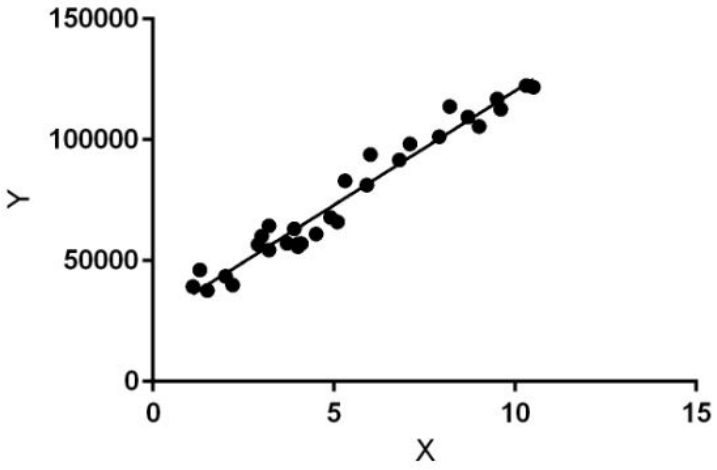
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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:

BOSTON HOUSE PREDICTION

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

Importing dataset

```
df=pd.read_csv("/content/BostonHousing.csv")
```

Data Preprocessing

```
df.head
```

```
<bound method NDFrame.head of         crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]>
```

```
df.shape
```

```
(506, 14)
```

```
df.dtypes
```

```
crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
```

```
lstat      float64
medv       float64
dtype: object
```

```
print(df.isnull().sum())
```

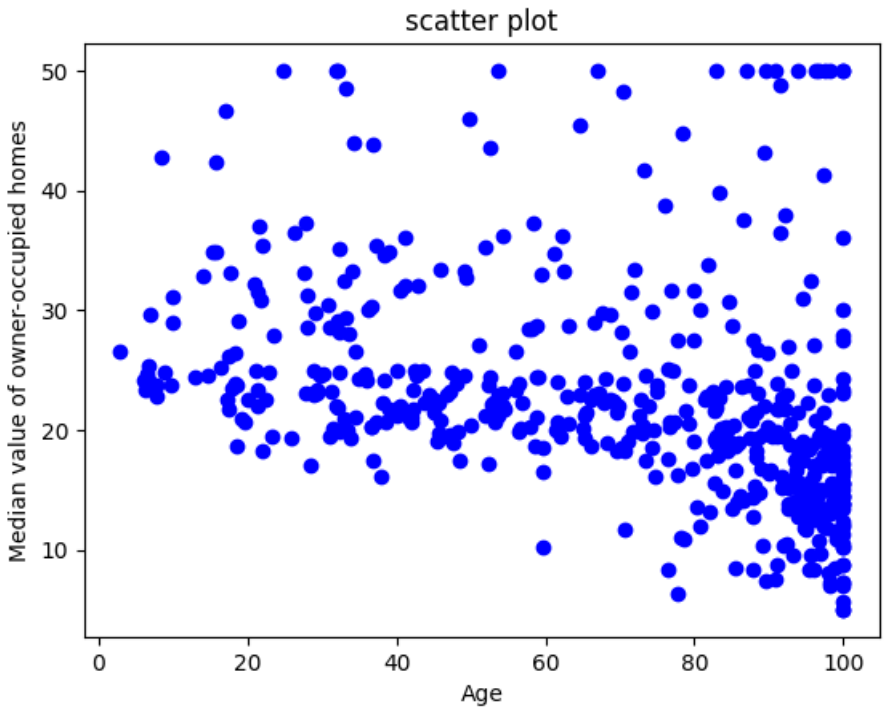
```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

```
df.describe()
```

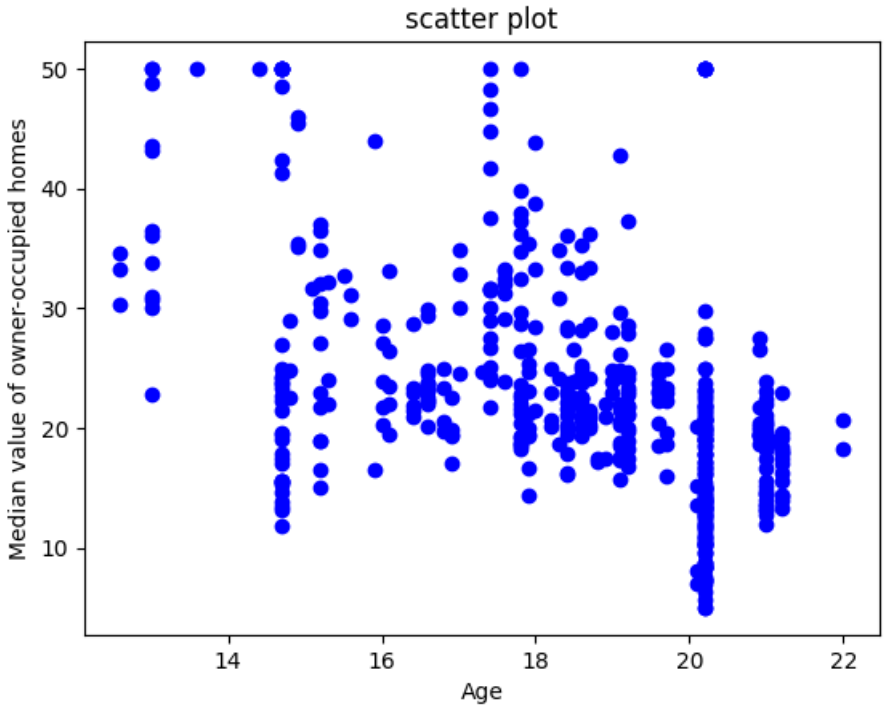
	crim	zn	indus	chas	nox	rm	age	dis	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24

Data Visualization

```
plt.scatter(df['age'],df['medv'], color='blue')
plt.title("scatter plot")
plt.xlabel("Age")
plt.ylabel("Median value of owner-occupied homes")
plt.show()
```

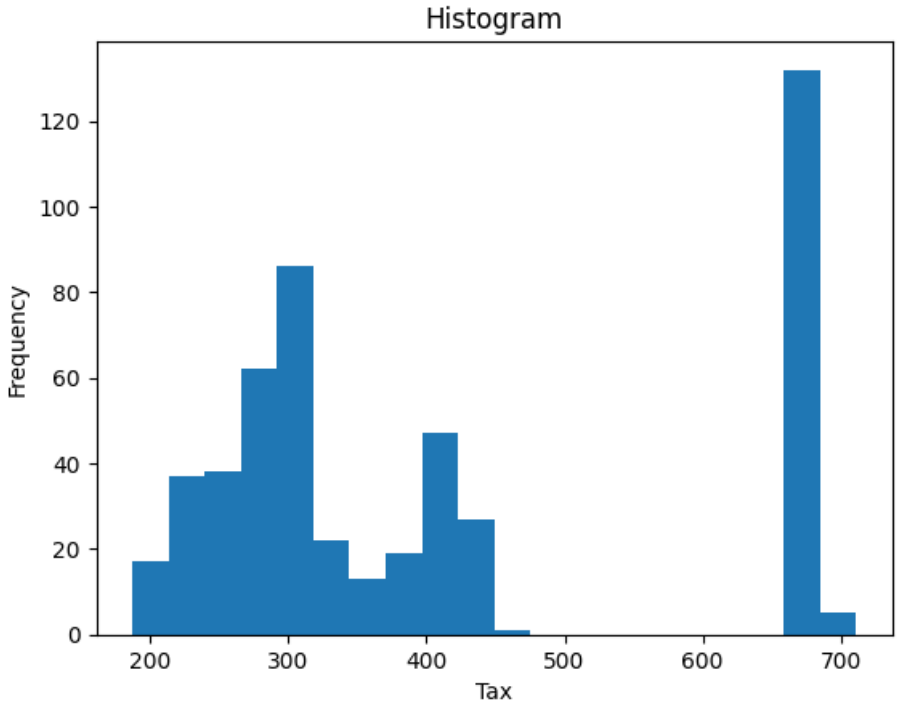



```
plt.scatter(df['ptratio'],df['medv'], color='blue')
plt.title("scatter plot")
plt.xlabel("Age")
plt.ylabel("Median value of owner-occupied homes")
plt.show()
```

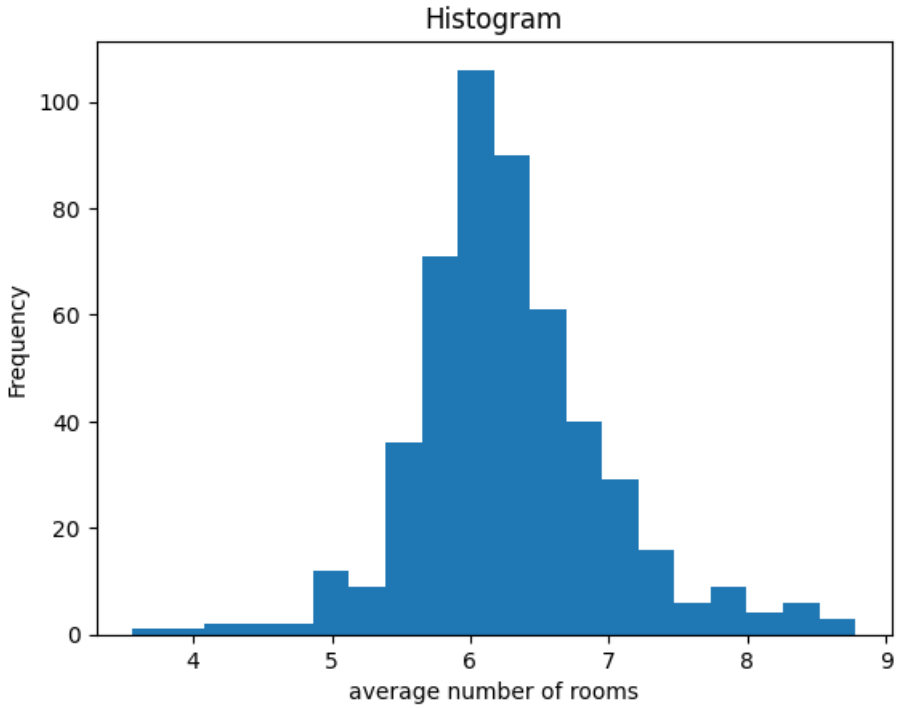



```
plt.hist(df['tax'], bins=20)
plt.title("Histogram")
plt.xlabel("Tax")
```

```
plt.ylabel("Frequency")
plt.show()
```

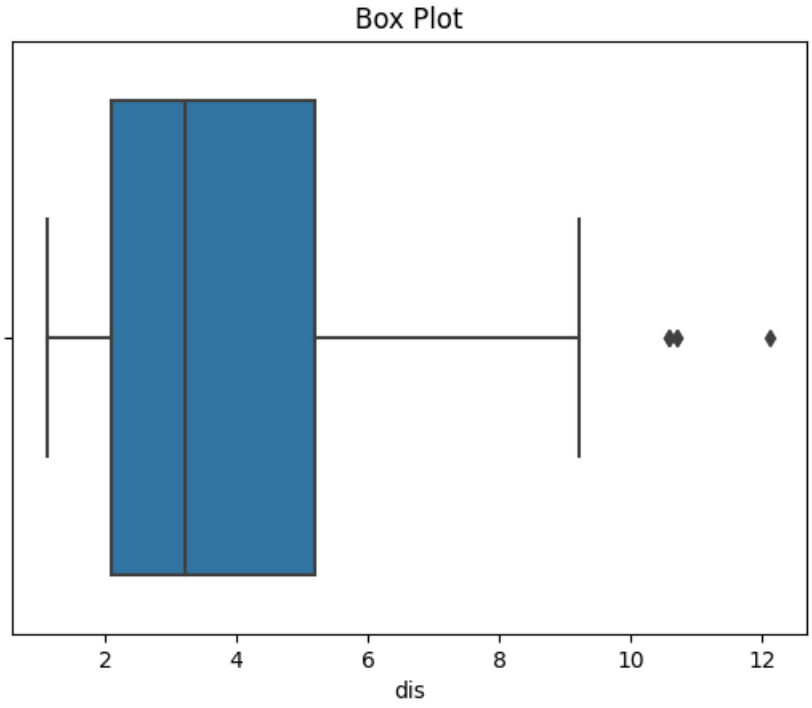



```
plt.hist(df['rm'], bins=20)
plt.title("Histogram")
plt.xlabel("average number of rooms")
plt.ylabel("Frequency")
plt.show()
```



```
sb.boxplot(x="dis", data=df)
plt.title("Box Plot")
```

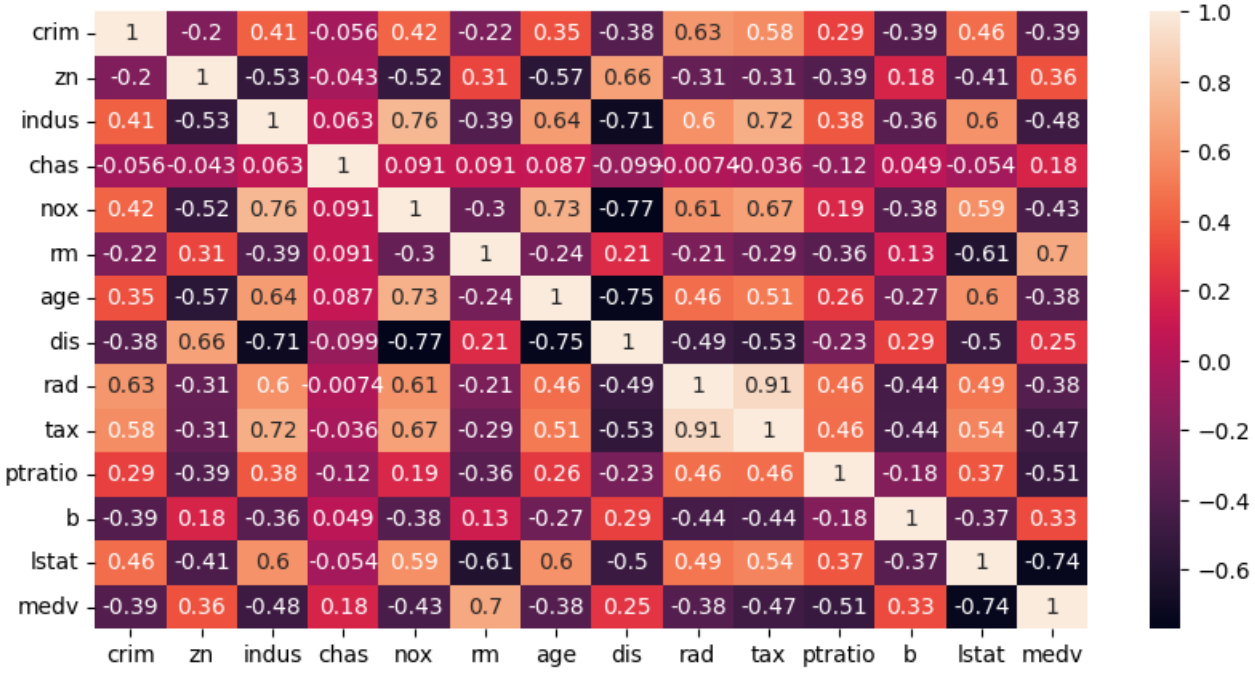
```
Text(0.5, 1.0, 'Box Plot')
```



Heatmap Generated

```
corr=df.corr()
plt.figure(figsize=(10,5))
sb.heatmap(corr,annot=True)
```

```
<Axes: >
```



Applying Algorithms

```
x=df.drop("medv",axis=1)
y=df["medv"]
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_lrp=lr.predict(x_test)
```

