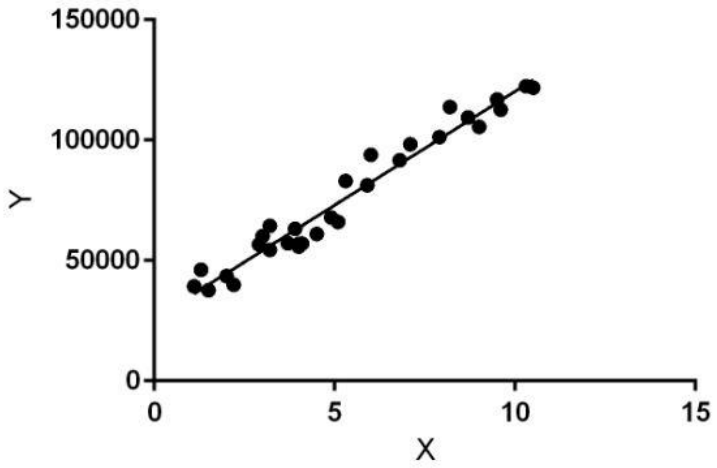
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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.

   • A linear regression model was developed to estimate house prices using the 'LSTAT' (percentage of the lower status of the population) feature. This choice is justified by the intuition that areas with a higher percentage of lower-status individuals might have lowerpriced housing due to economic disparities.

   • By focusing on the 'LSTAT' feature alone, the model simplifies the prediction process, which is important for illustration purposes. However, it's important to note that real-world housing price prediction models often consider multiple features like crime rate ('CRIM'), average number of rooms ('RM'), and accessibility to employment centers ('DIS') among others, as these collectively offer a more comprehensive understanding of housing dynamics.

   • While this simplified model captures a basic trend, a more robust model would involve incorporating a broader set of features to enhance predictive accuracy

2. Comment on the Mean Squared Error calculated.

- The Mean Squared Error (MSE) evaluates linear regression's performance by measuring the squared difference between actual and predicted house prices.

- Calculated MSE values for training and testing datasets reveal model accuracy and generalizability. Comparing MSE helps detect overfitting or underfitting. Reported MSE values serve as a reference for assessing predictive performance, specifically the 'LSTAT' feature's effectiveness.

- Consider data scale and characteristics when interpreting lower MSE values. Compare with complex models or algorithms for a deeper understanding of predictive accuracy

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

```
data = pd.read_csv("housing.csv")
```

```
data
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	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	242	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	242	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	222	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	222	18.7	396.90	NaN	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

```
X=data.iloc[:,:-1].values
Y=data.iloc[:,-1].values
```

```
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	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	242	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	242	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	222	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	222	18.7	396.90	NaN	36.2

```
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     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
```

```
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

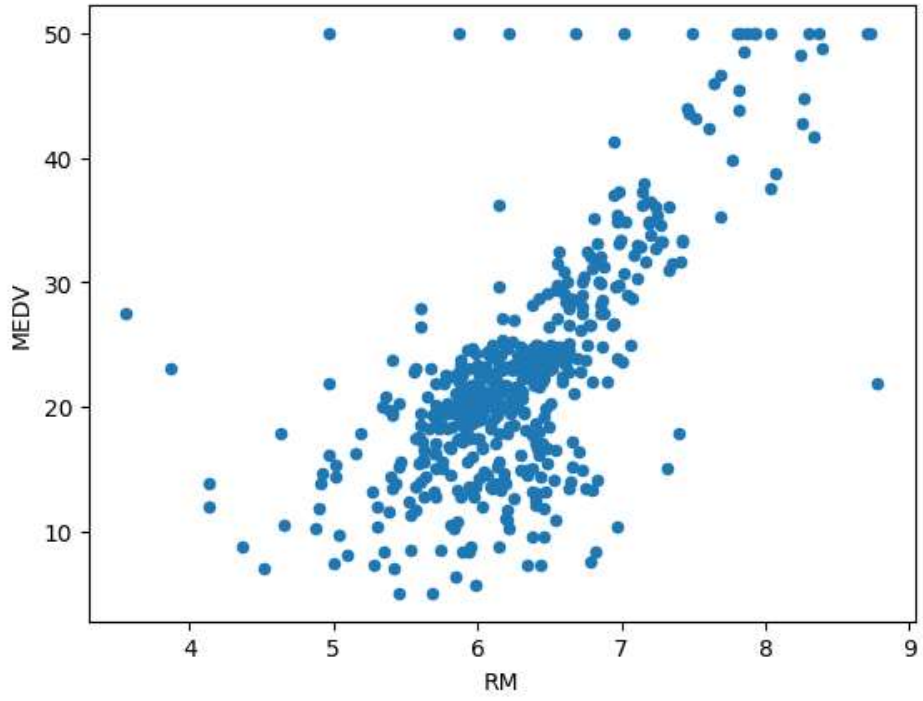
```
data['CRIM'].fillna(data['CRIM'].mean() , inplace = True)
data['ZN'].fillna(data['ZN'].mean() , inplace = True)
data['INDUS'].fillna(data['INDUS'].mean() , inplace = True)
data['CHAS'].fillna(data['CHAS'].mean() , inplace = True)
data['AGE'].fillna(data['AGE'].mean() , inplace = True)
data['LSTAT'].fillna(data['LSTAT'].mean() , inplace = True)
```

```
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
```

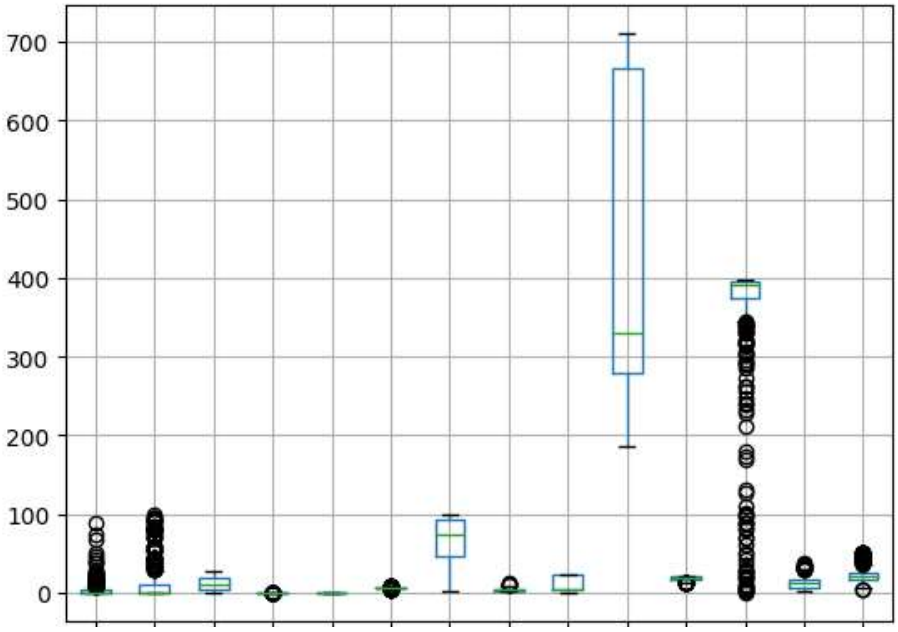
```
data.plot.scatter('RM', 'MEDV')
```

