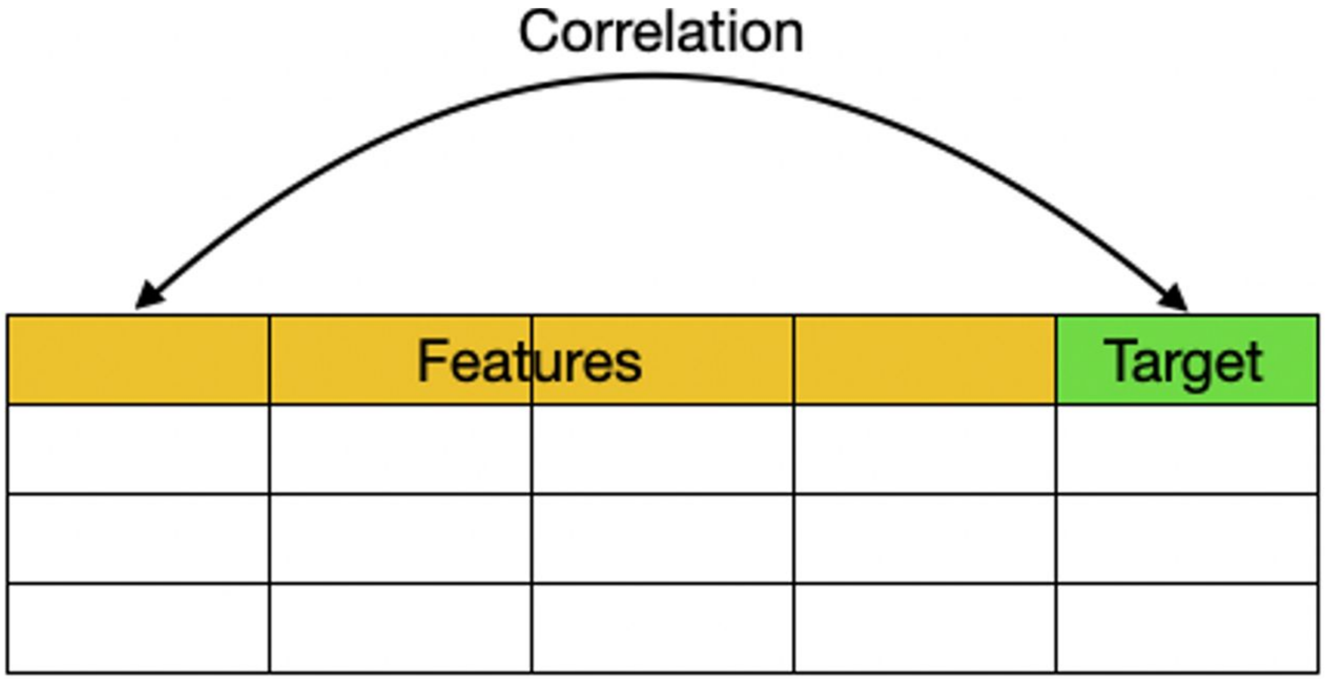
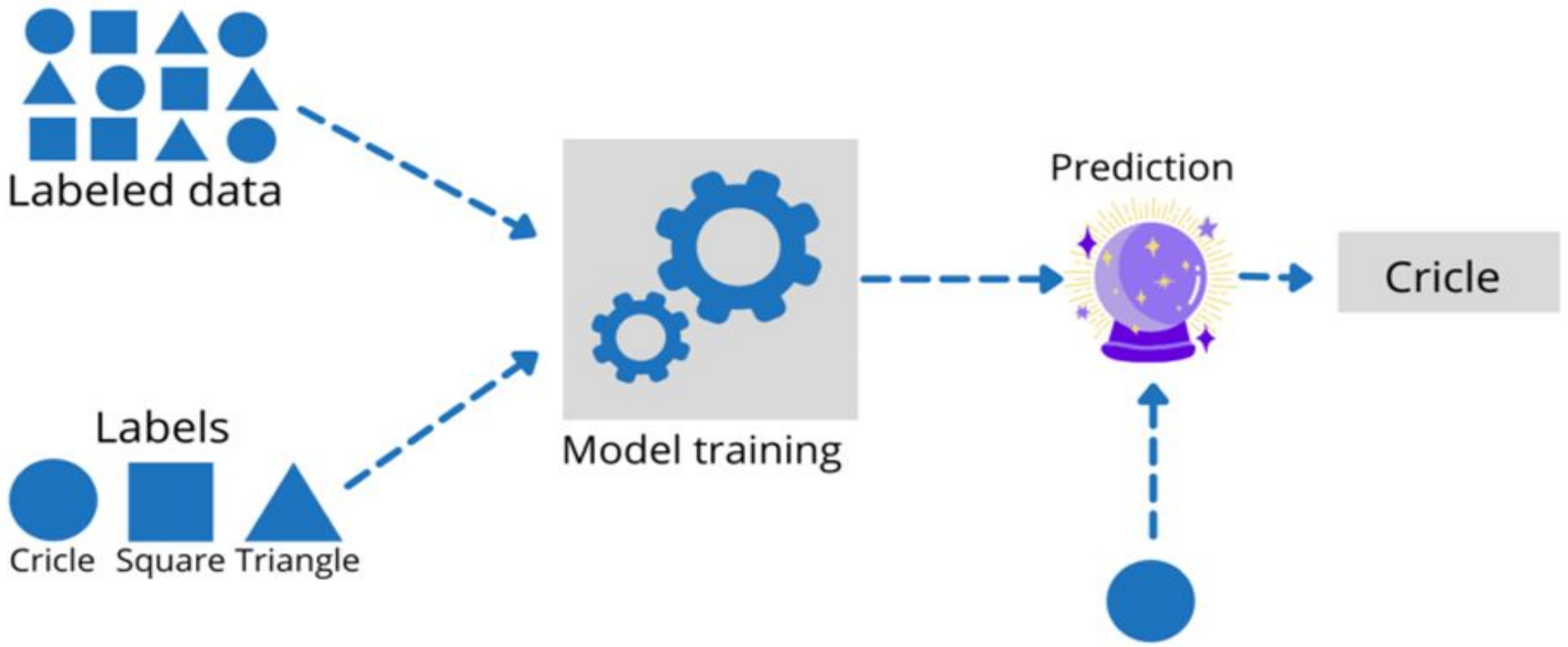
MACHINE LEARNING

