**Scikit-learn**

Scikit-learn (Import : sklearn) توابع و کلاس های مختلفی را برای پیش پردازش داده ها و مدل سازی ( یادگیری ماشین ) ارائه می دهد.

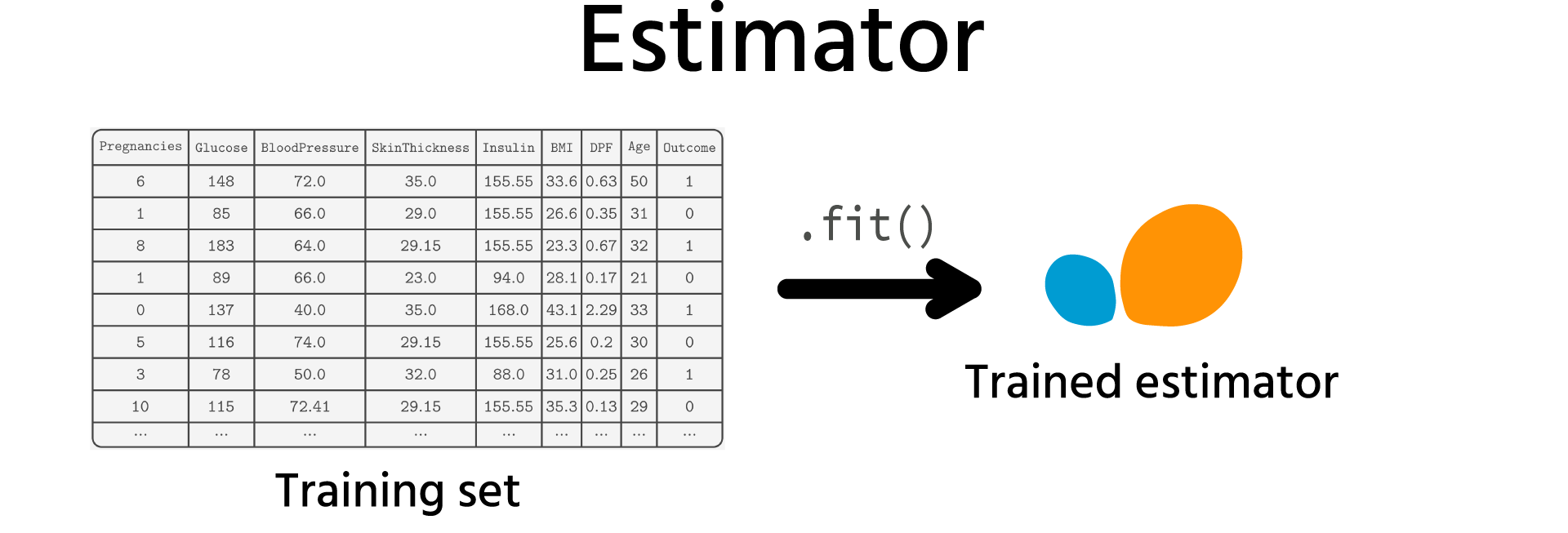
اشیاء اصلی sklearn ، برآوردگر (estimator)، تبدیل کننده (transformer) ، پیش بینی کننده (predictor)و مدل هستند.

