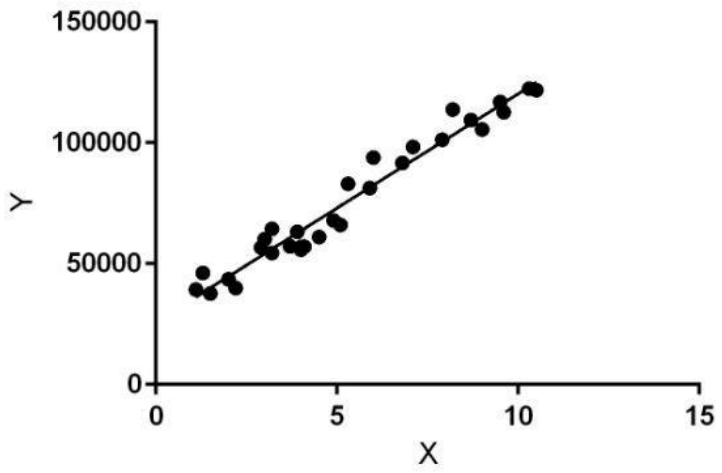
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

Aim: Analyze the Boston Housing dataset and apply appropriate Regression Technique.

Objective: Ablility to perform various feature engineering tasks, apply linear regression on the given dataset and minimise the error.

Theory:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Dataset:

The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

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 (1 if tract bounds river; 0 otherwise)

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 blacks by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

Code:

Conclusion:

1. What are features have been chosen to develop the model? Justify the features chosen to estimate the price of a house.
    - To fit a linear regression model, we select those features which have a high correlation with our target variable MEDV. By looking at the correlation matrix we can see that RM has a strong positive correlation with MEDV (0.7) whereas LSTAT has a high negative correlation with MEDV(-0.74).
    - An important point in selecting features for a linear regression model is to check for multi-co-linearity. The features RAD, TAX have a correlation of 0.91. These feature pairs are strongly correlated to each other.
    - Therefore we consider these four features for linear regression 'LSTAT', 'RM', 'RAD' , 'TAX'

2. Comment on the Mean Squared Error calculated.

    - Mean Squared Error (MSE) quantifies how far off a model's predictions are from actual data points. It calculates the average of the squared differences between predicted and actual values. A higher MSE (like 23.695) suggests larger prediction errors. Lower MSE indicates better model performance.
    - To improve the MSE of 23.695, we can focus on refining feature selection, optimizing model parameters, experimenting with algorithms, and enhancing data preprocessing. We can consider acquiring more data and using regularization techniques to mitigate overfitting.
    -

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.gofplots import ProbPlot
import sklearn.datasets
from sklearn.model_selection import train_test_split
from statsmodels.formula.api import ols
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
```

```
data=pd.read_csv('HousingData.csv')
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	N

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24

```
data.shape
```

```
(506, 14)
```

```
data.isnull().sum()
```

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
data.duplicated().sum()
```

```
0
```

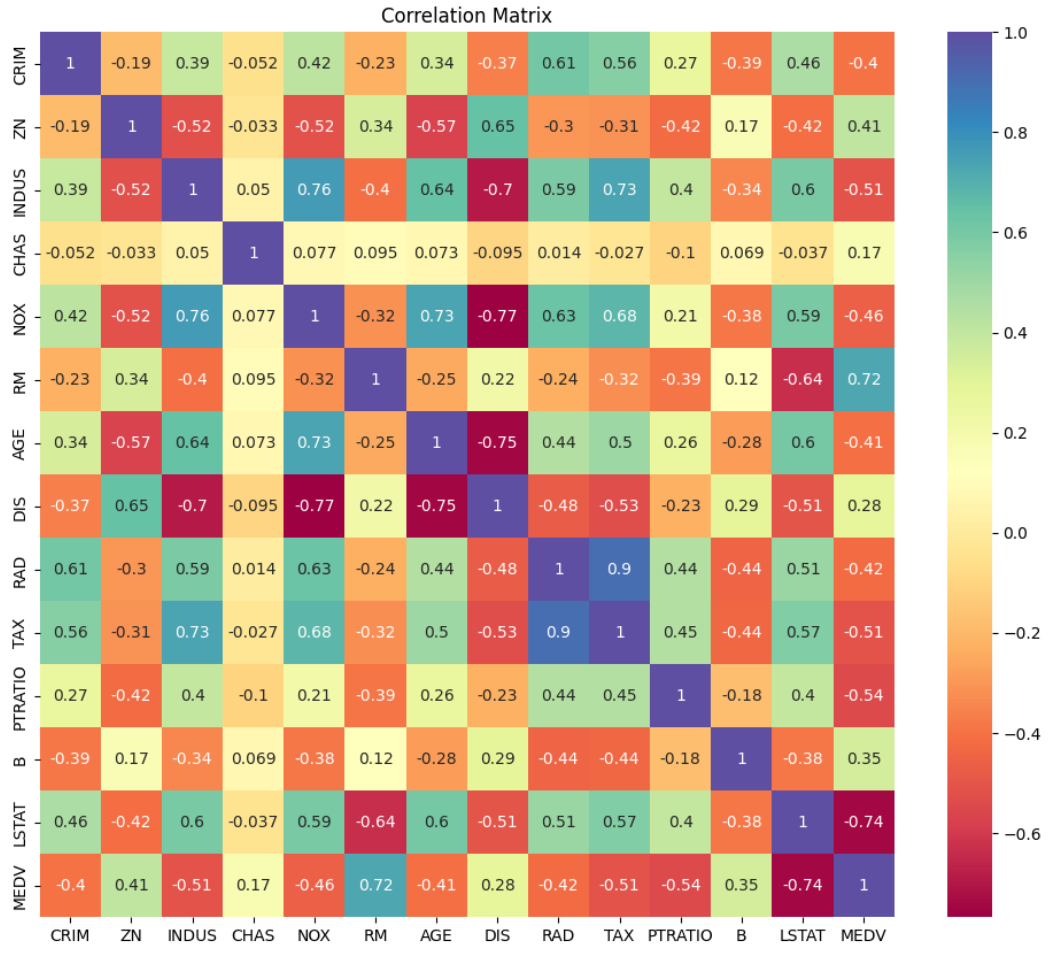
```
data = data.dropna()
data.isnull().sum()
```

```
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
```

```
# Correlation matrix
corr = data.corr()
plt.figure(figsize=(12,10))
sns.heatmap(data=corr, annot=True, cmap='Spectral').set(title="Correlation Matrix")
```

```
[Text(0.5, 1.0, 'Correlation Matrix')]
```



Observations:

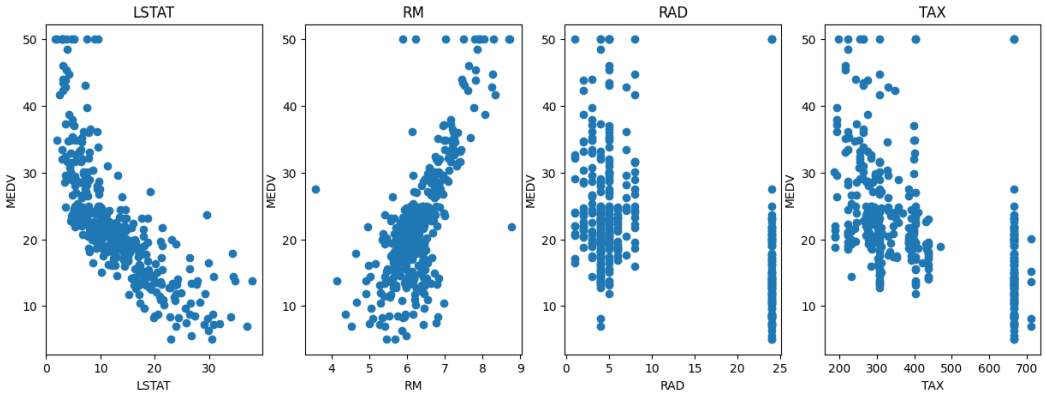
- To fit a linear regression model, we select those features which have a high correlation with our target variable MEDV. By looking at the correlation matrix we can see that RM has a strong positive correlation with MEDV (0.7) where as LSTAT has a high negative correlation with MEDV(-0.74).
- An important point in selecting features for a linear regression model is to check for multi-co-linearity. The features RAD, TAX have a correlation of 0.91. These feature pairs are strongly correlated to each other. We should not select both these features together for training the model. Check this for an explanation. Same goes for the features DIS and AGE which have a correlation of -0.75.

```
plt.figure(figsize=(15, 5))

features = ['LSTAT', 'RM', 'RAD' , 'TAX']
target = data['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = data[col]
```

```
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



Observations:

- The prices increase as the value of RM increases linearly. There are few outliers and the data seems to be capped at 50.
- The prices tend to decrease with an increase in LSTAT. Though it doesn't look to be following exactly a linear line.

```
X = pd.DataFrame(np.c_[data['LSTAT'], data['RM'] ,data['TAX'], data['RAD']], columns = ['LSTAT','RM', 'TAX', 'RAD'])
Y = data['MEDV']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(315, 4)
(79, 4)
(315,)
(79,)
```

```
# Train the model
model = LinearRegression()
model.fit(X_train, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Predict
y_pred = model.predict(X_test)
```

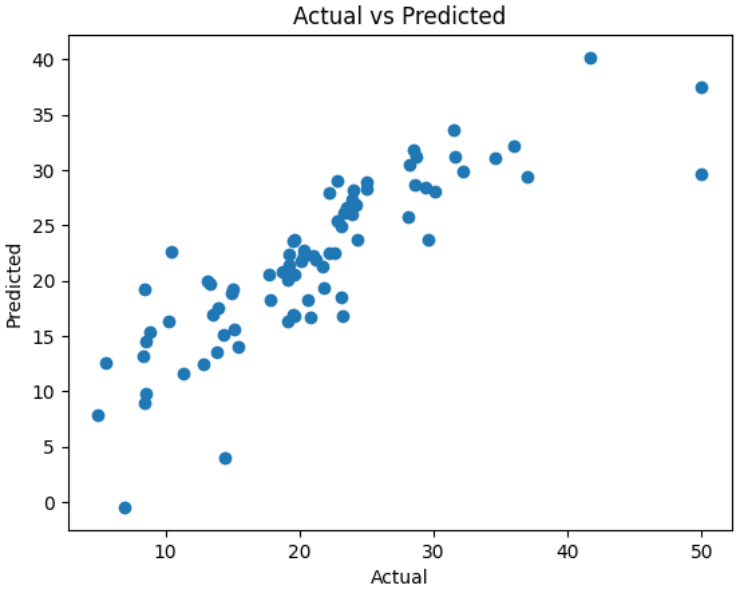
```
# Evaluate
from sklearn.metrics import mean_squared_error
print('Mean Squared Error: ', mean_squared_error(Y_test,y_pred))
```

```
Mean Squared Error:  23.695394676765567
```

```
# Root Mean Squared Error
print('Root Mean Squared Error: ', np.sqrt(mean_squared_error(Y_test,y_pred)))
```

```
Root Mean Squared Error:  4.867791560529843
```

```
# Plot
plt.scatter(Y_test,y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()
```



✓ 0s completed at 8:53 PM