Features Vs Target



TYPES OF MACHINE LEARNING

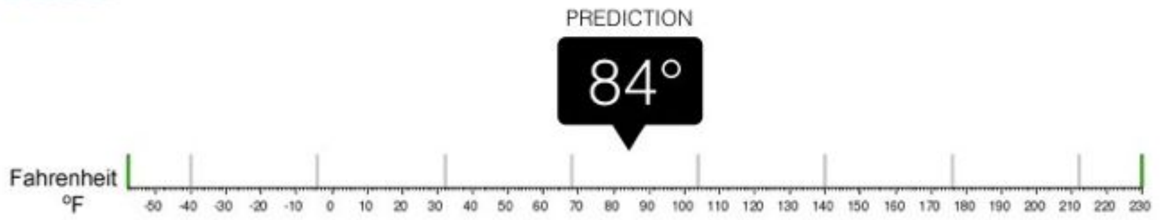
1. SUPERVISED LEARNING



Branches of Supervised Learning

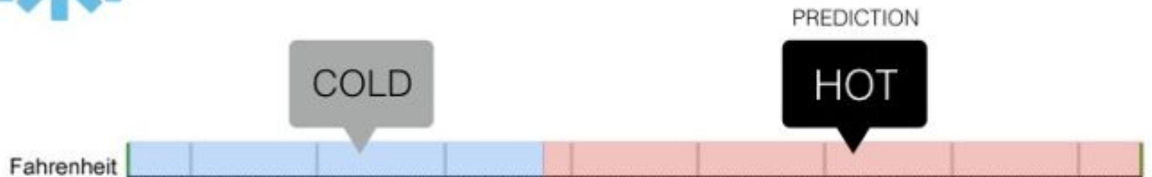
Regression

What is the temperature going to be tomorrow?

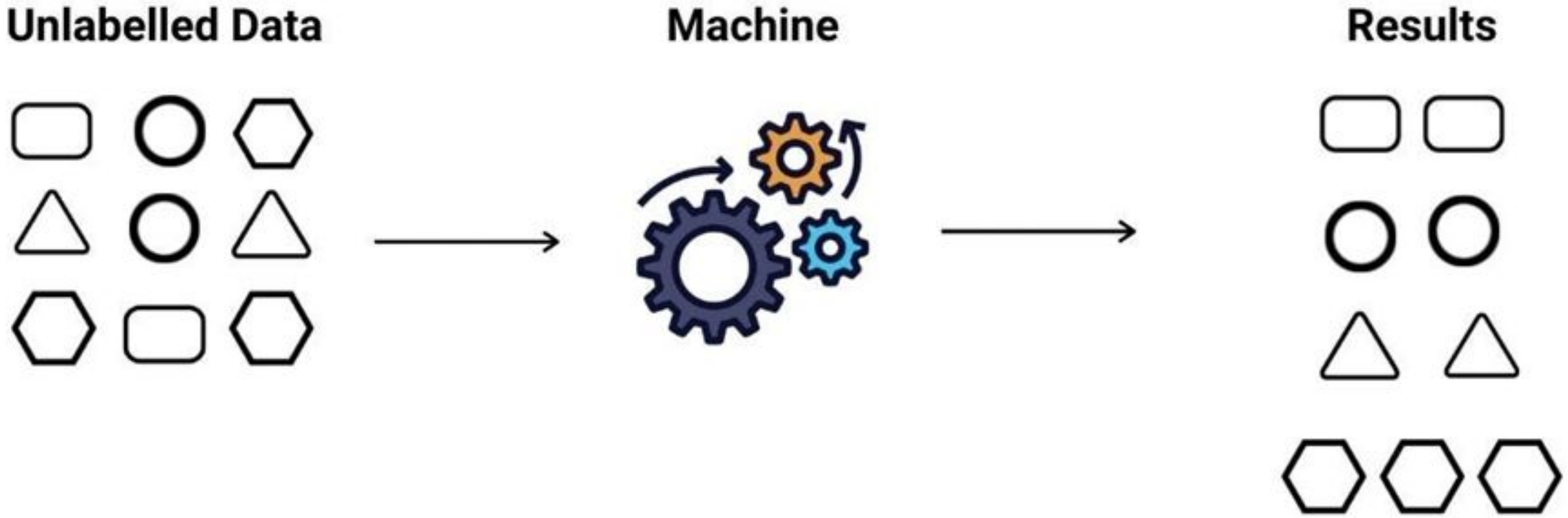


Classification

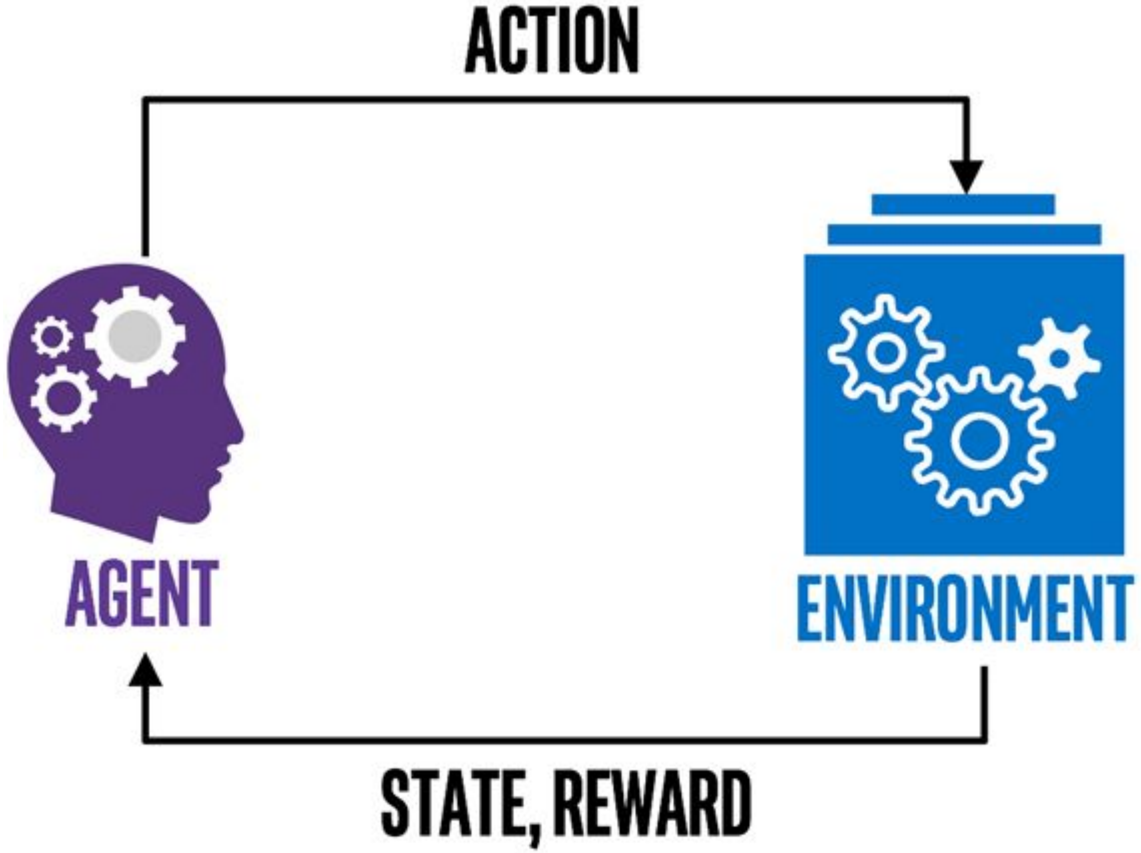
Will it be Cold or Hot tomorrow?



2. UNSUPERVISED LEARNING



3. REINFORCEMENT LEARNING



Scikit-learn

- Preprocessing tools
- Feauture selection
- train, test, split
- Algorithms
- Model evaluation

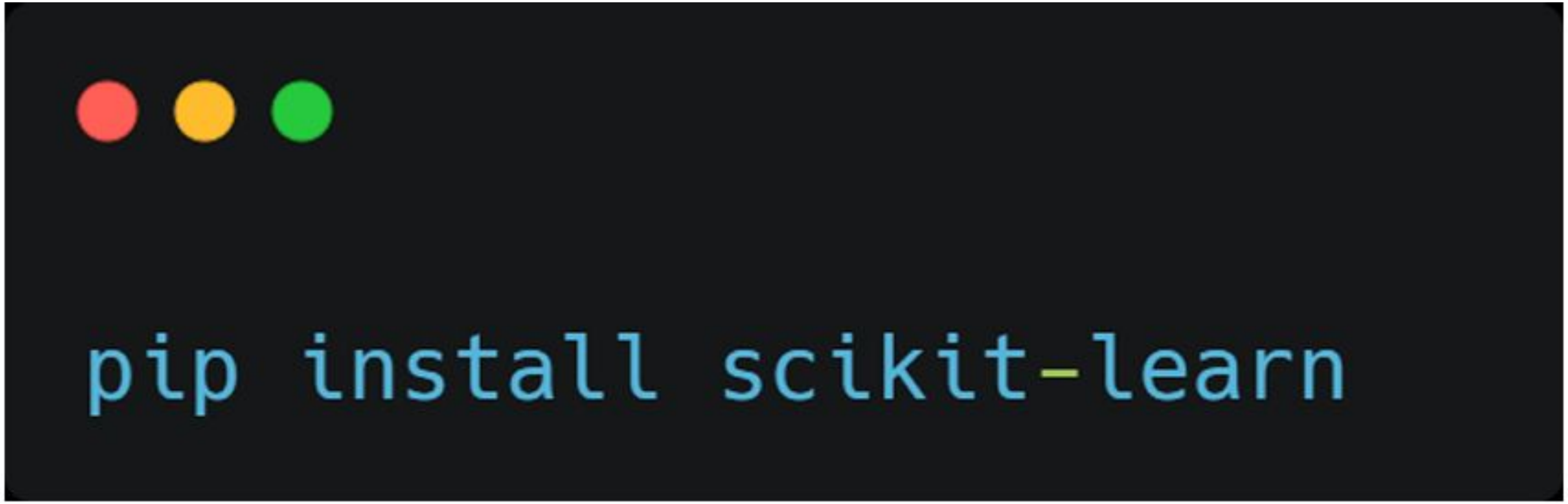
SCIKIT-LEARN

- Scikit-learn, also known as sklearn, is a popular open-source machine learning library in Python that provides a wide range of tools for data analysis, modeling, and evaluation.

- Sklearn is built on top of NumPy, SciPy, and Matplotlib, and supports integration with Pandas, which makes it easy to use in data science workflows.

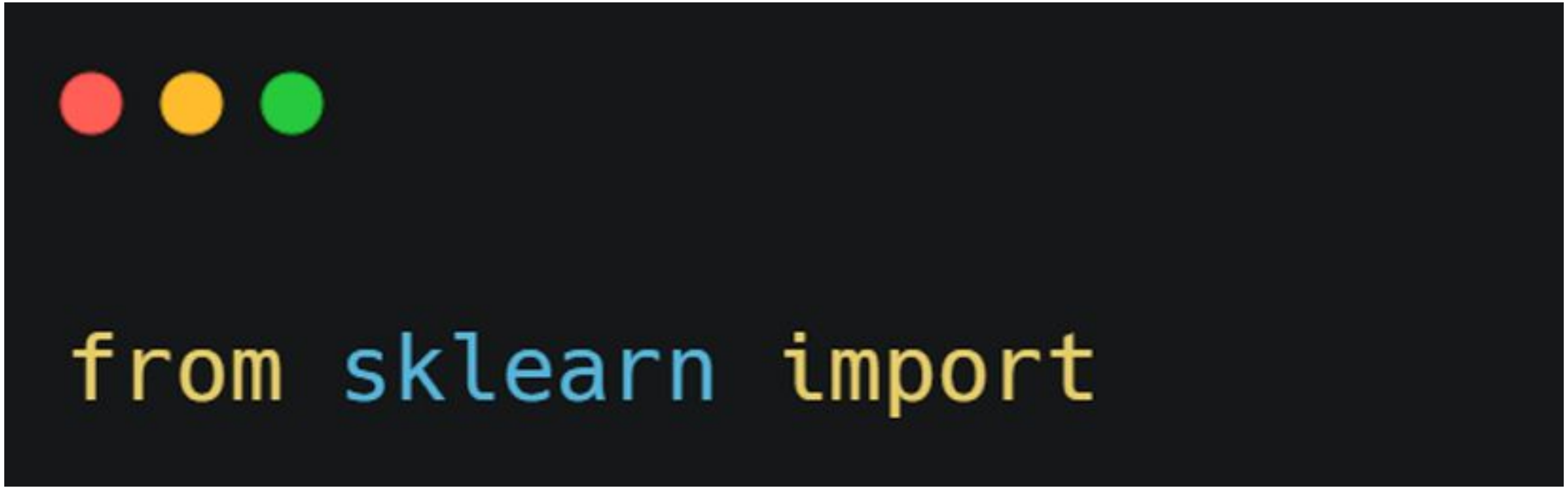
- Sklearn is widely used in the data science community for various applications such as predictive modeling, natural language processing, computer vision, and time series forecasting, among others.

INSTALLATION



```
pip install scikit-learn
```

IMPORT



```python
from sklearn import
```

PREPROCESSING

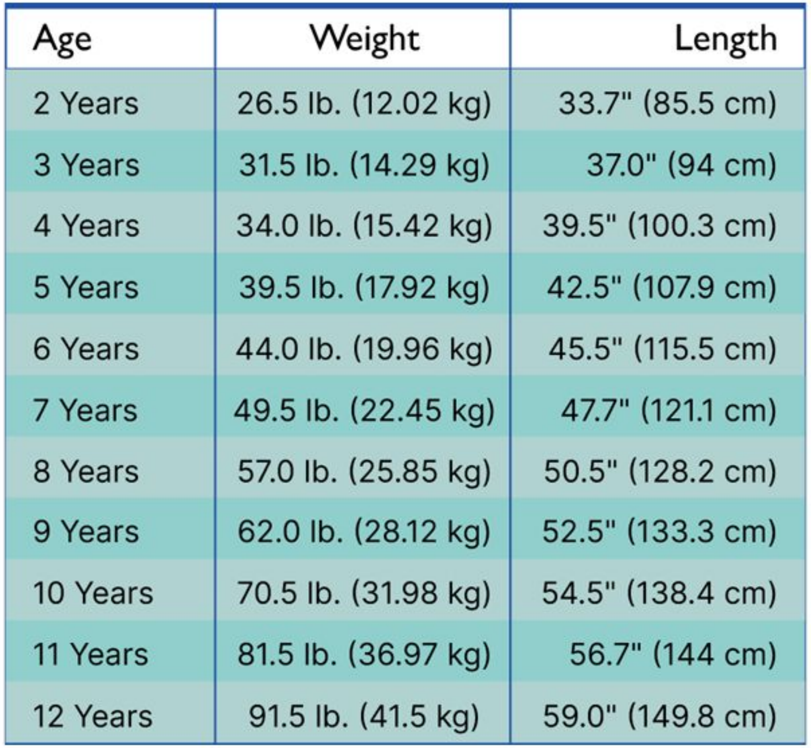
- Feature Scaling
- Encoding
- Imputing null values
- Outlier - detection & Handlin

FEATURE SCALING

- Feature scaling is a method used to normalize the range of features of data.
- Feature Scaling involves modifying values by methods like **Normalization or Standardization.**
- It helps to avoid bias in machine learning model.

WHY SCALING?

- When dataset has numerical features and each of them are in different scale.
- ML model can put weight on features with larger scale.
- Scaling helps to contribute all features equally.



Age	Weight	Length
2 Years	26.5 lb. (12.02 kg)	33.7" (85.5 cm)
3 Years	31.5 lb. (14.29 kg)	37.0" (94 cm)
4 Years	34.0 lb. (15.42 kg)	39.5" (100.3 cm)
5 Years	39.5 lb. (17.92 kg)	42.5" (107.9 cm)
6 Years	44.0 lb. (19.96 kg)	45.5" (115.5 cm)
7 Years	49.5 lb. (22.45 kg)	47.7" (121.1 cm)
8 Years	57.0 lb. (25.85 kg)	50.5" (128.2 cm)
9 Years	62.0 lb. (28.12 kg)	52.5" (133.3 cm)
10 Years	70.5 lb. (31.98 kg)	54.5" (138.4 cm)
11 Years	81.5 lb. (36.97 kg)	56.7" (144 cm)
12 Years	91.5 lb. (41.5 kg)	59.0" (149.8 cm)

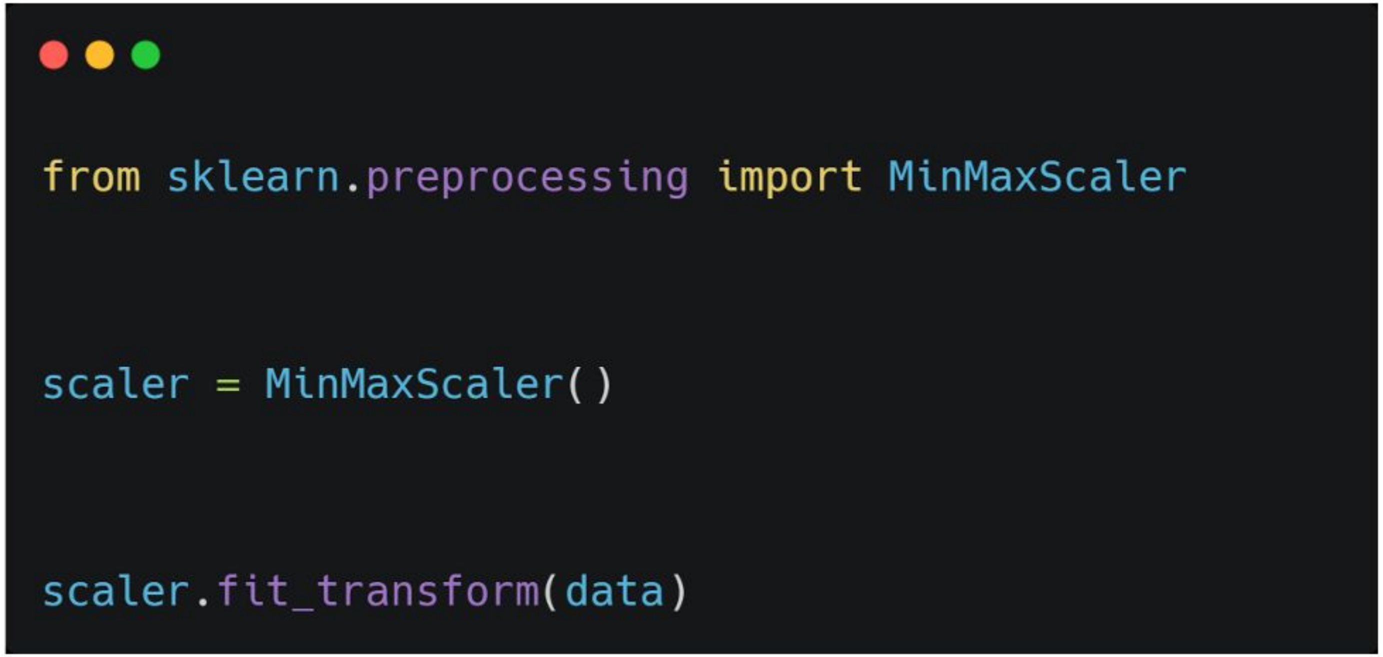
NORMALIZATION

It is the method of scaling the data by fitting the data points between a range of 0 to 1.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

MIN-MAX SCALER

MinMaxScaler from sklearn perform normalization



```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit_transform(data)
```

STANDARDIZATION

This converts all the data points to have a mean value of 0 and standard deviation of 1