Estimator

**هر کلاس sklearn با متد .fit () یک Estimator در نظر گرفته می شود.**

**متد ()fit به یک شی اجازه می دهد تا از داده ها یاد بگیرد.**

**Fit method args : پارامترهای X و y را می گیرد (y برای کارهای یادگیری بدون نظارت اختیاری است).**

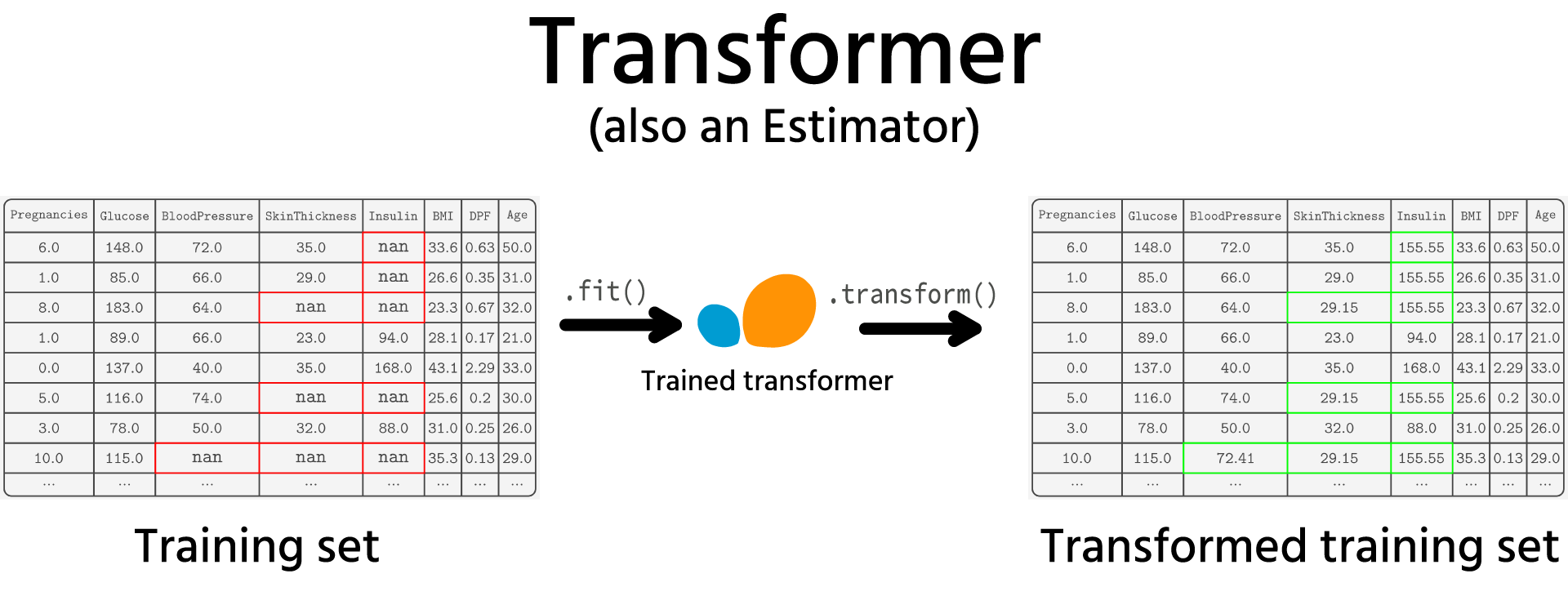
****

Transformer

**یک تبدیل کننده دارای روش .fit () و متد .transform است که داده ها را به نوعی تبدیل می کند.**

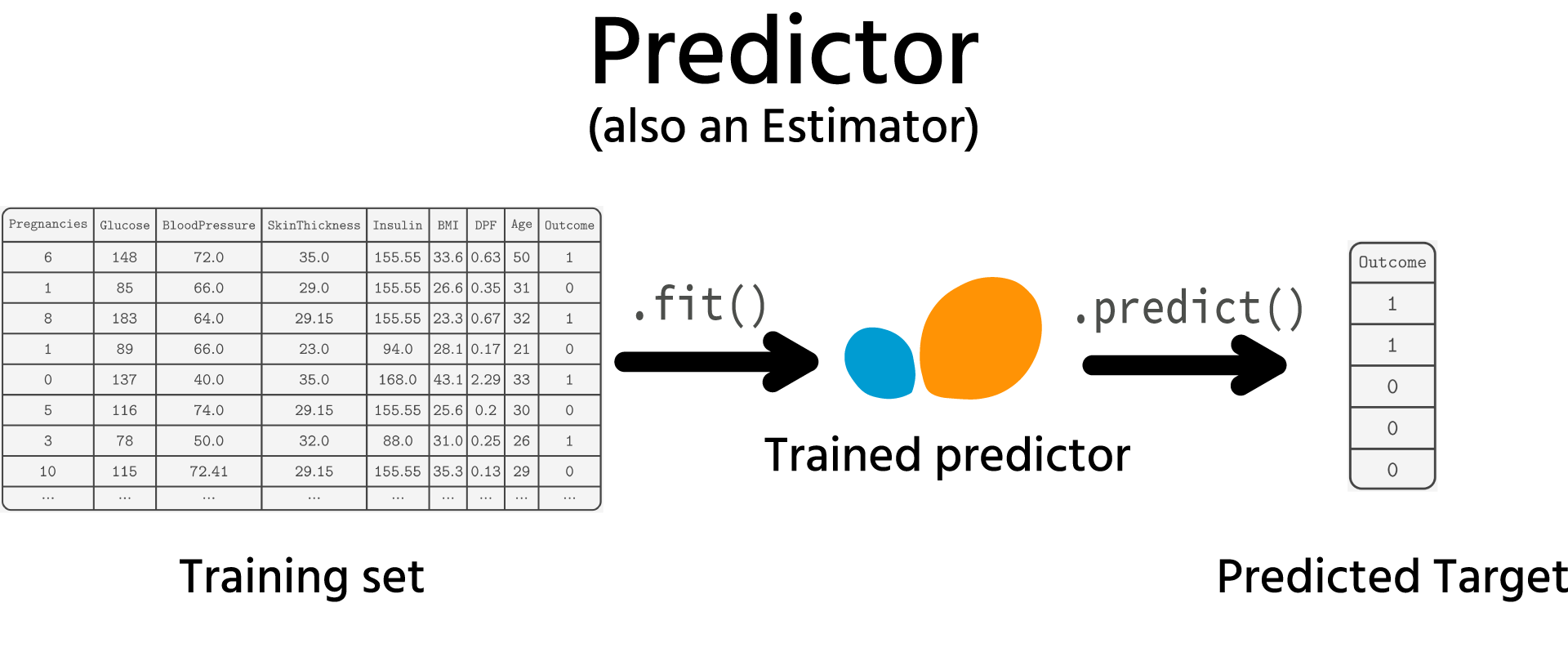
**معمولاً، تبدیل کننده ها قبل از تبدیل داده ها باید چیزی را از داده یاد بگیرند ، بنابراین باید از .fit() و سپس (.transform) استفاده کنید. برای جلوگیری از آن، ترانسفورماتورها متد ()fit\_transform را نیز دارند.**

