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

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

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

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu ×

```
        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

```
boston.data.shape
```

```
(506, 13)
```

```
boston.feature_names
```

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

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

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

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12
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
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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	5.0	311.0	15.2	386.63
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
10	0.22489	12.5	7.87	0.0	0.524	6.377	94.3	6.3467	5.0	311.0	15.2	392.52
11	0.11747	12.5	7.87	0.0	0.524	6.009	82.9	6.2267	5.0	311.0	15.2	396.90
12	0.09378	12.5	7.87	0.0	0.524	5.889	39.0	5.4509	5.0	311.0	15.2	390.50
13	0.62976	0.0	8.14	0.0	0.538	5.949	61.8	4.7075	4.0	307.0	21.0	396.90
14	0.63796	0.0	8.14	0.0	0.538	6.096	84.5	4.4619	4.0	307.0	21.0	380.02

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu ✕

```
boston.target.shape
```

```
(506,)
```

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

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

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	A
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.5749
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.1488
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.0250
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.5000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.0750
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.0000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu ✕

```
 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
```

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

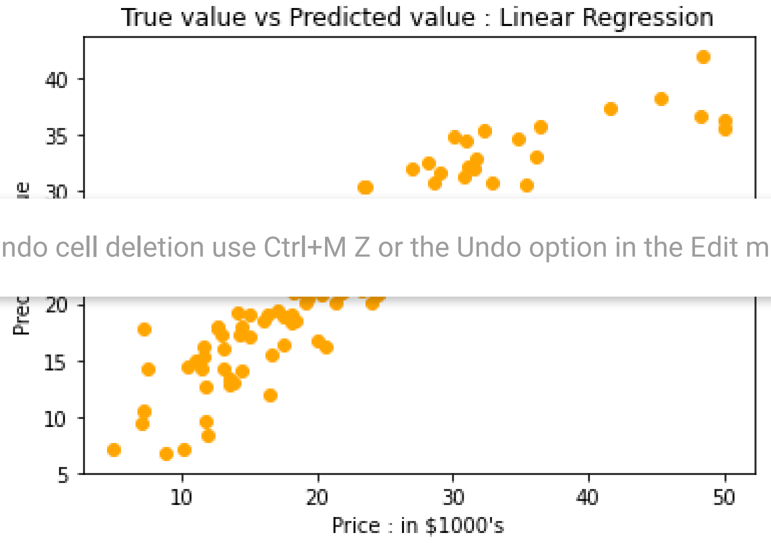
```
from sklearn.model_selection import train_test_split
```

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

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

y_pred = regressor.predict(xtest)
```

```
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()
```



To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

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

```
     Mean Square Error : 28.89318492981834
```

✓ 0s completed at 2:27 PM ● ×

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu ×