتر و مشخص کنید که کدام یک متعلق به آن شخص است.

یک مجموعه‌داده‌ی ناشناس از یک بیمارستان را در نظر بگیرید. اگر سن، جنس، مدت اقامت و نوع بیماری فردی را می‌دانید، احتمالاً می‌توانید سوابق دقیق او را در مجموعه‌داده‌های ناشناس مطابقت دهید. ناشناس‌سازی داده‌ها یک عامل بازدارنده است نه یک راه حل. به داده‌های ناشناس همان سطح حفاظت و محرمانه بودن منابع خام خود را اختصاص بدهید.

تکرارپذیری

اتمام مرحله‌ی پردازش داده‌ها در اسرع وقت، بدون یادداشت‌برداری و مستندسازی هر مرحله، می‌تواند وسوسه انگیز باشد. گفته می‌شود که چنین داده‌هایی که به سرعت جمع‌آوری شده، به خوبی پاک‌سازی نشده و به صورت ضعیف مستندسازی شده‌اند، موقتی آماده شده‌اند. به طور کلی، پردازش داده‌ها به این روش کار بسیار بدی است.

اغلب، نسخه‌های جدیدی از داده‌های خام ظاهر می‌شوند. گاهی اوقات، ما به سادگی اطلاعات جدیدی را با رکوردهای اضافی دریافت می‌کنیم. اگر فرایند پردازش به طور موقت انجام شود، تکرار آن‌ها در داده‌های جدید دشوار و زمان‌بر است. علاوه بر این، بررسی این‌که آیا خطایی در طول تبدیل داده‌های موقتی رخ داده است یا خیر، دشوار است زیرا نمی‌توانیم وضعیت مجموعه‌داده‌ها را با مراحل قبلی مقایسه کنیم. علاوه بر این، اصلاح یک خطای شناسایی شده می‌تواند بسیار دشوار باشد.

به منظور اجتناب از این مسائل، دانشمندان باتجربه در زمینه‌ی داده‌ها اطمینان حاصل می‌کنند که هر تغییری را می‌توان به راحتی تکرار کرد. به این فرایند **تکرارپذیری**[۱] می‌گویند. شما می‌توانید با مستند کردن تمام تغییرات موقتی که در یک گزارش انجام می‌دهید، به سطح مناسبی از تکرارپذیری دست پیدا کنید، اما همه‌ی عملیات همچنان باید به صورت دستی انجام شوند.

بهترین روش نوشتن برنامه‌ای است که تمام مراحل تبدیل داده‌ها را به صورت یک خط لوله انجام دهد. به این ترتیب، زمانی که منابع داده تغییر می‌کنند، می‌توانیم با اجرای برنامه بر روی منابع خام جدید، تبدیل‌ها را دوباره تکرار کنیم. هنگام کار در یک تیم، هر عضو می‌تواند این برنامه را بررسی کند، این کار یافتن اشکالات را آسان‌تر می‌کند. در ادامه اگر اشکالی پیدا شد، می‌توان آن را در برنامه برطرف کرد و با اجرای مجدد برنامه‌ی ارتقا یافته، یک مجموعه‌داده‌ی تصحیح شده جدید تولید کرد.

---

[۱] Reproducibility

مهم‌تر از همه، هرگز منابع خام خود را مستقیماً تغییر ندهید: داده‌های تازه آماده و ناشناس‌سازی شده باید در فایل‌های جدید نوشته شوند. منابع خام شما حقیقت را حفظ می‌کنند تا اگر در وضعیتی مشکوک به وجود خطا شدید و احساس کردید باید عملیات را به عقب برگردانید، از آن‌ها استفاده کنید. اگر منابع خام حاوی اطلاعات حساس هستند، فراموش نکنید که دسترسی آن‌ها محدود شده باشد.

اکنون که مجموعه‌داده‌های شما کاملاً پاک‌سازی شده است، برای مرحله بعدی آماده هستید: کاوش. اینجاست که لذت واقعی شروع می‌شود. در بخش‌های بعدی، مفاهیم بنیادین **تحلیل کاوشگرانه‌ی داده‌ها**[۱] یا EDA را خواهید دید. شما یاد خواهید گرفت که داده‌های خود را *خلاصه* و *مصور* کنید. همانطور که داده‌ها را می‌بینید، شهود شما توسعه می‌یابد. این شهود می‌تواند سؤالاتی را در مورد پدیده‌های اساسی که داده‌ها را شکل داده‌اند ایجاد کند. این پرسش‌ها به نوبه‌ی خود می‌توانند شما را در تحلیل عمیق‌تر راهنمایی کنند: وقتی می‌دانید به دنبال چه هستید، پیدا کردن آن چیز همیشه آسان‌تر است.

۳-۴- خلاصه‌سازی

ما می توانیم بسیاری از ویژگی‌های مهم یک مجموعه‌داده‌ها را در چند عدد کلیدی خلاصه کنیم. این به کار ما یک ایده‌ی سریع از آنچه داده‌ها می‌گویند، بدون نیاز به بازرسی تک به تک رکوردها، ارائه می‌دهد. بیایید برخی از مفیدترین اعداد خلاصه‌سازی را ببینیم:

تعداد

با نام مستعار n، تعداد[۲] رکوردهای منفرد شما را نشان می‌دهد. به عنوان مثال، اگر جدول مراقبت‌های بهداشتی، بستری شدن ۲۲ بیمار را در بیمارستان نشان می‌دهد پس $n = 24$.

متوسط‌ها

متوسط[۳]، مقدار «مرکزی» یا «معمولی» یک گروه است. چندین نوع متوسط وجود دارد. معروف‌ترین نمونه، **میانگین**[۴] است: مجموع همه‌ی مقادیر، تقسیم بر تعداد آن‌ها. میانگین‌ها متوسط‌های خوبی هستند، اما کامل نیستند. بیایید ببینیم چرا:

> **حباب بیت‌کوین:** پنج دوست ادعا می‌کنند که متخصص بلاک‌چین هستند و شما را متقاعد می‌کنند که ۱۰۰ دلار ارز دیجیتال بخرید. هر یک از این افراد به

---

[۱] Exploratory Data Analysis یا EDA

[۲] Count

[۳] Average

[۴] Mean

شما می‌گوید که فکر می‌کند ارزش کیف پول شما در عرض یک هفته و بسته به ارزی که می‌خرید، چند دلار تغییر می‌کند. شما بیت‌کوین، اتر و دوج‌کوین را مد نظر دارید. کدام یک از آن‌ها را باید بخرید؟

Expert	BTC	ETH	DOGE
Adam	8	NULL	7
Gavin	6	2	7
Roger	NULL	4	7
Sergey	NULL	3	-1
Vitalik	7	21	-2

کدام بهتر است؟ BTC(8,6,7) یا ETH(2,4,3,21)؟ مقایسه‌ی گروه‌های اعداد دشوار است. برای سهولت کار، می‌توانیم هر گروه را در یک عدد خلاصه کنیم. بیایید میانگین را بررسی کنیم:

پیش بینی درآمد به دلار

Coin	Adam	Gavin	Roger	Sergey	Vitalik	Mean
BTC	8	6	·	·	7	7.0
ETH	·	2	4	3	21	7.5
DOGE	7	7	7	−1	−2	3.6

میانگین بالاتر اتر نشان می‌دهد که بهترین پتانسیل درآمد را دارد. با این حال، توجه داشته باشید که میانگین اتر به شدت تحت تأثیر یک پیش‌بینی بسیار بالا قرار گرفته است. شاید بیت‌کوین با وجود میانگین پایین‌تر، شرایط بهتری داشته باشد.

در مقابل، **میانه**[۱] متوسطی است که تحت تأثیر مقادیر شدید گروه قرار نمی‌گیرد. برای به دست آوردن میانه، مقادیر را به ترتیب عددی مرتب و مقدار وسط را انتخاب کنید. برای دوج‌کوین مقادیر (-2, -1, 7, 7, 7) را داریم و مقدار میانه برابر با 7 است.. اگر تعداد عناصر زوج باشد، هیچ مقدار مرکزی واحدی وجود ندارد، مانند اتر با مقادیر (2, 3, 4, 21). در این شرایط میانه، میانگین دو

[۱] Median

عنصر مرکزی است. در هر صورت، میانه همیشه مرکز گروه را نشان می‌دهد: از نیمی از مقادیر بیشتر و از نیمی دیگر کمتر است.

	پیش‌بینی درآمد به دلار					
Coin	Adam	Gavin	Roger	Sergey	Vitalik	Median
BTC	8	6	·	·	7	7.0
ETH	·	2	4	3	21	3.5
DOGE	7	7	7	−1	−2	7.0

میانه هم کامل نیست. توجه کنید که چگونه بیت‌کوین و دوج‌کوین میانه‌ی هفت دلاری را نشان می‌دهند، در حالی که کارشناسان پیش‌بینی‌های بسیار متفاوتی ارائه می‌دهند: ۱۰۰٪ کارشناسان موافق هستند که بیت‌کوین حدود هفت دلار درآمد خواهد داشت، در حالی که ۴۰٪ ایشان می‌گویند که دوج‌کوین ضرر خواهد داد!

چون میانگین تمام اعداد یک گروه را در نظر می‌گیرد، به مقادیر شدید بسیار حساس است؛ ولی میانه تحت تأثیر افراط و تفریط قرار نمی‌گیرد، اما تمام مقادیر دیگر به غیر از مقدار مرکزی را نادیده می‌گیرد. میانگین و میانه با هم راه‌های مکملی برای متوسط اعداد هستند[۱].

پراکندگی

چه از میانگین و چه از میانه استفاده کنیم، نمی‌توانیم بدانیم این مقادیر چقدر دقیق گروه اعداد خود را نشان می‌دهند. برای مثال، `(9, 10, 11)` و `(0, 10, 20)` هر دو دارای میانگین `10` هستند حتی اگر مقادیر گروه اول بسیار نزدیک به `10` باشند.

انحراف معیار[۲] گروهی از مقادیر نشان می‌دهد که آن‌ها چقدر از میانگین فاصله دارند. هنگامی که انحراف معیار بزرگتر باشد، اعداد پراکندگی بیشتری دارند. بیشتر زبان‌های برنامه‌نویسی دارای توابع داخلی برای محاسبه‌ی انحراف معیار هستند[۳]. بیایید ببینیم چگونه می‌توان از آن استفاده کرد:

---

[۱] انواع بیشتری از متوسط‌ها مانند **میانگین هارمونیک** وجود دارد که در فصل بعدی مورد استفاده قرار خواهد گرفت. برخی از متوسطها برای نرخ رشد ایده‌آل هستند و برخی دیگر برای سرعت سفر. برای کسب اطلاعات بیشتر، به `http://code.energy/average` مراجعه کنید.

[۲] `Standard Deviation`

سرمایه‌گذاری هوشمندانه: مادربزرگ شما می‌خواهد در فناوری سرمایه گذاری کند! از آنجایی که او قصد دارد هزینه‌ی عضویت باشگاه خود را با سود حاصله پرداخت کند، به شما نیاز دارد که مشخص کنید کدام یک از سه سهم مورد نظر او می‌تواند پایدارترین درآمد سالانه را فراهم کند. بر اساس سابقه‌ی پنج ساله‌ی بازدهی آن‌ها (به درصد)، چگونه می‌توانید نوسانات این سه سهم را با یکدیگر مقایسه کنید؟ (جدول صفحه‌ی بعد را ببینید.)

Year	AAPL	GOOG	MSFT
2014	40.0	-2.4	27.2
2015	-2.8	46.6	22.2
2016	12.2	1.7	14.7
2017	48.2	32.9	40.2
2018	-5.1	-0.8	20.2

برای هر سهم، می‌توانید انحراف معیار بازده‌ی سالانه‌ی آن را محاسبه کنید، زیرا این مقدار سنجه‌ی خوبی برای میزان نوسان است.

	بازده‌ی کل به درصد					
Stock	**2014**	**2015**	**2016**	**2017**	**2018**	**Std Dev.**
AAPL	40.0	−2.8	12.2	48.2	−5.1	21.8
GOOG	−2.4	46.6	1.7	32.9	−0.8	20.2
MSFT	27.2	22.2	14.7	40.2	20.2	8.6

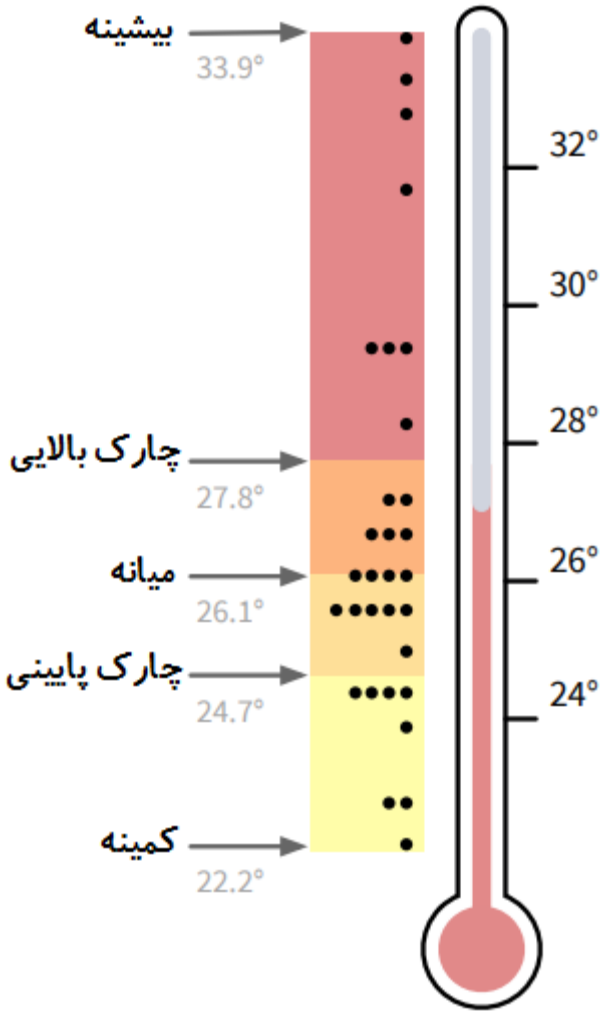
[۳] **واریانس** یکی دیگر از معیارهای رایج پراکندگی است. واریانس صرفاً مربع انحراف معیار است.

در میان این سه سهم، APPL و GOOG نوسانات مشابهی را نشان دادند، در حالی که MSFT سال به سال بازدهی پایدارتری داشت.

هر بار که میانگینی را محاسبه می‌کنید، می‌توانید از انحراف معیار استفاده کنید: این مقدار می‌تواند معیاری برای دقت در تیراندازی با کمان، تغییرات دما در یک شهر یا نابرابری درآمد در یک جمعیت باشد. به یاد داشته باشید می‌توانید از آن برای نشان دادن میزان دقت میانگین در توصیف مقادیر خلاصه‌شده استفاده کنید.

خلاصه‌ی پنج عددی

فرض کنید چند شهر را در دارید که تابستان خود را در آن‌ها بگذرانید، و برای هر شهر یک پایگاه داده حاوی بیشینه دمای هر روز از سال ۲۰۱۸ پیدا کرده‌اید. چگونه می‌توانید تعداد روزهای خوش آب‌وهوای هر شهر را تخمین بزنید؟ برای شروع، میانگین بیشینه دمای روزانه مفید است: روزها را به گونه‌ای تقسیم می‌کند که نیمی از میانگین خنک‌تر و نیمی گرم‌تر بوده‌اند. می‌توانیم این روند را تکرار کنیم و نصف‌ها را به چهار قسمت تقسیم کنیم:



شکل ۴–۸: بیشینه دمای روزانه (به درجه‌ی سانتیگراد) در لس آنجلس (ژوئیه‌ی ۲۰۱۸). هر نقطه نشان‌دهنده‌ی یک روز است. از آنجایی که $n = 31$، هر چارک داده‌ها شامل هشت نقطه است.

پس از اینکه میانه، نقاط داده‌ی مرتب‌شده را به دو نیم تقسیم کرد، **چارک بالایی** نیمه بالایی را به دو قسمت و **چارک پایینی** نیمه‌ی پایینی را به دو قسمت تقسیم می‌کنند. چارک‌های بالایی و پایینی همراه با میانگین، کمینه و بیشینه، **خلاصه‌ی پنج عددی**[۱] را تشکیل می‌دهند. بسیاری از زبان‌های برنامه‌نویسی مدرن دارای کتابخانه‌هایی برای تولید آسان این اعداد هستند. در اینجا خلاصه‌ای از اوج دمای روزانه چند شهر در کل سال ۲۰۱۸ ($n = 365$) آمده است:

---

[۱] Five-Number Summary

جدول ۴–۵: خلاصه‌ی پنج عددی دمای بیشینه

City	کمینه Min	چارک پایینی Lower Quartile	میانه Median	چارک بالایی Upper Quartile	بیشینه Max
Rio de Janeiro	20.1	26.8	29.9	32.8	39.1
Los Angeles	13.9	19.4	22.2	25.0	34.4
Honolulu	23.9	27.8	29.4	30.6	33.3
Minneapolis	−17.1	0.9	10.6	26.7	37.8

این خلاصه‌ها عوامل جالبی را نشان می‌دهند که هنگام انتخاب محل گذراندن تابستان باید در نظر بگیرید: حتی اگر مینیاپولیس یک چهارم سال یخبندان باشد، گرم‌ترین ۹۱ روز آن‌جا گرم‌تر از لس‌آنجلس هستند!

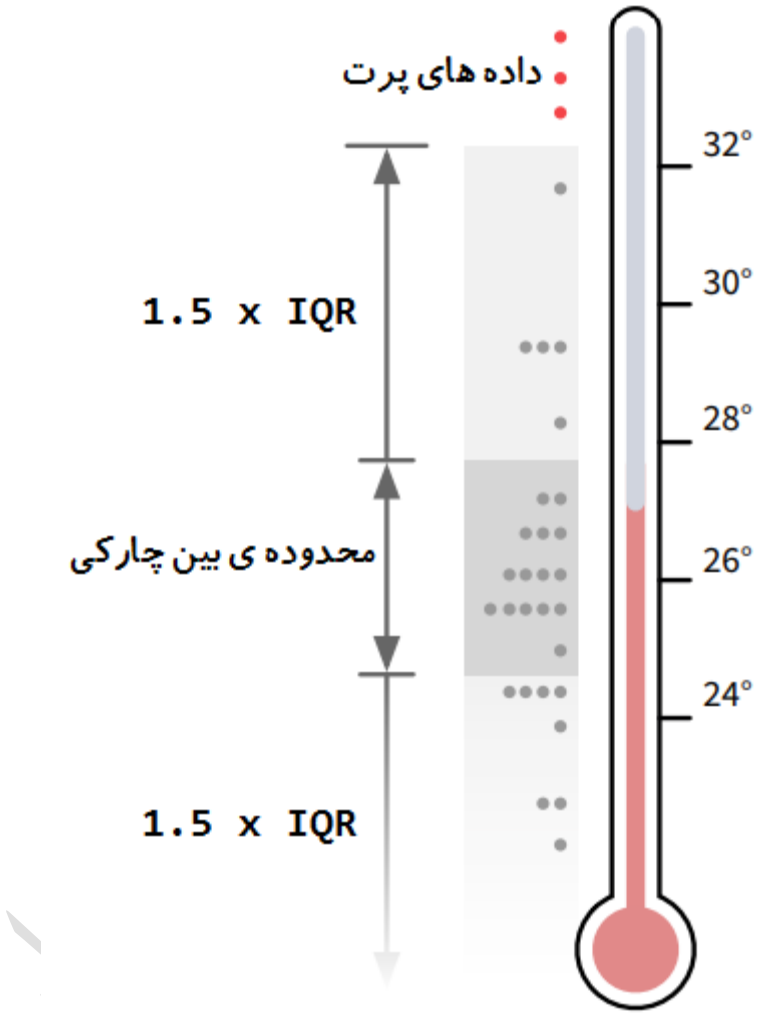
داده‌های پرت: گاهی اوقات به مقادیر بالا یا پایین غیرعادی برخورد می‌کنیم، که با در نظر گرفتن مقادیر دیگر در ستون، منطقی به نظر نمی‌رسند. این مقادیر معمولاً رویدادهای شدیدی را نشان می‌دهند که ماهیت آن‌ها به طور قابل توجهی متفاوت است. جدا کردن این مقادیر و مطالعه‌ی دقیق آن‌ها مفید است. به منظور شناسایی منظم این مقادیر، دانشمندان مرزهای نرمال بودن را تعریف می‌کنند. مقادیر خارج از مرزها، در صورت وجود، به عنوان **داده‌های پرت**[۱] علامت‌گذاری می‌شوند.

روش‌های زیادی برای تعریف مرز نرمال بودن وجود دارد. به عنوان مثال، اغلب با کم کردن چارک پایینی از چارک بالایی شروع می‌کنیم. این تفاوت **محدوده‌ی بین چارکی**[۲] یا IQR نامیده می‌شود و مقادیری که بیشتر از $1.5 \times IQR$ از آن فاصله داشته باشند، مقادیر پرت در نظر گرفته می‌شوند.

---

۱ Outlier

۲ Interquartile Range



شکل ۴–۹: شناسایی داده‌های پرت شکل ۴–۸ محدوده‌ی بین چارکی ۳.۱ درجه است، بنابراین $1.5 \times IQR \approx 4.5$ درجه. بنابراین بالاترین مرز ۳۲.۳ درجه و پایین‌ترین مرز ۲۰.۲ درجه است.

در اندازه‌گیری دما در ژوئیه‌ی ۲۰۱۸ در لس‌آنجلس هیچ نقطه‌ی پرت پایین‌تری وجود ندارد: همه مقادیر بالاتر از مرز پایینی هستند (شکل ۴–۹). از سوی دیگر، خوشه‌ای متشکل از چهار روز گرم وجود دارد که سه مورد از آن‌ها پرت با بیشینه‌ی دمای ۳۲.۴ درجه، ۳۳.۳ درجه و ۳۳.۹ درجه هستند.

این مقادیر در واقع ماهیت متفاوتی داشتند: یک موج گرمای رکوردشکن جنوب کالیفرنیا را از ۶ ژوئیه تا ۹ ژوئیه‌ی ۲۰۱۸ در نوردید. در روز هفتم، دمای بالا باعث شد آتش‌سوزی عظیمی بسیاری از

خیابان‌های سانتا باربارا را فرا بگیرد. بیش از دو هزار نفر مجبور به تخلیه‌یمنطقه شدند و دماسنج‌های داخل محوطه‌ی دانشگاه UCLA گرم‌ترین دمای خود را ثبت کردند.

خلاصه‌ی طبقه‌ای

رکوردهای داده‌های طبقه‌بندی‌شده حاوی برچسب‌ها هستند نه اعداد، بنابراین نمی‌توانیم مستقیماً آن‌ها را با میانگین، انحراف معیار یا خلاصه‌ی پنج عددی خلاصه کنیم. با این حال، می‌توانیم آن‌ها را بشماریم.

ما معمولاً داده‌های طبقه‌بندی‌شده را با شمارش تعداد دفعاتی که هر طبقه رخ می‌دهد، خلاصه می‌کنیم. وقتی تعداد طبقه‌ها به صورت درصدی از تعداد کل، یعنی n، بیان می‌شود، تعداد تکرار یا فرکانس هر طبقه را به دست می‌آوریم. به عنوان مثال، جدولی با تعداد مدال المپیک و کشور برنده‌ی آن در بازی‌های تابستانی ۱۸۹۶ را می‌توان با شمارش پرتکرارترین کشورها خلاصه کرد:

Country	Count	Frequency
Greece	46	38 %
United States	20	16 %
Germany	13	11 %
France	11	9 %
Great Britain	7	6 %
Others	25	20 %

ماتریس همبستگی

وقتی باران می‌بارد، جاده‌ها لغزنده‌تر می‌شوند و دید کاهش می‌یابد. بنابراین، شرکت‌های بیمه‌ی خودرو انتظار حوادث رانندگی بیشتری را دارند. تصور کنید که شما یک شرکت بیمه هستید و مجموعه داده‌ای با سه ستون جمع‌آوری کرده‌اید که هر ردیف آن حاوی تاریخ، میزان بارندگی در آن تاریخ و تعداد مربوط به حوادث رانندگی است. شما انتظار دارید که مقادیر دو ستون آخر با هم افزایش یا کاهش یابند: هر چه باران بیشتر باشد، تصادفات بیشتری رخ می‌دهد.

وقتی دو ستون از این قبیل به هم مرتبط هستند، می‌گوییم ستون‌ها همبسته هستند. ما می‌توانیم قدرت این پدیده را با عددی به نام **ضریب همبستگی**[۱] یا به اختصار «**همبستگی**» بیان کنیم[۲]. ضریب همبستگی صفر به این معنی است که مقادیر به هیچ وجه با هم متفاوت نیستند. ضریب همبستگی ۱ نشان می‌دهد که دو ستون با هم کاملاً متفاوت هستند.

---

[۱] Correlation Coefficient

[۲] چندین روش برای اندازه گیری همبستگی وجود دارد و هر کدام یک ضریب تقریباً متفاوت به دست می‌دهند. در اینجا، از ضریب همبستگی پیرسون، استفاده می‌کنیم.

در عمل، ضریب همبستگی به ندرت دقیقاً برابر با صفر یا ۱ است. برای جستجوی سریع روابط بین ستون‌های جداول، مقادیر همبستگی را برای هر جفت ستون در جدول محاسبه می‌کنیم. نتایج را می‌توان در یک **ماتریس همبستگی**[۱] ارائه کرد که به طور خلاصه نشان می‌دهد که چقدر ستون‌ها با هم متفاوت هستند.

برای مثال، فرض کنید در شهر لس آنجلس کار می‌کنید و جدولی درست کرده‌اید که در آن هر ردیف یک روز است و ستون‌ها نشان‌دهنده‌ی دمای بیشینه‌ی روزانه، میانگین سرعت باد در روز، حجم باران، تعداد تصادفات رانندگی و تعداد دعواهای ثبت شده توسط اداره پلیس هستند. ساخت یک ماتریس همبستگی روابط بین این متغیرها را نشان می‌دهد:

جدول ۴-۷: همبستگی اتفاقات روزمره (لس‌آنجلس، ۲۰۱۸، تعداد روزها:۳۶۵)

باد	1.	0.21	−0.18	0.05	−0.03
باران	0.21	1.	−0.19	0.19	−0.11
دما	−0.18	−0.19	1.	0.18	0.28
تصادفات	0.05	0.19	0.18	1.	0.07
دعواها	−0.03	−0.11	0.28	0.07	1.

از بالا سمت چپ به پایین سمت راست، یک قطر از مقادیر ۱ وجود دارد: ستون‌ها کاملاً با خودشان همبستگی دارند. علاوه بر این، نیمه‌ی سمت راست بالای ماتریس متقارن با نیمه‌ی سمت چپ پایین است: همبستگی بین باد و تصادفات برابر با همبستگی بین تصادفات و باد است. این مقادیر خاکستری شده هیچ اطلاعات اضافی ارائه نمی‌دهند و به طور کلی برای سادگی و اختصار از ماتریس حذف می‌شوند.

توجه داشته باشید که برخی از ستون‌ها دارای ضرایب همبستگی منفی هستند، مانند دما و باران. مقادیر همبستگی منفی نشان می‌دهند که آن ستون‌ها با هم متفاوت و *معکوس* هستند: با ثبت باران بیشتر، دمای بیشینه‌ی کمتری ثبت می‌شود.

وقتی مقادیر همبستگی نزدیک به صفر هستند، می‌گوییم که آن‌ها تقریباً همبسته نیستند. به عنوان مثال، به نظر نمی‌رسد تصادفات تحت تأثیر سرعت باد باشند. برعکس، قوی‌ترین همبستگی ما بین دعواها و دما است. به نظر می‌رسد داده‌ها حاکی از آن هستند که مجرمان زمانی که هوا گرم‌تر است، تهاجمی‌تر هستند.

---

[۱] Correlation Matrix

چنین ماتریس همبستگی به ما کمک می‌کند تا چگونگی ارتباط ستون‌ها با یکدیگر را کشف کنیم. وقتی دو متغیر همبسته هستند، علل را در دنیای واقعی بررسی کنید، ممکن است اکتشافات جالبی داشته باشید.

علیّت: اینجا یک دام وجود دارد. وقتی بین دو پدیده همبستگی پیدا می‌کنیم، اغلب سریع فرض می‌کنیم که یکی علت دیگری است. این امر همیشه درست نیست. به عنوان مثال، مراکز خرید معمولاً همبستگی بالایی بین فروش عینک آفتابی و بستنی دارند. آیا این به آن معنی است که استفاده از عینک آفتابی باعث می‌شود هوس بستنی کنید؟ یا این‌که بستنی خوردن باعث حساس شدن چشمان شما می‌شود؟ خیر. همیشه به خاطر داشته باشید که همبستگی *به معنای علّیت*[1] *نیست*.

شکل ۴-۱۰: «همبستگی» دریافت شده از http://xkcd.com.

تبدیلات: برای پیدا کردن همبستگی‌ها، ستون‌ها اغلب باید تبدیل شوند. به عنوان مثال، تصادفات ناشی از لغزش در شهر شما ممکن است با میزان بارندگی یا سرعت وسیله‌ی نقلیه ارتباط ضعیفی داشته باشند، اما به شدت با میزان بارندگی ضرب در مجذور حد مجاز سرعت ارتباط دارند. پیدا کردن تبدیلاتی که چنین همبستگی‌هایی را آشکار می‌کنند، می‌تواند مشکل باشد. اگر ضریب همبستگی دو ستون یا تبدیل آن‌ها نزدیک به صفر باشد، لزوماً به این معنی نیست که آن دو متغیر به یکدیگر مرتبط نیستند.

خلاصه کردن اعداد، موجب سادگی و کاهش حجم اطلاعاتی می‌شود که باید در نظر بگیریم. با این حال، بسیاری از جزئیات دور از دسترس ما باقی می‌مانند و یافتن مسیر پیشرفت می‌تواند دشوار باشد. گرفتن اطلاعات بیشتر با استفاده از اشکال و رنگ‌ها مرحله‌ی بعدی کاوش است. یک عکس به اندازه‌ی هزاران عدد می‌ارزد!

---

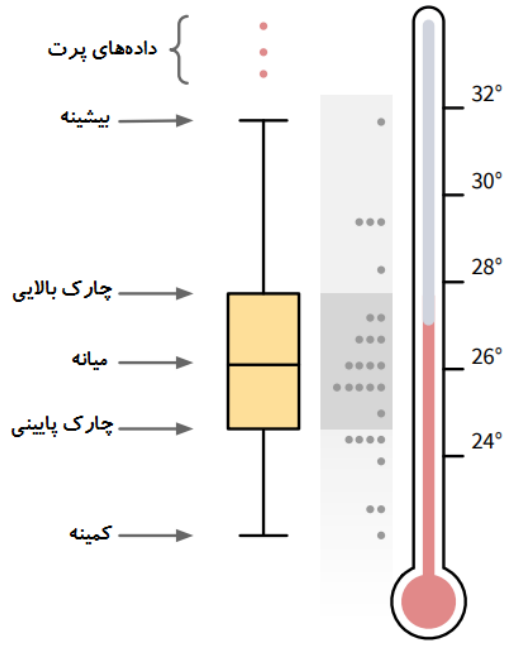
[1] Casuality

۴-۴- مصورسازی

میانگین، انحراف معیار و خلاصه‌ی پنج عددی یک نمای کلی مفید اما ساده از ماهیت یک مجموعه‌داده ارائه می‌دهند. برای کشف سرنخ‌های بیشتر، باید به غرایز تشخیص الگوی بصری خود متوسل شویم. نمودارها و اشکال به ما این امکان را می‌دهند که داده‌ها را ببینیم، الگوها و تفاوت‌های آن را بررسی و ناهنجاری‌های ناشی از رویدادهای خارق‌العاده یا، معمولاً، اشتباهات پردازشی ساده را آشکار کنیم.

نمودارهای جعبه‌ای

یک نمایش گرافیکی از خلاصه‌ی پنج عددی وجود دارد که به آن **نمودار جعبه‌ای**[۱] می‌گویند. چارک‌های بالایی و پایینی به عنوان دو ضلع بالا و پایین یک جعبه رسم می‌شوند و یک خط افقی کادر را در میانه تقسیم می‌کند. دو خط از بالا و پایین جعبه و تا مقادیر بیشینه و کمینه‌ی داده‌های غیرپرت نیز رسم می‌شوند. داده‌های پرت اغلب به عنوان نقاط جداگانه اضافه می‌شوند. بیایید یک نمودار جعبه‌ای برای دمای ژوییه‌ی ۲۰۱۸ در لس‌آنجلس بسازیم:

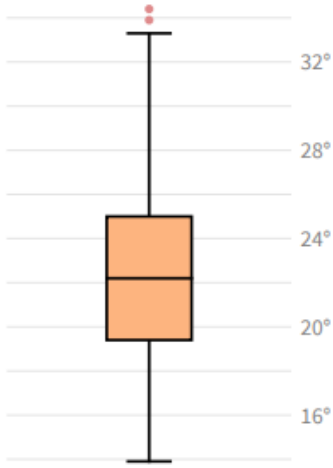


شکل ۴-۱۱: نمودار جعبه‌ای از دمای بیشینه‌ی روزانه در ژوییه‌ی ۲۰۱۸ لس‌آنجلس (۳۱ روز)، ساخته شده بر اساس شکل ۴-۸ (مقادیر بیشینه و کمینه با صرف نظر کردن از داده‌های پرت)

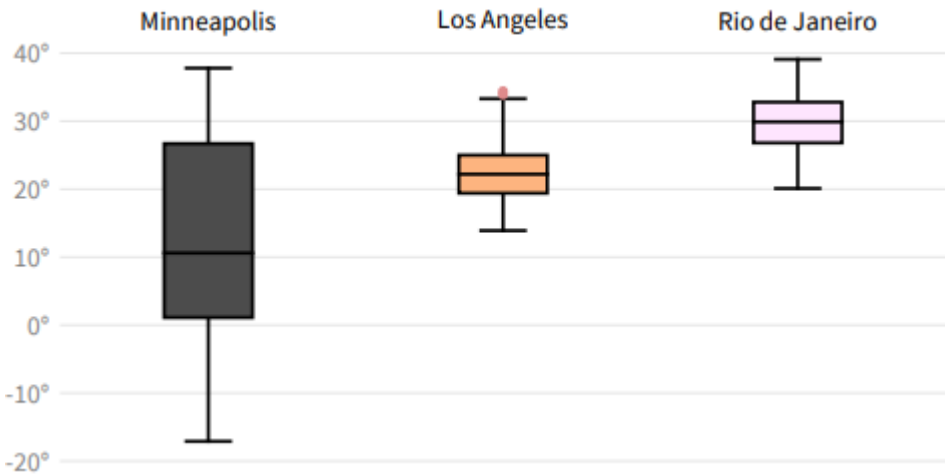
---

[۱] Box Plot

توجه داشته باشید که تعمیم داده‌ها به کل سال بر نمودار جعبه‌ای تأثیر می‌گذارد (شکل ۴-۱۲). حال بررسی کنید که چگونه می‌توان از نمودارهای جعبه‌ای برای مقایسه‌ی شهرهای مختلف در قالب یک نمودار استفاده کرد (شکل۴-۱۳).



شکل ۴-۱۲: نمودار جعبه ای بیشینه دمای روزانه در لس‌آنجلس در تمام سال ۲۰۱۸ (۳۶۵ روز). فقط دو داده‌ی پرت باقی مانده است زیرا کل مجموعه‌ی ۳۶۵ داده در محدوده‌ی بین چارکی دارد و مرزهای نرمال متفاوت از مجموعه‌ی منحصر به ماه ژوییه قرار هستند.



شکل ۴-۱۳: نمودارهای جعبه‌ای دمای بیشینه‌ی روزانه در سال ۲۰۱۸. مینیاپولیس و ریودوژانیرو داده‌های پرت ندارند.

در حالی که یک جدول خواندن مقادیر یک خلاصه‌ی پنج عددی را آسان می‌کند، نمودار جعبه‌ای مقایسه توزیع‌های توصیف شده توسط خلاصه‌های پنج عددی مختلف را آسان‌تر می‌نماید.

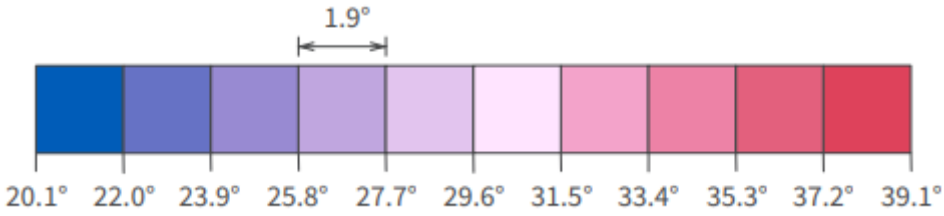
هیستوگرام‌ها

خلاصه‌ی پنج عددی زیرمجموعه‌های تحت پوشش گروه‌های اعداد را توصیف می‌کند، و شما یاد گرفته‌اید که آن را در دو مرحله به دست آورید: (الف) نقاط داده‌ای را به گروه‌هایی با اندازه‌ی مساوی تقسیم کرده، و (ب) ببینید که چه محدوده‌هایی را پوشش می‌دهند. برعکس آن، یک **هیستوگرام**[۱] یا **نمودار ستونی** دقیقاً محدوده‌ای را نشان می‌دهد که داده‌ها در آن متمرکز شده‌اند و از طریق فرآیند معکوس ایجاد می‌شود:

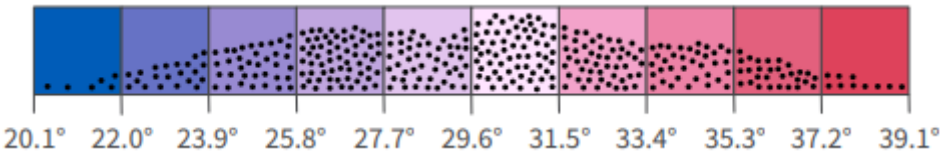
الف) محدوده را به بازه‌هایی با اندازه‌ی یکسان که **بین**[۲] نامیده می‌شوند، تقسیم کنید،

ب) مشخص کنید که هر بین دارای چند نقطه است.

در نهایت، می‌توانیم (ج) هر بین را در قالب یک ستون رسم کنیم که ارتفاع آن نشان‌دهنده‌ی تعداد نقاط داده‌ای آن است. شکل بعد ساخت یک هیستوگرام با ۱۰ بین را نشان می‌دهد، اما می‌توانستیم از هر عدد دیگری نیز استفاده کنیم. شکل‌های بعد از آن نشان می‌دهند که تعداد بین‌های مختلف چگونه نمودار را تغییر داده و چگونه هیستوگرام‌ها می‌توانند به ما در مقایسه‌ی دما در شهرهای مختلف کمک کنند.



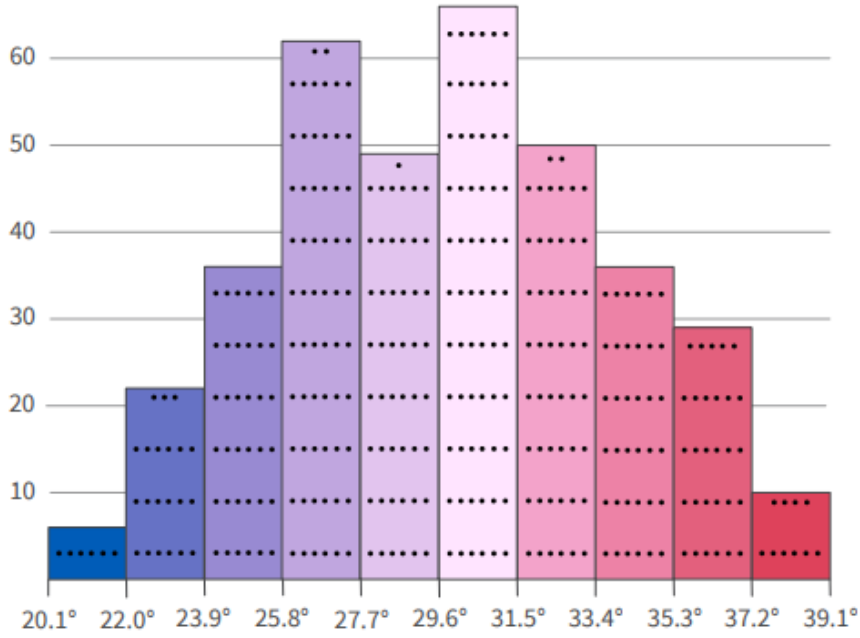
الف) تقسیم محدوده‌ی دما به ۱۰ بین.



ب) قرار دادن دمای بیشینه‌ی ۳۶۵ روز سال در بین مربوطه.

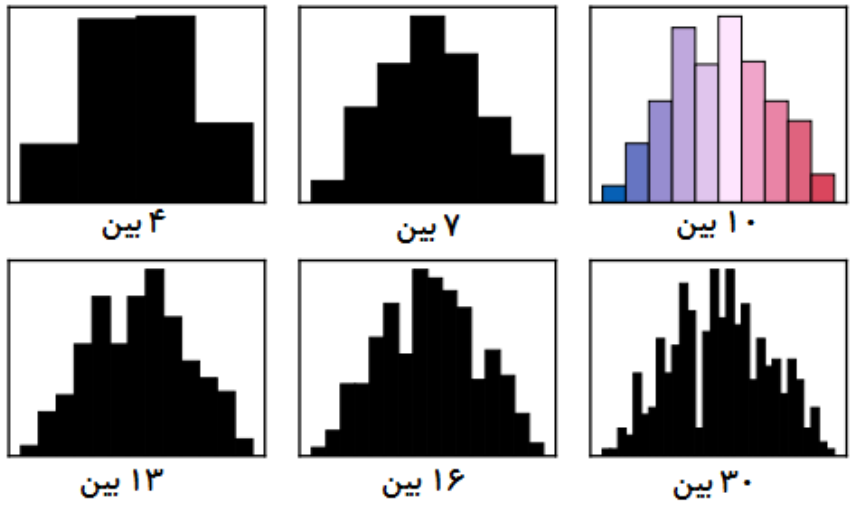
---

[۱] Histogram
[۲] Bin

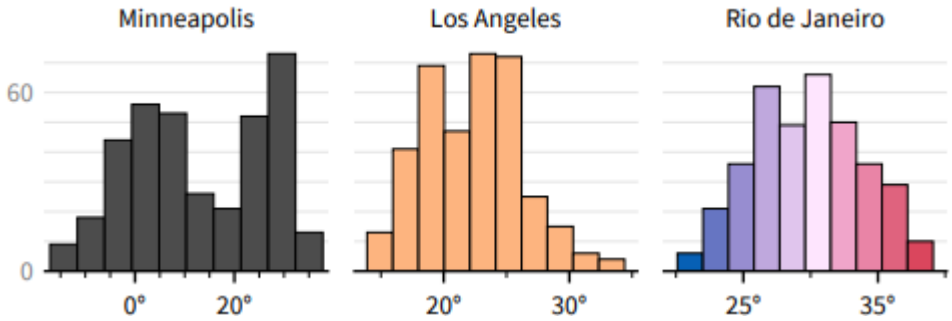


ج) هر بین را آن‌قدر بکشید که ارتفاع آن نشان‌دهنده‌ی تعداد نقاطش باشد.

شکل ۴-۱۴: ایجاد هیستوگرام دمای بیشینه‌ی روزانه در ریودوژانیرو، که از ۲۱.۱ درجه‌ی سانتی‌گراد تا ۳۹.۱ درجه‌ی سانتی‌گراد در سال ۲۰۱۸ متغیر بوده است. توجه داشته باشید که در هیستوگرام‌های معمولی، همه‌ی بین‌ها یک رنگ دارند.



شکل ۴-۱۵: شش هیستوگرام از همان مجموعه‌داده. بین‌های بیشتر تصویر دقیق‌تری از توزیع ارائه می‌دهند. با این حال، بین‌های زیاد باعث می‌شوند که هیستوگرام شما ناهموار و نامشخص به نظر برسد.

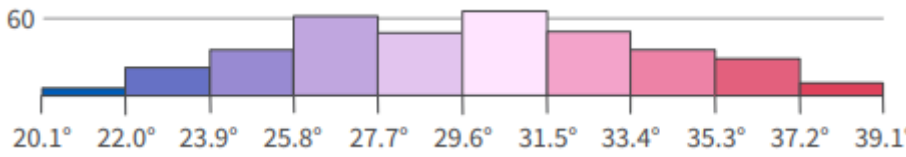


شکل ۴-۱۶: هیستوگرام دمای بیشینه برای شهرهای مختلف در سال ۲۰۱۸. اگر از دماهای ملایم خوشتان نمی‌آید و از دماهای بسیار بالا یا پایین لذت می برید، مینیاپولیس شهری عالی است، البته آنجا یا در حال سوختن هستید یا در حال یخ‌زدن!

هیستوگرام تجمعی: روش دیگری برای ساخت هیستوگرام وجود دارد. به جای اینکه ارتفاع هر بین فقط تعداد نقاط داده را نشان دهد، می‌توانیم تعداد بین‌های قبلی را نیز اضافه کنیم. بنابراین، بین‌ها با حرکت به سمت راست بلندتر می‌شوند. در شکل ۴-۱۷ نحوه‌ی ساخت یک **هیستوگرام تجمعی**[۱] از یک هیستوگرام معمولی آمده است.

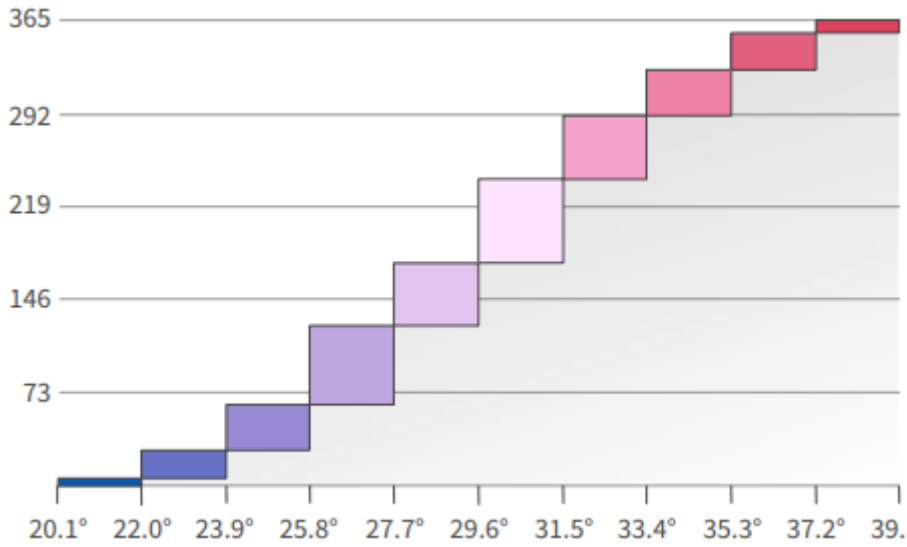
توجه داشته باشید بینی که به ۳۳.۴ درجه سانتیگراد ختم می‌شود دارای ارتفاع ۲۹۲ است. این به ما می‌گوید که ۲۹۲ روز یا ۸۰ درصد سال خنک‌تر از ۳۳.۴ درجه سانتیگراد بوده است. بیایید اکنون سه شهر خود را با هیستوگرام‌های تجمعی دارای ۱۰ بین مقایسه کنیم (شکل ۴-۱۸).

هنگام مشاهده‌ی هر نمودار، همیشه مقیاس‌های آن را بررسی کنید. دما در مینیاپولیس و ریودوژانیرو ممکن است در شکل ۴-۱۸ مشابه به نظر برسد؛ اما این نمودارها دارای مقیاس‌های بسیار متفاوت در محور افقی هستند. اگر یک هیستوگرام را بر روی هیستوگرام دیگری رسم کنیم، تفاوت بین شهرها آشکار می‌شود (شکل ۴-۱۹).

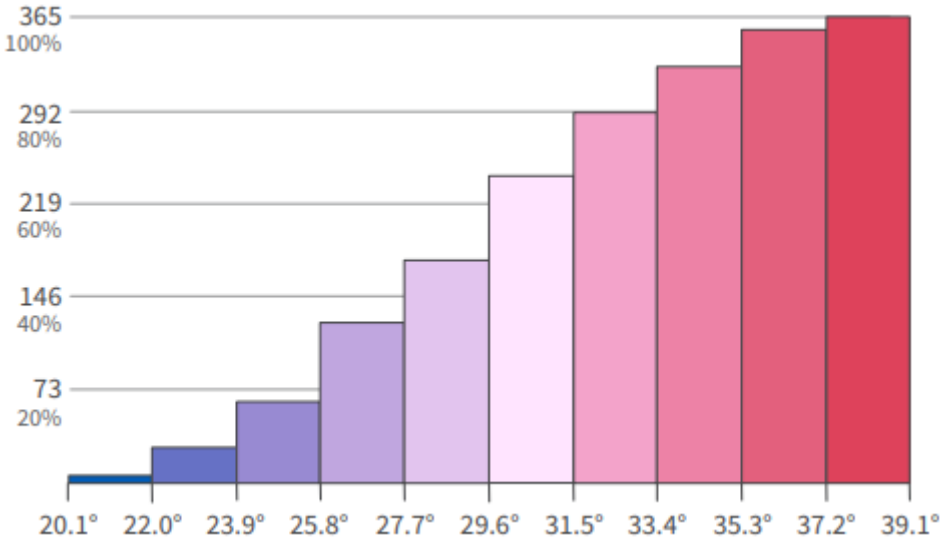


الف) کوچک کردن مقیاس شکل ۴-۱۴ج.

[۱] Cumulative Histogram

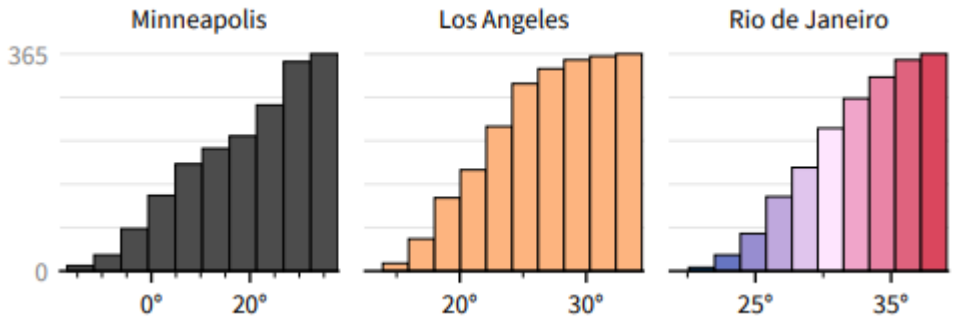


ب) قرار دادن هر بین بر روی بین قبلی.

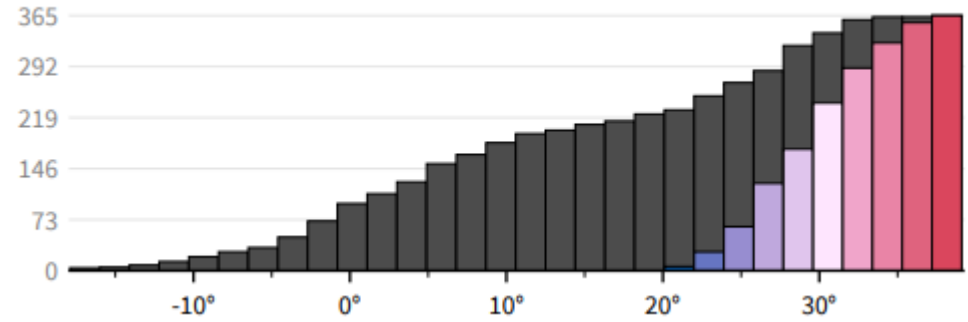


ج) رنگ‌آمیزی بین‌های تجمعی.

شکل ۴-۱۷: ساخت یک هیستوگرام تجمعی.



شکل ۴-۱۸: هیستوگرام‌های تجمعی دمای بیشینه (۲۰۱۸).



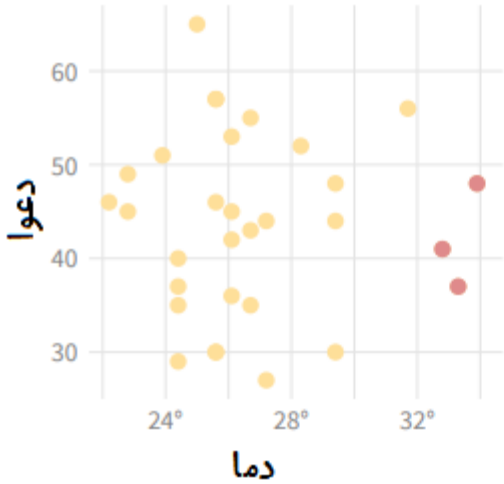
شکل ۴-۱۹: مقایسه‌ی هیستوگرام تجمعی دمای بیشینه‌ی مینیاپولیس (خاکستری) و ریودوژانیرو (رنگی). بین‌ها به طور کامل در فواصل دمایی یکسان برای هر دو شهر تعریف شده‌اند. از آنجایی که محدوده‌ها مطابقت ندارند، ریودوژانیرو چند بین خالی دارد!

نمودارهای نقطه‌ای

تاکنون، نمودارهایی را دیده‌ایم که یک متغیر یا ستون داده را نشان می‌دهند. **نمودار نقطه‌ای**[۱] چگونگی ارتباط دو متغیر با یکدیگر را نشان می‌دهد. در این نمودار، هر رکورد داده یک نقطه است. موقعیت یک نقطه در محور عمودی نشان‌دهنده‌ی مقدار رکورد در یک ستون است. موقعیت نقطه در محور افقی نشان‌دهنده‌ی مقدار رکورد در ستون دیگر است. شکل ۴-۲۰ نحوه‌ی ساخت نمودار نقطه‌ای از نقاط داده‌ی تعریف شده در شکل ۴-۱۱ را نشان می‌دهد.

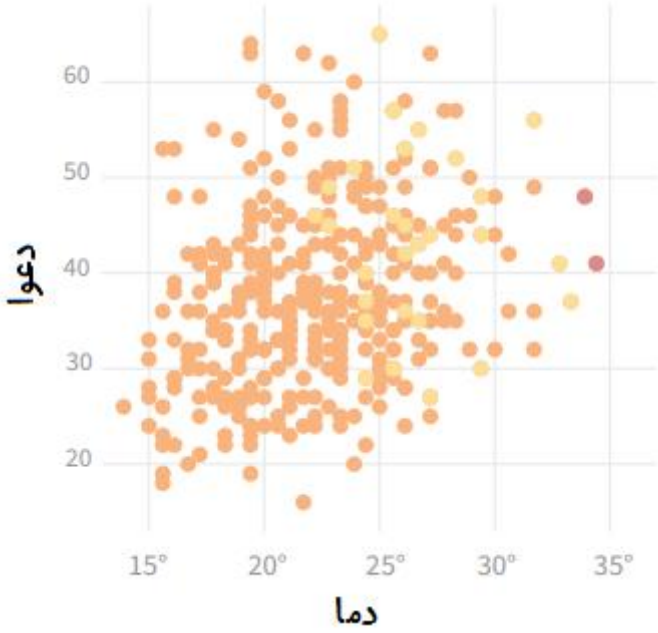
---

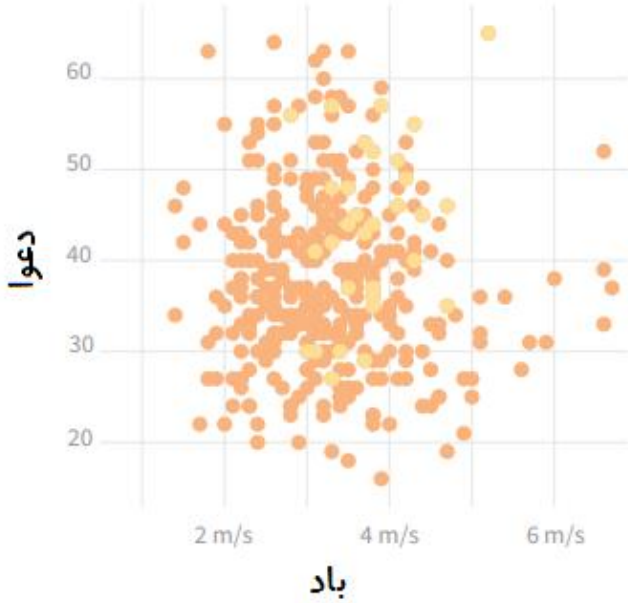
[۱] Scatter Plot



Date	Temp.	Assaults
Jul 05	26.7°	55
Jul 06	33.3°	37
Jul 07	33.9°	48
Jul 08	31.7°	56
Jul 09	32.8°	41
	⋮	

شکل ۴-۲۰: تعداد دعواهای ثبت‌شدهر لس‌آنجلس در مقایسه با دمای بیشینه‌ی روزانه در ژوئیه‌ی ۲۰۱۸. دماهای پرت با رنگ قرمز مشخص شده‌اند.

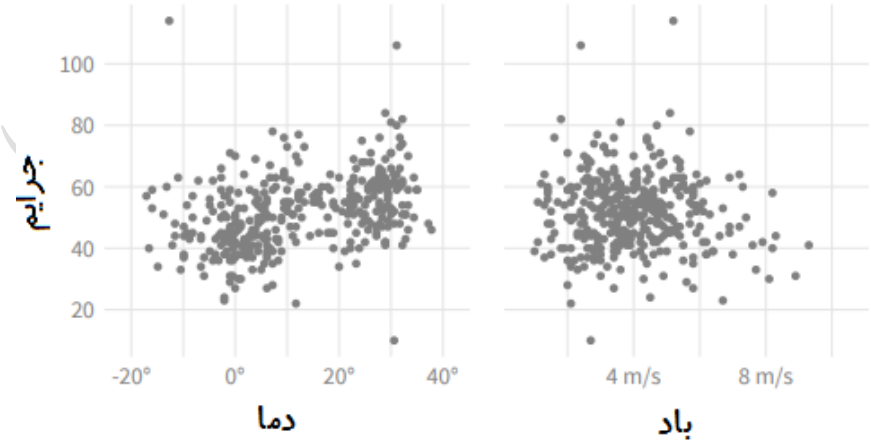




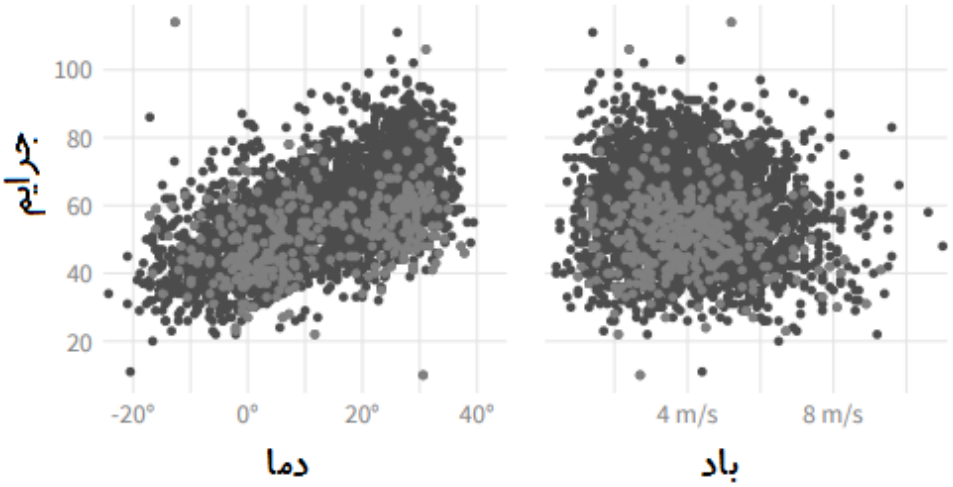
شکل ۴-۲۱: تعداد دعواها در مقایسه با بیشنه‌دمای روزانه، و میانگین سرعت باد برای بقیه‌ی سال ۲۰۱۸.

بر اساس ماتریس همبستگی جدول ۴-۷، می‌دانیم که تعداد دعواها و میزان باد با هم مرتبط نیستند، اما بین تعداد دعواها و دما همبستگی ۰.۲۸ وجود دارد. آیا می‌توانید بر اساس نمودار بگویید که روزهای گرم‌تر دعواهای بیشتری را به دنبال دارد؟

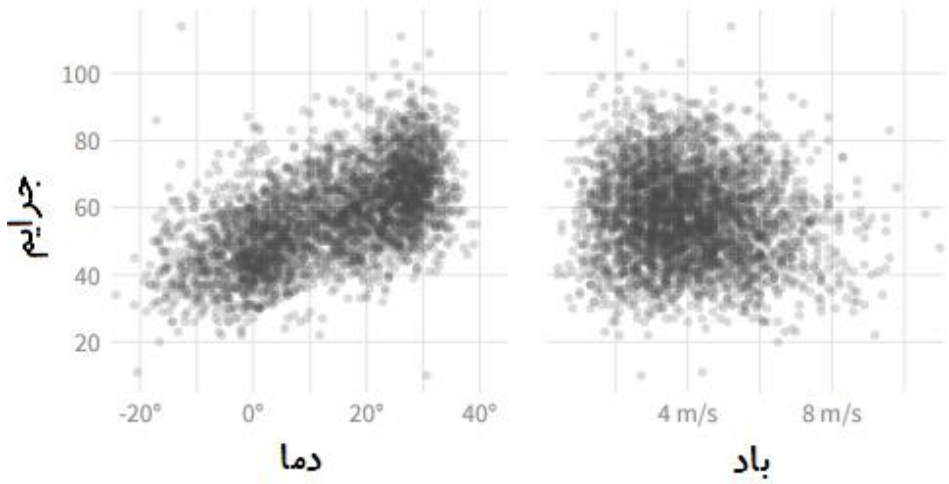
اگر داده‌هایی داشته باشیم که دامنه‌ی بزرگ‌تری از دما را پوشش دهند، تشخیص همبستگی با دما آسان‌تر خواهد بود. بیایید تعداد جرایم جنایی ثبت‌شده به صورت روزانه در مینیاپولیس را بررسی کنیم:



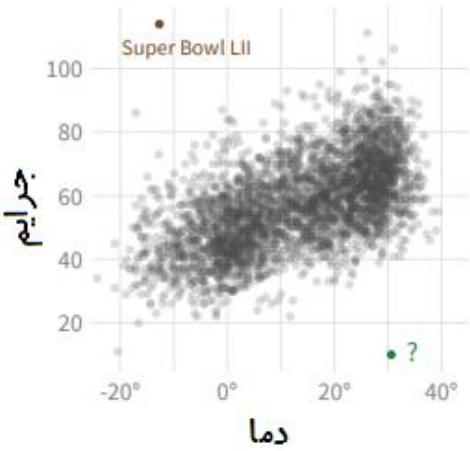
شکل ۴-۲۲: تعداد جرایم در مقایسه با دمای بیشینه‌ی روزانه و میانگین سرعت باد در مینیاپولیس برای هر روز از سال ۲۰۱۸.



شکل ۴-۲۳: تعداد جرایم ثبت‌شده در مینیاپولیس از سال ۲۰۱۰ تا ۲۰۱۸. هر نمدار بیش از ۳۰۰۰ نقطه دارد، بنابراین بسیاری از آن‌ها با هم همپوشانی دارند و تفسیر آن دشوار است.



شکل ۴-۲۴: تعداد جرایم ثبت‌شده در مینیاپولیس از سال ۲۰۱۰ تا ۲۰۱۸ با ۲۰ درصد شفافیت. سایه‌های تیره خاکستری در نواحی با غلظت نقاط بیشتر شکل می‌گیرند. نمودار سمت چپ دارای ضریب همبستگی ۰.۵۸ است: روزهای گرم‌تر معمولاً جرایم بیشتری را به همراه دارند. هیچ الگوی آشکاری بین باد و جرم وجود ندارد، و نقاط داده‌ی نشان داده شده در سمت راست همبستگی ۰.۱۱- را نشان می‌دهند.

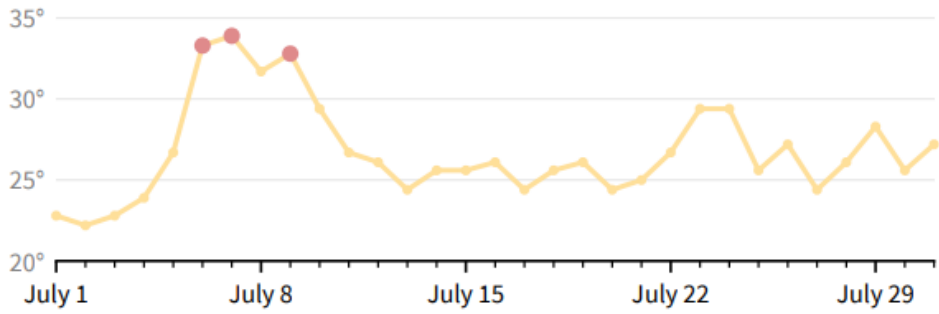


Date	Temp.	Offenses
04-Feb-18	-12.7°	114
⋮		
04-Jun-18	30.6°	10
⋮		

شکل ۴–۲۵: به داده‌های پرت توجه کنید: نقاط تنهای دور از بقیه. بررسی کنید که این نقاط با کدام ردیف‌ها مطابقت دارند. روز سرد با جرایم فراوان، ۴ فوریه ۲۰۱۸ است، زمانی که مسابقه‌ی Super Bowl در مینیاپولیس برگزار شد! روز گرم با جرایم کم هیچ توضیح واضحی ندارد، بنابراین می‌تواند مشکلی در سیستم جمع‌آوری داده‌های اداره پلیس باشد.

سری‌های زمانی

اگر همه‌ی نقاط داده‌ی شما دارای یک برچسب زمانی مرتبط هستند، می‌توانید آن‌ها را به ترتیب زمان رسم کرده و خطی را که نقاط متوالی را به هم متصل می‌کند، ردیابی کنید. به این نوع نمودار سری زمانی[۱] می‌گویند. این نمودار می‌تواند روابط و الگوهای جالبی را بین داده‌های شما و زمان آشکار کند. در اینجا یک سری زمانی از اوج دمای روزانه در لس‌آنجلس در ژوئیه‌ی ۲۰۱۸ با استفاده از داده‌های مشابه شکل ۴–۱۱ آورده شده است:

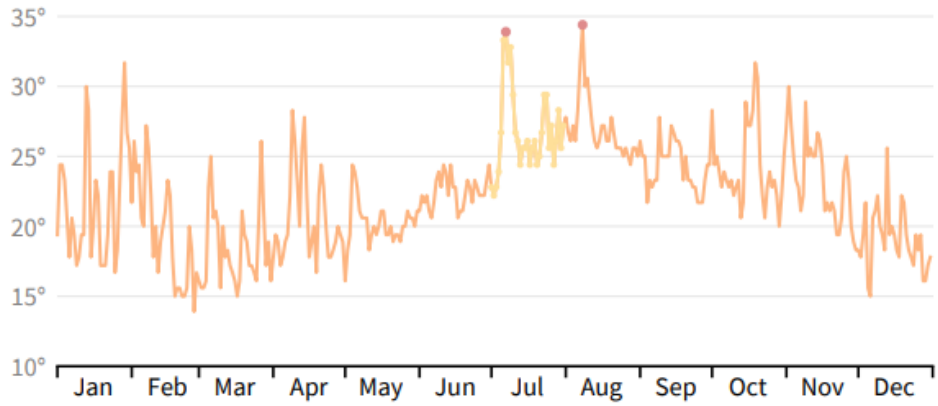


شکل ۴–۲۶: اوج دمای روزانه در لس‌آنجلس در ژوئیه‌ی ۲۰۱۸.

---

[۱] Time Series

هنگامی که یک سری زمانی نقاط زیادی دارد که خیلی نزدیک به هم قرار گرفته‌اند، تغییرات بین نقاط متوالی می‌تواند باعث ایجاد لرزش زیادی در طرح شود. مشاهده کنید چه اتفاقی می‌افتد اگر نمودار یکسانی با استفاده از داده‌های مشابه شکل ۴-۱۲ برای تمام روزهای سال ایجاد کنید:



شکل ۴-۲۷: اوج دمای روزانه در لس‌آنجلس در تمام ماه‌های سال ۲۰۱۸.

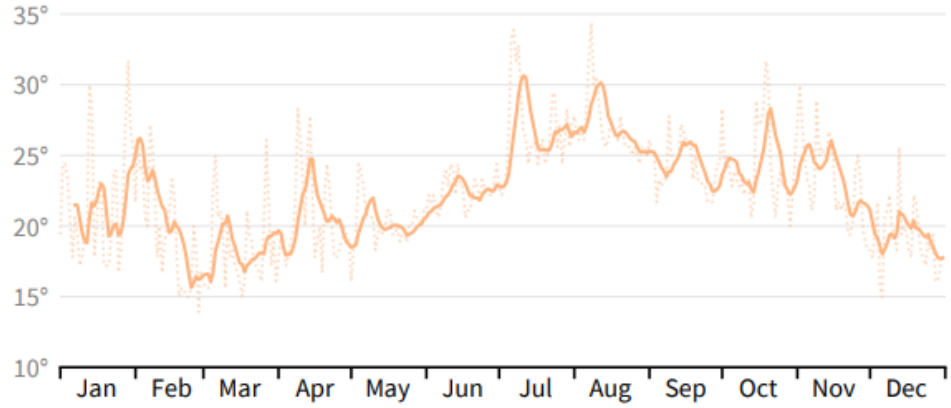
در چنین مواقعی می‌توانیم طرح نمودار را هموار کنیم تا بررسی آن آسان‌تر شود. یک تکنیک رایج **میانگین متحرک**[۱] نامیده می‌شود: هر نقطه‌ی داده با میانگین خود و چند نقطه‌ی قبلی جایگزین می‌شود. تعداد نقاط قبلی استفاده شده **اندازه‌ی پنجره**[۲] نامیده می‌شود. بیایید ببینیم این فرایند چه نتیجه‌ای دارد (شکل ۴-۲۸).

ما قبلاً بر اساس نمودارهای جعبه (شکل ۴-۱۳) و هیستوگرام (شکل ۴-۱۹) می‌دانستیم که اوج دما در ریودوژانیرو و مینیاپولیس توزیع‌های بسیار متفاوتی دارد. اکنون، به وضوح می‌توانیم ببینیم که از ژوئن تا آگوست، زمانی که در مینیاپولیس تابستان و در ریودوژانیرو زمستان است، تقریباً یکسان هستند! (شکل ۴-۲۹).

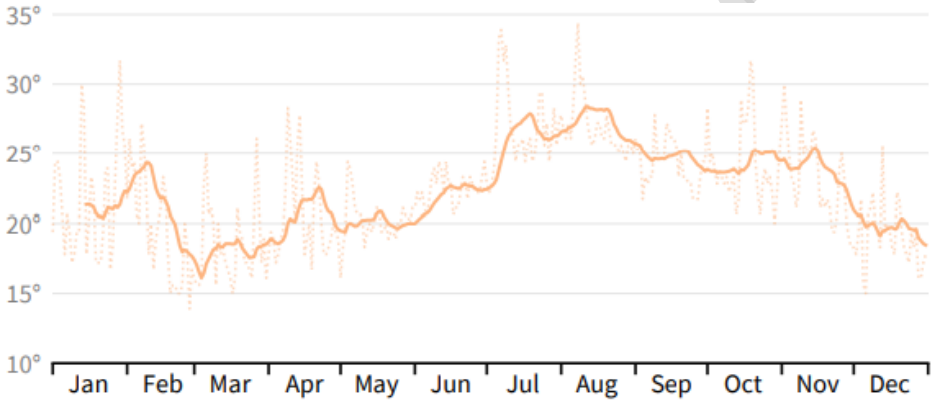
بسیاری از رویدادهای دیگر نیز تحت تأثیر زمان و شرایط زمانی سال هستند. به عنوان مثال، فروش بستنی در تابستان و بروز بیماری‌های تنفسی در زمستان بیشتر است. هنگامی که داده‌های خود را در یک سری زمانی رسم می‌کنید، تشخیص الگوهای فصلی بسیار آسان می‌شود.

---

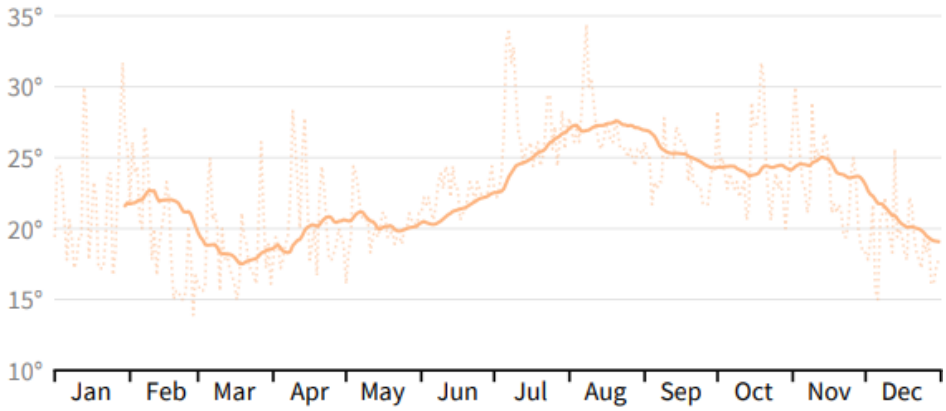
[۱] Moving Average
[۲] Window Size



الف) اندازه‌ی پنجره برابر با ۷ روز



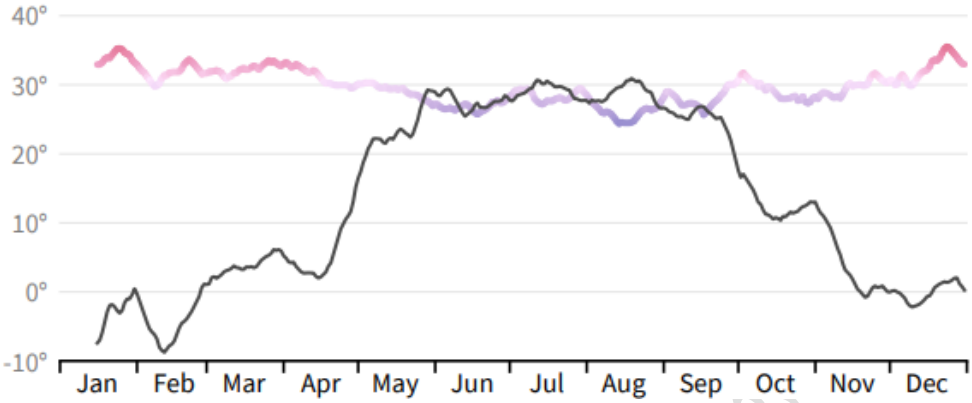
ب) اندازه‌ی پنجره برابر با ۱۵ روز



ج) اندازه‌ی پنجره برابر با ۳۰ روز

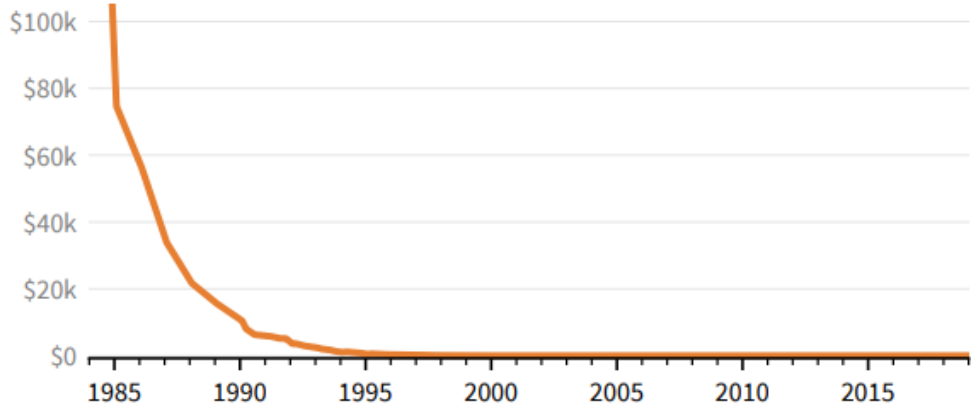
شکل ۴-۲۸: میانگین متحرک اوج دمای روزانه در سال ۲۰۱۸ در لس‌آنجلس با اندازه‌های مختلف پنجره.

افزایش اندازه‌ی پنجره میانگین متحرک سریهای زمانی ما را هر بار هموارتر می‌کند.



شکل ۴-۲۹: مقایسه میانگین‌های متحرک ۱۵ روزه‌ی سری‌های زمانی اوج دمای مینیاپولیس (خاکستری) و ریودوژانیرو (رنگی).

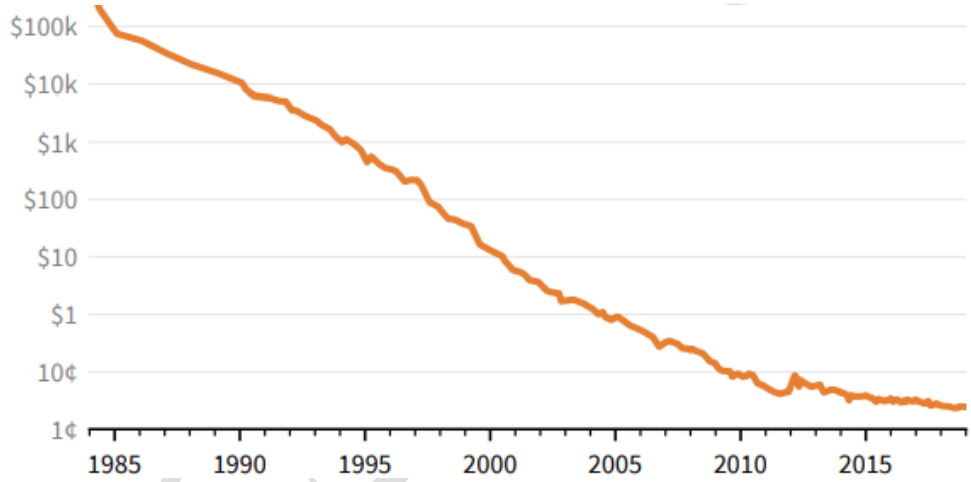
به طور کلی، نمودارهای زمانی هنگامی مفید هستند که نیاز به درک چگونگی بزرگ یا کوچک شدن یک متغیر در طول زمان داشته باشیم. به عنوان مثال، قیمت ذخیره‌سازی داده‌های دیجیتال کاهش یافته است و به انقلاب علم داده دامن می‌زند. برای درک کامل این تغییر که چقدر چشمگیر بوده است، بیایید هزینه‌ی ذخیره سازی داده‌ها را در طول زمان ترسیم کنیم:



شکل ۴-۳۰: هزینه‌ی تعدیل شده بر اساس تورم برای ذخیره‌سازی ۱ گیگابایت.

در سال ۱۹۸۵، ذخیره‌سازی یک گیگابایت داده بیش از صدهزار دلار (معادل دلار ۲۰۱۹) هزینه داشت. در سال ۲۰۱۸، قیمت آن کمتر از سه سنت بود. از آنجایی که مقیاس این قیمت‌ها بسیار از هم فاصله دارند، خط نمودار پس از سال ۱۹۹۵ صاف شده و تشخیص این‌که از آن نقطه به بعد چه اتفاقی می‌افتد دشوار است.