**نکته : ترانسفورماتورها معمولاً برای تبدیل آرایه X استفاده می شوند. با این حال، همانطور که در مثال LabelEncoder خواهیم دید، برخی از ترانسفورماتورها برای آرایه y ساخته شده اند.**

****

Predictor

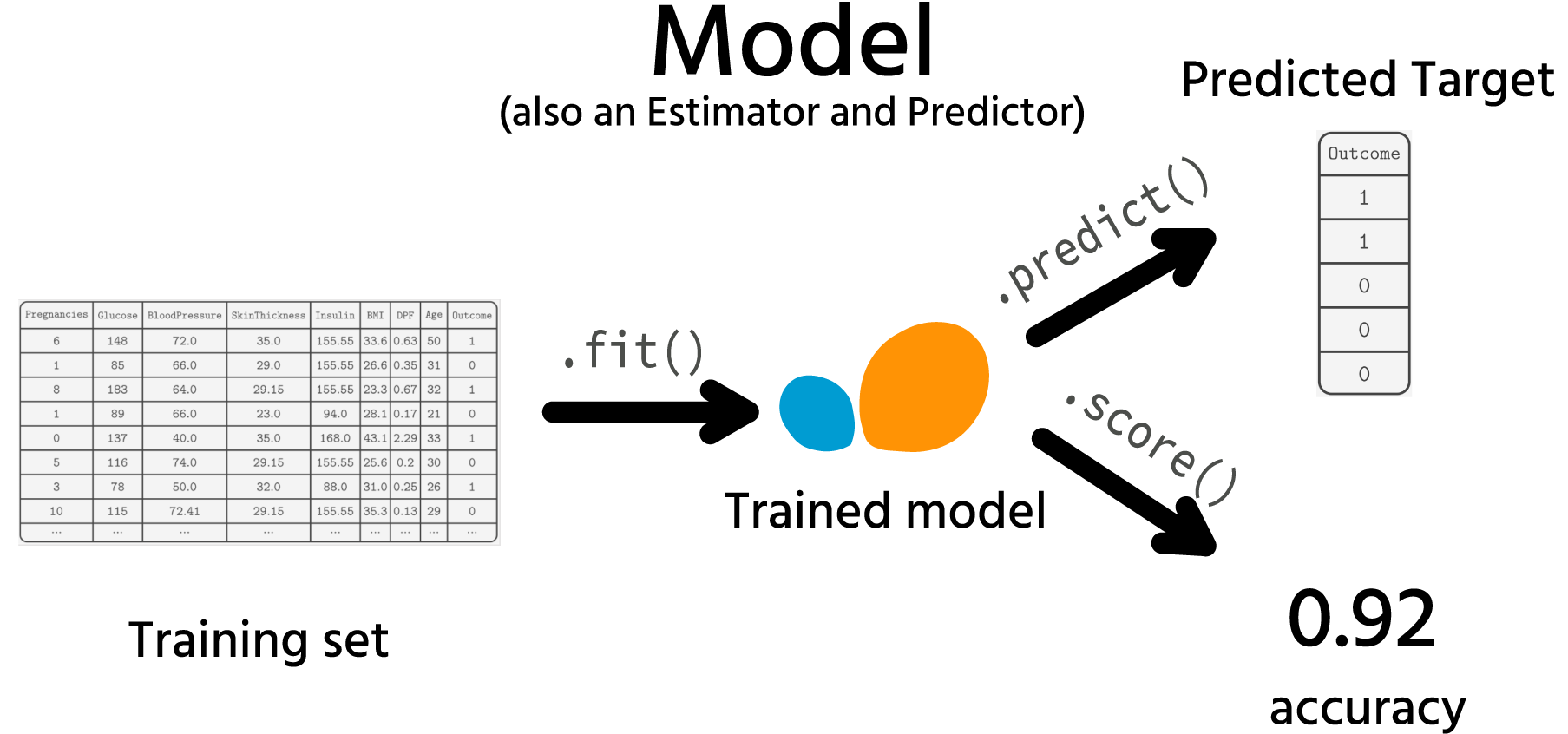
**پیش‌بینی‌کننده ، تخمین‌گر ( estimator ) است که دارای .fit() و متد ()predict است. از متد .predict() برای پیش بینی استفاده می شود.**

