Data Preparation

This notebook is for Data Cleaning and Feature Engineering

==

Data Dictionary

Field	Description

| CRIM| |per capita crime rate by town | ZN |proportion of residential land zoned for lots over 25,000 sq.ft | | INDUS |proportion of non-retail business acres per town | | CHAS |Charles River dummy variable | | NOX |nitric oxides concentration (parts per 10 million) | | RM|average number of rooms per dwelling | | AGE |proportion of owner-occupied units built prior to 1940 | | DIS| weighted distances to five Boston employment centres| | RAD|index of accessibility to radial highways | | TAX|full-value property-tax rate per 10,000 | | PTRATIO |pupil-teacher ratio by town | | B|1000(Bk - 0.63)^2 where Bk is the proportion of black people by town | | LSTAT | lower status of the population | | MEDV | Median value of owner-occupied homes in 1000's |

==

Import Libraries

In [1]:
```
import numpy as np
from numpy import count_nonzero, median, mean
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
#import squarify


import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder, Binar
from sklearn.preprocessing import OneHotEncoder, PolynomialFeatures, RobustScaler
from sklearn.datasets import load_boston

from sklearn.decomposition import PCA

%matplotlib inline
#sets the default autosave frequency in seconds
```

```
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)

plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

import warnings
warnings.filterwarnings('ignore')


pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

Data Quick Glance

In [2]:
```
boston = load_boston()
```

In [3]:
```
print(boston, sep='\n')
```

```
{'data': array([[  0.00632,  18.     ,   2.31   , ...,  15.3    , 396.9    ,
          4.98   ],
       [  0.02731,   0.     ,   7.07   , ...,  17.8    , 396.9    ,
          9.14   ],
       [  0.02729,   0.     ,   7.07   , ...,  17.8    , 392.83   ,
          4.03   ],
       ...,
       [  0.06076,   0.     ,  11.93   , ...,  21.     , 396.9    ,
          5.64   ],
       [  0.10959,   0.     ,  11.93   , ...,  21.     , 393.45   ,
          6.48   ],
       [  0.04741,   0.     ,  11.93   , ...,  21.     , 396.9    ,
          7.88   ]]), 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.
1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
```

```
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]), 'feature_
names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'), 'DESCR': ".. _boston_datase
t:\n\nBoston house prices dataset\n---------------------------\n\n**Data Set Charact
eristics:**  \n\n    :Number of Instances: 506 \n\n    :Number of Attributes: 13 num
eric/categorical predictive. Median Value (attribute 14) is usually the target.\n\n
   :Attribute Information (in order):\n        - CRIM     per capita crime rate by t
own\n        - ZN       proportion of residential land zoned for lots over 25,000 s
q.ft.\n        - INDUS    proportion of non-retail business acres per town\n
- CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n
      - NOX      nitric oxides concentration (parts per 10 million)\n        - RM
    average number of rooms per dwelling\n        - AGE      proportion of owner-occ
upied units built prior to 1940\n        - DIS      weighted distances to five Bosto
n employment centres\n        - RAD      index of accessibility to radial highways\n
        - TAX      full-value property-tax rate per $10,000\n        - PTRATIO  pupi
l-teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where Bk is the propor
tion of black people by town\n        - LSTAT    % lower status of the population\n
       - MEDV     Median value of owner-occupied homes in $1000's\n\n    :Missing At
tribute Values: None\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a
copy of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/ml/machine-learning-dat
abases/housing/\n\n\nThis dataset was taken from the StatLib library which is mainta
ined at Carnegie Mellon University.\n\nThe Boston house-price data of Harrison, D. a
nd Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Environ. Econo
mics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regressio
n diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the tab
le on\npages 244-261 of the latter.\n\nThe Boston house-price data has been used in
many machine learning papers that address regression\nproblems.   \n     \n.. topi
c:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying I
nfluential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R.
(1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Ten
th International Conference of Machine Learning, 236-243, University of Massachusett
s, Amherst. Morgan Kaufmann.\n", 'filename': 'boston_house_prices.csv', 'data_modul
e': 'sklearn.datasets.data'}
```

In [4]:
```
df = pd.DataFrame(boston.data, columns=boston.feature_names)
```

In [5]:
```
df
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.01	18.00	2.31	0.00	0.54	6.58	65.20	4.09	1.00	296.00	15.30	396.90
1	0.03	0.00	7.07	0.00	0.47	6.42	78.90	4.97	2.00	242.00	17.80	396.90
2	0.03	0.00	7.07	0.00	0.47	7.18	61.10	4.97	2.00	242.00	17.80	392.83
3	0.03	0.00	2.18	0.00	0.46	7.00	45.80	6.06	3.00	222.00	18.70	394.63
4	0.07	0.00	2.18	0.00	0.46	7.15	54.20	6.06	3.00	222.00	18.70	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06	0.00	11.93	0.00	0.57	6.59	69.10	2.48	1.00	273.00	21.00	391.99
502	0.05	0.00	11.93	0.00	0.57	6.12	76.70	2.29	1.00	273.00	21.00	396.90
503	0.06	0.00	11.93	0.00	0.57	6.98	91.00	2.17	1.00	273.00	21.00	396.90
504	0.11	0.00	11.93	0.00	0.57	6.79	89.30	2.39	1.00	273.00	21.00	393.45
505	0.05	0.00	11.93	0.00	0.57	6.03	80.80	2.50	1.00	273.00	21.00	396.90

506 rows × 13 columns

In [6]:
```
df['MEDV'] = boston.target
```

In [7]:
```
df['MEDV']
```

Out[7]:
```
0      24.00
1      21.60
2      34.70
3      33.40
4      36.20
        ...
501    22.40
502    20.60
503    23.90
504    22.00
505    11.90
Name: MEDV, Length: 506, dtype: float64
```

In [8]:
```
df.head()
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.01	18.00	2.31	0.00	0.54	6.58	65.20	4.09	1.00	296.00	15.30	396.90	
1	0.03	0.00	7.07	0.00	0.47	6.42	78.90	4.97	2.00	242.00	17.80	396.90	
2	0.03	0.00	7.07	0.00	0.47	7.18	61.10	4.97	2.00	242.00	17.80	392.83	
3	0.03	0.00	2.18	0.00	0.46	7.00	45.80	6.06	3.00	222.00	18.70	394.63	
4	0.07	0.00	2.18	0.00	0.46	7.15	54.20	6.06	3.00	222.00	18.70	396.90	

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [10]:
```
df.dtypes.value_counts()
```

Out[10]:
```
float64    14
dtype: int64
```

In [11]:
```
# Descriptive Statistical Analysis
df.describe(include="all")
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRA
count	506.00	506.00	506.00	506.00	506.00	506.00	506.00	506.00	506.00	506.00	50
mean	3.61	11.36	11.14	0.07	0.55	6.28	68.57	3.80	9.55	408.24	1
std	8.60	23.32	6.86	0.25	0.12	0.70	28.15	2.11	8.71	168.54	
min	0.01	0.00	0.46	0.00	0.39	3.56	2.90	1.13	1.00	187.00	1
25%	0.08	0.00	5.19	0.00	0.45	5.89	45.02	2.10	4.00	279.00	1
50%	0.26	0.00	9.69	0.00	0.54	6.21	77.50	3.21	5.00	330.00	1
75%	3.68	12.50	18.10	0.00	0.62	6.62	94.07	5.19	24.00	666.00	2
max	88.98	100.00	27.74	1.00	0.87	8.78	100.00	12.13	24.00	711.00	2

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [13]:
```
df.shape
```

Out[13]:
```
(506, 14)
```

===

Feature Scaling

Data Standardization

Data is usually collected from different agencies in different formats. (Data standardization is also a term for a particular type of data normalization where we subtract the mean and divide by the standard deviation.)

What is standardization?

