Linear Regression

1 Explain how a linear regression algorithm trains in your own words

Linear regration takes 2 dimentional values in form of X points (can be x1,x2,x3....a vector) and corrosponding Y point values.

Based of given values of X1 the corrosponding Y1 can be calculated with hellp of linear regression. Here X1 is the point which is not belongs to known values of X.

It tries to derive the optimum Y = mX + c a line equation which has minimum squar error for the given data.

These Data is also known as training data set and used to derive the model (here it is equation of a line). So, the equation can be used to derive unknown values of Y* for given X* valuues and this is how linear regration algorith works. To verify performance of the model, data set is divided into two parts training X,Y and testing X,*Y*.

Mean squared error and Coefficient of determination are used to check the performance of the model with respect to training and testing data set.

The model shift the line with possible values of m and c, such that error is minimum and in vase of multi dimentional X (multiple features) the equation is Y= m1X1 + m2X2 + m3X3 +.....+ mnXn + c and tries to calculate all m1, m2, ...mn and c for the same.

2 Load the sklearn boston dataset

```
#load and import all required libraries and functionalities
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

```
#Load data from the web, clean it for data frame format and store as row data (
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep = "\s+", skiprows = 22, header = None)
print('Data in two rows 11+3\n',raw_df.head())
```

```
df = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :3]])
target = raw_df.values[1::2, 2]
print('14 columns Data in one row\n',target)
```

```
Data in two rows 11+3
           0      1      2    3      4      5     6       7    8      9     10
0    0.00632  18.00   2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0  15.3
1  396.90000   4.98  24.00  NaN    NaN    NaN   NaN     NaN  NaN    NaN   NaN
2    0.02731   0.00   7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0  17.8
3  396.90000   9.14  21.60  NaN    NaN    NaN   NaN     NaN  NaN    NaN   NaN
4    0.02729   0.00   7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0  17.8
14 columns Data in one row
 [24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4 22.  17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.  20.8 21.2 20.3 28.  23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.  22.9 25.  20.6 28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8 19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.  20.3 20.5 17.3 18.8 21.4
 15.7 16.2 18.  14.3 19.2 19.6 23.  18.4 15.6 18.1 17.4 17.1 13.3 17.8
 14.  14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5 19.6 15.3 19.4
 17.  15.6 13.1 41.3 24.3 23.3 27.  50.  50.  50.  22.7 25.  50.  23.8
 23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9 37.2 39.8 36.2
 37.9 32.5 26.4 29.6 50.  32.  29.8 34.9 37.  30.5 36.4 31.1 29.1 50.
 33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.  22.6 24.4 22.5 24.4 20.
 21.7 19.3 22.4 28.1 23.7 25.  23.3 28.7 21.5 23.  26.7 21.7 27.5 30.1
 44.8 50.  37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.  24.  25.1 31.5
 23.7 23.3 22.  20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8
 29.6 42.8 21.9 20.9 44.  50.  36.  30.1 33.8 43.1 48.8 31.  36.5 22.8
 30.7 50.  43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.  33.2 33.1 29.1 35.1
 45.4 35.4 46.  50.  32.2 22.  20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9
 21.7 28.6 27.1 20.3 22.5 29.  24.8 22.  26.4 33.1 36.1 28.4 33.4 28.2
 22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1 21.  23.8 23.1
 20.4 18.5 25.  24.6 23.  22.2 19.3 22.6 19.8 17.1 19.4 22.2 20.7 21.1
 19.5 18.5 20.6 19.  18.7 32.7 16.5 23.9 31.2 17.5 17.2 23.1 24.5 26.6
 22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.  19.9 20.8 16.8
 21.9 27.5 21.9 23.1 50.  50.  50.  50.  50.  13.8 13.8 15.  13.9 13.3
 13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2 11.5 15.1 23.2
  9.7 13.8 12.7 13.1 12.5  8.5  5.   6.3  5.6  7.2 12.1  8.3  8.5  5.
 11.9 27.9 17.2 27.5 15.  17.2 17.9 16.3  7.   7.2  7.5 10.4  8.8  8.4
 16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.   9.5 14.5 14.1 16.1 14.3
 11.7 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8 11.8 14.9 12.6
 14.1 13.  13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9 20.  16.4 17.7
 19.5 20.2 21.4 19.9 19.  19.1 19.1 20.1 19.9 19.6 23.2 29.8 13.8 13.3
 16.7 12.  14.6 21.4 23.  23.7 25.  21.8 20.6 21.2 19.1 20.6 15.2  7.
  8.1 13.6 20.1 21.8 24.5 23.1 19.7 18.3 21.2 17.5 16.8 22.4 20.6 23.9
 22.  11.9]
```