****

Model

**یک مدل پیش بینی کننده ای است که متد .score() را نیز دارد.**

**روش .score() یک امتیاز (متریک) را برای اندازه گیری عملکرد پیش بینی محاسبه می کند.**

****

| **Type of class** | **Required methods** |
| --- | --- |
| Estimator | .fit() |
| Transformer | .fit(), .transform(), .fit\_transform() |
| Predictor | .fit(), .predict() |
| Model | .fit(), .predict(), .score() |

**مرحله پیش پردازش شامل کار با تبدیل کننده ها است و ما در مرحله مدل سازی با پیش بینی کننده ها (به طور خاص با مدل ها) کار می کنیم.**

**آشنایی با Dataset**

داده ها در یک فایل csv. یا .xlsx موجود است.

ما فایل را از یک ادرس با استفاده از تابع ()pd.read\_csv بارگذاری می کنیم.

برخی از مشکلاتی را که باید برای دیتا ست ها قبل مدلسازی حل کنیم :

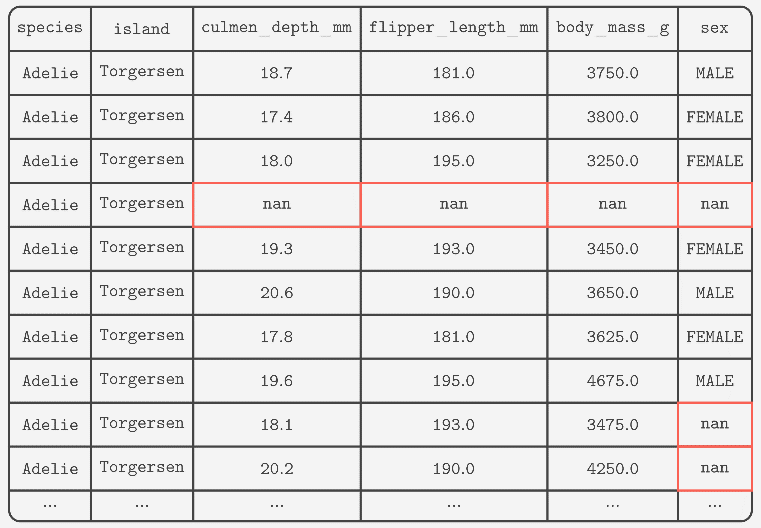
* داده های از دست رفته Missing data -
* متغیرهای طبقه بندی شده Categorical variables -
* مقیاس های مختلف - Different scales

Missing data

بیشتر الگوریتم‌های ML نمی‌توانند مقادیر از دست رفته را به طور خودکار مدیریت کنند، بنابراین قبل از اینکه مجموعه آموزشی را به یک مدل برسانیم، باید آن‌ها را حذف کنیم (یا با مقادیری جایگزین کنیم که به آن imputing می‌گویند).

کتابخانه pandas ، سلول های خالی جدول را با NaN پر می کنند .

