**Assignment 20**

**Problem Statement 1:**

Build the linear regression model using scikit learn in boston data to predict 'Price' based on other dependent variable.

Here is the code to load the data

import numpy as np

import pandas as pd

import scipy.stats as stats

import matplotlib.pyplot as plt

import sklearn

from sklearn.datasets import load\_boston

boston = load\_boston()

bos = pd.DataFrame(boston.data)

Answer :

# Get the data and convert the boston data into dataframe and add column names

import numpy as np

import pandas as pd

import scipy.stats as stats

import matplotlib.pyplot as plt

import sklearn

from sklearn.datasets import load\_boston

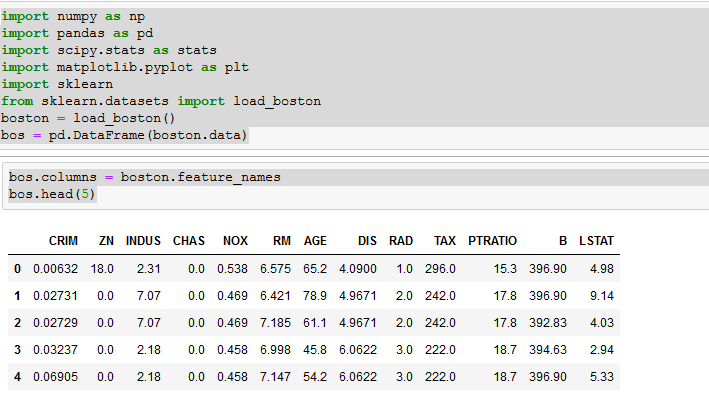
boston = load\_boston()

bos = pd.DataFrame(boston.data)

**## Add column names**

bos.columns = boston.feature\_names

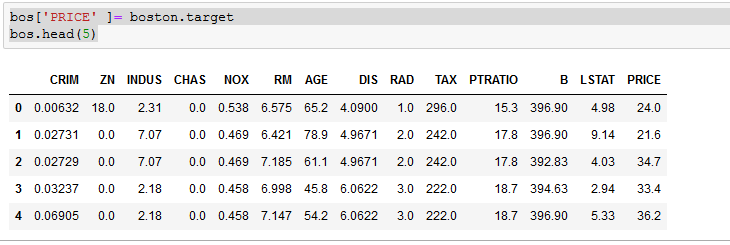
bos.head(5)



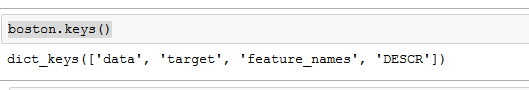
# Get Target Data – Price

bos['PRICE' ]= boston.target

bos.head(5)

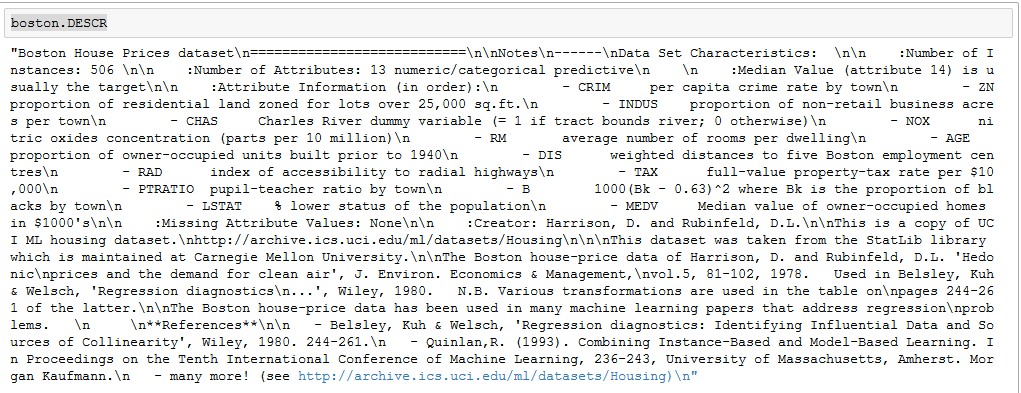


boston.keys()