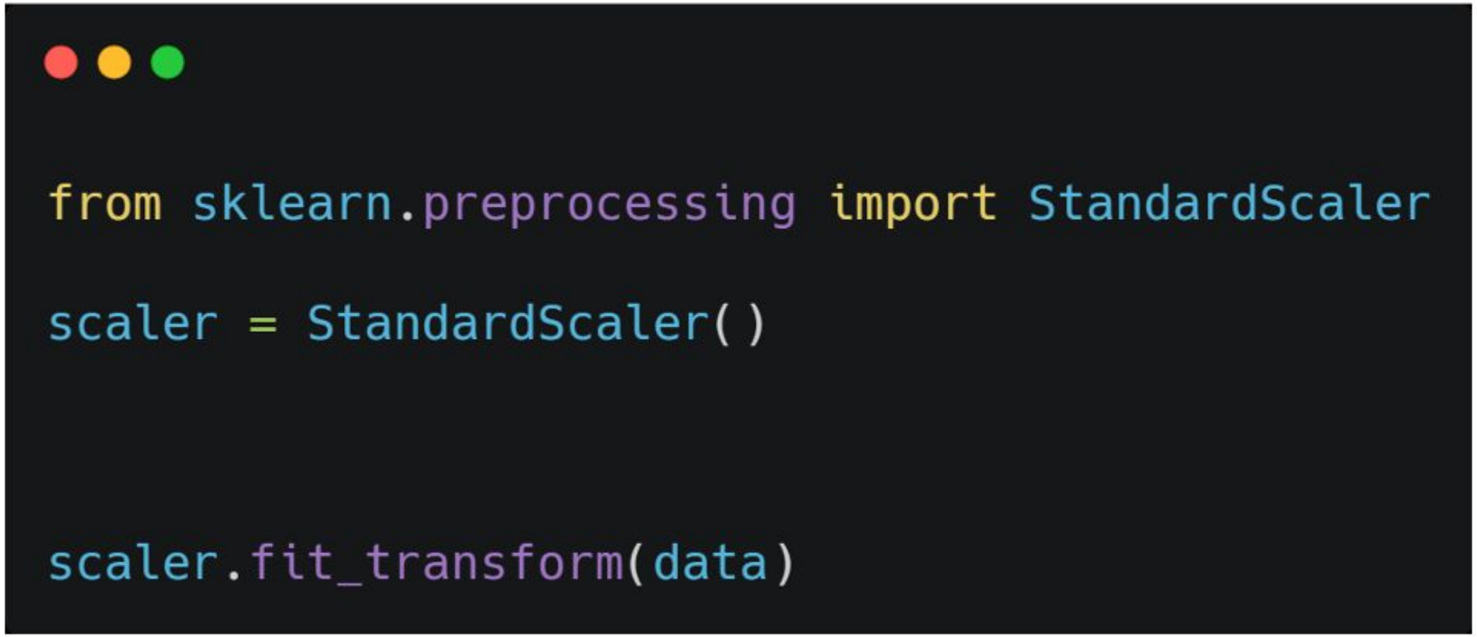
$$Z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean

$\sigma =$ Standard Deviation

STANDARD SCALER

StandardScaler from sklearn perform standardization



```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()


scaler.fit_transform(data)
```

