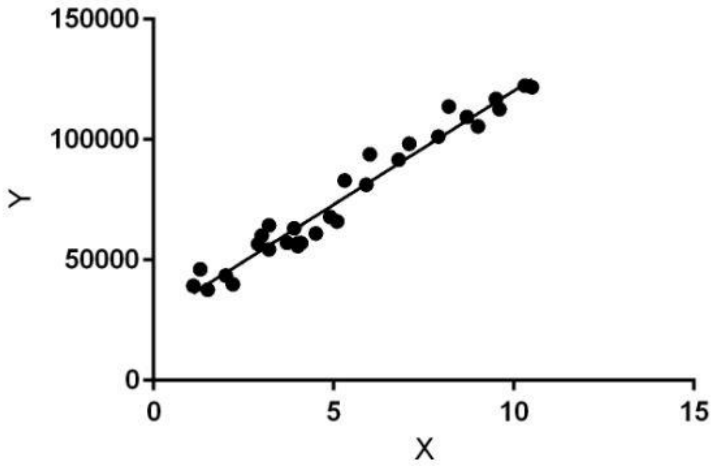
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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:

Features have been chosen to develop the model:

1. CRIM - Per capita crime rate by town
2. CHAS - Charles River dummy variable (1 if tract bounds river; else 0)
3. NOX - Nitric oxides concentration (parts per 10 million)
4. RM - Average number of rooms per dwelling
5. DIS - weighted distances to five Boston employment centres
6. RAD - Index of accessibility to radial highways
7. TAX - Full-value property-tax rate per $10,000
8. PTRATIO - Pupil-teacher ratio by town
9. LSTAT - Lower status of the population

Mean Squared Error calculated:

- Calculated Mean Squared Error: 0.04 (+/- 0.04)
- The Mean Squared Error measures how close a regression line is to a set of data points.
- Lesser the Mean Squared Error refers to Smaller is the error and Better the estimator

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

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
dataset = pd.read_csv("boston_train.csv")
dataset
dataset.head()
```

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
3	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
4	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9

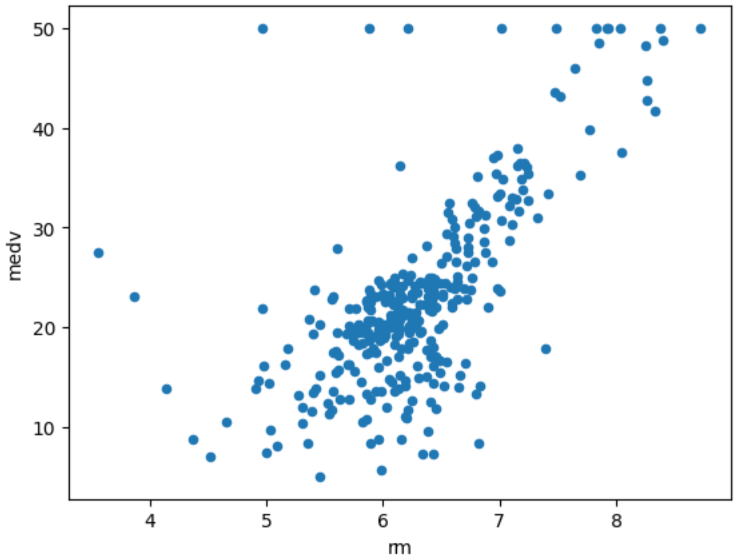
```
dataset.info()
# dataset.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   ID       333 non-null    int64
 1   crim     333 non-null    float64
 2   zn       333 non-null    float64
 3   indus    333 non-null    float64
 4   chas     333 non-null    int64
 5   nox      333 non-null    float64
 6   rm       333 non-null    float64
 7   age      333 non-null    float64
 8   dis      333 non-null    float64
 9   rad      333 non-null    int64
 10  tax      333 non-null    int64
 11  ptratio  333 non-null    float64
 12  black    333 non-null    float64
 13  lstat    333 non-null    float64
 14  medv     333 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 39.1 KB
```

