Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance: 27/07/23
Date of Submission: 17/08/23

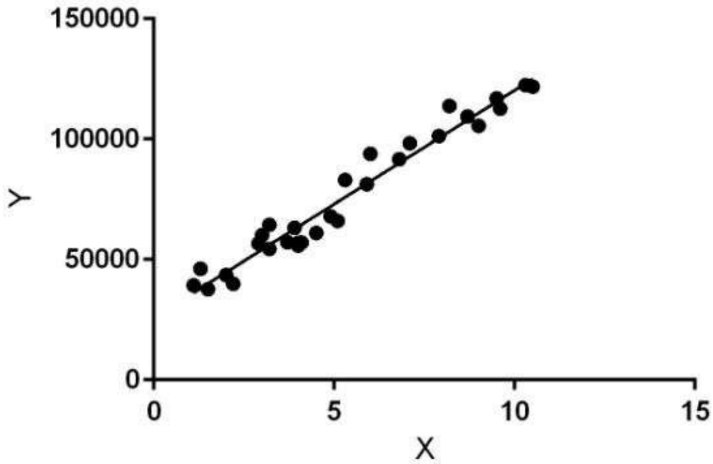
CSL701: Machine Learning Lab

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

In [9]:
```
# Importing the libraries
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [10]:
```
# Displaying the data
df = pd.read_csv('Boston Dataset.csv')
df.head(3)
```

Out[10]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83

In [11]:
```
df.drop(columns=['Unnamed: 0'], axis=0, inplace=True)
df.head()
```

Out[11]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	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	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	0.02729	0.0	7.07	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	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [1]:

```python
# Statistical info
df.describe()
```

Out[12]:

	crim	zn	indus	chas	nox	rm	age	
count	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
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12

In [13]:

```python
# datatype info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 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  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [1]:

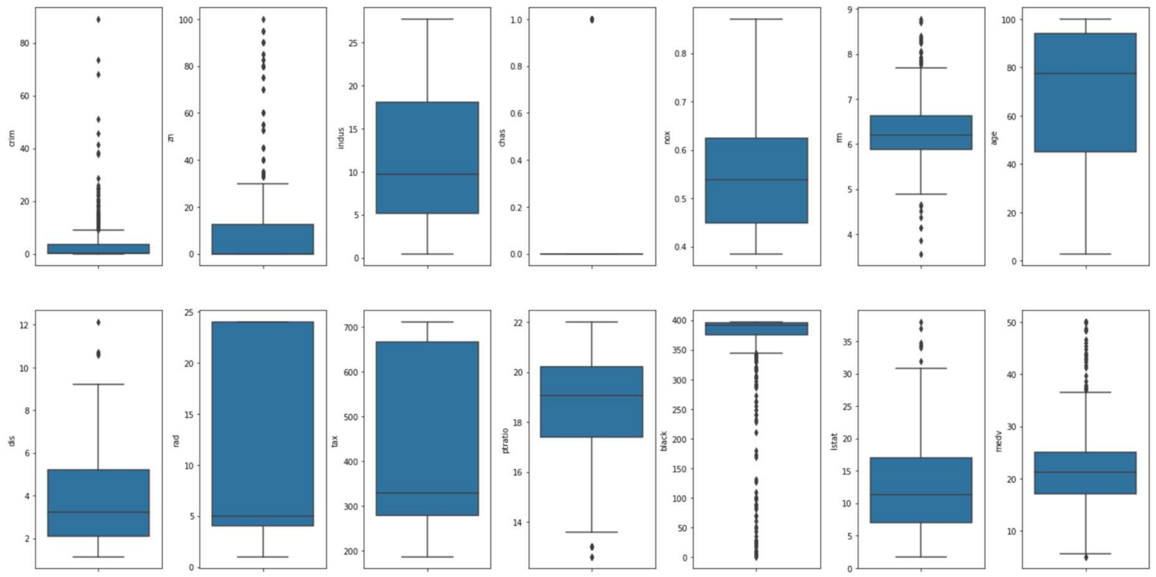
```python
# Checking for null values
df.isnull().sum()
```

Out[14]:

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

In [1]:

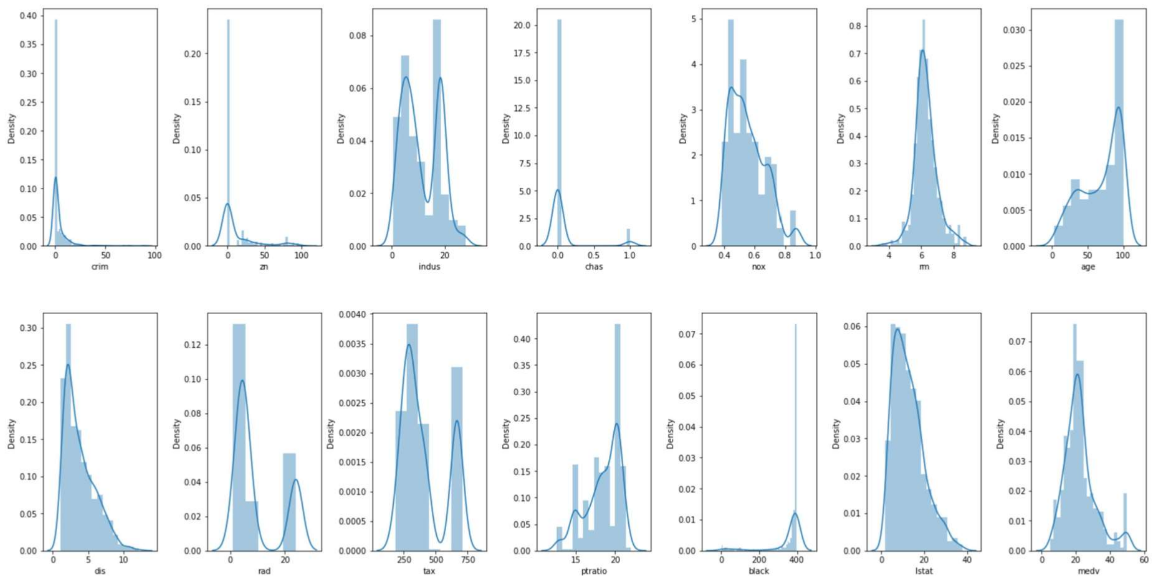
```
# Creating box plots for attributes
# box plots are used for indentifying outliners
# An outlier is an observation that lies an abnormal distance
# from other values in a random sample from a population

fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))  #7*2 = 14, since 1
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.boxplot(y=col, data=df, ax=ax[index])
    index +=1

# Hyper parameter tunning to display graph properly
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)

# The dot's in the box plot's are outliners
# By observing the below figure we can see that
# CRIM, ZM, B have many outliners
# To deal with outliners we can use min-max transformation or ignore the outli
```



In [1]:

6

```
# Create dist plot
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))  #7*2 = 14, since
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.distplot(value, ax=ax[index])
    index +=1

# Hyper parameter tunning to display graph properly
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)

# Left skewed - CRIM, ZN, DIS
# Right skewed - AGE, B
# Double bell - INDUS, RAD, TAX
# Complete uniform distribution - RM, LSTAT

# CRIM, ZN, TAX, B -> Min max normalization wil be done
```



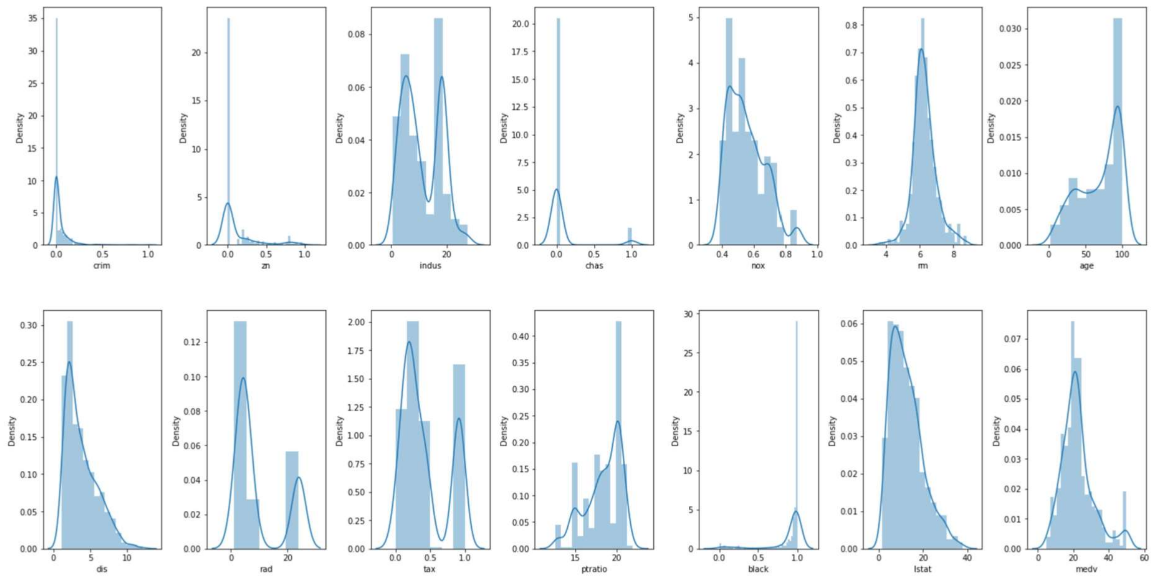
In [17]:

```
# Min-max normalization
cols = ['crim', 'zn', 'tax', 'black']
for col in cols:
    # Find minimum and maximum of that column
    minimum = min(df[col])
    maximum = max(df[col])
    df[col] = (df[col] - minimum) / (maximum - minimum)
```

In [1]:

8

```python
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))  #7*2 = 14, since
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.distplot(value, ax=ax[index])
    index +=1

# Hyper parameter tunning to display graph properly
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



In [19]:

```python
# standardization
from sklearn import preprocessing
scalar = preprocessing.StandardScaler()

# fit the data
scaled_cols = scalar.fit_transform(df[cols])
scaled_cols = pd.DataFrame(scaled_cols, columns=cols)
scaled_cols.head()
```

Out[19]:

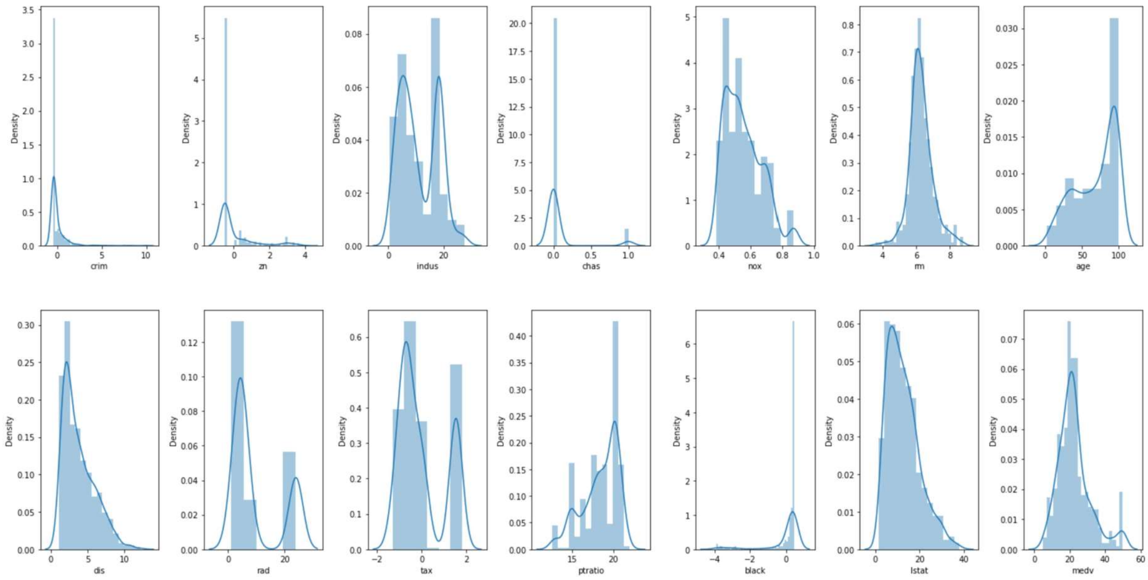
	crim	zn	tax	black
0	-0.419782	0.284830	-0.666608	0.441052
1	-0.417339	-0.487722	-0.987329	0.441052
2	-0.417342	-0.487722	-0.987329	0.396427
3	-0.416750	-0.487722	-1.106115	0.416163
4	-0.412482	-0.487722	-1.106115	0.441052

In [20]:

```python
for col in cols:
    df[col] = scaled_cols[col]
```

In [2]:

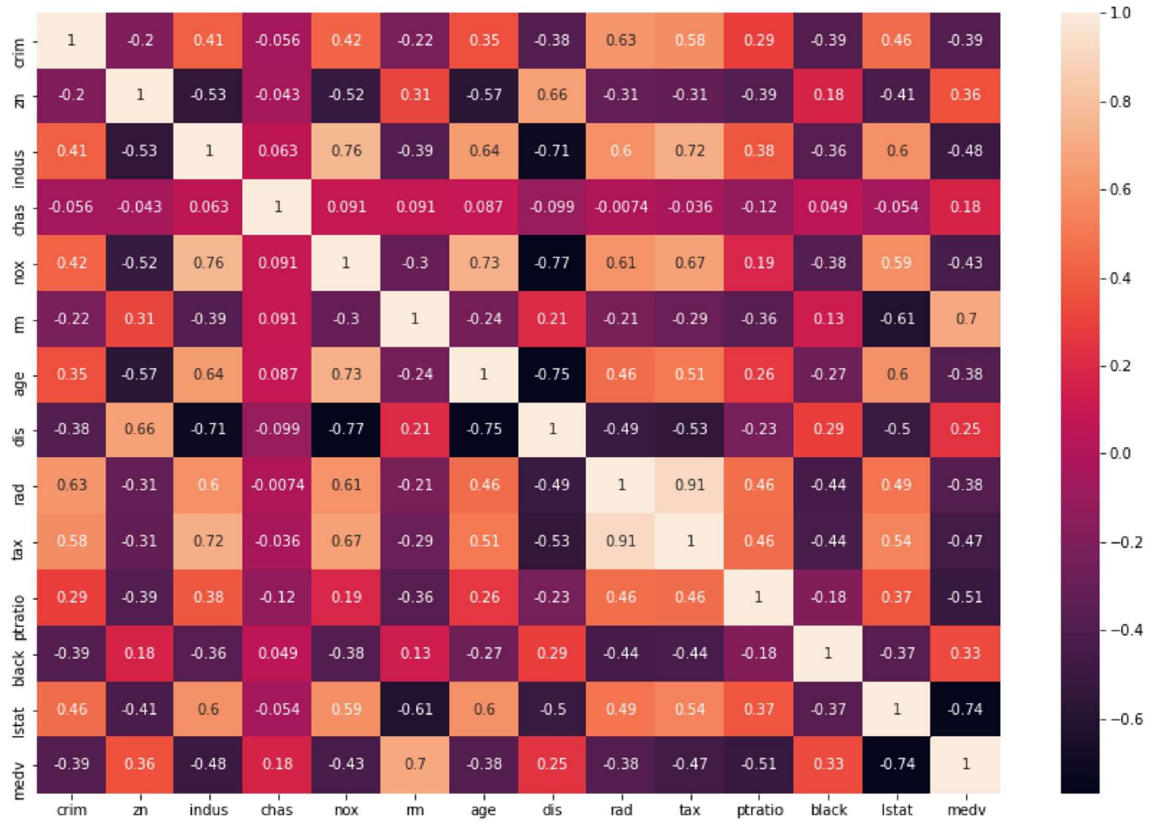
```
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))  #7*2 = 14, since 
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.distplot(value, ax=ax[index])
    index +=1

# Hyper parameter tunning to display graph properly
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



In [2]:

```python
# Coorelation matrix
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot=True)
```