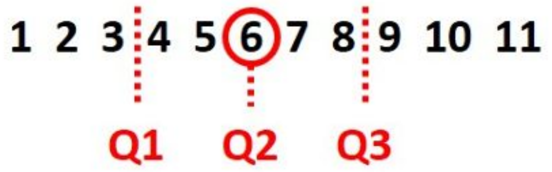
ROBUST SCALER

This uses interquartile range so

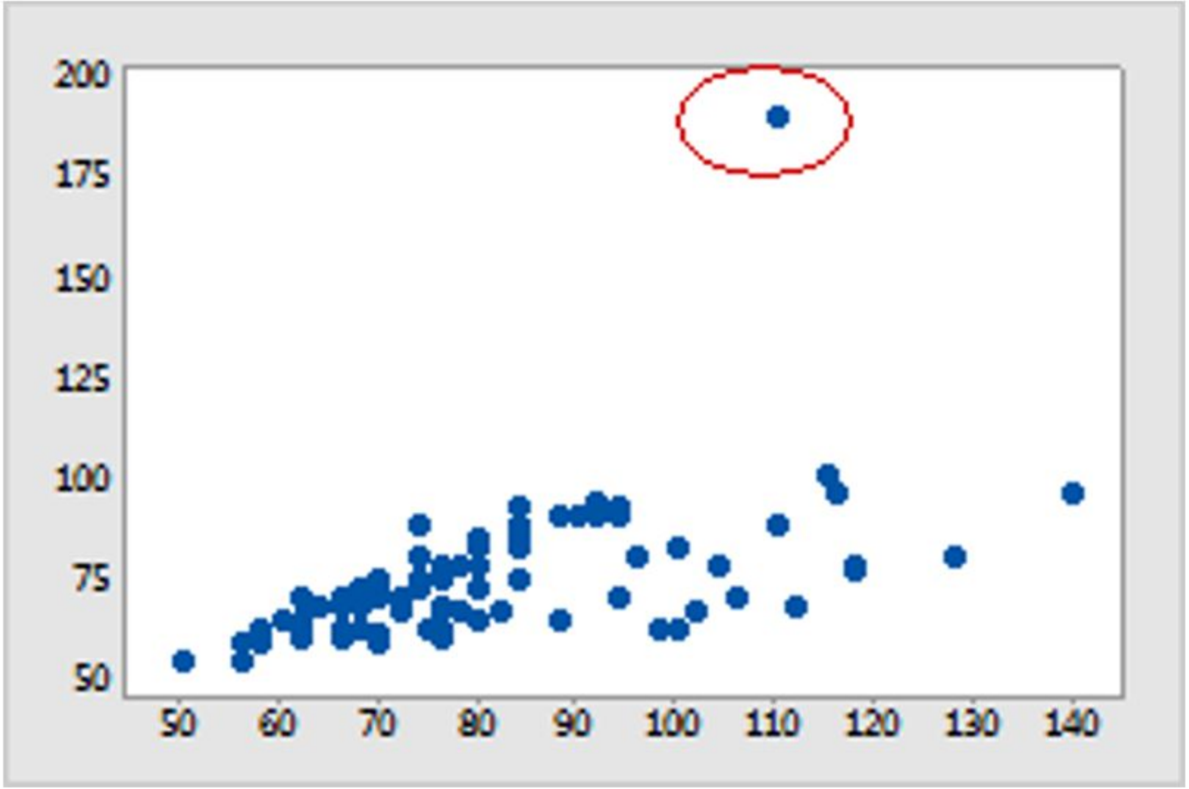
that it is robust to outliers

$$\boxed{X_{new} = \frac{X - X_{median}}{IQR}}$$

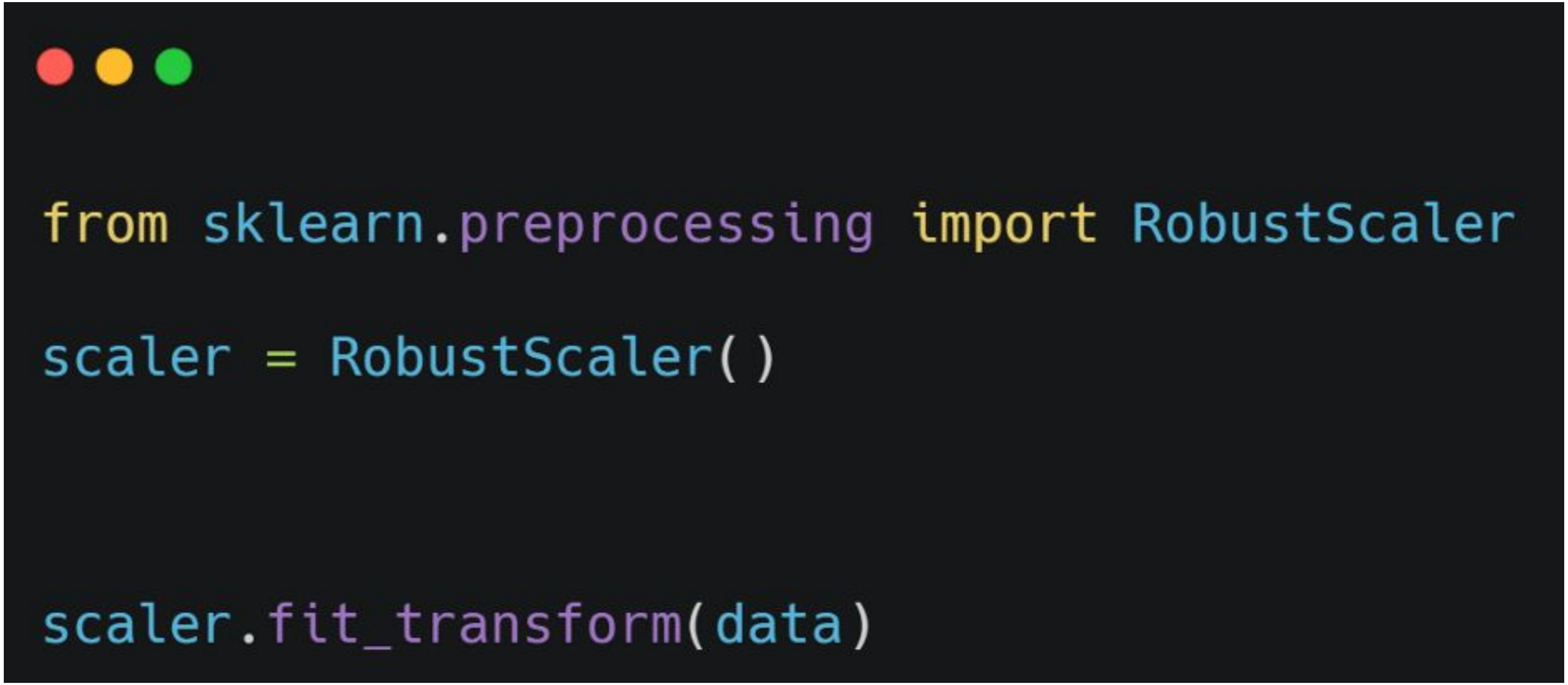
$$\begin{aligned} IQR &= Q3 - Q1 \\ &= 8.5 - 3.5 \\ &= 5 \end{aligned}$$



ROBUST SCALER



ROBUST SCALER



```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()



scaler.fit_transform(data)
```

WHICH IS BETTER?

Normalization:

- Useful when the data doesn't follow gaussian(normal) distrubution
- Useful in algorithms like KNN, and Neural networks like CNN, ANN

Standardization:

- When your data follows gaussian distribution

Robust Scaler:

- When your data has outliers

ENCODING

- Machine learning models can only work with numerical values.
- For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones.
- This process is called feature encoding.

