e09ny6s2w

July 31, 2023

1 Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression model.Create a model that helps him to estimate of what the house would sell for.

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
df=pd.read_csv("/content/11_winequality-red.csv")
df
```

[ ]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
…               …                 …          …               …        …
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
…                     …                   …      …     …         …
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
```

```
1598                 18.0                   42.0  0.99549  3.39       0.66

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
…         …       …
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

2 Data Cleaning and Data Preprocessing

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
```

```
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

[ ]:
```
df.describe()
```

[ ]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
```

```
75%        3.400000      0.730000    11.100000     6.000000
max        4.010000      2.000000    14.900000     8.000000
```

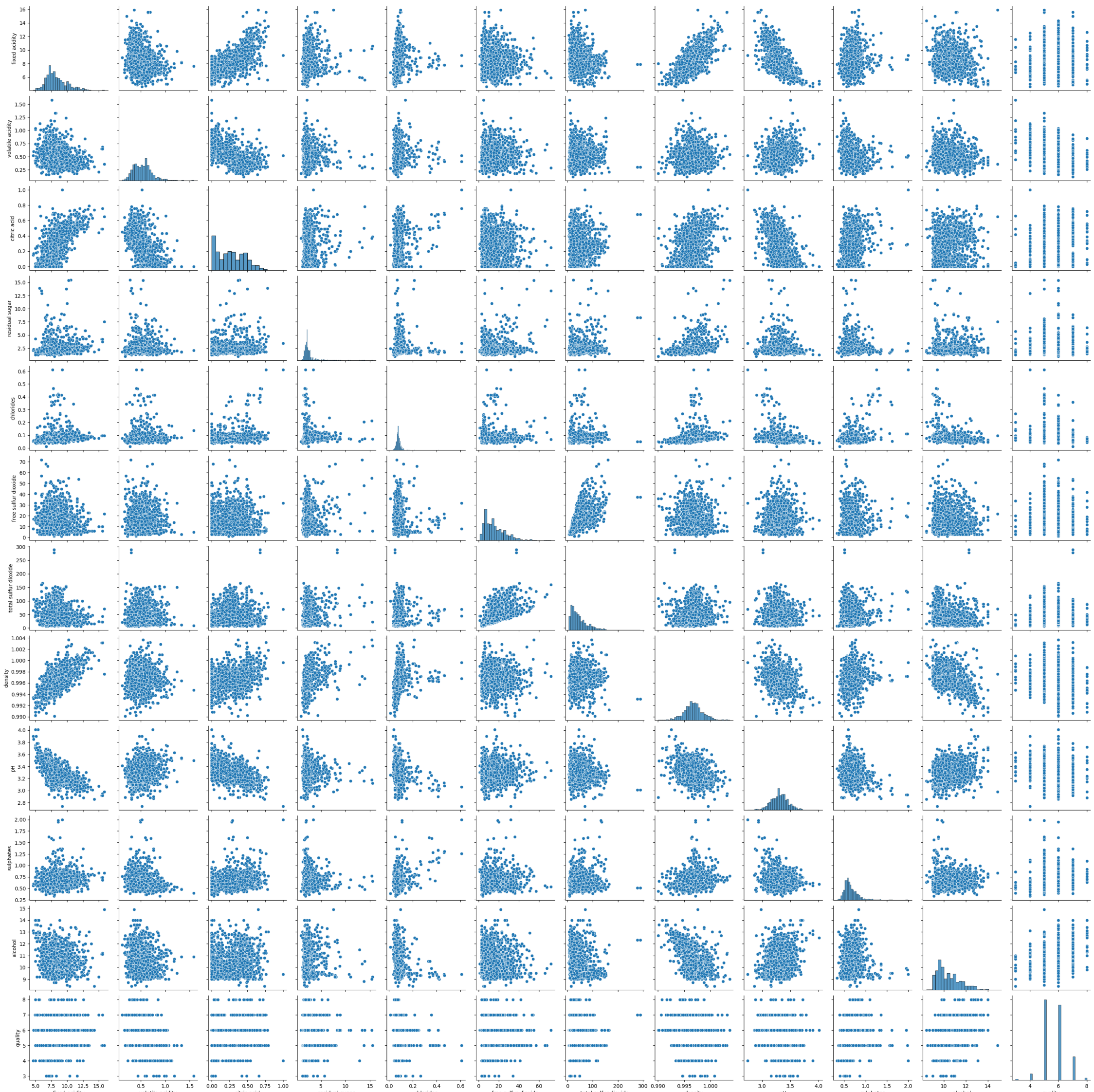
[ ]:
```
df.columns
```

[ ]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

3 EDA and Visualization

[ ]:
```
sns.pairplot(df)
```

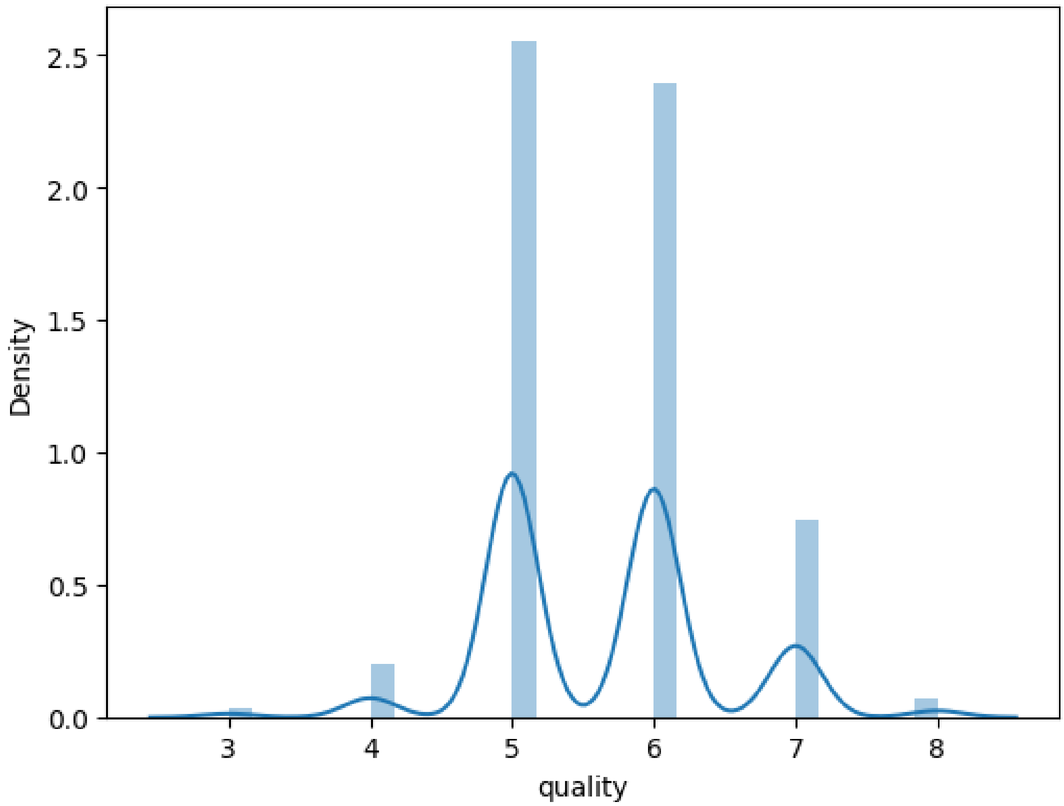
[ ]:
```
<seaborn.axisgrid.PairGrid at 0x78ecea064e20>
```



```
[ ]: sns.distplot(df['quality'])
```

```
<ipython-input-15-e9b2f3ff6ab5>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['quality'])
```

[ ]:
```
<Axes: xlabel='quality', ylabel='Density'>
```



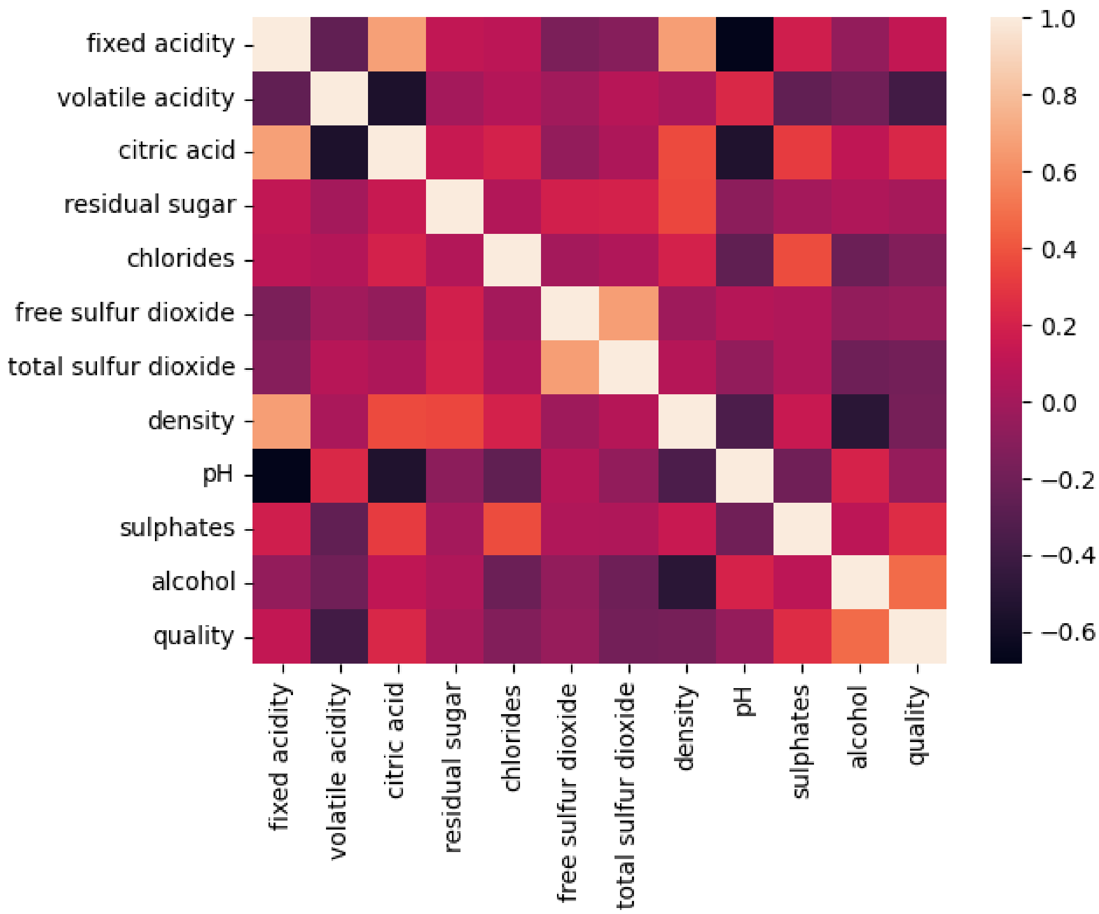
[ ]:
```
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
df1
```

[ ]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
…                …                 …            …               …         …
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
```

```
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
…                   …                   …     …     …        …
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
…        …       …
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]
```

[ ]:
```
sns.heatmap(df1.corr())
```

[ ]:
```
<Axes: >
```



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

[ ]:
```
x=df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol']]
y=df1['quality']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

[ ]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
print(lr.intercept_)
```

```
0.050257392639916354
```

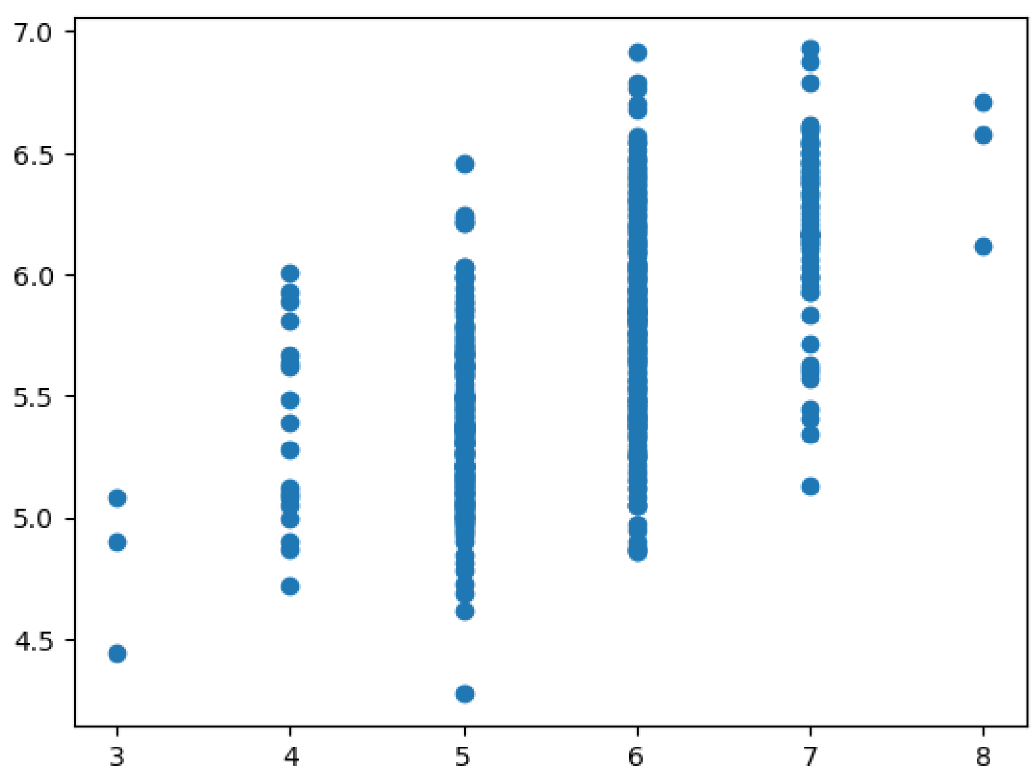
[ ]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[ ]:

	Co-efficient
fixed acidity	0.018465
volatile acidity	-1.024140
citric acid	-0.196371
residual sugar	-0.001517
chlorides	-1.384938
free sulfur dioxide	0.006097
total sulfur dioxide	-0.003816
density	3.808896
pH	-0.388887
sulphates	0.891870
alcohol	0.301287

[ ]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[ ]: <matplotlib.collections.PathCollection at 0x78ed30f400d0>



[ ]:
```
lr.score(x_test,y_test)
```

[ ]: 0.3691659656943862

[ ]:
```
lr.score(x_train,y_train)
```

[ ]: 0.35353834521490224

[ ]:
```
from sklearn.linear_model import Ridge,Lasso
```

[ ]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

[ ]: Ridge(alpha=10)

[ ]:
```
rr.score(x_test,y_test)
```

[ ]: 0.3455521805513252

[ ]:
```
rr.score(x_train,y_train)
```

[ ]: 0.34492329911839004

[ ]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

[ ]: Lasso(alpha=10)

[ ]:
```
la.score(x_test,y_test)
```

[ ]: -4.000455166131012e-05

[ ]:
```
la.score(x_train,y_train)
```

[ ]: 0.0

[ ]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

[ ]: ElasticNet()

[ ]:
```
en.coef_
```

```
[ ]: array([ 0.        , -0.        ,  0.        ,  0.        , -0.        ,
        0.00244197, -0.00514832, -0.        , -0.        ,  0.        ,
        0.        ])
```

```
[ ]: en.intercept_
```

```
[ ]: 5.836665247754942
```

```
[ ]: prediction = en.predict(x_test)
     prediction
```

```
[ ]: array([5.67356778, 5.65000376, 5.67006828, 5.53918265, 5.72425785,
       5.50809088, 5.66247799, 5.45337254, 5.66821761, 5.74835063,
       5.66736193, 5.75620531, 5.8025402 , 5.68848398, 5.72181588,
       5.65680091, 5.57825417, 5.74076034, 5.74023158, 5.75217705,
       5.66868384, 5.65059506, 5.70340018, 5.76841516, 5.58782021,
       5.56221079, 5.73534764, 5.5215601 , 5.78194692, 5.77059275,
       5.24605523, 5.30889262, 5.554892  , 5.55766089, 5.80524655,
       5.1539142 , 5.74405799, 5.73943843, 5.74293793, 5.76650195,
       5.77409224, 5.79197918, 5.67277463, 5.77112151, 5.72669982,
       5.53171743, 5.72260903, 5.61317236, 5.80280458, 5.21490092,
       5.68986842, 5.45904963, 5.72022959, 5.67251025, 5.71937391,
       5.64927315, 5.73943843, 5.78953721, 5.18942369, 5.65792097,
       5.7971275 , 5.76326684, 5.63244374, 5.46499109, 5.73561202,
       5.6698039 , 5.69257478, 5.54485973, 5.47964292, 5.41753614,
       5.42512643, 5.78735962, 5.73943843, 5.76518004, 5.78980159,
       5.77706298, 5.78980159, 5.5643955 , 5.63441949, 5.71475435,
       5.5901371 , 5.40888832, 5.62564659, 5.69145471, 5.80280458,
       5.46439979, 5.73778961, 5.79983385, 5.69851624, 5.5703995 ,
       5.76135363, 5.7031358 , 5.67660105, 5.73184814, 5.64544674,
       5.53898081, 5.49640979, 5.76353122, 5.79765626, 5.63462133,
       5.75620531, 5.75567654, 5.67898048, 5.61917636, 5.81310123,
       5.4275684 , 5.62564659, 5.73455449, 5.67059704, 5.6787161 ,
       5.72940617, 5.74267355, 5.72181588, 5.73561202, 5.73429011,
       5.44222022, 5.18539543, 5.70148698, 5.73455449, 5.77435662,
       5.73290567, 5.62346901, 5.6752166 , 5.56551556, 5.40288431,
       5.7507926 , 5.66577564, 5.70154951, 5.63079492, 5.72669982,
       5.40888832, 5.43786504, 5.44571971, 5.56280921, 5.6752166 ,
       5.33245664, 5.66300675, 5.68095623, 5.70340018, 5.60426016,
       5.72669982, 5.54380221, 5.65633468, 5.60069813, 5.4186562 ,
       5.59337222, 5.70340018, 5.63277066, 5.64485544, 5.72775735,
       5.72181588, 5.43924949, 5.64241347, 5.79983385, 5.65000376,
       5.78709524, 5.77382786, 5.72940617, 5.74973508, 5.37852715,
       5.37852715, 5.40968146, 5.42386706, 5.70148698, 5.64003403,
       5.36064022, 5.76676633, 5.72260903, 5.7692083 , 5.72940617,
       5.66874637, 5.40968146, 5.67277463, 5.70749098, 5.76650195,
       5.76867954, 5.75323457, 5.3634091 , 5.48717067, 5.5888152 ,
```

5.56657308, 5.64221162, 5.63759206, 5.69495421, 5.6275598 ,
5.8025402 , 5.76867954, 5.28235786, 5.6828069 , 5.71125486,
5.64485544, 5.72940617, 5.65633468, 5.72617106, 5.74023158,
5.72775735, 5.72617106, 5.59858308, 5.71693194, 5.12414433,
5.61699878, 5.79765626, 5.58934396, 5.76056048, 5.74617305,
5.6882196 , 5.80524655, 5.70042945, 5.80524655, 5.40261993,
5.73131938, 5.44981051, 5.79983385, 5.66168485, 5.60887972,
5.78359574, 5.72452223, 5.77303472, 5.42056941, 5.26671105,
5.73805399, 5.72537791, 5.74782187, 5.73943843, 5.5901371 ,
5.79765626, 5.62967485, 5.61614309, 5.409744 , 5.54300906,
5.35357869, 5.74782187, 5.75105699, 5.61567687, 5.6752166 ,
5.61911383, 5.69798748, 5.77924056, 5.72722858, 5.77706298,
5.48743505, 5.64571112, 5.73429011, 5.63673638, 5.64135594,
5.60426016, 5.77679859, 5.6275598 , 5.77165027, 4.49700569,
5.65897849, 5.70960603, 5.77409224, 5.58610885, 5.66848199,
5.55469015, 5.76894392, 5.61099477, 5.79494991, 5.77897618,
5.54109586, 5.5888152 , 5.66874637, 5.64241347, 5.70069383,
5.80009823, 5.80498217, 5.67686543, 5.63732768, 5.73805399,
5.74782187, 5.77924056, 5.54630672, 5.48188304, 5.64683118,
5.70419333, 5.29153444, 5.68148499, 5.61587871, 5.59634295,
5.71858077, 5.76894392, 5.41832928, 5.75297019, 5.75864728,
5.60999978, 5.66933767, 5.62941047, 5.76894392, 5.68557578,
5.58313811, 5.72749297, 5.75811851, 5.72802173, 5.50076496,
5.50182249, 5.76894392, 5.67224587, 5.76300245, 5.59475666,
5.75864728, 5.79983385, 5.79468553, 5.35304992, 5.29938912,
5.61884945, 5.58670015, 5.76570881, 5.53950957, 5.70148698,
5.73752523, 5.41271473, 5.69990069, 5.32321752, 5.74485114,
5.70960603, 5.74023158, 5.40591758, 5.77409224, 5.71369683,
5.52294454, 5.31186335, 5.23410976, 5.7583829 , 5.4275684 ,
5.70478463, 5.50505761, 5.7971275 , 5.79739188, 5.25985137,
5.7971275 , 5.56960635, 5.78709524, 5.61026417, 5.80009823,
5.64973938, 5.57574967, 5.68333566, 5.71693194, 5.7507926 ,
5.69171909, 5.49917868, 5.65244573, 5.67092396, 5.70340018,
4.44037415, 5.5519838 , 5.71448997, 5.77924056, 5.37773401,
5.73481888, 5.67548099, 5.72749297, 5.74564428, 5.76894392,
5.78194692, 5.70287142, 5.42077125, 5.72425785, 5.54485973,
5.68986842, 5.75052822, 5.6459755 , 5.58855082, 5.7418804 ,
5.79250794, 5.6329725 , 5.80009823, 5.72775735, 5.74973508,
5.63323689, 5.54762862, 5.55957409, 5.67277463, 5.70557777,
5.71858077, 5.34875728, 5.61396551, 5.71448997, 5.77085713,
5.62617536, 5.233581 , 5.72478661, 5.61911383, 5.76650195,
5.67059704, 5.71910953, 5.63435695, 5.65165258, 5.74432238,
5.75026384, 5.70340018, 5.58393126, 5.75349896, 5.68657077,
5.78980159, 5.7453799 , 5.68795522, 5.40577828, 5.76894392,
5.77679859, 5.59310784, 5.65680091, 5.13332091, 5.70472209,
5.54927745, 5.74214479, 5.76518004, 5.47125948, 5.54947929,
5.67904302, 5.73019932, 5.78194692, 5.67171711, 5.73534764,

