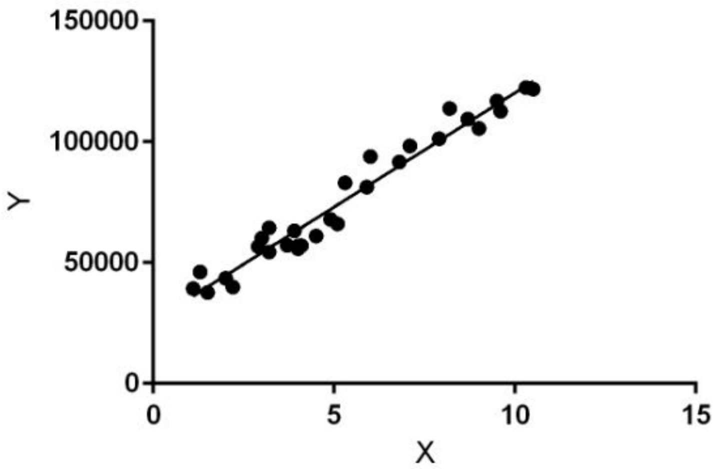
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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:

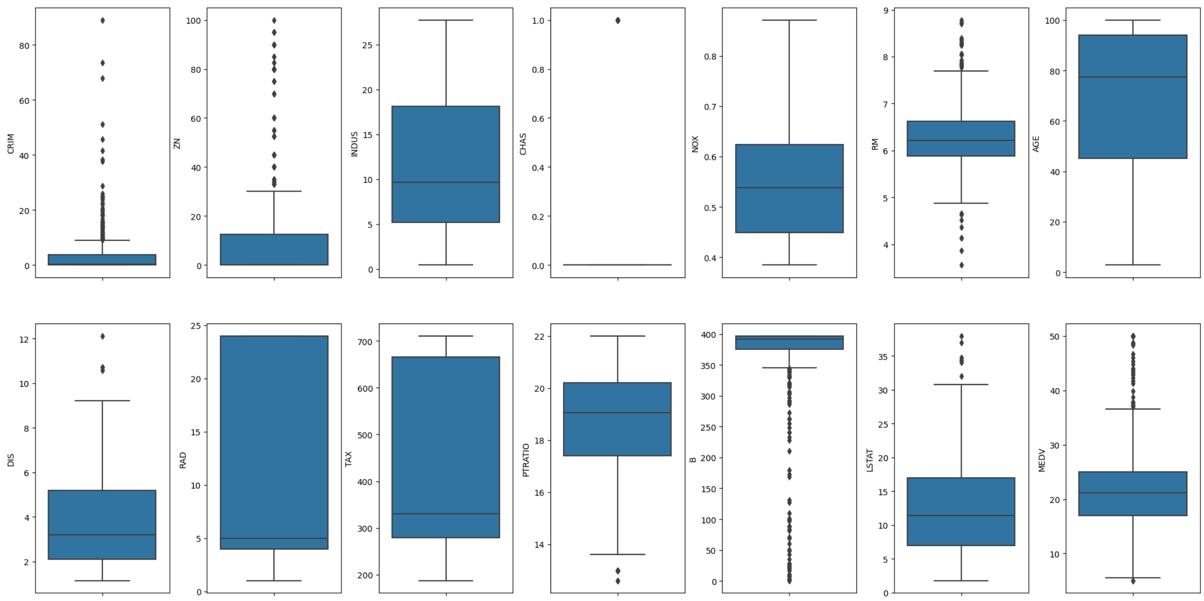
```
import numpy as np
import pandas as pd
import os
from pandas import read_csv
column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
data = read_csv('/content/housing.csv', header=None, delimiter=r"\s+",
names=column_names)
print(data.head(5))
```

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 1 296.0 1 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242.0 2 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242.0 3 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222.0 4 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222.0 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

```
print(data.describe())
```

