Wine Quality Pridiction

Bharat Machine Learning Internship

bjective : To Predict Quality of Wine Using Linear Regration

uthor : Sonar Daxita Ramakantbhai

ethod Use : Linear Regration

ata Set : https://www.kaggle.com/datasets/yasserh/wine-quality-dataset?resource=download

Steps We are Followed

- 1) Reading the dataset
- 2) Checking and cleaning the data
- 3) Visualize the data
- 4) predict the wine Quality
- 5) Draw the conclusions

import the required library set

```
import pandas as pd
import numpy as np
import matplotlib as mlt
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import sklearn
from sklearn.model_selection import train_test_split
#supress warning
import warnings
warnings.filterwarnings("ignore")
```

Read the data

```
data_wine=pd.read_csv("wine.csv")
print("data has been successfully import..")
```

```
data has been successfully import..
```

Set directory

```
import os
os.getcwd()
```

```
'C:\\Users\\Admin\\Bharat_intern'
```

```
os.chdir("E:\Bharat_intern\Task_2")
os.getcwd()
```

```
'E:\\Bharat_intern\\Task_2'
```

```
#check first five observations
data_wine.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  Id
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
```

```
#check last five observations
data_wine.tail()
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
1138            6.3             0.510         0.13             2.3
0.076
1139            6.8             0.620         0.08             1.9
0.068
1140            6.2             0.600         0.08             2.0
0.090
1141            5.9             0.550         0.10             2.2
0.062
1142            5.9             0.645         0.12             2.0
0.075

      free sulfur dioxide  total sulfur dioxide  density    pH
sulphates  \
1138                 29.0                  40.0  0.99574  3.42
0.75
1139                 28.0                  38.0  0.99651  3.42
0.82
1140                 32.0                  44.0  0.99490  3.45
0.58
1141                 39.0                  51.0  0.99512  3.52
0.76
1142                 32.0                  44.0  0.99547  3.57
0.71

      alcohol  quality    Id
1138     11.0        6  1592
1139      9.5        6  1593
1140     10.5        5  1594
1141     11.2        6  1595
1142     10.2        5  1597
```

```
#Checking the shape of data
data_wine.shape
```

```
(1143, 13)
```

```
#checking info of data
data_wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

```
#statistical summary of the data
data_wine.describe()
```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1143.000000       1143.000000  1143.000000     1143.000000   
mean        8.311111          0.531339     0.268364        2.532152   
std         1.747595          0.179633     0.196686        1.355917   
min         4.600000          0.120000     0.000000        0.900000   
25%         7.100000          0.392500     0.090000        1.900000   
50%         7.900000          0.520000     0.250000        2.200000   
75%         9.100000          0.640000     0.420000        2.600000   
max        15.900000          1.580000     1.000000       15.500000   

         chlorides  free sulfur dioxide  total sulfur dioxide
density  \
count  1143.000000          1143.000000           1143.000000
1143.000000   
mean      0.086933            15.615486             45.914698
0.996730   
std       0.047267            10.250486             32.782130
0.001925   
min       0.012000             1.000000              6.000000
0.990070   
25%       0.070000             7.000000             21.000000
0.995570   
50%       0.079000            13.000000             37.000000
0.996680   
75%       0.090000            21.000000             61.000000
0.997845   
```

```
max        0.611000            68.000000            289.000000
1.003690

                pH   sulphates     alcohol     quality          Id
count  1143.000000 1143.000000 1143.000000 1143.000000 1143.000000
mean      3.311015    0.657708   10.442111    5.657043  804.969379
std       0.156664    0.170399    1.082196    0.805824  463.997116
min       2.740000    0.330000    8.400000    3.000000    0.000000
25%       3.205000    0.550000    9.500000    5.000000  411.000000
50%       3.310000    0.620000   10.200000    6.000000  794.000000
75%       3.400000    0.730000   11.100000    6.000000 1209.500000
max       4.010000    2.000000   14.900000    8.000000 1597.000000
```

```
#checking the null value of our data
data_wine.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
```

Exploratory Data Analysis

Data Visualization

```
for col in data_wine:
    print(data_wine[col].value_counts(ascending=False), '\n\n\n')
```

```
7.2     43
7.1     41
7.0     40
7.8     40
```

```
7.5     37
        ..
4.6      1
13.7     1
13.4     1
13.5     1
12.2     1
Name: fixed acidity, Length: 91, dtype: int64
```

```
0.600    32
0.500    32
0.430    31
0.390    29
0.580    28
         ..
1.035     1
0.565     1
0.865     1
0.965     1
0.160     1
Name: volatile acidity, Length: 135, dtype: int64
```

```
0.00    99
0.49    47
0.24    42
0.02    35
0.01    26
        ..
0.61     1
0.72     1
1.00     1
0.75     1
0.62     1
Name: citric acid, Length: 77, dtype: int64
```

```
2.00    107
2.10    103
1.80     92
2.20     88
1.90     80
       ...
7.30      1
7.20      1
```

```
2.95      1
3.65      1
4.40      1
Name: residual sugar, Length: 80, dtype: int64
```

```
0.080    48
0.077    41
0.074    38
0.084    38
0.078    36
         ..
0.222     1
0.422     1
0.034     1
0.387     1
0.230     1
Name: chlorides, Length: 131, dtype: int64
```

```
6.0     99
5.0     80
12.0    58
10.0    52
15.0    51
7.0     51
9.0     48
16.0    47
8.0     45
17.0    40
11.0    39
13.0    39
14.0    38
18.0    37
3.0     33
19.0    32
4.0     31
21.0    30
23.0    23
26.0    21
24.0    21
27.0    21
25.0    20
20.0    18
32.0    18
31.0    16
28.0    15
29.0    14
```

```
22.0    12
34.0    12
30.0    10
36.0     9
35.0     9
33.0     8
38.0     8
41.0     5
48.0     4
1.0      3
40.0     3
42.0     3
43.0     2
45.0     2
52.0     2
51.0     2
37.5     2
68.0     2
39.0     2
46.0     1
53.0     1
40.5     1
55.0     1
37.0     1
66.0     1
Name: free sulfur dioxide, dtype: int64
```

```
28.0     36
15.0     28
14.0     27
20.0     27
18.0     26
         ..
114.0     1
135.0     1
129.0     1
165.0     1
151.0     1
Name: total sulfur dioxide, Length: 138, dtype: int64
```

```
0.99760    27
0.99720    25
0.99680    22
0.99940    22
0.99640    21
           ..
```

```
0.99438     1
0.99634     1
0.99426     1
0.99747     1
0.99651     1
Name: density, Length: 388, dtype: int64
```

```
3.30    41
3.36    40
3.38    38
3.39    37
3.26    33
        ..
2.86     1
2.95     1
2.74     1
3.75     1
2.90     1
Name: pH, Length: 87, dtype: int64
```

```
0.60    53
0.62    50
0.56    47
0.54    46
0.57    42
        ..
1.61     1
1.31     1
0.33     1
1.56     1
1.01     1
Name: sulphates, Length: 89, dtype: int64
```

```
9.500000     92
9.400000     72
9.800000     57
9.200000     50
10.000000    49
             ..
11.950000     1
9.950000      1
9.233333      1
9.250000      1
```

```
10.550000     1
Name: alcohol, Length: 61, dtype: int64
```

```
5    483
6    462
7    143
4     33
8     16
3      6
Name: quality, dtype: int64
```

```
0       1
1079    1
1087    1
1086    1
1085    1
       ..
543     1
544     1
545     1
546     1
1597    1
Name: Id, Length: 1143, dtype: int64
```

- No junk present in our data

```
#data split
from sklearn.model_selection import train_test_split
np.random.seed(0)
df_train,df_test=train_test_split(data_wine,train_size=0.7,test_size=0
.3,random_state=100)
```

```
# Data Visualization
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 97 to 792
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         800 non-null    float64
 1   volatile acidity      800 non-null    float64
 2   citric acid           800 non-null    float64
 3   residual sugar        800 non-null    float64
```

```
 4   chlorides             800 non-null    float64
 5   free sulfur dioxide   800 non-null    float64
 6   total sulfur dioxide  800 non-null    float64
 7   density               800 non-null    float64
 8   pH                    800 non-null    float64
 9   sulphates             800 non-null    float64
 10  alcohol               800 non-null    float64
 11  quality               800 non-null    int64
 12  Id                    800 non-null    int64
dtypes: float64(11), int64(2)
memory usage: 87.5 KB
```

```
df_train.shape
```

```
(800, 13)
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 343 entries, 459 to 166
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         343 non-null    float64
 1   volatile acidity      343 non-null    float64
 2   citric acid           343 non-null    float64
 3   residual sugar        343 non-null    float64
 4   chlorides             343 non-null    float64
 5   free sulfur dioxide   343 non-null    float64
 6   total sulfur dioxide  343 non-null    float64
 7   density               343 non-null    float64
 8   pH                    343 non-null    float64
 9   sulphates             343 non-null    float64
 10  alcohol               343 non-null    float64
 11  quality               343 non-null    int64
 12  Id                    343 non-null    int64
dtypes: float64(11), int64(2)
memory usage: 37.5 KB
```

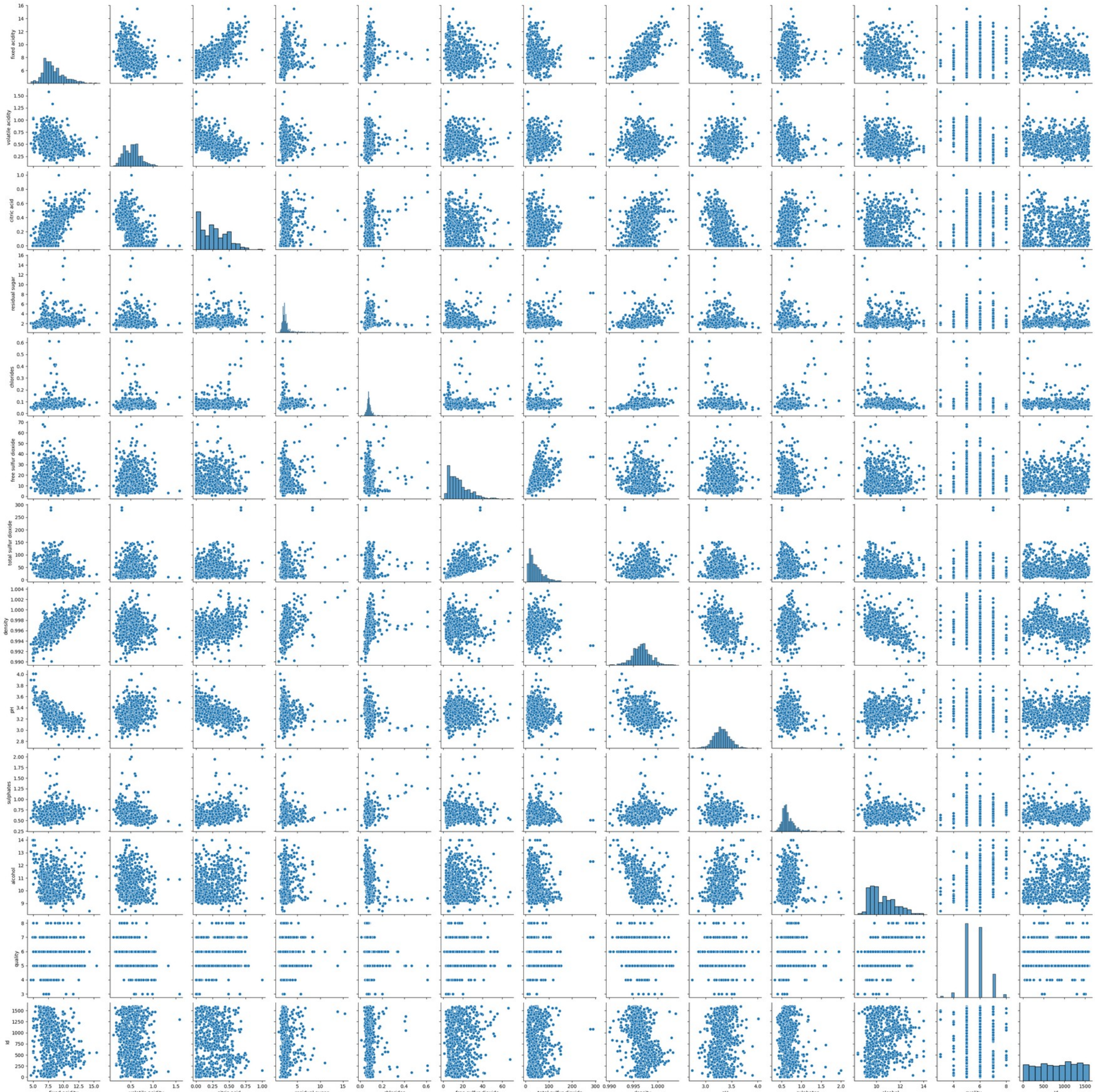
```
df_test.shape
```

```
(343, 13)
```

```
df_train.columns
```

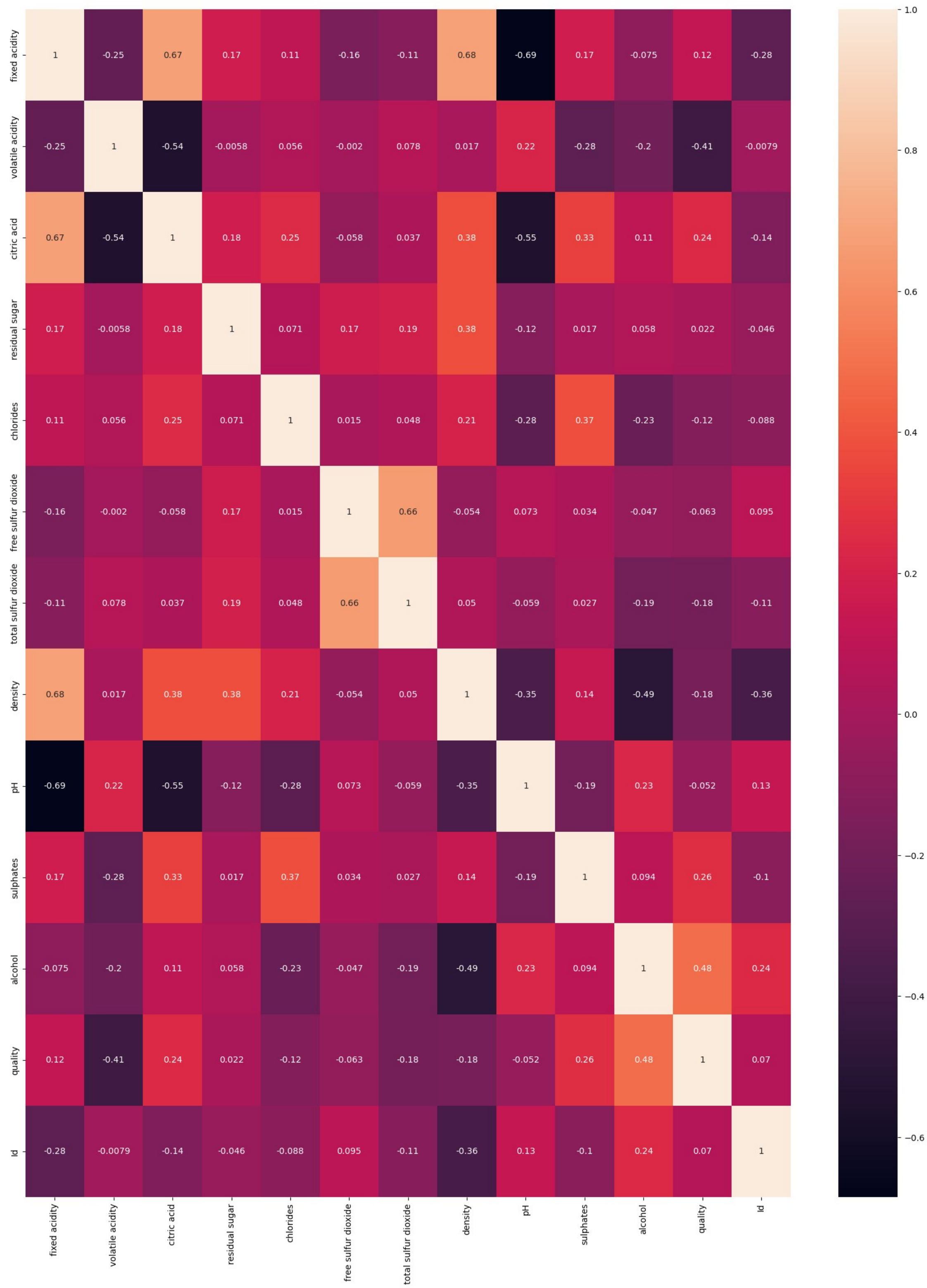
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual
sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',
'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

```
#using seabora lib
sns.pairplot(df_train)
plt.show()
```



Correlation plot

```
#using heatmap
plt.figure(figsize=(20,25))
sns.heatmap(data_wine.corr(), annot=True)
sns.light_palette("#a275ac", as_cmap=True)
plt.show()
```



Rescaling

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

```
df_train.head()
```

```
     fixed acidity  volatile acidity  citric acid  residual sugar  
chlorides  \
97             6.3              0.39         0.08             1.7      
0.066   
353            8.7              0.69         0.31             3.0      
0.086   
328           10.3              0.50         0.42             2.0      
0.069   
191            6.9              0.54         0.04             3.0      
0.077   
764            6.8              0.48         0.08             1.8      
0.074   

     free sulfur dioxide  total sulfur dioxide  density    pH  
sulphates  \
97                   3.0                  20.0  0.99540  3.34       
0.58   
353                 23.0                  81.0  1.00020  3.48       
0.74   
328                 21.0                  51.0  0.99820  3.16       
0.72   
191                  7.0                  27.0  0.99870  3.69       
0.91   
764                 40.0                  64.0  0.99529  3.12       
0.49   

     alcohol  quality    Id  
97       9.4        5   143  
353     11.6        6   499  
328     11.5        6   466  
191      9.4        6   268  
764      9.6        5  1085  
```

```
df_train.columns
```

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

```
df_train[:]=scaler.fit_transform(df_train[:])
```

```
df_train.head()
```

```
     fixed acidity  volatile acidity  citric acid  residual sugar  
chlorides  \
97        0.155963          0.184932         0.08        0.055172
```

```
0.090150
353       0.376147          0.390411         0.31        0.144828
0.123539
328       0.522936          0.260274         0.42        0.075862
0.095159
191       0.211009          0.287671         0.04        0.144828
0.108514
764       0.201835          0.246575         0.08        0.062069
0.103506

     free sulfur dioxide  total sulfur dioxide   density        pH
sulphates  \
97             0.029851              0.049470  0.391336  0.472441
0.149701
353            0.328358              0.265018  0.743759  0.582677
0.245509
328            0.298507              0.159011  0.596916  0.330709
0.233533
191            0.089552              0.074205  0.633627  0.748031
0.347305
764            0.582090              0.204947  0.383260  0.299213
0.095808

      alcohol  quality        Id
97   0.178571      0.4  0.089543
353  0.571429      0.6  0.312461
328  0.553571      0.6  0.291797
191  0.178571      0.6  0.167815
764  0.214286      0.4  0.679399
```

```
df_train.columns
```

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

Model fitting

```
y_train=df_train.pop('quality')

X_train=df_train

from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

#rfe = recursive feature elimination
lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
rfe = RFE(lm,n_features_to_select=9)
rfe = rfe.fit(X_train, y_train)
```

```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

```
[('fixed acidity', True, 1),
 ('volatile acidity', True, 1),
 ('citric acid', True, 1),
 ('residual sugar', True, 1),
 ('chlorides', True, 1),
 ('free sulfur dioxide', False, 3),
 ('total sulfur dioxide', False, 2),
 ('density', True, 1),
 ('pH', True, 1),
 ('sulphates', True, 1),
 ('alcohol', True, 1),
 ('Id', False, 4)]
```

Module Selection

```
col = X_train.columns[rfe.support_]
col
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual
sugar',
       'chlorides', 'density', 'pH', 'sulphates', 'alcohol'],
      dtype='object')
```

```
X_train.columns[~rfe.support_]
```

```
Index(['free sulfur dioxide', 'total sulfur dioxide', 'Id'],
dtype='object')
```

```
X_train_rfe = X_train[col]
```

```
# Module 1
#VIF check Variance Inflation Factor
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i
in range(X_train_rfe.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
           Features      VIF
5           density  1305.03
6                pH   979.43
8           alcohol   121.62
0     fixed acidity    71.94
7         sulphates    20.67
```

```
1  volatile acidity    16.42
2       citric acid     9.05
4         chlorides     6.58
3    residual sugar     5.04
```

```
import statsmodels.api as sm
X_train_lm1 = sm.add_constant(X_train_rfe)
lr1 = sm.OLS(y_train, X_train_lm1).fit()
```

```
#prameter
lr1.params
```

```
const               29.803076
fixed acidity        0.043829
volatile acidity    -1.204881
citric acid         -0.224131
residual sugar       0.023565
chlorides           -1.759261
density            -26.663378
pH                  -0.300273
sulphates            0.688950
alcohol              0.326336
dtype: float64
```

```
print(lr1.summary())
```

```
                            OLS Regression Results                            
======================================================================
========
Dep. Variable:                quality   R-squared:
0.395
Model:                            OLS   Adj. R-squared:
0.389
Method:                 Least Squares   F-statistic:
57.41
Date:                Sat, 20 May 2023   Prob (F-statistic):
2.00e-80
Time:                        16:49:20   Log-Likelihood:
-785.83
No. Observations:                 800   AIC:
1592.
Df Residuals:                     790   BIC:
1638.
Df Model:                           9

Covariance Type:            nonrobust

======================================================================
==============
                       coef    std err          t      P>|t|
```

```
[0.025      0.975]
------------------------------------------------------------------------------
--------------
const                29.8031     29.870      0.998      0.319     -
28.830      88.437
fixed acidity         0.0438      0.037      1.183      0.237      -
0.029      0.117
volatile acidity     -1.2049      0.166     -7.270      0.000      -
1.530     -0.880
citric acid          -0.2241      0.211     -1.061      0.289      -
0.639      0.190
residual sugar        0.0236      0.023      1.040      0.298      -
0.021      0.068
chlorides            -1.7593      0.597     -2.945      0.003      -
2.932     -0.587
density             -26.6634     30.485     -0.875      0.382     -
86.505      33.178
pH                   -0.3003      0.262     -1.144      0.253      -
0.815      0.215
sulphates             0.6889      0.160      4.318      0.000
0.376      1.002
alcohol               0.3263      0.038      8.695      0.000
0.253      0.400
==============================================================================
========
Omnibus:                         7.025   Durbin-Watson:
1.999
Prob(Omnibus):                   0.030   Jarque-Bera (JB):
9.664
Skew:                           -0.027   Prob(JB):
0.00797
Kurtosis:                        3.536   Cond. No.
2.62e+04
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.62e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```
# module 2
X_train_new = X_train_rfe.drop(["residual sugar"], axis = 1)
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
```

```
in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
            Features      VIF
4            density  1304.03
5                 pH   978.89
7            alcohol   120.69
0      fixed acidity    71.88
6          sulphates    20.54
1   volatile acidity    16.23
2        citric acid     8.90
3          chlorides     6.55
```

```
X_train_lm2 = sm.add_constant(X_train_new)

lr2 = sm.OLS(y_train, X_train_lm2).fit()

lr2.params
```

```
const                 11.550422
fixed acidity          0.026665
volatile acidity      -1.201916
citric acid           -0.200228
chlorides             -1.744789
density               -8.022114
pH                    -0.394655
sulphates              0.651506
alcohol                0.345695
dtype: float64
```

```
print(lr2.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:                quality   R-squared:
0.395
Model:                            OLS   Adj. R-squared:
0.388
Method:                 Least Squares   F-statistic:
64.44
Date:                Sat, 20 May 2023   Prob (F-statistic):
4.11e-81
Time:                        16:58:36   Log-Likelihood:
-786.37
No. Observations:                 800   AIC:
1591.
Df Residuals:                     791   BIC:
1633.
```

```
Df Model:                           8

Covariance Type:            nonrobust

======================================================================
==============
                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
--------------
const               11.5504     24.176      0.478      0.633     -
35.907      59.008
fixed acidity        0.0267      0.033      0.804      0.422      -
0.038       0.092
volatile acidity    -1.2019      0.166     -7.252      0.000      -
1.527      -0.877
citric acid         -0.2002      0.210     -0.954      0.341      -
0.612       0.212
chlorides           -1.7448      0.597     -2.922      0.004      -
2.917      -0.572
density             -8.0221     24.666     -0.325      0.745     -
56.440      40.396
pH                  -0.3947      0.246     -1.602      0.109      -
0.878       0.089
sulphates            0.6515      0.155      4.191      0.000
0.346       0.957
alcohol              0.3457      0.033     10.605      0.000
0.282       0.410
======================================================================
========
Omnibus:                        6.658   Durbin-Watson:
2.007
Prob(Omnibus):                  0.036   Jarque-Bera (JB):
9.003
Skew:                          -0.028   Prob(JB):
0.0111
Kurtosis:                       3.517   Cond. No.
2.09e+04
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.09e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```python
# module 3
X_train_new = X_train_rfe.drop(["density"], axis = 1)
```