مقیاس‌های لگاریتمی: یک ترفند مفید برای مشاهده داده‌ها در چنین وضعیت‌هایی وجود دارد: **مقیاس خطی**[۱] رایج را با **مقیاس لگاریتمی**[۲] عوض کنید. یک مقیاس خطی به گونه‌ای تعریف می‌شود که هر تیک[۳] برابر با عدد قبلی به اضافه‌ی یک مقدار ثابت باشد. در شکل ۴-۳۰ در صفحه‌ی قبل، هر تیک در محور عمودی ۲۰ هزار دلار بیشتر از تیک قبلی بود. از سوی دیگر، یک مقیاس لگاریتمی به گونه‌ای تعریف می‌شود که هر تیک برابر با حاصل‌ضرب ثابتی از تیک قبلی باشد. اغلب عدد ۱۰ برای محاسبه‌ی حاصل‌ضرب انتخاب می شود، زیرا به خوبی منجر به گرد شدن اعداد تیک‌ها می‌شود:



شکل ۴-۳۱: هزینه‌ی تعدیل شده بر اساس تورم برای ذخیره‌سازی ۱ گیگابایت. (مقیاس لگاریتمی)

مشاهده کنید که قیمت‌های ذخیره‌سازی از سال ۱۹۹۵ تا ۲۰۱۸ دیگر شبیه یک خط صاف نیستند. اکنون می‌توان به طور دقیق قیمت‌های ذخیره‌سازی هر سال را خواند. از سال ۱۹۸۵ تا ۲۰۱۰، قیمت هر گیگابایت تقریباً هر ۵ سال ده برابر ارزان‌تر شده است، به جز بین سال‌های ۱۹۹۵ تا ۲۰۰۰، که قیمت‌ها تقریباً ۱۰۰ برابر کاهش یافتند.!

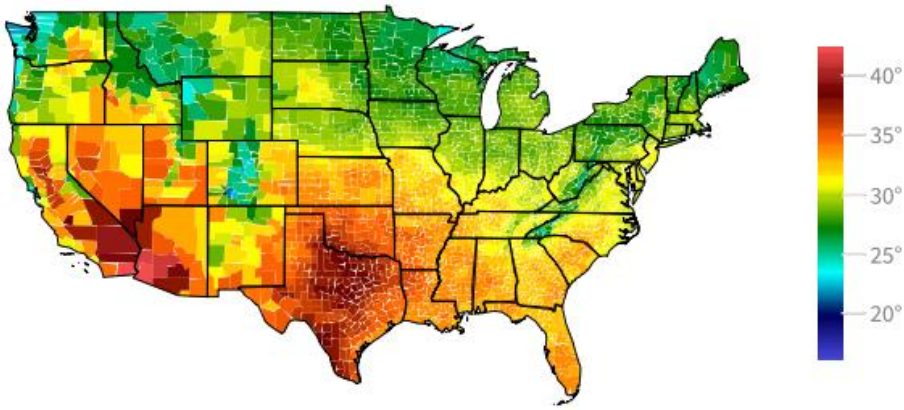
در سال ۲۰۱۱، سیل‌های وحشتناکی تایلند را درنوردید که جان ۸۱۵ نفر را گرفتند و میلیون‌ها نفر دیگر را تحت تاثیر قرار دادند. بسیاری از کارخانه‌ها به شدت آسیب دیدند و زنجیره‌ی تامین جهانی هارد دیسک در سال ۲۰۱۲ را مختل کرد. این رویداد غم انگیز افزایش کاملاً قابل مشاهده‌ی قیمت‌ها در نمودار ۴-۳۱ را توضیح می‌دهد.

---

۱ Linear Scale
۲ Logarithmic Scale
۳ Tick

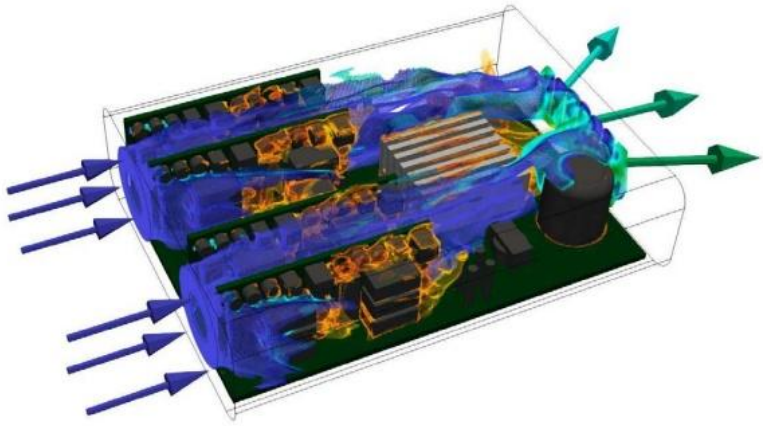
نقشه‌ها

وقتی داده‌های جغرافیایی دارید، آن‌ها را بر روی نقشه نشان دهید. برای مثال، اگر اوج دمای روزانه هر شهر را در ایالات متحده آمریکا می‌دانید، می‌توانید یک نقشه‌ی حرارتی ایجاد کنید:



شکل ۴-۳۲: میانگین اوج دمای روزانه در ژوئیه‌ی ۲۰۱۸، مناطق خنک اطراف رشته کوه‌های راکی و آپالاش را نشان می دهند.

ایجاد نقشه‌ها به داده‌های جغرافیایی محدود نمی‌شود. به عنوان مثال، هنگامی که مهندسان آیرودینامیک مقادیر دما را برای هر نقطه از فضای جریان هوا به دست می‌آورند، آن را در سه بعد ترسیم می‌کنند:



شکل ۴-۳۳: باد تولید شده توسط فن، برد مدار الکترونیکی را خنک می‌کند. هوای دریچه (سمت راست) کمی گرمتر از هوای هواکش‌ها (سمت چپ) است.

۴-۵- آزمایش کردن

در حین بررسی داده‌ها و شناسایی الگوها، اغلب نظریه‌هایی را برای توضیح آنچه می‌بینیم ابداع می‌کنیم که بسیاری از آن‌ها گمراه‌کننده یا اشتباه هستند. همانطور که کارل سِیگَن[۱] کیهان‌شناس معروف در آخرین مصاحبه خود گفته است:

«علم چیزی بیش از مجموعه‌ای از دانش‌ها است. علم یک طرز فکر است؛ راهی برای بررسی شکاکانه‌ای از جهان با درک دقیق خطاپذیری انسان.»

زیر سوال بردن مفروضات خود می‌تواند عمیقاً غیرطبیعی به نظر برسد، و متعاقباً، توضیح قطعی الگوها نیز چالش برانگیز است. خوشبختانه، دانشمندان برای شک و تردید ارزش قائل شده و ابزارهایی برای کمک به آن به وجود آورده‌اند.

فرضیه‌ها

برای پرداختن به یک سوال یا درکی شهودی، بدون نتیجه‌گیری نادرست، با قاب‌بندی مناسب آن شروع کنید. آن را به عنوان یک **فرضیه**[۲] بیان کنید: عبارتی که می‌تواند توسط داده‌هایی که قابل جمع‌آوری هستد، تایید یا رد شود. شما باید فرضیه را آزمایش کنید، بنابراین هنوز درست یا غلط بودن آن را قطعی نکنید. فرضیه‌ی خود را ساده و عینی مطرح کنید تا آزمایش آن آسان‌تر باشد.

آزمایش ذائقه: قهوه‌خانه‌ی شما در مینیاپولیس در حال از دست دادن مشتریان گیاهخوارش است. ده روز پیش تصمیم گرفتید منو را به روز کنید و چای سبز و بستنی شیر نارگیل ارائه دهید. شما این کار را بدون هیچ تحقیقی و بر اساس شهود زیر انجام داده‌اید:


مشتریان گیاهخوار در روزهایی که یوگا کار می‌کنند چای سبز می‌نوشند،


مشتریان بستنی‌های منوی قدیمی را دوست ندارند.

حال که منو آماده شده و مورد استفاده قرار گرفته است، می‌خواهید تحقیقی انجام دهید تا ببیند آیا تغییر منو تصمیم خوبی بوده است یا خیر. آیا می‌توانید هر یک از ادراکات شهودی خود را با یک فرضیه بیان کنید؟ سعی کنید موردی را که مرتبط با اهداف شما و به سادگی قابل آزمایش کردن است، فرموله کنید.

---

[۱] Carl Sagan (1934-1996): کیهان‌شناس آمریکایی- اوکراینی و از بنیان‌گذاران جستجوی هوش فرازمینی. م.

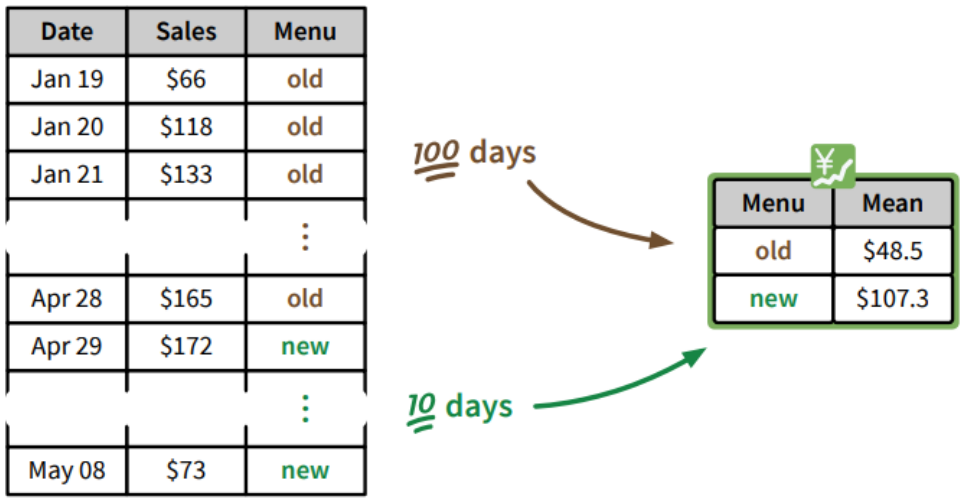
[۲] Hypothesis

در «شهود اول»، یافتن معیاری عینی برای ذائقه بسیار دشوار است. با این حال، یکی از اهداف کسب و کار شما باید فروش محصولاتتان باشد، بنابراین می‌توانید این فرضیه را به صورت زیر بیان کنید: *مشتریان گیاهخوار در روزهای تمرین یوگا چای سبز* ***می‌خرند***. آزمایش این فرضیه می‌تواند به شما کمک کند تا متوجه شوید که آیا تصمیمتان توجه مشتریان گیاهخوار را جلب می‌کند یا خیر. با این حال، این کار مستلزم کمی تلاش است: شما باید یک نظرسنجی با سوالات شخصی انجام دهید تا بفهمید چه مشتریانی گیاهخوار و اهل ورزش یوگا هستند.

آزمایش «شهود دوم» دشوار است، زیرا نظرسنجی از مشتریان قدیمی و وفادار می‌تواند شما را در معرض سوگیری انتخاب قرار دهد: شما فقط می‌توانید از کسانی که بازگشته‌اند و به احتمال زیاد از منوی قدیمی لذت برده‌اند، نظرسنجی کنید. با این حال، این شهود ارتباط نزدیکی با هدف تجاری فروش محصولات دارد، بنابراین یک فرضیه می‌تواند به صورت زیر باشد:

فروش بستنی‌ها با منوی جدید بیشتر است.

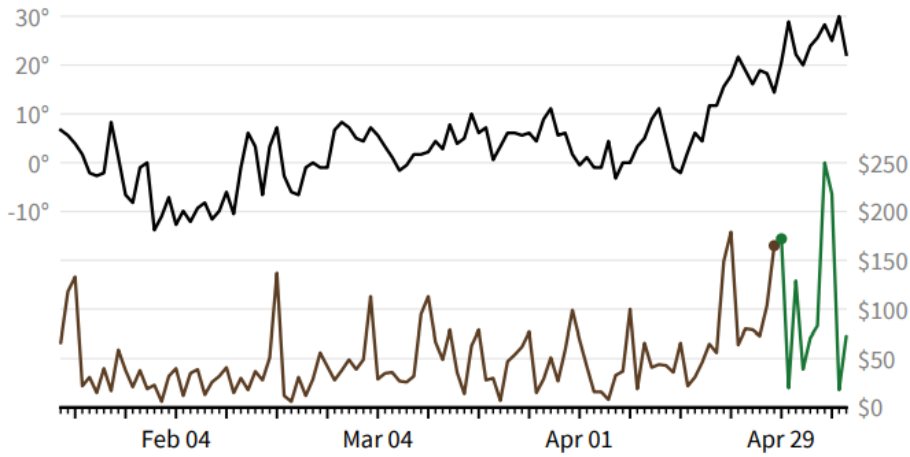
این فرضیه به شما بینش زیادی در مورد اینکه *چرا* به اهداف مرتبط با فروش بستنی خود می‌رسید یا خیر نمی‌دهد، اما حداقل به راحتی قابل آزمایش است: سیستم حسابداری شما احتمالاً داده‌های فروش را قبل و بعد از تغییر منو جمع‌آوری کرده است. شما تمام داده‌های مورد نیاز خود را از قبل دارید! بیایید ۱۰۰ روز آخر منوی قدیمی را با ۱۰ روز اول منوی جدید با استفاده از میانگین خلاصه و مقایسه کنیم:



شکل ۴-۳۴: خلاصه‌سازی فروش روزانه با هدف آزمایش فرضیه‌ی مورد نظر.

این داده ها با فرضیه موافق هستند: از زمان استفاده از منوی جدید فروش بالاتر رفته است. آیا این شواهد برای تأیید فرضیه‌ی ما که منوی جدید فروش بستنی را افزایش می‌دهد کافی است؟

خیر. با مقایسه‌ی میانگین‌ها در این مجموعه‌داده‌ی کوچک، هیچ نتیجه‌ای نمی توان گرفت. بسیاری از عوامل دیگر ممکن است بر داده‌ها تأثیر گذاشته باشند. تغییرات در رفتار مشتری می‌تواند ناشی از تغییرات دیگری باشد که در قهوه‌خانه شما ایجاد شده است، مانند چیدمان میزها. این تغییرات می‌توانند ناشی از عوامل خارج از کنترل شما باشند، مانند آب و هوا:



شکل ۴-۳۵: اوج دمای روزانه و فروش بستنی در قهوه‌خانه‌ی شما در مینیاپولیس. دما در بالا (مقیاس سمت چپ)، فروش بستنی در پایین (مقیاس سمت راست) نشان داده شده است.

صرف نظر از منوی غذایی شما، مشتریان مطمئناً در یک روز گرم بیشتر بستنی می‌خورند تا در هنگام کولاک! شاید دمای بالاتر باعث افزایش فروش شده است نه منوی جدید. شاید خوش شانس بوده‌اید. اگر نمی‌توانید مطمئن شوید که چه چیزی باعث تغییر در فروش شده است، چگونه می‌توانید بفهمید که آیا نتیجه‌ی آزمون فرضیه حقیقت را منعکس می‌کند؟ در بخش‌های بعدی، روش‌هایی را که دانشمندان برای پرداختن به این موضوع با دقت توسعه داده‌اند، بررسی خواهیم کرد.

آزمایش‌ها

محققان خوب روش‌هایی را برای کاهش عدم قطعیت در مورد نتایج خود توسعه می‌دهند. یک روش علمی برای آزمودن یک فرضیه، **آزمایش**[۱] نامیده می‌شود. فرضیه ها را فقط می‌توان از طریق آزمایش‌های خوب طراحی شده به درستی آزمایش کرد.

---

[۱] Experiment

اولین مرحله‌ی طراحی آزمایش، شناسایی متغیرهایی است که باید اندازه‌گیری و داده‌هایی که باید متعاقباً جمع‌آوری کنید. برای «فرضیه‌ی مثال قبل»، این متغیرها *فروش* و *منو* هستند. ما قبلاً آن‌ها را به عنوان ستونی در شکل ۴-۳۴ در اختیار داشته‌ایم.

همیشه متغیرهای دیگری نیز وجود دارند که بر چیزهایی که می‌خواهید بررسی کنید تأثیر می‌گذارند. ما آن‌ها را **متغیرهای خارجی**[۱] می‌نامیم و باید در مورد آن‌ها مراقب باشیم: آن‌ها به طور بالقوه می‌توانند شما را به سمت نتیجه‌گیری‌های اشتباه سوق دهند. اگر داده‌های قدیمی شما شامل این متغیرها نمی‌شوند، سعی کنید داده‌های مرتبط با متغیرهای خارجی را نیز جمع‌آوری کنید.

اگر شکل ۴-۳۵ را با دقت مشاهده کنید، متوجه خواهید شد که اوج میزان فروش در زمانی که هوا گرم‌تر است، بالاتر به نظر می‌رسد.میزان فروش نسبتاً نامنظم است، اما ما می‌توانیم هر هفت روز یک بار اوج میزان فروش را مشاهده کنیم. شاید مردم در تعطیلات آخر هفته بیشتر بستنی بخورند! تا کنون دو متغیر خارجی را شناسایی کرده‌ایم: *دما* و *روز هفته*.

هنگامی که در حال کاوش در داده‌ها هستید، به جستجوی برای یافتن سرنخ‌هایی به‌منظور شناسایی متغیرهای خارجی مختلف ادامه دهید. هنگامی که آن‌ها را پیدا کردید، مطمئن شوید که داده‌های شما بر اساس آن‌ها به روز می‌شوند:

Date	Sales	Menu	Day	Temp.
Jan 19	$66	old	Fri	6.7°
Jan 20	$118	old	Sat	5.6°
Jan 21	$133	old	Sun	3.9°
⋮				
Apr 28	$165	old	Sat	14.4°
Apr 29	$172	new	Sun	20.6°
⋮				
May 08	$73	new	Tue	22.2°

شکل ۴-۳۶: جمع‌آوری داده‌ها در مورد عوامل تاثیرگذار بر میزان فروش.

---

[۱] Extraneous Variables

در یک آزمایش خوب طراحی‌شده، فرضیه با استفاده از رکوردهایی که در آن واریانس متغیرهای خارجی به حداقل می‌رسد، آزمایش می‌شود. با این حال، این کار ممکن است به شدت تعداد رکوردهای قابل استفاده را محدود کند. به عنوان مثال، اگر فقط یک روز از هفته را با دمای اوج مشابه در نظر بگیریم، تنها دو رکورد داده باقی می‌مانند:

Date	Sales	Menu	Day	Temp
Apr 22	$179	old	Sun	17.8°
Apr 29	$172	new	Sun	20.6°

شکل ۴-۳۷: مقایسه‌ی میزان فروش در شرایط مشابه.

میزان فروش با منوی جدید در این آزمایش کمتر بوده است. داده‌ها از «فرضیه» پشتیبانی نمی‌کنند، بنابراین ما کمی به رد آن نزدیک‌تر می‌شویم. اما آیا می‌توانیم آن را رد کنیم؟

خیر. حتی هنگام طراحی یک آزمایش در یک محیط کنترل شده نیز ما هرگز نمی‌توانیم همه چیز را کنترل کنیم. همیشه عوامل دیگری بر داده‌ها تأثیر می گذارند که شناسایی برخی از آن‌ها دشوار است. آیا می‌دانستید که ممکن است میزان فروش شما هم بر اساس بازدید تصادفی مشتریان گروهی از آن منطقه و هم‌چنین به دلیل افزایش غیرعادی تعداد خانه‌هایی که یخچال‌های خراب دارند، افزایش یابد؟ برعکس، آیا می‌دانستید که ممکن است افزایش غیرمنتظره‌ی آلرژی افراد به گرده‌ی گل‌ها، مردم را در خانه نگه داشته و فروش بالقوه‌ی شما را کاهش دهد؟ دنیا برای شما پیچیده‌تر از آن است که فارغ از میزان تلاشتان بتوانید همه‌ی متغیرهای خارجی را شناسایی کنید.

نویز: باید توجه داشته باشید که پیچیدگی جهان ما همیشه نوسانات تصادفی را مستقل از هر متغیر خارجی که بتوانیم پیدا کنیم، به همراه خواهد داشت. چنین نوسانات تصادفی غیرقابل توضیحی به صورت عامیانه **نویز**[1] نامیده می‌شوند. هنگامی که مقدار داده‌ی بسیار کمی در دسترس است، دشوار است بدانید کدام نوسانات نویز هستند و کدام نویز نیستند، و حقیقت در هاله‌ای از ابهام فرومی‌رود. در نهایت، هر چه داده‌های بیشتری داشته باشیم که یک فرضیه را پشتیبانی می‌کنند، به تأیید آن نزدیک‌تر می‌شویم.

---

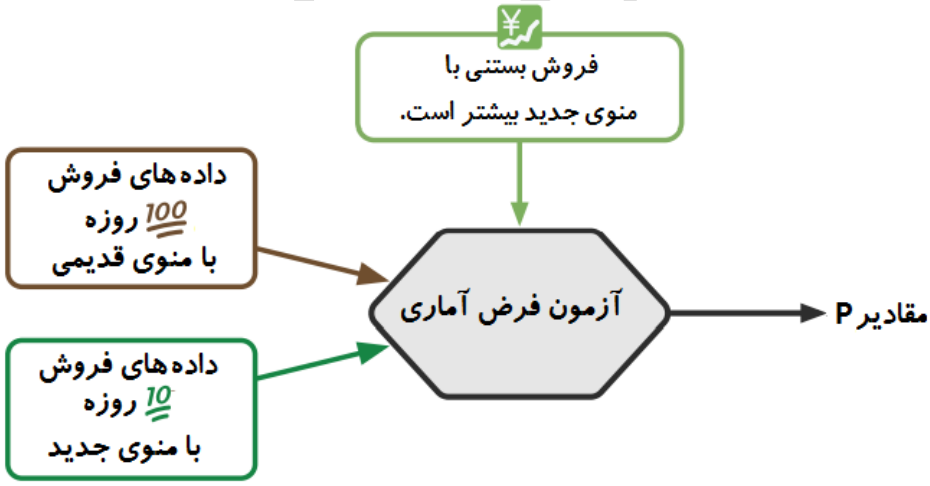
[1] Noise

برعکس، هر چه داده‌های بیشتری داشته باشیم که با فرضیه مخالف باشند، به رد آن نزدیک‌تر می‌شویم. متأسفانه در شکل ۴–۳۷ تنها دو نقطه‌ی داده وجود دارد و این برای تأیید یا رد فرضیه کافی نیست. ما به داده‌های بیشتری نیاز داریم.

از آنجایی که زیاد بودن داده‌ها عدم قطعیت را کاهش می‌دهد، ممکن است فرد وسوسه شود که برای همیشه داده‌های بیشتری را جمع آوری کند. بیایید سعی کنیم بفهمیم چه مقدار داده برای اطمینان از اینکه نتیجه‌ی آزمون فرضیه منعکس‌کننده‌ی حقیقت است، کافی است.

مقادیر P

از آنجایی که همیشه ممکن است بدشانس باشید و داده‌هایی به دست آورید که از یک فرضیه‌ی نادرست پشتیبانی می‌کنند، متخصصان آمار روش‌هایی را برای *تخمین احتمال* وقوع این اتفاق توسعه داده‌اند. به این نوع روش، **آزمون فرض آماری**[۱] می‌گویند:



شکل ۴–۳۸: استفاده از آزمون فرض آماری.

احتمال بدشانس بودن ما مقدار p[۲] (یا مقدار احتمال) نامیده می‌شود. هر چه مقدار p کوچکتر باشد، به تایید فرضیه نزدیکتر می‌شویم. اگر فرضیه درست باشد، جمع آوری داده‌های بیشتر باعث کاهش مقدار p خواهد شد.

---

۱ Statistical Hypothesis Test
۲ P-Value

انواع مختلفی از آزمون‌های فرض وجود دارد که هر کدام برای یک سناریوی خاص مناسب هستند. برای این که مقدار p معنی‌دار باشد، بررسی کنید و مطمئن شوید آزمونی را که با داده‌ها و فرضیه‌های شما مطابقت دارد، انتخاب می‌کنید[۱]. همه‌ی این آزمون‌ها به روشی مشابه کار می‌کنند: شما داده‌های آزمایش را وارد می‌کنید، و آزمون یک مقدار p را برای فرضیه مورد نظر شما به عنوان خروجی ارائه می‌دهد.

مقدار احتمال همیشه عددی بین ۰ و ۱ است و مقدار p نیز از این قاعده مستثنی نیست. به دست آوردن p = 0.2 به این معنی است که به دلیل بدشانسی، به احتمال ۲۰٪ داده‌های شما از یک فرضیه اشتباه پشتیبانی می‌کنند. اگر بابت این ریسک ناراحت هستید، داده‌های بیشتری جمع آوری کنید. دوباره آزمون را اجرا کنید و اگر فرضیه شما درست باشد، مقدار p باید کاهش یابد. در نهایت آیا این روش برای تایید یا رد فرضیه با قطعیت کامل کافی خواهد بود؟

جواب منفی است. مقدار p هرگز دقیقاً ۰ یا ۱ نیست. به عبارت دیگر، هرگز ادعا نکنید که یک فرضیه به احتمال ۰٪ یا ۱۰۰٪ نادرست است. علاوه بر این، در تفسیر این احتمالات بسیار مراقب باشید زیرا نتیجه‌ی آزمون فرض تنها می‌تواند به اندازه‌ی خود داده‌ها قابل اعتماد باشد. اگر داده‌های شما دارای مشکل سوگیری انتخاب باشند، مقدار p حاصل بی‌ربط خواهد بود. آزمون‌های فرض جایگزین آزمایش‌های کنترل‌شده نمی‌شوند، بلکه فقط آنها را تکمیل می‌کنند.

معنی‌داری آماری: اگرچه مقدار p هرگز نمی‌تواند صفر باشد، زمانی فرا می‌رسد که باید بتوانیم نتیجه‌گیری کنیم. معمولاً، ما حداکثر مقدار p قابل قبول را برای تعیین درست بودن فرضیه تعیین می‌کنیم. این آستانه **سطح معنی‌داری**[۲] نامیده می‌شود. وقتی مقدار p کمتر آن باشد، می‌توان گفت که شواهد آماری معنی‌داری وجود دارند که فرضیه ما را تأیید می‌کنند. دانشمندان معمولاً با سطوح معنی‌داری بین ۰/۰۵ تا ۵ درصد کار می‌کنند.

نتیجه‌گیری از داده‌ها بدون معنی‌داری آماری یک اشتباه رایج است. در بسیاری از موارد، نمی‌توانیم داده‌های کافی برای تولید شواهد آماری معنی‌دار که فرضیه‌های ما را پشتیبانی می‌کنند به دست آوریم، و بنابراین باید تصمیم‌های خود را بر اساس مفروضات خاصی قرار دهیم. مهم است که محدودیت‌های دانش خود را بشناسیم و آماده باشیم تا نادرستی فرضیات‌مان ثابت شود.

---

[۱] برای مروری بر آزمون‌های فرض رایج و نحوه‌ی استفاده از هر آزمون، به http://code.energy/hypothesis-test مراجعه کنید.

[۲] Significance Level

شکل ۴-۳۹: «دوست» دریافت شده از http://xkcd.com.

بازه‌های اطمینان

اگر بتوانیم منوی جدید را برای تعداد نامحدودی روز امتحان کنیم، میانگین فروش روزانه نشان‌دهنده‌ی عملکرد واقعی آن خواهد بود. آیا می‌توانیم مطمئن باشیم که میانگین فروش روزانه‌ی برابر با ۱۰۷/۳ دلاری که در آزمایش به دست آوردیم، حقیقت دارد؟ به عبارت دیگر، با آزمایش فرضیه‌ی زیر چه مقداری برای p به دست می‌آوریم؟

NEW میانگین فروش روزانه‌ی بستنی ۱۰۷/۳ دلار است.

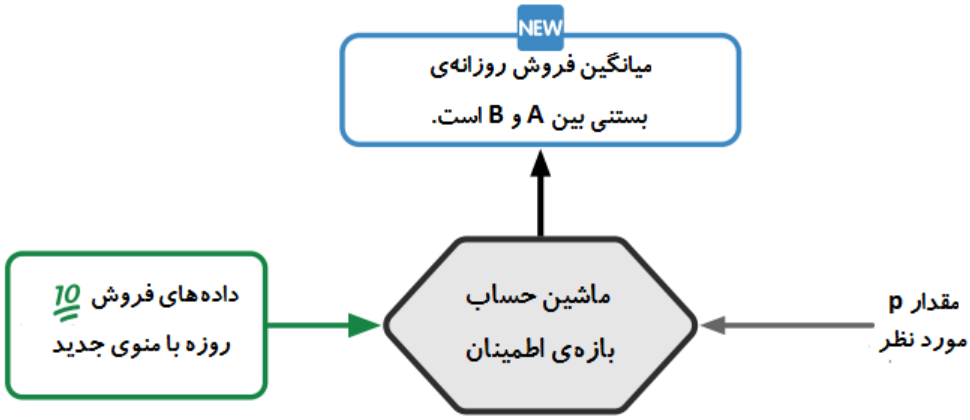
تقریباً غیرممکن است که میانگین فروش را *دقیقاً* ۱۰۷/۳ دلار به دست آوریم زیرا جمع‌آوری داده‌ها همچنان ادامه دارد، بنابراین احتمال اینکه به عدد ۱۰۷/۳ دلار نرسیم تقریباً ۱۰۰٪ است. به عبارت دیگر، مقدار p تقریباً ۱ خواهد بود، که با سطح معنی‌داری نزدیک به صفر که مد نظر ما است فاصله‌ی زیادی دارد! برای به دست آوردن مقدار p پایین‌تر، باید مقادیر نزدیک به ۱۰۷/۳ دلار را بررسی کنیم. به عنوان مثال، می‌توانیم آزمایش کنیم که آیا میانگین فروش بین A=۹۷/۳ و B=۱۰۷/۳ دلار است یا خیر. پس فرضیه‌ی ما به صورت زیر تغییر می‌کند:

NEW میانگین فروش روزانه‌ی بستنی بین ۹۷/۳ و ۱۰۷/۳ دلار است.

هرچه فاصله‌ی بین A و B را بزرگ‌تر کنیم، مقدار p کوچک‌تر می‌شود. وقتی فاصله طوری تنظیم شود که مقدار p برابر با سطح معنی‌داری مورد نظر ما باشد، بازه‌ی (A, B) را **بازه‌ی اطمینان**[۱] می‌نامیم. به طور معمول، ما داده‌ها و سطح معنی‌داری مورد نظر خود را در تابعی به نام ماشین حساب بازه‌ی اطمینان وارد می‌کنیم و این تابع بازه‌ی (A, B) را به عنوان خروجی به ما می‌دهد. بسته به ماهیت داده‌ها و فرضیه، ماشین حساب‌های مختلفی برای بازه‌ی اطمینان وجود دارد که می‌توانید از بین آن‌ها انتخاب کنید. با این حال، همه‌ی آن‌ها به روشی مشابه کار می‌کنند:

---

[۱] Confidence Interval



شکل ۴-۴۰: پیدا کردن بازه‌ی اطمینان.

ماشین‌حساب‌های بازه‌ی اطمینان اغلب به جای سطح معنی‌داری، **سطح اطمینان**[۱] را تقاضا می‌کنند. سطح اطمینان برابر با یک *منهای* سطح معنی‌داری است. به عبارت دیگر، سطح اطمینان ۹۵٪ با سطح معنی‌داری ۵٪ یکسان است و بالعکس سطح اطمینان ۹۹٪ معادل سطح معنی‌داری ۱٪ است.

نتیجه‌گیری

تحلیل داده‌ها فرآیندی است که شما باید با جدیت دنبال کنید تا به نتیجه‌گیری قانع کننده برسید. برای اطمینان از یکپارچگی نتایجتان، داده‌های خود را در هر مرحله از خط لوله هدایت کنید:

خود را به دردسر نیاندازید و هیچ یک از مراحل را نادیده نگرفته و ترتیب آن‌ها را تغییر ندهید. اگر مرحله‌ی کاوش نشان داد که داده‌های کافی برای آزمایش فرضیه‌های خود ندارید، مطمئن شوید که تمام مراحل به طور کامل و به ترتیب انجام شده‌اند: جمع‌آوری، پردازش، کاوش و تنها پس از آن آزمایش.

---

[۱] Confidence Level

جمع‌آوری: در زمان شروع، تمرکز اصلی شما جمع‌آوری و ذخیره‌ی داده‌ها از هر نوع ممکن و از هر منبع ممکن است. همان‌طور که داده‌های موجود را بررسی یا مکانیزم‌هایی برای گرفتن داده‌های جدید ایجاد می‌کنید، نگرانی اصلی شما این است که از هرگونه سوگیری انتخاب دوری کنید.

پردازش: پس از جمع‌آوری داده‌ها، درک آن‌ها را برای کامپیوتر آسان کنید. در طی این مرحله، کمال‌گرا باشید: مطمئن شوید که تمام جنبه‌های مجموعه‌داده سازماندهی شده و فقط داده‌های تمیز، معتبر و ثابت باقی مانده‌اند. علاوه بر این، شما این مسئولیت بزرگ *ناشناس‌سازی* داده‌های حساس را نیز بر عهده دارد.

کاوش: پس از پاک‌سازی، مجموعه داده خود را *خلاصه* و *مصور* کنید. گروه‌های مختلف مقادیر را مقایسه، نحوه‌ی توزیع آن‌ها در محدوده‌هایشان را مشاهده و آن‌ها را در طول زمان رسم کنید. وقتی خواص غیرطبیعی را تشخیص دادید، علل آن را بررسی کنید. چگونی انعکاس پیچیدگی‌های دنیای واقعی با استفاده از داده‌هایتان را بررسی کنید.

آزمایش: در نهایت، بایدببینید آن چه را که از کاوش به دست آورده‌اید چه ارتباطی با اهداف شما دارد؟ جداول و نمودارهایی را که می‌توانند بر تصمیم‌گیری شما تأثیر بگذارند، بررسی کنید. فرضیه‌ها را فرموله و بررسی کنید که آیا داده‌های شما آن‌ها را با معنی‌دار آماری تایید می‌کنند یا خیر.

پیروی از این اصول به شما این امکان را می‌دهد که بر اساس شواهد و نه فقط شهود تصمیم‌گیری کنید. با این حال، مراحل بهتری نیز در پیش است. با کمی تغییر در داده‌های خود، می‌توانید آن‌ها را به عنوان خوراک به الگوریتم‌هایی بدهید که اطلاعات پیچیده‌ای را در مورد آینده ارائه می‌دهند. اطلاعاتی که هوش انسانی به تنهایی هرگز نمی‌تواند آن‌ها راحدس بزند. آماده‌اید؟

منابع و مراجع

- The Data Science Design Manual, by Skiena
    - http://code.energy/skiena
- Everything is Obvious, by Watts
    - http://code.energy/watts
- Naked Statistics, by Wheelan
    - http://code.energy/wheelan

فصل ۵

یادگیری

سوال در مورد این که آیا کامپیوترها می‌توانند فکر کنند، شبیه به این سوال است که آیا زیردریایی‌ها می‌توانند شنا کنند.

– ادسخر دایکسترا [۱]

پیش‌بینی‌ها به بهبود تصمیم‌گیری کمک می‌کنند. صدها سال پیش، مایاها [۲]ی باستان به مشاهده‌ی الگوهای نجومی برای پیش‌بینی و تعیین بهترین زمان کشت ذرت می‌پرداختند. امروزه کشاورزان آمریکای شمالی به لطف پیش‌بینی پیشرفته‌ی آب و هوا می‌توانند تصمیمات دقیق‌تری بگیرند.

یکی از راه‌های پیش‌بینی آینده، نگاه به گذشته است، زیرا اغلب بهترین پیش‌بینی‌کننده‌ی نتایج آینده را می‌توان در الگوهای گذشته یافت. این امر شهودی است: اگر متوجه شدید که یک رستوران در اکثر شب‌هایی که یک بازی لیگ ملی هاکی در حال انجام است پر می‌شود، به این فکر می‌کنید که شب بازی بعدی لیگ نیز رستوران شلوغ خواهد بود.

یادگیری ماشین [۳] عبارت از استفاده از کامپیوترها برای تجزیه کردن داده‌های رویدادهای گذشته، یافتن الگوها و آزاد کردن قدرت پیش‌بینی آن‌ها است. این فناوری در همه جا وجود دارد. وقتی بیمه خودرو می‌خرید یا وام می‌گیرید، شرکت‌ها اطلاعات شما را ثبت می‌کنند. با گذشت زمان، بانک‌ها یاد می‌گیرند که پیش‌بینی کنند چه کسی احتمالاً نمی‌تواند بدهی‌های خود را پرداخت کند، و بیمه‌گران حدس می‌زنند چه کسی ممکن است با خودروی آن‌ها تصادف کند. این فصل به شما نشان می‌دهد که چگونه آن‌ها این کار را انجام می‌دهد. شما یاد خواهید گرفت که:

---

[۱] Edsger Dijkstra (1930-2002): دانشمند هلندی در علوم ریاضی و کامپیوتر و برنده‌ی جایزه‌ی تورینگ ۱۹۷۲.م.

[۲] تمدنی مربوط به گروهی از سرخ‌پوست‌ها که حدود ۱۵۰۰ سال قبل از میلاد مسیح در جنوب مکزیک زندگی می‌کردند. این قوم دست‌آوردهای چشمگیری در هنر، معماری، ستاره‌شناسی و ریاضیات داشته‌اند. هم‌اکنون نیز گروهی از بازماندگان این تمدن در مکزیک و گواتمالا زندگی می‌کنند. م.

[۳] Machine Learning

داده‌های جمع‌آوری شده را به **ویژگی‌ها**[۱] تبدیل کنید،
یک ماشین را آموزش داده و آن را **ارزیابی**[۲] کنید،
پیش‌بینی‌های آن را به صورت روشمند **اعتبارسنجی**[۳] کنید،
چرخ‌دنده‌های آن را برای نتایج بهتر **تنظیم**[۴] کنید.

بدون کامپیوترها، ما باید زمان و انرژی زیادی را صرف و برای پیش‌بینی با کارشناسان برتر مشورت کنیم. با استفاده از کامپیوترها، می‌توانیم کارها را در مقیاس بزرگ خودکارسازی کنیم. برای مثال، بانک‌ها می‌توانند وام‌ها را به‌طور خودکار تأیید و در هزینه‌های نیروی کار صرفه‌جویی و در عین حال خدمات را سریع‌تر به مشتریان ارائه کنند. در بحث مراقبت‌های بهداشتی، ماشین‌ها می‌توانند به‌طور خودکار بیماران پرخطر را برای ارائه مراقبت‌های پیشگیرانه و نجات جان‌های بیشتر غربالگری کنند.

مدل‌ها

در زمینه‌ی یادگیری ماشین به الگوریتمی که پیش‌بینی می‌کند **مدل**[۵] می‌گویند. انواع مختلفی از مدل‌ها وجود دارند که هر کدام از ترفندهای ریاضی مختلفی استفاده می‌کنند. برخی از این ترفندها کاملاً پیشرفته هستند؛ اما نگران نباشید: ما به نحوه عملکرد مدل‌ها نمی‌پردازیم. در عوض، نحوه‌ی استفاده از مدل‌ها را یاد خواهیم گرفت.

تصور کنید که شما یک مشاور املاک در شهر سویت‌واتر هستید و می خواهید قیمت سه آپارتمان را پیش‌بینی کنید. ابتدا سرنخ‌هایی مانند فاصله تا مرکز شهر، سن و منطقه‌ی آن‌ها را جمع آوری می‌کنید. می‌توانید این اطلاعات را در جدول X سازماندهی کنید، به طوری که هر ستون یک سرنخ است و به عنوان یک **ویژگی** شناخته می‌شود؛ هر ردیف نیز مربوط به فروش یک آپارتمان است. سپس می‌توانید یک ستون y_{pred} برای قیمت‌های پیش‌بینی‌شده‌ی خود اضافه کنید (شکل ۵-۱).

پر کردن ستون y_{pred} کار آسانی نیست. به خودی خود، جدول ویژگی‌های X هیچ مقدار قیمت مرجعی را ارائه نمی دهد. اگر آپارتمان ها در توکیو یا تیمبوکتو بودند، مطمئناً قیمت‌ها متفاوت بود! برای اینکه بتوانیم پیش‌بینی‌های مناسبی داشته باشیم، به مثال‌هایی نیاز داریم که از آن‌ها درس بگیریم. بیایید آپارتمان‌هایی را که قبلاً در سویت‌واتر فروخته شده‌اند را با قیمت واقعی فروششان یعنی مقدار y به جدول اضافه کنیم و این رکوردها را **مجموعه‌داده‌ی برچسب‌گذاری‌شده**[۶] بنامیم (شکل ۵-۲).

---

۱ Features
۲ Evaluate
۳ Validate
۴ Fine-Tune
۵ Model
۶ Labeled Dataset

X			y_{pred}
Distance	**Age**	**Area**	**Pred. Price**
0.4 km	4 yr	114 m^2	?
2.4 km	10 yr	68 m^2	?
3.4 km	16 yr	40 m^2	?

