Data Analytics I

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 [9]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import fetch_california_housing
boston = fetch_california_housing()
```

In [10]:

```
boston.data.shape
```

Out[10]:

```
(20640, 8)
```

In [11]:

```
boston.feature_names
```

Out[11]:

```
['MedInc',
 'HouseAge',
 'AveRooms',
 'AveBedrms',
 'Population',
 'AveOccup',
 'Latitude',
 'Longitude']
```

In [12]:

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

In [13]:

```
data.head(15)
```

Loading [MathJax]/extensions/Safe.js

Out[13]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitu
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.
5	4.0368	52.0	4.761658	1.103627	413.0	2.139896	37.
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405	37.
7	3.1200	52.0	4.797527	1.061824	1157.0	1.788253	37.
8	2.0804	42.0	4.294118	1.117647	1206.0	2.026891	37.
9	3.6912	52.0	4.970588	0.990196	1551.0	2.172269	37.
10	3.2031	52.0	5.477612	1.079602	910.0	2.263682	37.
11	3.2705	52.0	4.772480	1.024523	1504.0	2.049046	37.
12	3.0750	52.0	5.322650	1.012821	1098.0	2.346154	37.
13	2.6736	52.0	4.000000	1.097701	345.0	1.982759	37.
14	1.9167	52.0	4.262903	1.009677	1212.0	1.954839	37.

In [14]:

```
boston.target.shape
```

Out[14]:

```
(20640,)
```

In [15]:

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

Out[15]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85

In [16]:

```
data.describe()
```

Out[16]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744
std	1.899822	12.585558	2.474173	0.473911	1132.462122
min	0.499900	1.000000	0.846154	0.333333	3.000000
25%	2.563400	18.000000	4.440716	1.006079	787.000000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000
max	15.000100	52.000000	141.909091	34.066667	35682.000000

In [17]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   Price       20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [18]:

```
x=boston.data
y=boston.target

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 : (16512, 8)
xtest shape : (4128, 8)
ytrain shape : (16512,)
ytest shape : (4128,)
```

In [19]:

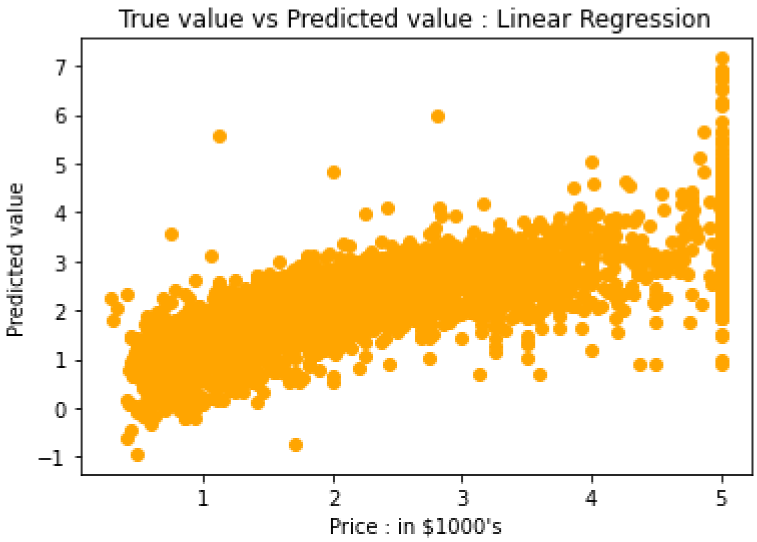
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain,ytrain)
```

Loading [MathJax]/extensions/Safe.js

```
y_pred = regressor.predict(xtest)
```

In [20]:

```
plt.scatter(ytest,y_pred, c = 'orange')
plt.xlabel("Price : in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs Predicted value : Linear Regression")
plt.show()
```



In [21]:

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

```
Mean Square Error : 0.5074335030836559
```

In [ ]: