Name:- Rohini Janardan Devkar
Roll no:- 23272
PRN no:- 72030818G
TE-2 (comp)
DSBDA Lab

Practical No.4.
Data Analytics I

* **Aim:-** Create a linear regression model using python / R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

The objective is to predict the value of prices of the house using the given features.

* **Theory:-**

In this project, we will develop the performance and the predictive power of a model trained and tested on data collected from houses in Boston's Suburbs.

The Boston housing market is highly competitive, and you want to be the best real estate agent in the area.

① Getting the data and Previous Processes:-

The dataset used in this project comes from the UCI machine learning repository. This data was collected in 1978 and each of the 506 entries represents aggregate information about 14 features of homes from various suburbs located in Boston.

The features can be summarized as follows:

- CRIM: This is the per capita crime rate by town.

- ZN :- This is the proportion of residential land zoned for lots larger than 25,000 sq.ft.

- INDUS :- This is the proportion of non-retail business acres per town.

- CHAS :- This is the Charles River dummy variable.

- NOX :- This is the nitric oxides concentration

- RM :- This is the avg number of rooms per dwelling.

- AGE :- This is the proportion of owner-occupied units built prior to 1940.

- DIS :- This is the weighted distances to five Boston employment centres.

- RAD :- This is the index of accessibility to radial highways.

- TAX :- This is the full-value property-tax rate per $10,000.

- PTRATIO :- This is the pupil-teacher ratio bytown.

- B : This is calculated as $1000(Bk - 0.63)^2$, where Bk is the proportion of people of African American descent by town.

- LSTAT :- This is the percentage lower status of the population.

- MEDV :- This is the median value of owner-occupied homes in $1000s.

For the purpose of the project the dataset has been preprocessed as :-

- The essential features for the project are :- 'RM', 'LSTAT', 'PTRATIO' and 'MEDV'. The remaining features have been excluded.
- 16 data points with a 'MEDV' value of 50.0 have been removed. As they likely contain censored or missing values.

② ~~Ex~~ Data Exploration:-

In the first section of the project, we will make an exploratory analysis of the dataset and provide some observations.

③ Feature Observation:-

1) Houses with more rooms (higher 'RM' value) will worth more. Usually houses with more rooms are higher and can fit more people, so it is reasonable that they cost more money.

2) Neighbourhoods with more lower class workers (higher 'LSTAT' value) will worth less. If the percentage of lower working class people is higher, it is likely that they have low purchasing power and therefore, they houses will cost less. They are inversely proportional variables.

3) Neighbourhoods with more students to teachers ratio (higher 'PTRATIO' value) will be worth less. If the percentage of students to ratio people is higher, it is likely that in the neighbourhood there are less schools, this could be because there is less tax income which could be because in that neighbourhood people earn less money.

④ Exploratory Data Analysis:- (scatterplot and histograms):

We will start by creating a scatterplot matrix that will allow us to visualize the pair-wise relationships and correlations between the different features.

It is also quite useful to have a quick overview of how data is distributed and whether it contains or not outliers.

Page No.

Date

We can spot a linear relationsplitship between 'RM' and House prices 'MEDV'. In addition, we can infer from the histogram that the 'MEDV' variable seems to be normally distributed but contain several outliers.

5 Correlation Matrix:-

We are going to create now a correlation matrix to quantify and summarize the relationships between the variables. It is a square matrix that contains the Person's r correlation coefficient.

~~7 Developing Models:-~~

7 Training and Testing:-

It is useful to evaluate our model once it is trained. If it has learned properly from a training split of a data. There can be 3 different situations:-

1) The model didn't learn well on the data, and can't predict even the outcomes of the training set. this is called underfitting and it is caused because a high bias.

2) The model learn too well the training data, up to the point that is memorized it and is not able to generalize on new data. this is called overfitting, it is caused because high variance.

3) The model just had the right balance between bias and variance, it learned well and is able to predict correctly the outcomes on new data.

8 Cross-Validation:-

k-Fold cross-validation is a technique used for making sure that our model is well trained, without using the test set. It consist in splitting data into k partitions of equal size. For each partition i,

Page No.
Date

we train the model on the remaining k-1 parameters and evaluate it on partition i. The final score is the average of k scores obtained.

However, by partitioning the available data into three sets (training, validating and testing sets), we drastically reduce the number of samples which can be used for learning the model, and the resulting model may not be sufficiently well trained.

* Linear regression :-

Linear regression is a statistical method for modeling relationships between a dependent variable with a given set of independent variables.