TYPES OF ENCODING

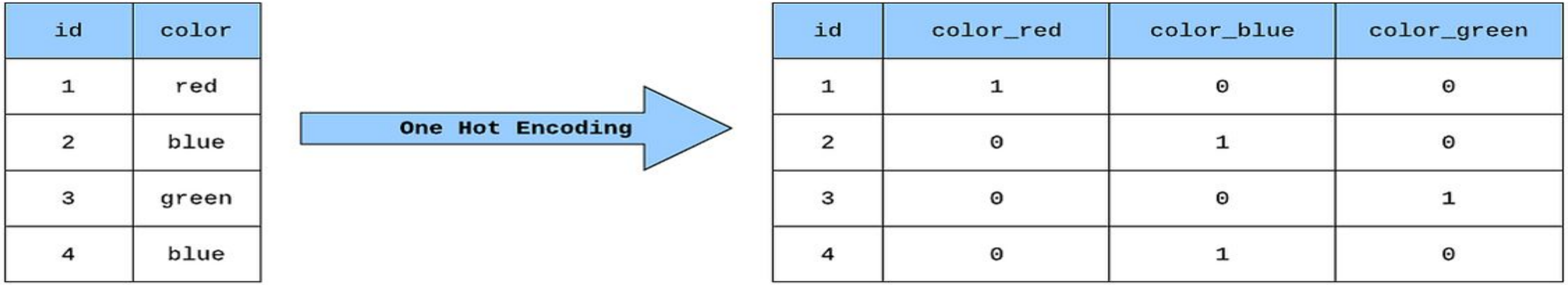
1. Nominal encoding :

- Represent data without any order or hierarchy
- It can be done with **OneHotEncoder**

2. Ordinal Encoding :

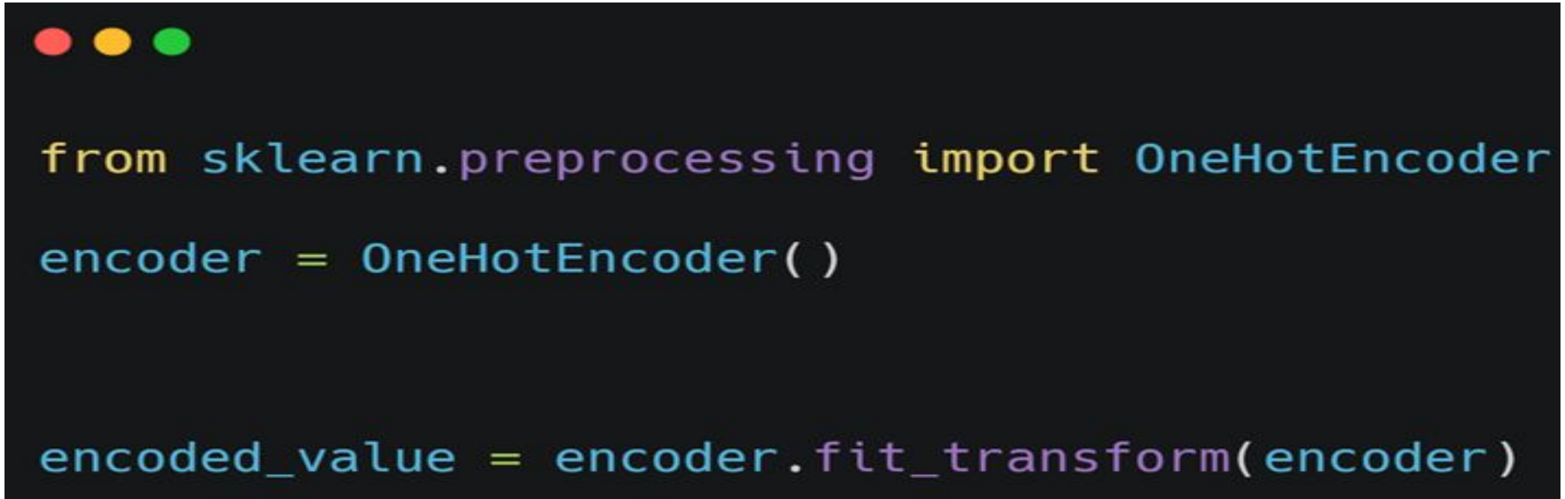
- Assigning unique integer based on rank/order
- It can be done with **LabelEncoder**

ONEHOT ENCODER



id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0



```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()


encoded_value = encoder.fit_transform(encoder)
```

LABEL ENCODER

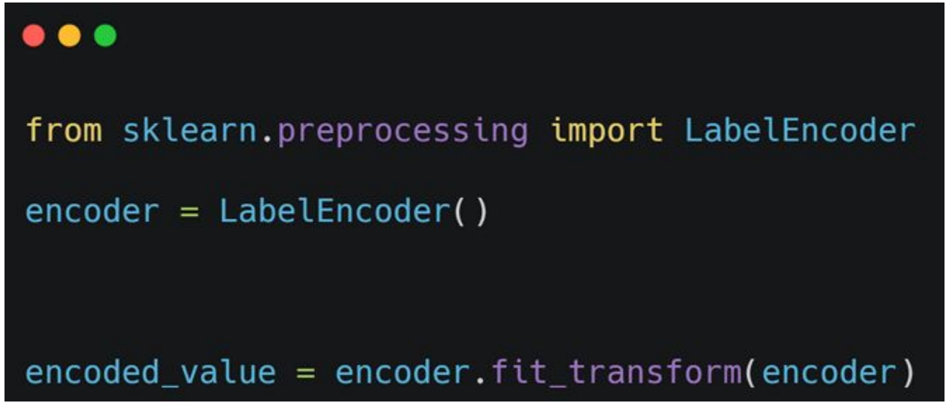
Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

$\longrightarrow$

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29



```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()


encoded_value = encoder.fit_transform(encoder)
```

END