```
dataset = dataset.drop('ID',axis=1)
```

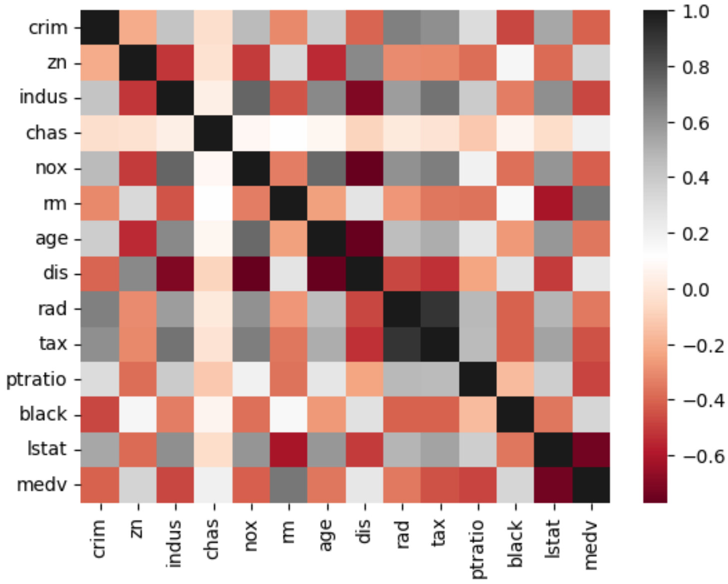
```
dataset.plot.scatter('rm', 'medv')
# dataset.plot.scatter('dis', 'medv')
```

```
<Axes: xlabel='rm', ylabel='medv'>
```




```
sns.heatmap(dataset.corr(), cmap = 'RdGy')
```

```
<Axes: >
```




```
#split x and y
x=dataset[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = dataset['medv']
```

```
# split train and test set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

```
X_train.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
74	0.26363	0.0	8.56	0	0.520	6.229	91.2	2.5451	5	384	20.9	391.23	15.55
127	0.06888	0.0	2.46	0	0.488	6.144	62.2	2.5979	3	193	17.8	396.90	9.45
46	0.13554	12.5	6.07	0	0.409	5.594	36.8	6.4980	4	345	18.9	396.90	13.09
55	0.04462	25.0	4.86	0	0.426	6.619	70.4	5.4007	4	281	19.0	395.63	7.22
318	2.81838	0.0	18.10	0	0.532	5.762	40.3	4.0983	24	666	20.2	392.92	10.42

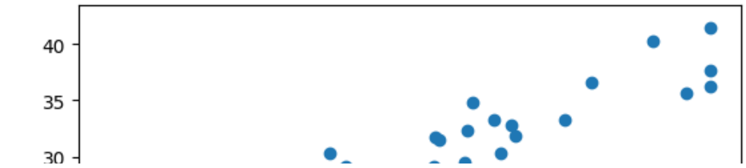
```
lr = LinearRegression()
lr.fit(X_train,y_train)
# print(lr)
```

```
▾ LinearRegression
LinearRegression()
```

```
predictions = lr.predict(X_test)
```

```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.3439431972740317
MSE: 20.634772839844114
RMSE: 4.5425513579753956
```

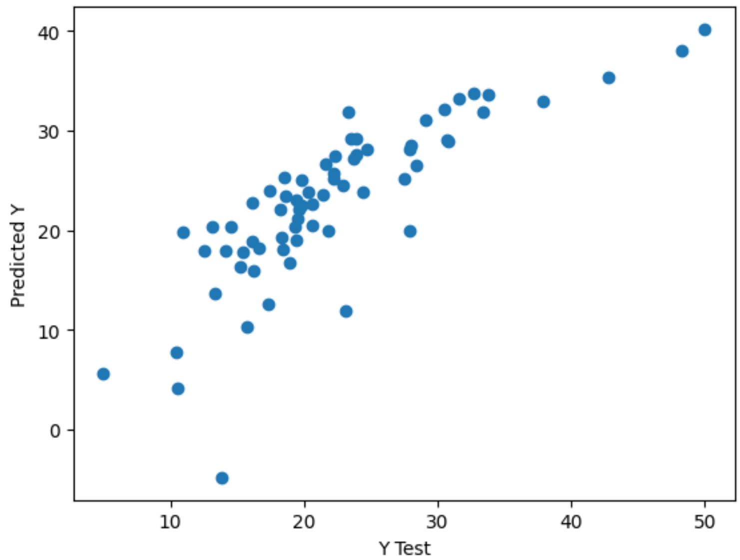
```
x=dataset[['crim', 'indus', 'rm', 'age',  'tax','ptratio', 'lstat']]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

lr = LinearRegression()
lr.fit(X_train,y_train)

predictions = lr.predict(X_test)
```

```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.7949957884176535
MSE: 25.076245805594446
RMSE: 5.007618775984694
```