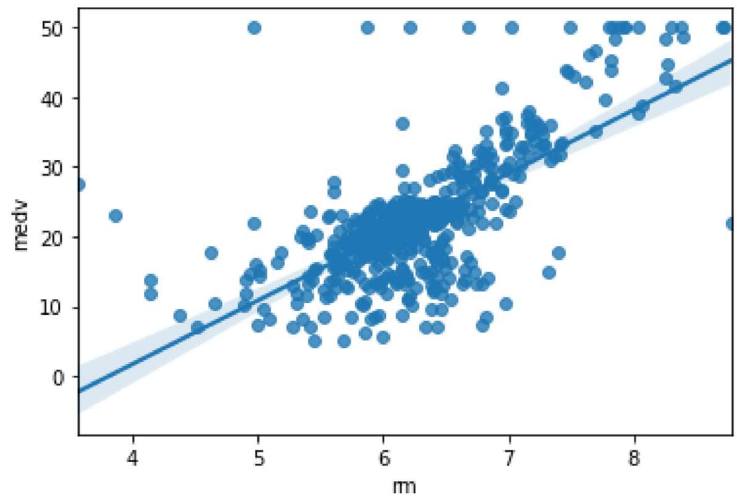
Out[22]: <AxesSubplot:>



In [23]:

```python
sns.regplot(y=df['medv'], x=df['rm'])
```

Out[23]: <AxesSubplot:xlabel='rm', ylabel='medv'>

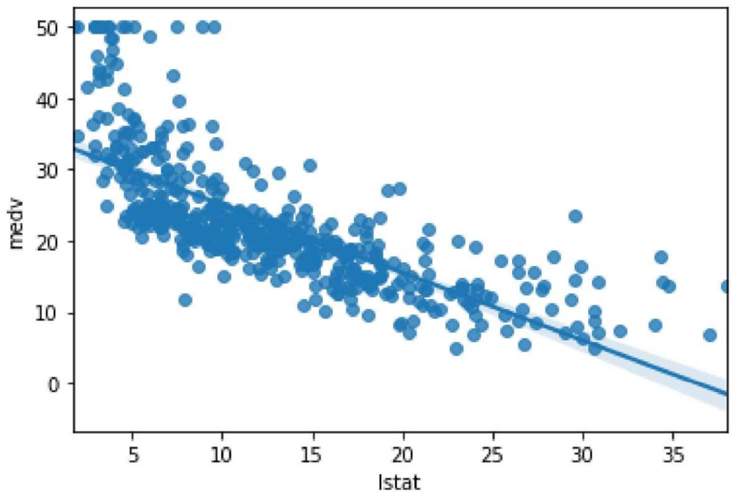


In [2]:

```python
sns.regplot(y=df['medv'], x=df['lstat'])
```

Out[24]:

```
<AxesSubplot:xlabel='lstat', ylabel='medv'>
```



In [25]:

```python
# input split
X = df.drop(columns=['medv', 'rad'], axis=1)
y = df['medv']
```

In [26]:

```python
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import mean_squared_error

from sklearn.linear_model import LinearRegression
model = LinearRegression()

# train the model
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
model.fit(X_train, y_train)

# predict the training set
pred = model.predict(X_test)

# perform cross-validation
cv_score = cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv=
cv_score = np.abs(np.mean(cv_score))

print("MSE:", mean_squared_error(y_test, pred))
print('CV Score:', cv_score)
```

```
MSE: 23.871005067364873
CV Score: 35.58136621076918
```

Conclusion:

1. The Features chosen to develop the model are as follows: -
    a. CRIM - per capita crime rate by town
    b. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
    c. INDUS - proportion of non-retail business acres per town.
    d. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
    e. NOX - nitric oxides concentration (parts per 10 million)
    f. RM - average number of rooms per dwelling
    g. AGE - proportion of owner-occupied units built prior to 1940
    h. DIS - weighted distances to five Boston employment centers
    i. TAX - full-value property-tax rate per $10,000
    j. PTRATIO - pupil-teacher ratio by town
    k. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    l. LSTAT - % lower status of the population

These features were chosen after performing appropriate feature engineering on the dataset such as standardization and min max normalization on the attributes CRIM, ZN, TAX and black These features also contributed in the prediction of the final outcome

2. The Mean Squared error obtained by our linear regression model was 23.87 which means our model is 78% accurate on the given test data.
3. The Overall Accuracy can be improved by performing tasks such as feature selection, cross-validation and hyper parameter tuning.