ASSIGNMENT 3

AIDI 1003 – CAPSTONE TERM 1

AMIT MARAJ

AHNCH BALA 100424062

SONAKSHI KARKERA 100720763

SURBHI THAKUR 100732335

ARUN KALAESWARAN 100771700

October 20, 2019

DATASET – BOSTON HOUSING PRICE

We will predict the price of houses using four algorithms

1. Decision Tree
2. Random Forest
3. KNN Regression
4. Linear Regression

DECISION TREE ALGORITHM

```
#used http://gearons.org/2016-12-15-boston-housing/ and https://acadgild.com/blog/using-decision-trees-for-regression-problem
#https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/ as a guide
import numpy as np
import pandas as pd
%matplotlib inline #a magic function
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import mean_squared_error

from sklearn.tree import DecisionTreeRegressor
```

```
data = pd.DataFrame(boston.data,columns=boston.feature_names)
data = pd.concat([data,pd.Series(boston.target,name='MEDV')],axis=1)
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	1.0	296.0	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	2.0	242.0	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	2.0	242.0	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	3.0	222.0	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	3.0	222.0	18.7	396.90	5.33	36.2

```
X = data.iloc[:,:-1]
y = data.iloc[:,-1] #selecting predictor and targer variables
```

```
x_training_set, x_test_set, y_training_set, y_test_set = train_test_split(X,y,test_size=0.10,random_state=42,
shuffle=True) #splitting train test data
```

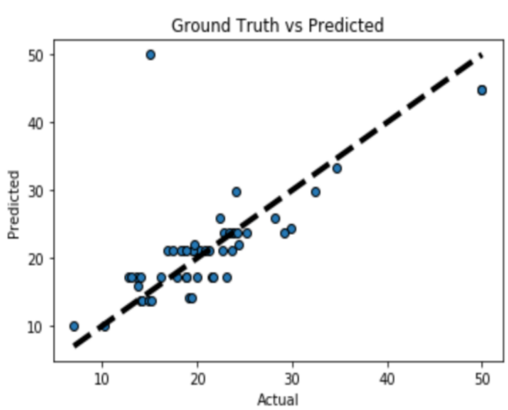
```
#use decisiontreeregressor to fit model
model =  DecisionTreeRegressor(max_depth=5,random_state=0)
model.fit(x_training_set, y_training_set)
```

```
]: DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

```
#metrics to evaluate
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
model_score = model.score(x_training_set,y_training_set)
print("Coefficient of Determination R^2 of the prediction: ",model_score)
y_predicted = model.predict(x_test_set)

# The mean squared error
print("Mean squared error: %.2f"% mean_squared_error(y_test_set, y_predicted))
print("Mean absolute error: %.2f"% mean_absolute_error(y_test_set, y_predicted))
# Explained variance score: 1 is perfect prediction
print('Test Variance score: %.2f' % r2_score(y_test_set, y_predicted))
```

```
Coefficient of Determination R^2 of the prediction:  0.9174967918577124
Mean squared error: 33.19
Mean absolute error: 3.15
Test Variance score: 0.47
```

```
#run model against test data
from sklearn.model_selection import cross_val_predict

fig, ax = plt.subplots()
ax.scatter(y_test_set, y_predicted, edgecolors=(0, 0, 0))
ax.plot([y_test_set.min(), y_test_set.max()], [y_test_set.min(), y_test_set.max()], 'k--', lw=4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Ground Truth vs Predicted")
plt.show()
```



In order to evaluate the model I used 3 metrics:

R^2 score: Explains how well the selected independent variable explain the variability in the selected dependent variable. The higher the R^2, the better our model fits the data. R^2 is between 0 and 1 (or 0% to 100%). In this case, the R^2 we achieved is 0.92 rounded. This shows that the independent variable does explain the variability of the dependent variables quite well.

Mean Squared Error (MSE): The MSE tells us how close the regression line is to our data. The smaller the value of the MSE, the better it is. In our case we got 33.19. This means that our data isn't the best match to our regression line. You can even tell by looking at the scatter plot below there are points that are away from the line. There is even an outlier that is very far from the line.

Mean Absolute Error (MAE): Basically, the MAE tells us how inaccurate our predictions were as well, like MSE. Meaning, if we had chosen a random point from our data, it means that our prediction would be 3.15 away from the true value. Since it really depends on our scale, its hard to conclude on whether this score is "good" or "bad".

Test Variance Score: Tells us how much different the actual value is from the average of the predicted value. The lower the score the better, but the value of "low" is determined by the R^2 score. A score of 0.47 is quite low for variance.