```python
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

	Features	VIF
5	pH	161.56
7	alcohol	120.85
0	fixed acidity	41.09
6	sulphates	20.46
1	volatile acidity	16.35
2	citric acid	9.05
4	chlorides	6.19
3	residual sugar	5.03

```python
X_train_lm3 = sm.add_constant(X_train_new)

lr3 = sm.OLS(y_train, X_train_lm).fit()

lr3.params
```

```
const                3.687835
fixed acidity        0.018087
volatile acidity    -1.222027
citric acid         -0.222621
residual sugar       0.011922
chlorides           -1.790639
pH                  -0.433890
sulphates            0.649692
alcohol              0.351453
dtype: float64
```

```python
print(lr3.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:                quality   R-squared:
0.395
Model:                            OLS   Adj. R-squared:
0.389
Method:                 Least Squares   F-statistic:
64.51
Date:                Sat, 20 May 2023   Prob (F-statistic):
3.51e-81
Time:                        20:55:48   Log-Likelihood:
```

```
-786.21
No. Observations:                 800   AIC:
1590.
Df Residuals:                     791   BIC:
1633.
Df Model:                           8

Covariance Type:            nonrobust

======================================================================
==============
                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
--------------
const                3.6878      0.816      4.517      0.000
2.085       5.290
fixed acidity        0.0181      0.023      0.803      0.422      -
0.026       0.062
volatile acidity    -1.2220      0.165     -7.426      0.000      -
1.545      -0.899
citric acid         -0.2226      0.211     -1.054      0.292      -
0.637       0.192
residual sugar       0.0119      0.018      0.651      0.515      -
0.024       0.048
chlorides           -1.7906      0.596     -3.003      0.003      -
2.961      -0.620
pH                  -0.4339      0.213     -2.033      0.042      -
0.853      -0.015
sulphates            0.6497      0.153      4.244      0.000
0.349       0.950
alcohol              0.3515      0.024     14.545      0.000
0.304       0.399
======================================================================
========
Omnibus:                        7.064   Durbin-Watson:
2.002
Prob(Omnibus):                  0.029   Jarque-Bera (JB):
9.732
Skew:                          -0.028   Prob(JB):
0.00770
Kurtosis:                       3.537   Cond. No.
529.
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
# module 4
X_train_new = X_train_rfe.drop(["pH"], axis = 1)
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
            Features     VIF
5            density  215.27
7            alcohol  112.89
0      fixed acidity   53.44
6          sulphates   20.66
1   volatile acidity   16.33
2        citric acid    8.98
4          chlorides    6.37
3     residual sugar    5.03
```

```
X_train_lm4 = sm.add_constant(X_train_new)
```

```
lr4 = sm.OLS(y_train, X_train_lm4).fit()
```

```
lr4.params
```

```
const                48.994264
fixed acidity         0.073537
volatile acidity     -1.203757
citric acid          -0.206833
residual sugar        0.032520
chlorides            -1.637087
density             -46.963559
sulphates             0.720237
alcohol               0.301736
dtype: float64
```

```
print(lr4.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:                quality   R-squared:
0.394
Model:                            OLS   Adj. R-squared:
0.388
Method:                 Least Squares   F-statistic:
64.40
Date:                Sat, 20 May 2023   Prob (F-statistic):
```

```
4.59e-81
Time:                        20:58:12   Log-Likelihood:
-786.49
No. Observations:                 800   AIC:
1591.
Df Residuals:                     791   BIC:
1633.
Df Model:                           8

Covariance Type:            nonrobust

======================================================================
==============
                        coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
--------------
const                48.9943     24.719      1.982      0.048
0.471      97.518
fixed acidity         0.0735      0.026      2.781      0.006
0.022       0.125
volatile acidity     -1.2038      0.166     -7.262      0.000      -
1.529      -0.878
citric acid          -0.2068      0.211     -0.982      0.327      -
0.620       0.207
residual sugar        0.0325      0.021      1.530      0.126      -
0.009       0.074
chlorides            -1.6371      0.588     -2.785      0.005      -
2.791      -0.483
density             -46.9636     24.794     -1.894      0.059     -
95.633       1.706
sulphates             0.7202      0.157      4.581      0.000
0.412       1.029
alcohol               0.3017      0.031      9.806      0.000
0.241       0.362
======================================================================
========
Omnibus:                        7.851   Durbin-Watson:
2.001
Prob(Omnibus):                  0.020   Jarque-Bera (JB):
11.188
Skew:                          -0.027   Prob(JB):
0.00372
Kurtosis:                       3.577   Cond. No.
2.09e+04
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
```

```
correctly specified.
[2] The condition number is large, 2.09e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```python
# Module 5
X_train_new = X_train_rfe.drop(["sulphates"], axis = 1)
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
           Features      VIF
5           density  1292.05
6                pH   979.09
7           alcohol   119.81
0     fixed acidity    71.93
1  volatile acidity    15.81
2       citric acid     9.02
4         chlorides     5.55
3    residual sugar     5.01
```

```python
X_train_lm5 = sm.add_constant(X_train_new)

lr5 = sm.OLS(y_train, X_train_lm5).fit()

lr5.params
```

```
const                 -6.149806
fixed acidity          0.007604
volatile acidity      -1.359458
citric acid           -0.166521
residual sugar         0.001509
chlorides             -0.819417
density               10.361338
pH                    -0.494495
alcohol                0.373168
dtype: float64
```

```python
print(lr5.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:               quality   R-squared:
0.381
```

```
Model:                            OLS   Adj. R-squared:
0.375
Method:                 Least Squares   F-statistic:
60.90
Date:                Sat, 20 May 2023   Prob (F-statistic):
2.18e-77
Time:                        21:00:45   Log-Likelihood:
-795.16
No. Observations:                 800   AIC:
1608.
Df Residuals:                     791   BIC:
1650.
Df Model:                           8

Covariance Type:            nonrobust

======================================================================
==============
                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
--------------
const               -6.1498     29.004     -0.212      0.832     -
63.083      50.784
fixed acidity        0.0076      0.036      0.208      0.835      -
0.064      0.079
volatile acidity    -1.3595      0.164     -8.308      0.000      -
1.681     -1.038
citric acid         -0.1665      0.213     -0.781      0.435      -
0.585      0.252
residual sugar       0.0015      0.022      0.068      0.946      -
0.042      0.045
chlorides           -0.8194      0.562     -1.457      0.146      -
1.923      0.285
density             10.3613     29.579      0.350      0.726     -
47.701      68.424
pH                  -0.4945      0.261     -1.891      0.059      -
1.008      0.019
alcohol              0.3732      0.036     10.272      0.000
0.302      0.444
======================================================================
========
Omnibus:                        6.231   Durbin-Watson:
2.025
Prob(Omnibus):                  0.044   Jarque-Bera (JB):
8.337
Skew:                          -0.006   Prob(JB):
0.0155
Kurtosis:                       3.500   Cond. No.
2.52e+04
```

```
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.52e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```python
# Module 6
X_train_new = X_train_rfe.drop(["chlorides"], axis = 1)
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
in range(X_train_new.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
           Features      VIF
4           density  1227.36
5                pH   947.68
7           alcohol   114.61
0     fixed acidity    68.01
6         sulphates    17.43
1  volatile acidity    15.06
2       citric acid     8.36
3    residual sugar     5.01
```

```python
X_train_lm6 = sm.add_constant(X_train_new)

lr6 = sm.OLS(y_train, X_train_lm6).fit()

lr6.params
```

```
const               34.502282
fixed acidity        0.064234
volatile acidity    -1.340852
citric acid         -0.395668
residual sugar       0.022011
density            -32.055519
pH                  -0.162084
sulphates            0.517704
alcohol              0.338818
dtype: float64
```

```python
print(lr6.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:                quality   R-squared:
0.389
Model:                            OLS   Adj. R-squared:
0.383
Method:                 Least Squares   F-statistic:
62.89
Date:                Sat, 20 May 2023   Prob (F-statistic):
1.72e-79
Time:                        21:02:48   Log-Likelihood:
-790.19
No. Observations:                 800   AIC:
1598.
Df Residuals:                     791   BIC:
1641.
Df Model:                           8

Covariance Type:            nonrobust

======================================================================
==============
                       coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
--------------
const               34.5023     29.971      1.151      0.250     -
24.331      93.335
fixed acidity        0.0642      0.037      1.756      0.079      -
0.008       0.136
volatile acidity    -1.3409      0.160     -8.383      0.000      -
1.655      -1.027
citric acid         -0.3957      0.204     -1.940      0.053      -
0.796       0.005
residual sugar       0.0220      0.023      0.967      0.334      -
0.023       0.067
density            -32.0555     30.578     -1.048      0.295     -
92.078      27.967
pH                  -0.1621      0.259     -0.625      0.532      -
0.671       0.347
sulphates            0.5177      0.149      3.467      0.001
0.225       0.811
alcohol              0.3388      0.037      9.041      0.000
0.265       0.412
======================================================================
========
Omnibus:                        8.039   Durbin-Watson:
2.021
```

```
Prob(Omnibus):                  0.018   Jarque-Bera (JB):
11.633
Skew:                          -0.005   Prob(JB):
0.00298
Kurtosis:                       3.591   Cond. No.
2.62e+04
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.62e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Suggestions are always welcome.

Thank You !