import numpy as np

import pandas as pd

import os

import statsmodels.formula.api as stats

from statsmodels.formula.api import ols

import sklearn

from sklearn import linear\_model, datasets

from sklearn.metrics import mean\_squared\_error

import plotly.plotly as py

import plotly.graph\_objs as go

from plotly.offline import download\_plotlyjs,init\_notebook\_mode,plot,iplot

init\_notebook\_mode(connected=True)

valuation=pd.read\_csv("Real-estate-valuation-dataset.csv")

print(valuation.head())

print(valuation.describe())

dropped\_valuation=valuation.drop(["No","X5 latitude","X6 longitude"],axis=1)

corr\_matrix\_valuation=dropped\_valuation.corr()

trace\_corr\_valuation=go.Heatmap(z=np.array(corr\_matrix\_valuation),x=corr\_matrix\_valuation.columns,y=corr\_matrix\_valuation.columns)

data\_valuation=[trace\_corr\_valuation]

plot(data\_valuation)

X= dropped\_valuation.drop(['Y house price of unit area'],axis=1)

Y= dropped\_valuation['Y house price of unit area']

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y,test\_size=0.20)

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

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

Y\_pred= lm.predict(X\_test)

result\_lm=pd.DataFrame({

'Actual':Y\_test,

'Predict':Y\_pred

})

result\_lm['Diff']=Y\_test - Y\_pred

result\_lm.head()

print("Estimated Intercept is",lm.intercept\_)

print("Coefficients of the model:",lm.coef\_)

coef=zip(X.columns,lm.coef\_)

coef\_df = pd.DataFrame(list(zip(X.columns,lm.coef\_)),columns=['features','coefficients'])

print(coef\_df)

from sklearn.metrics import mean\_absolute\_error

print('Mean Absolute Error:',mean\_absolute\_error(Y\_test,Y\_pred) )

print('Mean Squared Error:',mean\_squared\_error(Y\_test,Y\_pred) )

print('Root Mean Squared Error:', np.sqrt(mean\_squared\_error(Y\_test,Y\_pred)) )

print('Variance score:%.2f'%lm.score(X\_test,Y\_test))

import seaborn as sns

sns.regplot(x='Actual', y='Predict',data=result\_lm)

print(result\_lm.head())