شکل ۵-۱: آپارتمان‌های فروشی در سویت‌واتر.

	X		y_{pred}	y
Distance	**Age**	**Area**	**Pred. Price**	**True Price**
0.4 km	4 yr	114 m^2	?	
2.1 km	8 yr	120 m^2		$840,000
10.6 km	9 yr	91 m^2		$540,000
0.5 km	3 yr	90 m^2		$745,000
3.5 km	26 yr	35 m^2		$185,000
2.4 km	10 yr	68 m^2	?	
3.4 km	16 yr	40 m^2	?	

مجموعه داده‌ی برچسب‌گذاری‌شده

شکل ۵-۲: آپارتمان‌های فروشی و آپارتمان‌های به فروش‌رفته در سویت‌واتر.

با داشتن این جدول ممکن است شروع به فکر کردن به صورت شهودی کنید. قیمت آپارتمان ۱۱۴ مترمربعی احتمالاً به ۱,۰۰۰,۰۰۰ دلار نزدیک‌تر است تا ۱۰۰,۰۰۰ دلار. اگر نمونه‌های برچسب‌دار بیشتری را جمع‌آوری کنید، می‌توانید از یک مدل پیش‌بینی‌کننده به عنوان کمک استفاده کنید.

برای شروع نیازی نیست خودتان هیچ مدلی را کدنویسی کنید، زیرا کتابخانه‌های یادگیری ماشین منبع‌باز برای اکثر زبان‌های برنامه‌نویسی در دسترس هستند. هنگامی که یک کتابخانه نصب می‌شود، می‌توانید مدل‌های آن را به کد خود وارد کرده و به سرعت مورد استفاده قرار دهید.

آموزش: قبل از اینکه بتوانید از یک مدل برای پیش‌بینی استفاده کنید، باید آن را آموزش[۱] دهید. دستورات خاص مورد نیاز برای این کار به زبان برنامه‌نویسی و کتابخانه‌ای که استفاده می‌کنید بستگی

[۱] Training

دارد، اما همیشه شامل فراخوانی یک *تابع آموزش* است که جداول X و y مجموعه‌داده‌ی برچسب‌گذاری‌شده را به عنوان آرگومان دریافت می‌کند. از آنجایی که این جداول فهرستی مشابه از رویدادها را توصیف می‌کنند، باید دارای تعداد سطر یکسانی باشند. در زمانی که مدل آموزش می‌بیند، متغیرهای داخلی خود را طوری تنظیم می‌کند که یک فرمول ریاضی را برای تخمین زدن مقادیر y بر اساس سطرهای X بیابد. به طور معمول، X و y باید نمونه‌های زیادی داشته باشند تا مدل بتواند تخمین‌های خوبی ارائه دهد. بسته به نوع استفاده، تعداد نمونه‌ها می‌تواند هزاران، میلیون‌ها یا میلیاردها باشد.

```
import ModelABC
model ← ModelABC.new()
model.train(X, y)
```

2.1 km	8 yr	120 m^2	$840,000
10.6 km	9 yr	91 m^2	$540,000
0.5 km	3 yr	90 m^2	$745,000
3.5 km	26 yr	35 m^2	$185,000

پیش‌بینی: هنگامی که مدل آموزش داده شد، می توانیم پیش‌بینی [۱] را شروع کنیم. اکنون اگر سطرهایی از X را که مقدار y برای آن‌ها نامعلوم است در نظر بگیریم، می‌توانیم از یک تابع پیش‌بینی برای تخمین y_{pred} استفاده کنیم. این خروجی بهترین حدس مدل برای مقادیر y است که در آینده در زندگی واقعی خواهیم داد. پیش‌بینی‌های این مثال احتمالاً بسیار غیرقابل اعتماد هستند. برای پیش‌بینی بهتر، باید مدل را با بیشتر از چهار مثال برچسب‌گذاری‌شده آموزش دهیم. علاوه بر این، این مدل تنها سه ویژگی را در نظر

```
y_pred ← model.predict(X)
```

$1,332,182	0.4 km	4 yr	114 m^2
$23,958	2.4 km	10 yr	68 m^2
$348,296	3.4 km	16 yr	40 m^2

[۱] Predicting

می‌گیرد. بیایید ببینیم چگونه می‌توان داده‌های بیشتری را برای آموزش مدل‌های قدرتمندتر مورد استفاده قرار داد.

شکل ۵-۳: «به لطف الگوریتم‌های یادگیری ماشین، شورش آخرالزمانی ربات‌ها عمر کوتاهی داشت.» دریافت شده از http://smbc-comics.com.

۱-۵- ویژگی‌ها

برای استفاده از داده‌ها برای یادگیری ماشین، جمع‌آوری و پردازش آن‌ها باید با دقت انجام شود. حتی قبل از اینکه نگران آموزش یک مدل باشیم، باید مطمئن شویم می‌دانیم داده‌هایی که به آن می‌دهیم چگونه باید باشند. برای مثال تصور کنید که در یک بیمارستان کار می‌کنید و برای اهداف لجستیکی، می‌خواهید پیش‌بینی کنید که هر بیمار چه مدت در آنجا می‌ماند. اگر شروع به جستجوی اطراف خود برای یافتن داده‌ها کنید، صدها جدول مختلف در قالب‌های گوناگون پیدا خواهید کرد.

ماشین‌ها هنوز آنقدر هوشمند نیستند که بتوانیم فقط داده‌ها را به آن‌ها بدهیم و انتظار داشته باشیم که خودشان آموزش ببینند و شروع به پیش‌بینی کنند. ما باید داده‌ها را در جداول X و y با ساختاری خوب آماده کنیم که مدل‌ها بتوانند با آن‌ها کار کنند. بیایید ببینیم این کار چگونه انجام می‌شود.

patients

ID	Sex	Blood	Latitude	Longitude	Birthdate
57	F	O	36°53'22" N	27°17'05" E	30-Jan-1957
58	F	B	36°53'24" N	27°17'20" E	29-May-1999
59	M	A	36°45'10" N	26°59'21" E	28-Apr-1956
60	F	O	NULL	NULL	18-Feb-2006
61	NULL	NULL	36°53'36" N	27°17'23" E	04-Nov
62	M	AB	36°53'27" N	27°17'05" E	12-Oct

admissions

Patient	In	Pain	Out
57	May 28	😟	NULL
58	May 28	NULL	May 29
59	May 28	😫	June 7
60	May 28	😖	June 1
61	May 28	😖	May 29
62	May 29	😐	June 5
58	May 30	😟	May 30

patient 58

Time	T
9:00 pm	37.0
10:00 pm	37.2
11:00 pm	37.0

patient 62

Time	T
12:00 am	38.1 °C
1:00 am	38.9 °C
2:00 am	40.3 °C
3:00 am	39.1 °C
4:00 am	38.5 °C

شکل ۵-۴: داده‌های خام بیمارستانی.

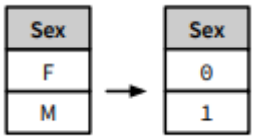
تطبیق داده‌ها

اولین گام در ساختن جداول X و y از مجموعه‌داده‌های خود، این است بررسی کنیم که نوع داده‌ی مناسبی در اختیار داریم. اکثر الگوریتم‌های پیش‌بینی فقط با **داده‌های عددی**[۱] کار می‌کنند. بیایید ببینیم چگونه باید انواع دیگر داده‌ها را پردازش کنیم تا بتوان آن‌ها را نیز به عنوان ویژگی در نظر گرفت.

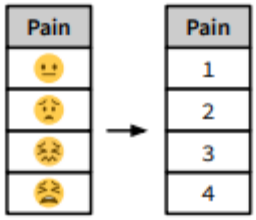
---

۱ Numerical Data

داده‌های طبقه‌بندی‌شده: داده‌هاطبقه‌بندی‌شده[۱] باید به در قالب داده‌های عددی **کدگذاری**[۲] شوند. روش کدگذاری به دودویی، ترتیبی یا اسمی[۳] بودن داده‌ها وابسته است. بیایید ببینیم این به چه معنی است؟

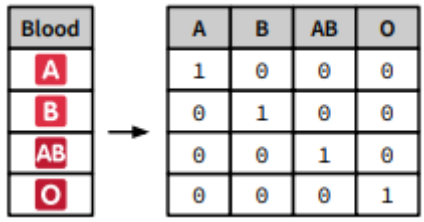


کدگذاری داده‌های دودویی: این داده‌ها در دو دسته قرار دارند، بنابراین یک دسته با 0 و دیگری با 1 نمایش داده می‌شود. مثال روبرو داده‌های خام جنسیت بیولوژیکی[۴] یک بیمار را نشان می دهد.



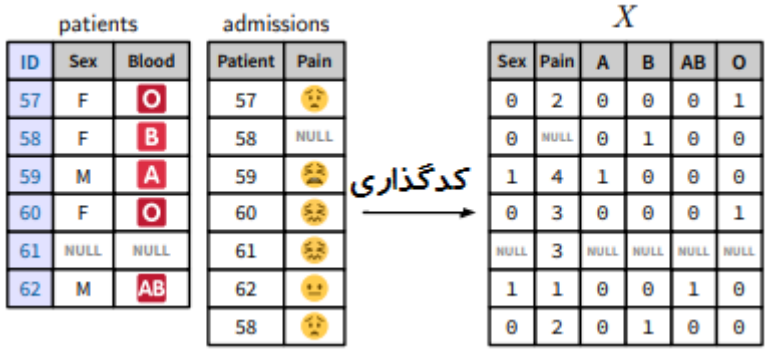
کدگذاری داده‌های ترتیبی: در این داده‌ها طبقه‌ها مرتب شده‌اند، بنابراین ما به آن‌ها شماره می‌دهیم. به عنوان مثال، اگر یک فرم پذیرش در اورژانس از بیماران بخواهد که یکی از چهار ایموجی را برای توصیف درد خود انتخاب کنند، می توان آن را با یک عدد از ۱ تا ۴ نمایش داد.

کدگذاری داده‌های اسمی: وقتی طبقه‌های ستون ما ترتیب ذاتی ندارند، ما معمولاً ستون را با چندین ویژگی دودویی کدگذاری می‌کنیم: یک ویژگی برای هر طبقه. این تکنیک را رمزگذاری **یک-داغ**[۵] می‌نامند. مثال زیر نشان می‌دهد که چگونه از این روش برای نمایش گروه خونی بیمار در قالب چهار ویژگی استفاده می‌شود. از آنجایی که یک بیمار فقط می‌تواند یک گروه خونی داشته باشد، در رکورد کدگذاری شده باید یک سلول حاوی 1 و تمام سلول‌های دیگر حاوی 0 باشند.



---

۱ Categorical Data
۲ Encoding
۳ Binary, Ordinal or Nominal
۴ حدود ۲ درصد از انسان‌ها بیناجنسی به دنیا می‌آیند، برای مثال با کروموزوم XXY. اگر پیش‌بینی‌ها مستقیماً بر روی بیماران تأثیر نگذارند، برای مثال اگر زمان بستری شدن در بیمارستان برای اهداف لجستیکی داخلی تخمین زده شود، می‌توان از کدگذاری دودیی برای این کار استفاده کرد.
۵ One-Hot



ID	Sex	Blood
57	F	O
58	F	B
59	M	A
60	F	O
61	NULL	NULL
62	M	AB

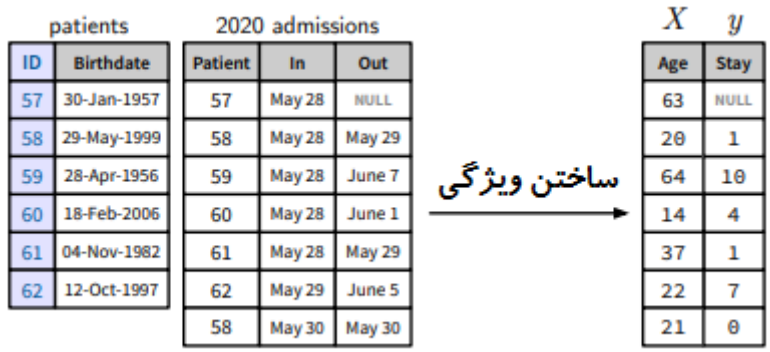
Patient	Pain
57	😟
58	NULL
59	😫
60	😣
61	😣
62	😐
58	😟

Sex	Pain	A	B	AB	O
0	2	0	0	0	1
0	NULL	0	1	0	0
1	4	1	0	0	0
0	3	0	0	0	1
NULL	3	NULL	NULL	NULL	NULL
1	1	0	0	1	0
0	2	0	1	0	0

شکل ۵-۵: گرفتن ویژگی‌ها از انواع مختلف داده‌های طبقه‌بندی‌شده.

داده‌های زمانی: ساده‌ترین راه برای اضافه کردن رکوردهای تاریخ یا زمان استخراج بخش‌های عددی آن‌ها است: یک تاریخ را می‌توان به سه ویژگی جداگانه برای سال، ماه و روز تقسیم کرد. زمان را نیز می‌توان به ساعت، دقیقه و ثانیه تقسیم کرد.

با این حال، این روش به ندرت برای یک الگوریتم پیش‌بینی مفید است. به عنوان مثال، اینکه یک بیمار در ماه آوریل یا نوامبر متولد شده باشد، نباید بر زمان بهبودی او تأثیر بگذارد. از سوی دیگر، بیست ساله یا شصت ساله بودن مهم است. بنابراین اگر تاریخ تولد و تاریخ پذیرش را دارید، می‌توانید با محاسبه‌ی سن بیماران به سال، یک ویژگی جدید **بسازید**. اگر می‌خواهید پیش‌بینی کنید چه مدت در بیمارستان می‌مانند، همین کار را برای y انجام می‌دهید: مدت اقامت تعداد روزهای بین پذیرش و ترخیص است.



ID	Birthdate
57	30-Jan-1957
58	29-May-1999
59	28-Apr-1956
60	18-Feb-2006
61	04-Nov-1982
62	12-Oct-1997

Patient	In	Out
57	May 28	NULL
58	May 28	May 29
59	May 28	June 7
60	May 28	June 1
61	May 28	May 29
62	May 29	June 5
58	May 30	May 30

Age	Stay
63	NULL
20	1
64	10
14	4
37	1
22	7
21	0

شکل ۵-۶: ساختن ویژگی از داده‌های زمانی.

یکی دیگر از راه‌های ساختن یک ویژگی، تبدیل و کدگذاری داده‌های زمانی به عنوان داده‌های طبقه‌بندی‌شده است. به عنوان مثال، اگر بستری شدن در بیمارستان به علت مسمومیت در شب‌ها یا آخر هفته‌ها بیشتر است، این پدیده می‌تواند بر y تأثیر بگذارد. می‌توانید این اطلاعات را در X به عنوان داده‌ی دودویی در دسترس قرار دهید. بسیاری از اطلاعات پنهان را می‌توان از این طریق آشکار کرد، مانند در دسترس بودن داروخانه‌های باز، فصل سال یا وقوع تعطیلات. ویژگی‌هایی برای الگوهای زمانی و فصلی بسازید که احتمالاً با آنچه میخواهید پیش‌بینی کنید مرتبط هستند.

Date	Time
May 28	2:56 am
May 28	4:32 pm
May 29	11:05 pm
May 30	8:40 pm

→

Wknd	Night
0	1
0	0
0	1
1	1

داده‌های جغرافیایی: داده‌های جغرافیایی اغلب با اعداد در قالب مختصات طول و عرض جغرافیایی نشان داده می‌شوند. این اعداد به تنهایی نشان می‌دهند که یک مکان چقدر از شمال و شرق فاصله دارد. این اعداد ممکن است به ویژگی‌های عددی متفاوتی تقسیم شوند، اما درست مانند داده‌های زمانی، این ویژگی‌ها معمولاً کمک چندانی به ما نمی‌کنند.

برای ساختن ویژگی‌های معنادار از داده‌های جغرافیایی، به این فکر کنید مکان‌هایی که دارید با آنچه می‌خواهید پیش‌بینی کنید چه ارتباطی دارند. محاسبه‌ی فواصل اغلب مفید است: اگر می‌خواهید قیمت آپارتمان‌ها را پیش‌بینی کنید، می‌توانید ویژگی‌هایی را برای فاصله‌ی آن‌ها تا مرکز شهر، تا نزدیک‌ترین ساحل یا نزدیک‌ترین مدرسه، سوپرمارکت و بیمارستان ایجاد کنید.

patients

ID	Latitude	Longitude
57	36°53'22" N	27°17'05" E
58	36°53'24" N	27°17'20" E
59	36°45'10" N	26°59'21" E
60	NULL	NULL
61	36°53'36" N	27°17'23" E
62	36°53'27" N	27°17'05" E

ساختن

hospital

Latitude	Longitude
36°52'32" N	27°15'25" E

→

X

Distance
2.9
3.3
27.5
NULL
3.5
3.0

شکل ۵-۷: با تبدیل آدرس بیماران به مختصات جغرافیایی، فاصله‌ی آن‌ها را از بیمارستان می‌توان محاسبه کرد.

مختصات عددی را می‌توان به داده‌های طبقه‌بندی‌شده نیز تبدیل کرد، مانند شهر، محله یا کد پستی. برخی از این داده‌های طبقه‌بندی‌شده حتی می‌توانند به شما اجازه دهند اطلاعات بیشتری را به عنوان ویژگی‌های جدید اضافه کنید، مانند شاخص‌های تطبیق برای سطوح درآمد و جرم و جنایت.

داده‌های غیرساخت‌یافته: تخمین زده می‌شود که حدود ۸۰ درصد از داده‌های دیجیتالی جهان غیرساخت‌یافته هستند و بنابراین نمی‌توان آن‌ها را در قالب ردیف و ستون کدگذاری کرد، بلکه باید آن‌ها را در قالب فایل‌های جداگانه نگهداری کرد. ساختن ویژگی از داده‌های غیرساخت‌یافته دشوار است. برای به دست آوردن داده‌های عددی از داده‌های غیرساخت‌یافته، باید ببینیم که کدام جنبه‌های قابل سنجش آن‌ها با پروژه‌ی ما مرتبط است.

برای مثال تصور کنید که مجموعه‌ای از پرونده‌های بیماران حاوی سی‌تی‌اسکن از ریه‌های آن‌ها باشد. یک عدد مفید که پزشکان می‌توانند از این تصاویر بدست آورند، مساحت ریه‌های هر بیمار است.

رایج ترین شکل داده های غیرساخت‌یافته قالب متنی است. ما به کدام جنبه‌ی کمّی از یک فایل متنی علاقمند هستیم؟ تعداد کل کلمات جنبه‌ای ساده و محبوب است. یکی دیگر از ویژگی‌های رایج، وقوع کلمات خاص است. به عنوان مثال، این واقعیت که یک گزارش پزشکی حاوی کلمات «تومور» یا «سرطان» است، می‌تواند در قالب داده‌های طبقه‌بندی‌شده‌ی دودویی کدگذاری شود. یا اگر تعداد دفعات یک کلمه‌ی کلیدی را در نظر بگیریم و آن را بر تعداد کل کلمات تقسیم کنیم، *فراوانی* ظاهر شدن آن را به دست می‌آوریم[1].

حدس زدن اینکه کدام ویژگی‌های مفید می‌توانند در داده‌های غیرساخت‌یافته پنهان شوند، زمانی آسان‌تر است که بتوانید با کارشناسانی مشورت کنید که به طور معمول آن‌ها را جمع‌آوری و تجزیه و تحلیل می‌کنند. اگر از پزشکان بپرسید که هنگام بررسی گزارش‌های پزشکی یا سی‌تی‌اسکن دقیقاً به دنبال چه چیزی هستند، ممکن است ایده‌های جالبی برای ایجاد ویژگی‌های مرتبط به شما ارائه دهند.

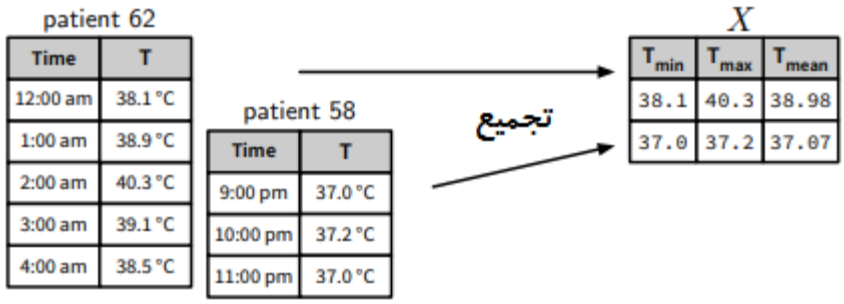
ترکیب داده‌ها

داده‌های خام شما اغلب در اشکال و اندازه‌های متفاوتی هستند و گاهی اوقات نمی‌توان ویژگی‌ها را بدون پردازش قبلی کدگذاری کرد. علاوه بر این، برخی از داده‌ها به تنهایی معنی زیادی ندارند، اما می‌توانیم آن‌ها را با دیگر داده‌ها ترکیب کنیم. بیایید ببینیم چگونه این کار را انجام می‌دهیم.

---

[1] مدل بسته‌ی کلمات یا `Bag-of-Words` پراستفاده‌ترین روش برای ایجاد ویژگی‌ها از فراوانی کلمه‌ها است. برای اطلاعات بیشتر به `http://code.energy/bag-of-words` مراجعه کنید.

تجمیع: اگر اطلاعاتی داشته باشیم که در قالب مجموعه ای از اعداد برای هر رکورد در X یا y ارائه شده‌اند، اغلب نمی‌توانیم هر عدد را به عنوان یک ویژگی کدگذاری کنیم. به عنوان مثال، اگر دمای بدن بیمار را چندین بار در روز بررسی کنیم، مجموعه‌داده‌ی ما حاوی مقادیر زیادی برای دمای بدن هر بیمار خواهد بود. ابزارهای خلاصه‌سازی فصل قبل (بخش ۴-۳) می‌توانند در تبدیل این داده‌ها به ویژگی‌ها کمک کنند:



شکل ۵-۸: ساختن ویژگی‌ها با تجمیع داده‌ها.

هنگام تلاش برای پیش‌بینی عملکرد آینده، تجمیع داده‌ها معمولاً بر روی داده‌های تاریخی استفاده می‌شود زیرا عملکرد گذشته اغلب شاخص خوبی برای عملکرد آینده است. برای مثال، اگر می‌خواهید پیش‌بینی کنید که آیا یک تیم برنده‌ی لیگ ملی هاکی می‌شود یا خیر، ویژگی‌هایی را از خلاصه‌ی پنج عددی امتیازات آن تیم در سال‌های گذشته بسازید.

امتیازها: می‌توانیم یک ویژگی مورد علاقه را در قالب یک عدد بیان کنیم، حتی زمانی که اطلاعات مربوط به ویژگی از متغیرهای مختلف باشد. **امتیاز**[۱] عددی است که توسط یک فرمول ریاضی برای نشان دادن یک ویژگی خاص ساخته می‌شود. برای مثال، شاخص توده‌ی بدن اغلب در مراقبت‌های بهداشتی استفاده می‌شود و بر اساس قد بیمار بر حسب متر و وزن بر حسب کیلوگرم محاسبه می‌شود.

امتیازات خوب طراحی‌شده دانش چگونگی تعامل و همکاری چندین عامل برای ایجاد یک ویژگی معنادار را شامل می‌شوند. محاسبه‌ی یک امتیاز و اضافه کردن آن به X باعث می‌شود آن دانش برای مدل قابل دسترسی باشد. به عنوان مثال، اگر می‌خواهید پیش‌بینی کنید که افراد مبتلا به دیابت می‌شوند و قد و وزن افراد را در اختیار دارید، امتیاز شاخص توده‌ی بدنی را به عنوان یک ویژگی در نظر بگیرید. چاقی یک عامل خطر برای دیابت است و اغلب بهترین معیار قابل دستیابی برای آن، شاخص توده‌ی بدنی است.

---

[۱] Score

admissions

Patient	Weight	Heigth
57	73 kg	1.76 m
58	83 kg	1.65 m
59	56 kg	1.68 m
60	71 kg	1.59 m
61	69 kg	NULL
62	90 kg	2.01 m
58	82 kg	1.65 m

امتیاز

X

BMI
23.6
30.5
19.8
28.1
NULL
22.3
30.1

شکل ۵-۹: شاخص توده‌ی بدنی محاسبه شده بر اساس قد و وزن بیمار.

در علوم اجتماعی، شاخص توسعه‌ی انسانی (HDI) توسعه‌ی یک کشور را از روی نرخ سواد، میانگین امید به زندگی و تولید ناخالص داخلی (GDP) سرانه امتیازدهی می‌کند. در امور مالی، امتیاز اعتباری FICO، اعتبار یک فرد را با توجه به خالص بدهی و سابقه‌ی پرداخت وی توصیف می‌کند.

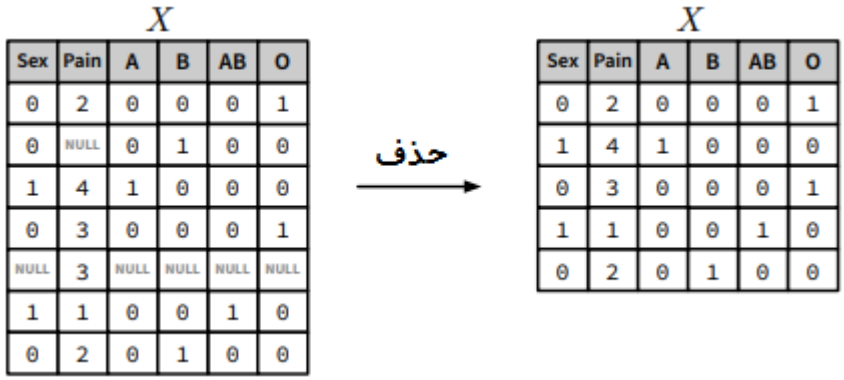
هنگامی که برای یک معیار مشخص امتیاز معینی وجود ندارد، می‌توانید امتیاز مورد نظر خود را ایجاد کنید. فرض کنید در حال تلاش برای پیش‌بینی قیمت فروش آپارتمان‌ها در آینده هستید و معتقدید که ایمنی ساختمان یک شاخص خوب است. شما می‌توانید «امتیاز ایمنی» خود را بسازید که عوامل متعددی را با هم ترکیب می‌کند: تعداد جرایم ثبت‌شده در محله، فاصله‌ی ساختمان تا پاسگاه پلیس، و اینکه آیا ساختمان، نگهبان یا سیستم نظارت تصویری دارد یا خیر.

پس از ایجاد ویژگی‌های جدید، برخی از تکنیک‌های کاوش داده ها را از فصل ۴ مجدداً مرور کنید. برای ویژگی‌های جدید خلاصه و نمودار بسازید. این به شما کمک می‌کند بررسی کنید که آیا ویژگی‌ها به درستی ساخته شده‌اند یا خیر. همچنین، ضریب همبستگی یک ویژگی با y گاهی اوقات می‌تواند پیش‌نمایشی از قدرت پیش‌بینی آن ارائه دهد.

مقادیر از دست‌رفته

به طور معمول، الزام برای *عددی* بودن داده‌ها در X سختگیرانه است: بیشتر الگوریتم‌های پیش‌بینی حتی نمی‌توانند سلول‌های NULL را مدیریت کنند. وقتی مقدار یک ویژگی برای برخی رکوردها در دسترس نیست، باید از شر آن سلول‌های خالی خلاص شویم. چندین راه برای انجام آن وجود دارد.

حذف کردن: ساده‌ترین روش این است که تمام سطرهایی حاوی یک یا چند سلول NULL را از X حذف کنید:



شکل ۵-۱۰: حذف سطرهای ناکامل در X.

زمانی که سطرهای کمی تحت تأثیر قرار می‌گیرند، حذف کردن روش خوبی است. با این حال، کاهش مقدار داده‌های X برای آموزش عموماً کیفیت عملکرد یک مدل پیش‌بینی را کاهش می‌دهد و حذف تعداد زیاد سطر قدرت پیش‌بینی آن را محدود می‌کند. علاوه بر این، حذف سطرها تنها به این معنی نیست که داده‌های کمتری برای آموزش دارید، بلکه به این معنی است که فرصت پیش‌بینی برای هر رویداد مهم با داده‌های از دست رفته را از دست خواهید داد.

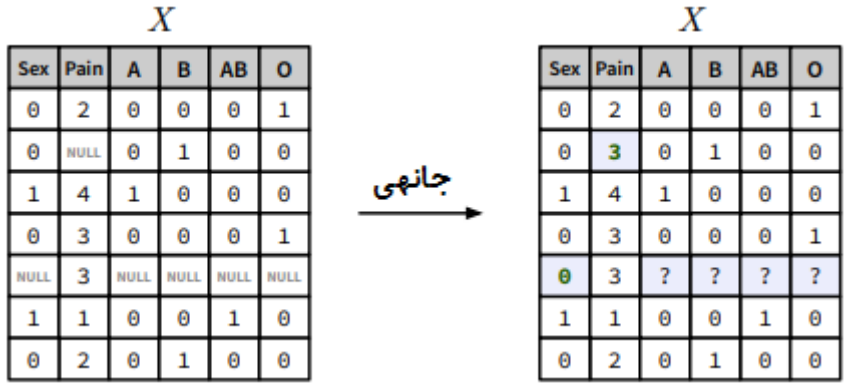
گاهی اوقات، *ستونی* داریم که در آن بیشتر مقادیر از دست‌رفته هستند. در چنین مواردی، اغلب حذف آن ویژگی از X گزینه‌ی بهتری است زیرا به شما امکان می‌دهد سطرهای زیادی را ذخیره کنید.

راه دیگری برای مقابله با داده‌های از دست‌رفته وجود دارد: پر کردن سلول‌های NULL با مقداری که به نظرتان منطقی است. به این فرآیند **جانهی**[۱] می‌گویند و سه روش پرکاربرد برای این کار عبارتند از: *متداول‌ترین مقدار*، *مقدار متوسط* و *برچسب جدید* هستند.

متداول‌ترین مقدار: اگر مقادیر زیادی در یک ستون تکرار شوند، پرتکرارترین مقدار در بین آن‌ها در تمام سلول‌های NULL آن ستون نیز استفاده می‌شود. این روش به ویژه برای حل مشکل مقادیر از دست‌رفته در داده‌های طبقه‌بندی‌شده مفید است.

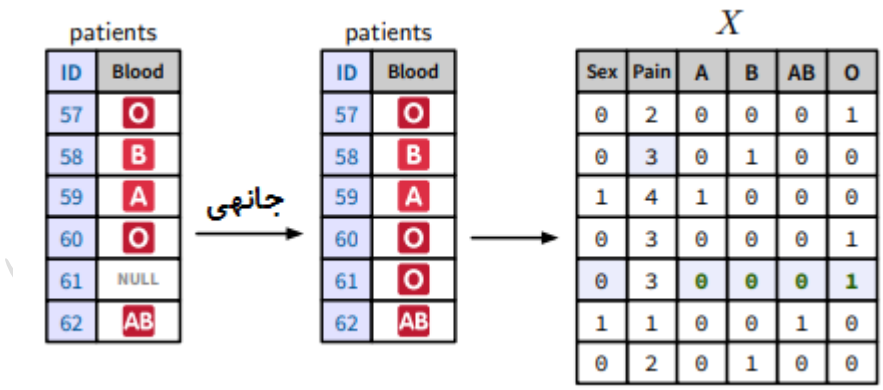
---

[۱] Imputation



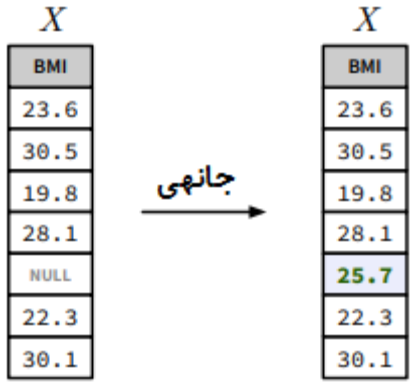
شکل ۵-۱۱: مشکل جانهی برای کدگذاری‌های دودویی و اسمی مانند جنیست بیولوژیک و میزان درد را می‌توان به صورت مستقیم در X حل کرد. با این حال، مشکل کدگذاری یک-داغ برای گروه خونی را نمی‌توان به این روش حل کرد، زیرا تقریباً تمام آن ستون برابر با صفر است.

توجه داشته باشید که این عملیات باید *قبل* از کدگذاری یک-داغ انجام گیرد تا اطمینان حاصل شود که هر سطر کدگذاری شده یک-داغ حداقل حاوی یک سلول است که عدد ۱ را نشان می‌دهد. به این ترتیب از جایگزین کردن تمام علامت سوال در شکل ۵-۱۱ با مقدار صفر اجتناب می‌کنیم.



شکل ۵-۱۲: داده‌های اسمی باید قبل از رمزگذاری یک-داغ جانهی شوند. به این ترتیب یکی از سلول‌های جانهی‌شده برابر با ۱ خواهد بود.

متوسط‌گیری: در این روش، سلول‌های NULL با مقدار متوسط سایر سلول‌های ستون پر می‌شوند:



شکل ۵-۱۳: جانهی سلول‌ها با مقدار متوسط ستون.

برای این کار معمولاً از میانگین یا میانه استفاده می‌شود. مراقب این استراتژی باشید: این روش برای ویژگی‌های طبقه‌بندی‌شده کار نمی‌کند. به عنوان مثال، اگر آن را با داده‌های دودویی امتحان کنید، سلول‌های خالی را ممکن است با مقادیر اعشاری به جای یک و صفر پر شوند.

برچسب جدید: این استراتژی فقط برای متغیرهای طبقه‌بندی‌شده کار می‌کند. وقتی سلول‌های NULL در یک ستون طبقه‌بندی‌شده مکرر هستند، می‌توانیم یک برچسب جدید ایجاد کنیم و آن را به تمام سلول‌های دارای مقادیر از دست‌رفته اختصاص دهیم.

> **آزمایش برای درمان:** بیمارستان شما با یک بیماری همه‌گیر کروناویروس جدید دست و پنجه نرم می‌کند، و شما باید تخمین بزنید که احتمال نیاز بیماران ورودی به دستگاه تنفسی چقدر است. شما بیماران را برای یافتن علائم ویروس آزمایش و یک ویژگی طبقه‌بندی‌شده‌ی ایجاد می‌کنید که در آن هر بیمار «منفی» یا «مثبت» برچسب‌گذاری می‌شود. ابزار آزمایش کافی برای همه وجود ندارد، بنابراین اولویت با بیمارانی است که علائمی مانند تب و سرفه را نشان می‌دهند. در نهایت فقط نیمی از بیماران مورد آزمایش قرار گرفته‌اند که ۶۰ درصد آن‌ها مثبت شده‌اند. چگونه باید سلول‌های NULL بیماران آزمایش‌نشده را جانهی کرد؟

حذف کردن سطرهای مربوط به بیماران آزمایش‌نشده یک گزینه نیست، زیرا نیمی از رکوردها را حذف می‌کند و مدل را برای پیش‌بینی بی‌فایده می کند؛ به عنوان مثال برای پیش‌بینی تعداد کل

ماسک‌های مورد نیاز در آینده. استفاده از متداول‌ترین مقدار نیز مشکل‌ساز است: اکثر آزمایش‌ها مثبت بودند، اما آزمایش‌ها به‌طور تصادفی انجام نشده‌اند، بنابراین نتایج لزوماً به خوبی به بیماران بدون علامت تعمیم نمی‌یابند. این نوعی سوگیری انتخاب است (بخش ۴-۱): احتمال ابتلای به ویروس در بیماران بدون علائم کمتر از افراد دارای علائم است، بنابراین اگر به صورت پیش‌فرض آن‌ها را «مثبت» در نظر بگیرید، اشتباه خواهد بود.

بنابراین، شما دو گزینه دارید: از بیمارستان بخواهید نمونه کوچکی از بیماران بدون علامت را آزمایش کند و نتیجه‌ی آن را تعمیم دهد، یا به سادگی یک برچسب جدید برای بیماران آزمایش‌نشده ایجاد کنید. اولی به شما این امکان را می‌دهد که یک کدگذاری دودویی را برای ویژگی استفاده کنید، در حالی که دومی برای سه برچسب نهایی («منفی»، «مثبتم و «آزمایش‌نشده») به یک کدگذاری یک-داغ نیاز دارد.

حذف رکوردها، انتخاب متداول‌ترین مقادیر، محاسبه‌ی متوسط‌ها و ایجاد برچسب‌های جدید ساده‌ترین و رایج‌ترین روش‌های مقابله با مقادیر از دست‌رفته هستند. دانشمندان به طور فعال در حال تحقیق در مورد روش‌های جدید و موثرتر برای جانهی مجموعه‌داده‌های بزرگ هستند. به طور کلی، در صورتی که داده‌های از دست‌رفته به صورت تصادفی ظاهر شوند، جانهی موثر است. اگر دلیلی اساسی برای NULL بودن برخی سلول‌ها وجود داشته باشد، این استراتژی‌ها ممکن است بیشتر از آن‌که کمک کنند مانع به وجود آورند.

نشت داده‌ها

وقتی X و y را بر اساس رویدادهای گذشته می‌سازیم، ممکن است به طور تصادفی اطلاعاتی را در X بگنجانیم که فقط پس از مشاهده‌ی y وجود دارند. به این پدیده، **نشت داده‌ها**[۱] می‌گویند. اگر این اتفاق بیفتد، مدل به گونه‌ای آموزش داده می‌شود که آینده را از روی داده‌های آینده پیش‌بینی کند، و این کار واقعاً بی‌معنی است.

همزمان با این که ویژگی‌هایی را به X اضافه می‌کنید، به اطلاعاتی فکر کنید که هنگام پیش‌بینی در دسترس خواهند بود. جدول X را بررسی کنید و مطمئن شوید که دقیقاً شبیه داده‌هایی است که مدل شما برای ارزیابی y_{pred} در اختیار خواهد داشت. بیایید این موضوع را در عمل امتحان کنیم:

---

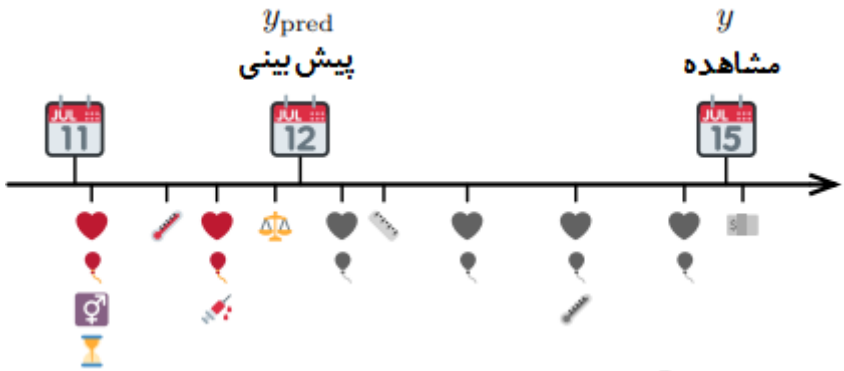
[۱] Data Leakage

تخت‌های شلوغ: شما مسئول سیستم هوش تجاری یک بیمارستان هستید. روز بعد از پذیرش هر بیمار، از شما خواسته می‌شود که مدت زمان بستری شدن وی را در بیمارستان پیش‌بینی کنید. شما به اطلاعات زیر در مورد بیماران قدیمی دسترسی دارید: سن، جنس، شاخص توده‌ی بدنی، گروه خونی، علائم حیاتی، و هزینه‌ی کل بستری. چگونه می‌توانید انتخاب کنید که کدام داده‌ها به عنوان ستون‌های مجموعه‌داده‌ی شما لحاظ شوند؟

اولین قدم این است که بررسی کنید این اطلاعات چه زمانی و چگونه جمع‌آوری شده‌اند. به عنوان مثال، فرض کنید سن ( ) و جنس ( ) در هنگام پذیرش و گروه خونی ( ) در مدت کوتاهی پس از آن جمع‌آوری می شوند. با این حال، قد ( ) و وزن ( ) برای به دست آوردن شاخص توده‌ی بدنی معمولاً پس از چند روز به دست می‌آیند. اندازه‌گیری علائم حیاتی مانند ضربان قلب ( )، فشـار خون ( ) و دمای بدن ( ) در تمام مدت اقامت بیمار انجام می‌شود. در نهایت، هزینه اقامت ( ) تنها پس از ترخیص بیمار قابل محاسبه است:

شکل ۵-۱۴: خط زمان مشاهدات برای یک بیمار فرضی.