```
#convert target to data frame format and apply column names for all 14 columns
data = pd.DataFrame(df)
data.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD'
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63

3 Visualize the dataset using graphs

```
#Get idea about each columns statistics
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AG
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57490
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14886
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.90000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.02500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.50000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.07500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00000

Scatter plot for all columns 1 to 13 with table MEDV

Observations

Positive correlation: RM, DIS

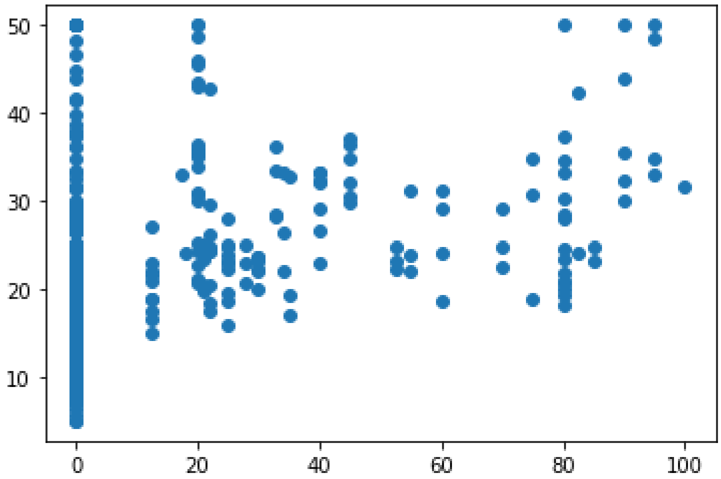
Negative correlation: AGE, LSTAT

```
plt.scatter(data['CRIM'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc937864d10>
```

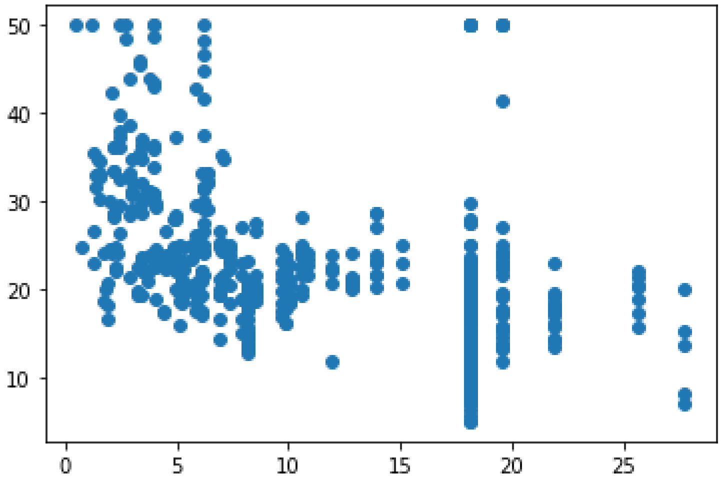

```
plt.scatter(data['ZN'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc9377ac6d0>
```



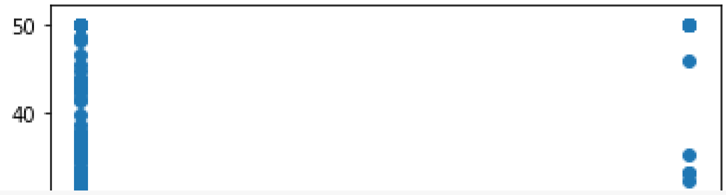

```
plt.scatter(data['INDUS'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc9378acbd0>
```



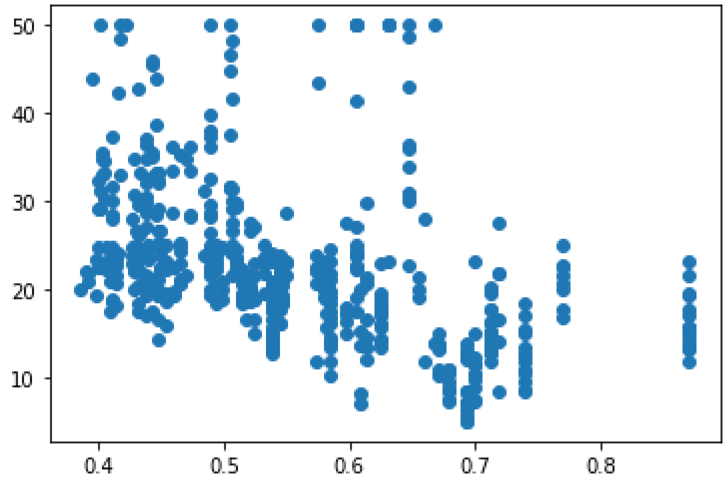

```
plt.scatter(data['CHAS'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc9377feb90>
```



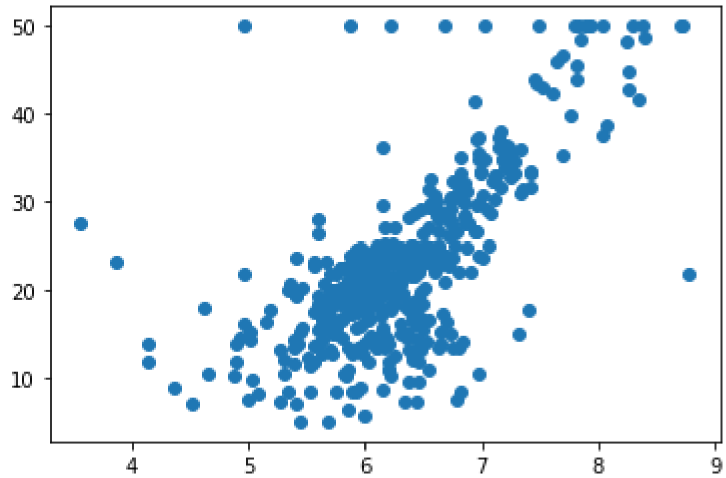

```
1 plt.scatter(data['NOX'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc934063510>
```



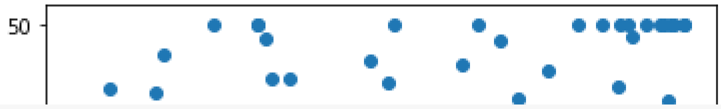

```
1   plt.scatter(data['RM'],data['MEDV'])
```

```
<matplotlib.collections.PathCollection at 0x7fc933fce710>
```



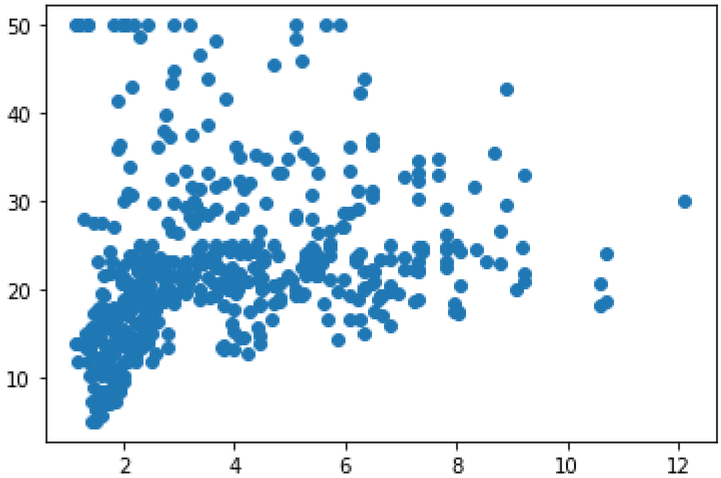

```
1 plt.scatter(data['AGE'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933f2fb10>



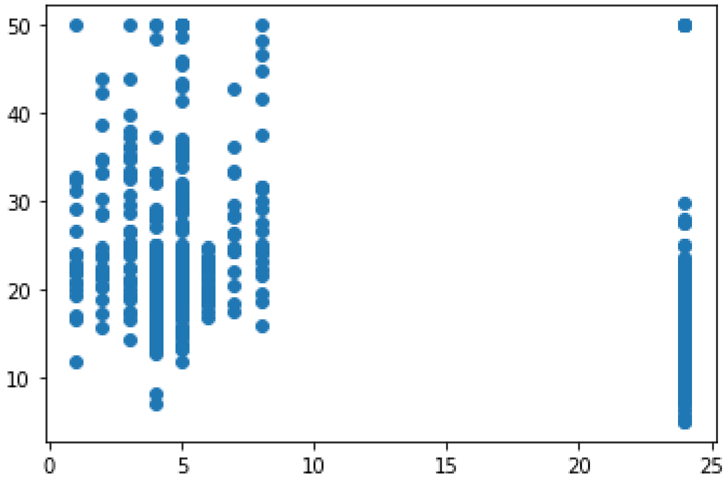

```
plt.scatter(data['DIS'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933f1e6d0>




