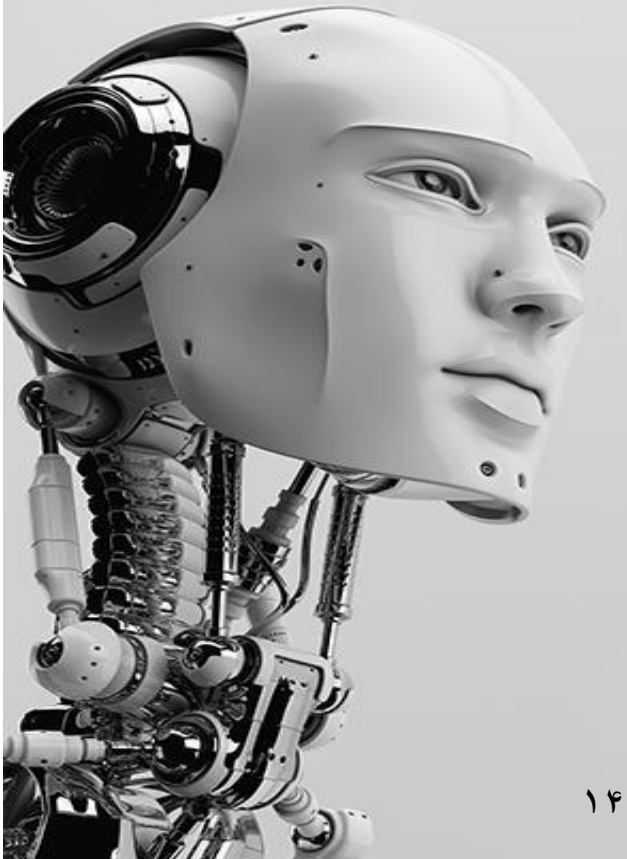
دوره آشنایی با روش‌های پیش پردازش داده‌ها

06

دوره آشنایی با روش‌های پیش پردازش داده‌ها

06



داده‌کاوی فرآیندِ استخراجِ دانش از میان انبوهی از داده‌ها است. مثال‌های متعددی که در دوره‌ی داده‌کاوی آوردیم، می‌تواند کمک کننده‌ی شما در فهم بهتر این رشته باشد. اگر مثال‌ها و دروس را دنبال کرده باشد، حتماً متوجه شده‌اید که در فرآیندهای داده‌کاوی و استخراج دانش، این داده‌ها هستند که نقش بسیار مهم و حیاتی دارند. زیرا الگوریتم‌های مختلف بایستی بر روی این داده‌ها عملیاتِ خود را انجام دهند. در واقع هر چه داده‌های بهتر و مناسب‌تری داشته باشیم، الگوریتم، بهتر یادگیری را انجام می‌دهد. مانند یک دانش‌آموزی که منابعِ بهتری برای امتحانِ ریاضی دارد، پس ریاضی را بهتر یاد می‌گیرد.

اهمیت پیش پردازش داده‌ها **data preprocessing** در فرآیندهای داده‌کاوی بسیار مهم و حیاتی است، به گونه‌ای که در بسیاری از مواقع، یک پیش پردازشِ خوب، می‌تواند باعثِ هموار شدنِ ادامه فرآیندهای داده‌کاوی شود

پیش پردازش داده‌هاData Preprocessingچیست؟

فرض کنید صاحبِ یک نانوایی هستید. برای تهیه‌ی **نان**، نیاز به **آرد** دارید. آرد نیز خود از گندم به دست می‌آید. یعنی گندم بایستی از **شکلِ اولیه‌ی** خود خارج شده و به آرد تبدیل شود (با فرآیندهای مختلفی که بر روی گندم انجام می‌شود) تا ماده‌ی اولیه‌ی تولیدِ نان، آماده شود. **در فرآیندهای داده‌کاوی مانند طبقه‌بندی و خوشه‌بندی، نیاز داریم تا داده‌ها برای الگوریتمْ آماده شوند**. زیرا معمولاً نمی‌توان داده‌ها را به صورت خام به الگوریتم‌های داده‌کاوی و یادگیری ماشین تزریق کرد.

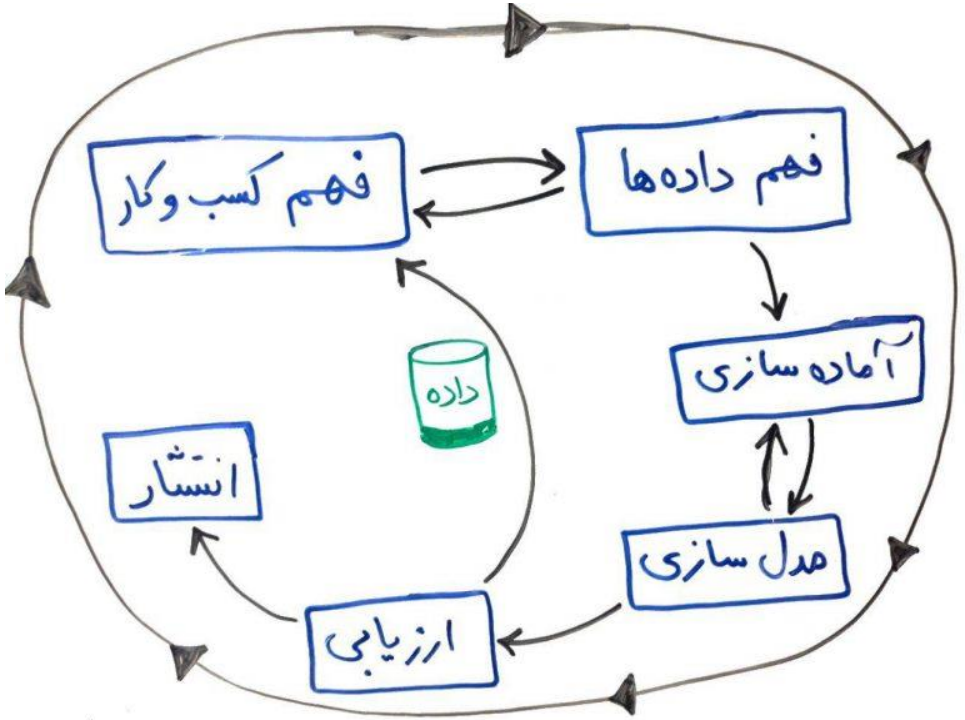
از آن‌جایی که داده‌ها معمولاً از منابعی تهیه می‌شوند که این منابع بدونِ توجه به فرآیندهای داده‌کاوی، داده‌ها را تولید یا نگهداری کرده‌اند، نیاز است تا داده‌ها، با توجه به شرایط و مسئله، **به داده‌های مناسب** جهتِ تزریق به الگوریتم‌های داده‌کاوی **تبدیل** شوند.

برای آماده‌سازیِ داده‌ها، نیاز است تا آن‌ها را از **شکل** و **حالتِ اولیه**، **خارج کرده** و به **شکلی** که برای الگوریتم **مناسب** باشد **تبدیل** کنیم. همچنین داده‌های موجود معمولاً دارای زواید مختلفی هستند که ممکن است **الگوریتم** را دچار **خطا** کنند. همان مثالِ نانوایی را به یاد بیاورید. فرض کنید در میانِ گندم‌ها، خورده سنگ هم وجود داشته باشد! طبیعتاً وجودِ خورده سنگ کیفیت آرد و به تبعِ آن، کیفیت نان را کاهش می‌دهد، پس نیاز است تا خورده سنگ‌ها از میانِ گندم‌ها پاک شوند. در داده‌کاوی هم نیاز داریم تا داده‌های اضافی که به مسئله و الگوریتم کمکی نمی‌کنند را حذف کنیم.

در درسِ فرآیند کریسپ CRISPدر داده کاوی بیان گردید، که در فرآیند کریسپ، پیش از مدل‌سازی توسط الگوریتم‌های یادگیری ماشین، نیاز به آماده‌سازی داده‌هاست که یکی از **مراحل** آن **همین پیش‌پردازش داده‌ها و استفاده از تکنیک‌هایی جهت آماده‌سازی** **prepare** آن‌ها قبل از تزریق به الگوریتم‌های داده کاوی و یادگیری ماشین در مرحله‌ی بعدی (یعنی مدل‌سازی) بود.

پیش پردازش داده‌هاData Preprocessingچیست؟

برای عملیاتِ پیش پردازش، روش‌ها و راهکارهای مختلفی طراحی شده است که در ادامه‌ی این درس به آن‌ها خواهیم پرداخت. همان‌طور که از نامِ این روش‌ها پیداست، عملیات پیش‌پردازش یا همان preprocessing معمولاً قبل از عملیات اصلیِ الگوریتم‌های داده‌کاوی انجام می‌گیرند و باعث تسهیل و کمک به الگوریتم‌ها می‌شوند.



نمونه فرمولهای ارزیابی مدل

Coefficient of Determination (R^2):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Normalized Mean Squared Error (NMSE) :

$$NMSE = \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i)^2}$$

Normalized Mean Absolute Error (NMAE) :

$$NMAE = \frac{\sum| \hat{y}_i - y_i|}{\sum| y_i|}$$

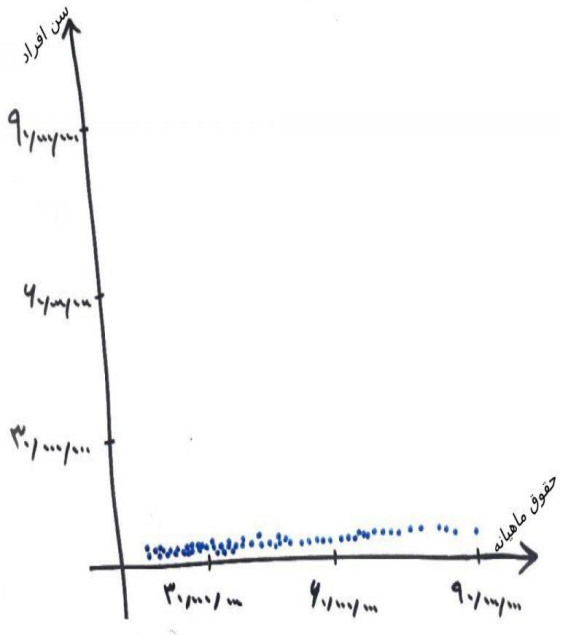
Normalized Root Mean Squared Error (NRMSE):

$$NRMSE = \frac{1}{\bar{y}}\sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}$$

نرمال کردن داده‌ها Data Normalizationو انواع آن

مسابقاتِ کشتی را تماشا کرده‌اید؟ در مسابقات کشتی هیچ‌گاه یک فرد با وزن ۹۰ کیلوگرم را با فردی با وزن ۱۲۰ کیلوگرم رو در رو نمی‌کنند. در واقع هر شخص باید در محدوده‌ی وزنِ خود کشتی بگیرد. در داده‌ها نیز شما نمی‌توانید یک مجموعه‌ی داده که مثلاً در بازه‌ی بین ۰ تا ۲۰ متغیر هستند را با مجموعه‌ی داده‌ای که در بازه‌ی بین ۰ تا ۱۰۰۰۰ قرار دارد، مقایسه کنید.در واقع این **دو مجموعه‌ی داده بایستی ابتدا هم وزن شوند** تا تاثیرِ یکی بیشتر از دیگر نباشد و به اصطلاح **fair** و منصف باشند.

اگر با درسِ ابعاد و ویژگی‌ها آشنایی داشته باشید احتمالاً شکل زیر برای شما قابل فهم است. فرض کنید می‌خواهید مشتریانِ خود را بر اساس ۲ ویژگیْ خوشه‌بندی کنید (یعنی به دو گروهِ مختلف تقسیم کنید). ویژگیِ اول، سنِ افراد (محور عمودی) و ویژگیِ دوم، حقوقِ ماهیانه‌ی افراد (محور افقی) است:



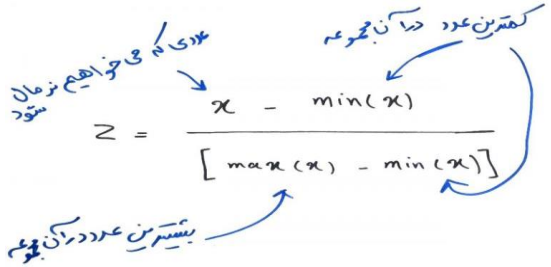
همان‌طور که مشاهده می‌کنید، داده‌ها در ۲ بُعد گسترش یافته‌اند. بُعدِ اول (محور عمودی)،سن که معمولاً بین ۲۰ تا ۹۰ سال است و بُعدِ دوم (محور افقی) حقوقِ ماهیانه که معمولاً بین ۹،۰۰۰،۰۰۰ تا ۱۰۰،۰۰۰،۰۰۰ ریال متغیر است. حال اگر بخواهی با استفاده از الگوریتم‌های خوشه‌بندی، عملیاتِ خوشه‌بندی را بر روی این داده‌ها انجام دهیم . ویژگیِ حقوقِ ماهیانه (محورِ افقی) ، تاثیر بسیار زیادی خاطراینکه بازه‌ی بزرگ‌تری از اعداد را در برمی‌گیرد و در اصطلاح scaleبیشتری دارد.یعنی تقریباً ویژگیِ سن، تاثیری بر روی الگوریتم ندارد. این یکی از مواقعی است که داده‌ها در بازه‌ی تغییراتِ متفاوت می‌توانند تاثیر غیرِ دلخواهی بر روی همدیگر و به تبعِ آن بر روی الگوریتم ، قرار دهند. پس داده‌ها باید در یک بازه‌ی **range** مساوی نسبت به یکدیگر قرار بگیرند ، مثلاً همه **در یک بازه‌ای مانند ۰ تا ۱** قرار داشته باشند و به این کار **نرمال‌سازی داده‌ها** یا data Nnormalization گفته می‌شود.

سید ایمان غفوریان - عضو هیات علمی دانشگاه آزاد مشهد-زمستان ۱۴۰۲

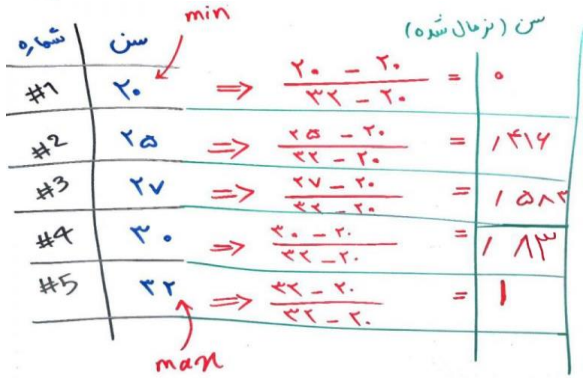
نرمال کردن داده‌ها Data Normalizationو انواع آن

روش‌های مختلفی جهتِ نرمال‌سازی داده‌ها وجود دارند که سعی داریم در دوره‌ای جدا در مورد هر کدام به تفکیک صحبت کنیم. اما در این بخش به یکی از معروف‌ترینِ این روش‌ها خواهیم پرداخت که به MinMaxNormalizationمعروف است. در این روش هر کدام از داده‌ها را می‌توان به یک بازه‌ی دلخواه تبدیل کرد.

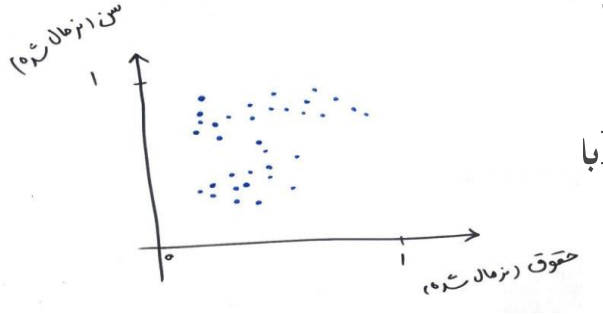
فرمول کلی MinMaxNormalizationبرای تبدیل داده‌ها به بازه‌ی بین ۰ تا ۱ به صورت زیر است:



برای مثال فرض کنید داده‌های سن برای افراد مختلف مانند شکل زیر است و ما می‌خواهیم سنِ این افراد را در یک بازه‌ی ۰ تا ۱ قرار دهیم. با توجه به فرمول بالا نتیجه به این صورت است:



همان‌طور که می‌بینید هر کدام از نمونه‌ها با توجه به مقادیرِ کمینه minو بیشینه maxبه بازه‌ی ۰ تا ۱ تبدیل شده‌اند. همین کار را می‌توان برای ستون‌های دیگر مانند حقوق انجام داد. شکل اولِ این بخش را ببینید. با نرمال‌سازیِ داده‌ها در بازه‌ی ۰ تا ۱، نمودار در ۲ بُعدی چیزی شبیه به شکل زیر می‌شود:



همان‌طور که می‌بینید هر کدام از نمونه‌ها با توجه به مقادیرِ کمینه min و بیشینه max به بازه‌ی ۰ تا ۱ تبدیل شده‌اند.

همین کار را می‌توان برای ستون‌های دیگر مانند حقوق انجام داد. شکل اولِ این درس را ببینید. با نرمال‌سازیِ داده‌ها در بازه‌ی ۰ تا ۱ نمودار در ۲ بُعدی چیزی شبیه به شکل زیر می‌شود:

یعنی مقیاسِ هر دو ویژگی در بازه‌ی ۰ تا ۱ قرار گرفته و حالا می‌توان الگوریتم‌های مختلف خوشه‌بندی و یا طبقه‌بندی را بر روی آن‌ها به صورت منصفانه اجرا کرد.

عملیات نرمال‌سازی قبل از بسیاری از الگوریتم‌های داده‌کاوی مانند شبکه‌های عصبیANN، SVM، KNNو KMeansبا انجام بگیرد تا ابعادِ مختلف به صورت عادلانه توسط الگوریتم بررسی شوند و تاثیرِ یکی بیشتر از بقیه نباشد.

سید ایمان غفوریان - عضو هیات علمی دانشگاه آزاد مشهد-زمستان ۱۴۰۲

تبدیل داده‌ها Data Transformationبه فُرمت قالب فهم برای الگوریتم داده‌کاوی

اکثرِ الگوریتم‌های داده‌کاوی، نیاز دارند تا داده‌های عددی را دریافت کنند و ساختارِ یادگیریِ آن‌ها بر اساسِ یادگیری از ماتریس‌های عددی است. در درسِ طبقه‌بندی دیدید که چگونه می‌توان یک سری ویژگی را به صورت ماتریس ساخت و به الگوریتمِ طبقه‌بندی داد. اما یادمان باشد که همیشه داده‌ها به صورتِ عددی آماده نیستند و بعضاً نیاز دارند تا به فرمتِ دلخواهِ الگوریتم (یعنی همان فرمتِ ماتریسِ عددی) تبدیل شوند. این دست از داده‌ها بایستی قبل از تزریق به الگوریتم، به فُرمتِ مناسب تبدیل transform شوند. روش‌های تبدیل داده بسیار گسترده و متنوع است و در این درس، یکی از آن‌ها را با هم مرور می‌کنیم.

فرض کنید در این‌جا تعدادی دانش‌آموز داریم که هر کدام ویژگی‌های مختلفی دارند. سن، معدل، قد و جنسیت ۴ویژگیِ دانش‌آموزان هستند که می‌خواهیم بر روی آن‌ها عملیاتی مانند عملیاتِ خوشه‌بندی را انجام دهیم.

	سن	معدل	قد	مرد	زن
Student #1	19	18.5	185	1	0
Student #2	21	17.0	162	0	1
Student #3	22	15.5	177	0	1
Student #4	20	18	156	1	0
⋮					

همان‌طور که می‌بینید، ۳ ویژگیِ اولْ عددی هستند و ویژگیِ آخر یعنی جنسیت ۲ مقدار دارد، مرد و زن. در اصطلاح، ویژگیِ جنسیت یک ویژگیِ categorical است، به این معنی که یک مقدارِ عددی نیست که به توان بزرگی یا کوچکی را با آن مشخص کرد.مثلا زن از مرد بزرگ‌تر نیست و یا برعکس. این دستْ از ویژگی ها برای بسیاری از الگوریتم‌های داده‌کاوی نامفهوم هستند. بنابراین بایستی به ویژگی‌های عددی تبدیل شوند.

در مثالِ بالا اگر بخواهیم جنسیت را به ویژگیِ عددی تبدیل کنیم، شکلِ مقابل به شکلِ زیر تبدیل می‌شود:

تبدیل داده‌ها Data Transformationبه فُرمت قالب فهم برای الگوریتم داده‌کاوی

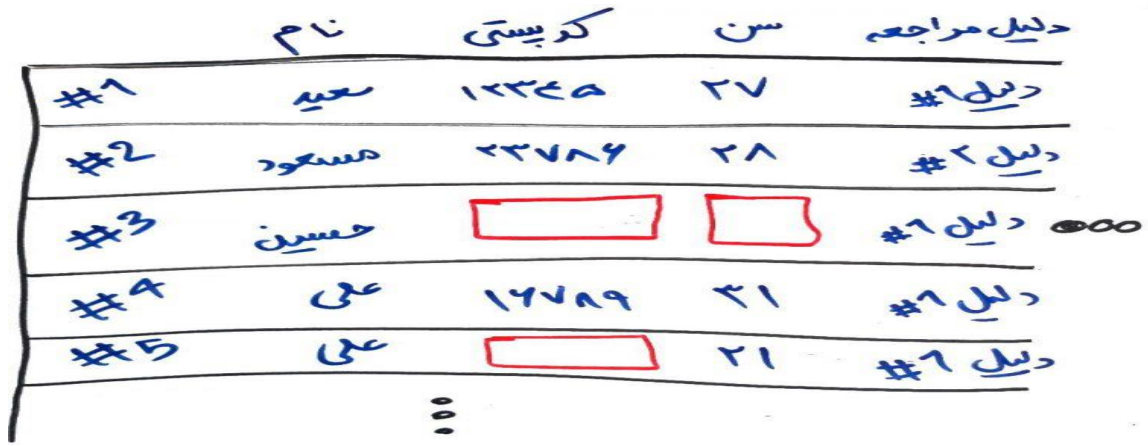
در این‌جا به ازای هر نوع جنسیت، یک ستون اضافه کردیم (ستون جنسیت نیز حذف شد). همان‌طور که مشاهده می‌کنید چون ۲ نوع جنسیت (مرد و زن) داشتیم، پس دو ستون یعنی دو ویژگی اضافه کردیم و با توجه به هر دانش‌آموز، مقدارِ مرد بودن و یا زن بودن را در سطرِ مربوطه برابرِ ۱ قرار دادیم و دیگری را برابرِ ۰. مثلا دانش‌آموزِ اول student#1 یک مرد است. پس ستون مرد، برای این دانش‌آموز برابرِ ۱ و ستون زن برای این دانش‌آموز برابر ۰ قرار می‌گیرد، و به همین ترتیب برای بقیه‌ی سطرها همین‌کار را تکرار می‌کنیم. مثلاً اگر این ستون مقدار دیگری مانندِ جنسیت نامشخص داشت، آن‌گاه بایستی این ویژگی به سه ستون شکسته شود. مرد، زن و نامشخص و در هر سطر، یکی از این ویژگی‌ها ۱ و بقیه ۰ می‌شدند. در اصطلاح به این فرآیند one hot encodingنیز گفته می‌شود، چون در هر سطر فقط یکی از ستون‌های آن نوع (مثلاً جنسیت) ۱ و بقیه صفر

	سن	معدل	قد	مرد	زن
Student #1	19	18.5	185	1	0
student #2	21	17.0	162	0	1
student #3	22	15.5	177	0	1
student #4	20	18	156	1	0
⋮					

داده‌های گم‌شده Missing Valuesو راهکارهای مقابله با آن

فرض کنید مدیرِ یک اداره هستید و در روز عده‌ی کثیری از مراجعه‌کنندگان به اداره‌ی شما مراجعه می‌کنند. در این مراجعه به هر فرد یک فُرمِ دریافت مشخصات داده می‌شود و از او خواسته می‌شود تا اطلاعاتی مانند نام، جنسیت، سن، کدپستی، نامِ پدر، شماره‌ی شناسنامه، آدرس منزل، تعداد دفعات مراجعه‌ی قبلی، و دلیلِ مراجعه‌ی خود را در آن فُرم بنویسد تا بعداً عملیاتِ پردازشی بر روی داده‌ها انجام شود (مثلاً با عملیات متن‌کاوی text mining متوجه شوید که معمولاً مراجعه‌کنندگان به چه دلایلی به اداره مراجعه می‌کنند یا اینکه معمولاً این مراجعه‌کنندگان ساکن کدام محدوده‌ی شهر با توجه به کدپستی هستند). با این تفاسیر آیا تضمین می‌کنید که تمامِ مراجعه‌کنندگان، اطلاعاتِ خود را کامل وارد کنند؟ مثلاً اگر شخصی کدپستیِ خود را فراموش کرده باشد چه؟ در این‌جا طبیعتاً داده‌ها ناقص است و در واقع بعضی از داده‌ها وجود ندارند. به این دست از داده‌ها، missing valuesیا داده‌های گم شده یا داده‌های مفقود می‌گویند که موردِ بحثِ این درس است.

فقدانِ داده‌ها معمولاً یکی از مسائلِ اصلی در حوزه‌ی جمع‌آوری داده است. برای همین راه‌حل‌های فراوانی برای غلبه بر این مشکل به وجود آمده است. یکی از راه‌حل‌ها غلبه بر داده‌ی گم شده با پر کردنِ آن توسط داده‌ی واقعی است. مثلاً در مثالِ بالا به افرادی که اطلاعاتِ ناقص گذاشته‌اند تلفن بزنید و کدپستیِ آن‌ها را به صورتِ دقیق بپرسید. ولی معمولاً روش‌های غیر از این هم وجود دارد. به شکل زیر نگاه



داده‌های گم‌شده Missing Valuesو راهکارهای مقابله با آن

تصویر بالا یک قسمت از داده‌های آن فرم را نمایش می‌دهد. در این‌جا ستونِ کدِپستی برای مراجعه‌کنندگانِ ۳ و ۵ خالی است. همچنین ستونِ سن، برای مراجعه‌کننده‌ی ۳ فاقدِ داده است. در ادامه، روش‌هایی را با هم مرور می‌کنیم که می‌توانند برای مقابله با این داده‌های مفقود شده، کمک کنند.

یکی از روش‌ها این است که مراجعه‌کنندگانِ ۳ و ۵ را از بین داده‌ها حذف کنیم. این‌کار باعث می‌شود که معادلاتِ آماری در داده‌ها (مخصوصاً هنگامی که داده‌های فراوان داریم) تقریباً مانندِ قبل بماند و در واقع حذفِ چند رکورد، عموماً در این معادلات خللی وارد نمی‌کند. ولی، به نوعی پاک کردن صورت مسئله است.

راه‌حلِ دیگر این است که ستونِ کدپستی را از بینِ داده‌ها حذف کنیم! که البته در این مثال کاری منطقی به نظر نمی رسد.

در بعضی از مواقع می‌توان میانگین یا میانه‌ی median اعدادِ موجودِ رکوردهای دیگر را برای یک ستونِ خاص محاسبه کرده و به جای مقادیرِ مفقود شده قرار داد. مثلاً اگر سنِ فردی دارای مقدارِ مفقود شده بود، می‌توانیم میانگینِ سنِ افرادی که ستونِ سن، برای آن‌ها وارد شده است را حساب کنیم و به جای این مقادیر مفقود شده برای مراجعه‌کنندگان دیگر قرار دهیم. در مثالِ بالا برای ستونِ کدِپستی این کار منطقی به نظر نمی‌رسد، چون مقدارِ کدِپستی مقداری نیست که بتوان میانگینِ آن را محاسبه کرد و میانگینِ آن بی‌معنی به نظر می‌رسد.

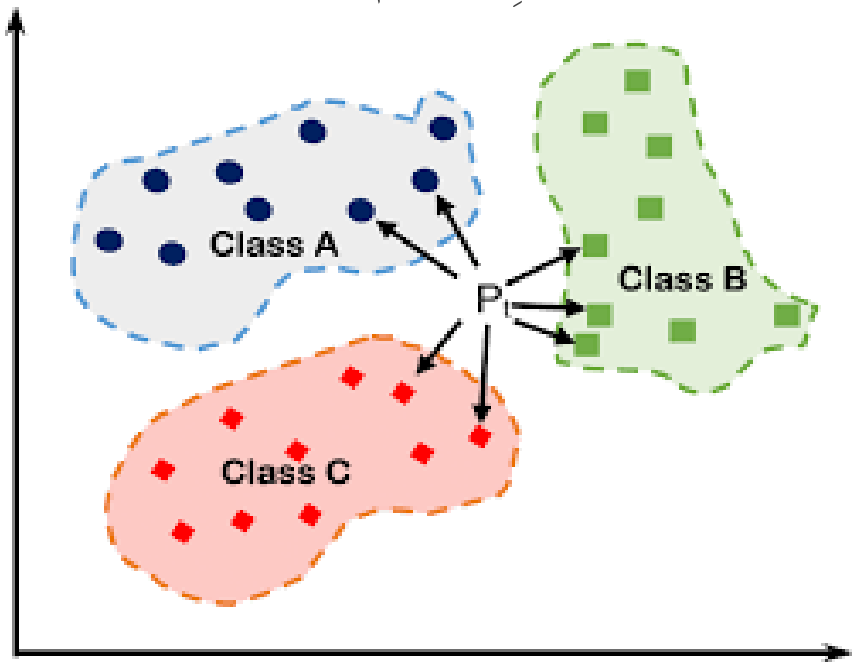
همچنین می‌توانیم یک مقدار ثابت را در نظر گرفته و به جای مقادیر مفقود شده قرار دهیم. مثلا برای سن، عددی مانند ۲۵ را برای اشخاصی که سنِ آن‌ها وارد نشده است قرار دهیم. یا کدپستی را یک کد پستی واقع در مرکز شهر در نظر بگیریم.

این روش‌ها کاملاً بدیهی به نظر می‌رسند و احتمالاً می‌توانید روش‌های دیگری را نیز با کمی تعقل ابدا کنید. اما اجازه بدهید یک روشِ دیگر که کمی با روش‌های فوق متفاوت هست را با یکدیگر نگاهی بیندازیم.

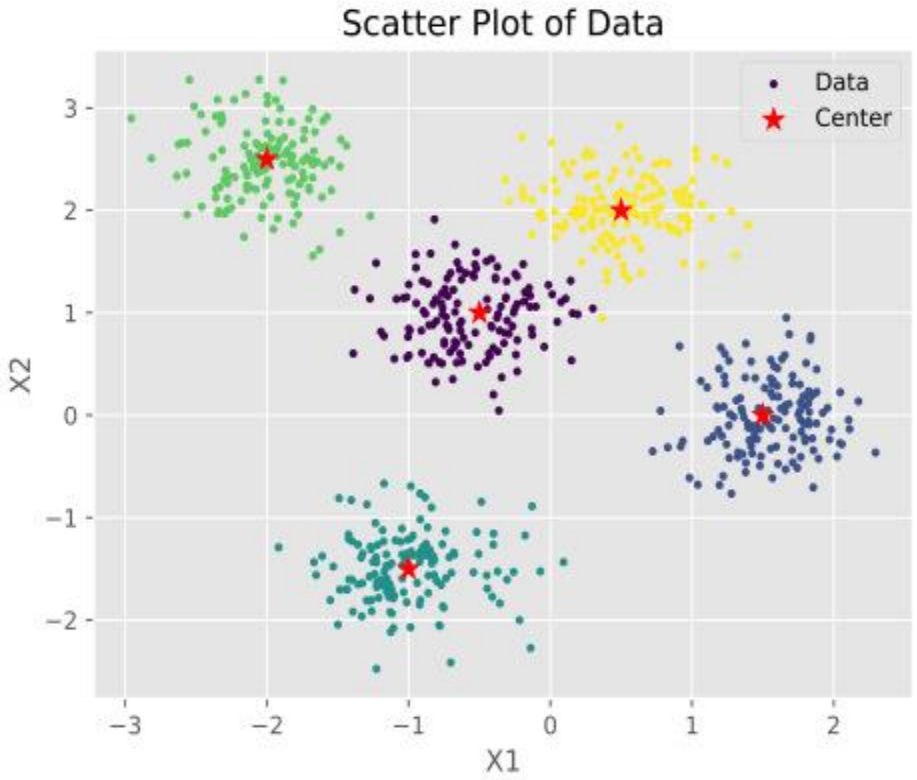
داده‌های گم‌شده Missing Valuesو راهکارهای مقابله با آن

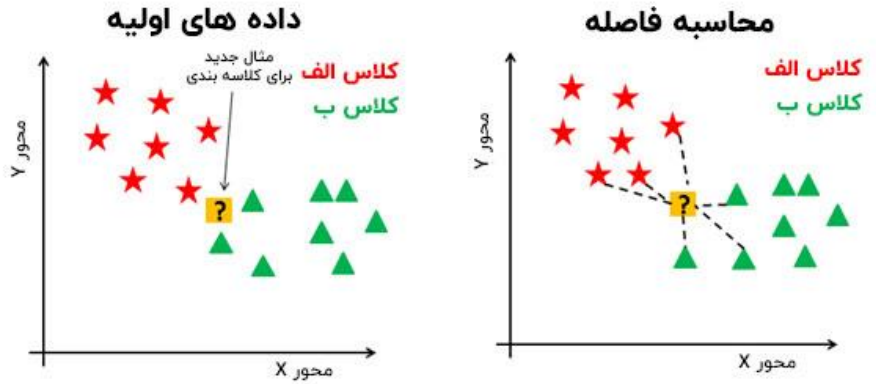
الگوریتمِ KNNرا از دوره‌ی طبقه‌بندی به یاد بیاورید. این الگوریتم به دنبالِ نزدیک‌ترین همسایه در بین نمونه‌های دیگر در داده‌های موجود می‌گشت. برای حلِ مشکلِ داده‌های مفقود نیز می‌توان از این روش استفاده کرد. در همان مثالِ بالا، فرض کنید مراجعه‌کننده‌ی شماره‌ی ۳ را، که مقدار سنِ خود را وارد نکرده است، در نظر داریم. با توجه به ویژگی‌های دیگرِ این مراجعه‌کننده (مثلاً علت مراجعه، جنسیت، تعداد دفعات مراجعه‌ی قبلی و...) به دنبال نزدیک‌ترین فرد در بینِ مراجعه‌کنندگانی که فرمِ خود را کامل پر کرده‌اند، می‌گردیم. فرض کنید مراجعه‌کننده‌ی شماره‌ی ۱، نزدیک‌ترین فرد به این شخص باشد (بر طبق الگوریتم KNNو فاصله‌ی اقلیدسی). حال سنِ مراجعه‌کننده‌ی شماره‌ی ۱، یعنی ۲۷سال را برای مراجعه‌کننده‌ی شماره‌ی ۳ نیز قرار می‌دهیم. در واقع در این‌جا با استفاده از روشِ الگوریتم نزدیک‌ترین همسایه KNN توانستیم سنِ مراجعه‌کننده‌ی شماره‌ی ۳ را تخمین بزنیم.