مدت اقامت y در هنگام ترخیص مشخص می‌شود. از قبل می‌توانیم بفهمیم که گنجاندن در نمونه‌های برچسب‌گذاری‌شده در جدول ویژگی X بی‌معنی است، زیرا به منزله پیش‌بینی گذشته است! برای تعیین دقیق اینکه چه داده‌هایی نمی‌توانند بر X تأثیر بگذارند، بیایید لحظه‌ی مشاهده y، و زمانی را که می‌خواهیم مدل y_{pred} را تخمین بزند، روی خط زمان اضافه کنیم.



شکل ۵-۱۵: مشاهداتی که به رنگ خاکستری هستند نباید در X حضور داشته باشند.

مقادیر اندازه‌گیری یا مشاهده‌شده پس از لحظه‌ی ارزیابی ypred نباید در X گنجانده شوند، حتی در مرحله‌ی آموزش. در مثال ما، شاخص توده‌ی بدنی نباید در ویژگی‌ها گنجانده شود، زیرا بستگی به دارد، که معمولاً پس از انجام پیش‌بینی اندازه‌گیری می‌شود. علائم حیاتی را در صورتی می‌توان در X گنجاند که تجمیعی از مقادیر جمع‌آوری‌شده در روز اول باشند.

تا زمانی که مطمئن هستید هیچ داده‌ای نشت نمی‌کند، تا جایی که می‌توانید ویژگی‌های بیشتری را استخراج کنید. این کار شانس این که مدل شما الگوهای خوبی برای پیش‌بینی‌های قابل اعتماد بیابد را افزایش می‌دهد. اکنون بیایید ببینیم که چگونه یک مدل را انتخاب کنیم و میزان دقت پیش‌بینی‌های آن را ارزیابی کنیم.

۲-۵- ارزیابی

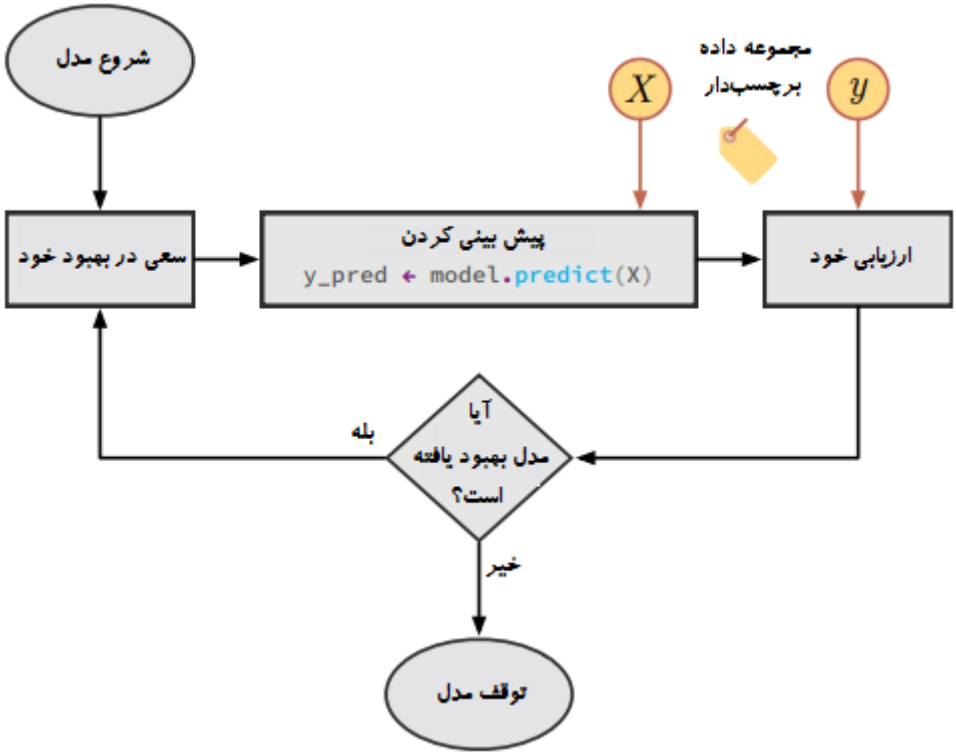
مرحله‌ی بعد از ساخت X و y، انتخاب مدل برای آموزش است. اولین چیزی که متوجه خواهید شد این است که مدل‌ها یا **رگرسور**[۱] یا **طبقه‌بندی‌کننده**[۲] هستند. رگرسورها کمیت‌ها را پیش‌بینی می‌کنند، در حالی که طبقه‌بندی‌کننده‌ها برچسب‌ها را پیش‌بینی می‌کنند. برای پیش‌بینی ملیت افراد، از یک طبقه‌بندی‌کننده استفاده می‌کنید. اما برای پیش‌بینی سن افراد، بهتر است از یک رگرسور استفاده کنید.

یکی از مدل‌های کتابخانه‌ی خود را انتخاب و شروع کنید. در مورد انتخاب یک مدل خاص نگران نباشید، می‌توانید هر کدام را انتخاب کنید. در این مرحله، فقط اطمینان حاصل کنید که اگر y شما از نوع عددی است، یک رگرسور و اگر داده‌ها طبقه‌بندی‌شده هستند، یک طبقه‌بندی‌کننده انتخاب می‌کنید.

---

[۱] Regressor
[۲] Classifier

اکثر مدل ها از تکنیکی به نام **یادگیری نظارت‌شده**[۱] استفاده می‌کنند. روش کار به ای صورت است که مدل یک سری از نمونه‌های آموزشی خود را حدس بزند. هر بار، مدل از تابع پیش‌بینی داخلی خود استفاده می‌کند و سپس با مقایسه خروجی y_{pred} با y واقعی، پیش‌بینی خود را ارزیابی می‌کند. با استفاده از این ارزیابی و برخی ترفندهای ریاضی، سعی می‌کند خود را بهبود بخشد و دوباره حدس بزند. زمانی که بهبود در پیش‌بینی‌ها متوقف شد، مدل آموزش را متوقف می‌کند. فلوچارت زیر فرآیند را به صورت خلاصه نشان می‌دهد:



شکل ۵-۱۶: روال کار یادگیری نظارت‌شده.

اکنون ما یک مدل آموزش‌دیده داریم که می‌توانیم از آن برای پیش‌بینی آینده استفاده کنیم. اما این مدل چقدر خوب می‌تواند پیش‌بینی کند؟ چگونه می‌توان آن را ارزیابی کرد؟ چندین معیار ارزیابی مختلف وجود دارد که یا برای رگرسورها یا برای طبقه‌بندی‌کننده‌ها کار می‌کنند. بیایید با اولی شروع کنیم.

---

[۱] Supervised Learning

ارزیابی رگرسورها

از آنجایی که رگرسورها کمیت‌ها را پیش‌بینی می‌کنند، یک روش طبیعی برای ارزیابی میزان نزدیک بودن پیش‌بینی y_{pred} به y واقعی، محاسبه‌ی **خطای**[۱] بین این دو است:

$$\text{error} = y_{pred} - y$$

این رابطه یک مقدار خطا به ازای هر سطر X و y ارائه می‌دهد. به طور معمول، ما تمام این خطاها را با استفاده از یک مقدار متوسط خلاصه می‌کنیم. چند راه متداول برای انجام این کار وجود دارد:

خطای میانگین قدرمطلق: میانگین پرکاربردترین نوع متوسط است. با این حال، هنگام محاسبه‌ی میانگین، خطاهای مثبت و منفی یکدیگر را خنثی می‌کنند و به ما یک حس کاذب دقت می‌دهند. برای جلوگیری از این امر، **خطای میانگین قدرمطلق**[۲] محاسبه می‌شود، یعنی میانگین تمام خطاها با *نادیده گرفتن علائم منفی آن‌ها*. برای مثال، فرض کنید در حال آموزش مدلی برای پیش‌بینی قیمت آپارتمان به هزار دلار، بر اساس نمونه‌های برچسب‌گذاری شده در شکل ۵-۲ هستیم:

y_{pred} Pred. Price	y True Price	Error	Abs. Error
650	840	-190	190
662	540	122	122
835	745	90	90
179	185	-6	6

$$\text{MAE} = \frac{190 + 122 + 90 + 6}{4} = 102$$

شکل ۵-۱۷: محاسبه‌ی خطای میانگین قدرمطلق برای چهار پیش‌بینی.

با استفاده از این روش، میانگین خطا ۱۰۲,۰۰۰ دلار است. اگر به جای قدرمطلق خطاها، میانگین خطاها را محاسبه کرده بودیم، میانگین گمراه‌کننده‌ی ۴,۰۰۰ دلار را به دست می‌آوردیم! هنگام محاسبه خطای میانگین قدرمطلق همیشه مطمئن شوید که مقادیر مثبت را جمع می‌زنید.

---

[۱] Error

[۲] Mean Absolute Error یا MAE

خطای جذر میانگین مربعات: در برخی موارد، یک پیش‌بینی بسیار نادرست ممکن است منجر به فاجعه شود. اگر نسبت به خطاهای شدید حساس هستیم، می‌توانیم ارزیابی خود را با محاسبه‌ی میانگین مجذور خطاها دقت بیشتری ببخشیم:

y_{pred} Pred. Price	y True Price
650	840
662	540
835	745
179	185

Error	(Error)²
−190	36,100
122	14,884
90	8,100
−6	36

$$\text{RMSE} = \sqrt{\frac{36100 + 14884 + 8100 + 36}{4}} \approx 122$$

شکل ۵–۱۸: محاسبه‌ی خطای جذر میانگین مربعات برای چهار پیش‌بینی.

این بار، پیش‌بینی‌ها متوسط خطای ۱۲۱٫۶۰۰ دلاری را نشان می‌دهند. در اینجا، با اعداد منفی مشکلی وجود ندارد، زیرا مجذور یک عدد منفی همیشه مثبت است. علاوه بر این، خطای جذر میانگین مربعات[۱] جذر میانگین است که مقدار نهایی را به مقیاس اصلی خطاها برمی‌گرداند.

بیایید ببینیم که چگونه انتخاب بین خطای میانگین قدرمطلق و خطای جذر میانگین مربعات بر آموزش تأثیر می‌گذارد. تصور کنید که مدل سعی دارد خودش را بهبود بخشد و اکنون پیش‌بینی‌های زیر را به دست می‌دهد:

y_{pred} Pred. Price	y True Price
748	840
650	540
887	745
281	185

Error	Abs. Error	(Error)²
−92	92	8,464
110	110	12,100
142	142	20,164
96	96	9,216

$$\text{MAE} = 110, \qquad \text{RMSE} \approx 112.$$

شکل ۵–۱۹: ارزیابی پیش‌بینی‌های جدید به دو روش.

[۱] Root Mean Square Error یا RMSE

خطای میانگین قدرمطلق از ۱۰۲,۰۰۰ دلار به ۱۱۰,۰۰۰ دلار افزایش یافته، در حالی که خطای جذر میانگین مربعات از حدود ۱۲۲,۰۰۰ دلار به حدود ۱۱۲,۰۰۰ دلار کاهش یافته است. این امر نشان می‌دهد که سوال «آیا مدل در حال بهبود است؟» از یک مرحله‌ی تکرار در فرایند آموزش به مرحله‌ی دیگر (شکل ۵-۱۶)، بسته به اینکه کدام روش ارزیابی استفاده شود، می تواند پاسخ های متفاوتی داشته باشد.

هنگام پیش‌بینی خروجی‌های با مقیاس‌های مختلف، مراقب باشید که خطاهای مدل را با هم مقایسه نکنید. برای مثال، اگر همان مدل برای پیش‌بینی اجاره‌ی ماهانه‌ی همان آپارتمان‌ها آموزش داده شود، خطای میانگین قدرمطلق و خطای جذر میانگین مربعات احتمالاً حدود ۵۰۰ دلار خواهند بود. این امتیازات کوچکتر نشان نمی‌دهد که مدل بهتر شده است: آن‌ها به سادگی این واقعیت را یادآور می‌شوند که کل ستون y برچسب‌ها حدود ۲۰۰ برابر کوچک‌تر است.

خطای میانگین قدرمطلق و خطای جذر میانگین مربعات می‌توانند درکی از توانایی مدل در پیش‌بینی آینده به وجود آورند. هرچه مدل بهتر به داده‌های جدید تعمیم یابد، این امتیازات به خطاهای پیش‌بینی‌های آینده نزدیک‌تر خواهند بود.

در مورد طبقه‌بندی‌کننده‌ها چطور؟ اگر مدل شما یک طبقه‌بندی‌کننده باشد، برچسب‌ها را همانطور که پیش‌بینی‌می‌کند، خروجی می‌دهد، نه به صورت عدد؛ به این معنی که تکنیک‌های امتیازدهی که دیده‌ایم نمی‌توانند بر روی آن استفاده شوند. امتیاز دادن به یک طبقه‌بندی‌کننده می‌تواند بسیار پیچیده‌تر از امتیاز دادن به یک رگرسور باشد. برای جلوگیری از پرن شدن حواس شما از موضوع اصلی نحوه‌ی ساخت یک سیستم پیش‌بینی، توضیح جامع خود را در مورد نحوه‌ی امتیازدهی طبقه‌بندی‌کننده‌ها در پیوست IV قرار داده‌ایم. مطمئن شوید که آن را بعداً بررسی خواهید کرد!

۳-۵- اعتبارسنجی

گاهی اوقات، یک مدل به گونه‌ای آموزش داده می‌شود که بیش از حد به رویدادهای خاصی که بر اساس آن‌ها آموزش دیده است، وابسته می‌شود. وقتی این اتفاق می‌افتد، مدل مقادیر y را برای رکوردهای مجموعه‌داده‌ی برچسب‌گذاری‌شده بسیار بهتر از رکوردهایی که هرگز ندیده است، تقریب می‌زند. به این پدیده **بیش‌برازش** یا **اورفیتینگ**[1] گفته می‌شود.

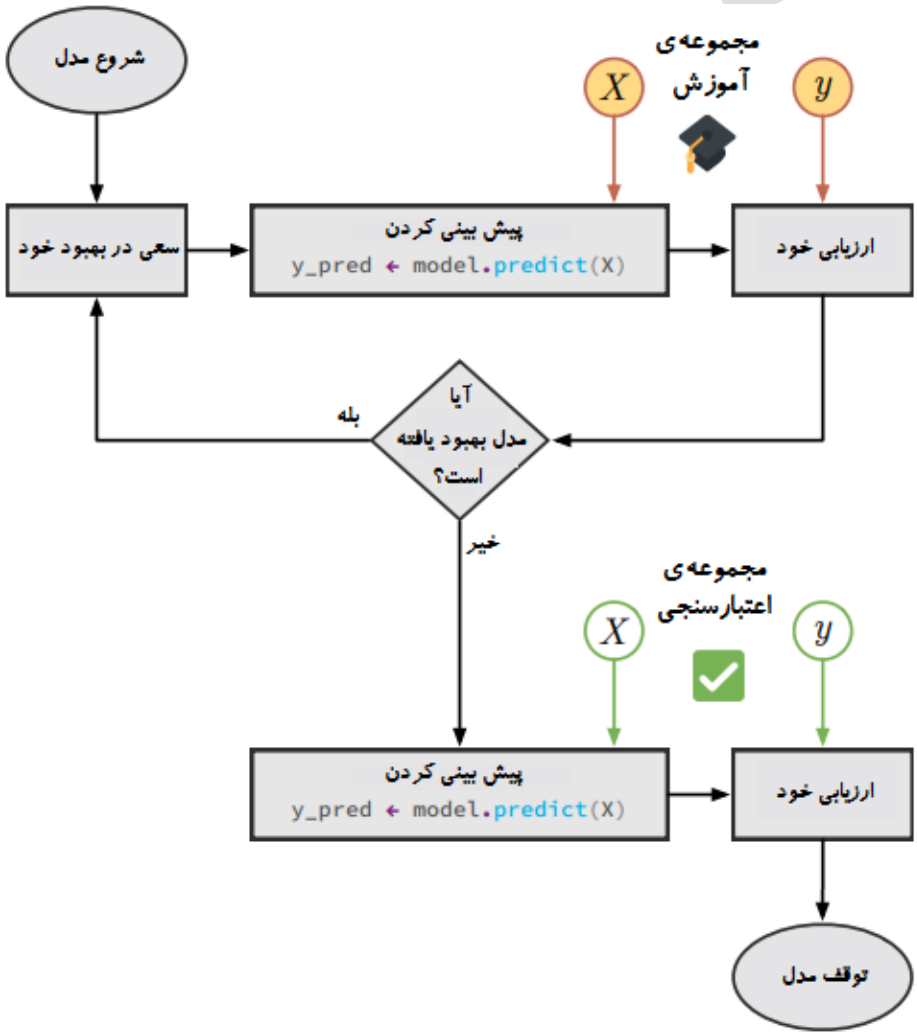
هنگامی که یک مدل دارای بیش‌برازش را برای رویدادهایی که بر اساس آن‌ها آموزش دیده است ارزیابی می‌کنیم، امتیاز حاصل گمراه کننده است. از آن‌جایی که این خطر همیشه وجود دارد، ما مدل را

---

[1] Overfitting

اعتبارسنجی می‌کنیم: پیش‌بینی‌های آن را برای رویدادهایی که در آموزش استفاده *نمی‌شوند* ارزیابی می‌کنیم. این تنها راه برای اطمینان از این است که خوب بودن امتیاز مدل به دلیل برازش بیش از حد نیست. این امتیاز *عملکرد واقعی* مدل را بهتر منعکس می‌کند: امتیازی که می‌توانیم از آن در هنگام پیش‌بینی واقعی در دنیای حقیقی انتظار داشته باشیم.

برای اینکه بتوانیم یک مدل را اعتبارسنجی کنیم، *قبل* از شروع آموزش، باید چند مثال برچسب‌گذاری‌شده را کنار بگذاریم. برای این منظور، سطرهای X و y را به دو **مجموعه‌ی آموزش** و **مجموعه‌ی اعتبارسنجی** تقسیم می‌کنیم. مدل با استفاده از داده‌های مجموعه‌ی آموزش، آموزش داده می‌شود و بر اساس نحوه‌ی پیش‌بینی رویدادها در مجموعه‌ی اعتبارسنجی امتیاز می‌گیرد.



شکل ۵-۲۰: مجموعه‌ی اعتبارسنجی برای ارزیابی نهایی و مستقل مدل کنار گذاشته می‌شود.

توجه داشته باشید که سطرهای مجموعه‌داده‌ی برچسب‌گذاری‌شده باید به صورت *تصادفی* تقسیم شوند. تقسیم مجموعه‌داده بدون به هم ریختن سطرهای آن موجب سوگیری انتخاب در فرایند می‌شود.

هیچ قانون دقیقی برای تعیین اندازه‌ی هر مجموعه وجود ندارد. به طور معمول، ۱۰٪ تا ۲۰٪ از داده‌ها برای اعتبارسنجی کنار گذاشته می‌شوند. هر چه مجموعه‌ی آموزش بزرگ‌تر باشد، مدل بهتر می‌تواند عملکرد را یاد بگیرد. هرچه مجموعه‌ی اعتبارسنجی بزرگ‌تر باشد، مطمئن‌تر می‌شویم که امتیاز نهایی، عملکرد واقعی مدل را منعکس می‌کند. با این حال، دانشمندان داده به سختی تنها به امتیاز واحدی که از طریق این تقسیم ساده‌ی مجموعه‌داده به دست می‌آید تکیه می‌کنند، مگر اینکه مجموعه‌ی اعتبارسنجی دارای صدها هزار رکورد باشد.

متأسفانه، رکوردهایی که برای قرار دادن در مجموعه‌های آموزش یا اعتبارسنجی انتخاب می‌کنیم بر امتیاز نهایی ما تأثیر می‌گذارند. اغلب، توانایی یادگیری مدل بر اساس انتخاب‌های مختلف مجموعه‌داده‌های برچسب‌گذاری‌شده یکسان نیست. اگر مدل را فقط یک بار ارزیابی کنید، این خطر وجود دارد که به دلیل استفاده از یک تقسیم غیرمعمول مجموعه‌داده برا ی آموزش و آزمایش، عملکرد واقعی مدل به خوبی منعکس نشود.

اعتبار سنجی متقابل[۱] با چندین بار تکرار کل فرآیند، این خطر را کاهش می دهد. هر بار، مجموعه‌داده‌ی برچسب‌گذاری‌شده به مجموعه‌های آموزش و اعتبارسنجی متفاوتی تقسیم می‌شود. این کار امتیازات زیادی را ایجاد می‌کند که در مجموع عملکرد مدل را توصیف می‌کنند. بیایید سه روش رایج برای انجام اعتبارسنجی متقابل را بیاموزیم.

اعتبارسنجی K-فولد

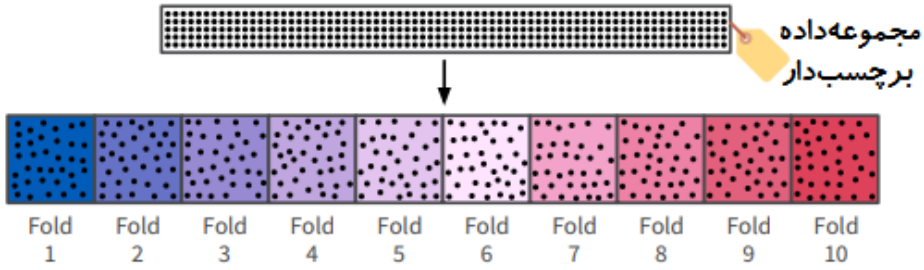
به جای تقسیم مجموعه‌داده‌های به هم‌ریخته به دو قسمت، همانطور که در هنگام ایجاد یک جفت مجموعه‌ی آموزش و اعتبارسنجی عمل کردیم، بیایید مجموعه‌داده‌ها را به ده گروه مختلف با اندازه‌ی (تقریباً) مساوی، که **چین** یا **فولد**[۲] نامیده می‌شوند، تقسیم کنیم (شکل ۵-۲۱).

حال با استفاده از این فولدها، می‌توانیم یک مجموعه‌ی آموزش با پیوستن رکوردها در نه فولد اول بسازیم، در حالی که آخرین فولد را برای اعتبارسنجی نگه می‌داریم (شکل ۵-۲۲).

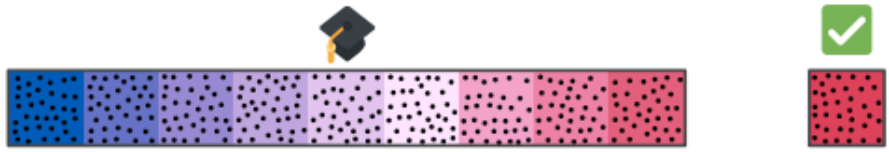
با انتخاب یک فولد متفاوت برای اعتبارسنجی، می‌توانیم ده مجموعه‌ی آموزش و اعتبارسنجی مختلف بسازیم. این امر به ما امکان می‌دهد مدل خود را ده بار ارزیابی کرده و ده امتیاز مختلف به دست آوریم (شکل ۵-۲۳).

---

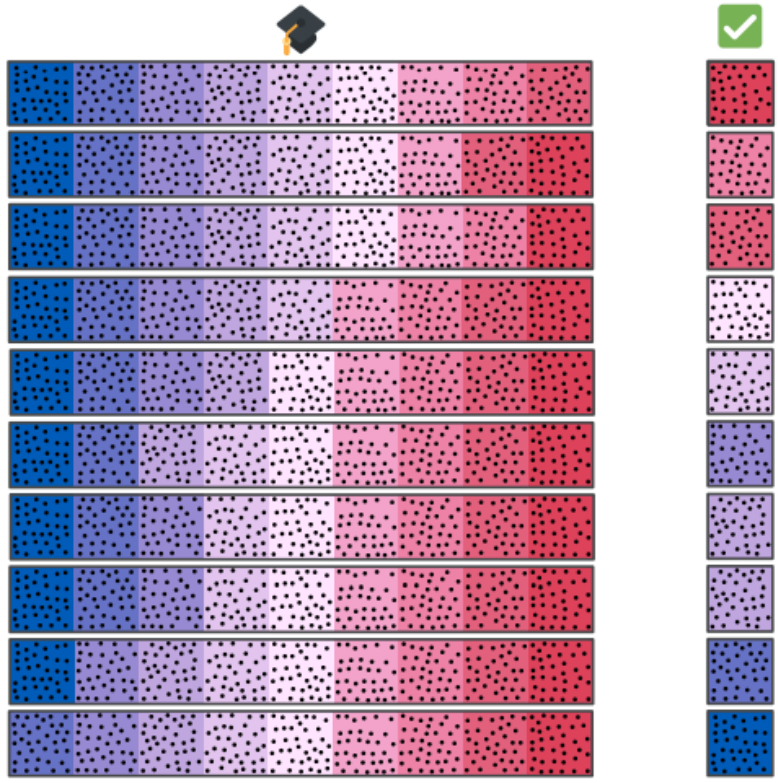
۱ Cross Validation
۲ Fold



شکل ۵-۲۱: در هم‌ریختن ۳۶۵ رکورد در قالب فولدهایی با ۳۶ یا ۳۷ عنصر.



شکل ۵-۲۲: فولدهای ۱ تا ۹ مجموعه‌ی آموزش هستند و فولد ۱۰ مجموعه‌ی اعتبارسنجی است.



شکل ۵-۲۳: ده مجموعه‌ی مختلف آموزش و اعتبارسنجی.

این روش **K–فولد**[۱] نامیده می‌شود، زیرا می‌توان از هر تعداد فولد استفاده کرد. با فولدهای بیشتر، مدل خود را بارها ارزیابی می‌کنید و تعداد رکوردها در مجموعه‌های آموزش افزایش می‌یابد. با این حال، فولدهای بیشتر به محاسبات بیشتری نیز نیاز دارند. همچنین مجموعه‌های آموزش شباهت بیشتری به یکدیگر پیدا می‌کنند که این امر برخی از مزایای اعتبارسنجی مکرر مدل را تحت‌الشعاع قرار می‌دهد. به طور کلی بهتر است به پیکره‌بندی پیش فرضی که اکثر دانشمندان استفاده می‌کنند پایبند باشید: پنج یا ده فولد.

مونت کارلو

در روش K–فولد، هر رکورد فقط یک بار در مجموعه‌ی اعتبارسنجی قرار می‌گیرد. این بدان معناست که هر رکورد یک بار برای اعتبارسنجی مدل استفاده می‌شود. این امر می‌تواند یک مشکل به شمار رود: اگر مدل با انتخابی متفاوت از داده‌های آموزش آموزش داده شده بود، احتمالاً برخی از رویدادها را متفاوت پیش‌بینی می‌کرد. این وضعیت هرگز با استفاده از روش K–فولد بررسی نمی‌شود.

در مقابل، **روش مونت کارلو**[۲] به یک رکورد اجازه می‌دهد تا در مجموعه‌های اعتبارسنجی زیادی برای اعتبارسنجی کامل‌تر مدل حضور داشته باشد. این روش به‌جای استفاده از مجموعه‌ای ثابت از فولدها، قبل از هر تقسیم مجموعه‌داده‌ی برچسب‌گذاری‌شده را دوباره کاملاً به هم می‌ریزد. به این ترتیب، در این روش مجموعه‌های آموزش و اعتبارسنجی تصادفی متعددی انتخاب می‌شوند، سپس مدل برای هر انتخاب تصادفی آموزش داده شده و اعتبارسنجی می‌شود.

روش مونت کارلو زمانی استفاده می‌شود که قدرت محاسباتی فراوانی در اختیار باشد و بتوان مدل را با هزینه‌ی کمی به طور مکرر آموزش داد و اعتبارسنجی کرد. اگر بخواهیم با ۹۰ درصد از ۳۶۵ رکورد خود آموزش را انجام داده و با ۱۰ درصد باقیمانده اعتبارسنجی کنیم، مونت کارلو تقریباً می‌تواند مدل را به تعداد دفعات نامحدود[۳] با مجموعه‌های آموزش و اعتبارسنجی تصادفی متفاوت ارزیابی کند. پس از چند میلیون ارزیابی، در مجموع می‌توان اطمینان بیشتری داشت که امتیازات مختلف عملکرد واقعی مدل را منعکس نمایند.

---

[۱] K-Fold

[۲] Monte Carlo

[۳] از لحاظ نظری، تریلیون‌ها تریلیون تریلیون مجموعه‌ی مختلف اعتبارسنجی ممکن وجود دارد که از ۱۰ درصد از ۳۶۵ رکورد ما استفاده می‌کنند. اولین کتاب ما، مبانی و مفاهیم علوم کامپیوتر، نحوه‌ی شمارش چنین ترکیباتی را توضیح می‌دهد.

اعتبارسنجی یک‌طرفه

گاهی اوقات، مجبوریم با یک مجموعه‌داده‌ی برچسب‌گذاری‌شده بسیار کوچک که تنها چند ده رکورد دارد کار کنیم. در چنین مواردی منطقی است که مجموعه‌ی آموزش را تا حد امکان بزرگ نگه داریم. برای این منظور، مدل را بر روی همه‌ی رکوردها به جز یک رکورد آموزش می‌دهیم و با استفاده از تک رکورد باقی‌مانده اعتبارسنجی می‌کنیم.

این استراتژی را **اعتبارسنجی یک‌طرفه**[1] می‌نامند. می‌توان آن را به عنوان یک نمونه‌ی خاص از روش K-فولد لحاظ کرد: به اندازه‌ی تعداد کل رکورد تعداد فولد وجود دارد، بنابراین هر فولد فقط یک رکورد دارد.

اما یک نکته وجود دارد: همه‌ی نمونه‌های آموزش بسیار شبیه به یکدیگر هستند. اگرچه مدل بارها اعتبارسنجی می‌شود، اعتبارسنجی‌های مختلف بسیار مشابه خواهند بود و برخی از مزایای انجام اعبتارسنجی‌های مکرر را از بین می‌برد. تنها زمانی که نمی‌توانید اندازه‌ی مجموعه‌داده‌ی کوچک خود را افزایش دهید، از اعتبارسنجی یک‌طرفه به عنوان آخرین راه حل استفاده کنید.

تفسیر

اعتبار سنجی متقابل امتیازهای زیادی را برای شما به ارمغان می‌آورد که در مجموع عملکرد مدل شما را توصیف می‌کنند. ما معمولاً این مقادیر را برای تفسیر بهتر آن‌ها خلاصه می‌کنیم، به عنوان مثال با خلاصه‌ی پنج عددی آن‌ها. تفسیر این نتایج می‌تواند به خواسته‌ی شما از مدل بستگی داشته باشد.

به عنوان مثال، اگر مدل گاهی اوقات عملکرد خوبی دارد و عملکرد بد آن برای شما اهمیتی ندارد، بر روی چارک بالا و حداکثر امتیاز تمرکز کنید. از سوی دیگر، اگر نمی‌توانید موارد نادری از عملکرد ضعیف مدل را تحمل کنید، رویکرد بدبینانه را در پیش گرفته و بر روی چارک پایین و حداقل امتیاز تمرکز کنید.

به طور کلی، میانگین به عنوان مرتبط‌ترین خلاصه از نحوه‌ی عملکرد مدل شناخته می‌شود. اما میانگین فقط یک نمایش تقریبی از عملکرد مدل است. بهتر است که آن را همراه با انحراف معیار لحاظ کنید: درجه‌ای از اختلاف عملکرد مدل نسبت به میانگین.

امتیازات اعتبارسنجی متقابل علاوه بر ارائه‌ی معیاری برای عملکرد مورد انتظار مدل، به ما امکان مقایسه مدل‌های مختلف را نیز می‌دهند. انواع مختلفی از مدل‌ها با مکانیزم‌های داخلی متفاوت وجود دارند. برای مقایسه‌ی دو مدل و تصمیم‌گیری درباره‌ی این که کدام بهتر است، به میانگین نگاه نکنید.

---

[1] Leave-One-Out

یک آزمون فرض بر روی امتیازات اعتبارسنجی متقابل هر دو مدل تعریف کنید تا بفهمید که بالاتر بودن میانگین امتیاز یک مدل از مدل دیگر آیا از لحاظ آماری اهمیت دارد یا خیر.

اغلب اوقات، میانگین امتیاز یک مدل با محاسبات سنگین کمی بالاتر از یک مدل ساده‌تر است. با این حال، اگر یک آزمون فرض به ما بگوید که این تفاوت از نظر آماری معنی‌دار نیست، در این شرایط باید مدل ساده‌تر را انتخاب کنید.

۴-۵- تنظیم دقیق

تا کنون یاد گرفتیم که داده‌ها را آماده کرده، یک مدل یادگیری ماشین عمومی را آموزش داده و عملکرد آن را اندازه‌گیری کنیم. اگر در این مرحله توقف کرده و تنظیماتی را در سیستم خود انجام ندهید، تنها به کسری از قدرت پیش‌بینی در دسترس مدل خود خواهید رسید.

یک سیستم پیش‌بینی را می‌توان به روش‌های مختلفی تغییر داد. به عنوان مثال، تنظیم ویژگی‌های انتخاب شده توسط شما می‌تواند عملکرد پیش‌بینی را به طور چشمگیری بهبود بخشد. انتخاب مناسب‌ترین نوع مدل و تنظیم پارامترهای خاص آن برای کار مورد نظر شما نیز کمک بسیاری می‌کند.

دانشمندان داده با آموزش و اعتبارسنجی متقابل یک مدل پیش‌بینی ساده شروع می‌کنند. آن‌ها سپس تغییرات تدریجی کوچکی را در سیستم خود ایجاد کرده و هر بار اعتبارسنجی متقابل جدیدی را انجام می‌دهند. امتیازات قبل و بعد از هر تغییر با هم مقایسه می‌شوند و تغییراتی که از نظر آماری پیشرفت قابل توجهی را به همراه داشته باشند، حفظ می‌شوند. این روند تا زمانی ادامه می‌یابد که پیشرفت بیشتری در سیستم حاصل نشود.

فرآیند تبدیل ویژگی‌ها برای بهبود عملکرد آن‌ها در یک مدل، **مهندسی ویژگی**[۱] نامیده می‌شود. از بین تمام تنظیماتی که می‌توان در یک سیستم پیش‌بینی انجام داد، این مورد اغلب دارای بیشترین پتانسیل برای بهبود است. بیایید برخی از رایج ترین روش‌های مهندسی ویژگی را با هم ببینیم.

بیایید با تنظیماتی که می‌توان در ویژگی‌ها انجام داد شروع کنیم. ویژگی‌ها زمانی موثرتر هستند که اطلاعات واضحی را به مدل ارائه دهند. ستون‌های X شما باید مجموعه‌ای از سرنخ‌هایی باشند که تا حد امکان مستقیماً به آنچه پیش‌بینی می‌کنید مرتبط باشند. گاهی اوقات، تبدیل مقادیر در X می‌تواند به رسیدن به این امر کمک کند.

---

[۱] Feature Engineering

شکل ۵-۲۴: «یادگیری ماشین»، دریافت شده از http://xkcd.com.

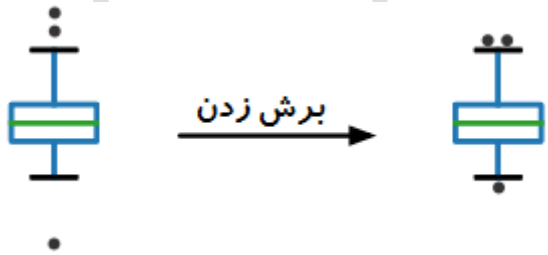
جانهی

در بخش ۵-۱، گفتیم که X نمی‌تواند حاوی سلول‌های NULL باشد، و استراتژی‌هایی را برای مدیریت آن‌ها مورد بحث قرار دادیم: حذف سطرهای دارای داده‌های از دست رفته، حذف ستون‌های دارای داده‌های از دست رفته، پر کردن سلول‌های خالی با مقدار متوسط یک ستون یا با یک مقدار پیش‌فرض. برای اینکه بفهمیم کدام استراتژی‌ها به بهترین وجه با وظایف و داده‌های پیش‌بینی شما مطابقت دارند، آن‌ها را امتحان و عملکرد آن‌ها را ارزیابی کنید!

به عنوان مثال، اگر سلول‌ها را با مقدار میانه‌ی ستون پر می‌کنید، میانگین و یک مقدار پیش‌فرض ثابت را نیز امتحان کنید. برای هر ستونی که بیش از ۱٪ سلول‌های آن NULL هستند، آزمایش کردن اثر چند استراتژی جانهی مختلف به طور کلی ارزشمند است. اگر هیچ کدام از استراتژی‌ها به طور قابل توجهی بهتر از سایر استراتژی‌ها نیست، ساده‌ترین راه را دنبال کنید: تمام سلول‌های خالی را با یک مقدار پیش‌فرض ثابت پر کنید.

داده‌های پرت

وجود داده‌های پرت در X می‌تواند برخی از مدل‌ها را گیج کند. همانند جانهی مقادیر از دست رفته، روش‌های مختلفی برای مقابله با داده‌های پرت وجود دارد. به عنوان مثال، اگر یک ستون فقط تعداد انگشت‌شماری داده‌ی پرت دارد، اغلب بهتر است سطرهای مربوط به آن‌ها را از X حذف کنید. اگر داده‌های پرت مکرر هستند (به عنوان مثال بیش از ۱٪ از سطرها)، می‌توانید سعی کنید آن‌ها را با مقادیر نزدیک‌تر جایگزین کنید. این روش را برش زدن[۱] می‌نامند.



شکل ۵–۲۵: برش زدن سه داده‌ی پرت.

نرمال‌سازی

برخی از مدل‌ها در کار با ویژگی‌هایی که محدوده‌های بسیار متفاوتی را نشان می‌دهند، دچار مشکل می‌شوند. یک ویژگی که مقادیر آن از صفر تا چند هزار متغیر است، اگر مقدار ستون‌های دیگر از صفر تا چند ده متغیر باشند، ممکن است وزن دارای سوگیری بگیرد. برای جلوگیری از این مشکل، ستون‌ها را می‌توان به گونه‌ای تغییر مقیاس داد که همگی محدوده‌ی مشابهی را نمایش دهند. به این روش **نرمال‌سازی**[۲] می‌گویند. ساده ترین روش نرمال‌سازی **MinMax** نام دارد. کمترین مقدار در ستون برابر با صفر و بیشترین مقدار برابر با یک در نظر گرفته می‌شود. تمام مقادیر دیگر در این بین قرار می‌گیرند:

---

۱ Clipping
۲ Normalization

$$x_{new} = \frac{x - min}{max - min}$$

2	5,522
90	7,840
72	9,138
34	8,431
27	1,388
-20	6,983

MinMax →

0.20	0.53
1.00	0.83
0.84	1.00
0.49	0.91
0.43	0.00
0.00	0.72

شکل ۵-۲۶: روش MinMax تمام مقادیر را به بازه‌ی صفر تا یک تقلیل می‌دهد.

همچنین می‌توانید مقادیر را بر اساس فاصله‌ی آن‌ها تا میانگین و انحراف معیار ستون نرمال کنید: مقداری که برابر با میانگین است صفر و مقداری که برابر با انحراف معیار است یک می‌شود. این روش نرمال‌سازی، **نمره‌ی استاندارد** یا **نمره‌ی z**[۱] نامیده می‌شود:

$$x_{new} = \frac{x - \text{mean}}{\text{std. dev.}}$$

2	5,522
90	7,840
72	9,138
34	8,431
27	1,388
-20	6,983

Z-SCORE →

-0.85	-0.20
1.48	0.50
1.00	1.00
0.00	0.73
-0.19	-2.00
-1.43	0.17

شکل ۵-۲۷: در روش نمره‌ی استاندارد، بازه‌ها ثابت نیستند.

در زمان استفاده از MinMax، همه‌ی ستون‌ها *دقیقاً* یک بازه دارند. علاوه بر این، وجود داده‌های پرت باعث می‌ود که داده‌های غیرپرت به یکدیگر نزدیکتر و در مناطق کوچکتر در محدوده‌ی صفر تا یک فشرده شوند. با استفاده از نمره‌ی استاندارد، داده‌های پرت تأثیر کمتری دارند زیرا موجب این فشردگی نمی‌شوند، و محدوده‌ی داده‌های غیرپرت بین ستون‌ها شباهت بیشتری دارد. برای اینکه بفهمید از کدام روش نرمال‌سازی استفاده کنید، هر دو را امتحان و نتایج اعتبارسنجی متقاطع آن‌ها را مقایسه کنید.