```
<Axes: xlabel='RM', ylabel='MEDV'>
```



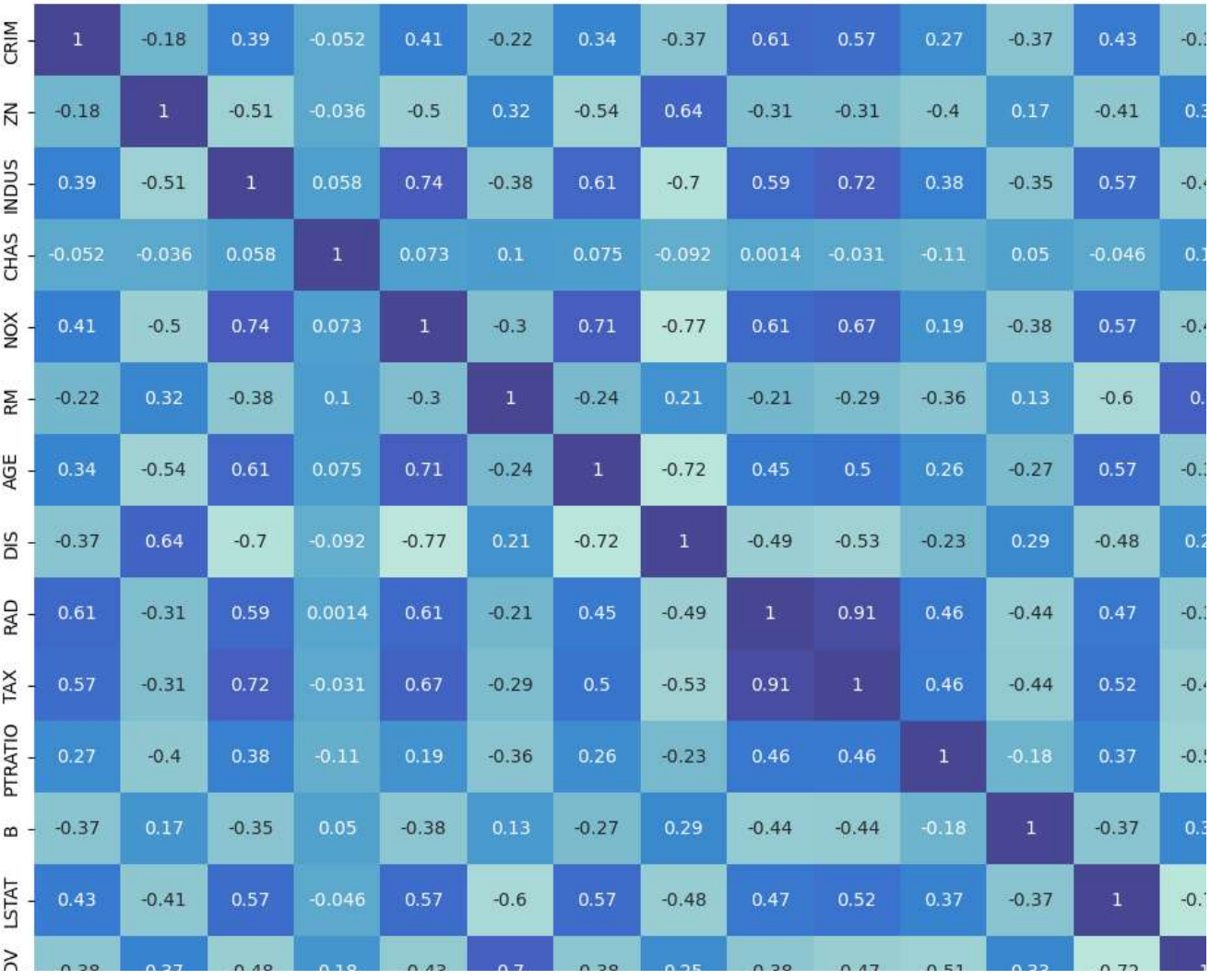

```
data.boxplot(column_names, rot=15)
```

<Axes: >




```
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

```
plt.figure(figsize=(15,10))
sns.heatmap(data.select_dtypes(include=['int','float']).corr(),annot=True,center = 2)
plt.show()
```



```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)



print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(404, 13)
(102, 13)
(404,)
(102,)
```

```
X = data[['LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' , 'RAD' ,
'AGE' , 'CRIM' , 'ZN']]
Y = data['MEDV']
seed= 1
X_train , X_test, Y_train , Y_test = train_test_split(X, Y,
test_size=0.20, random_state=seed)


X.shape
```

```
(506, 10)
```

```
Y.shape
```

```
(506,)
```

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_train , Y_train)
LinearRegression()
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred= LR.predict(X_test)


from sklearn import metrics
import numpy as np

print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squred Error:",metrics.mean_squared_error(y_test,y_pred))
print("Root Mean Squred Error:",metrics.mean_squared_error(y_test,y_pred))
```

```
Mean Absolute Error: 4.311333848096257
Mean Squred Error: 29.58597268132346
Root Mean Squred Error: 29.58597268132346
```

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,y_pred))
```

```
Mean Absolute Error: 4.311333848096257
```

```
plt.scatter(y_test, y_pred, c = 'Blue')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
```

```
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```