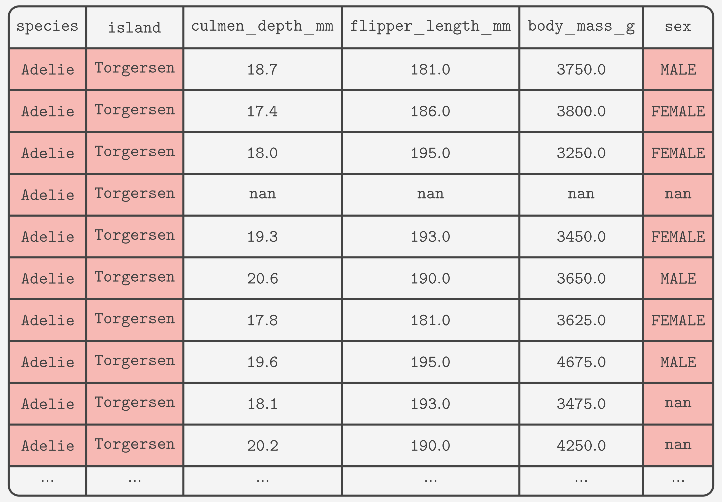
اگر حداقل یک NaN در داده ها وجود داشته باشد، اکثر مدل های ML با خطا مواجه می شوند.



Categorical data

داده‌ها حاوی داده‌های category هستند که قبلاً می‌دانیم مدل‌های یادگیری ماشین نمی‌توانند آن‌ها را مدیریت کنند.

بنابراین باید داده های طبقه بندی شده را به عددی رمزگذاری کنیم.



Different scales

مقادیر «culmen\_depth\_mm» از 13.1 تا 21.5 متغیر است در حالی که مقادیر «body\_mass\_g» از 2700 تا 6300 متغیر است. به همین دلیل، برخی از مدل‌های ML ویژگی «body\_mass\_g» را بسیار مهم‌تر از «culmen\_depth\_mm» می‌دانند.

**Scaling** solves this problem

Dealing with Missing Values

فقط تعداد کمی از مدل‌های یادگیری ماشین داده‌های با مقادیر گمشده را تحمل می‌کنند. بنابراین باید اطمینان حاصل کنیم که داده‌های ما حاوی مقادیر گمشده نیستند. اگر این کار را کرد، می توانیم:

* ردیف حاوی مقادیر از دست رفته را حذف کنید.
* سلول های خالی را با مقداری پر کنید. به آن انتساب (imputing) نیز می گویند.

برای بررسی اینکه آیا مجموعه داده شما مقادیری ندارد ، می توانید از متد info() DataFrame استفاده کنید.

Null is another Name for missing values.(NAN)

**حذف ردیف ( نمونه )** : ما می‌توانیم با خیال راحت ردیف‌هایی را حذف کنیم که اطلاعات بسیار کمی را به ما میدهند .

**نسبت دادن به ردیف** : ردیف‌های دیگر که حاوی اطلاعات بسیار مفیدتری هستند و فقط حاوی NaN هستند، به جای حذف کامل آنها ما فقط می توانیم مقادیری را برای سلول های NaN در نظر بگیریم. اغلب با استفاده از تبدیل کننده SimpleImputer به دست می آید.

Imputing Challenge

SimpleImputer مقادیر از دست رفته را با یک مقدار منفرد خاص جایگزین می کند (برای هر ستون مقدار واحد خودش).

اما باید مقداری را انتخاب کنیم که به آن نسبت داده شود.

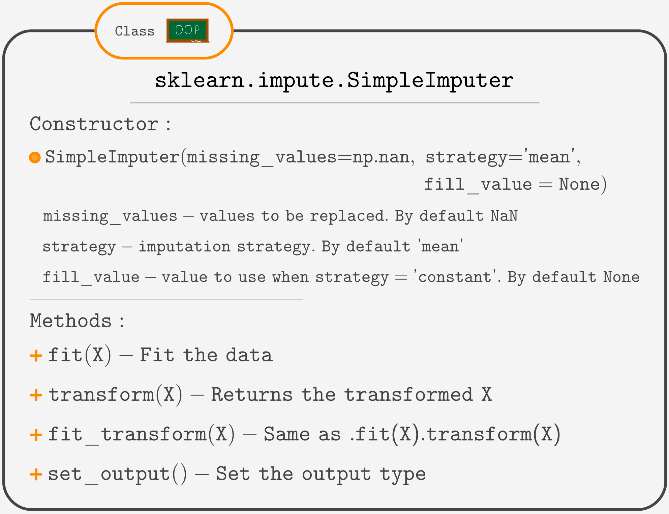
انتخاب رایج این است که میانگین را برای مقادیر عددی و حالت (متداول ترین مقدار) را برای مقادیر طبقه بندی کنیم .

 mean for numerical values and mode (most frequent value) for categorical values

می توان آن را با استفاده از پارامترهای استراتژی کنترل کرد :

* strategy='mean' – impute with mean along each column
* strategy='median' – impute with median along each column
* strategy='most\_frequent' – impute with mode along each column (مقدار mode)
* strategy='constant' – impute with constant number specified in fill\_value parameter

پارامتر missing\_values ​​کنترل می کند که چه مقادیری گم شده در نظر گرفته می شوند. به طور پیش فرض، NaN است، اما در مجموعه داده های مختلف، می تواند یک رشته خالی '' یا هر چیز دیگری باشد.