```
       5.66247799, 5.65053252, 5.48769943, 5.3634091 , 5.15312106,
       5.77409224, 5.74293793, 5.75864728, 5.58096052, 5.74835063,
       5.53112613, 5.68036493, 5.4574008 , 5.74320231, 5.61944075,
       5.53818766, 5.13932492, 5.35549189, 5.76300245, 5.78735962,
       5.76326684, 5.74835063, 5.712048  , 5.72775735, 5.78438889,
       5.80498217, 5.63462133, 5.75270581, 5.1913369 , 5.2812378 ,
       5.7583829 , 5.35990961, 5.47687403, 5.77138589, 5.7637956 ,
       5.54895053, 5.45245432, 5.65000376, 5.579903  , 5.73778961,
       5.7822113 , 5.71448997, 5.72749297, 5.72399347, 5.78683086,
       5.73534764, 5.76300245, 5.77165027, 5.79983385, 5.70392895,
       5.46254912, 5.78438889, 5.62432469, 5.51185475, 5.70881289,
       5.76676633, 5.68986842, 5.51053285, 5.63323689, 5.68577763,
       5.66815507, 5.64326915, 5.65950726, 5.78980159, 5.65415709])
```

[ ]:
```
en.score(x_test,y_test)
```

[ ]: 0.019999918114022575

[ ]:
```
from sklearn import metrics
```

[ ]:
```
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error:  0.6500543527570254
```

[ ]:
```
print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error:  0.6342395495230034
```

[ ]:
```
print("Root Mean Squared Error: ", np.sqrt(metrics.
 ↪mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error:  0.796391580519912
```