```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\nishant\anaconda3\
lib\site-packages (0.0)Note: you may need to restart the kernel to use
updated packages.

Requirement already satisfied: scikit-learn in c:\users\nishant\
anaconda3\lib\site-packages (from sklearn) (0.24.2)
Requirement already satisfied: scipy>=0.19.1 in c:\users\nishant\
anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\
nishant\anaconda3\lib\site-packages (from scikit-learn->sklearn)
(2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\nishant\
anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\nishant\
anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.20.3)
```

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn.model_selection import train_test_split
```

```
data= pd.read_csv(R"C:\Users\Nishant\Downloads\QualityPrediction.csv")
```

```
data.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
```

```
data.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
```

```
data.isnull().sum()
```

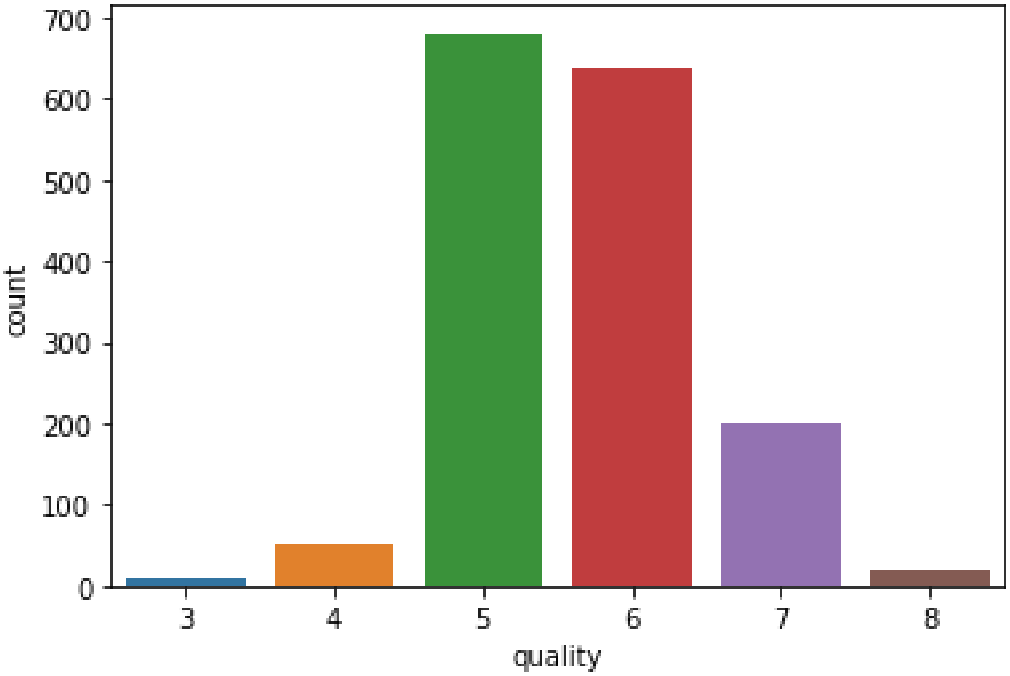
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality   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

```
data['quality'].value_counts()
```

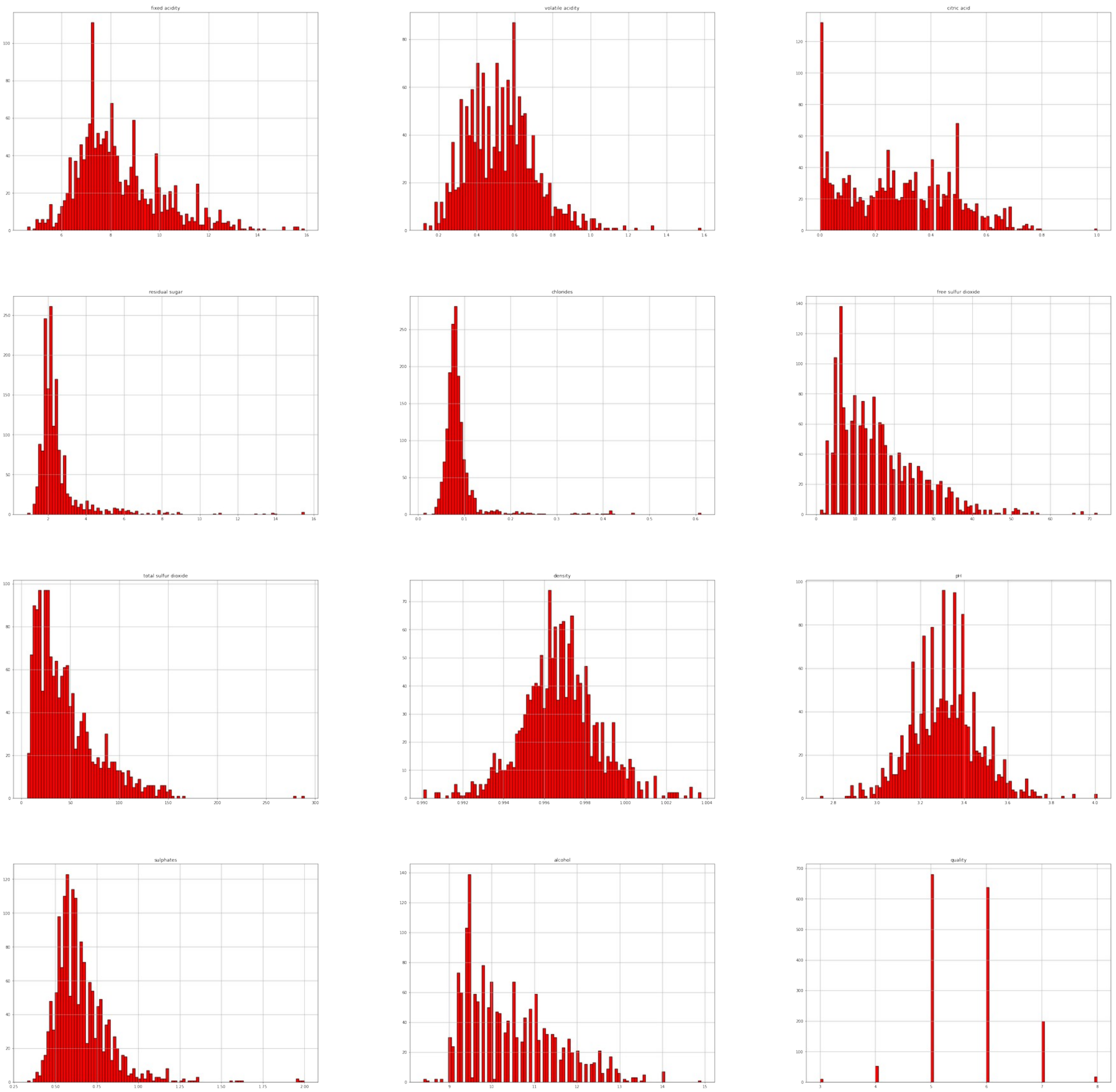
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
sb.countplot(data['quality'])
xlable='quality'
ylable='count'
plt.savefig('bar')
```

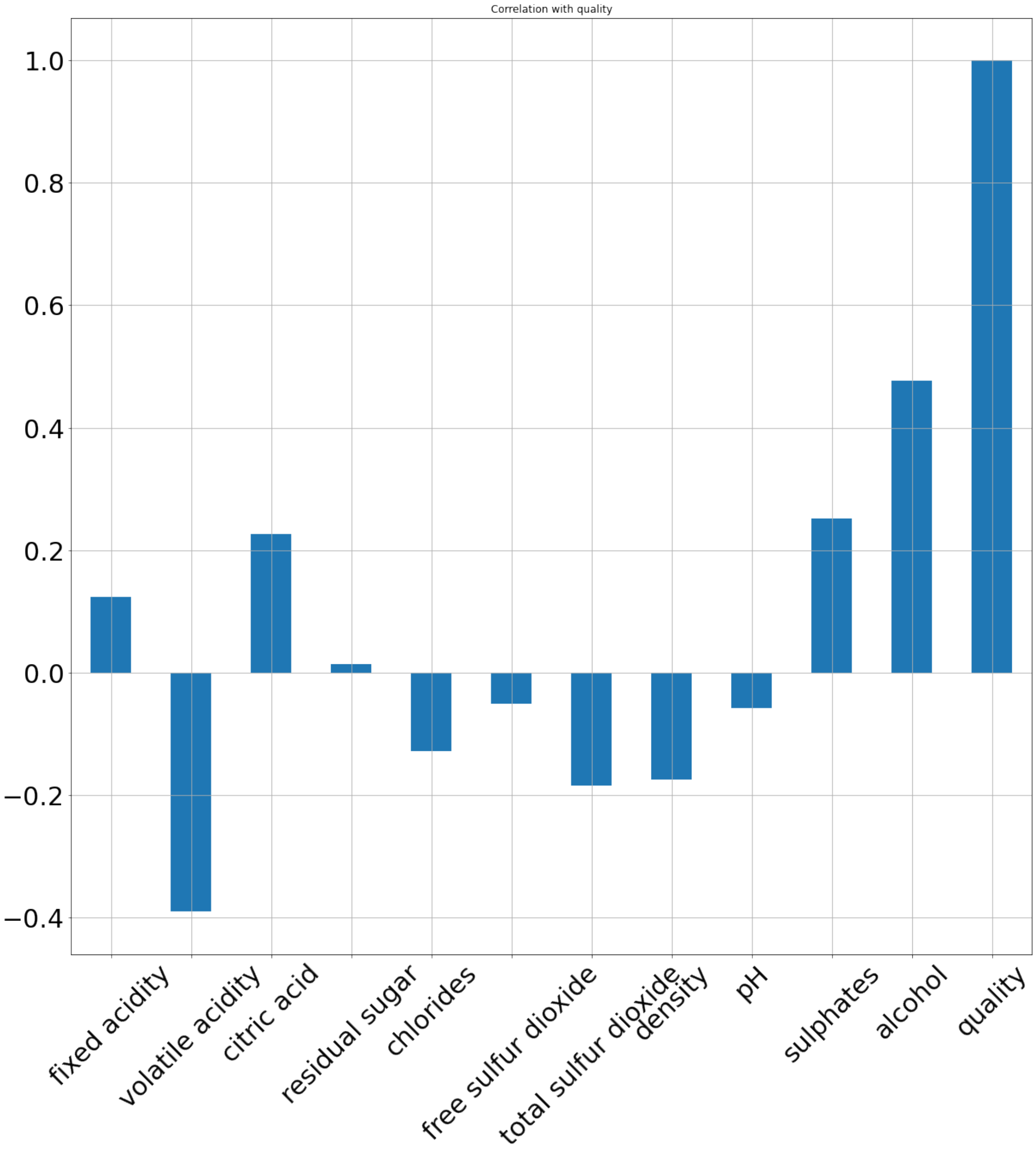
```
C:\Users\Nishant\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
data.hist(bins=100, figsize=(50,50),color='red',edgecolor='black')
plt.show()
```



```
data.corrwith(data.quality).plot.bar(
    figsize=(20,20),title='Correlation with
quality',fontsize='30',rot='45',grid=True)
plt.savefig('same1')
```



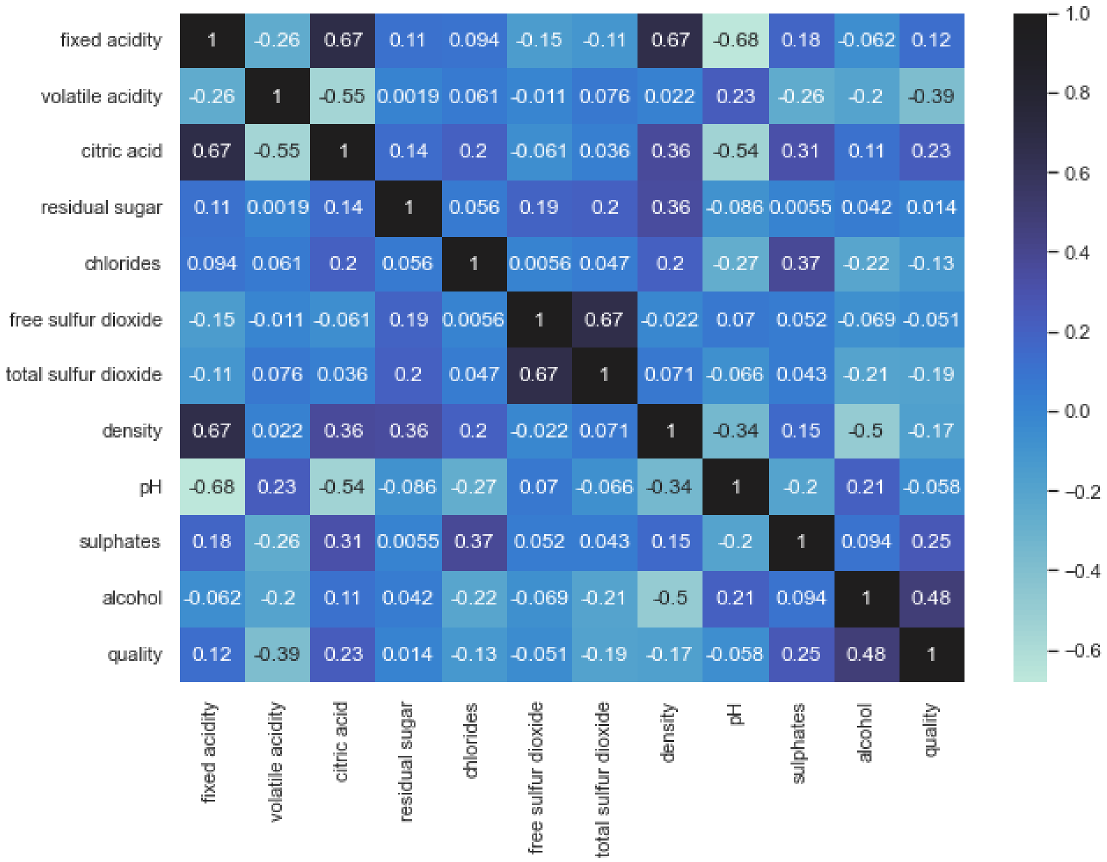
```
sb.set(style='white')
corr=data.corr()

corr.head
```

```
<bound method NDFrame.head of                          fixed acidity  
volatile acidity  citric acid  \
fixed acidity                 1.000000         -0.256131     0.671703
volatile acidity             -0.256131          1.000000    -0.552496
citric acid                   0.671703         -0.552496     1.000000
residual sugar                0.114777          0.001918     0.143577
chlorides                     0.093705          0.061298     0.203823
free sulfur dioxide          -0.153794         -0.010504    -0.060978
```

```
total sulfur dioxide      -0.113181          0.076470     0.035533
density                    0.668047          0.022026     0.364947
pH                        -0.682978          0.234937    -0.541904
sulphates                  0.183006         -0.260987     0.312770
alcohol                   -0.061668         -0.202288     0.109903
quality                    0.124052         -0.390558     0.226373

                      residual sugar  chlorides  free sulfur
dioxide  \
fixed acidity               0.114777   0.093705            -0.153794

volatile acidity            0.001918   0.061298            -0.010504

citric acid                 0.143577   0.203823            -0.060978

residual sugar              1.000000   0.055610             0.187049

chlorides                   0.055610   1.000000             0.005562

free sulfur dioxide         0.187049   0.005562             1.000000

total sulfur dioxide        0.203028   0.047400             0.667666

density                     0.355283   0.200632            -0.021946

pH                         -0.085652  -0.265026             0.070377

sulphates                   0.005527   0.371260             0.051658

alcohol                     0.042075  -0.221141            -0.069408

quality                     0.013732  -0.128907            -0.050656


                      total sulfur dioxide   density        pH
sulphates  \
fixed acidity                    -0.113181  0.668047 -0.682978
0.183006
volatile acidity                  0.076470  0.022026  0.234937  -
0.260987
citric acid                       0.035533  0.364947 -0.541904
0.312770
residual sugar                    0.203028  0.355283 -0.085652
0.005527
chlorides                         0.047400  0.200632 -0.265026
0.371260
free sulfur dioxide               0.667666 -0.021946  0.070377
0.051658
total sulfur dioxide              1.000000  0.071269 -0.066495
```

```
0.042947
density                           0.071269  1.000000 -0.341699
0.148506
pH                               -0.066495 -0.341699  1.000000  -
0.196648
sulphates                         0.042947  0.148506 -0.196648
1.000000
alcohol                          -0.205654 -0.496180  0.205633
0.093595
quality                          -0.185100 -0.174919 -0.057731
0.251397

                       alcohol   quality
fixed acidity        -0.061668  0.124052
volatile acidity     -0.202288 -0.390558
citric acid           0.109903  0.226373
residual sugar        0.042075  0.013732
chlorides            -0.221141 -0.128907
free sulfur dioxide  -0.069408 -0.050656
total sulfur dioxide -0.205654 -0.185100
density              -0.496180 -0.174919
pH                    0.205633 -0.057731
sulphates             0.093595  0.251397
alcohol               1.000000  0.476166
quality               0.476166  1.000000  >
```

```
plt.figure(figsize=(10,7))
sb.heatmap(data.corr(), annot=True,center=1)
plt.title('Correlation of Data',size=20,color='white')
plt.savefig('same')
```



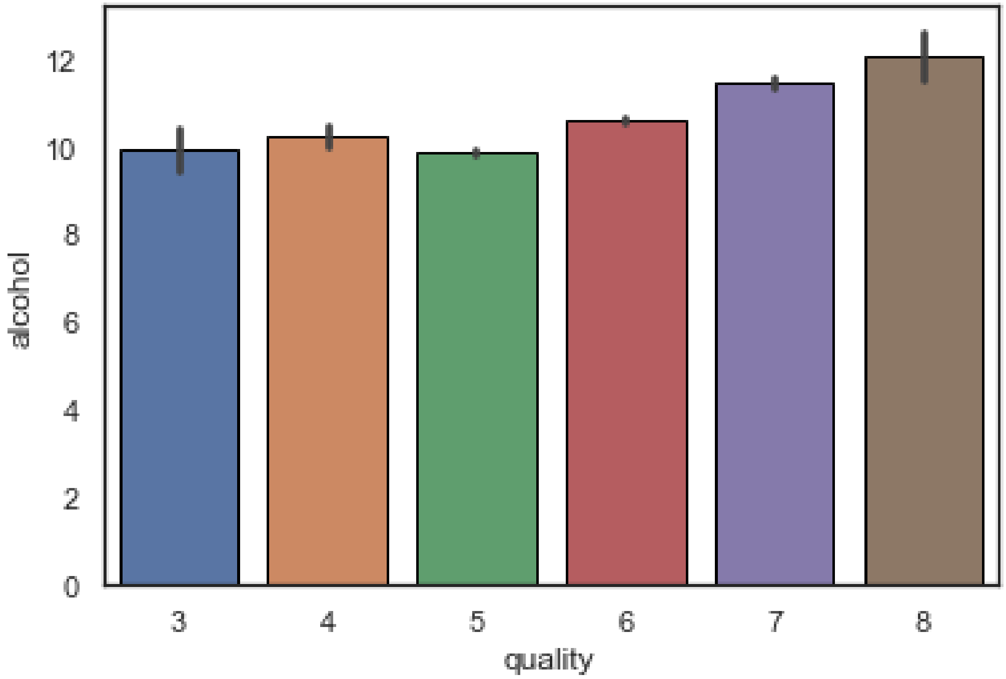
```
data.corr()['quality'].sort_values()
```

```
volatile acidity       -0.390558
total sulfur dioxide   -0.185100
density                -0.174919
chlorides              -0.128907
pH                     -0.057731
free sulfur dioxide    -0.050656
residual sugar          0.013732
fixed acidity           0.124052
citric acid             0.226373
sulphates               0.251397
alcohol                 0.476166
quality                 1.000000
Name: quality, dtype: float64
```

```
sb.barplot(data['quality'],data['alcohol'],edgecolor='black')
plt.savefig('sample')
```

```
C:\Users\Nishant\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variables as
keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an
```

```
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
 data['quality']= data.quality.apply(lambda x:1if x>=7 else 0)
```

```
data['quality'].value_counts()
```

```
0    1382
1     217
Name: quality, dtype: int64
```

```
x=data.drop('quality',axis=1)
y=data['quality']
```

```
x.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
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: quality, dtype: int64
```

```
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.3,random_state=0)
```

```
print('x_train',x_train.shape)
print('y_train',y_train.shape)
print('x_test',x_test.shape)
print('y_test',y_test.shape)
```

```
x_train (1119, 11)
y_train (1119,)
x_test (480, 11)
y_test (480,)
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import tree
```

```
decision_tree = tree.DecisionTreeClassifier()
decision_tree.fit(x_train,y_train)
decision_tree_pred = decision_tree.predict(x_test)
decision_tree_acc= accuracy_score(y_test,decision_tree_pred)
print("Test accuracy: {:2f}%".format(decision_tree_acc*100))
```

```
Test accuracy: 88.333333%
```

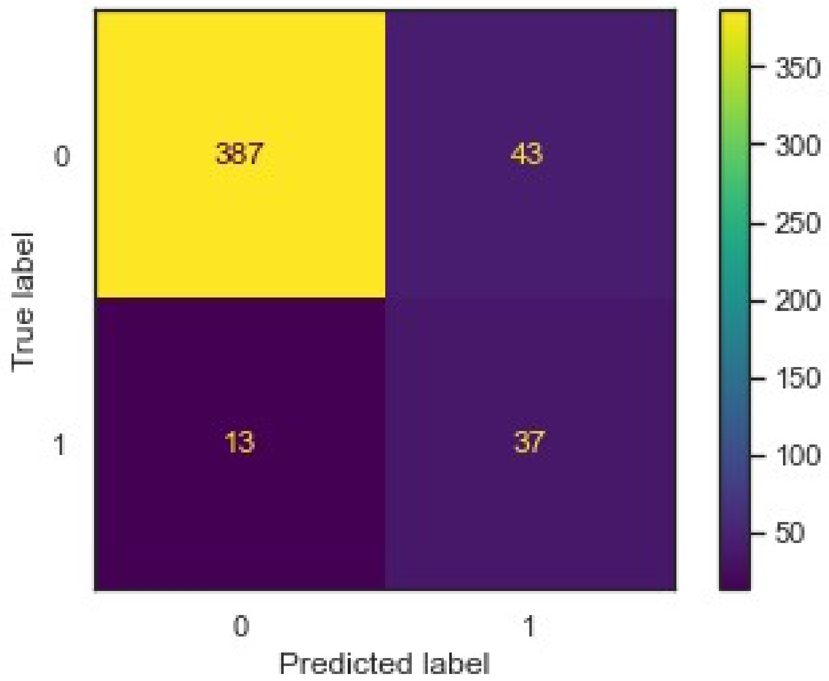
```
print(classification_report(y_test, decision_tree_pred))
plt.savefig('CM1')
```

```
              precision    recall  f1-score   support

           0       0.97      0.90      0.93       430
           1       0.46      0.74      0.57        50

    accuracy                           0.88       480
   macro avg       0.72      0.82      0.75       480
weighted avg       0.91      0.88      0.89       480


<Figure size 432x288 with 0 Axes>
```

```
from sklearn.metrics import confusion_matrix
import random
import itertools
from sklearn.metrics import ConfusionMatrixDisplay
```

```
cm = confusion_matrix(y_test,decision_tree_pred)
disp= ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
print('True Negative: ',cm[0][0])
print('False Negative: ',cm[1][0])
print('True Positive: ',cm[1][1])
print('False Positive: ',cm[0][1])
plt.savefig('CM')
```

```
True Negative:  387
False Negative:  13
True Positive:  37
False Positive:  43
```



```
from sklearn.ensemble import RandomForestClassifier
randforest=RandomForestClassifier()
randforest.fit(x_train, y_train)
randforest_pred=randforest.predict(x_test)
randforest_acc=accuracy_score(randforest_pred,y_test)
print('Test accuracy:{:.2f}%'.format(randforest_acc*100))
```

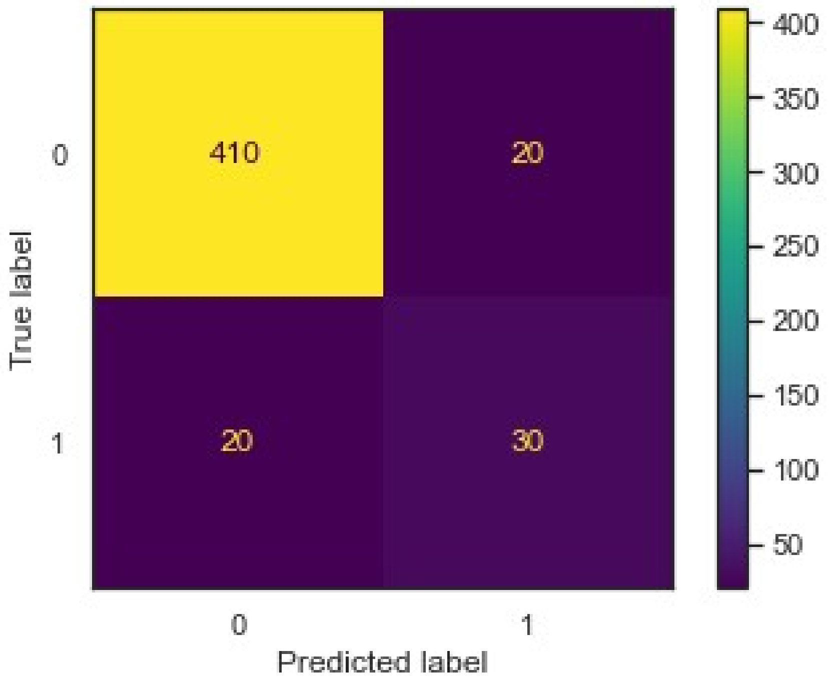
```
Test accuracy:91.67%
```

```
print(classification_report(y_test, randforest_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       430
           1       0.60      0.60      0.60        50

    accuracy                           0.92       480
   macro avg       0.78      0.78      0.78       480
weighted avg       0.92      0.92      0.92       480
```

```
cm = confusion_matrix(y_test,randforest_pred)
disp= ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
print('True Negative: ',cm[0][0])
print('False Negative: ',cm[1][0])
print('True Positive: ',cm[1][1])
print('False Positive: ',cm[0][1])
```

```
True Negative:  410
False Negative:  20
True Positive:  30
False Positive:  20
```

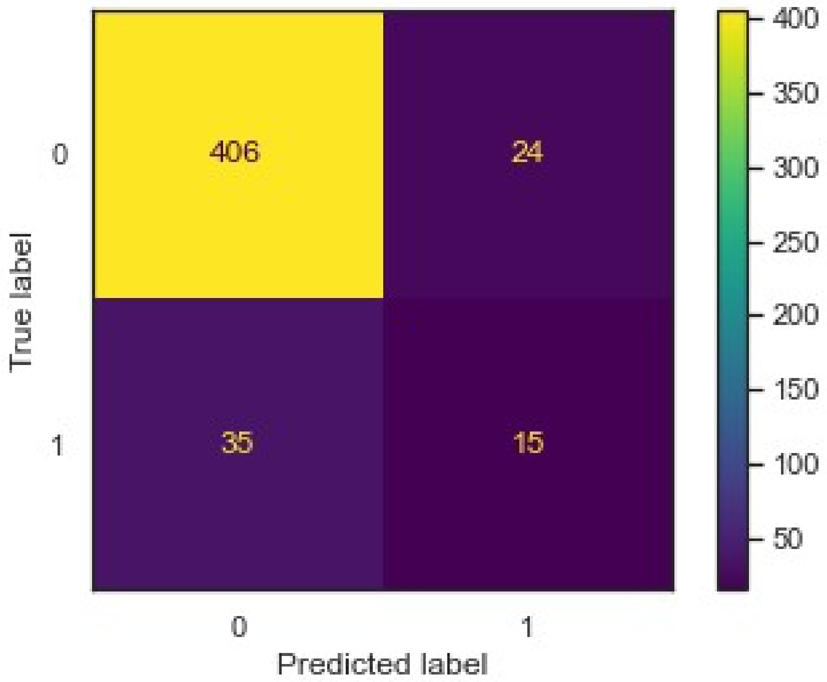


```
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import KNeighborsClassifier
kclassifier=KNeighborsClassifier()
kclassifier.fit(x_train, y_train)
kclassifier_pred=kclassifier.predict(x_test)
kclassifier_acc=accuracy_score(kclassifier_pred,y_test)
print('Test accuracy:{:.2f}%'.format(kclassifier_acc*100))
```

```
Test accuracy:87.71%
```

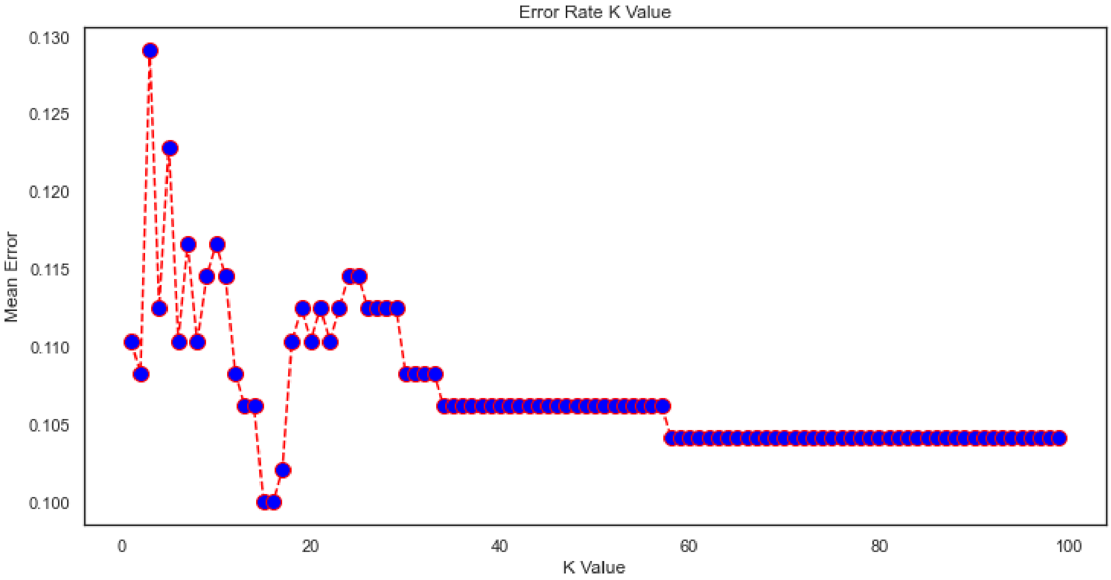
```
cm = confusion_matrix(y_test,kclassifier_pred)
disp= ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
print('True Negative: ',cm[0][0])
print('False Negative: ',cm[1][0])
print('True Positive: ',cm[1][1])
print('False Positive: ',cm[0][1])
```

```
True Negative:  406
False Negative:  35
True Positive:  15
False Positive:  24
```



```
error = []
for i in range(1, 100):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train,y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1,100), error, color='red', linestyle='dashed',
marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

```
Text(0, 0.5, 'Mean Error')
```


Error Rate K Value
Mean Error
0.130
0.125
0.120
0.115
0.110
0.105
0.100
K Value
0
20
40
60
80
100