SimpleImputer و بسیاری از ترانسفورماتورهای دیگر با series پاندا کار نمی کنند، فقط با DataFrame کار می کنند.

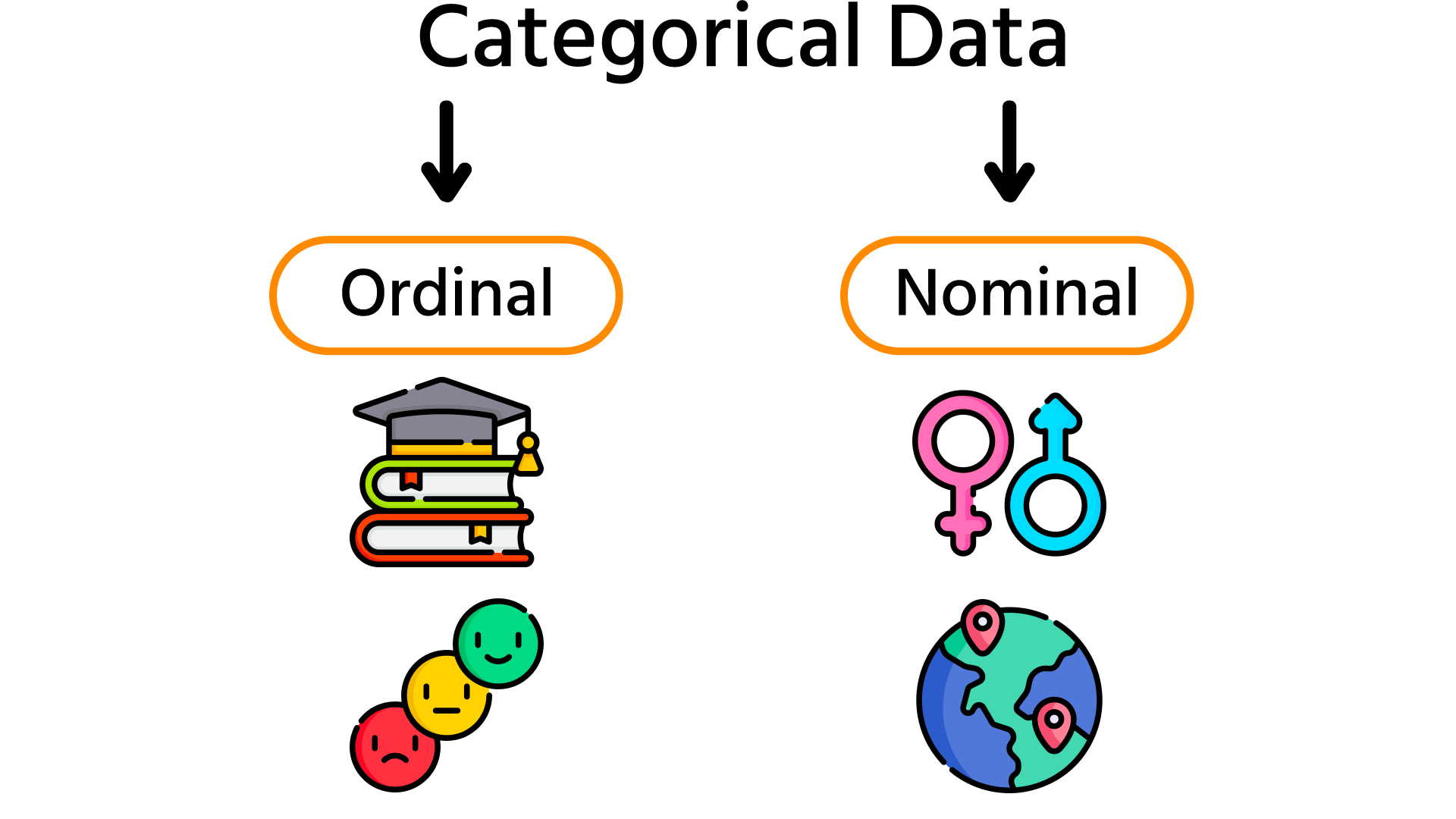
اما انتخاب یک ستون از یک DataFrame (df['column']) یک سری را برمی‌گرداند. برای جلوگیری از آن، می‌توانید از df[['column']] (به دو براکت توجه کنید) مانند این استفاده کنید

هنگامی که از متد .fit\_transform() SimpleImputer استفاده می کنید، یک آرایه دوبعدی برمی گرداند، اما pandas هنگام اختصاص دادن به ستون DataFrame انتظار یک آرایه (یا یک سری) یک بعدی را دارند.

برای حل این مشکل، می‌توانید از متد ravel() برای تبدیل آرایه دو بعدی به یک آرایه ۱ بعدی استفاده کنید

عالی! ما با مشکل مقادیر از دست رفته در مجموعه داده خود برخورد کردیم. ما ردیف‌هایی را که بیش از یک عدد تهی داشتند حذف کردیم و ستون «جنس» را با بیشترین مقدار - MALE نسبت دادیم.

OrdinalEncoder(Categorical Data Type)



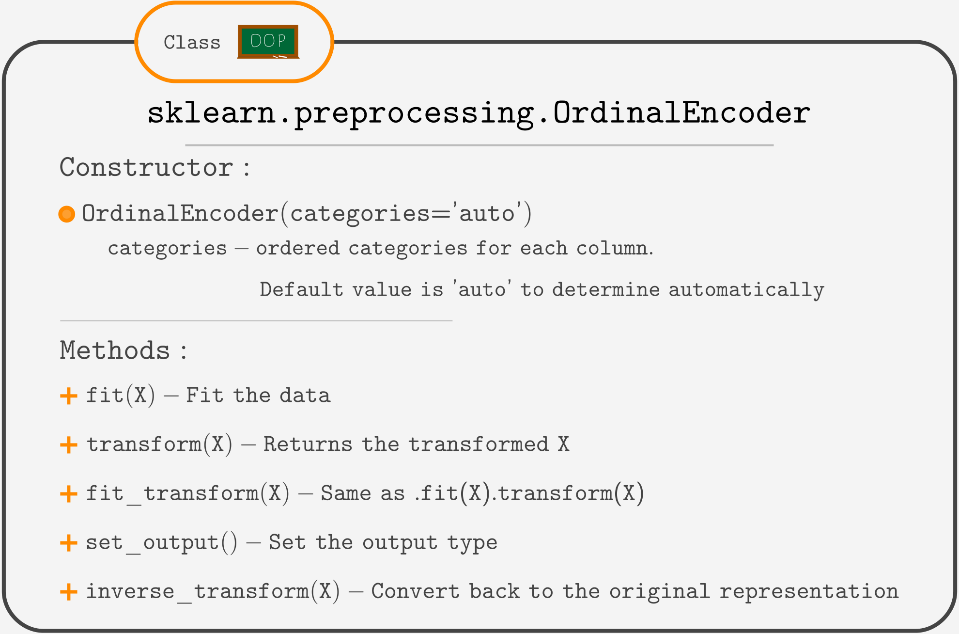
داده های ترتیبی (ordinal)از نظم طبیعی پیروی می کنند، در حالی که اسمی (nomial) این گونه نیست.

از آنجایی که یک نظم طبیعی وجود دارد، می‌توانیم دسته‌ها را به اعداد در آن ترتیب رمزگذاری کنیم.

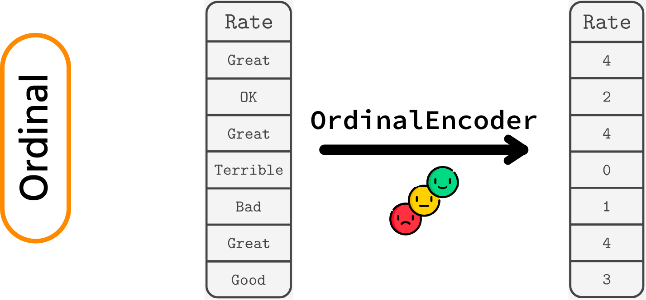
برای مثال، ستون «نرخ» حاوی مقادیر «وحشتناک»، «بد «خوب» و «عالی» را رمزگذاری می‌کنیم:

'Bad' – 1  
'OK' – 2  
'Good' – 3  
'Great' – 4

برای رمزگذاری داده های ترتیبی از OrdinalEncoder استفاده می شود. این فقط دسته ها را به 0، 1، 2، ... رمزگذاری می کند.



استفاده از OrdinalEncoder مانند هر ترانسفورماتور دیگری آسان است. تنها مشکل این است که آرگومان دسته ها را به درستی مشخص کنید.



OrdinalEncoder بیشتر برای تبدیل ویژگی ها (متغیر X) استفاده می شود. و متغیر X معمولا یک DataFrame حاوی بیش از 1 ستون است.

به همین دلیل، آرگومان دسته‌ها اجازه می‌دهد تا لیست دسته‌ها را برای هر ستون مشخص کنید،

اگر می‌خواهید فقط 1 ستون را تغییر دهید، همچنان باید فهرستی حاوی فهرست دیگری ارسال کنید،

همچنین به همین دلیل است که متد .fit\_transform() انتظار DataFrame را دارد و با Series کار نمی کند، بنابراین شما باید df[['column']] را برای تبدیل تنها یک ستون ارسال کنید.

One-Hot Encoder(nomial type)

داستان با مقادیر اسمی کمی پیچیده تر است.

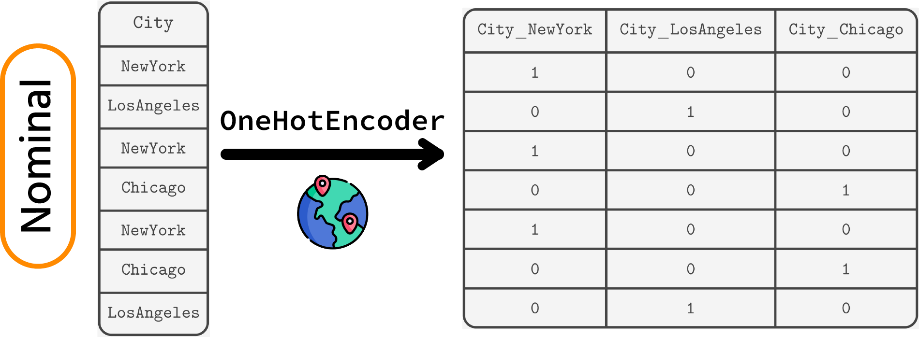
فرض کنید این ویژگی حاوی داده های ترتیبی است، به عنوان مثال، نرخ های کاربر. مقادیر آن وحشتناک، بد، خوب، خیلی خوب و عالی است. رمزگذاری آن مقادیر به صورت 0 تا 4 منطقی به نظر می رسد.

مدل ML ترتیب را در نظر می گیرد. حال، ویژگی «شهر» را تصور کنید که شامل پنج شهر مختلف است. اگر آنها را از 0 تا 4 رمزگذاری کنیم، حالت ML نیز فکر می کند که یک نظم منطقی وجود دارد، اما وجود ندارد.

برای رمزگذاری داده های اسمی، از ترانسفورماتور OneHotEncoder استفاده می شود.

برای هر مقدار منحصر به فرد یک ستون ایجاد می کند.

سپس برای هر سطر، 1 را به ستون مقدار این سطر و 0 را برای ستون های دیگر تنظیم می کند.



آنچه در ابتدا "NewYork" بود، اکنون دارای 1 در ستون City\_NewYork و 0 در سایر ستون‌های City\_ است.

برای استفاده از OneHotEncoder، فقط باید یک شی را مقداردهی اولیه کنید و مانند هر ترانسفورماتور دیگری، ستون ها را به .fit\_transform() ارسال کنید.

OneHotEncoder یک ماتریس پراکنده را برمی گرداند، که می تواند با استفاده از روش .toarray() به آرایه NumPy تبدیل شود .

LabelEncoder

OrdinalEncoder و OneHotEncoder معمولا برای رمزگذاری ویژگی ها (متغیر X) استفاده می شوند.

اما هدف (متغیر y) نیز می تواند طبقه بندی شود.

LabelEncoder برای رمزگذاری هدف، صرف نظر از اینکه اسمی یا ترتیبی باشد، استفاده می شود.

مدل‌های ML ترتیب هدف را در نظر نمی‌گیرند و به آن اجازه می‌دهند به عنوان هر مقدار عددی کدگذاری شود.

LabelEncoder هدف را روی اعداد 0، 1، ... رمزگذاری می کند.

هدف را با استفاده از LabelEncoder کد می کند و سپس از متد ()inverse\_transform برای تبدیل آن به نمایش اصلی استفاده می کند.

از آنجایی که LabelEncoder برای تبدیل هدف (y)، که معمولاً یک ستون است، استفاده می شود، برخلاف OrdinalEncoder، با pandas series به خوبی کار می کند.

بنابراین ما فقط می توانیم متغیر y را به متد .fit\_transform() پاس کنیم.