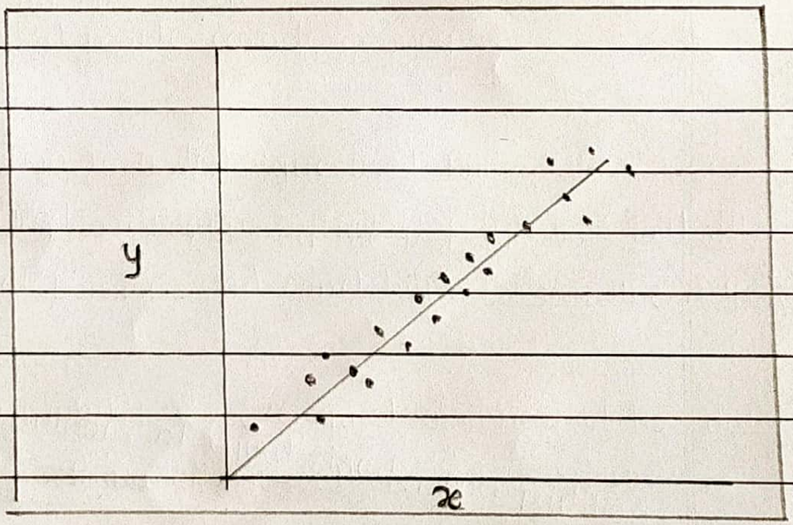
There are 2 types of linear regression :-

1) Simple linear Regression :-

Simple linear regression is an approach for predicting a response using a single feature. It is assumed that the two variables are linearly related.

2) Multiple linear regression :-

Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to the observed data. Clearly, it is nothing but an extension of simple linear regression.



Linear Regression

* Applications of linear regression:

- Trend lines :- A trend line represents the variation in quantitative data with the passage of time (like GDP, oil prices, etc.). These trends usually follow a linear relationship. Hence, linear regression can be applied to predict future values.

- Economics :- Linear regression is the predominant empirical tool in economics. For example, it is used to predict consumer spending, fixed investment spending, inventory investment, purchases of a country's exports, spending on imports.

- Finance :- The capital price asset model uses linear regression to analyze and quantify the systematic risks of an investment.

- Biology :- Linear regression is used to model casual relationship between parameters in biological systems.

Data Science And Big Data Analytics Practical 4

===

Name:- Rohini Janardan Devkar

Roll no:- 23272

PRN NO:- 72030818G

Class :- TE2(COMP)

===

Problem Statement:-

Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset. The objective is to predict the value of prices of the house using the given features.

===

In [1]:

```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()
```

In [2]:

```
boston.data.shape
```

Out[2]:

```
(506, 13)
```

In [3]:

```
boston.feature_names
```

Out[3]:

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [4]:

```
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.head(10)
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

In [5]:

```
# Adding 'Price' (target) column to the data
boston.target.shape
```

Out[5]:

```
(506,)
```

In [6]:

```
data['Price'] = boston.target
data.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [7]:

```
data.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [8]:

```
data.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  Price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [9]:

```
# Input Data
x = boston.data

# Output Data
y = boston.target


# splitting data to training and testing dataset.

#from sklearn.cross_validation import train_test_split
#the submodule cross_validation is renamed and reprecated to model_selection
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,


print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

```
xtrain shape :  (404, 13)
xtest shape :  (102, 13)
ytrain shape :  (404,)
ytest shape :  (102,)
```

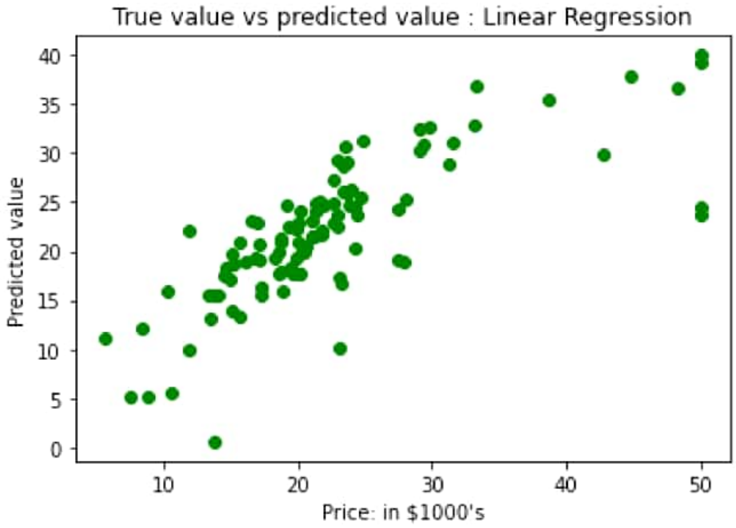
In [10]:

```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

# predicting the test set results
y_pred = regressor.predict(xtest)
```

In [11]:

```
# Plotting Scatter graph to show the prediction
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```



In [12]:

```
# Results of Linear Regression.
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(ytest, y_pred)
print("Mean Square Error : ", mse)
```

```
Mean Square Error :  33.4489799976765
```

Page No.
Date

* Conclusion:-

Throughout this, we made a machine learning regression project from end-to-end and we learned and obtained several insights about regression models and how they are developed.