boston.DESCR

##Boston House Prices dataset



# Predicting Housing Prices with Linear Regression

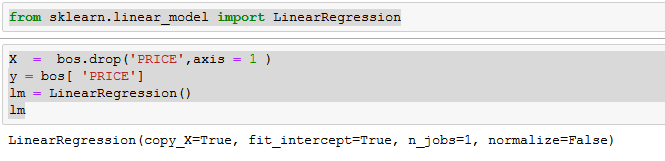
from sklearn.linear\_model import LinearRegression

X = bos.drop('PRICE',axis = 1 )

y = bos[ 'PRICE']

lm = LinearRegression()

lm



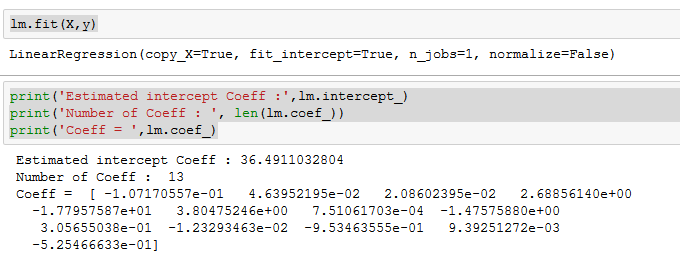
# Fitting the Model

lm.fit(X,y)

print('Estimated intercept Coeff :',lm.intercept\_)

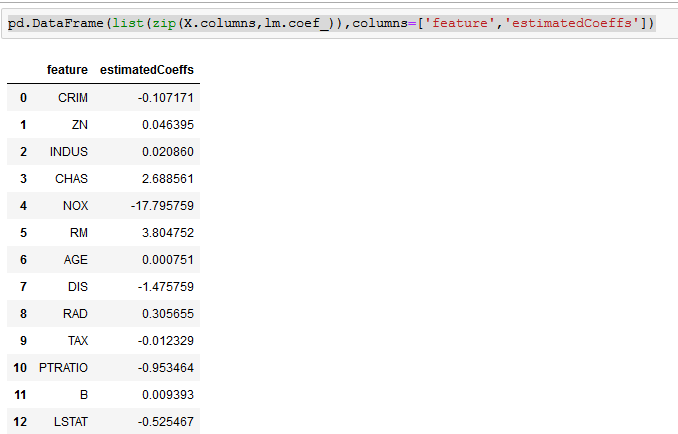
print('Number of Coeff : ', len(lm.coef\_))

print('Coeff = ',lm.coef\_)



**Create a dataframe with the coeffs**

pd.DataFrame(list(zip(X.columns,lm.coef\_)),columns=['feature','estimatedCoeffs'])



**Generate a plot of Price versus Rooms (Avg # of Rooms per dwelling)**

import matplotlib.pyplot as plt

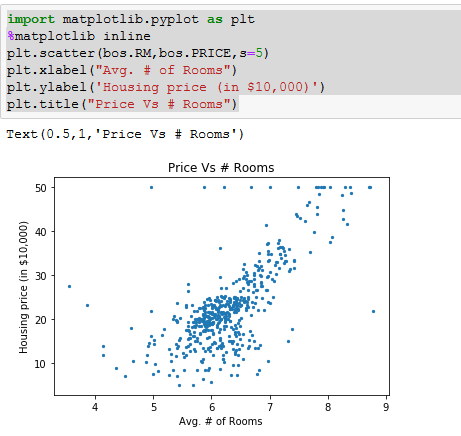
%matplotlib inline

plt.scatter(bos.RM,bos.PRICE,s=5)

plt.xlabel("Avg. # of Rooms")

plt.ylabel('Housing price (in $10,000)')

plt.title("Price Vs # Rooms")