Standardisation involves centering the variable at zero, and standardising the variance to 1. The procedure involves subtracting the mean of each observation and then dividing by the standard deviation:

z = (x - x_mean) / std

The result of the above transformation is **z**, which is called the z-score, and represents how many standard deviations a given observation deviates from the mean. A z-score specifies the location of the observation within a distribution (in numbers of standard deviations respect to the mean of the distribution). The sign of the z-score (+ or -) indicates whether the observation is above (+) or below (-) the mean.

The shape of a standardised (or z-scored normalised) distribution will be identical to the original distribution of the variable. If the original distribution is normal, then the standardised distribution will be normal. But, if the original distribution is skewed, then the standardised distribution of the variable will also be skewed. In other words, **standardising a variable does not normalize the distribution of the data** and if this is the desired outcome, we should implement any of the techniques discussed in section 7 of the course.

In a nutshell, standardisation:

- centers the mean at 0
- scales the variance at 1
- preserves the shape of the original distribution
- the minimum and maximum values of the different variables may vary
- preserves outliers

Good for algorithms that require features centered at zero.

Feature magnitude matters because:

- The regression coefficients of linear models are directly influenced by the scale of the variable.
- Variables with bigger magnitude / larger value range dominate over those with smaller magnitude / value range
- Gradient descent converges faster when features are on similar scales
- Feature scaling helps decrease the time to find support vectors for SVMs
- Euclidean distances are sensitive to feature magnitude.
- Some algorithms, like PCA require the features to be centered at 0.

The machine learning models affected by the feature scale are:

- Linear and Logistic Regression
- Neural Networks
- Support Vector Machines
- KNN
- K-means clustering
- Linear Discriminant Analysis (LDA)
- Principal Component Analysis (PCA)

Feature scaling refers to the methods or techniques used to normalize the range of independent variables in our data, or in other words, the methods to set the feature value range within a similar scale. Feature scaling is generally the last step in the data preprocessing pipeline, performed **just before training the machine learning algorithms**.

There are several Feature Scaling techniques, which we will discuss throughout this section:

- Standardisation
- Mean normalisation
- Scaling to minimum and maximum values - MinMaxScaling
- Scaling to maximum value - MaxAbsScaling

- Scaling to quantiles and median - RobustScaling
- Normalization to vector unit length

Name	Sklearn_class
Standard scaler	Standard scaler
MinMaxScaler	MinMax Scaler
MaxAbs Scaler	MaxAbs Scaler
Robust scaler	Robust scaler
Quantile Transformer_Normal	Quantile Transformer(output_distribution ='normal')
Quantile Transformer_Uniform	Quantile Transformer(output_distribution = 'uniform')
PowerTransformer-Yeo-Johnson	PowerTransformer(method = 'yeo-johnson')
Normalizer	Normalizer

In [14]:
```
df.head()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.01	18.00	2.31	0.00	0.54	6.58	65.20	4.09	1.00	296.00	15.30	396.90	
1	0.03	0.00	7.07	0.00	0.47	6.42	78.90	4.97	2.00	242.00	17.80	396.90	
2	0.03	0.00	7.07	0.00	0.47	7.18	61.10	4.97	2.00	242.00	17.80	392.83	
3	0.03	0.00	2.18	0.00	0.46	7.00	45.80	6.06	3.00	222.00	18.70	394.63	
4	0.07	0.00	2.18	0.00	0.46	7.15	54.20	6.06	3.00	222.00	18.70	396.90	

In [15]:
```
df.shape
```

Out[15]: (506, 14)

In [16]:
```
X = df.iloc[:, 0:13]
y = df.iloc[:, 13]
```

In [17]:
```
X.values, y.values
```