RANDOM FOREST ALGORITHM

In [87]:
```
import numpy
from numpy import arange
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.metrics import mean_squared_error

from sklearn.ensemble import RandomForestRegressor
```

In [71]:
```
filename='housing.csv'
dataset = pd.read_csv(filename, delim_whitespace=True , names=names)
```

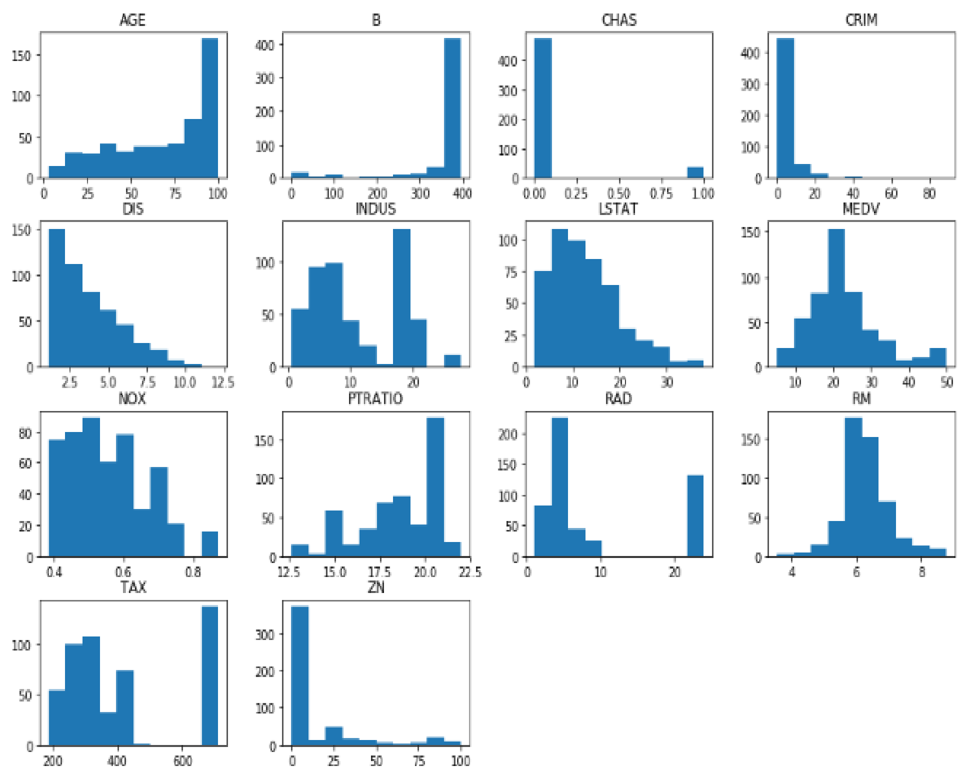
In [72]:
```
# Descriptive statistics
# shape
print(dataset.shape)
dataset.describe()
```

```
(506, 14)
```

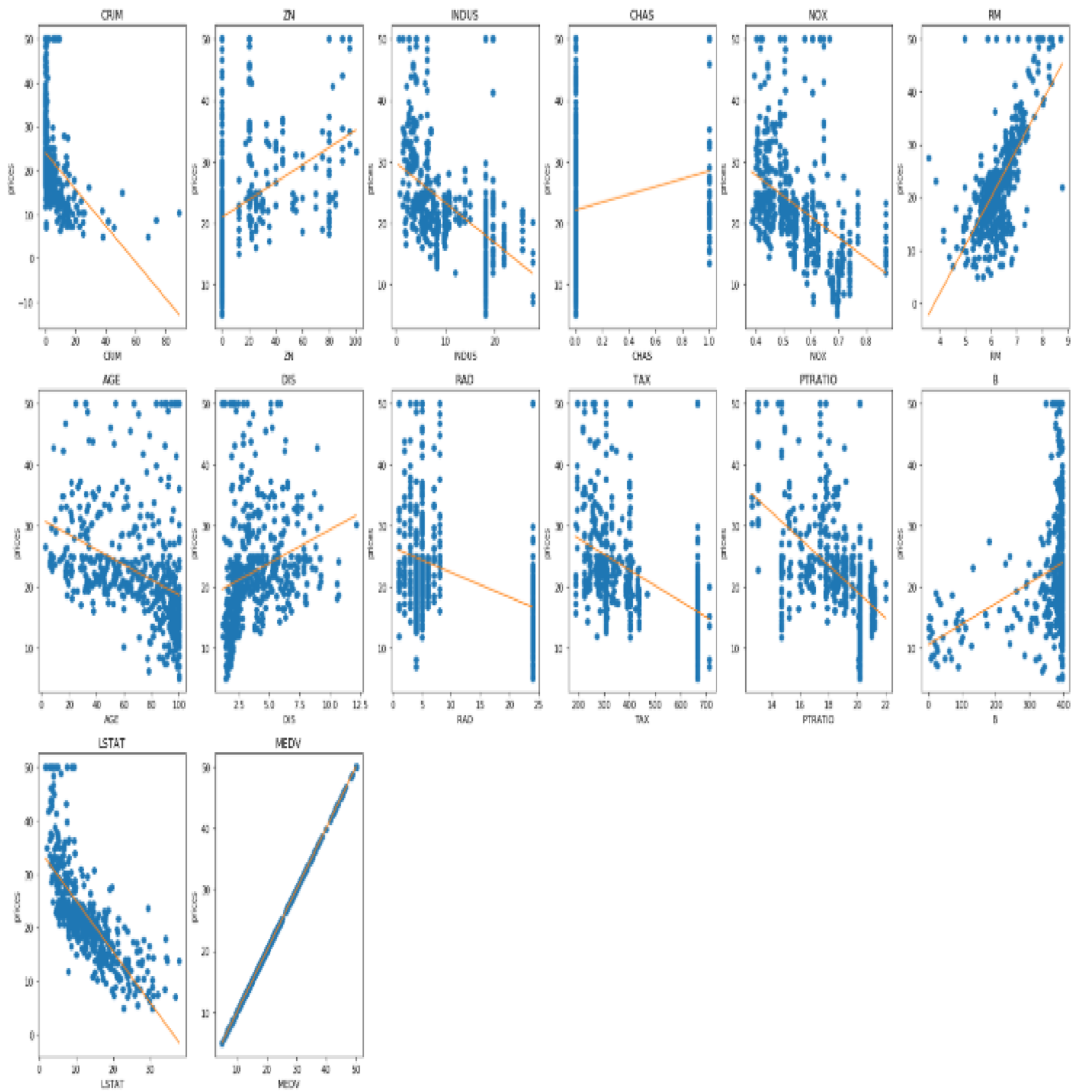
Out[72]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	506.000000	506.000000	506.000000	506.000000	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	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

In [73]:
```
# histograms
dataset.hist(bins=10,figsize=(15,10),grid=False);
```



In [74]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(25, 18))
# i: index
for i, col in enumerate(dataset.columns):
    # 3 plots here hence 1, 3
    plt.subplot(3, 6, i+1)
    x = dataset[col]
    y =dataset['MEDV']
    plt.plot(x, y, 'o')
    # Create regression line
    plt.plot(numpy.unique(x), numpy.poly1d(numpy.polyfit(x, y, 1))(numpy.unique(x)))
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('prices')
```

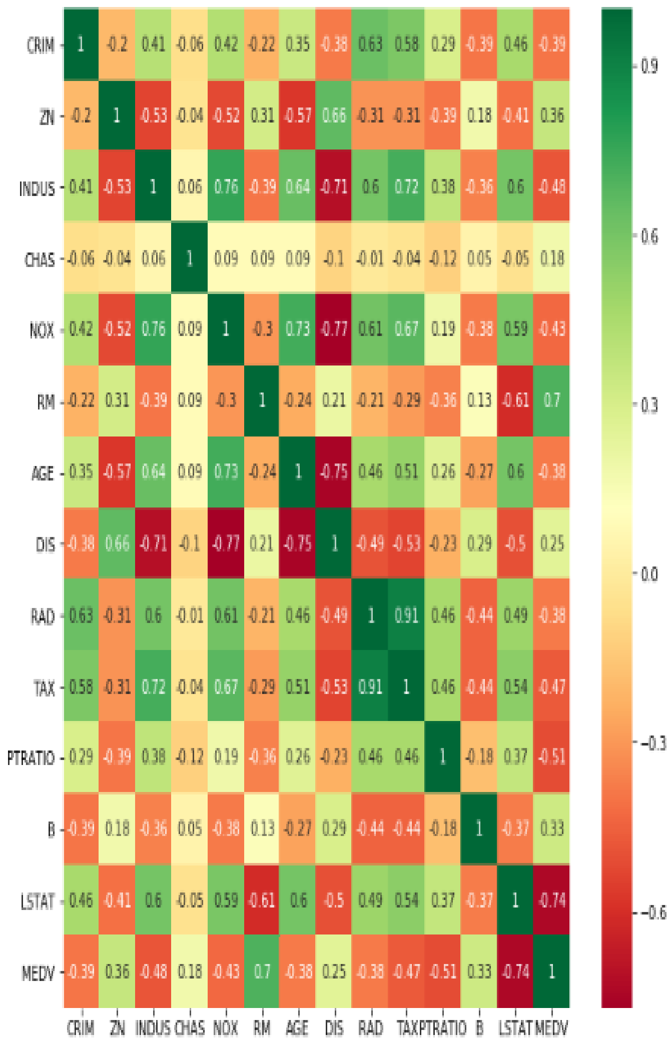


From my analysis, Price increases with RM and Price decreases with increase in PTRATO and LSTAT

In [75]:

```python
#Heat Map
fig, ax = plt.subplots(figsize=(10,10))
correlation_matrix = dataset.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True,cmap="RdYlGn")
```

Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x1de32a0cda0>

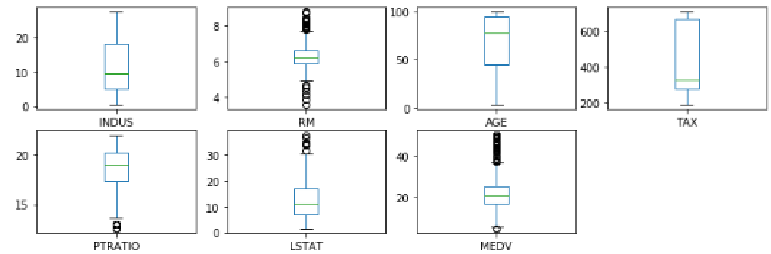


From Correlation data, We can confim that variable LSTAT, RM, AGE, and PTRATIO have good corelation with our output variable MEDV

In [76]:

```python
dataset = dataset.drop(['CRIM','ZN','CHAS','NOX','DIS','RAD','B'], axis = 1)
```

In [77]:
```
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(4,4), sharex=False, sharey=False, figsize=(12,8) )
pyplot.show()
```



From the above fig, we see that LSTAT,RM,PTRATIO,MEDV,B ,CHAS,DIS,CRIM,ZN has outliers

In [55]:
```
dataset.isnull().sum()
#None of the features have null values
```

Out[55]:
```
INDUS      0
RM         0
AGE        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [63]:
```
X=dataset.iloc[:, :-1]
X.head()
y=dataset.iloc[:,-1]
```

Out[63]:
```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

In [64]:
```
# Scale the data to be between -1 and 1
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

In [65]:
```
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)
```

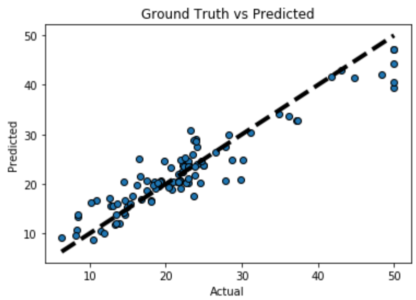
In [79]:
```
n_estimators=100
# Fit regression model
# Estimate the score on the entire dataset, with no missing values
model = RandomForestRegressor(random_state=0, n_estimators=n_estimators)
model.fit(X_train, y_train)
```

Out[79]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [91]:
```
from sklearn.metrics import mean_squared_error, r2_score
model_score = model.score(X_train,y_train)
print("coefficient of determination R^2 of the prediction.: ",model_score)
y_predicted = model.predict(X_test)

# The mean squared error
print("Mean squared error: %.2f"% mean_squared_error(y_test, y_predicted))
print("Root Mean squared error: %.2f"% np.sqrt(mean_squared_error(y_test, y_predicted)))
print('Test Variance score: %.2f' % r2_score(y_test, y_predicted))
```

```
coefficient of determination R^2 of the prediction.:  0.9748759354045127
Mean squared error: 11.86
Root Mean squared error: 3.44
Test Variance score: 0.88
```

In [86]:
```
fig, ax = plt.subplots()

ax.scatter(y_test, y_predicted, edgecolors=(0, 0, 0))

ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)

ax.set_xlabel('Actual')

ax.set_ylabel('Predicted')

ax.set_title("Ground Truth vs Predicted")

plt.show()
```



For this dataset, we will be calculating the coefficient of determination,
<u>R^2:</u>
It is used to quantify your model's performance. The coefficient of determination for a model is a useful metric in regression analysis, as it often describes how "good" is the model at making predictions. The values for R2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the **dependent variable**. A model with an R2 of 0 is no better than a model that always predicts the *mean* of the target variable, whereas a model with an R2 of 1 perfectly predicts the target variable.
The R^2 coefficient is 0.97, meaning our model can explain 97% of the total variation of the data around its mean.

<u>Mean Squared Error</u>: Mean Squared Error (MSE) is a measure of how close a fitted line is to data points. The smaller the Mean Squared Error, the closer the fit is to the data.

<u>Root Mean Squared Error</u>: measures the average magnitude of the error. It ranges from 0 to ∞. In this case we get RMSE as 3.44 which means the model is able to predict the values fairly.

<u>Conclusion:</u>
We can see that our R2 score and MSE are good. This means that we have found a good fitting model to predict the median price value of a house. This can be further improvement to the metric by doing some preprocessing and removing outliers before fitting the data.

KNN REGRESSION ALGORITHM

In [ ]:
```
#IMPLEMENTING KNN REGRESSION ALGORITHM FOR BOSTON HOUSING DATASET
#K Nearest Neighbour is the most common algorithm used for classification problems however it can alse be used for Reression
#Usedhttps://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html
#https://www.ritchieng.com/machine-learning-project-boston-home-prices/
#https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
```

In [102]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import KNeighborsRegressor
```

In [66]:
```
b_dataset = pd.read_csv('housing.data.txt', delim_whitespace=True, names=('CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
```

In [67]:
```
b_dataset.head(10)
```

Out[67]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	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	242.0	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	242.0	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	222.0	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	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311.0	15.2	386.71	17.10	18.9

In [98]:
```
b_dataset.shape
```

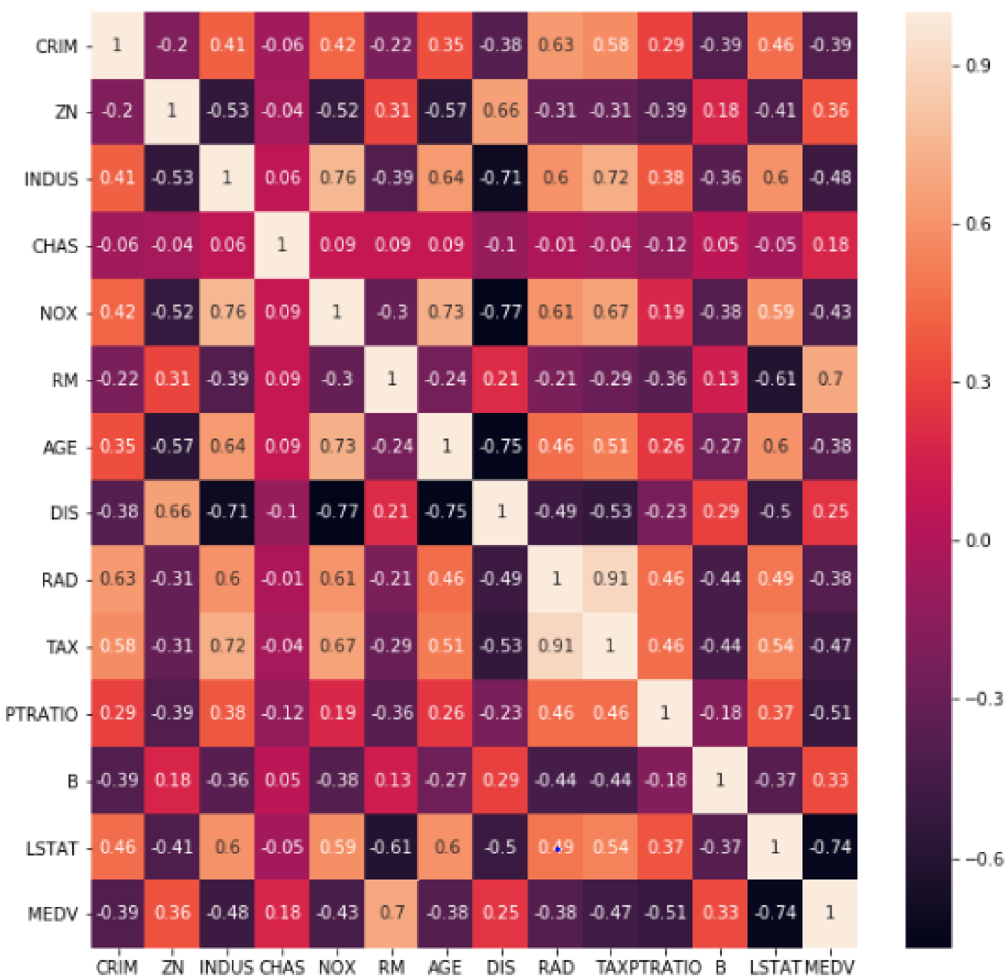
Out[98]: (506, 14)

```
In [69]: b_dataset.isnull().sum() #checking for null values for the features
```

