#

import numpy as np

import pandas as pd

import sklearn

import seaborn as sns

from sklearn.preprocessing as StandardScaler

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

import matplotlib.pyplot as plt

%matplotlib inline

#

from IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast\_node\_interactivity="all"

#

from sklearn import datasets

boston=datasets.load\_boston()

boston.feature\_names

#

print(boston.DESCR)

#

import pandas as pd

boston\_df=pd.DataFrame(boston.data,columns=boston.feature\_names)

boston\_df.head(5)

#

boston\_df['House\_Price']=boston.target

boston\_df.head(5)

boston\_df.describe()

#

x=boston\_df.corr()

#

x

#

plt.subplots(figsize=(20,20))

sns.heatmap(x,cmap='RdYlGn',annot=True)

plt.show();

#

x=boston\_df.drop('House\_Price',axis=1)

y=boston\_df['House\_Price']

x.head()

y.head()

#

train\_x,test\_x,train\_y,test\_y=train\_test\_split(x,y,test\_size=0.3,random\_state=1)

train\_x.shape

train\_y.shape

test\_x.shape

test\_y.shape

#

from sklearn.linear\_model import LinearRegression

lm=LinearRegression()

#

lm.fit(train\_x,train\_y)

#

predict\_test\_lm=lm.predict(test\_x)

#

print(lm.coef\_)

#

df\_m=pd.DataFrame({'features':x.columns,'coeff':lm.coef\_})

#

df\_m=df\_m.sort\_values(by=['coeff'])

#

df\_m

#

df\_m.plot(x='features',y='coeff',kind='bar',figsize=(15,10))

plt.show();

#

print("R square value for regression test data is-")

np.round(lm.score(test\_x,test\_y)\*100,2)

#

print("R square value for regression train data is-")

np.round(lm.score(train\_x,train\_y)\*100,2)

#

import numpy as np

from sklearn import metrics

print ("simple linear regression Mean square error (MSE) for TEST data is:")

np.round(metrics.mean\_squared\_error(test\_y,predict\_test\_lm),2)

#

from sklearn.metrics import mean\_absolute\_error

print ("simple linear regression Mean absolte error (MSE) for TEST data is:")

np.round(metrics.mean\_absolute\_error(test\_y,predict\_test\_lm),2)

#

fdf=pd.concat([test\_x,test\_y],1)

#

predict\_test=lm.predict(test\_x)

#

fdf['Predicted']=np.round(predict\_test,1)

#

fdf['Predicted\_Error']=fdf['House\_Price']-fdf['Predicted']

fdf

#

import statsmodels.formula.api as smf

#

model=smf.ols('House\_Price ~ CRIM+ZN+INDUS+CHAS+NOX+RM+AGE+DIS+RAD+TAX+PTRATIO+B+LSTAT',data=boston\_df).fit()

print(model.summary())

#

model=smf.ols('House\_Price ~ CRIM+ZN+CHAS+NOX+RM+DIS+RAD+TAX+PTRATIO+B+LSTAT',data=boston\_df).fit()

print(model.summary())

#

robust\_model=model.get\_robustcov\_results()

print(robust\_model.summary())

#