```
plt.scatter(data['RAD'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933edee50>

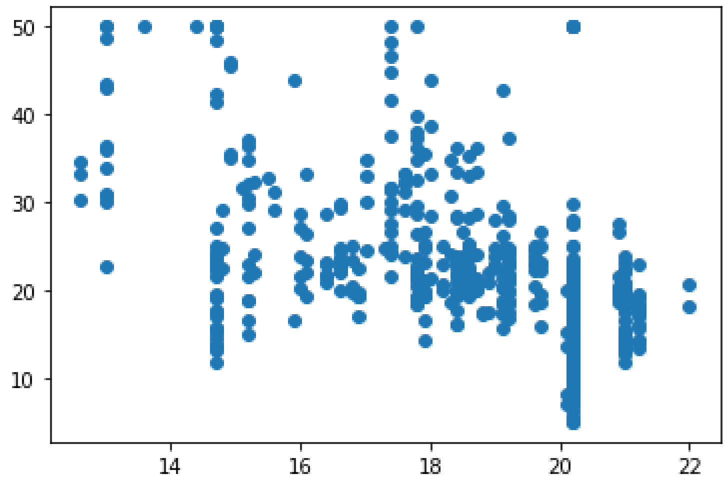



```
plt.scatter(data['TAX'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933e6aa50>

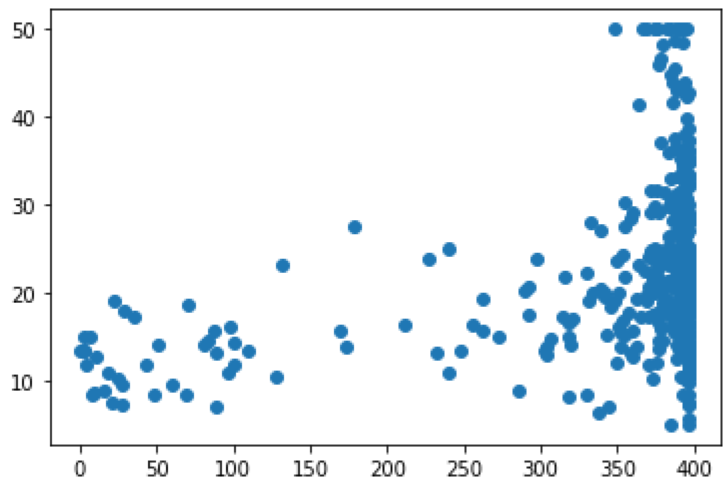
```
plt.scatter(data['PTRATIO'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933dd4ed0>



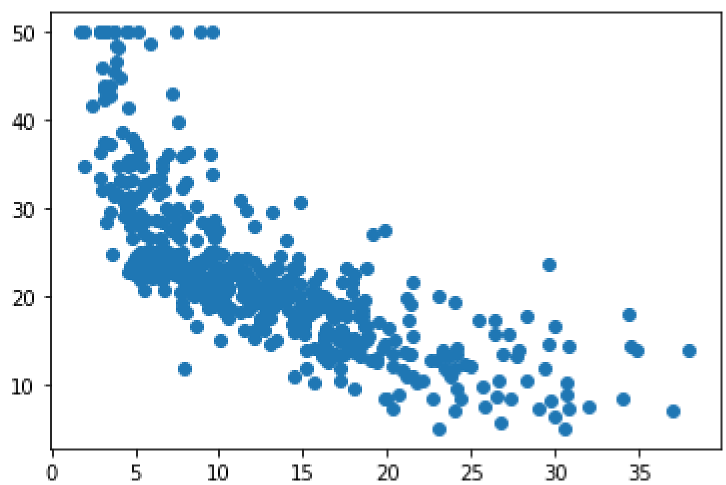

```
plt.scatter(data['B'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933d3ce90>




```
plt.scatter(data['LSTAT'],data['MEDV'])
```

<matplotlib.collections.PathCollection at 0x7fc933d25050>

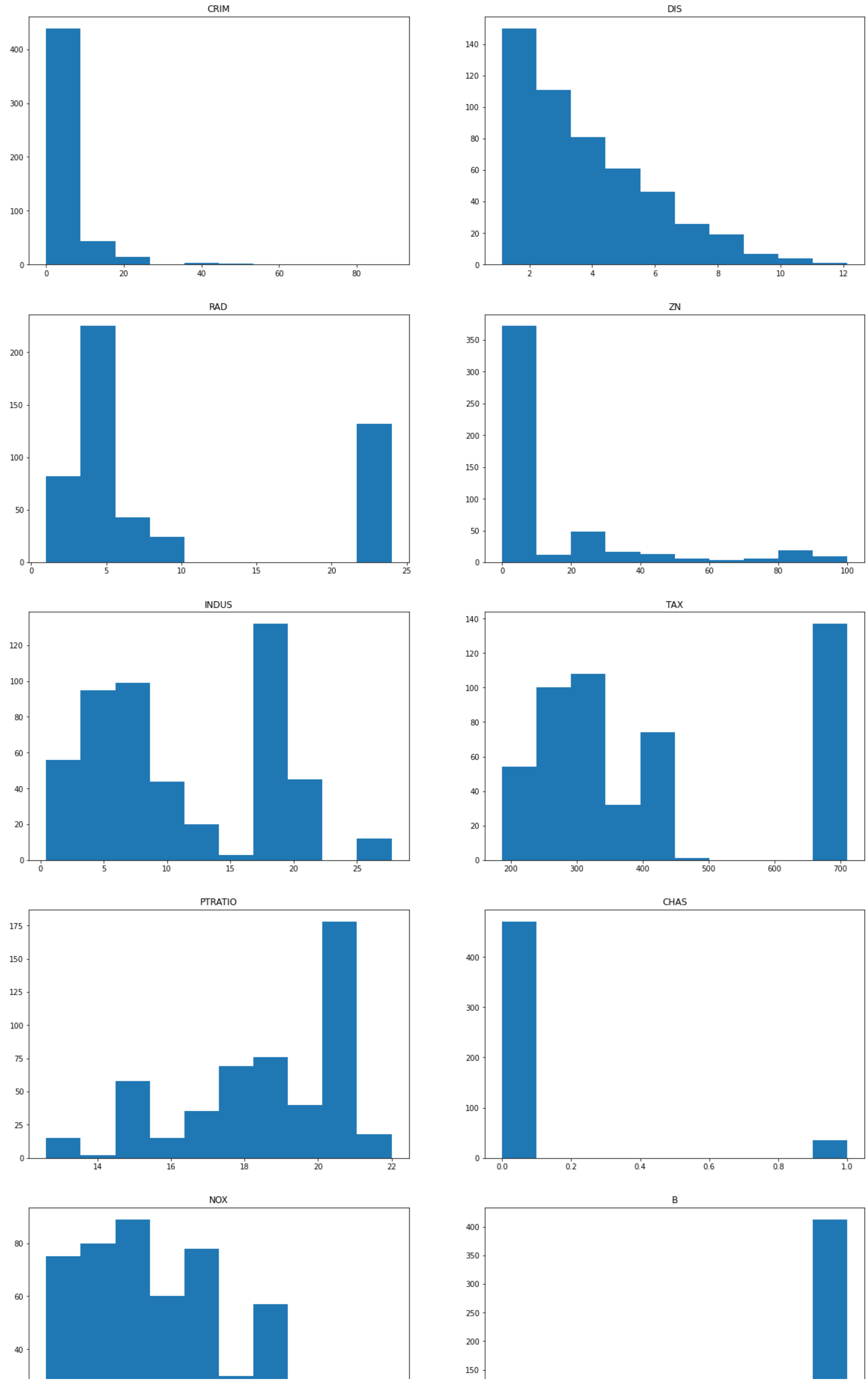