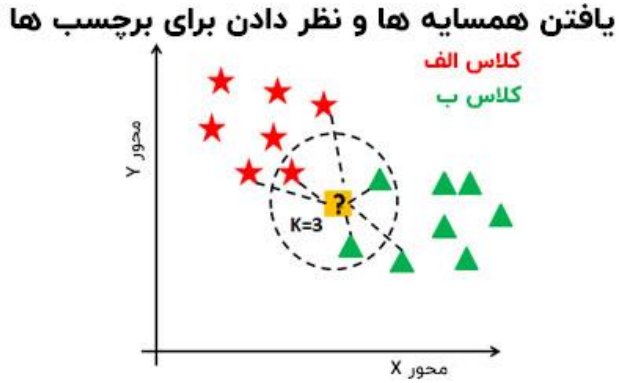
روش‌های متعدد و گسترده‌ای در زمینه‌ی مقابله با داده‌های مفقود شده وجود دارد .



KNNالگوریتم







تشخیص داده‌های پرت و دارای نویز Noiseو راه‌کار مقابله با آن‌ها

داده‌های پرت و داده‌هایی که دارای نویز noise هستند، در بسیاری از مجموعه‌ها، دیده می‌شوند. فرض کنید شما مدیرِ یک وب‌سایتِ فروشگاهی هستید و می‌خواهید سنِ کاربران خود را تحلیل کنید. مثلاً این‌که افراد در بازه‌ی سنیِ مختلف، بیشتر به کدام محصولات تمایل نشان می‌دهند. برای این‌کار در هنگام خرید، سنِ خریدار را از او دریافت می‌کنید. آیا مطمئن هستید که افراد معمولاً سنِ خود را در بازه‌ی ۰ تا ۱۰۰ سال وارد می‌کنند؟ برای مثال شخصی ممکن است سهواً سنِ خود را به جای ۲۵ سال، ۲۵۰ سال درج کند و یا شخصی به جای این‌که سن خود را درج کند، سهواً سالِ تولد خود را وارد نماید! به این دست از داده‌ها که معمولاً با بقیه‌ی داده‌ها ناسازگار هستند داده‌های پرت outliers می‌گویند و مجموعه‌ی داده را دارای نویز noiseمی‌دانند. در این درس می‌خواهیم ببینیم چه راه‌کارهایی برای مقابله با این دست از داده‌ها وجود دارد.

نویزها که به داده‌های غیرطبیعی anomaliesنیز شهرت دارند، باعث خراب شدنِ آمارها و داده‌های مجموعه‌ی داده می‌شوند. برای مثال فرض کنید سنِ افرادِ مختلف از آن‌ها دریافت کرده و در جدولی مانندِ جدولِ زیر قرار داده‌اید.

کاربر	سن
#1	17
#2	15
#3	25
#4	250 → ?
#5	71
#6	62
#7	33
#8	44
#9	1363 → ?

کاربرانِ شماره‌ی ۴ و ۹ داده‌هایی غیر طبیعی در ستون سن دارند. مثلاً کاربرِ شماره‌ی ۹، سهواً تاریخِ تولدِ خود را وارد کرده است و کاربرِ شماره‌ی ۴ نیز، به اشتباه یک صفرِ اضافی برای عددِ سنِ خود درج کرده. پس به سادگی می‌توان تشخیص داد که این مجموعه‌ی داده برای مقدارِ سن دارای داده‌های پرت است.

روش‌های حذف داده‌های دارای نویز زیاد است و در این بخش به چند روشِ ساده و کاربردی در شناسایی و حذف نویز خواهیم پرداخت . یکی از این روش‌ها حذفِ مقادیر بالا و پایین داده‌ها به تعداد مشخص است. برای مثال در همین جدول بالا، می‌توانیم مقادیری که کمتر از ۱۰ و یا بیش‌تر از ۱۰۰ هستند را حذف کنیم و یا مقادیری که در بازه‌ی بین ۱۰ تا ۱۰۰ قرار ندارد را با میانگینِ سن‌های باقی‌مانده جایگزین کنیم. با این‌کار داده‌ها در یک بازه‌ی مشخص و معقول قرار می‌گیرند. پس در مثال بالا ، می‌توانیم کاربران ۴ و ۹ را حذف کنیم و یا مقدار سن را برای آن‌ها برابر ۳۸ که میانگین سن‌های باقی‌مانده افراد است، قرار می‌دهیم.

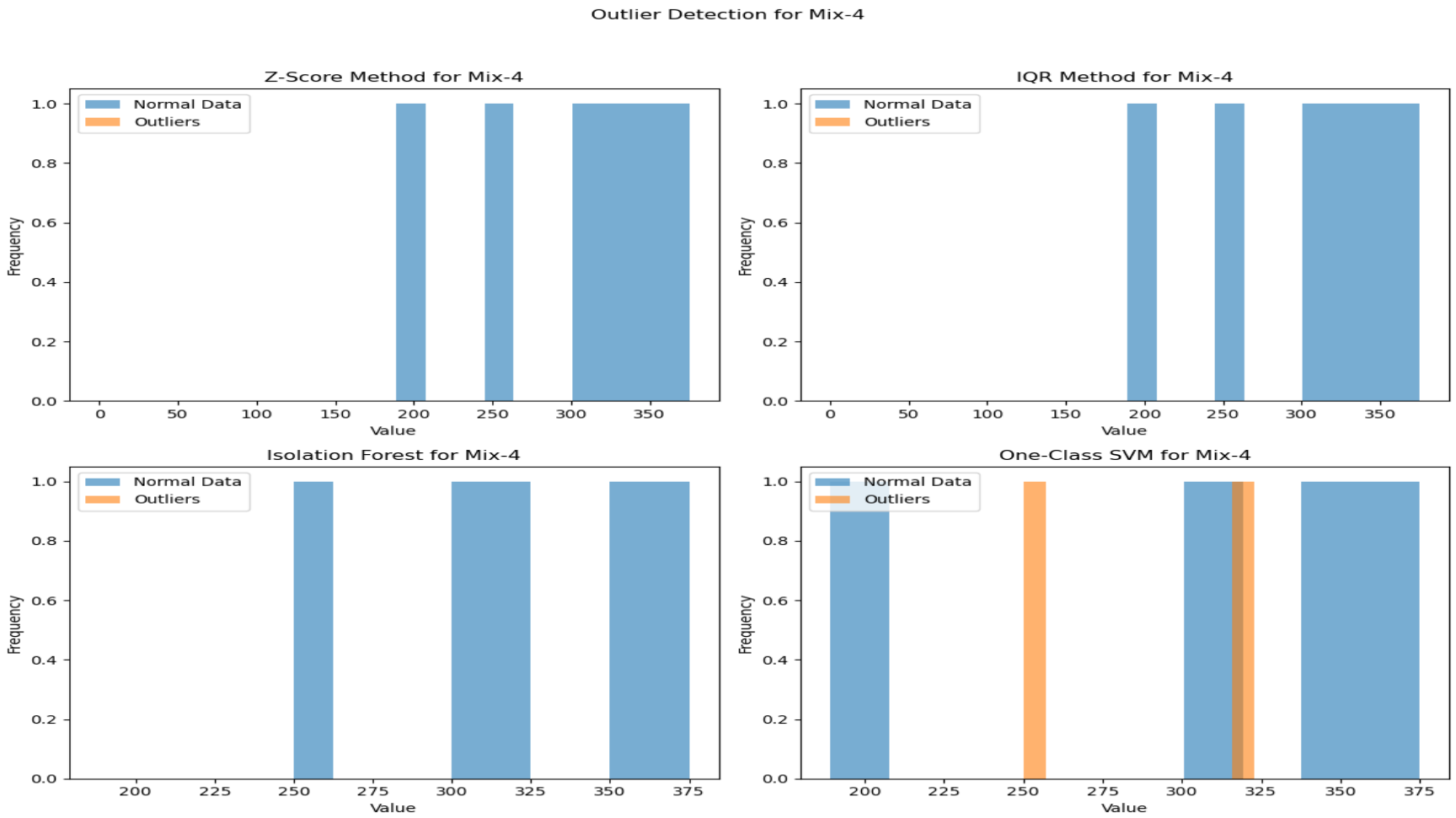
تشخیص داده‌های پرت و دارای نویز Noiseو راه‌کار مقابله با آن‌ها

البته در بعضی از مواقع ما به دنبال پیدا کردنِ نویزها هستیم تا داده‌ها را با توجه به مقادیرِ غیرطبیعی anomalies تحلیل کنیم. مثلاً می‌خواهیم در یک سری تراکنش‌های بانکی، آن دسته از تراکنش‌هایی که رفتارِ غیرِ عادی داشتند را کشف کرده و به تخلف‌های یک فرد در بانک رسیدگی کنیم. اگر درسِ خوشه‌بندیِ DBSCAN را خوانده باشید متوجه می‌شوید که این الگوریتم یکی از الگوریتم‌هایی است که می‌تواند داده‌های پَرت را تشخیص دهد. در واقع DBSCAN را هم می‌توان برای خوشه‌بندی مورد استفاده قرار داد و هم می‌توان از آن به عنوانِ یک الگوریتمْ جهتِ تشخیص داده‌های پرت استفاده کرد. همچنین روشی به عنوان SVM تک کلاسه one class SVM موجود است که می‌تواند داده‌های پرت را تشخیص دهد.اما ممکن است در بعضی از مواقع، کلاً یک ویژگی (یک بُعد) پَرت باشد مثالِ زیر را از درسِ درخت تصمیم در نظر بگیرید:در این مجموعه‌ی داده، یک رئیس دانشکده می‌خواهد بر اساسِ ویژگی‌های دانشجویان و سابقه‌ی دانشجویانِ گذشته، به این نتیجه برسد که کدام یک از دانشجوهای جدید، می‌توانند در آزمون دکتر قبول شوند. همان‌طور که مشاهده می‌کنید، ویژگی‌هایی مانندِ معدلِ کل، تعداد مقالات، مدرک زبان IELTو سنوات تحصیلی در تشخیص و ساختِ مدل جهت پیش‌بینی داده‌های آینده، کاربرد دارند.

	معدل کل	تعداد مقالات	مدرک زبان IELTS	سنوات تحصیلی	دکتری قبول شده؟
#1	19,5	3	1	3	بلی
#2	16,5	0	1	4	خیر
#3	15	0	0	3	خیر
#4	17	2	1	2,5	بلی
#5	18,5	2	0	2,5	بلی
#6	15,5	1	1	2,5	خیر
#7	19	3	1	3	بلی

حال فرض کنید یک ویژگیِ دیگر مانندِ جنسیت به ویژگی‌های بالا اضافه شده باشد. آیا این ویژگی می‌تواند در تشخیصِ این‌که شخصی دکتری قبول شود یاخیر موثر باشد؟فرض کنیم جواب منفی است،یعنی ویژگیِ جنسیت با توجه به معیارهای آماری و از روی دانشجویان گذشته، تاثیری بر قبول شدن یا نشدنِ افراد در مقطع دکتری ندارد. پس این ویژگی یک ویژگیِ نویز به حساب می‌آید. یعنی برخی اوقات یک ویژگی یا بُعد نیز می‌تواند نویز باشد به این صورت که در تصمیم گیریِ نهایی تاثیرِ چندانی نداشته باشد. روش‌های تشخیص ویژگی‌های نویز بسیار هستند. یکی از آن‌ها که در دوره‌ی جبرخطی در موردِ آن صحبت کردیم، الگوریتمِ PCAاست که ویژگی‌هایی با تاثیرِ کم را از میان ویژگی‌های خود تقریبا حذف می‌کند. روش‌های دیگری مانند chi2نیز وجود دارند که قادر به تشخیصِ ویژگی‌های کم‌اهمیت هستند.