```
print(y_lrp)
```

```
[24.88963777 23.72141085 29.36499868 12.12238621 21.44382254 19.2834443
 20.49647539 21.36099298 18.8967118  19.9280658   5.12703513 16.3867396
 17.07776485  5.59375659 39.99636726 32.49654668 22.45798809 36.85192327
 30.86401089 23.15140009 24.77495789 24.67187756 20.59543752 30.35369168
 22.41940736 10.23266565 17.64816865 18.27419652 35.53362541 20.96084724
 18.30413012 17.79262072 19.96561663 24.06127231 29.10204874 19.27774123
 11.15536648 24.57560579 17.5862644  15.49454112 26.20577527 20.86304693
 22.31460516 15.60710156 23.00363104 25.17247952 20.11459464 22.90256276
 10.0380507  24.28515123 20.94127711 17.35258791 24.52235405 29.95143046
 13.42695877 21.72673066 20.7897053  15.49668805 13.98982601 22.18377874
 17.73047814 21.58869165 32.90522136 31.11235671 17.73252635 32.76358681
 18.7124637  19.78693475 19.02958927 22.89825374 22.96041622 24.02555703
 30.72859326 28.83142691 25.89957059  5.23251817 36.72183202 23.77267249
 27.26856352 19.29492159 28.62304496 19.17978838 18.97185995 37.82397662
 39.22012647 23.71261106 24.93076217 15.88545417 26.09845751 16.68819641
 15.83515991 13.10775597 24.71583588 31.25165267 22.16640989 20.25087212
  0.59025319 25.44217132 15.57178328 17.93719475 25.30588844 22.3732326 ]
```

```
pred_df = pd.DataFrame({"Predicted_Prices": y_lrp, "Actual_Prices": y_test})
print("Predicted and Actual Price Data frame")
print(pred_df)
```

```
Predicted and Actual Price Data frame
     Predicted_Prices  Actual_Prices
329         24.889638           22.6
371         23.721411           50.0
219         29.364999           23.0
403         12.122386            8.3
78          21.443823           21.2
..                ...            ...
56          25.442171           24.7
455         15.571783           14.1
60          17.937195           18.7
213         25.305888           28.1
108         22.373233           19.8

[102 rows x 2 columns]
```

Root Mean Square Error

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_test, y_lrp))


print(rms)
```

```
     5.783509315085123
```

Conclusion:

1. What features have been chosen to develop the model? Justify the features chosen to estimate the price of a house.

   CRIM (Per Capita Crime Rate): The crime rate in a neighborhood can significantly impact property values. Higher crime rates might lead to lower property values due to safety concerns.

   ZN (Proportion of Residential Land Zoned for Large Lots): This feature could be indicative of the neighborhood's overall development and zoning regulations. Larger residential lots might be associated with more spacious and potentially higher-priced properties.

   INDUS (Proportion of Non-Retail Business Acres per Town): Areas with a higher proportion of non-retail businesses might have different property value dynamics compared to more residential areas.

   CHAS (Charles River Dummy Variable): This binary variable indicates whether a property is located along the Charles River. Properties with river views or access might command higher prices.

   NOX (Nitric Oxides Concentration): Pollution levels can impact the desirability of an area. Lower pollution levels might be associated with higher property values.

   RM (Average Number of Rooms per Dwelling): The size of a property, as indicated by the number of rooms, is a strong predictor of its value. Larger properties generally command higher prices.

   AGE (Proportion of Owner-Occupied Units Built Prior to 1940): The age of a property or neighborhood can influence its attractiveness. Older properties might have historic charm but could also require more maintenance.

   DIS (Weighted Distances to Employment Centers): Proximity to employment centers can affect property values. Properties closer to workplaces might be more valuable due to reduced commuting times.

   RAD (Index of Accessibility to Radial Highways): Easy access to highways can be desirable, as it facilitates commuting and transportation, potentially influencing property values.

   TAX (Property Tax Rate): Higher property tax rates might negatively impact property values, as potential buyers consider ongoing expenses.

PTRATIO (Pupil-Teacher Ratio): The quality of local schools can affect property values. Lower pupil-teacher ratios might indicate better educational opportunities and thus attract higher prices.

B (Proportion of Black Residents: This feature could be used to account for racial demographics, which can influence property values due to historical and social factors.

LSTAT (Percentage of Lower Status Population): This can provide insights into the socioeconomic status of the neighborhood, which can affect property values.

2. Comment on the Mean Squared Error calculated.

Mean Squared Error is a commonly used metric in statistics and machine learning to measure the average squared difference between predicted values and actual values. It's often used to evaluate the performance of regression models or any situation where you are making continuous predictions.

Mathematically, the formula for MSE is:

$MSE = (1/n) * \Sigma(y_i - \hat{y}_i)^2$

Here in this experiment the mean square error obtained is 5.78