```
Out[17]: (array([[  0.00632,  18.     ,   2.31   , ...,  15.3    , 396.9    ,
            4.98   ],
         [  0.02731,   0.     ,   7.07   , ...,  17.8    , 396.9    ,
            9.14   ],
         [  0.02729,   0.     ,   7.07   , ...,  17.8    , 392.83   ,
            4.03   ],
         ...,
         [  0.06076,   0.     ,  11.93   , ...,  21.     , 396.9    ,
            5.64   ],
         [  0.10959,   0.     ,  11.93   , ...,  21.     , 393.45   ,
            6.48   ],
         [  0.04741,   0.     ,  11.93   , ...,  21.     , 396.9    ,
            7.88   ]]),
  array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
         18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
         15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
         13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
         21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
         35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
         19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
         20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
         23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
         33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
         21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
         20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
         23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
         15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
         17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
         25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
         23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
         32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
         34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
         20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
         26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
         31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
         22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
         42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
         36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
         32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
         20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
         20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
         22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
         21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
         19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
         32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
         18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
         16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
         13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
          7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
         12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
         27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
          8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
          9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
         10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
         15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
         19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
```

```
29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]))
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [19]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[19]:
```
((404, 13), (102, 13), (404,), (102,))
```

In [20]:
```
# standardisation: with the StandardScaler from sklearn

# set up the scaler
scaler = StandardScaler()
```

In [21]:
```
# fit the scaler to the train set, it will learn the parameters
scaler.fit(X)
```

Out[21]:
```
StandardScaler()
```

In [22]:
```
X_scaled = scaler.fit_transform(X)
```

In [23]:
```
X_scaled
```

Out[23]:
```
array([[-0.41978194,  0.28482986, -1.2879095 , ..., -1.45900038,
         0.44105193, -1.0755623 ],
       [-0.41733926, -0.48772236, -0.59338101, ..., -0.30309415,
         0.44105193, -0.49243937],
       [-0.41734159, -0.48772236, -0.59338101, ..., -0.30309415,
         0.39642699, -1.2087274 ],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.98304761],
       [-0.40776407, -0.48772236,  0.11573841, ...,  1.17646583,
         0.4032249 , -0.86530163],
       [-0.41500016, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.66905833]])
```

In [24]:
```
X_scaled.shape
```

Out[24]:
```
(506, 13)
```

In [25]:
```
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
```

In [26]:
```
X_scaled
```

Out[26]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	-0.42	0.28	-1.29	-0.27	-0.14	0.41	-0.12	0.14	-0.98	-0.67	-1.46	0.44
1	-0.42	-0.49	-0.59	-0.27	-0.74	0.19	0.37	0.56	-0.87	-0.99	-0.30	0.44
2	-0.42	-0.49	-0.59	-0.27	-0.74	1.28	-0.27	0.56	-0.87	-0.99	-0.30	0.40
3	-0.42	-0.49	-1.31	-0.27	-0.84	1.02	-0.81	1.08	-0.75	-1.11	0.11	0.42
4	-0.41	-0.49	-1.31	-0.27	-0.84	1.23	-0.51	1.08	-0.75	-1.11	0.11	0.44
...	...	...	...	...	...	...	...	...	...	...	...	...
501	-0.41	-0.49	0.12	-0.27	0.16	0.44	0.02	-0.63	-0.98	-0.80	1.18	0.39
502	-0.42	-0.49	0.12	-0.27	0.16	-0.23	0.29	-0.72	-0.98	-0.80	1.18	0.44
503	-0.41	-0.49	0.12	-0.27	0.16	0.98	0.80	-0.77	-0.98	-0.80	1.18	0.44
504	-0.41	-0.49	0.12	-0.27	0.16	0.73	0.74	-0.67	-0.98	-0.80	1.18	0.40
505	-0.42	-0.49	0.12	-0.27	0.16	-0.36	0.43	-0.61	-0.98	-0.80	1.18	0.44

506 rows × 13 columns

In [27]:

```
y
```

Out[27]:

```
0      24.00
1      21.60
2      34.70
3      33.40
4      36.20
        ...
501    22.40
502    20.60
503    23.90
504    22.00
505    11.90
Name: MEDV, Length: 506, dtype: float64
```

In [28]:

```
y.shape
```

Out[28]:

```
(506,)
```

```
=================================================================
```