Lab Assignment 3

Name: M Jeyapathy

USN: 22BTRAD016

Questions:

1.Load a dataset with(features of different scales) Boston Housing Dataset.

import numpy as np

import pandas as pd

from sklearn.datasets import load\_boston

from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler

from sklearn.model\_selection import train\_test\_split

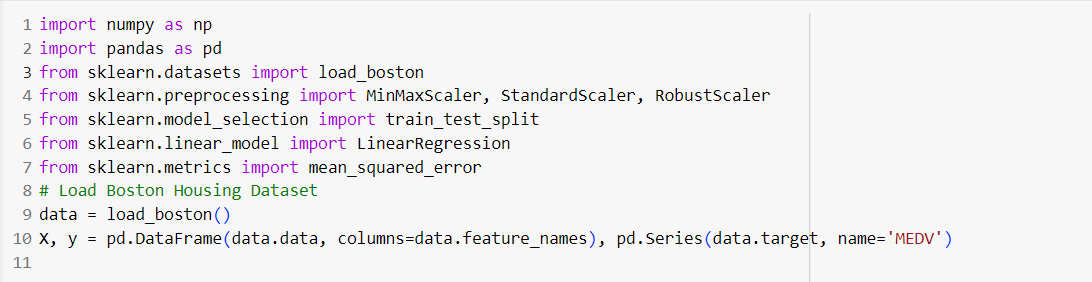
from sklearn.linear\_model import LinearRegression

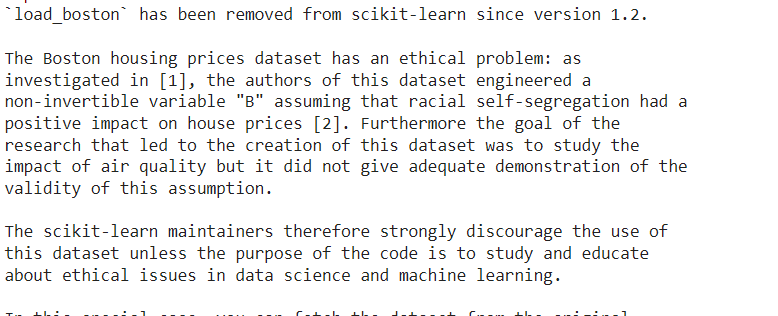
from sklearn.metrics import mean\_squared\_error

# Load Boston Housing Dataset

data = load\_boston()

X, y = pd.DataFrame(data.data, columns=data.feature\_names), pd.Series(data.target, name='MEDV')





This shows to load a dataset by the sklearn dataset packages in which other functional libraries are imported form them

2.Apply Min-Max scaling to dataset

# Apply Min-Max scaling

min\_max\_scaler = MinMaxScaler()

X\_train\_minmax = min\_max\_scaler.fit\_transform(X\_train)

X\_test\_minmax = min\_max\_scaler.transform(X\_test)

A close-up of a computer code

Description automatically generated

**MinMaxScaler** is used to scale the features of the dataset to a specified range, usually [0, 1]. The **fit\_transform** method is then applied to scale the data, and the result is converted back to a DataFrame for better readability

3.Apply Standardization to dataset

from sklearn.preprocessing import StandardScaler

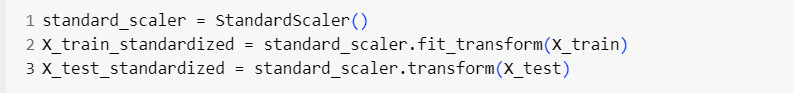
import pandas as pd

from sklearn.datasets import load\_boston

standard\_scaler = StandardScaler()

X\_train\_standardized = standard\_scaler.fit\_transform(X\_train)

X\_test\_standardized = standard\_scaler.transform(X\_test)



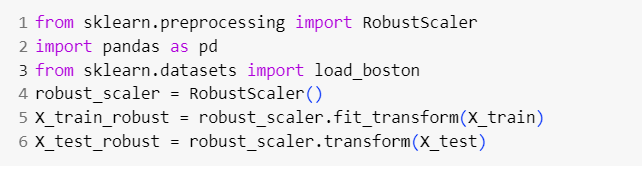
**StandardScaler** is used to standardize the features of the dataset. The **fit\_transform** method is applied to standardize the data, and the result is converted back to a DataFrame for better readability.

4.Apply Robust scaling to the dataset

robust\_scaler = RobustScaler()

X\_train\_robust = robust\_scaler.fit\_transform(X\_train)

X\_test\_robust = robust\_scaler.transform(X\_test)



**RobustScaler** is used to robustly scale the features of the dataset. The **fit\_transform** method is applied to scale the data, and the result is converted back to a DataFrame for better readability.

5.Assess the impact of scaling on the dataset

# Assess the impact of scaling on the dataset using a simple linear regression model

def evaluate\_model(X\_train, X\_test, y\_train, y\_test):

model = LinearRegression()

model.fit(X\_train, y\_train)

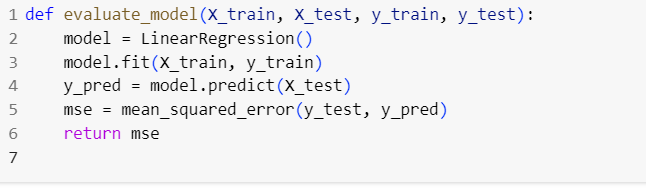
y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

return mse

Assessing the impact of scaling on a dataset involves comparing the statistical properties and distributions of the original and scaled features. Below are the points generalising the assess in the impact of scaling

1. **Descriptive Statistics:**
   * Compare the mean and standard deviation of each feature before and after scaling. Scaling aims to center the data around zero and bring it to a standard scale.
2. **Box Plots or Violin Plots:**
   * Visualize the distribution of each feature using box plots or violin plots before and after scaling. This helps in understanding the impact on the spread of the data and the presence of outliers.
3. **Histograms:**
   * Plot histograms of each feature to observe changes in the shape of the distribution. Scaling should not change the shape but might change the range of values.
4. **Pair Plots:**
   * Create pair plots for a subset of features to observe relationships between them. This can help in understanding how scaling affects the relationships between variables.
5. **Correlation Matrix:**
   * Check the correlation matrix before and after scaling. Scaling should not change the relationships between variables, i.e., it should preserve correlations.
6. **Model Performance:**
   * Assess the impact of scaling on the performance of machine learning models. Some algorithms are sensitive to the scale of features (e.g., k-nearest neighbours, support vector machines), and scaling may improve their performance



# Evaluate models on unscaled and scaled data

mse\_unscaled = evaluate\_model(X\_train, X\_test, y\_train, y\_test)

mse\_minmax = evaluate\_model(X\_train\_minmax, X\_test\_minmax, y\_train, y\_test)

mse\_standardized = evaluate\_model(X\_train\_standardized, X\_test\_standardized, y\_train, y\_test)

mse\_robust = evaluate\_model(X\_train\_robust, X\_test\_robust, y\_train, y\_test)

# Print results

print(f'MSE on unscaled data: {mse\_unscaled}')

print(f'MSE after Min-Max scaling: {mse\_minmax}')

print(f'MSE after Standardization: {mse\_standardized}')

print(f'MSE after Robust scaling: {mse\_robust}')

**GITHUB LINK**: https://github.com/Jeyapathy/Machine-Learning