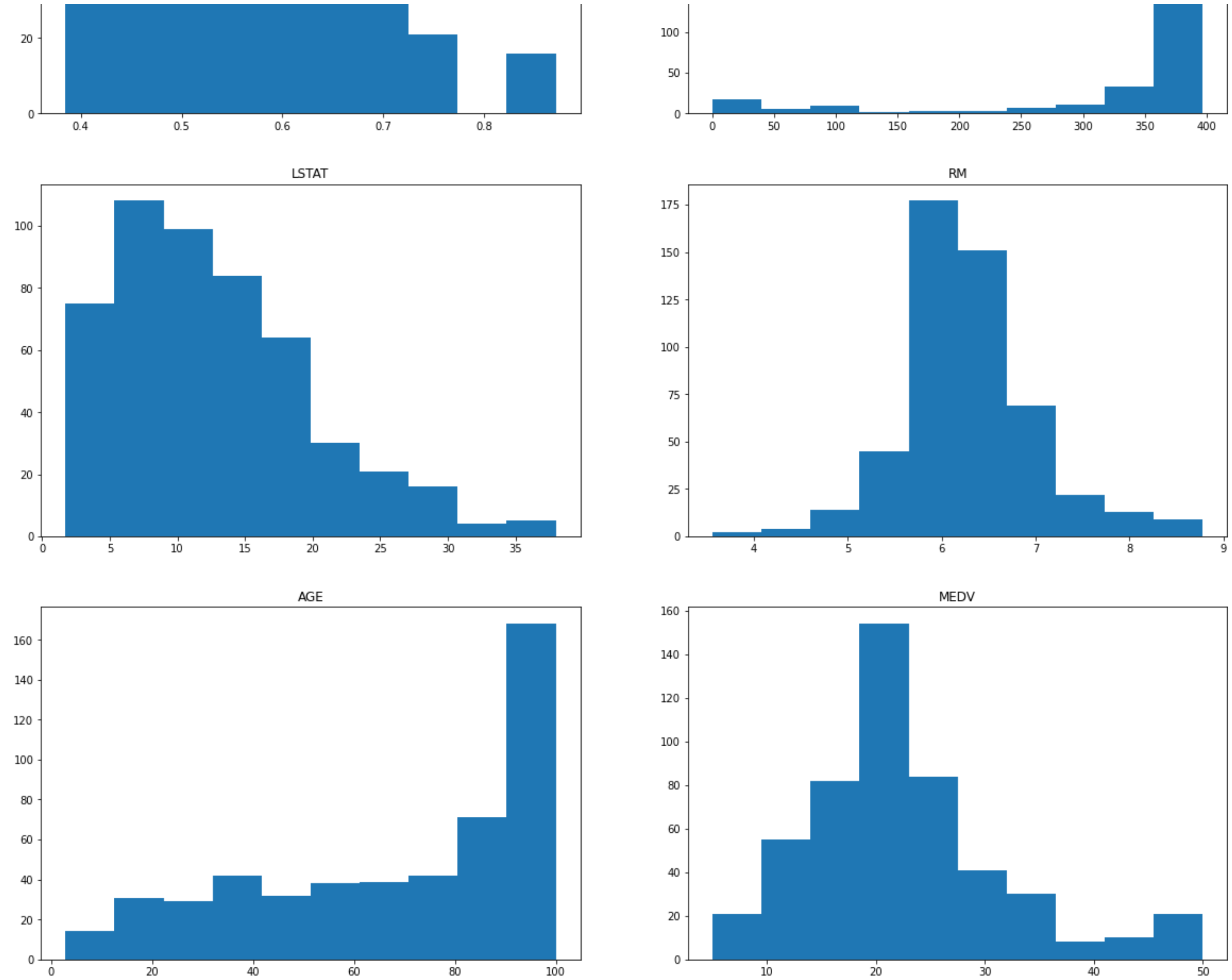
Histogram for all columns 1 to 14

Observations

Similar in shape: LSMT, RM, PTRATIO, NOX

```
#Lets explire histogram characteristics and find 1 peak graphs as MEDV
fig, axs = plt.subplots(7,2,figsize=(20,50))
for i in range(len(data.iloc[0])):
  axs[i%7,i%2].set_title(data.columns[i])
  axs[i%7,i%2].hist(data.iloc[:,i])
```


CRIM
DIS
RAD
ZN
INDUS
TAX
PTRATIO
CHAS
NOX
B



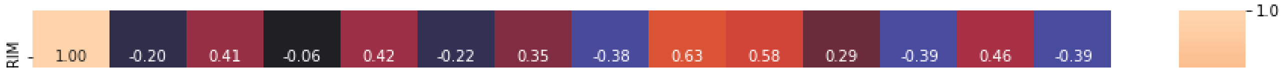
4 Calculate the pearson correlation matrix of the data

```
# use covariance to check relation between each feature with other with pearso
corrs=data.corr()
print('Numeric values of correlation matrix\n',corrs)
print()
plt.figure(figsize=(16,16))
sns.heatmap(corrs,annot=True,fmt='0.2f',center=0)
```

```
Numeric values of correlation matrix
             CRIM        ZN     INDUS  ...         B     LSTAT      MEDV
CRIM     1.000000 -0.200469  0.406583  ... -0.385064  0.455621 -0.388305
ZN      -0.200469  1.000000 -0.533828  ...  0.175520 -0.412995  0.360445
INDUS    0.406583 -0.533828  1.000000  ... -0.356977  0.603800 -0.483725
CHAS    -0.055892 -0.042697  0.062938  ...  0.048788 -0.053929  0.175260
NOX      0.420972 -0.516604  0.763651  ... -0.380051  0.590879 -0.427321
RM      -0.219247  0.311991 -0.391676  ...  0.128069 -0.613808  0.695360
AGE      0.352734 -0.569537  0.644779  ... -0.273534  0.602339 -0.376955
DIS     -0.379670  0.664408 -0.708027  ...  0.291512 -0.496996  0.249929
RAD      0.625505 -0.311948  0.595129  ... -0.444413  0.488676 -0.381626
TAX      0.582764 -0.314563  0.720760  ... -0.441808  0.543993 -0.468536
PTRATIO  0.289946 -0.391679  0.383248  ... -0.177383  0.374044 -0.507787
B       -0.385064  0.175520 -0.356977  ...  1.000000 -0.366087  0.333461
LSTAT    0.455621 -0.412995  0.603800  ... -0.366087  1.000000 -0.737663
MEDV    -0.388305  0.360445 -0.483725  ...  0.333461 -0.737663  1.000000

[14 rows x 14 columns]

<matplotlib.axes._subplots.AxesSubplot at 0x7fc9335f77d0>
```



5 Select features which are more related to the target variable using graphs and the correlation matrix, are the both indicating the same features? (MEDV is the target variable)

According to scatter plots RM, DIS, AGE, LSTAT are the predicted features,

Based on histogram LSMT, RM, PTRATIO, NOX are the most relavent features but

according to pearson correlation matrix LSTAT, RM, INDUS, TAX are most relavent features.

So they are not exactly same.


```
from pandas.core.frame import DataFrame
#Selection of features which has greater then 0.5 correlation with Lable
thr_high=0.5
Fechs_No=len(corrs)-1
data_3=pd.DataFrame()
for i in range(Fechs_No):
  if(abs(corrs.iloc[i,Fechs_No])>thr_high):
    print(corrs.iloc[i,Fechs_No],corrs.columns[i],i)
    data_3[corrs.columns[i]]=data.iloc[:,i:i+1]
data_3.head()
```

```
0.6953599470715401 RM 5
-0.5077866855375623 PTRATIO 10
-0.7376627261740145 LSTAT 12
```

	RM	PTRATIO	LSTAT
0	6.575	15.3	4.98
1	6.421	17.8	9.14
2	7.185	17.8	4.03
3	6.998	18.7	2.94
4	7.147	18.7	5.33

6 Compare correlation of feature between themselves and if two features are highly correlated remove one of them (the one with lesser correlation with the target variable)

```
#comparation of corr values among features with threshold of 75% (0.75)
thr_sim=0.75
col_no_del=[]
Fechs_No=len(corrs)-1
data_11=data.copy(deep=True);
for i in range(Fechs_No):
  for j in range(Fechs_No-i):
    if(corrs.iloc[i+j,j]>thr_sim and corrs.iloc[i+j,j]!=1):
      print(corrs.iloc[i+j,j],corrs.columns[i+j],i+j,corrs.iloc[i+j,Fechs_No],
      if(abs(corrs.iloc[i+j,Fechs_No]) < abs(corrs.iloc[j,Fechs_No])):
        col_no_del.append(corrs.columns[i+j])
      else:
        col_no_del.append(corrs.columns[j])

for i in col_no_del:
  print('Removing column :',i)
  data_11.drop(data_11[[i]], axis = 1, inplace = True)
  print(data_11.columns[:],'\n')
```

```
0.9102281885331865 TAX 9 -0.4685359335677667 RAD 8 -0.38162623063977735

0.7636514469209139 NOX 4 -0.42732077237328203 INDUS 2 -0.48372516002837274

Removing column : RAD
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')

Removing column : NOX
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'RM', 'AGE', 'DIS', 'TAX', 'PTRATIO',
       'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

7 perform the following steps 3 times, 1 with the entire dataset, 1 with the selected features from step 5 and 1 with the selected features from step 6

NOTE: FEATURE SELECTION IS USUALLY REQUIRED ONLY WHEN THERE ARE 100s TO 1000s OF FEATURES OR MORE, BUT HERE FOR EDUCATIONAL PURPOSES WE ARE PRACTICING FEATURE SELECTION ON A SMALLER DATASET

Full Data set: data

Data with only selected lables with 0.5 coorrelation threshold: data_3

Data with removed features with > 0.75 correlation with Lables: data_11

8 Split into train and test

```
# Selecting the Xdata,X3data,X11data and Y data
Fechers_No_data=len(data.iloc[0])-1
Fechers_No_data3=len(data_3.iloc[0])
Fechers_No_data11=len(data_11.iloc[0])-1
Xdata = data.iloc[:,:Fechers_No_data]  #selecting all columns except "MEDV"
Xdata_3 = data_3.iloc[:,:Fechers_No_data3]  #selecting all columns except "MEDV
Xdata_11 = data_11.iloc[:,:Fechers_No_data11]  #selecting all columns except "M
ydata = data.iloc[:,Fechers_No_data:Fechers_No_data+1] #selecting target(price
print(ydata.columns[:])

#Split into train data and test data
X_train, X_test, y_train, y_test = train_test_split(Xdata, ydata, test_size = (
X3_train, X3_test, y3_train, y3_test = train_test_split(Xdata_3, ydata, test_s
X11_train, X11_test, y11_train, y11_test = train_test_split(Xdata_11, ydata, te
```

```
Index(['MEDV'], dtype='object')
```

9 Normalize the data

```
# min max scaling the variables
scaler =  MinMaxScaler()
scaler.fit(Xdata)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

scaler3 =  MinMaxScaler()
scaler3.fit(Xdata_3)
X3_train_scaled = scaler3.transform(X3_train)
X3_test_scaled = scaler3.transform(X3_test)

scaler11 =  MinMaxScaler()
scaler11.fit(Xdata_11)
X11_train_scaled = scaler11.transform(X11_train)
X11_test_scaled = scaler11.transform(X11_test)
```

10 Train the model and perform hyper parameter tuning using cross validation

```
# training of linear regression model
regressor = LinearRegression()
regressor.fit(X_train_scaled,y_train)

regressor3 = LinearRegression()
regressor3.fit(X3_train_scaled,y3_train)

regressor11 = LinearRegression()
regressor11.fit(X11_train_scaled,y11_train)
```

```
LinearRegression()
```

11 Test the model on test set

```
# making predictions for the Xdata, data test set
y_test_pred = regressor.predict(X_test_scaled)
MSE=mean_squared_error(y_test, y_test_pred)
COD=r2_score(y_test, y_test_pred)
print('Mean squared error for testing set: %.2f'%MSE)
print('Coefficient of determination for testing set: %.2f'%COD)
```

```
Mean squared error for testing set: 25.53
Coefficient of determination for testing set: 0.68
```

```
# making predictions for the X3data, data_3 test set
y3_test_pred = regressor3.predict(X3_test_scaled)
MSE3=mean_squared_error(y3_test, y3_test_pred)
COD3=r2_score(y3_test, y3_test_pred)
```

```
print('Mean squared error for testing set: %.2f'%MSE3)
```

```
Mean squared error for testing set: 28.86
Coefficient of determination for testing set: 0.64
```

```
# making predictions for the X11data, data_11 test set
y11_test_pred = regressor11.predict(X11_test_scaled)
MSE11=mean_squared_error(y11_test, y11_test_pred)
COD11=r2_score(y11_test, y11_test_pred)
print('Mean squared error for testing set: %.2f'%MSE11)
print('Coefficient of determination for testing set: %.2f'%COD11)
```

```
Mean squared error for testing set: 26.81
Coefficient of determination for testing set: 0.67
```

```
#Performance comparation for all 3 models
print('##########################################################################
print('No of features','\t','13 features','\t\t','11 features','\t\t','3 featu
print('MSE values','\t',MSE,'\t',MSE3,'\t',MSE11)
print('COD values ','\t',COD,'\t',COD3,'\t',COD11)
print('##########################################################################
```

```
##############################################################################
No of features   13 features           11 features           3 features
MSE values       25.525190907490796    28.85919204788034     26.8053681477
COD values       0.6845664376687204    0.6433657328300743    0.66874634571
##############################################################################
```

Conclusion and Learning:

Number of features does not gaurrantee for the best performance as here 3 features are also working as similar as 13

✓ 0s completed at 10:17 PM