CRIM ZN INDUS CHAS NOX RM \ count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 mean 3.613524 11.363636 11.136779 0.069170 0.554695 6.284634 std 8.601545 23.322453 6.860353 0.253994 0.115878 0.702617 min 0.006320 0.000000 0.460000 0.000000 0.385000 3.561000 25% 0.082045 0.000000 5.190000 0.000000 0.449000 5.885500 50% 0.256510 0.000000 9.690000 0.000000 0.538000 6.208500 75% 3.677083 12.500000 18.100000 0.000000 0.624000 6.623500 max 88.976200 100.000000 27.740000 1.000000 0.871000 8.780000 AGE DIS RAD TAX PTRATIO B \ count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 mean 68.574901 3.795043 9.549407 408.237154 18.455534 356.674032 std 28.148861 2.105710 8.707259 168.537116 2.164946 91.294864 min 2.900000 1.129600 1.000000 187.000000 12.600000 0.320000 25% 45.025000 2.100175 4.000000 279.000000 17.400000 375.377500 50% 77.500000 3.207450 5.000000 330.000000 19.050000 391.440000 75% 94.075000 5.188425 24.000000 666.000000 20.200000 396.225000 max 100.000000 12.126500 24.000000 711.000000 22.000000 396.900000 LSTAT MEDV count 506.000000 506.000000 mean 12.653063 22.532806 std 7.141062 9.197104 min 1.730000 5.000000 25% 6.950000 17.025000 50% 11.360000 21.200000 75% 16.955000 25.000000 max 37.970000 50.000000

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in data.items():
    sns.boxplot(y=k, data=data, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```

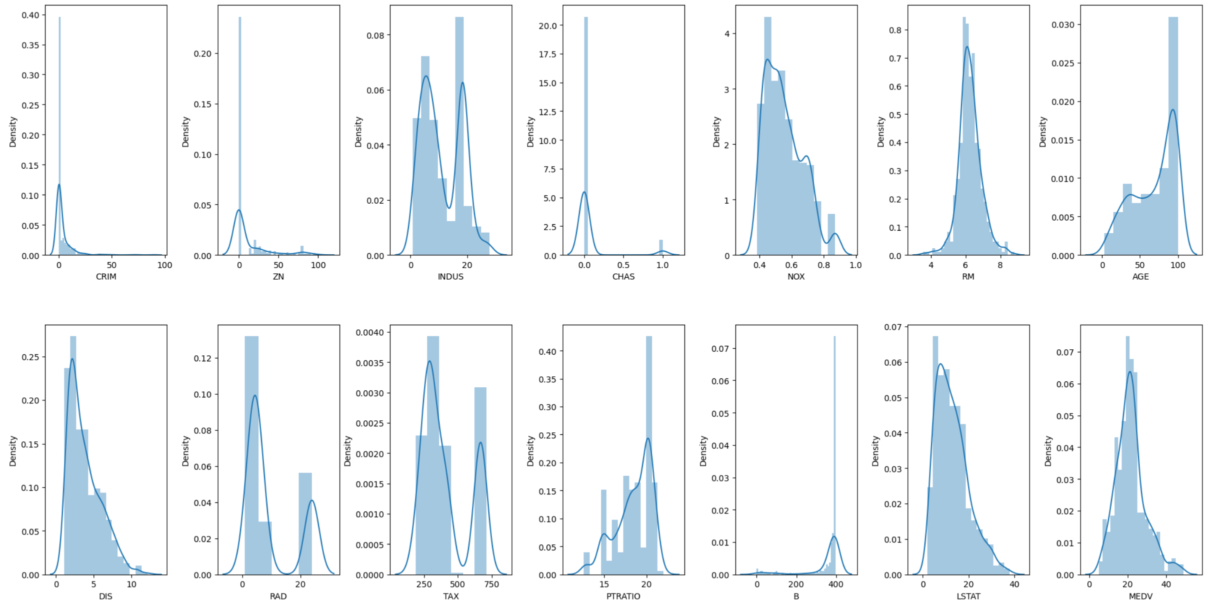



```python
for k, v in data.items():
        q1 = v.quantile(0.25)
        q3 = v.quantile(0.75)
        irq = q3 - q1
        v_col = v[(v <= q1 - 1.5 * irq) | (v >= q3 + 1.5 * irq)]
        perc = np.shape(v_col)[0] * 100.0 / np.shape(data)[0]
        print("Column %s outliers = %.2f%%" % (k, perc))
```

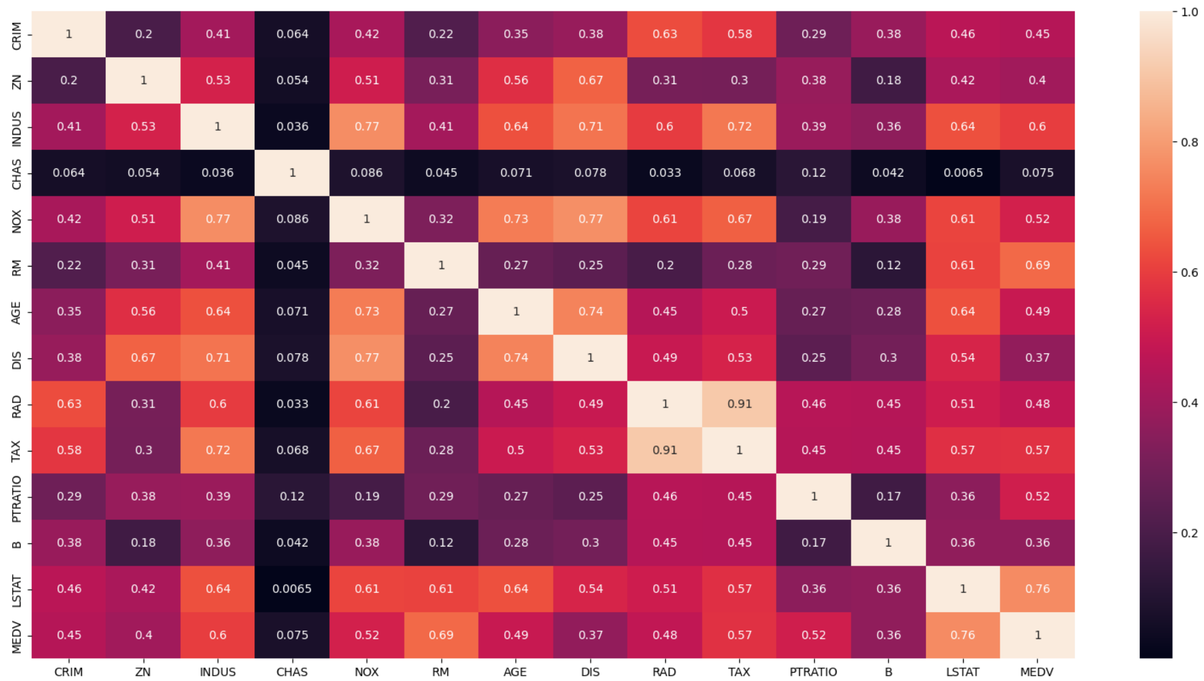
```
Column CRIM outliers = 13.04%
 Column ZN outliers = 13.44%
Column INDUS outliers = 0.00%
Column CHAS outliers = 100.00%
Column NOX outliers = 0.00%
 Column RM outliers = 5.93%
Column AGE outliers = 0.00%
Column DIS outliers = 0.99%
Column RAD outliers = 0.00%
Column TAX outliers = 0.00%
Column PTRATIO outliers = 2.96%
Column B outliers = 15.22%
Column LSTAT outliers = 1.38%
Column MEDV outliers = 7.91%
```

```python
data = data[~(data['MEDV'] >= 50.0)]
print(np.shape(data))
(490, 14)
```

```python
fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in data.items():
    sns.distplot(v, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```

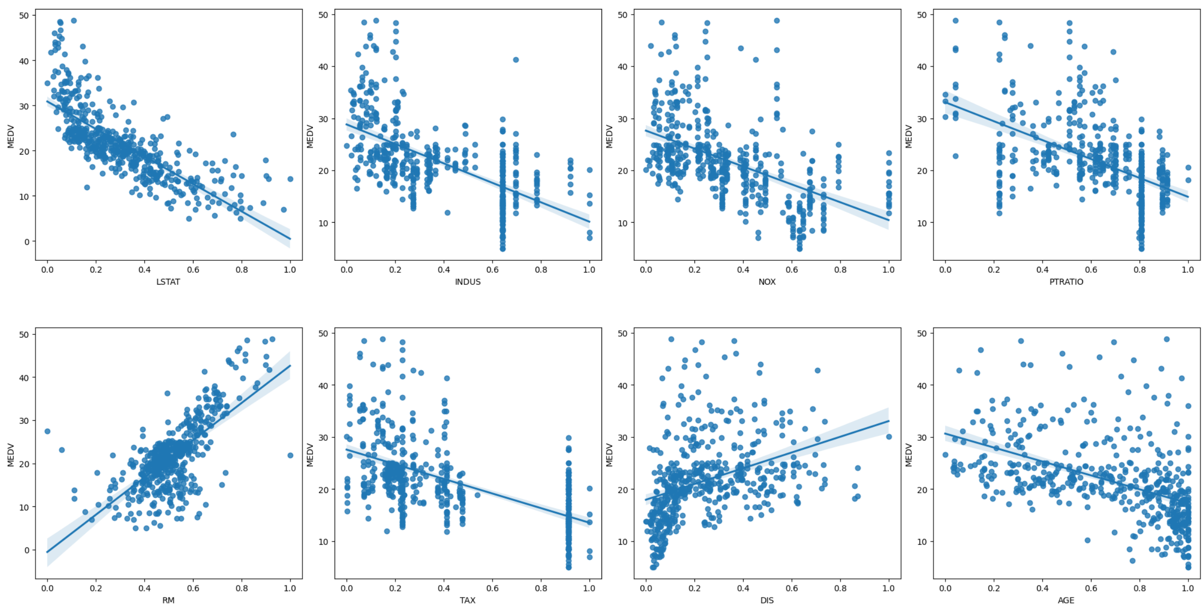



```python
plt.figure(figsize=(20, 10))
sns.heatmap(data.corr().abs(),  annot=True)
```




```python
from sklearn import preprocessing
# Let's scale the columns before plotting them against MEDV
min_max_scaler = preprocessing.MinMaxScaler()
```

```
column_sels = ['LSTAT', 'INDUS', 'NOX', 'PTRATIO', 'RM', 'TAX', 'DIS',
'AGE']
x = data.loc[:,column_sels]
y = data['MEDV']
x          =          pd.DataFrame(data=min_max_scaler.fit_transform(x),
columns=column_sels)
fig, axs = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for i, k in enumerate(column_sels):
    sns.regplot(y=y, x=x[k], ax=axs[i])
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```



[29]

0s

```
from sklearn import datasets, linear_model

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import KFold

import numpy as np

l_regression = linear_model.LinearRegression()

kf = KFold(n_splits=10)

min_max_scaler = preprocessing.MinMaxScaler()

x_scaled = min_max_scaler.fit_transform(x)
```

```
scores = cross_val_score(l_regression, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')

print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))

scores_map = {}

scores_map['LinearRegression'] = scores

l_ridge = linear_model.Ridge()

scores = cross_val_score(l_ridge, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')

scores_map['Ridge'] = scores

print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))

from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import PolynomialFeatures

model = make_pipeline(PolynomialFeatures(degree=3), linear_model.Ridge())

scores = cross_val_score(model, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')

scores_map['PolyRidge'] = scores

print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
MSE: -21.48 (+/- 18.15)
MSE: -20.08 (+/- 13.00)
MSE: -13.61 (+/- 6.78)
```

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV

svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
scores = cross_val_score(svr_rbf, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')
scores_map['SVR'] = scores
print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))

MSE: -11.62 (+/- 5.91)

from sklearn.tree import DecisionTreeRegressor

desc_tr = DecisionTreeRegressor(max_depth=5)
```

```
scores = cross_val_score(desc_tr, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')
scores_map['DecisionTreeRegressor'] = scores
print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
```

```
MSE: -17.45 (+/- 5.91)
```

[32]

0s

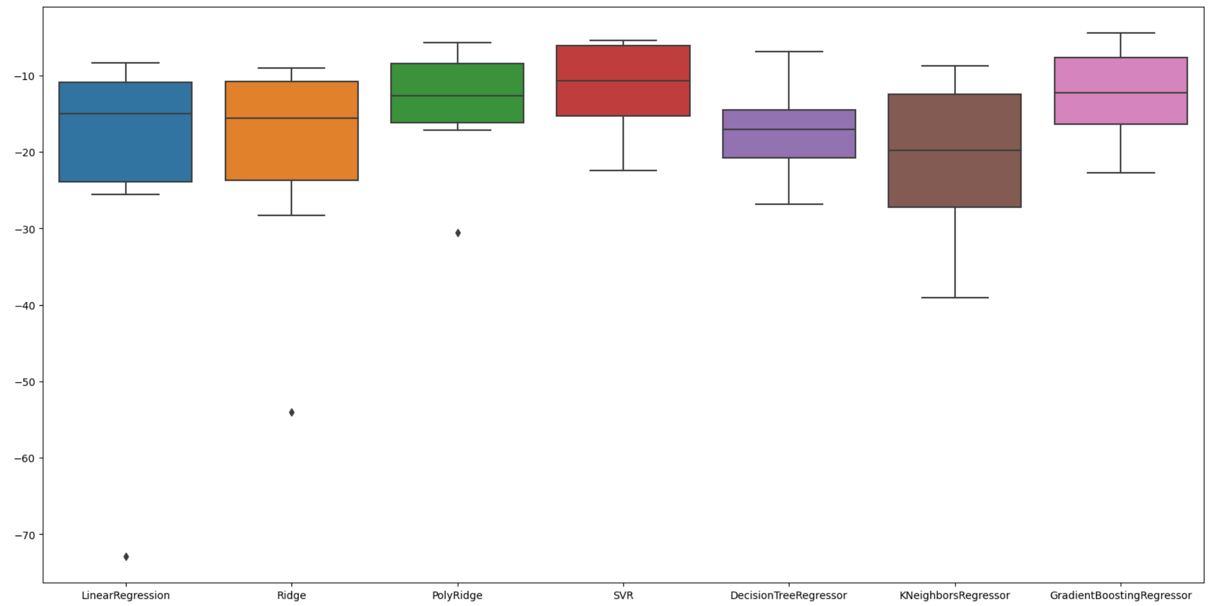
```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(n_neighbors=7)

scores = cross_val_score(knn, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')

scores_map['KNeighborsRegressor'] = scores

print("KNN Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
```

KNN Accuracy: -20.77 (+/- 9.54)

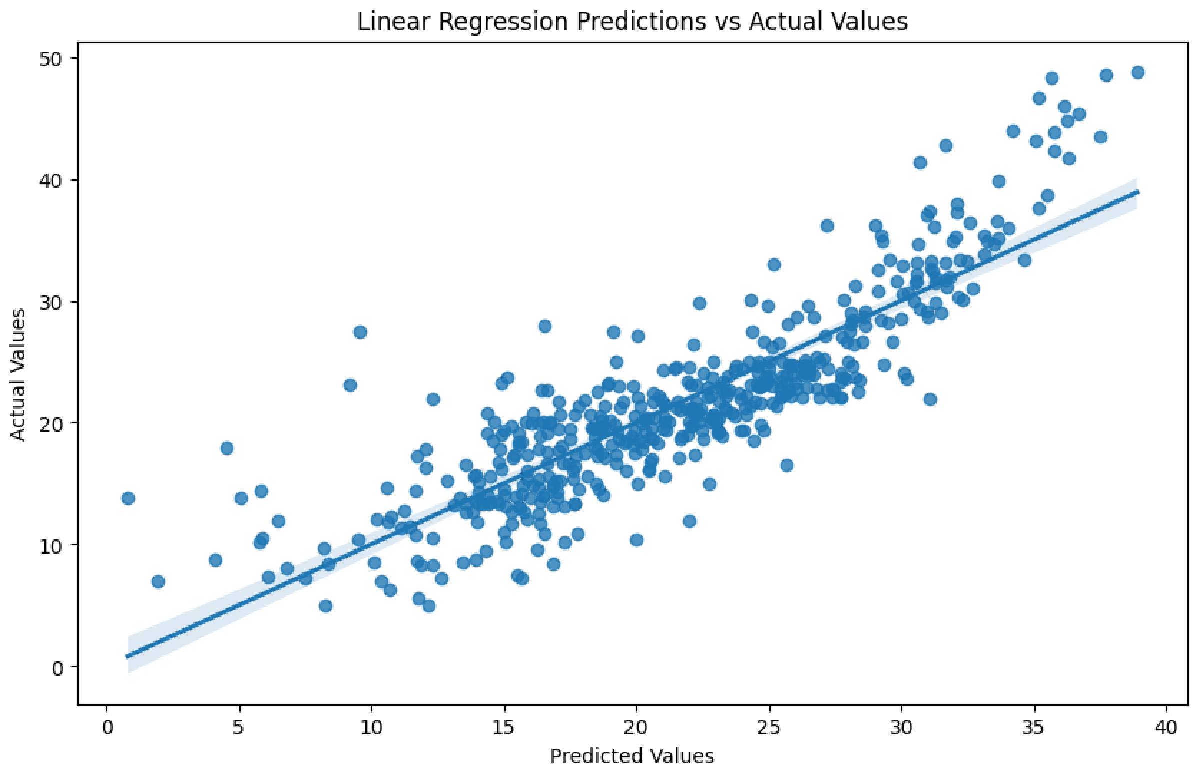
```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(alpha=0.9,learning_rate=0.05, max_depth=2, min_samples_leaf=5, min_samples_split=2, n_estimators=100, random_state=30)

scores = cross_val_score(gbr, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')
scores_map['GradientBoostingRegressor'] = scores
print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
```

```
MSE: -12.39 (+/- 5.86)
```

```
plt.figure(figsize=(20, 10))

scores_map = pd.DataFrame(scores_map)

sns.boxplot(data=scores_map)
```




```
from sklearn.linear_model import LinearRegression

l_regression = LinearRegression()

l_regression.fit(x_scaled, y)

plt.figure(figsize=(10, 6))

sns.regplot(y=y, x=l_regression.predict(x_scaled))

plt.xlabel('Predicted Values')

plt.ylabel('Actual Values')
plt.title('Linear Regression Predictions vs Actual Values')
plt.show()
scores = cross_val_score(l_regression, x_scaled, y, cv=kf, scoring='neg_mean_squared_error')
print("Linear Regression MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
```



Linear Regression MSE: -21.48 (+/- 18.15)

Conclusion:

1. What are features have been chosen to develop the model? Justify the features chosen to estimate the price of a house.
   - The X coordinate is chosen from the "LSTAT" column (% lower status of the population), while the Y coordinate is derived from the "MEDV" column (median value of owner-occupied homes in $1000's). In this context, "LSTAT" represents the percentage of lower status in the population, and "MEDV" stands for the median value of owner-occupied homes in thousands of dollars.
   - The generated heat map shows a significant correlation between LSTAT and MEDV. The scatterplot reveals that prices generally decline with higher LSTAT values.
   - As a result, the dataset is divided into training and test sets, with 80% of the data allocated for training and 20% for testing the trained model. The model development process involves using the "LSTAT" and "MEDV" columns to predict values through Linear Regression.

2. Comment on the Mean Squared Error calculated.

    - The calculated Mean Squared Error (MSE) is a crucial indicator of how well the model's predicted values align with the actual data points. In this context, a lower MSE signifies that the model's predictions are, on average, closer to the true values. This suggests that the model has a stronger ability to capture the underlying patterns in the data, resulting in more accurate predictions. A higher MSE, on the other hand, would imply a larger divergence between predicted and actual values, indicating a less effective fit. Therefore, a lower MSE is desirable as it indicates a higher level of precision and reliability in the model's performance.