---

[۱] Z Score

تبدیل لگاریتمی

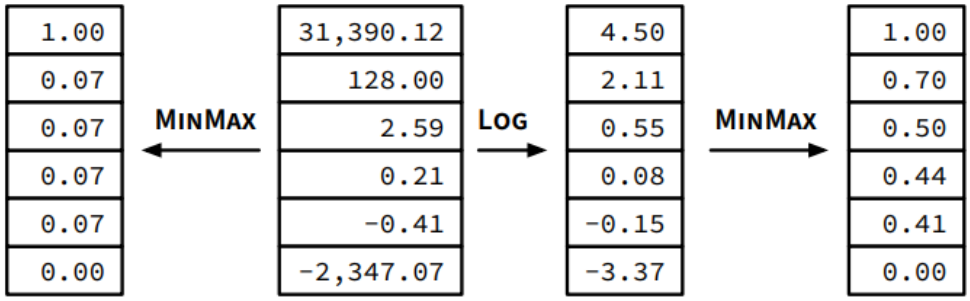
در بخش ۴-۴، هزینه‌های رو به کاهش ذخیره‌سازی کامپیوتر را در دهه‌های گذشته ترسیم و کشف کردیم که داده‌های بسیار بزرگ، نیاز به رسیدگی ویژه دارند. همین امر برای افزودن این نوع داده‌ها به مدل‌های پیش‌بینی نیز صادق است. ویژگی‌های با واریانس زیاد اغلب عملکرد پیش‌بینی را تضعیف می‌کنند.

اگر جدول X شما دارای چنین ستون‌هایی است، سعی کنید مقادیر آن‌ها را با استفاده از یک تابع لگاریتمی (Log) تبدیل کنید. اگر مقادیری بین صفر و یک در ستون وجود دارند، قبل از محاسبه‌ی لگاریتم یک عدد اضافه تا از خروجی‌های منفی بزرگ جلوگیری کنید.

اگر ستون شامل اعداد منفی نیز باشد، تبدیل پیچیده‌تر می شود، زیرا لگاریتم برای چنین اعدادی تعریف نشده است. یکی از راه‌های دور زدن این موضوع حذف علامت منفی، اضافه کردن یک، اعمال لگاریتم و اعمال مجدد علامت منفی است:

$$x_{\log} = \begin{cases} \log(x+1) & if \quad x \geq 0, \\ -1 \times \log\big((-1 \times x)+1\big) & otherwise. \end{cases}$$

علاوه بر این، می‌توانید یک ستون را درست پس از انجام یک تبدیل لگاریتمی، نرمال‌سازی کنید. به عنوان مثال:



شکل ۵-۲۸: روش MinMax با و بدون تبدیل لگاریتمی اولیه.

نرمال‌سازی مستقیم داده‌های دارای واریانس زیاد منجر به نزدیک شدن مقادیر زیادی به یکدیگر می‌شود. با انجام یک تبدیل لگاریتمی قبل از نرمال‌سازی، اعداد بهتر در بازه پخش می‌شوند و درک تفاوت بین مقادیر برای مدل آسان‌تر است.

ظرف‌بندی

گاهی اوقات ایجاد یک ویژگی حاوی اطلاعات کمتر به مدل کمک می‌کند. فرض کنید بیمارستان شما سعی دارد مدت اقامت بیماران ورودی را پیش‌بینی کند و شما یک ستون برای سن آن‌ها دارید. همانطور که بررسی می‌کنید، متوجه می‌شوید که تفاوت‌های سنی اندک تأثیر کمی بر طول مدت بستری شدن در بیمارستان دارند، با این حال تفاوت قابل توجهی بین کودکان و بزرگسالان وجود دارد. در این مورد، سعی کنید به مدل کمک کنید تا بین این گروه‌ها تمایز قائل شود. به عنوان مثال، می‌توانید سن را با برچسب‌های «کودک»، «نوجوان» و «بزرگسال» به یک ویژگی طبقه‌بندی‌شده تبدیل کنید.

این روش **ظرف‌بندی**[۱] نام دارد. این فرایند شامل سازماندهی داده‌های اصلی در قالب چند «ظرف» است که مرتبط‌تر از مقادیر اصلی هستند. همچنین زمانی که ویژگی‌های طبقه‌بندی‌شده دارید و برخی از برچسب‌ها در آن به ندرت ظاهر می‌شوند، این روش می‌تواند مفید باشد؛ آن‌ها را مطالعه و موارد مرتبط با هم را در قالب یک ظرف تجمیع کنید.

به عنوان مثال، یک ویژگی طبقه‌بندی‌شده که نشان‌دهنده‌ی بیماری یک بیمار است، ممکن است هزاران برچسب داشته باشد. قرار دادن آن‌ها در گروه‌هایی از بیماری‌های مرتبط می‌تواند به مدل کمک کند تا داده‌ها را معنا کند. اگر زمان برای ایجاد ظرف‌های پیچیده ندارید، سعی کنید به سادگی همه‌ی برچسب‌های با تکرار کم را با یک برچسب تحت عنوان «سایر» جایگزین کنید. این کار به تنهایی می‌تواند کمک زیادی به مدل شما کند.

خوشه‌بندی

در بیمارستان‌ها، بیماران ورودی اغلب با توجه به علائم یا بیماری‌شان به گروه‌هایی تقسیم می‌شوند. این کار به پزشکان و پرستاران کمک می‌کند: آن‌ها می‌توانند بررسی کنند که یک بیمار به کدام گروه تعلق دارد و سریعاً اطلاعات ارزشمندی در مورد بیمار به دست آورند.

اگر مدیر یک رستوران هستید، تقسیم مشتریان به گروه‌های مختلف نیز می‌تواند مفید باشد. افراد پرخور، عاشقان غذای سالم، افراد خسیس و غیره نمونه‌هایی از این گروه‌ها هستد. مطلع بودن از این گروه‌ها به شما امکان می‌دهد تبلیغات مختلفی را برای مشتریانی که به احتمال زیاد پاسخ می‌دهند، مد نظر قرار دهید. همچنین، رستوران ممکن است پیشنهادات خود را با مشتریان سودآور خود تطبیق دهد.

مراقب باشید: باید موارد را با احتیاط گروه‌بندی کنید تا به مدل شما کمک کند. بیشتر مجموعه‌داده‌ها شامل مجموعه‌هایی از سطرهایی با مشخصه‌های مشابه خواهند بود. اگر در گروه‌بندی این رکوردها بر

---

[۱] Binning

اساس این شباهت‌ها موفق شدید، یک ستون به X اضافه کنید که نشان می‌دهد هر رکورد بخشی از کدام گروه است. اغلب، این ویژگی طبقه‌بندی جدید عملکرد یک مدل پیش‌بینی را بهبود می‌دهد.

فرآیند یافتن ویژگی‌های مشابه و اختصاص هر رکورد به یک گروه، **خوشه‌بندی**[۱] نامیده می‌شود. الگوریتم هایی وجود دارند که می‌توانند خوشه‌بندی را به صورت خودکار انجام دهند. پرکاربردترین الگوریتم خوشه‌بندی کی-میانگین یا **k-means** نام دارد. کتابخانه‌ای را نصب کنید که دارای این الگوریتم باشد و آن را بر روی مجموعه‌داده‌ی خود اجرا کنید. علاوه بر X، الگوریتم از شما می‌خواهد مقدار k را نیز وارد کنید، منظور در اینجا تعداد گروه‌های مختلفی است که باید ایجاد کند. برای شروع، آن را چندین بار اجرا کرده و k را از ۲ تا ۱۰ قرار دهید. هر بار، باید یک ستون اضافی متفاوت برای X به دست آورید. مدل را به ازای هر گروه‌بندی به صورت متقابل اعتبارسنجی کنید و موردی را که بیشتر به مدل شما کمک می‌کند نگه دارید.

اگر از گروه‌بندی خودکار سطرها به‌عنوان یک ویژگی جدید استفاده می‌کنید، یک تمرین اکتشاف داده‌ی جالب وجود دارد که می‌توانید آن را انجام دهید. به رکوردهای اختصاص داده شده به هر گروه نگاه کنید و سعی کنید نامی بیابید که بهترین وجه نشان‌دهنده آن باشد. این بازی گاهی شما را به اکتشافات جالبی می‌رساند!

استخراج ویژگی

در حین جمع‌آوری ویژگی‌ها، X ممکن است ستون‌هایی داشته باشد که ارتباط زیادی با یکدیگر دارند. هنگامی که این اتفاق می‌افتد، اطلاعات افزونه‌ای به مدل منتقل می‌شوند، که در بیشتر موارد چندان مفید نیستند. اکثر مدل‌ها زمانی بهترین عملکرد را دارد که هر ستون X حاوی اطلاعات متمایزی باشد. خوشبختانه، ترفندهای ریاضی خاصی وجود دارند که به شما امکان می‌دهند اطلاعات موجود در گروهی از ستون‌ها را در قالب ویژگی‌های کمتر ولی مفیدتر، فشرده کنید. این فرآیند **استخراج ویژگی**[۲] نامیده می‌شود.

یک ترفند معروف استخراج ویژگی، **تجزیه و تحلیل اجزای اصلی** یا **PCA**[۳] است[۴]. در این روش لازم است که ستون‌های X در ابتدا و از طریق نمره‌ی استاندارد z نرمال‌سازی شوند. روش PCA ضمن حفظ

---

[۱] Clustering

[۲] Feature Extraction

[۳] **P**rincipal **C**omponent **A**nalysis یا **PCA**

[۴] برای توضیح کامل‌تر در مورد PCA صفحه‌ی http://code.energy/pca را ببینید.

اطلاعات اصلی، برخی از ستون‌ها را به ستون‌هایی با همبستگی کمتر تبدیل می‌کند. بیایید مثالی را با هم ببینیم که در آن ستون‌ها میانگین، حداقل و حداکثر دمای روزانه را در یک شهر نشان می‌دهند:

Z-SCORE PCA

T_{avg}	T_{max}	T_{min}	T_{avg}	T_{max}	T_{min}	C_1	C_2	C_3
6.9	7.8	5.0	1.06	0.61	-1.70	-1.93	-0.80	-0.03
6.6	10.6	-0.5	0.97	-1.00	0.03	-1.37	0.94	-0.01
-1.0	1.7	-3.8	-1.16	0.61	-0.98	1.81	-0.01	0.09
4.3	7.8	1.7	0.33	0.74	0.69	-0.94	-0.12	0.27
6.3	8.3	1.1	0.89	-0.87	0.51	-1.25	0.18	-0.15
1.2	2.2	-2.1	-0.54	2.2	-0.46	1.08	-0.24	-0.18
-2.4	0.0	-5.5	-1.55	-1.45	-1.49	2.59	0.06	0.01

ضریب همبستگی

T_{avg}/T_{max}	0.96	T_{avg}/T_{max}	0.96	C_1/C_2	0.00
T_{avg}/T_{min}	0.88	T_{avg}/T_{min}	0.88	C_1/C_3	0.00
T_{max}/T_{min}	0.77	T_{max}/T_{min}	0.77	C_2/C_3	0.00

انحراف معیار

T_{avg}	3.85	T_{avg}	1.08	C_1	1.79
T_{max}	4.08	T_{max}	1.08	C_2	0.52
T_{min}	3.56	T_{min}	1.08	C_3	0.15

شکل ۵-۲۹: ویژگی‌ها از طریق z-score و PCA تغییر یافته‌اند. در هر مرحله، ضریب همبستگی بین ستون‌ها و همچنین انحراف معیار هر ستون نشان داده شده است.

روش PCA یک X جدید را برمی‌گرداند که در آن ستون‌ها همبستگی کمی با یکدیگر دارند. علاوه بر این، PCA سعی می‌کند اطلاعات بیشتری را در ستون‌های سمت چپ فشرده و با کاهش سطوح واریانس آن‌ها را مرتب کند. سپس می‌توانیم ستون‌های سمت راست که واریانس کمی را نشان می‌دهند، کنار بگذاریم و ببینیم آیا مدل بهتر عمل می‌کند یا خیر. اغلب این کار جواب می‌دهد!

یکی دیگر از ترفندهای استخراج ویژگی، **فاکتورگیری نامنفی ماتریس** یا **NMF**[۱] است. همانطور که از نام آن پیداست، این روش فقط در صورتی کار می‌کند که X دارای مقادیر منفی نباشد. این روش به ما امکان می‌دهد اطلاعات X را در تعدادی از ستون‌های انتخاب شده‌ی خود فشرده کنیم. بیایید از NMF برای تبدیل داده‌های مثال قبلی خود به دو ستون استفاده کنیم:

MinMax NMF(2)

T_{avg}	T_{max}	T_{min}
6.9	7.8	5.0
6.6	10.6	-0.5
-1.0	1.7	-3.8
4.3	7.8	1.7
6.3	8.3	1.1
1.2	2.2	-2.1
-2.4	0.0	-5.5

T_{avg}	T_{max}	T_{min}
1.00	0.74	1.00
0.97	1.00	0.48
0.15	0.16	0.16
0.72	0.74	0.69
0.93	0.78	0.63
0.39	0.21	0.32
0.00	0.00	0.00

C_1	C_2
0.67	0.67
0.91	0.00
0.13	0.09
0.62	0.34
0.74	0.27
0.22	0.24
0.00	0.00

ضریب همبستگی

T_{avg}/T_{max}	0.96
T_{avg}/T_{min}	0.88
T_{max}/T_{min}	0.77

T_{avg}/T_{max}	0.96
T_{avg}/T_{min}	0.88
T_{max}/T_{min}	0.77

C_1/C_2	0.34

انحراف معیار

T_{avg}	3.85
T_{max}	4.08
T_{min}	3.56

T_{avg}	0.41
T_{max}	0.48
T_{min}	0.34

C_1	0.35
C_2	0.24

شکل ۵-۳۰: ویژگی ها از طریق MinMax و NMF تغییر یافته‌اند. در هر مرحله، ضریب همبستگی بین ستون‌ها و همچنین انحراف معیار هر ستون نشان داده شده است.

برخلاف PCA، NMF ستون‌های نامرتبی تولید می‌کند که همه به یک اندازه مرتبط هستند. در حالی که PCA مجموعه‌ای از ستون‌ها را با میزان ارتباط نزولی به عنوان خروجی ارائه می‌دهد، NMF تلاش

[۱] Non-negative Matrix Factorization یا NMF

می‌کند اطلاعات را به ستون‌هایی تقسیم کند که رفتارهای متمایز را نشان می‌دهند. طبق معمول، آزمایش با تعداد متفاوتی از ستون‌های خروجی مفید است. مدل خود را با جداول NMF در اندازه‌های مختلف آزمایش و عددی که بهترین کارایی را دارد انتخاب کنید.

پیش از استفاده از PCA یا NMF، ستون‌های X معمولاً یک مقدار یا ویژگی خاص را نشان می‌دهند. پس از هر یک از این تبدیل‌ها، ستون‌ها دیگر پیوند مستقیمی به یک ویژگی از دنیای واقعی ندارند، حتی اگر به طور تجمعی اطلاعات اصلی را حفظ کنند. این امر ممکن است به یک ماشین کمک کند، اما شفافیت داده‌ها را برای انسان کمتر می‌کند. به همین دلیل، استخراج ویژگی معمولاً آخرین مرحله از فرآیند مهندسی ویژگی است.

انتخاب ویژگی

برخی از ویژگی‌ها ممکن است به جای کمک به عملکرد بهتر مدل، مانع از عملکرد خوب آن شوند. این پدیده اغلب در مورد ویژگی‌های اضافی، ویژگی‌های حاوی داده‌های با کیفیت بد، یا ویژگی‌هایی که واقعاً به y مرتبط نیستند اتفاق می‌افتد. حذف ستون‌های غیرمفید از X را **انتخاب ویژگی**[۱] می‌نامند.

زمانی که از زیرمجموعه‌های مختلفی از ویژگی‌ها استفاده می‌کنیم عملکرد مدل متفاوت خواهد بود؛ در این شرایط می‌خواهیم زیرمجموعه‌ای را انتخاب کنیم که عملکرد را به حداکثر می‌رساند. تنها راه تضمین این امر، ارزیابی تمام انتخاب‌های ممکن از ستون‌ها و بررسی بهترین انتخاب یافت شده است. این کار یک **جستجوی جامع**[۲] نامیده می‌شود، و تنها در صورتی امکان‌پذیر است که X ستون‌های کمی داشته باشد. در صورت زیاد بودن تعداد ویژگی‌ها، تعداد زیرمجموعه‌هایی که باید بررسی شوند، بسیار زیاد خواهد بود[۳].

در این موارد، ویژگی‌ها را از طریق *روش‌های اکتشافی* انتخاب می‌کنیم: روش‌هایی که بدون تضمین رسیدن به بهترین راه‌حل یا بهینه بودن آن، به یک راه‌حل منجر می‌شوند. ساده‌ترین این روش‌ها، روش **حذف رو به عقب**[۴] نامیده می‌شود. در این روش بررسی می‌کنیم که آیا حذف یکی از ویژگی‌ها عملکرد را بهبود می‌بخشد یا خیر. اگر باعث بهبود عملکرد می‌شود، آن ویژگی حذف می‌شود. زمانی که همه‌ی ویژگی‌ها را حذف کرده ولی موفقیتی به دست نیاوردیم، فرایند را متوقف کرده و ویژگی‌های باقی‌مانده را حفظ می‌کنیم:

---

۱ Feature Selection

۲ Exahuastive Search

۳ تعداد زیرمجموعه‌های یک مجموعه‌ی دارای n عنصر، 2^n است.

۴ Backward Elimination

```
keep_changing ← True
while(keep_changing)
    keep_changing ← False
    for each column in X
          X_new ← X.remove(column)
          if is_better(X_new, X)
                X ← X_new
                keep_changing ← True
                break
```

فرآیند معکوس این روش، **انتخاب رو به جلو**[۱] نامیده می‌شود: با مجموعه‌ی ویژگی‌های تهی شروع کنید و تا زمانی که عملکرد را بهبود می‌بخشند، یک به یک به اضافه کردن ویژگی‌های جدید ادامه دهید. هنگامی که بین یک گروه بزرگ‌تر و یک گروه کوچک‌تر از ویژگی‌ها تفاوت وجود دارد، گزینه‌ی کوچک‌تر را انتخاب کنید[۲].

استراتژی‌های انتخاب ویژگی بسیار بیشتری وجود دارند. این روش‌ها به طور کلی یک مدل را ساده‌تر و سریع‌تر می‌کنند، علاوه بر آن عملکرد آن را نیز بهبود می‌بخشند.

نشت داده‌ها، دوباره

دیدیم که مجموعه‌داده‌های برچسب‌گذاری‌شده‌ی ما باید به مجموعه‌های آموزش و اعتبارسنجی تقسیم شوند تا یک مدل ارزیابی شود. از آنجایی که مجموعه‌ی اعتبارسنجی برای سنجش عملکرد واقعی مدل در خارج از قلمرو آموزش آن استفاده می‌شود، هرگز نباید بر آموزش مدل تأثیر بگذارد.

این امر در مورد مهندسی ویژگی نیز صدق می‌کند: همه‌ی تبدیل‌ها بر آموزش مدل تأثیر می‌گذارند و بنابراین *فقط* باید با استفاده از مجموعه‌ی آموزش کالیبره شوند. عدم انجام این کار، همانطور که در بخش ۱-۵ توضیح داده شد، به منزله‌ی نشت داده است. بیایید چند نمونه را مرور کنیم تا مطمئن شویم این موضوع را درست درک کرده‌اید.

فرض کنید که در حال اجرای جانهی میانگین در یکی از ستون‌های خود هستید. مقادیر میانگین فقط از سطرهای مجموعه‌ی آموزش محاسبه می‌شوند. سلول‌های خالی مجموعه‌های آموزش و اعتبارسنجی هر دو با *میانگین مجموعه‌ی آموزش* پر می‌شوند. به این ترتیب هیچ اطلاعاتی از مجموعه‌ی اعتبارسنجی به مجموعه آموزش نشت نمی‌کند.

---

[۱] Forward Selection

[۲] حذف رو به عقب و انتخاب رو به جلو نمونه‌هایی از روش‌های اکتشافی حریصانه هستند که در کتاب اول خود، مبانی و مفاهیم علوم کامپیوتر، توضیح دادیم.

به طور مشابه، همیشه تنها بر اساس مجموعه‌ی آموزش باید تصمیم بگیرید که کدام مقادیر پرت در نظر گرفته شوند. پس از اتخاذ این تصمیم، دقیقاً همان عملیات عددی را برای مجموعه‌های آموزش و اعتبارسنجی اعمال کنید.

در زمان نرمال‌سازی از طریق `MinMax`، از مقادیر حداقل و حداکثر مجموعه‌ی آموزش به عنوان مرجع خود برای تبدیل مجموعه‌ی آموزش و اعتبارسنجی استفاده کنید. اگر نرمال‌سازی را از طریق نمرات استاندارد انجام می‌دهید، میانگین و انحراف معیار مجموعه‌ی آموزش را در نظر بگیرید. هنگام به کارگیری استراتژی‌های ظرف‌بندی، وجود برچسب‌ها در مجموعه‌ی آموزش را فقط برای تصمیم‌گیری در مورد نحوه‌ی طبقه‌بندی داده‌ها در نظر بگیرید.

قوانین مشابهی برای استخراج ویژگی اعمال می‌شوند: الگوریتم `PCA` یا `NMF` باید با استفاده از مجموعه‌ی آموزش کالیبره شوند. سپس همان عملیات ریاضی در هر دو مجموعه‌ی آموزش و اعتبارسنجی انجام خواهد شد.

دیدیم که مجموعه‌ی آموزش برای هر ارزیابی مستقل در طول اعتبارسنجی متقابل تغییر می‌کند. این بدان معناست که تغییرات مهندسی ویژگی شما باید مجدداً کالیبره شده و برای هر جفت مجموعه‌ی آموزش و اعتبارسنجی جدید اعمال شوند. مراقب باشید: تغییرات مهندسی ویژگی را فقط یک بار و قبل از اعتبارسنجی متقابل کالیبره و اعمال نکنید، بلکه چندین بار در طول اعتبارسنجی متقابل آن‌ها را انجام دهید. این یک اشتباه رایج است که اغلب منجر به پیش‌بینی‌های نادرست می‌شود.

انتخاب مدل

کتابخانه‌های یادگیری ماشین منبع‌باز برای اکثر زبان‌های برنامه نویسی مدرن در دسترس هستند. این کتابخانه‌ها معمولاً با ده‌ها مدل مختلف از قبل آماده شده‌اند و می‌توانید به راحتی از آن‌ها استفاده کنید.

هر مدل مختلف از ترفندهای ریاضی متفاوتی برای آموزش و پیش‌بینی استفاده می‌کند. وقتی به یک متخصص باتجربه‌ی یادگیری ماشین تبدیل می‌شوید، در نهایت خواهید آموخت که مدل‌ها در موقعیت‌های مختلف چگونه کار و رفتار می‌کنند. این تجربه به شما کمک می‌کند از مدل‌های مختلف به طور موثرتری استفاده کنید.

اگر تازه یادگیری ماشین را شروع کرده‌اید، اشکالی ندارد که همه‌ی مدل‌های موجود در کتابخانه‌ی یادگیری ماشین خود را امتحان کنید، حتی اگر هیچ سرنخی از نحوه‌ی عملکرد آن‌ها در ذهن ندارید. همانطور که آن‌ها را امتحان می‌کنید، به طور تجربی کشف خواهید کرد که کدام یک برای کارهای پیش‌بینی که بر روی آن‌ها کار می‌کنید، بهترین هستند.

از هر مدلی که استفاده می‌کنید، فقط باید به یک نکته توجه داشته باشید: تنظیمات پیکره‌بندی آن مدل که به عنوان **فراپارامترها**[۱] شناخته میشود. این پارامترها باید با مطابق با کار پیش‌بینی شما تنظیم شوند. پایبندی به تنظیمات پیش‌فرض اغلب منجر به عملکرد بد می‌شود و شما را از ارزیابی منصفانه‌ی پتانسیل یک مدل باز می‌دارد.

برای تنظیم صحیح فراپارامترهای خود، با بررسی مستندات مدل شروع کنید. هنگامی که می‌دانید گزینه‌های شما چه هستند، شروع به آزمایش کنید. برای مثال، می‌توانید مقادیر پیش‌فرض هر فراپارامتر را دوبرابر یا نصف کنید و ببینید که چگونه بر عملکرد مدل تأثیر می‌گذارد. به بررسی فراپارامترها ادامه دهید تا زمانی که مطمئن شوید تنظیمات مناسب برای هر کدام را پیدا کرده‌اید[۲]. پس از تنظیم فراپارامترهای هر مدلی که قصد ارزیابی آن را دارید، می‌توانید عملکرد آن‌ها را مقایسه و در نهایت بهترین عملکرد را انتخاب کنید.

گام‌های نهایی

پس از تنظیم ویژگی‌ها و مدل خود، زمان آن است که سیستم را برای استقرار در دنیای واقعی آماده کنید. از آنجایی که قبلاً در مورد همه‌ی تنظیمات خود تصمیم‌گیری کرده‌اید، دیگر نیازی به مجموعه‌ی اعتبارسنجی ندارید. تمام قوانین مربوط به جانهی، مدیریت داده‌های پرت، نرمال‌سازی، و قوانین ظرف‌بندی، و همچنین تبدیل‌های استخراج ویژگی خود را در نظر گرفته و آن‌ها را با استفاده از کل مجموعه‌داده‌ی برچسب‌گذاری‌شده‌ی خود کالیبره کنید.

گام بعدی این است که مدل انتخابی را با فراپارامترهای ایده‌آل خود تنظیم کرده و آن را با استفاده از مجموعه‌داده‌ی برچسب‌گذاری‌شده‌ی خود آموزش دهید. قواعد تبدیل ویژگی و مدل آموزش‌دیده‌تان در کنار هم، سیستم پیش‌بینی نهایی شما و آماده‌ی استفاده است.

هنگامی که یک رویداد جدید و هرگز دیده‌نشده رخ می‌دهد، ویژگی‌های آن را به سیستم بدهید. اگر همه‌ی کارها را به درستی انجام داده باشید، سیستم نگاهی اجمالی به آینده‌ی ناشناخته‌ی رویداد خواهد داشت. این نگاه به اندازه‌ی میزان دقت مدل در اعتبارسنجی متقابل دقیق خواهد بود. آیا این گونه است؟

متأسفانه، از آنجایی که شما بارها و بارها از کل مجموعه‌داده برای تنظیم سیستم خود استفاده کرده‌اید، هنوز این احتمال وجود دارد که در مرحله‌ی تنظیم دقیق مقداری بیش‌برازش رخ داده باشد. این

---

[۱] Hyperparameter

[۲] با کسب کمی تجربه، روش‌هایی را کشف خواهید کرد که به طور خودکار در تنظیمات بسیاری از فراپارامترهای یک مدل خاص جستجو کرده و بهترین را به طور خودکار انتخاب می‌کنند.

به آن معنی است که سیستم شما با مجموعه‌داده‌های شما بهتر از هر رویداد دیگری تنظیم می‌شود. به عبارت دیگر، سیستم در رابطه با رکوردهای مجموعه‌داده‌ی برچسب‌گذاری‌شده‌ی شما بهتر از رویدادهایی که قبلاً دیده نشده‌اند، عمل می‌کند.

مجموعه‌ی آزمایش: تنها راه برای کسب اطمینان از ارزیابی کاملاً بی‌طرفانه‌ی یک سیستم پیش‌بینی، ارزیابی آن با داده‌هایی است که هرگز با آن‌ها در تماس نبوده است. برای این منظور، می‌توانستیم قبل از شروع اعتبارسنجی متقابل، بخش کوچکی از مجموعه‌داده‌ی برچسب‌گذاری‌شده‌ی خود را در قالب یک مجموعه‌ی آزمایش جدا کنیم. مجموعه‌ی آزمایش هرگز برای آموزش، اعتبارسنجی یا آماده‌سازی نهایی مدل استفاده نمی‌شود. این مجموعه فقط برای یک ارزیابی نهایی و منفرد از مدلی که قرار است فعال شود، استفاده می‌شود. اگر داده‌های برچسب‌گذاری‌شده بی‌طرفانه باشند، عملکرد مدل در مجموعه‌ی آزمایش باید نمونه‌ای حقیقی از عملکرد آن در دنیای واقعی باشد.

نتیجه‌گیری

ساختن یک سیستم یادگیری ماشین یک فرآیند پیچیده است. برای به دست آوردن نتایج پیش‌بینی قابل استفاده، جزئیات زیادی وجود دارند که باید به درستی به دست آورید. بنابراین از کتابخانه‌های موجود استفاده کنید: آن‌ها به شما در انجام مهندسی ویژگی، اعتبارسنجی متقابل و ارزیابی مدل کمک می‌کنند. بسیاری از آن‌ها می‌توانند تقریباً هر کاری را که در این فصل دیدیم انجام دهند.

هنگام استفاده از چنین کتابخانه‌هایی، تبدیل داده‌ها و مراحل اعتبارسنجی متقابل فقط به چند خط کد نیاز دارند. معمولا برنامه‌نویسان از نوت‌بوک‌های کد برای به اشتراک‌گذاری این نوع کد استفاده می‌کنند. نوت‌بوک‌های کد، متن، کد و نمودارها را در یک فایل ترکیب می‌کنند و به راحتی می‌توان آنها را مشاهده، اجرا و آزمایش کرد.

اصل تکرارپذیری که در فصل قبل به طور مفصل مورد بحث قرار گرفت، در مورد یادگیری ماشین نیز صدق می‌کند. اطمینان حاصل کنید که همه‌ی تبدیل‌های داده‌های شما قابل تکرار هستند و در خط لوله‌ای ساخته شده‌اند که به راحتی قابل تغییر است. همین امر در مورد فرایندهای تنظیم دقیق مدل شما و فراپارامترهای آن نیز صدق می‌کند.

به خاطر داشته باشید که این مقدمه‌ای برای تمرین یادگیری ماشین بود. ترفندهای زیادی وجود دارند که به آن‌ها اشاره نکردیم. به عنوان مثال، روش‌هایی برای ترکیب نتایج پیش‌بینی مدل‌های مختلف در یک مجموعه‌ی کلی وجود دارد که اغلب از هر مدل مجزایی بهتر عمل می‌کند.

یادگیری ماشین با سرعت زیادی در حال توسعه است. روش‌های جدیدی برای مهندسی ویژگی و انتخاب مدل هر سال منتشر می‌شوند و شرکت‌های بزرگ فناوری اکنون خدمات محاسبات ابری را با بسترهای خودکار برای ایجاد مدل پیش‌بینی ارائه می‌کنند. با تمرین مفاهیم ارائه شده در این فصل، می‌توانید بهترین استفاده را از این منابع داشته باشید... و انقلاب هوش مصنوعی را در زمان رخ دادن آن درک کنید.

می‌تواند ترسناک باشد که فکر کنیم ماشین‌ها بر تصمیمات ما تأثیر می‌گذارند، یا این‌که کاملاً آنها را برای ما تصمیم می‌گیرند. هر زمان که ممکن است پیش‌بینی بر زندگی یک شخص تأثیر بگذارد، به عواقب احتمالی آن فکر کنید. یادگیری ماشین نه خوب است و نه بد: این وظیفه‌ی شماست که از آن به صورت اخلاقی و مسئولانه استفاده کنید.

شکل ۵-۳۱: یادگیری ماشین هر مشکلی را حل نمی‌کند.

منابع و مراجع

- Machine Learning for Absolute Beginners, by Theobald
  - https://code.energy/theobald
- The Data Science Design Manual, by Skiena
  - https://code.energy/skiena
- Deep Learning, by Goodfellow
  - https://code.energy/goodfellow
- The Elements of Statistical Learning, by Tibshirani
  - https://code.energy/tibshirani
- Prof. Raschka's paper on Model Evaluation and Selection
  - https://code.energy/raschka

نتیجه‌گیری

> ما در واقع موفق شده‌ایم که رشته‌ی خود را به یک علم تبدیل کنیم، و آن هم به روشی بسیار ساده: صرفاً با نام‌گذاری آن به نام «علوم کامپیوتر».
>
> – دانلد کنوث [1]

این کتاب راه‌هایی را برای استفاده از علوم کامپیوتر در توانمند ساختن برنامه‌نویسان به منظور ساخت دنیای دیجیتالی که ما در آن زندگی می‌کنیم، ارائه می‌کند. شما با استفاده از این مفاهیم و کار با شبکه‌ها و علم داده به گسترش جامعه‌ای با منابع بیشتر کمک خواهید کرد.

ما سعی کردیم عمق کاوش خود را به سطح مقدماتی محدود کنیم. هدف این بود نکاتی ضروری را به شما نشان دهیم که همه‌ی کسانی که با کد کار می‌کنند باید بدانند. امیدواریم که کنجکاوی شما را برای ادامه‌ی کاوش با کتاب‌های ارائه شده در پایان هر فصل، برانگیخته باشیم. همچنین پس از این نتیجه‌گیری، یک فصل پاداش در مورد عبارات باقاعده خواهید یافت: روشی برای جستجوی الگوها که به شما در مراحل مختلف پردازش در هر فرایند تحلیل داده یا یادگیری ماشین کمک می‌کند.

می‌خواستیم موضوعات بیشتری را در این کتاب بگنجانیم، اما حجم کتاب خیلی زیاد می‌شد. به عنوان مثال، هنگام بحث در مورد یادگیری ماشین، از الگوریتم‌هایی که کامپیوترها را قادر می‌سازند دانش را بدون نظارت انسان جمع‌آوری کنند، صرف نظر کردیم. در مورد رایانش ابری، که باعث می‌شود جهان‌های جدید بسیار سریع‌تر ایجاد شوند، صحبت نکردیم. اگر به این موضوعات علاقمند هستید، با ما همراه باشید؛ ممکن است در کتاب آینده آن‌ها را پوشش دهیم! اگر این کتاب را از کد انرژی خریداری کرده‌اید، ایمیل‌های اطلاعیه‌ای از ما دریافت خواهید کرد. در غیر این صورت، می توانید برای دریافت به‌روزرسانی‌های ما در `http://code.energy/list` ثبت نام کنید.

امیدواریم این کتاب درک نظری شما را از سیستم‌های محاسباتی بهبود بخشد. اکنون، وقت آن است که از این فضا بیرون آمده و شروع به کار کنیم! یک پروژه‌ی کدنویسی جدید شروع و آنچه را که یاد گرفته‌اید تمرین کنید. همانطور که زمانی یک کدنویس عاقل گفته است:

---

[1] Donald Knuth (1938-): دانشمند علوم کامپیوتر و استاد افتخاری دانشگاه استنفورد آمریکا. شهرت او به دلیل نگارش مجموعه کتاب‌های هنر برنامه‌نویسی کامپیوتر است. وی پایه‌گذار مبحث تحلیل الگوریتم‌ها است و سیستم حروف‌چینی تِک و سیستم متافونت یا فراقلم ازجمله کارهای وی به شمار می‌روند. م.

«اگر متوجه شدید که تقریباً تمام وقت خود را صرف مباحث نظری می‌کنید، توجه خود را به چیزهای عملی معطوف کنید. این امر نظریه‌های شما را بهبود می بخشد. اگر متوجه شدید که تقریباً تمام وقت خود را صرف تمرین می‌کنید، کمی توجه خود را به موارد نظری معطوف کنید. این کار تمرین‌های شما را بهبود می‌بخشد.»

در نهایت، ارسال نظرات خود در مورد کتاب را از ما دریغ نکنید: به آدرس hi@code.energy به ما ایمیل بزنید. بازخوردی که از اولین کتابمان، مبانی و مفاهیم علوم کامپیوتر، دریافت کردیم ما را به نوشتن این کتاب ترغیب و توجه ما را به جزئیات سبک نوشتاری‌مان جلب کرد و فهمیدیم که می‌توان آن را بهبود بخشید. متشکریم!

پاداش

الگوها

عبارات باقاعده به شما امکان می دهند بر داده‌های خود مسلط شوید. آن‌ها را کنترل کنید. آن‌ها را به کار بگیرید. تسلط بر عبارات باقاعده به معنای تسلط بر داده‌ها است.

– جفری فریدل[۱]

برنامه‌نویس‌ها اغلب با داده‌هایی کار می‌کنند که با یک الگوی مشخص مطابقت دارند. سندی را در نظر بگیرید که حاوی تاریخ هایی است که به صورت «۲۷ فوریه ۲۰۱۳»، «۱۳/۲/۲۷» و «۲۷-۰۲-۲۰۱۳» نوشته شده‌اند. چگونه تمام تاریخ‌های چنین فایلی را پیدا کنیم؟ نوشتن برنامه‌هایی برای تشخیص الگوها زمان‌بر و کاملاً خسته کننده است.

خوشبختانه، ما می‌توانیم الگوهای متنی مانند قالب‌های تاریخ را با استفاده از **عبارات باقاعده**[۲] تعریف کنیم. سپس این عبارات را می‌توان توسط نرم‌افزارهای موجود تفسیر کرد، بنابراین نیازی نیست خودمان کد تطبیق الگو را بنویسیم.

در این فصل یاد خواهیم گرفت که:

- الگوهای اصلی را **مطابقت** دهید،
- با دقت الگوهای تکرارشونده را به صورت **کمّی** بیان کنید،
- الگوها را به مکان‌ها **مرتبط** کنید،
- **گروه‌هایی** از عناصررا در یک الگو به دست آورید.

---

[۱] Jeffrey Friedl (-1966): دانشمند آمریکایی علوم کامپیوتر و متخصص حوزه‌ی عبارات باقاعده . م.

[۲] Regular Expressions

شکل ۷-۱: از رئیس خود تقاضای قرار ملاقات (دیت) نکنید[۱].

به عبارات باقاعده regex یا regexp نیز گفته می‌شوند. این عبارات صرفه‌جویی زیادی در زمان به دنبال دارند که مورد علاقه اکثر برنامه‌نویسان است. به لطف کتابخانه‌ها و ابزارهای موجود، به راحتی در کد شما ادغام می‌شوند. در واقع، آن‌ها از قبل در اکثر زبان‌های برنامه‌نویسی، ویرایشگرهای کد منبع، ابزارهای خط فرمان و غیره ساخته شده‌اند.

تطابق

یک عبارت باقاعده مانند یک عبارت جستجو است. خط زیر را در نظر بگیرید:

```
District 1, Paris
```

این یک عبارت باقاعده‌ی ساده است و به عنوان یک جستجوی معمولی کار می‌کند. **موتور عبارت باقاعده**[۲] برنامه‌ای است که این عبارت را تفسیر کرده، آن را بر روی یک متن اجرا نموده و در صورت یافتن رشته‌ی «District 1, Paris»، مطابقت را گزارش می‌کند. تفاوت‌های جزئی در نحو، ویژگی‌ها و رفتارهای موتورها وجود دارد. مشتاقان فناوری به آنها **فلیور**[۳] (به معنای واقعی طعم یا مزه) می‌گویند.

