Data preprocessing using Scikit-learn

→ import sklearn

standard format refers to data that has 0 mean and unit variance (ie standard deviation = 1). The process of converting data into this format is called data standardisation.

$$z = \frac{x - \mu}{\sigma}$$

standardised

$z \rightarrow$ new data value $x \rightarrow$ old data value
$\mu \rightarrow$ overall mean $\sigma \rightarrow$ standard deviation

```
from sklearn.preprocessing import scale
col_standardized = scale (pizza_data)
```

scale also takes the axis parameter.

Range scaling

This is a two step process. For a given data value, x, we first compute the proportion of the value with respect to the min and max of the data dmin and dmax

$$x_{prop} = \frac{x - dmin}{dmax - dmin}$$

This only works if not all data values are same ie $dmax \neq dmin$

We then use the proportion of the value to scale to the specified range $[r_{min}, r_{max}]$

$$x_{scale} = x_{prop} \cdot (r_{max} - r_{min}) + r_{min}$$

→ from sklearn.preprocessing import MinMaxScaler
default_scaler = MinMaxScaler()
transformed = default_scaler.fit_transform(data)

custom_scaler = MinMaxScaler(feature_range=(-2,3))
transformed = custom_scaler.fit_transform(data)

Robust scaling

→ Outliers → An outlier, is a data point that is significantly farther away from the other data points.

To avoid being affected by outliers, using the data's median and interquartile range

from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
transformed = robust_scaler.fit_transform(data)

Normalising data

→ L2 normalisation
All the previous techniques are applied to columns. But sometimes, we need to scale individual data observations (ie. rows)

For instance, when clustering, we need to apply L2 normalisation to each row in order to calculate cosine similarity scores.

In general terms L2 norm of a row is just the square root of the sum of squared values of the row.

$$x = [x_1, x_2 \dots x_m]$$

$$x_{l2} = \left[\frac{x_1}{l}, \frac{x_2}{l} \dots \frac{x_m}{l}\right], \quad l = \sqrt{\sum_{i=1}^{m} x_i^2}$$

(l: L2 norm)

→
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
transformed = normalizer.fit_transform(data)
```

Data imputation.

If only few values are missing, we can perform data imputation to substitute the missing data

The four methods are
- Using the mean value
- Using the median
- Using the most frequent value
- Using a constant value

→ default method is using mean

→
```
from sklearn.impute import SimpleImputer
imp_mean = SimpleImputer()
transformed = imp_mean.fit_transform(data)
```

using the strategy keyword, we can change the behaviour.
strategy = 'median' || 'most_frequent'

for a constant value,
SimpleImputer(strategy='constant', fill_value=-1)

→ These are more advanced imputation methods such as K-nearest neighbours and MICE but are rarely used in industries since data is cleaned.

PCA - Principal component analysis

Used for reducing the features in the dataset which are less important.

PCA extracts the principal components of the dataset, which are uncorrelated set of latent variables that encompass most of the information from the original dataset.

→ we can use the n_components keyword to specify the number of principal components. default setting is to extract m-1 principal components

→ from sklearn.decomposition import PCA
pca_obj = PCA()
pca_obj = PCA(n_components=3)
pc = pca_obj.fit_transform(data).round(3)

Labelled data

A big part of data science is to classify a data set into seperate categories. For example, we can classify a dataset of breast tumors as either malignant or benign.