تشخیص داده‌های پرت و دارای نویز



و کاهش ابعاد Feature Sectionانتخاب ویژگی

فرض کنید مدیرِ یک مجموعه هستید و می‌خواهید با توجه به ویژگی‌های یکی از کارکنانِ خود، وظیفه‌ای را به او محول کنید (مثلاً او را مدیرِ یک پروژه کنید). ولی این شخص تا به حال پروژه‌ای برای شما انجام نداده است. برای این کار ویژگی‌های مختلفی از کارکنانِ قبلیِ خود را جمع‌آوری کرده و آن‌ها را در جدولی مانندِ جدول زیر رسم می‌کنید:

همان‌طور که می‌بینید، ویژگی‌هایی مانندِ سن، تعداد سالِ تجربه کاری، تعداد فرزندان و جنسیت را گردآوری می‌کنید و ستونِ آخر هم برچسب یا طبقه است و به این معناست که این شخص توانسته آخرین پروژه‌ی کاریِ خود را موفقیت به پایان برساند یا خیر؟ در واقع شما می‌خواهید با توجه به ویژگی‌های این کارکنان، نوعی یادگیری را بر روی داده‌ها انجام داده و ببینید که فردِ شماره‌ی ۱۰۰۰ که فردِ جدیدی هست و تا به حال پروژه‌ای انجام نداده، آیا می‌تواند از عهده‌ی این پروژه برآید یا خیر؟ (حتماً می‌دانید که این یک مثالِ طبقه‌بندی است)در مثالِ بالا، ۴ ویژگیِ مختلف داریم که امیدواریم در تصمیم‌گیریِ نهایی تاثیر داشته باشند. نگاهی به ویژگیِ جنسیت بیندازید. دقیقاً ۲نفر از مردان و ۲نفر از زنان آخرین پروژه‌ی خود را با موفقیت پشت سر گذاشته‌اند و ۲نفر از مردان و ۲نفر از زنان نتوانسته‌اند در آخرین پروژه‌ی خود موفقیت باشند. نتیجه این است که **جنسیت در انجامِ موفق پروژه نقشی نداشته اشت**. به عبارتِ دیگر، ویژگیِ جنسیت، اطلاعاتِ خاصی به ما نمی‌دهد و الگوریتمِ طبقه‌بندی نیز نمی‌تواند از آن اطلاعاتی کسب کند. در واقع این ویژگی دارای ارزش کمی است و بهتر است از میانِ ویژگی‌ها کنار گذاشته شود . پس در اینجا تعداد ویژگی‌های ما می‌تواند از ۴ به ۳ کاهش پیدا کند.مشاهده می‌کنید، با یک نگاه توانستیم آن ویژگی‌ای که ارزش مناسبی نداشت (در این‌جا جنسیت) را پیدا کنیم. اما با بزرگ‌تر شدنِ مجموعه‌ی داده و تعدادِ ویژگی‌ها، نمی‌توان این کار را بدون استفاده از روش‌های آماری و الگوریتم‌های مخصوص به آن انجام داد. برای همین در این موارد می‌توان از الگوریتم‌های کاهشِ ویژگی استفاده کرد و تعداد ویژگی‌ها یا همان ابعاد را کاهش داد. به این الگوریتم‌های در اصطلاح الگوریتم‌های کاهشِ ابعاد یا dimensionality reductionنیز می‌گویند.البته این‌طور نیست که فقط به خاطرِ پایین بودنِ ارزشِ یک ویژگی بخواهیم آن را کنار بگذاریم. در بعضی از موارد تعدادِ ویژگی‌های مجموعه‌ی داده بسیار زیاد است والگوریتم‌های داده‌کاوی (مانند طبقه‌بندی یا خوشه‌بندی) در ابعاد زیاد دچار خطا می‌شوند و یا سرعت انجامِ عملیات در آن ها کاهش پیدا می‌کند. همچنین در بعضی از موارد می‌خواهیم با کاهشِ تعداد ابعاد، آن‌ها را در یک نمودار یا چارت رسم کنیم. برای همین بایستی داده‌ها را به تعداد ۲ یا ۳ بُعد تبدیل کرده تا قابل نمایش باشند. الگوریتم‌های مختلفی برای کاهش ابعاد وجود دارد.در موردِ الگوریتم PCAدر دوره‌ی جبر خطی صحبت کردیم. این الگوریتم می‌تواند داده‌ها به ابعاد کوچک‌تر تبدیل کند . همچنین الگوریتم هایی مانند chi2نیز می‌توانند با شناساییِ ویژگی‌ها ارزشمند، آن‌ها را از ویژگی‌های غیرِ ارزشمند جدا کرده و به نوعی کاهش ابعاد انجام دهند.

	سن	تجربه کاری	فرزندان	جنسیت	پروژه قبل
#1	25	3	0	مرد	موفق
#2	35	10	1	مرد	شکست
#3	22	2	0	زن	موفق
#4	27	4	0	مرد	موفق
#5	30	3	2	زن	شکست
#6	19	1	0	زن	شکست
#7	21	3	1	مرد	شکست
#8	29	9	1	زن	موفق
#1000	30	3	1	مرد	?

انتخاب نمونهInstance Selectionدر پیش پردازش داده‌ها

معمولاً در بحثِ پردازش داده‌ها و داده‌کاوی، یکی از محدودیت‌ها، محدودیت در منابع **سخت افزاری** است. برای مثال فرض کنید، ۵۰ گیگابایت داده در اختیار داریم ولی مقدار حافظه‌ی موقتِ **RAM** موجود، ۴ گیگابایت است. یکی از راه‌کارها، برای حل این دست مسائل، **کاهش دادن داده‌ها** است. در درسِ قبل دیدیم که چگونه با حذفِ یک ویژگی (یک بُعد)، حجمِ داده‌ها کاهش پیدا می‌کند. در این درس می‌خواهیم ببینیم که چگونه به جای حذفِ یک ویژگی، نمونه‌های مختلف رامی‌توان از بین داده‌ها کنار گذاشت.

در این مثال، ما ۸ نمونه کارمند داریم که هر کدام ۴ویژگی دارند و می‌خواهیم با توجه به این ۴ ویژگی، یاد بگیریم که معمولاً یک شخص با چه ویژگی‌هایی می‌تواند یک پروژه را به موفقیت برساند (که این مثالی از طبقه‌بندی بود). ولی در میانِ داده‌ها، برخی از نمونه‌ها هستند که اطلاعاتِ مفیدی برای الگوریتمِ طبقه‌بندی فراهم نمی‌آورند. الگوریتم‌های انتخابِ نمونه یا همان **instance selection**، می‌توانند این نمونه‌ها را شناسایی کرده و آن‌ها را از میانِ داده‌ها حذف کنند.

با این‌کار **سرعت** در عملیاتِ یادگیریِ ماشین و طبقه‌بندی **بیشتر** می‌شود ولی **دقتِ** طبقه‌بندی تقریباً **مانندِ قبل باقی می‌ماند** یا ممکن است **کمتر** شود. توجه کنید که الگوریتم‌های **طبقه‌بندی** معمولاً **خطاهای** معقولی دارند و عملیاتِ **کاهشِ نمونه**، بایستی با **ثابت** نگه‌داشتن **خطای** یک الگوریتمِ طبقه‌بندی، **تعداد نمونه‌ها** را **کاهش** دهد.

	سن	تجربه کاری	فرزندان	جنسیت	پروژه قبل
#1	25	3	0	مرد	موفق
#2	35	10	1	مرد	شکست
#3	22	2	0	زن	موفق
#4	27	4	0	مرد	موفق
#5	30	3	2	زن	شکست
#6	19	1	0	زن	شکست
#7	21	3	1	مرد	شکست
#8	29	9	1	زن	موفق
#1000	30	3	1	مرد	?