بیشتر فلیورها به‌طور پیش‌فرض به حروف کوچک و بزرگ حساس هستند، بنابراین عبارت «district 1, paris» با عبارت فوق مطابقت داده نمی‌شود. خوشبختانه، این ابزارها به شما

---

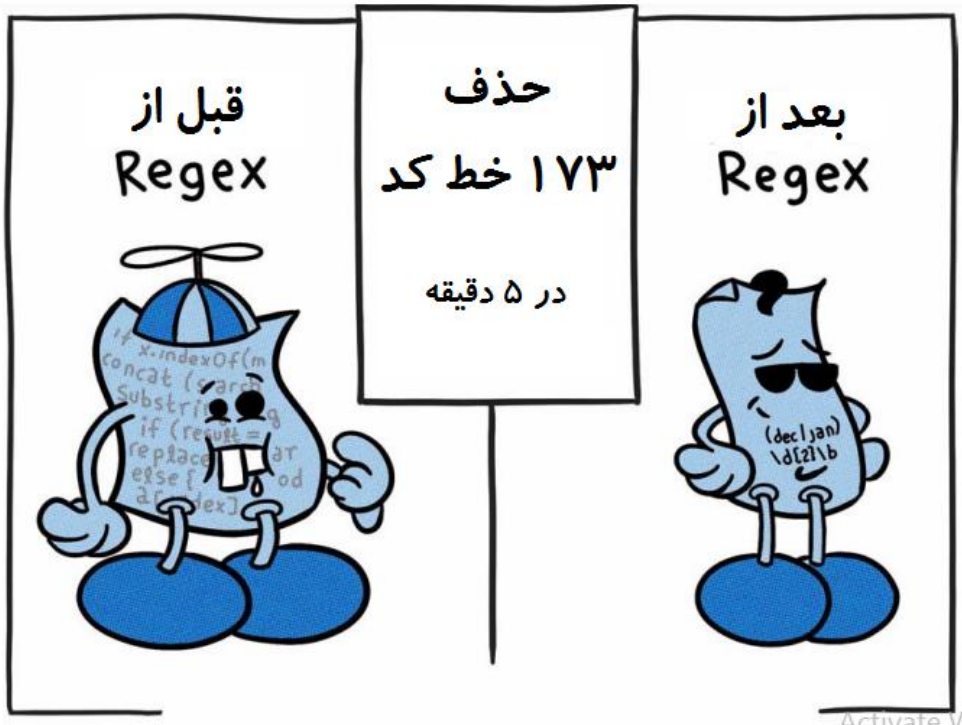
[۱] این کارتون اشاره به کلمه‌ی دیت (Date)دارد که در زبان انگلیسی هم به معنای تاریخ و هم به معنای قرار ملاقات است. م.

[۲] Regular Expression Engine

[۳] Flavor

این امکان را می‌دهند که روش تقسیم‌کردن داده‌ها را نیز با استفاده از **پرچم‌ها**[۱] پیکره‌بندی کنید. برای مثال، پرچم عدم حساسیت به حروف بزرگ و کوچک (معمولاً با علامت `i` نشان داده می‌شود) به ما امکان می‌دهد «`district 1, paris`» را در بین عبارات دارای حروف بزرگ نیز پیدا کنیم[۲].

ما *موکداً* شما را تشویق می‌کنیم که تمام عبارات باقاعده‌ی این فصل را امتحان کنید. صفحه‌ی `http://code.energy/regex` را باز کنید. در آنجا یک فیلد برای عبارت باقاعده‌ی مورد نظر خود و یک کادر متنی برای ورودی آزمایشی خواهید دید. این صفحه نشان می‌دهد که کجا عبارت با ورودی مطابقت دارد. همچنین میتوانید پرچم‌ها را نیز تنظیم کنید. این کار را امتحان و مشاهده کنید که پرچم عدم حساسیت به حروف بزرگ و کوچک چگونه بر روی تطابق‌های شما تأثیر می‌گذارد!



شکل ۷-۲: «من قبلاً یک قطعه کد بزرگ گنگ بودم. حالا من دقیق و واضح هستم، و به سادگی تاریخ‌ها را پیدا می‌کنم!».

---

[۱] `Flag`

[۲] نحوه‌ی تنظیم پرچم‌ها در فلیورها متفاوت است. پرچم‌ها گاهی بعد از اسلش رو به جلو اضافه می‌شوند، به عنوان مثال به صورت `District 1, Paris/i`.

نقطه

فرض کنید می‌خواهید هر یک از این عبارات را جستجو کنید:

- “District 1, Paris”,
- “District 2, Paris”,
- “District 3, Paris”,
- “District 4, Paris”.

به جای انجام چهار جستجو، می‌توانیم تطابق کارکتر دهم یک جستجو با هر چیزی را بررسی کنیم:

```
District ., Paris
```

در عبارات باقاعده، **نقطه** (.)[1] معنای خاصی دارد: تطابق با یک کاراکتر مشخص. عبارت ما حتی با عباراتی که آن‌ها را نمی‌خواهیم مانند عبارت «District 0, Paris» و عبارت «District !, Paris» نیز مطابقت دارد. با این وجود، نقطه راه سریعی برای یافتن متنی ارائه می‌دهد که تا حد زیادی با الگوی شما مطابقت دارد. این عبارت را در نظر بگیرید:

```
c.........y
```

این عبارت با «calligraphy»، «consistency» و «chaotically» مطابقت دارد، و از آنجایی که نقطه با کاراکتر فاصله نیز مطابقت دارد، «can be easy» و «code energy» نیز با عبارت فوق مطابقت دارند. توجه داشته باشید که مرزهای بین کلمات در اینجا در نظر گرفته نشده است، بنابراین «cisco enjoy» در «Francisco enjoyed» نیز پیدا می‌شود. بیایید ببینیم که این روش‌ها در کنار هم چگونه به ما امکان می‌دهند جستجویی فراتر از یک جستجوی ایستا انجام دهیم:

> **نام دامنه:** ایمیلی را به خاطر دارید که در آن به وب‌سایتی اشاره شده بود که به mil و کد یک کشور ختم می‌شود. چیز دیگری را به خاطر ندارید: ممکن است mil.br/، eng.mil.ru/، mil.be/frite، و غیره باشد. چگونه می‌توانید دامنه را پیدا کنید؟

همه‌ی دامنه‌های سطح بالای کد کشورها دارای دو کاراکتر هستند. نقطه با هر کاراکتر، حتی یک نقطه‌ی واقعی، مطابقت دارد. این عبارت همه مطابقت‌های ممکن را پیدا می‌کند:

---

[1] Dot

```
mil.../
```

نقطه بیشتر برای کارهای سریع و نه چندان دقیق استفاده می‌شود و معمولاً خیلی بیشتر از آنچه ما انتظار داریم مطابقت‌هایی را پیدا می‌کند. اکنون بیایید ببینیم که چگونه باید با عبارات باقاعده‌ی خاص کار کنیم.

مجموعه

عبارتی را که در داخل کروشه به صورت [•••] نوشته می‌شود **مجموعه**[۱] می‌نامیم. یک مجموعه مانند یک نقطه عمل می‌کند، با این تفاوت که مطابقت با یکی از کاراکترهای درون کروشه کفایت می‌کند. جایگزین کردن نقطه با یک مجموعه، جستجوی ما برای مناطق پاریس را دقیق‌تر می‌کند:

```
District [1234], Paris
```

این عبارت فقط با چهار موردی که ما به آن‌ها علاقمندیم مطابقت دارد. چنین طیفی از کاراکترها اغلب در مجموعه‌ها استفاده می‌شوند، بنابراین روش بیان کوتاه‌تری وجود دارد که آن‌ها را فشرده و خوانا می‌کند:

- عبارت `[1-4]` معادل عبارت `[1234]` است.
- عبارت `[h-p]` معادل عبارت `[hijklmnop]` است.

استفاده از مجموعه‌ها موجب بهبود جستجوی ما در مسئله‌ی نام دامنه می‌شود:

```
mil.[a-z][a-z]/
```

یک مجموعه می‌تواند شامل چندین محدوده باشد. به عنوان مثال، عبارت باقاعده‌ی زیر با کاراکتری مطابقت دارد که در محدوده‌ی `0-9` یا در محدوده‌ی `A-F` قرار دارد. به بیان دیگر، این عبارت با یک رقم هگز مطابقت دارد[۲]:

```
[A-F0-9]
```

---

[۱] Set

[۲] در هگز، یک رقم یک مقدار از ۰ تا ۱۵ است. از آنجایی که تعداد اعداد هندوعربی ما تمام می‌شوند، حروف `a-f` به عنوان اعداد برای مقادیر ۱۰ تا ۱۵ استفاده می‌شوند. برای اطلاعات بیشتر در مورد ارقام هگزادسیمال، به پیوست I مراجعه کنید.

با قرار دادن یک # و شش عدد از این مجموعه‌ها، می‌توانیم کدهای رنگی هگزا[۱]، مانند #E67F31 را مطابقت دهیم:

```
#[A-F0-9][A-F0-9][A-F0-9][A-F0-9][A-F0-9][A-F0-9]
```

برخی از مجموعه‌ها در عبارات باقاعده بسیار رایج هستند. برنامه‌نویسان تنبل تطبیق یک رقم و یک کاراکتر الفبایی را با **کوتاه‌نویسی** تعریف کرده‌اند:

- عبارت \d معادل عبارت [0-9] است.
- عبارت \w معادل عبارت [A-Za-z0-9_] است[۲].

برای تطبیق یک شماره تلفن ده رقمی (به عنوان مثال 307-555-0177)، باید [0-9] را ده بار تکرار کنیم. تکرار کوتاه‌نویسی راحت‌تر است:

```
\d\d\d-\d\d\d-\d\d\d\d
```

به زودی راه‌های بهتری برای تکرار مجموعه‌ها کشف خواهیم کرد.

مجموعه‌ی نفی‌شده

گاهی اوقات می‌خواهیم *تقریباً* با همه‌ی کارکترها مطابقت داشته باشیم. به جای فهرست کردن تعداد زیادی کاراکتر مورد قبول در یک مجموعه، کاراکترهایی را که مورد قبول نیستند در یک **مجموعه‌ی نفی‌شده**[۳] فهرست می‌کنیم. این کار دقیقاً برخلاف یک مجموعه است، زیرا با هر کاراکتری که در داخل

---

[۱] نماد # و به دنبال آن سه جفت رقم هگزا روشی رایج برای کدگذاری رنگ ها است. هر جفت نشان دهنده‌ی میزان نور قرمز، سبز یا آبی است که از آن ساخته شده است. به عنوان مثال، #FF0000 حداکثر نور قرمز را نشان می‌دهد، اما هیچ درجه‌ای از سبز یا آبی را نشان نمی دهد. جستجوی چنین رشته‌ای در گوگل بلافاصله رنگ مربوطه را به شما برمی‌گرداند!

[۲] توجه کنید \w شامل زیرخط است. زیرخط در نام متغیرهای کد منبع مجاز است، بنابراین برنامه نویسان می‌توانند متغیرهایی مانند beer_count داشته باشند. افزودن خط زیر به \w تطبیق نام متغیرها را در فایل‌های منبع آسان‌تر می‌کند.

[۳] The Negated Set

کروشه فهرست *نشده* است مطابقت انجام می‌گیرد. به نظر می رسد این مجموعه مانند یک مجموعه‌ی معمولی است، اما با یک علامت کارت[۱] بعد از اولین کروشه: `[•••^]`.

> **تجارت**: شما فهرستی از ارزها دارید که هر کدام به صورت کدی سه حرفی نوشته شده است، مانند (¥) JPY، (€) USD و (£) GPB. برخی از این ارزها مانند XBT (بیت کوین) و XAU (طلا) خاص هستند. موارد خاص همیشه با X شروع می‌شوند. همه‌ی کدهای ارز غیرخاص را فهرست کنید.

با استفاده از مجموعه‌ها، باید همه‌ی حرف را به صورت زیر فهرست کنیم:

```
[A-WY-Z][A-Z][A-Z]
```

تطبیق کاراکترهای غیرX با اولین کاراکتر ساده‌تر است:

```
[^X][A-Z][A-Z]
```

توجه داشته باشید که این عبارت با «0AA»، «.AA» و «AA » نیز مطابقت دارد، زیرا فهرست نفی‌شده با هر کاراکتری که در کروشه نباشد مطابقت داده می‌شود. اینکه آیا `[^X][A-Z][A-Z]` فقط با کدهای ارز واقعی مطابقت دارد یا خیر، به ورودی ما بستگی دارد. مجموعه‌های نفی‌شده همچنین برای تطبیق مواردی که انتظارشان را نداشتیم نیز مفید هستند:

> **نمونه‌خوانی**: در حین بررسی این کتاب، باید بررسی می‌کردیم که بعد از هر علامت نقطه یک فاصله قرار داشته باشد. چگونه باید این موارد را جستجو می‌کردیم؟

اگر بعد از یک علامت نقطه چیزی غیر از فاصله وجود داشته باشد، می‌توانیم آن را مطابقت داده و بررسی کنیم:

```
[.,:;?!][^ ]
```

---

[۱] Caret

آیا متوجه شده‌اید ما از یک کاراکتر نقطه در مجموعه استفاده می‌کنیم که با یک نقطه‌ی واقعی مطابقت دارد؟ دلیلش این است که کاراکترهای درون مجموعه قدرت‌های خاص خود را از دست می‌دهند. اکنون بیایید بیاموزیم که چگونه این کار را خارج از مجموعه‌ها انجام دهیم.

کاراکترهای خاص

برای اینکه یک کاراکتر خاص[۱] عملکرد `regex` خود را از دست بدهد، از کاراکتر گریز[۲] یا همان `\` استفاده می‌کنیم. برای مثال، `.\` با یک کاراکتر نقطه‌ی واقعی مطابقت دارد؛ در مورد `[.]` نیز همین طور است. مسئله‌ی نام دامنه اکنون راه‌حل دقیق‌تری دارد:

```
mil\.[a-z][a-z]/
```

کاراکترهای دیگر را نیز می‌توان با دنباله‌ی گریز بیان کرد. این کار در هنگام تطبیق کاراکترهای غیر قابل چاپ مفید است:

- عبارت `\t` با یک تب مطابق است.
- عبارت `\n` با یک خط جدید مطابق است.
- عبارت `\r` با یک سرخط مطابق است (در ویندوز خیلی مفید است).
- عبارت `\f` با یک صفحه‌ی جدید مطابق است.
- عبارت `\s` با `[ \t\r\n\f]` است.

کوتاه‌نویسی `\s` **فضای خالی** نامیده می‌شود. توجه داشته باشید که این عبارت شامل کاراکتر فضای خالی معمولی است! این امر به ما امکان می‌دهد راه‌حل مسئله‌ی نمونه‌خوانی را بهبود ببخشیم تا زمان ما را بابت بازگرداندن نقطه‌هایی که به دنبال آن‌ها خط جدید و صفحه‌ی جدید می‌آید، تلف نکند:

```
[.,:;?!][^\s]
```

با استفاده از نقطه‌ها، مجموعه‌ها، مجموعه‌های نفی‌شده، کاراکترهای خاص و کوتاه‌نویسی، اکنون می‌توانیم به طور انعطاف‌پذیر کاراکترهای مورد نظر را مطابقت دهیم. اما اگر دقیقاً ندانیم به دنبال چند کاراکتر هستیم، چه؟

---

[۱] منظور این کاراکترها است: `\ ( ) { } [ ] | $ ^ + * ? .`

[۲] Escape Character

سورها

در عبارت قبلی ما که شماره تلفن‌های ده رقمی را مطابقت می‌داد، \d تکراری و زیاد بود و به سختی خوانده می‌شد. برای ساده‌تر کردن کار، می‌توانیم از **سورهایی**[۱] استفاده کنیم که تعداد کاراکترهایی را که با یک \d مطابقت دارند تغییر می‌دهند.

آکولادها

می‌توانیم با استفاده از آکولادها تعداد دقیق کاراکترها یا مجموعه کاراکترهایی که تطابق داده می‌شوند را دقیقاً مشخص کنیم. به عنوان مثال، عبارت باقاعده‌ای شماره تلفن‌های ده رقمی را مطابقت می‌دهد، میتواند به صورت زیر بازنویسی شود:

```
\d{3}-\d{3}-\d{4}
```

این عبارت معادل \d\d\d-\d\d\d-\d\d\d\d و فقط کمی تمیزتر است و به این صورت خوانده می‌شود«تطبیق سه رقم، سپس یک خط تیره، سپس سه رقم، سپس یک خط تیره، سپس چهار رقم». یک سور را می‌توان برای بهبود عبارتی که کدهای رنگ هگزادسیمال را پیدا می‌کند نیز استفاده کرد:

```
#[A-F0-9]{6}
```

اجزای عبارت دقیقاً یک بار به صورت پیش فرض مطابقت داده می‌شوند، بنابراین {1} هیچ تأثیری ندارد: B{1}e{2}r{1} همان Beer است. همچنین، علاوه بر این الزامات ثابت تطبیق، می‌توانیم محدوده‌ای را نیز مشخص کنیم. می‌توان حداقل یا حداکثر تعداد کاراکتر منطبق را تعیین کرد:

- دقیقاً n بار: {n}.
- حداقل n بار و حداکثر m بار : {n,m}.
- حداقل n بار و بدون کران بالا : {n,}.

توجه کنید که مقدار n می‌تواند صفر نیز باشد که موجب اختیاری شدن عبارت خروجی می‌شود.

---

[۱] Quantifiers

اعداد درهم و برهم: شماره تلفن‌های برزیل دارای طول متغیر هستند. کد منطقه دو رقمی است و شماره‌ی محلی می‌تواند دارای هشت یا نه رقم باشد. همچنین خط تیره قبل از چهار رقم آخر گاهی حذف می‌شود. چگونه می‌توانیم این اعداد را در یک سند پیدا کنیم؟

می‌توانیم از یک سور بازه برای تعیین تعداد ارقام قابل قبول و از سور دیگری برای اختیاری کردن خط تیره دوم استفاده کنیم:

```
\d{2}-\d{4,5}-{0,1}\d{4}
```

این عبارت ممکن است مبهم به نظر برسد، بنابراین بیایید آن را بخش به بخش رمزگشایی کنیم:

- بخش `\d{2}` با دو رقم مطابقت می‌کند،
- بخش `-` با یک خط تیره مطابقت می‌کند،
- بخش `\d{4,5}` با چهار یا پنج رقم مطابقت می‌کند،
- بخش `-{0,1}` با صفر یا یک خط تیره مطابقت می‌کند،
- بخش `\d{4}` با چهار رقم مطابقت می‌کند.

سور اختیاری

سوری که ما برای اختیاری کردن خط تیره در آخرین عبارتمان استفاده کردیم، آنقدر رایج است که یک نمونه‌ی کوتاه‌نویسی‌شده برای آن وجود دارد. می توانیم `{0,1}` را با `?` جایگزین کنیم. به این ترتیب، عبارت مربوط به شماره تلفن‌ها به صورت زیر نوشته می‌شود.:

```
\d{2}-\d{4,5}-?\d{4}
```

نماد `?` سور اختیاری نامیده می‌شود. در واقع وجود کاراکتری که درست قبل از `?` قرار دارد اهمیت ندارد و مطابقت بدون در نظر گرفتن آن انجام می‌شود. سور اختیاری در زمان جستجوی URL در صفحات وبی که ممکن است از رمزگذاری استفاده کنند مفید است[1]:

```
https?://code\.energy
```

[1] از فصل ۳ به یاد بیاورید که در URL از `https` به جای `http` استفاده می‌شود تا نشان دهد که باید با استفاده از رمزگذاری SSL به آن دسترسی پیدا کنید.

این عبارت هر دو آدرس `http://code.energy` و `https://code.energy` را پیدا می‌کند.

سور به‌علاوه

سور **به‌علاوه** (+) یکی دیگر از کوتاه‌نویسی‌های مفید و معادل `{1,}` است، که نشان می‌دهد چیزی باید حداقل یک بار و بدون کران بالا ظاهر شود. به عنوان مثال:

```
du+de
```

این عبارت با «dude»، «duude»، «duuude» و غیره مطابقت دارد. اگر بخواهیم پیوندهای همه‌ی دامنه‌های `.com` را به دست بیاوریم[1]:

```
https?://[a-z0-9-]+\.com
```

سور ستاره

قوی ترین سور **ستاره** (*)، که معادل `{0,}` است و اثر سور اختیاری و سور به‌علاوه را ترکیب می‌کند. این سور یک عبارت اختیاری را نشان می‌دهد که می‌تواند بارها و به صورت نامحدود رخ دهد. مثلا:

```
yea*h
```

این عبارت با «yeh»، «yeah»، «yeaah»، «yeaaah» و غیره مطابقت دارد. یک نقطه و یک ستاره (`*.`) با هر چیزی مطابقت دارد: هر کاراکتری به هر تعداد بار را می‌پذیرد. اگر فکر می‌کنید ورودی شما حاوی کلمه «restaurant»، مقداری متن تصادفی و سپس کلمه‌ی «London» است، می توانید به روش زیر عمل کنید:

```
restaurant.*London
```

ترکیب `*.` هر متنی بین دو کلمه را می‌پذیرد. همچنین می‌توانید از این ترفند برای مطابقت با هر نقل قولی استفاده کنید:

```
".*"
```

حریص بودن

عبارت باقاعده‌ی اخیر (`".*"`) مشکل دارد. آن را با ورودی زیر امتحان کنید:

---

[1] این عبارت با برخی از دامنه‌های نامعتبر نیز مطابقت دارد، زیرا خط تیره در ابتدا یا انتهای نام مجاز نیست. برای حل این مشکل، به یک ویژگی پیشرفته‌ی regex به نام lookarounds نیاز داریم. می‌توانید در صفحه‌ی `http://code.energy/lookaround` چیزهای بیشتری در این رابطه بیاموزید.

```
Avaritia porro "hominem" ad quod "vis maleficium" impellit.
```

سورها حریص هستند: آن‌ها سعی می‌کنند تا حد امکان کاراکترهای بیشتری را به کار بگیرند[1]. نقطه و ستاره از اولین علامت نقل قول ورودی ما تا آخرین علامت نقل قول اجرا می‌شوند و خرجی زیر را برمی‌گردند:

```
"hominem" ad quod "vis maleficium"
```

برای مطابقت با `"hominem"`، نقطه را با یک مجموعه نفی‌شده جایگزین کنید:

```
"[^"]*"
```

به صورت پیش‌فرض، موتورهای عبارات باقاعده اولین تطابق خود را برمی‌گردانند. برای به دست آوردن چندین تطابق، یک پرچم عمومی وجود دارد که معمولاً با g نشان داده می‌شود. این پرچم باعث می‌شود آخرین جستجوی ما با `"hominem"` و `vis maleficium"` مطابقت داشته باشد. توجه داشته باشید که یک کاراکتر نمی‌تواند بخشی از دو تطابق باشد، بنابراین `"ad quod"` را بر نمی‌گرداند.

انکورها

تا اینجا، عبارات باقاعده‌ی ما می‌توانستند کاراکترها و عبارات را در هر نقطه از ورودی مطابقت دهند. **انکورها** دقیقاً *جایی* را که تطابق‌ها مجاز هستد مشخص می‌کنند. انکورها کاراکترها را مصرف نمی‌کنند، بلکه مکان‌های ممکن برای تطابق‌ها را محدود می‌کنند.

کارت

علمات **کارت** (^) هنگامی که از بیرون ازکروشه استفاده شود تطابق را به ابتدای یک خط محدود می‌کند. عبارت زیر را در نظر بگیرید:

```
^Once upon a time
```

[1] بسیاری از فلیورها نیز سورهای تنبل را به عنوان جایگزین ارائه می‌دهند. سورهای تنبل تا حد امکان کاراکترهای کمتری مصرف می‌کنند.

این عبارت فقط با شروع خطی که یک آغاز یک قصه است مطابقت دارد. بسیاری از کدنویس‌ها از دنباله‌ای از خطوط تیره (----) برای ایجاد جداکننده‌های افقی در فایل‌های متنی ساده استفاده می‌کنند. می‌توانیم آن‌ها را با این عبارت پیدا کنیم:

```
^-+
```

این عبارت با دنباله‌ای از خطوط تیره مطابقت دارد، اما فقط در صورتی که در شروع یک خط باشند.

دلار

علامت **دلار** تطابق را به انتهای یک خط محدود می‌کند. به عنوان مثال عبارت زیر را در نظر بگیرید:

```
happily ever after\.$
```

این عبارت فقط با خطی مطابقت دارد که با جمله‌ی فوق به اتمام برسد. می‌توانید از هر دو انکور در یک عبارت استفاده کنید. عبارت زیر یک افسانه را جستجو می‌کند[1]:

```
^Once upon a time.*happily ever after\.$
```

می‌توانیم عبارت خود را به روز کنیم تا جداکننده‌های ساخته‌شده از خطوط تیره را برای مطابقت با خطوطی که *فقط* حاوی خط تیره هستند پیدا کند:

```
^-+$
```

آیا می‌توانید حدس بزنید چگونه می‌توان یک خط خالی را مطابقت داد؟ نکته: شما فقط به دو انکور در عبارت خود نیاز دارید.

> **کد تمیز:** در بسیاری از زبان‌های برنامه‌نویسی، پایان دادن به یک خط کد با فاصله، بسیار ناپسند است، زیرا هیچ کاربردی ندارد. چگونه می‌توانید این «فضاهای خالی دنباله‌دار» را پیدا کنید؟

یافتن فضاهای دنباله دار ساده است، یک یا چند فاصله را در انتهای یک خط مطابقت می‌دهیم:

```
+$
```

[1] در بیشتر فلیورها، نقطه تنها در صورتی با کاراکتر خط جدید مطابقت دارد که از پرچم s استفاده شود. از این پرچم برای پیدا کردن یک افسانه‌ی پریان شامل چند پاراگراف استفاده کنید!

پرچم چندخطی: در برخی از فلیورها، کارت و دلار به ابتدا و انتهای کل ورودی اشاره دارند و شکستگی‌های خط را نادیده می‌گیرند. برای تغییر این رفتار، **پرچم چندخطی**[۱] وجود دارد که معمولاً با m نشان داده می‌شود. این پرچم کارت و دلار را مجبور می‌کند تا کاراکترهای خط جدید را تشخیص دهند.

مرز

برخی از کلمات در درون برخی دیگر گنجانده شده‌اند. به عنوان مثال، جستجوی fun با کارکترهای درون «funeral» مطابقت می‌کند، و جستجوی rude نیز «prudent» را پیدا می‌کند. به این کلمات، **کلمات کوآلا**[۲] می‌گویند. جستجوی زیر را انجام دهید:

```
art          Martial arts impart artists as art.
```

اگر فقط به دنبال کلمه «art» باشیم چه؟ انکور مرزی (\b) این مشکل را با محدود کردن تطابق‌ها به ابتدا یا انتهای یک کلمه حل می‌کند:

```
\bart        Martial arts impart artists as art.
art\b        Martial arts impart artists as art.
\bart\b      Martial arts impart artists as art.
\barts?\b    Martial arts impart artists as art.
```

در آخرین عبارت، شکل جمع یک کلمه‌ی کوآلا مورد قبول است. اگر فقط به کلماتی که با a شروع و با s ختم می‌شوند علاقه‌مند باشیم، از یک مجموعه و یک به‌علاوه یا ستاره استفاده می‌کنیم:

```
\ba[a-z]+s\b  Martial arts impart artists as art.
\ba[a-z]*s\b  Martial arts impart artists as art.
```

اجازه ندهید کوآلاها شما را گیج کنند. وضعیت عبارات خود را با انکور مشخص کنید.

گروه‌ها

---

[۱] Multiline Flag

[۲] ما این عبارت را از خودمان ساخته‌ایم.

تا کنون فقط سورهایی را دیدیم که فقط بر روی یک کاراکتر قبل از خود عمل می‌کردند. **گروه‌ها** به سورها اجازه می‌دهند که بر روی دنباله‌ای از کاراکترها، یا حتی یک عبارت کامل کار کنند. گروه‌ها با استفاده از پرانتزها و به صورت `(•••)` ساخته می‌شوند. به عنوان مثال،

```
(meta-)*analysis
```

در اینجا، بخش `meta-` در عبارت از لحاظ نظری می‌تواند صفر تا بی‌نهایت بار تطابق داشته باشد. این عبارت، مواردی مانند «`analysis`»، «`meta-analysis`» و حتی نمونه‌ای مانند «`meta-meta-analysis`» و غیره را پیدا می‌کند.

اعداد بزرگ: شما باید اعداد بزرگی را در یک گزارش طولانی پیدا کنید. هزارگان‌ها با کاما از هم جدا شده و برخی از اعداد رقم اعشاری نیز دارند. چگونه می‌توانید همه‌ی اعداد را بازیابی کنید؟

می‌توانیم تنبل باشیم و به صورت زیر عمل کنیم:

```
\b[0-9,.]+\b
```

با این حال، این عبارت می‌تواند موراد دیگری مانند «`10.12.1815`» را نیز پیدا کنید. برای بازیابی اعدادی که به صورت صحیح از ممیز اعشار و کاماها استفاده می‌کنند، می‌توانیم از گروه‌های همراه با سورها استفاده کنیم:

```
\b\d{1,3}(,\d{3})*(\.\d+)?\b
```

بیایید این عبارت را از چپ به راست تجزیه کنیم:

- بخش`\b` : تطابق باید با مرز یک کلمه شروع شود.
- بخش`\d{1,3}` : حداکث سه رقم سمت چپ.
- بخش`*(,\d{3})` : اختیاری، یک کاما و به دنبال آن سه رقم[1].
- بخش`*(,\d{3})` : اختیاری، یک نقطه و به دنبال آن حداقل یک رقم.
- بخش`\b` : تطابق باید با مرز یک کلمه تمام شود.

گروه‌های ضبط‌کننده

[1] می‌تواند به دفعات نامحدود تکرار شود تا با هزار، میلیون، میلیارد و غیره مطابقت داشته باشد.

به گروه‌ها، گروه‌های ضبط‌کننده نیز می‌ویند: موتور عبارت باقاعده متنی را که مطابقت می‌دهد به عنوان متغیرهای داخلی نگه می‌دارد. این متغیرها اغلب برای ایجاد طرح‌های پیچیده به منظور پیدا کردن و جایگزینی عبارات استفاده می‌شوند.

نقل قول‌های عجیب: یک مترجم فرانسوی متنی را به شما تحویل داده که شامل نقل‌قول‌هایی که با علامت نقل قول صاف است، "مثل این". او از شما می‌خواهد که آنها را با علامت‌های نقل قول فرانسوی جایگزین کنید، تا نقل قول‌ها «اینگونه» به نظر برسند.

شما نمی‌توانید مستقیماً کاراکتر " را پیدا و جایگزین کنید. بسته به موقعیت آن باید به یک کاراکتر «یا » تبدیل شود. اگر گروهی را به عبارت باقاعده‌ی قبلی خود اضافه کنیم که علامت نقل قول را پیدا کند، متن داخل علامت نقل قول در یک متغیر داخلی به نام \1 ذخیره می‌شود. می‌توانیم به موتور عبارت باقاعده بگوییم که رشته مطابقت داده‌شده را با متن جدیدی جایگزین کند که از این متغیر استفاده می‌کند:

```
Find:          "([^"]*)"
Replace:       « \1 »
```

اگر عبارت باقاعده دارای چندین گروه باشد، متغیرهای ضبط‌کننده‌ی بیشتری ایجاد می‌شوند. برای قالب‌بندی مجدد شماره تلفن‌های ایالات متحده که با استفاده از پرانتز و خط تیره جدا شده‌اند، می‌توانید این کار را انجام دهید:

```
Find:          (\d{3}) (\d{3}) (\d{4})
Replace:       (\1) \2-\3
```

در اینجا، \1 متنی را که با گروه اول مطابقت داشته باشد ذخیره می‌کند، \2 متن گروه دوم را ذخیره می‌کند و الی آخر. هر زمان که نیاز به انجام عملیات یافتن و جایگزینی داشتید که ورودی‌های آن ثابت نیستند، به یاد داشته باشید که از گروه‌ها استفاده کنید!

تناوب

زمانی که می‌خواهیم در بین چند عبارت یکی را مطابقت دهیم، از حالت **تناوب**[1] استفاده می‌کنیم که با | نشان داده می‌شود. این نماد شبیه به عملگر یا منطقی کار می‌کند. به عنوان مثال، برای مطابقت دادن شماره تلفن‌هایی که فقط به پورتوریکو تعلق دارند، به یکی از این دو کد منطقه نیاز داریم:

```
(787|939)-\d{3}-\d{4}
```

توجه داشته باشید که ما تناوب را در یک گروه محدود کردیم زیرا کمترین اولویت را دارد. به عبارت دیگر، با حذف پرانتز:

```
787|939-\d{3}-\d{4}
```

معادل با `(787)|(939-\d{3}-\d{4})` است.

این عبارت غلط است: ماشین این گونه می‌خواند: «مطابقت 787 یا یک شماره تلفن ده رقمی که با 939 شروع می‌شود». همیشه هنگام استفاده از تناوب در خارج از گروه مراقب باشید. بیایید مثال دیگری را بررسی کنیم.

> **حیوانات کیسه‌دار**: یک محیط‌بان در صحرای استرالیا در حال مطالعه‌ی مراحل رشد کانگوروها است. سازمان مربوطه‌ی او جمعیت آن‌ها را رصد و آمار آن را در قالب `Mon YY` ثبت می‌کند. او می‌داند که تولد کانگوروها در ماه‌های تابستان به اوج خود می‌رسد. وی چگونه می‌تواند داده‌های این آمار تابستانی را جستجو کند؟

جستجو باید به ماه‌های تابستانی نیمکره‌ی جنوبی محدود شود: دسامبر، ژانویه و فوریه:

```
(Dec|Jan|Feb) \d\d
```

اگر محیط‌بان ما در حال تحقیق در مورد یک تابستان خاص باشد، باید پرانتز را حذف کرده و سال‌ها را مشخص کند:

```
Dec 92|Jan 93|Feb 93
```

---

[1] Alternation

در اینجا، استفاده از دو گروه (92|93) (Dec|Jan|Feb) بسیار گسترده است. این عبارت با سه تابستان مطابقت دارد:

```
~~Jan 92~~, ~~Feb 92~~, Dec 92, Jan 93, Feb 93, ~~Dec 93~~
```

نتیجه‌گیری

در این فصل، نحوه‌ی جستجوی دقیق الگوهای پیچیده را آموختیم. مشاهده کردیم که چگونه عبارات باقاعده از ترکیبی از کاراکترها ساخته می‌شوند که برخی خود و برخی عملکردهای خاص را نشان می‌دهند. شما باید از این ابزار زمانی استفاده کنید که:

- باید تأیید کنید که یک ورودی با قالب مورد انتظار مطابقت دارد (به عنوان مثال شماره‌ی تلفن، تاریخ، آدرس IP، شماره کارت اعتباری)،
- دنباله‌ی دقیق کاراکترهایی را که به دنبال آن‌ها هستید نمی‌دانید، اما می‌دانید که چگونه باید ظاهر شوند،
- شما باید عملیات یافتن و جایگزینی پیچیده‌ای را انجام دهید.

اگر روی یک الگوی عمومی کار می‌کنید، مانند آدرس IP یا شماره‌ی تلفن، می‌توانید عبارات باقاعده‌ی معمولی آماده برای استفاده را به صورت آنلاین پیدا کنید. بسیاری از کتابخانه‌ها و دستورالعمل‌ها مربوط به عبارات باقاعده وجود دارند که به شما کمک می‌کنند الگوهای رایج را پیدا کنید.

هنگام طراحی عبارت خود، با چیزهای ساده که به طور کلی با آنچه جستجو می‌کنید مطابقت دارد شروع کرده و از آنجا، عبارت خود را به گونه‌ای اصلاح کنید که تاهر چه بیشتر دقیق باشد.

تقریباً همه‌ی زبان‌ها و ابزارهای برنامه‌نویسی از عبارات باقاعده پشتیبانی می‌کنند، اما مراقب باشید: اختلاف‌های ظریف زیادی در بین فلیورها وجود دارد. برخی پرانتزها را به عنوان کاراکترهای واقعی تفسیر می‌کنند و بنابراین از شما می‌خواهند که به جای (•••) از (\•••)\ برای ایجاد یک گروه استفاده کنید. برخی از موتورهای قدیمی از انکورهای خاصی مانند مرز کلمه پشتیبانی نمی‌کنند. هم‌چنین، چگونه کاراکترهای ژاپنی را مطابقت دهیم؟ مطالبی که این فصل یاد گرفتید به شما کمک می‌کنند تا موضوع عبارات باقاعده را مراجع و منابع آنلاین جستجو کرده و پاسخ سوال خود را بیابید.

اکنون یک افزونه‌ی عبارت باقادعه را در مرورگر خود نصب کنید تا بتوانید آن را در صفحات وب آزمایش کنید. علاوه بر این، نحوه‌ی استفاده از عبارات باقاعده را در ویرایشگر کد منبع خود بیابید. برای نوشتن این کتاب، ما از موردی به نام `Vim` استفاده کردیم که تمام جستجوها را از طریق عبارات باقاعده انجام می‌دهد. اگر راهی وجود دارد که عبارات باقاعده را به حالت جستجوی پیش‌فرض ویرایشگر خود تبدیل کنید، این کار را انجام دهید؛ این کار شما را مجبور می‌کند آن‌ها را تمرین کرده و در دراز مدت در زمان صرفه‌جویی کنید.

حتی اگر درگیر یک پرونده‌ی جنایی اضطراری نباشید (شکل ۷-۳)، دسترسی سریع به اطلاعات خاص برای کارآمد بودن ضروری است. هرچه با عبارات باقاعده بیشتر کار کنید، سریع‌تر می‌توانید داده‌های مورد نیاز خود را بازیابی کرده، آن‌ها را یاد بگیرید و ساعت‌ها در زمان ارزشمند کدنویسی صرفه‌جویی کنید.

شکل ۷-۳: «عبارات باقاعده» دریافت شده از http://xkcd.com.

منابع و مراجع

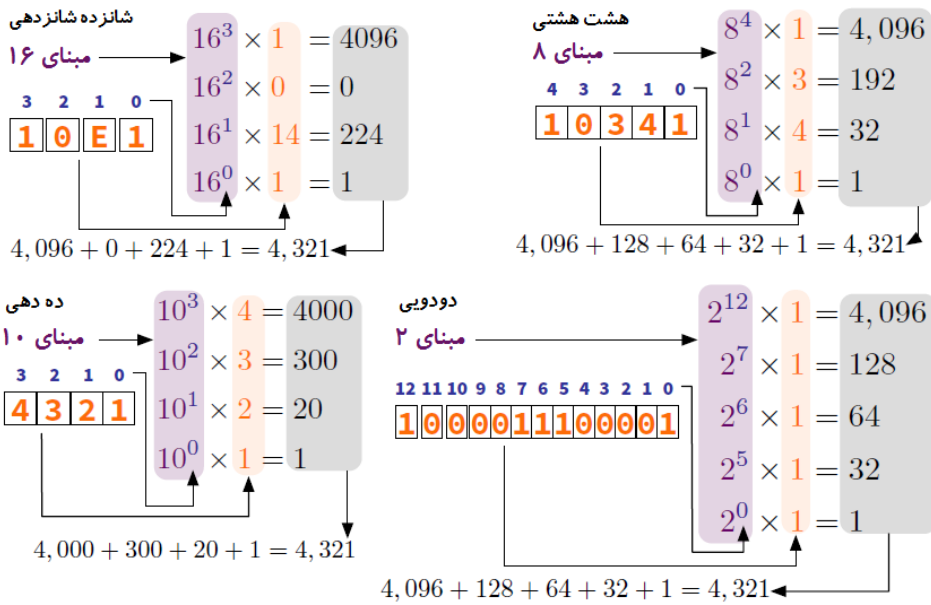
- Mastering Regular Expressions, by Friedl
  - https://code.energy/friedl
- Regular Expressions Cookbook, by Goyvaerts and Levithan
  - https://code.energy/goyvaerts

پیوست

I. مبناهای عددی

محاسبات را می‌توان به کار با اعداد کاهش داد، زیرا اطلاعات قابل‌نمایش در قالب اعداد هستند. حروف را می‌توان به اعداد نگاشت کرد، بنابراین متن را نیز می‌توان به‌صورت عددی نوشت. رنگ‌ها ترکیبی از شدت نور قرمز، آبی و سبز هستند، که می‌توان آن را با عدد نشان داد. تصاویر را می‌توان با موزاییک‌هایی از مربع‌های رنگی نمایش داد، بنابراین می‌توان آن‌ها در قالب اعداد نیز بیان کرد.

سیستم‌های عددی قدیمی (مانند اعداد رومی: I، II، III و غیره) اعداد را به‌صورت مجموع ارقام نمایش می‌دهند. سیستم عددی مورداستفاده‌ی امروزی نیز بر اساس مجموع ارقام عمل می‌کند، ولی ارزش هر رقم در موقعیت i در d به توان i ضرب می‌شود، به‌طوری‌که d تعداد ارقام منحصربه‌فرد را نشان می‌دهد. به d **مبنا**[1] گفته می‌شود. ما به‌صورت معمول از $d = 10$ استفاده می‌کنیم، زیرا ده انگشت داریم، ولی این سیستم برای هر مبنای d کار می‌کند:



شکل ۸–۱: عدد ۴,۳۲۱ در مبناهای مختلف.