**Predicting Prices**

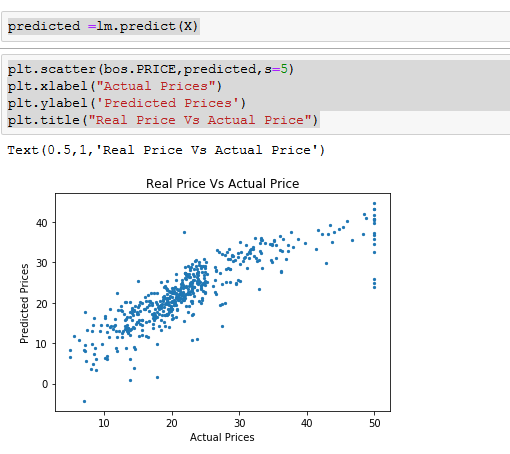
predicted =lm.predict(X)

plt.scatter(bos.PRICE,predicted,s=5)

plt.xlabel("Actual Prices")

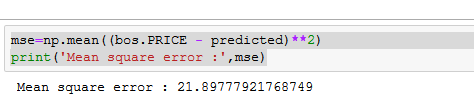
plt.ylabel('Predicted Prices')

plt.title("Real Price Vs Actual Price")



mse=np.mean((bos.PRICE - predicted)\*\*2)

print('Mean square error :',mse)

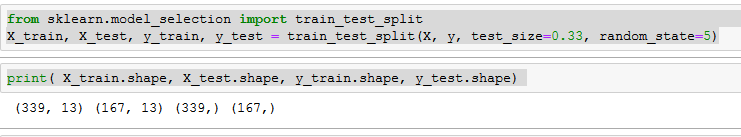


# Training and Validating

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=5)

print( X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)



lm=LinearRegression()

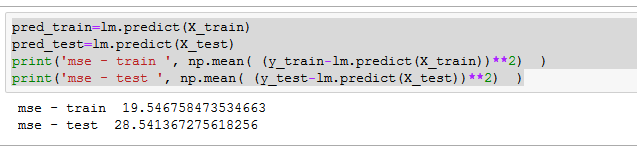
lm.fit(X\_train,y\_train)

pred\_train=lm.predict(X\_train)

pred\_test=lm.predict(X\_test)

print('mse - train ', np.mean( (y\_train-lm.predict(X\_train))\*\*2) )

print('mse - test ', np.mean( (y\_test-lm.predict(X\_test))\*\*2) )



# Plotting the Residuals

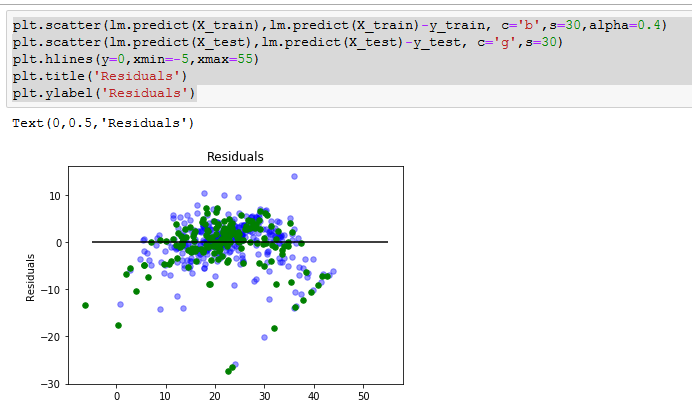
plt.scatter(lm.predict(X\_train),lm.predict(X\_train)-y\_train, c='b',s=30,alpha=0.4)

plt.scatter(lm.predict(X\_test),lm.predict(X\_test)-y\_test, c='g',s=30)

plt.hlines(y=0,xmin=-5,xmax=55)

plt.title('Residuals')

plt.ylabel('Residuals')



from sklearn.metrics import r2\_score

r2\_score(y\_test,pred\_test)