```
Out[69]: 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
```

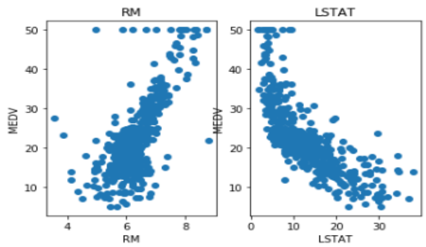
```
In [105]: fig, ax = plt.subplots(figsize=(10,10))    # Plotting the correlation matrix
          correlation_matrix = b_dataset.corr().round(2)
          sns.heatmap(data=correlation_matrix, annot=True)
```

```
Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x2b8ce7e1748>
```



In [95]:

```
# From correlation matrix:- RM has strong possitive corelation with the output MEDV
# LSTAT has negative strong corelation with the output MEDV
features = ['RM', 'LSTAT']
target = b_dataset['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = b_dataset[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



In [80]:

```
X=b_dataset[['LSTAT','RM']] #Selecting features
Y=b_dataset['MEDV'] # Target
```

In [81]:

```
from sklearn.model_selection import train_test_split #splitting train test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
```

In [82]:

```
#Using KNN Regressor Algorithm
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor(n_neighbors=1)
knr.fit(X_train, Y_train)
```

Out[82]:

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                    weights='uniform')
```

In [83]:

```
Y_pred = knr.predict(X_test)
```

In [85]:

```
#printing both Actual and Predicted MEDV
print("KNN Regresson Model")
knn = pd.DataFrame(X_test)
knn['MEDV '] = Y_test
knn['Predicted MEDV '] = Y_pred
print(model_knn.head(10))
```

```
KNN Regresson Model
     LSTAT     RM  MEDV  Predicted MEDV
329   7.34  6.333  22.6             24.7
371   9.53  6.216  50.0             25.0
219  10.50  6.373  23.0             21.2
403  19.77  5.349   8.3             16.3
78   12.34  6.232  21.2             20.1
15    8.47  5.834  19.9             23.4
487  11.45  5.905  20.6             20.0
340   9.29  5.968  18.7             19.6
310  12.64  4.973  16.1             15.3
102  10.63  6.405  18.6             21.2
```

In [59]:

```
# finding Mean Squared Error (MSE)
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_test, Y_pred)
print(mse)
```

```
34.71254901960784
```

In [107]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(Y_test, Y_pred)
print(mae)
```

```
3.729411764705882
```

In [86]:
```
#finding rmse error
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(Y_test, Y_pred))
print(rmse)
```

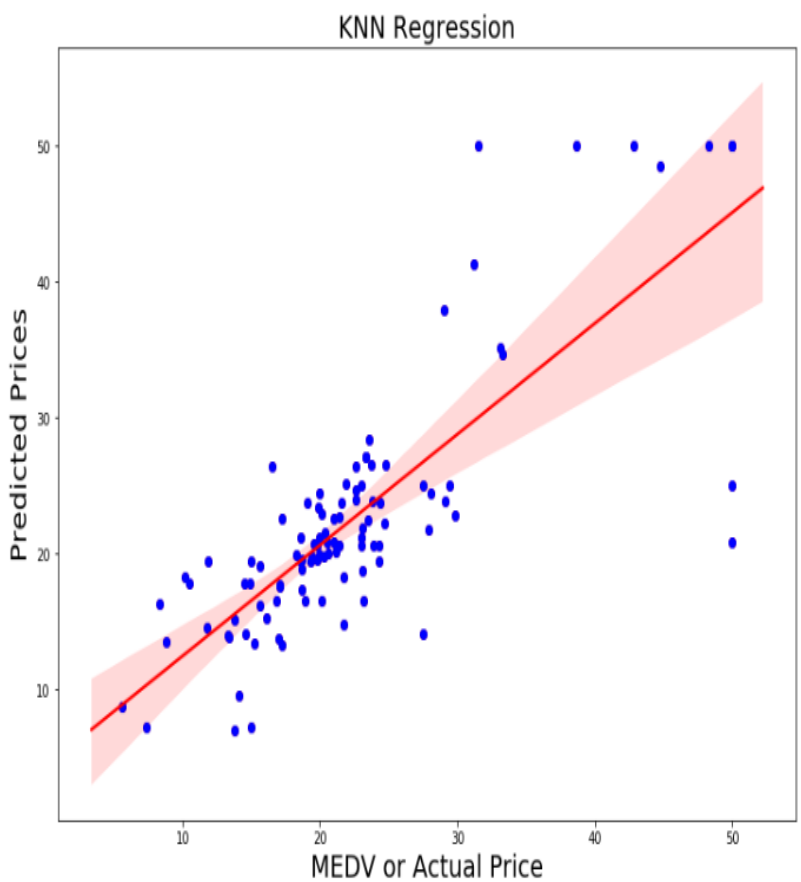
```
5.891735654254003
```

In [88]:
```
#finding r square error
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_pred)
print(r2)
```

```
0.5737048453892075
```

In [103]:
```
fig = plt.figure(figsize=(12,9))
ax = sns.regplot(Y_test, Y_pred, marker = 'o', color = 'red')
plt.scatter(Y_test, Y_pred, color='blue')
ax.set_title('KNN Regression', fontsize=20)
ax.set_xlabel('MEDV or Actual Price', fontsize=20)
ax.set_ylabel('Predicted Prices', fontsize=20)
```

Out[103]:
```
Text(0, 0.5, 'Predicted Prices')
```



RMSE (Root mean squared error) is the standard deviation error of the predicted values. It shows how concentrated predicted values are near the best line fit. The smaller the RMSE the better which means smaller distance between the residual and the actual regression line. In this case we get RMSE as 5.87 which means the model is not able to predict the values efficiently.

R^2 Score range from 0 to 1 or 0% to 100%. The more the R^2 score, the better the model fits data. It is also known as coefficient of determination. For KNN Regressor algorithm on Boston Dataset the R^2 SCORE is 0.57. If the value of r2 is between 0.5 and 0.7 it depicts the moderate behavior of model.

MSE Score is the average square difference between the estimated and the actual values. The higher the MSE, the worse performance of model. In this case the MSE is 34.71 which means our data doesn't fit perfectly with the regression line as there are some points away from the regression line which is shown in the graph.

MAE Mean Absolute) Error is a linear score which means all the individual differences between the target and predicted values are weighted equally. The MAE is 3.7 which means the average difference between the predicted and actual prices is 3.7

LINEAR REGRESSION MODEL

In [1]:
```
#linear regression model on boston housing dataset
#importing all the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [3]:
```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

In [4]:
```
print(boston_dataset.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [6]:
```
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
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	4.98
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	9.14
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	4.03
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	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

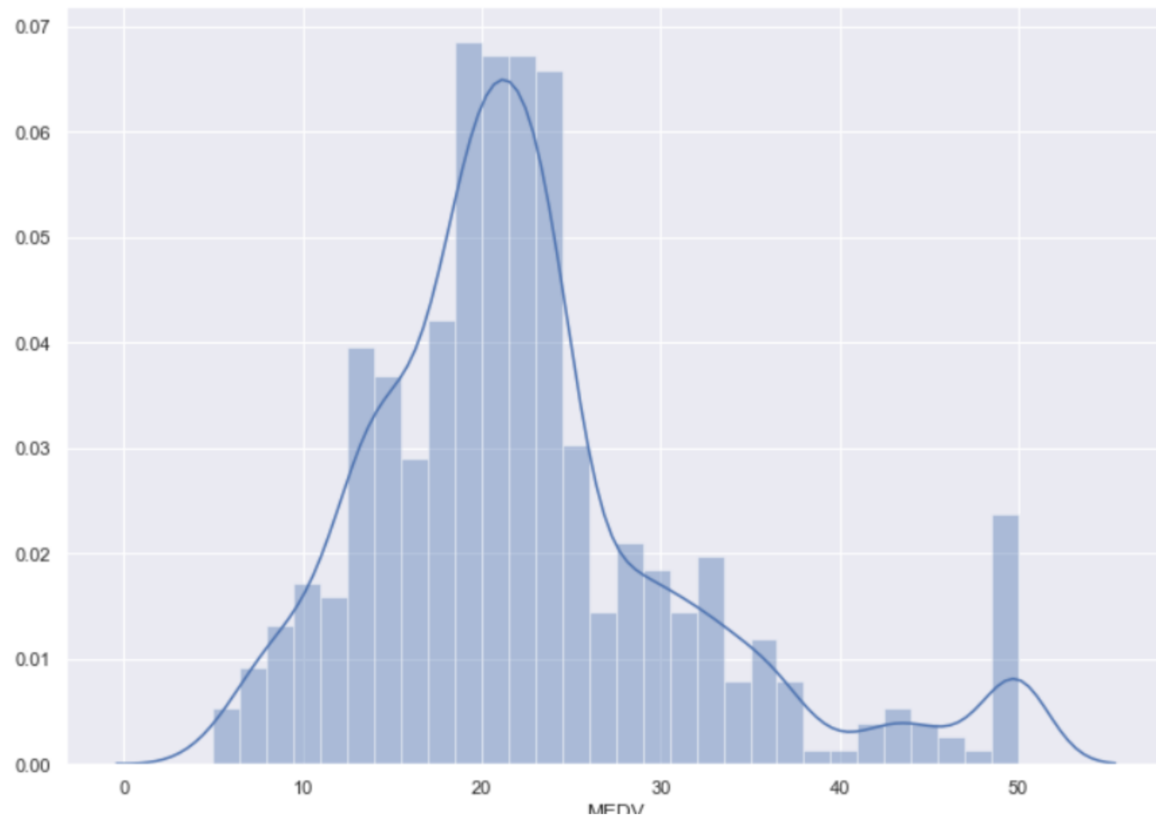
In [7]:
```
boston['MEDV'] = boston_dataset.target
```


```
boston['MEDV'] = boston_dataset.target
```


```
boston.isnull().sum()
#checking for missing values in the features, there are none
```

```
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
```

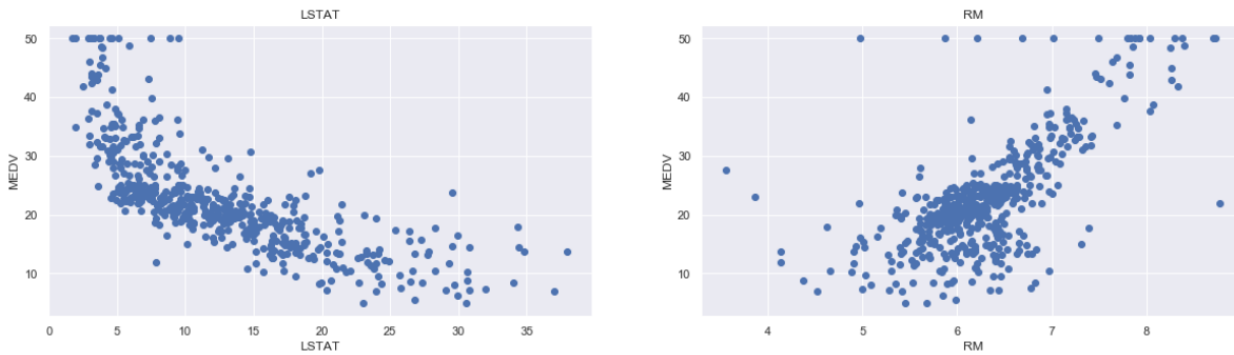
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston['MEDV'], bins=30)
plt.show()
#MEDV is distrubuted normally with some outliers
```




```
correlation_matrix = boston.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
# we see that RM has a strong positive correlation
# we see that LSTAT has a strong negative correlation
# we will use RM and LSTAT as our features with MEDV
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24e1a8fc828>
```

```
plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']
target = boston['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = boston[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
#as LSTAT increase (% of lower status of the population), median value home decreases
#as RM increases (rooms), median value home increases
```




```
X = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT','RM'])
Y = boston['MEDV']
#prepping the data for the training model
```

```
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=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
#splitting the data into training and testing sets
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
#running linear regression model
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```python
# model evaluation for training set
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))


print("The model performance for training set")
print('RMSE is {}'.format(rmse))
print("\n")

# model evaluation for testing set in RMSE
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))


print("The model performance for testing set")
print('RMSE is {}'.format(rmse))
```

```
The model performance for training set
RMSE is 5.6371293350711955


The model performance for testing set
RMSE is 5.137400784702911
```

```python
#model evaluation for r2 score
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, y_test_predict)
print(r2)
```

```
0.6628996975186953
```

```python
#model evaluation for mean absolute error
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(Y_test, y_test_predict)
print (mae)
```

```
3.7913102133431047
```

Root mean squared error (RMSE) is the standard deviation error of the predicted values. It shows how concentrated predicted values are near the best fit line. The smaller the RMSE the smaller distance between the residual and the actual regression line. The larger the RMSE the larger the distance between residual and actual regression line. In this case we get RMSE as 5.13 on our testing set, which means the model is not able to predict the values efficiently.

R^2 Score range from 0 to 1 or 0% to 100%. The higher the R^2 score, the better the model fits data. It is also known as coefficient of determination. For linear regression algorithm on Boston Dataset the R^2 SCORE is 0.66. If the value of r2 is between 0.5 and 0.7 it depicts the moderate behaviour of model.

Mean Absolute Error (MAE) is a linear score which means all the individual differences between the target and predicted values are weighted equally. The MAE is 3.8 which means the average difference between the predicted and actual prices is 3.8 (in thousands).