[1] Base

II. شکستن رمز جابجایی

در مسئله‌ی «کد مخفی» در بخش ۳-۱، ما شما را به چالش کشیدیم تا رمز جابجایی که پیام زیر را رمزگذاری کرده بود، بشکنید:

```
MAXI KBVX HYLX VNKB MRBL XMXK GTEO BZBE TGVX VTKX EXLL
VHFF NGBV TMBH GLVH LMEB OXL
```

یک روش ساده این است که همه جابجایی‌های ممکن را در چند حرف اول متن رمز امتحان کنید:

```
LZWH JAUW    KYVG IZTV    JXUF HYSU       NBYJ LCWY
   ↑↑↑          ↑↑↑          ↑↑↑             ↑↑↑
    1            2            3      ...      25
MAXI KBVX    MAXI KBVX    MAXI KBVX       MAXI KBVX
```

این یک راه مناسب برای یافتن راه حل است، اما با یک قلم و کاغذ می‌توانیم کمی سریع‌تر پیش برویم. ابتدا حرف اول متن رمز شده را بردارید و بقیه‌ی حروف الفبا را به عقب بنویسید، مانند:

```
  1 2 3 . . . . . . . . . . . . . . . . . . . . . . . . 25
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
```

توجه داشته باشید که پس از رسیدن به حرف، از آخرین حرف الفبا به شمارش ادامه می‌دهیم: X، Y، Z و الی آخر. بیایید خطوط بیشتری از الفبای معکوس را اضافه کنیم و هر بار با حرف بعدی متن رمز شروع کنیم:

```
  1 2 3 . . . . . . . . . . . . . . . . . . . . . . . . 25
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
A Z Y X W V U T S R Q P O N M L K J I H G F E D C B

  1 2 3 . . . . . . . . . . . . . . . . . . . . . . . . 25
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
A Z Y X W V U T S R Q P O N M L K J I H G F E D C B
X W V U T S R Q P O N M L K J I H G F E D C B A Z Y

  1 2 3 . . . . . . . . . . . . . . . . . . . . . . . . 25
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
A Z Y X W V U T S R Q P O N M L K J I H G F E D C B
X W V U T S R Q P O N M L K J I H G F E D C B A Z Y
I H G F E D C B A Z Y X W V U T S R Q P O N M L K J
```

آیا می‌توانید یک الگو را در یک ستون مشاهده کنید؟ اجازه دهید چند خط دیگر را ادامه دهیم:

```
  1 2 3 . . . . . . . . . . . . . . . . . . 19 . . . . .25
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
A Z Y X W V U T S R Q P O N M L K J I H G F E D C B
X W V U T S R Q P O N M L K J I H G F E D C B A Z Y
I H G F E D C B A Z Y X W V U T S R Q P O N M L K J
K J I H G F E D C B A Z Y X W V U T S R Q P O N M L
B A Z Y X W V U T S R Q P O N M L K J I H G F E D C
V U T S R Q P O N M L K J I H G F E D C B A Z Y X W
X W V U T S R Q P O N M L K J I H G F E D C B A Z Y
```

ستون ۱۹ به صورت THE PRICE است. از آنجایی که تمام ستون‌های دیگر گنگ هستند، می‌دانیم که کلید خود را پیدا کرده‌ایم:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
                  19
                  ↓↓↓↓
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

حال می‌توانیم کل متن را رمزگشایی کنیم:

```
THEP RICE OFSE CURI TYIS ETER NALV IGIL ANCE CARE LESS
COMM UNIC ATIO NSCO STLI VES
                  ↑↑↑↑
                  19
MAXI KBVX HYLX VNKB MRBL XMXK GTEO BZBE TGVX VTKX EXLL
VHFF NGBV TMBH GLVH LMEB OXL
```

در نهایت با چیدمان مجدد فاصله‌ها و افزودن علائم نگارشی متن اصلی را پیدا می‌کنیم:

```
THE PRICE OF SECURITY IS ETERNAL VIGILANCE.
CARELESS COMMUNICATIONS COST LIVES.
```

III. شکستن رمز جایگزینی

در مسئله‌ی «سکه‌ی توخالی» در بخش ۳-۱، ما شما را به چالش کشیدیم تا رمز جایگزینی که پیام زیر را رمزگذاری کرده بود، بشکنید:

```
DUA KVYBVHA PVJ OAZQMASAO DW CWES PQLA KASJWFVZZC.
AMASCDUQFH QJ VZZ SQHUD PQDU DUA LVGQZC.
PA PQJU CWE JEYYAJJ. HSAADQFHJ LSWG DUA YWGSVOAJ.
```

بیایید با اصلی‌ترین نتایج تحلیل فرکانس شروع کنیم: در زبان انگلیسی رایج‌ترین حرف E و رایج‌ترین جفت حروف متوالی TH است. از آنجایی که رایج‌ترین حرف متن رمز A و رایج‌ترین جفت حروف DU است، می توانیم نگاشت‌های A → E و T,H → D,U را امتحان کنیم:

```
THE KVYBVHE PVJ OEZQMESEO TW CWES PQLE KESJWFVZZC.
EMESCTHQFH QJ VZZ SQHHT PQTH THE LVGQZC.
PE PQJH CWE JEYYEJJ. HSEETQFHJ LSWG THE YWGSVOEJ.
```

کار ما با این واقعیت آسان‌تر می‌شود که متن رمز، فاصله‌ها و نقطه‌گذاری‌های متن ساده را حفظ کرده است. بلافاصله می‌بینیم که نگاشت آزمایشی ما T,H,E → D,U,A شانس خوبی برای درست بودن دارد، زیرا کلمه رایج THE سه بار ظاهر شده است!

بر اساس این فرض، جمله‌ای وجود دارد که با PE شروع میشود، که می تواند معادل «be»، «he» یا «we» باشد. از قبل نگاشتی برای H داریم، بنابراین B → P و W → P را امتحان می‌کنیم. اولی ما را به بن‌بست می‌رساند: جمله با عبارت BE BQJH شروع می‌شود و تنها در صورتی می‌تواند معنا داشته باشد که معادل «be both» باشد. این نکته با فرض قبلی ما ما T → D تضاد دارد. بنابراین فرض دیگر یعنی W → P را حفظ می‌کنیم:

```
THE KVYBVHE WVJ OEZQMESEO TW CWES WQLE KESJWFVZZC.
EMESCTHQFH QJ VZZ SQHHT WQTH THE LVGQZC.
WE WQJH CWE JEYYEJJ. HSEETQFHJ LSWG THE YWGSVOEJ.
```

بیایید نگاهی به کلماتی بیندازیم که کمترین حروف گم شده را دارند. به عنوان مثال، WQTH بلافاصله I → Q را به ما می دهد. با این نگاشت جدید، WQJH تبدیل به WIJH می شود که به نوبه خود به ما S → J می‌دهد:

```
THE KVYBVHE WVS OEZIMESEO TW CWES WILE KESSWFVZZC.
EMESCTHIFH IS VZZ SIHHT WITH THE LVGIZC.
WE WISH CWE SEYYESS. HSEETIFHS LSWG THE YWGSVOES.
```

دوباره، به دنبال کلماتی می‌گردیم که کمترین حروف را دارند: **WVS** به ما V → A و **TW** بـه مـا W → O را می‌دهد. علاوه بر این، این واقعیت که یک کاراکتر ناشناخته دو بار در یک کلمه ظاهر می‌شود نیز می‌تواند یک سرنخ باشد: **SEYYESS** و به ما E,Y → U,C و HS**EETI**FHS به ما G,R,N → H,S,F را می‌دهد:

THE K**AC**B**AGE WAS** O**EZIMERE**O **TO** C**OUR WILE** K**ERSWNA**ZZC.
EMERCTHING IS AZZ RIGHT WITH THE L**AGIZC.**
WE WISH C**OU SUCCESS. GREETINGS** L**RO**G **THE CO**G**RA**O**ES.**

هر چه به راه‌حل نزدیک‌تر می‌شویم، سریع‌تر پیش می‌رویم! **COUR** و **COU** به ما C → Y را می‌دهند، هم‌چنین **EMERYTHING** نیز به ما M → V را می‌دهد، و **EVERYTHING IS AZZ** به ما Z → L را می‌دهد.

THE K**AC**B**AGE WAS** O**ELIVERE**O **TO YOUR WILE** K**ERSWNALLY.**
EVERYTHING IS ALL RIGHT WITH THE L**AGILY.**
WE WISH YOU SUCCESS. GREETINGS L**RO**G **THE CO**G**RA**O**ES.**

تقریبا به پایان رسیدیم! نگاشت‌های از دست رفته را می‌توان به سرعت به صورت زیر P,K,D,F,O,M → K,B,O,L,W,G پیدا کرد و متن ساده راز دیگری در خود ندارد:

THE PACKAGE WAS DELIVERED TO YOUR WIFE PERSONALLY.
EVERYTHING IS ALL RIGHT WITH THE FAMILY.
WE WISH YOU SUCCESS. GREETINGS FROM THE COMRADES.

در نهایت، می‌توانیم آنچه را که از کلید رمزگذاری می‌دانیم بنویسیم، تا در صورتی که متن رمزی جدیدی پیدا کردیم که از آن استفاده می‌کند، دوباره به کمک آن رمز را بشکنیم. حروف از دست رفته آن‌هایی هستند که در پیام ظاهر نشده‌اند:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
E K Y T U N M G _ S P F V _ D W I _ R _ H A O _ C L
```

همانطور که می‌بینید، این رمز جایگزینی ساده چندان امن نیست، زیرا ما توانستیم آن را در چند مرحله‌ی ساده بشکنیم. اگرچه این رمزها زمانی که وضعیت فضاها ناشناخته است چالش برانگیزتر هستند، کامپیوترها بدون توجه به آن می‌توانند سریعاً آن‌ها را بشکنند. برای امتحان کردن آن، به `http://code.energy/substitution` سر بزنید.

IV. ارزیابی طبقه‌بندی‌کننده‌ها

در فصل ۵، با مدل‌های پیش‌بینی آشنا شدیم: الگوریتم‌هایی که می‌توانند با دریافت داده‌های مربوط به هر پیش‌بینی به نام **X**، دنباله‌ای از پیش‌بینی‌ها انجام دهندکه آنها را **y** می نامیم. دو نوع مدل وجود دارد. رگرسورها پیش‌بینی‌های عددی انجام می‌دهند، در حالی که طبقه‌بندی کننده‌ها نتیجه‌ای را پیش‌بینی می‌کنند که به گروه‌های برچسب‌گذاری‌شده تقسیم می‌شود. پیشتر نحوه‌ی امتیازدهی به کیفیت یک رگرسور پیش‌بینی‌کننده را یاد گرفتیم. بیایید اکنون یاد بگیریم چگونه به طبقه‌بندی‌کننده‌ها نیز امتیاز دهیم.

از آنجایی که طبقه‌بندی‌کننده‌ها برچسب‌ها را پیش‌بینی می‌کنند، اجازه دهید ساده شروع کنیم: فرض کنید **y** از داده‌های دودویی ساخته شده است. از آنجایی که فقط *دو* برچسب دارد، تنها چیزی که نیاز داریم یک پاسخ بله یا خیر برای هر سطر است. ما به این مدل **طبقه‌بندی دودویی**[۱] می‌گوییم. پیش‌بینی اینکه آیا بیمار مبتلا به دیابت می‌شود یا اینکه خرید با کارت اعتباری تقلبی انجام شده است، نمونه‌هایی از طبقه‌بندی دودویی هستند.

یک رویکرد طبیعی ارزیابی، محاسبه‌ی درصد پیش‌بینی‌های صحیح است که به آن **صحت**[۲] می‌گویند. اما، این روش اغلب یک روش بد برای ارزیابی عملکرد یک مدل است. با هم ببینیم چرا:

> **پیدا کردن تقلب:** شما تکنسین بانکی هستید که سعی می‌کند از تقلب در کارت اعتباری جلوگیری کند. آن‌ها از شما خواستند مدلی ایجاد کنید که بر اساس جزئیات تراکنش، تقلبی بودن پرداخت را پیش‌بینی کند. شما دو مدل، یک مدل خونسرد و یک مدل حساس را برای پیش‌بینی تقلب در داده‌های موجود از ۱۰۰۰ تراکنش، که ۱۰ مورد از آن‌ها موارد تقلبی هستند، آموزش داده‌اید. اکنون می‌خواهید عملکرد آن‌ها را با یک مدل تنبل آموزش‌ندیده مقایسه کنید که به طور تصادفی ۱٪ از تراکنش‌ها را تقلبی می‌خواند. در اینجا نحوه‌ی عملکرد هر یک از این سه مدل آمده است:

---

۱ Binary Classification
۲ Accuracy

- مدل تنبل هیچ تقلبی را صحیح شناسایی نمی‌کند و ۱۰ مورد را هم به اشتباه تقلبی می‌داند.
- مدل خونسرد ۸ تقلب را شناسایی کرده و ۱۹ مورد را هم به اشتباه تقلبی می‌داند.
- مدل حساس همه‌ی موارد تقلب را شناسایی کرده و ۸۷ مورد دیگر را نیز به اشتباه تقلب می‌داند.

کدام مدل از همه صحیح‌تر است؟

مدل حساس هیچ تقلبی را از دست نداده است، بنابراین 87=87+0 اشتباه و 913 پیش‌بینی درست انجام داده و میزان صحت 91.3% را به دست آورده است.

مدل خونسرد دو مورد تقلب را از دست داده، بنابراین 21=19+2 اشتباه و 979 پیش‌بینی درست انجام داده و میزان صحت 97.9% را به دست آورده است. از سوی دیگر، مدل تنبل هر ده تقلب را از دست داده است، بنابراین 20=10+10 اشتباه و 980 پیش‌بینی درست انجام داده و میزان صحت 98% را به دست آورده است. مدل تنبل از همه صحیح‌تر است، حتی اگر بلااستفاده باشد!

برای ارزیابی بهتر طبقه‌بندی‌کننده‌ها، باید ببینیم که وضعیت پیش‌بینی‌های درست و غلط آن‌ها چگونه است. برای این منظور، ما یک **جدول درهم‌ریختگی**[۱] ایجاد می‌کنیم که پیش‌بینی‌های صحیح و نادرست را برای هر برچسب شمارش می‌کند (شکل ۸-۲).

وقتی که برچسب پیش‌بینی‌شده و برچسب واقعی متفاوت باشند، پیش‌بینی نادرست است، این موارد در شکل ۸-۲ به رنگ قرمز نشان داده شده‌اند. توجه کنید که می‌توان میزان صحتی را که پیشتر در مورد آن صحبت شد برای هر جدول با تقسیم مجموع خانه‌های سیاه بر ۱۰۰۰ به دست آورد. اکنون ببینیم از چه معیارهای دیگری می‌توانیم برای درک جداول درهم‌ریختگی استفاده کنیم.

حساسیت: درصد تقلبی که یک مدل قادر به شناسایی آن است، **امتیاز حساسیت**[۲] آن مدل را مشخص می‌کند. به عنوان مثال:

- مدل تنبل صفر از ۱۰ تقلب را شناسایی می‌کند، پس حساسیت آن 0% است.
- مدل خونسرد ۸ از ۱۰ تقلب را شناسایی می‌کند، پس حساسیت آن 80% است.
- مدل حساس ۱۰ از ۱۰ تقلب را شناسایی می‌کند، پس حساسیت آن 100% است.

---

[۱] Confusion Table
[۲] Sensitivity Score

تنبل

		Predicted	
		Fraud (10)	**No Fraud** (990)
Truth	**Fraud** (10)	0	**10**
	No Fraud (990)	**10**	980

خونسرد

		Predicted	
		Fraud (27)	**No Fraud** (973)
Truth	**Fraud** (10)	8	**2**
	No Fraud (990)	**19**	971

حساس

		Predicted	
		Fraud (97)	**No Fraud** (903)
Truth	**Fraud** (10)	10	**0**
	No Fraud (990)	**87**	903

دقیق

		Predicted	
		Fraud (10)	**No Fraud** (990)
Truth	**Fraud** (10)	10	**0**
	No Fraud (990)	**0**	990

شکل ۸-۲: جداول درهم‌ریختگی برای طبقه‌بندی‌کننده های تنبل، خونسرد و حساس. یک طبقه‌بندی‌کننده‌ی دقیق با دقت صد در صد به عنوان مرجع آورده شده است.

در اینجا، بهترین امتیاز حساسیت توسط مدل حساس به دست می‌آید، زیرا قادر به شناسایی لیست کامل تقلب‌ها بود. از سوی دیگر، مدل خونسرد بخش عمده‌ای از تقلب‌ها را پیدا کرد، اما لیست ناقص بود. مدل تنبل کمترین حساسیت را داشت، زیرا هیچ چیز جالبی پیدا نکرد!

دقت: درصد پیش‌بینی‌های درست یک مدل از تقلب، **امتیاز دقت**[۱] است. به عنوان مثال:

- مدل تنبل صفر از ۱۰ پیش‌بینی درست انجام می‌دهد، پس دقت آن 0% است.
- مدل خونسرد ۸ از ۲۷ پیش‌بینی درست انجام می‌دهد، پس دقت آن تقریباً 29.6% است.
- مدل حساس ۱۰ از ۹۷ پیش‌بینی درست انجام می‌دهد، پس دقت آن تقریباً 10.3% است.

بهترین امتیاز دقت توسط مدل خونسرد به دست آمد، زیرا تقریباً یک سوم از تقلب‌های پیش‌بینی شده تقلب واقعی بودند. مدل حساس دقت کمتری دارد: فقط ۱۰٪ از پیش‌بینی‌های این مدل درست هستند. در نهایت، مدل تنبل کمترین دقت را داشت، زیرا تمام پیش‌بینی‌های آن بی‌فایده هستند!

---

[۱] Precision Score

اکنون مشخص است که مدل تنبل وحشتناک است. با این حال، انتخاب بین مدل‌های خونسرد و حساس سخت است، زیرا یکی حساسیت بهتر و دیگری دقت بالاتری دارد. خوشبختانه، یک معیار رایج وجود دارد که این مشکل را حل می‌کند.

امتیاز F_1: یک ارزیابی خوب از عملکرد یک مدل، هم حساسیت و هم دقت را در نظر می‌گیرد. اگر هر کدام صفر باشند، مدل قدرت پیش‌بینی ندارد، بنابراین نمره‌ی نهایی آن نیز باید برابر با صفر باشد. برای رسیدن به این هدف، دانشمندان نوع خاصی از میانگین (به نام میانگین هارمونیک) را بر روی امتیازهای حساسیت و دقت محاسبه به کار گرفته‌اند. آنها به این **امتیاز F_1** می‌گویند:

$$F_1 = 2 \times \frac{P \times S}{P + S}$$

به طوری که P مقدار دقت و S مقدار حساسیت است.

امتیاز F_1 از ۰ تا ۱ (یا ۱۰۰٪) متغیر است. با اضافه کردن امتیازهای دقت و حساسیت در فرمول برای هر مدل، متوجه می‌شویم:

- برای مدل تنبل: F_1=0%.
- برای مدل خونسرد: F_1=43%.
- برای مدل حساس: F_1=19%.

بر اساس این معیار، مدل خونسرد بیش از دو برابر مدل حساس قدرت پیش‌بینی دارد. این امر عمدتاً از دقت پایین مدل حساس ناشی می‌شود.

هشدارهای کاذب: از دیدگاه مشتری، یک معیار مهم دیگر وجود دارد: چند بار معاملات صحیح من رد می‌شود؟ درصدی از تراکنش‌های واقعاً درست که به اشتباه به عنوان تقلب طبقه‌بندی شده‌اند، **نرخ هشدار کاذب**[۱] نامیده می‌شود که به عنوان **نرخ مثبت کاذب**[۲] نیز شناخته می‌شود. برای مثال:

- مدل تنبل ۱۰ از ۹۹۰ تراکنش صحیح را رد می‌کند، FAR=1.0%.
- مدل خونسرد ۱۹ از ۹۹۰ تراکنش صحیح را رد می‌کند، FAR=1.9%.
- مدل حساس ۸۷ از ۹۹۰ تراکنش صحیح را رد می‌کند، FAR=8.8%.

---

۱ False Alarm Rate
۲ False Positive Rate

در اینجا، مدل حساس با به صدا درآوردن بسیاری از هشدارهای نادرست، خود را از سایرین متمایز می‌کند: تقریباً از هر ده معامله‌ی صحیح یک مورد به عنوان تقلب طبقه‌بندی می‌شود. بسته به کاربرد، از نرخ هشدار کاذب به جای دقت برای تعیین میزان سودمندی یک مدل استفاده می‌شود.

هنگام ارزیابی مدل خود، سعی کنید آنچه را که محاسبه می‌کنید در یک جمله فرموله کنید و نه فقط با ریاضیات. در مثال ما، می‌توان گفت: «مدل حساس ۸۸ درصد از تراکنش‌های مشتریان صادق را مسدود کرده است». این می‌تواند به شما کمک کند تا انتخاب کنید کدام معیارها در موقعیت خاص برای شما بیشترین اهمیت را دارند.

ایجاد تعادل در طبقه‌بندی

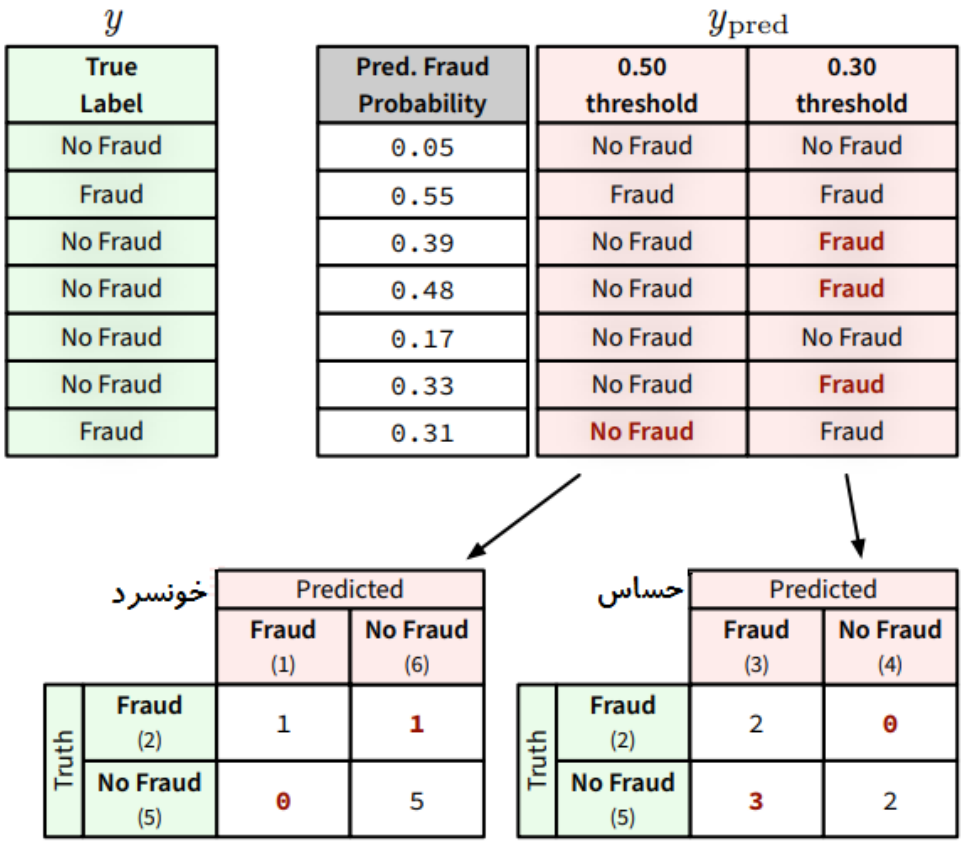
تابع پیش‌بینی طبقه‌بندی‌کننده برچسبی را به عنوان خروجی می‌دهد که احتمال بیشتری برای آن قائل است. علاوه بر این، بیشتر طبقه‌بندی‌کننده‌ها می‌توانند تخمین خود را از احتمال تعلق یک سطر به یک برچسب ارائه دهند. به این ترتیب، می‌توانیم منطق تخصیص برچسب در یک مدل را برای پیش‌بینی بیشتر یک برچسب خاص تغییر دهیم. این روش به ما اجازه می‌دهد تا دو مدل به ظاهر بسیار متفاوت را از یک موتور واحد به دست آوریم (شکل ۸-۳).

با تعیین آستانه‌ی پایین تر، مدل نسبت به سرنخ‌های تقلب حساس‌تر است، بنابراین حساسیت افزایش می‌یابد. از سوی دیگر، خریدهای بیشتری به اشتباه به عنوان جعلی طبقه‌بندی می‌شوند و دقت را کم می‌کند.

از آنجایی که امتیارهای دقت، حساسیت و F_1 بر اساس آستانه تغییر می‌کنند، یک اندازه‌گیری واحد از این امتیازات برای ارزیابی جامع طبقه بندی کننده کافی نیست. در نمونه‌ی کوچک ما، آستانه‌ی بالاتر منجر به $F_1 \approx 0.67$ می‌شود، در حالی که آستانه‌ی پایین به $F_1 \approx 0.57$ منجر می‌شود. با این حال، این امر لزوماً نشان نمی‌دهد که آستانه‌ی بالاتر ارجح است.

روش ارزیابی و آستانه‌ی طبقه‌بندی خود را با توجه به نیازهای دنیای واقعی انتخاب کنید. اگر مزیت شناسایی تقلب زیاد و جنبه‌ی منفی طبقه‌بندی نادرست خریدهای معتبر کم باشد، آستانه‌ی بالاتر ممکن است بهتر باشد.

به طور کلی، ما سعی می‌کنیم حداقل امتیازات حساسیت و دقت قابل قبول را سریع ارزیابی کنیم. اگر افراد دیگری قرار است از طبقه‌بندی‌کننده‌ی ما استفاده کنند، باید در فرایند ایجاد تعادل به منظور تنظیم آستانه کمک کنند.



شکل ۸-۳: طبقه‌بندی با دو مقدار آستانه‌ی متفاوت.

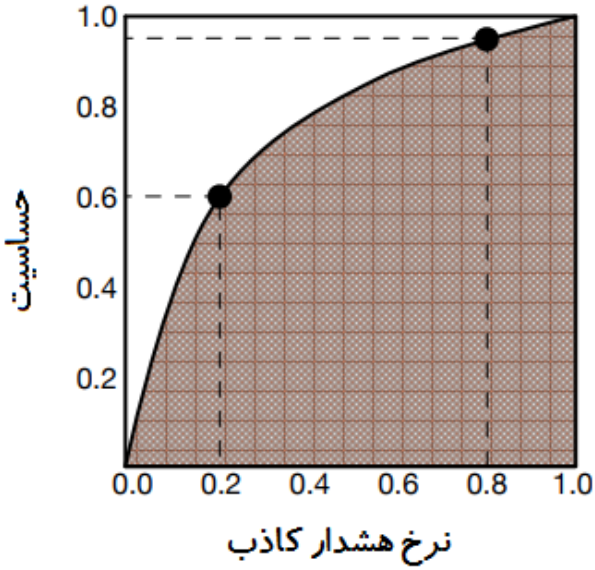
منحنی ROC

رادار هشدار سریع یک فناوری کاملاً جدید در آغاز جنگ جهانی دوم بود. رادارها گاهی اوقات سیگنال‌هایی را دریافت می‌کردند که اپراتورها را متحیر می‌کرد. آن‌ها همیشه مطمئن نبودند که سیگنال‌های ضعیف از هواپیماهای دشمن می‌آید یا از پارازیت‌های رادیویی، با این حال باید تصمیم می‌گرفتند که زنگ هشدار را به صدا دربیاورند یا خیر. هشدار اشتباه منابع گرانبها را هدر می‌داد، اما نادیده گرفتن یک تهدید واقعی می‌توانست عواقب ویرانگری داشته باشد.

در حین تلاش برای تعیین اینکه کدام سیگنال‌های رادیویی تهدید واقعی هستند، یک اپراتور رادار به شبیه به یک طبقه‌بندی‌کننده کار می‌کرد. معیارهای مورد استفاده برای امتیاز دادن به طبقه‌بندی‌کننده‌ها برای سنجش عملکرد رادار نیز کار می‌کنند: حساسیت درصدی از هواپیماهای ورودی دشمن است که

زنگ هشدار را به صدا در آورده‌اند، و نرخ هشدار کاذب درصد سیگنال‌های رادیویی غیرتهدیدکننده‌ای است که زنگ هشدار را فعال کرده‌اند.

اپراتورهای رادار به سرعت متوجه شدند که بین حساسیت و نرخ هشدار کاذب تعادلی وجود دارد. حساسیت را می‌توان با ثبت سیگنال‌های ضعیف‌تر در گیرنده‌های رادیویی افزایش داد، اما انجام این کار باعث می‌شود هشدارهای کاذب بیشتری ایجاد شوند. برای اینکه تصمیم بگیرند گیرنده‌های رادیویی خود چقدر باید حساس باشند، تصمیم گرفتند داده‌ها را بر اساس تنظیمات مختلف جمع‌آوری و امتیازهای حاصل را ترسیم کنند:



شکل ۸-۴: اگر این رادار قدیمی برای شناسایی ۹۵ درصد هواپیماهای دشمن تنظیم شده بود، حدود ۸۰ درصد از سیگنال‌های رادیویی غیرتهدیدکننده به هشدارهای کاذب منجر می‌شدند. حساسیت کمتر گیرنده، نرخ هشدار کاذب را به ۲۰ درصد کاهش داد، اما با انجام این کار، تنها ۶۰ درصد از هواپیماهای ورودی قابل شناسایی بودند.

آنها این منحنی را منحنی **مشخصه عملکرد گیرنده**[۱] یا **منحنی ROC** نامیدند. رادارها سپس بهبود یافتند و حساسیت بالاتری را برای همان نرخ هشدار کاذب ارائه کردند و منحنی‌های ROC بر روی نمودارهایشان به سمت بالا حرکت کردند. به طور کلی، هرچه فناوری رادار بهتر باشد، ناحیه‌ی سایه‌دار زیر منحنی (امتیاز AUC) که در شکل ۸-۴ نشان داده شده است بزرگتر خواهد بود.

---

[۱] Receiver Operating Characteristic

شما می‌توانید با محاسبه‌ی حساسیت و نرخ مثبت کاذب برای تمام آستانه‌های ممکن طبقه‌بندی، منحنی ROC را برای طبقه‌بندی‌کننده‌ی خود رسم کنید. منحنی می‌تواند به شما کمک کند تصمیم بگیرید که از کدام آستانه استفاده کنید.

همچنین، امتیاز AUC معیار خوبی برای عملکرد کلی طبقه‌بندی‌کننده است. این امتیاز به عنوان یک ارزیابی عمومی از قدرت پیش‌بینی مفید است، به خصوص اگر هنوز آستانه‌ای را که برای طبقه بندی‌کننده استفاده می‌کنید انتخاب نکرده باشید.

طبقه‌بندی چندطبقه‌ای

امتیازهای طبقه‌بندی‌کننده‌های دودویی را می‌توان برای طبقه‌بندی‌کننده‌های دارای بیش از دو برچسب نیز تطبیق داد. برای مثال، فرض کنید باید حدس بزنید که آیا فردی در ایالات متحده، مکزیک یا کانادا متولد شده است. شما یک طبقه‌بندی‌کننده را آموزش می‌دهید، و سپس آن را با ۱۰۰ نفر امتحان می‌کنید: ۶۵ آمریکایی، ۲۵ مکزیکی، و ۱۰ کانادایی. نتایج پیش‌بینی را می‌توان در یک جدول درهم‌ریختگی ۳ در ۳ خلاصه کرد، بنابراین می‌توانیم وضعیت درست یا غلط بودن پیش‌بینی‌ها برای هر برچسب را مشاهده کنیم:

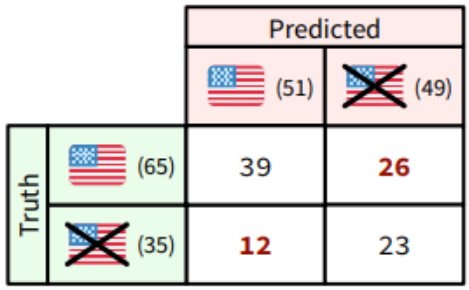
		Predicted		
		(51)	(37)	(12)
Truth	(65)	39	24	2
	(25)	11	13	1
	(10)	1	0	9

شکل ۸-۵: جدول درهم‌ریختگی یک طبقه‌کننده‌ی چندطبقه‌ای.

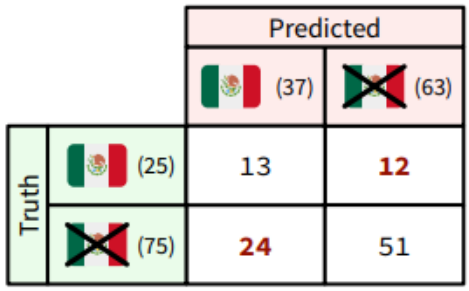
جدول نشان می دهد که مدل در کجا اشتباه کرده است. این مدل در تمایز قائل شدن بین آمریکا و مکزیک مشکل دارد : آمریکایی‌ها ۲۴ بار به اشتباه به عنوان مکزیکی و مکزیکی‌ها به اشتباه ۱۱ بار به عنوان آمریکایی طبقه بندی شده‌اند. این امر نشان می‌دهد که ویژگی‌های بیشتری با سرنخ‌هایی برای تشخیص آمریکایی‌ها از مکزیکی‌ها باید اضافه شوند.

قطر حاوی اعداد سیاه حدس‌های صحیح را نشان می‌دهد. مدل 39+13+9=61 بار از 100 درست پیش‌بینی کرده است، بنابراین دقت آن 61% است. همانطور که دیدیم، صحت می‌تواند

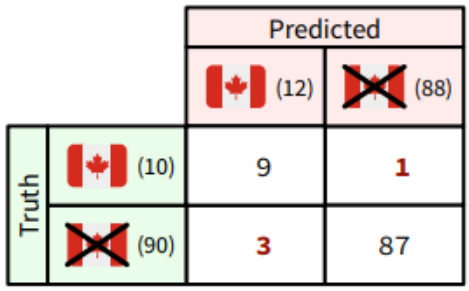
گمراه‌کننده باشد، و ما باید از معیارهایی مانند حساسیت و دقت استفاده کنیم تا بتوانیم قدرت پیش‌بینی واقعی یک مدل را به دست آوریم. از آنجایی که این معیارها برای طبقه‌بندی دودویی تعریف شده‌اند، باید طبقه‌بندی‌کننده‌ی خود را به صورت ترکیبی از طبقه‌بندی‌کننده‌های دودویی دوباره تعریف کنیم:



$$\text{sensitivity} = 60\%,$$
$$\text{precision} = 76\%,$$
$$F_1 = 67\%.$$



$$\text{sensitivity} = 52\%,$$
$$\text{precision} = 35\%,$$
$$F_1 = 42\%.$$



$$\text{sensitivity} = 90\%,$$
$$\text{precision} = 75\%,$$
$$F_1 = 82\%.$$

شکل ۸-۶: یک طبقه‌بندی‌کننده‌ی چندطبقه‌ای در قالب سه طبقه‌بندی‌کننده‌ی دودویی دیده می‌شود.

برای هر برچسب، یک جدول درهم‌ریختگی ساخته می‌شود که در آن همه‌ی برچسب‌های دیگر در یک گروه جا گرفته‌اند. به عنوان مثال، اولین جدول درهم‌ریختگی نشان می‌دهد که چگونه این مدل افراد را به عنوان آمریکایی یا غیرآمریکایی طبقه‌بندی می‌کند[1]. با این ترفند می‌توانیم دقت، حساسیت و F_1 را برای هر برچسب محاسبه کنیم.

[1] اگر خروجی y_{pred} به صورت یک-داغ کدگذاری شده باشد، هر یک از این جداول درهم‌ریختگی مربوط به یکی از ستون‌های کدگذاری شده یک-داغ است.

برای به دست آوردن یک امتیاز واحد برای مدل، میانگین سه امتیاز F_1 مجزا را محاسبه می‌کنیم که برابر با 0.64 است. این امتیاز، **امتیاز ماکرو F_1**[۱] نامیده می‌شود. با این حال، این امتیاز می‌تواند ناعادلانه باشد: برچسب کانادا به اندازه‌ی برچسب آمریکا به امتیاز ماکرو کمک می‌کند، حتی اگر تعداد برچسب‌های آمریکا بیش از شش برابر باشد. در هنگام میانگین‌گیری امتیازات برچسب‌ها، بهتر است میانگین وزنی را محاسبه کنید، که در آن وزن هر برچسب تعداد آن در y است. ما آن را **امتیاز میکرو F_1**[۲] می‌نامیم:

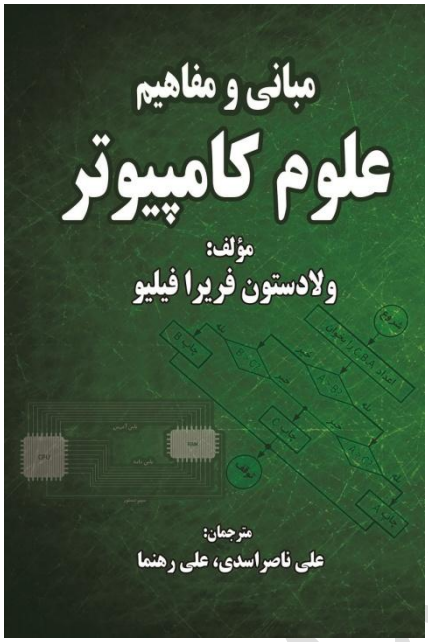
$$\frac{(65 \times 0.67) + (25 \times 0.42) + (10 \times 0.82)}{65 + 25 + 10} \approx 0.62 = 62\%$$

حتماً در شکل ۸-۶ ببینید که هر یک از اعداد ۶۵، ۲۵ و ۱۰ از کجا می‌آیند. علاوه بر این، به یاد داشته باشید که این امتیازها فقط نشان‌دهنده‌ی عملکرد مدل برای آستانه‌های خاص برچسبی هستند که استفاده می‌کنید، همانطور که برای طبقه‌بندی‌کننده‌های دودویی نیز این گونه است. اگر آستانه‌ها را تغییر دهید، همان مدل نمرات متفاوتی به دست می‌آورد.

---

۱ Macro F_1 Score
۲ Micro F_1 Score

معرفی کتاب



در کتاب حاضر دیدیم که چگونه علوم کامپیوتر نحوه‌ی تبادل اطلاعات و یادگیری از طریق حجم زیادی از داده‌ها را به طور اساسی تغییر داده است، اما نحوه‌ی کار خود کامپیوترها و نحوه‌ی تعامل برنامه‌نویسان با آن‌ها را بررسی نمی‌کردیم. این امر به آن علت است که ما قبلاً مفاهیم اصلی علوم کامپیوتر را در اولین کتاب خود، مبانی و مفاهیم علوم کامپیوتر، پوشش داده‌ایم. به شدت توصیه می‌کنیم آن کتاب را نیز مطالعه کنید!

مبانی و مفاهیم علوم کامپیوتر شرح مختصری از مفاهیم اساسی علوم کامپیوتر است که باید بدانید. سبک نوشتن آن شبیه به کتاب حاضر است: تشریفات دانشگاهی به حداقل رسیده و زبان آن واضح و قابل فهم است. این کتاب نیز برای مبتدیان طراحی شده است.

این کتاب با مقدمه‌ای ساده بر ریاضیات گسسته شروع می‌شود و سپس الگوریتم‌ها و ساختارداده‌های رایج را معرفی می‌کند. در نهایت، اصول کار سخت افزار کامپیوتر و زبان‌های برنامه نویسی را تشریح می‌کند. درک این موضوعات گام بعدی شما برای تسلط بر علوم کامپیوتر است، زیرا به خوبی محتوای این کتاب را تکمیل می‌کنند. می‌توانید کتاب مبانی و مفاهیم علوم کامپیوتر را از طریق کد زیر دریافت کنید: