## \*\* 1. Overview \*\*

Input Variables:

* **fixed acidity:**  most acids involved with wine or fixed or nonvolatile
* **volatile acidity:**  the amount of acetic acid in wine
* **citric acid:**  found in small quantities, citric acid can add 'freshness' and flavor to wines
* **residual sugar:**  the amount of sugar remaining after fermentation stops
* **chlorides:**  the amount of salt in the wine
* **free sulfur dioxide:**  the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion
* **total sulfur dioxide:**  amount of free and bound forms of S02
* **density:**  the density of water is close to that of water depending on the percent alcohol and sugar content
* **pH:**  describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic)
* **sulphates:**  a wine additive which can contribute to sulfur dioxide gas (S02) levels
* **alcohol:**  the percent alcohol content of the wine

Output Variable:

* **quality:**  output variable (based on sensory data, score between 0 and 10)

## \*\* 2. Importing Libraries and Reading the Dataset \*\*

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.metrics import confusion\_matrix  
from collections import Counter  
from IPython.core.display import display, HTML  
sns.set\_style('darkgrid')

dataset = pd.read\_csv('wine\_quality.csv')  
dataset.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

## \*\* 3. Data Visualization and Preprocessing \*\*

dataset.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

bins = (2, 6.5, 8)  
labels = ['bad', 'good']  
dataset['quality'] = pd.cut(x = dataset['quality'], bins = bins, labels = labels)

dataset['quality'].value\_counts()

bad 1382  
good 217  
Name: quality, dtype: int64

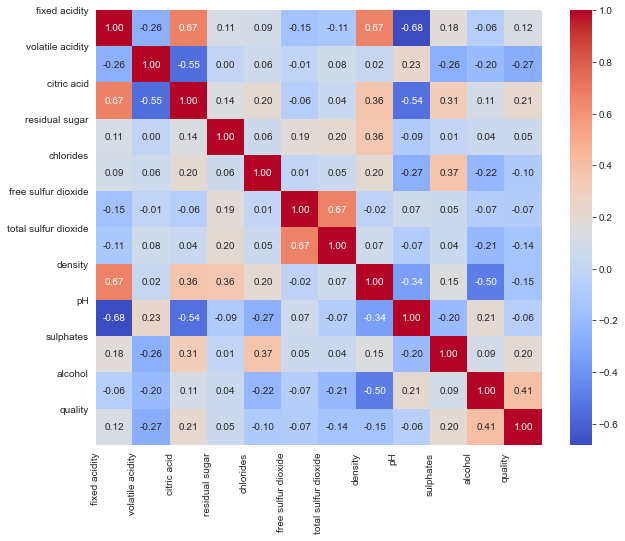
Not bad! I mean the result :)

from sklearn.preprocessing import LabelEncoder  
labelencoder\_y = LabelEncoder()  
dataset['quality'] = labelencoder\_y.fit\_transform(dataset['quality'])

dataset.head()

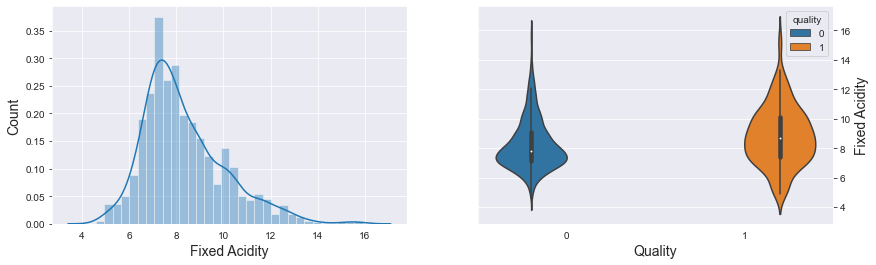
fixed acidity volatile acidity citric acid residual sugar chlorides \  
0 7.4 0.70 0.00 1.9 0.076   
1 7.8 0.88 0.00 2.6 0.098   
2 7.8 0.76 0.04 2.3 0.092   
3 11.2 0.28 0.56 1.9 0.075   
4 7.4 0.70 0.00 1.9 0.076   
  
 free sulfur dioxide total sulfur dioxide density pH sulphates \  
0 11.0 34.0 0.9978 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 0   
1 9.8 0   
2 9.8 0   
3 9.8 0   
4 9.4 0

corr = dataset.corr()  
#Plot figsize  
fig, ax = plt.subplots(figsize=(10, 8))  
#Generate Heat Map, allow annotations and place floats in map  
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt=".2f")  
#Apply xticks  
plt.xticks(range(len(corr.columns)), corr.columns);  
#Apply yticks  
plt.yticks(range(len(corr.columns)), corr.columns)  
#show plot  
plt.show()



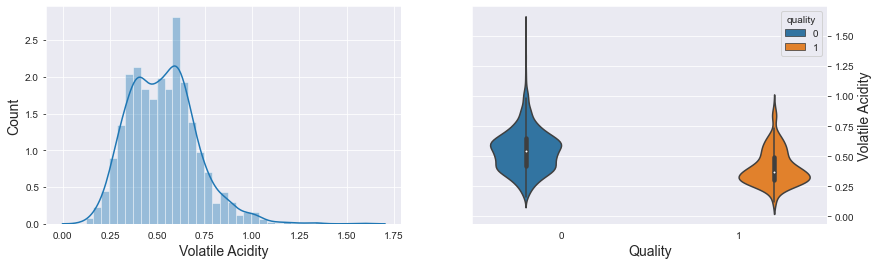
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['fixed acidity'], ax = axes[0])  
axes[0].set\_xlabel('Fixed Acidity', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'fixed acidity', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Fixed Acidity', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



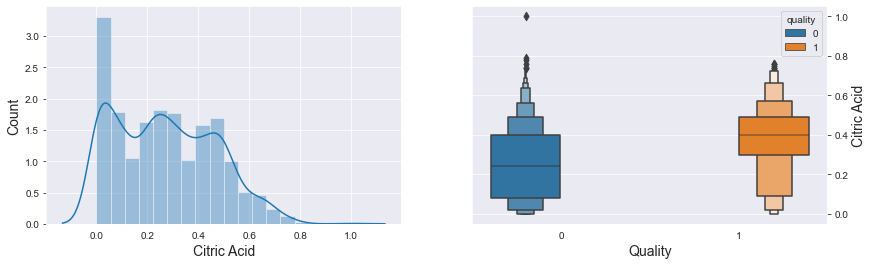
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['volatile acidity'], ax = axes[0])  
axes[0].set\_xlabel('Volatile Acidity', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'volatile acidity', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Volatile Acidity', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



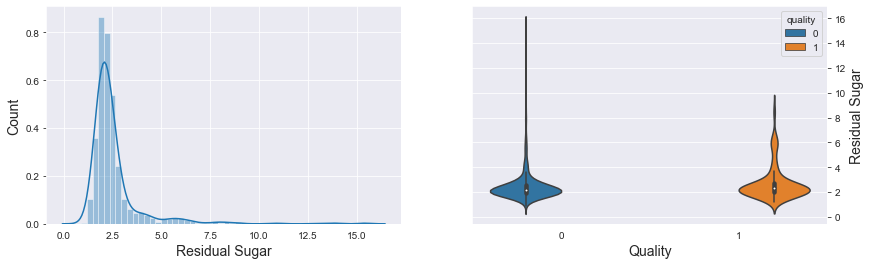
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['citric acid'], ax = axes[0])  
axes[0].set\_xlabel('Citric Acid', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.boxenplot(x = 'quality', y = 'citric acid', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Citric Acid', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



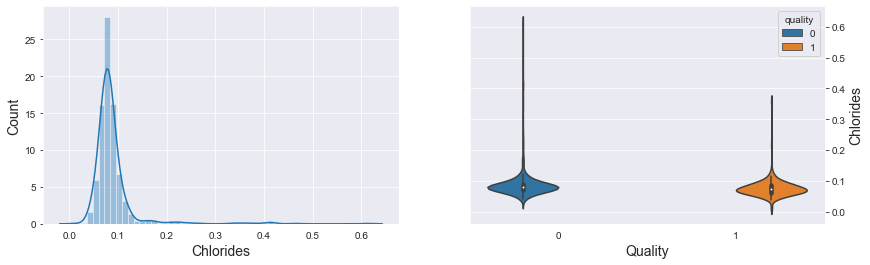
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['residual sugar'], ax = axes[0])  
axes[0].set\_xlabel('Residual Sugar', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'residual sugar', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Residual Sugar', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



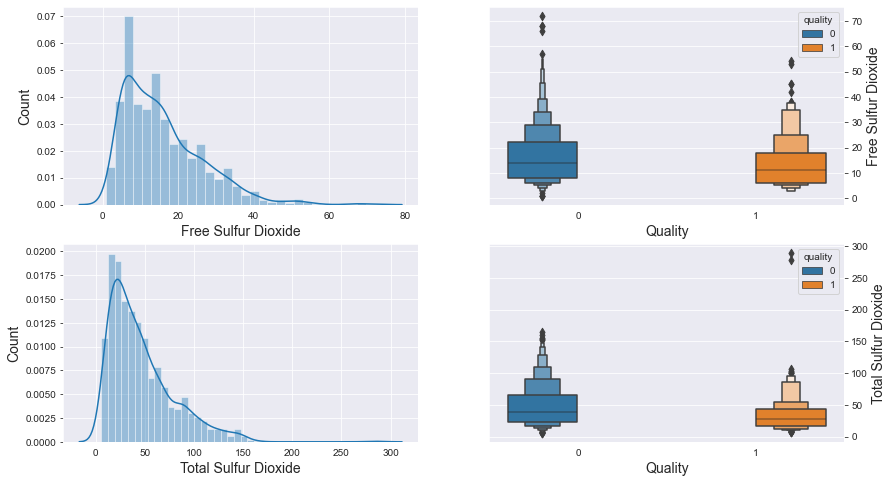
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['chlorides'], ax = axes[0])  
axes[0].set\_xlabel('Chlorides', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'chlorides', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Chlorides', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



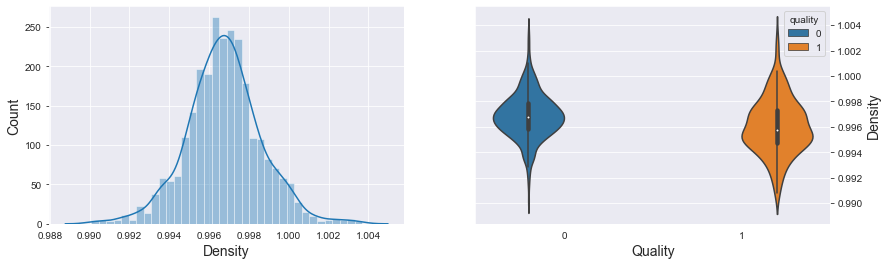
f, axes = plt.subplots(2,2,figsize=(14,8))  
  
sns.distplot(dataset['free sulfur dioxide'], ax = axes[0,0])  
axes[0,0].set\_xlabel('Free Sulfur Dioxide', fontsize=14)  
axes[0,0].set\_ylabel('Count', fontsize=14)  
axes[0,0].yaxis.tick\_left()  
  
sns.boxenplot(x = 'quality', y = 'free sulfur dioxide', data = dataset, hue = 'quality',ax = axes[0,1])  
axes[0,1].set\_xlabel('Quality', fontsize=14)  
axes[0,1].set\_ylabel('Free Sulfur Dioxide', fontsize=14)  
axes[0,1].yaxis.set\_label\_position("right")  
axes[0,1].yaxis.tick\_right()  
  
sns.distplot(dataset['total sulfur dioxide'], ax = axes[1,0])  
axes[1,0].set\_xlabel('Total Sulfur Dioxide', fontsize=14)  
axes[1,0].set\_ylabel('Count', fontsize=14)  
axes[1,0].yaxis.tick\_left()  
  
sns.boxenplot(x = 'quality', y = 'total sulfur dioxide', data = dataset, hue = 'quality',ax = axes[1,1])  
axes[1,1].set\_xlabel('Quality', fontsize=14)  
axes[1,1].set\_ylabel('Total Sulfur Dioxide', fontsize=14)  
axes[1,1].yaxis.set\_label\_position("right")  
axes[1,1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)  
D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



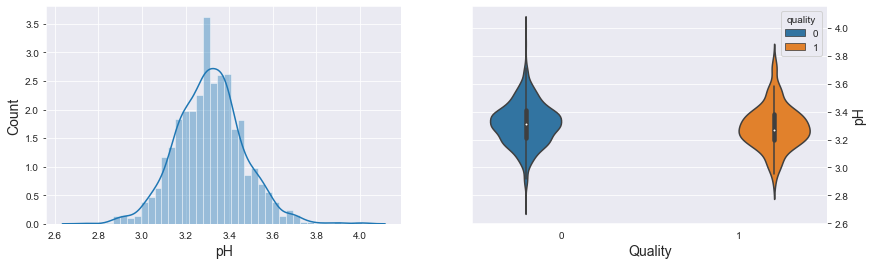
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['density'], ax = axes[0])  
axes[0].set\_xlabel('Density', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'density', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Density', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



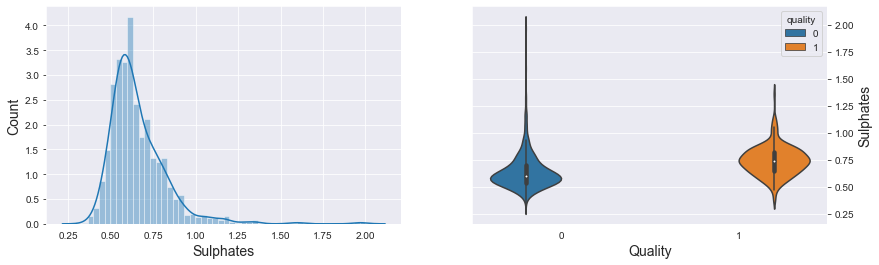
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['pH'], ax = axes[0])  
axes[0].set\_xlabel('pH', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'pH', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('pH', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



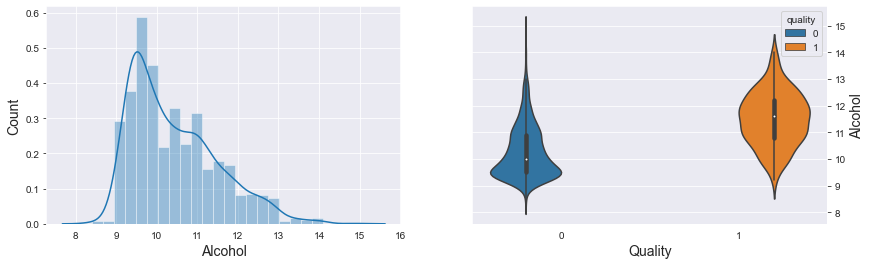
f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['sulphates'], ax = axes[0])  
axes[0].set\_xlabel('Sulphates', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'sulphates', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Sulphates', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



f, axes = plt.subplots(1,2,figsize=(14,4))  
  
sns.distplot(dataset['alcohol'], ax = axes[0])  
axes[0].set\_xlabel('Alcohol', fontsize=14)  
axes[0].set\_ylabel('Count', fontsize=14)  
axes[0].yaxis.tick\_left()  
  
sns.violinplot(x = 'quality', y = 'alcohol', data = dataset, hue = 'quality',ax = axes[1])  
axes[1].set\_xlabel('Quality', fontsize=14)  
axes[1].set\_ylabel('Alcohol', fontsize=14)  
axes[1].yaxis.set\_label\_position("right")  
axes[1].yaxis.tick\_right()  
  
plt.show()

D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)



dataset['quality'].value\_counts()

0 1382  
1 217  
Name: quality, dtype: int64

X = dataset.drop('quality', axis = 1).values  
y = dataset['quality'].values.reshape(-1,1)

# Splitting the dataset into the Training set and Test set  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

print("Shape of X\_train: ",X\_train.shape)  
print("Shape of X\_test: ", X\_test.shape)  
print("Shape of y\_train: ",y\_train.shape)  
print("Shape of y\_test",y\_test.shape)

Shape of X\_train: (1279, 11)  
Shape of X\_test: (320, 11)  
Shape of y\_train: (1279, 1)  
Shape of y\_test (320, 1)

## \*\* 4. Classification Models \*\*

# Feature Scaling  
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X\_train\_scaled = sc.fit\_transform(X\_train)  
X\_test\_scaled = sc.transform(X\_test)

### \*\* Logistic Regression \*\*

# Fitting Logistic Regression to the Training set  
from sklearn.linear\_model import LogisticRegression  
classifier\_lr = LogisticRegression(C=1, fit\_intercept=True, max\_iter=1000, penalty = 'l2', solver='liblinear')  
classifier\_lr.fit(X\_train\_scaled, y\_train.ravel())

LogisticRegression(C=1, max\_iter=1000, solver='liblinear')

# Predicting Cross Validation Score  
cv\_lr = cross\_val\_score(estimator = classifier\_lr, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_lr.mean())  
  
y\_pred\_lr\_train = classifier\_lr.predict(X\_train\_scaled)  
accuracy\_lr\_train = accuracy\_score(y\_train, y\_pred\_lr\_train)  
print("Training set: ", accuracy\_lr\_train)  
  
y\_pred\_lr\_test = classifier\_lr.predict(X\_test\_scaled)  
accuracy\_lr\_test = accuracy\_score(y\_test, y\_pred\_lr\_test)  
print("Test set: ", accuracy\_lr\_test)

CV: 0.885857529527559  
Training set: 0.8858483189992181  
Test set: 0.865625

confusion\_matrix(y\_test, y\_pred\_lr\_test)

array([[264, 9],  
 [ 34, 13]], dtype=int64)

tp\_lr = confusion\_matrix(y\_test, y\_pred\_lr\_test)[0,0]  
fp\_lr = confusion\_matrix(y\_test, y\_pred\_lr\_test)[0,1]  
tn\_lr = confusion\_matrix(y\_test, y\_pred\_lr\_test)[1,1]  
fn\_lr = confusion\_matrix(y\_test, y\_pred\_lr\_test)[1,0]

### \*\* K-Nearest Neighbors (K-NN) \*\*

# Fitting classifier to the Training set  
from sklearn.neighbors import KNeighborsClassifier  
classifier\_knn = KNeighborsClassifier(leaf\_size = 1, metric = 'minkowski', n\_neighbors = 32, weights = 'distance')  
classifier\_knn.fit(X\_train\_scaled, y\_train.ravel())

KNeighborsClassifier(leaf\_size=1, n\_neighbors=32, weights='distance')

# Predicting Cross Validation Score  
cv\_knn = cross\_val\_score(estimator = classifier\_knn, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_knn.mean())  
  
y\_pred\_knn\_train = classifier\_knn.predict(X\_train\_scaled)  
accuracy\_knn\_train = accuracy\_score(y\_train, y\_pred\_knn\_train)  
print("Training set: ", accuracy\_knn\_train)  
  
y\_pred\_knn\_test = classifier\_knn.predict(X\_test\_scaled)  
accuracy\_knn\_test = accuracy\_score(y\_test, y\_pred\_knn\_test)  
print("Test set: ", accuracy\_knn\_test)

CV: 0.9022699311023622  
Training set: 1.0  
Test set: 0.89375

confusion\_matrix(y\_test, y\_pred\_knn\_test)

array([[264, 9],  
 [ 25, 22]], dtype=int64)

tp\_knn = confusion\_matrix(y\_test, y\_pred\_knn\_test)[0,0]  
fp\_knn = confusion\_matrix(y\_test, y\_pred\_knn\_test)[0,1]  
tn\_knn = confusion\_matrix(y\_test, y\_pred\_knn\_test)[1,1]  
fn\_knn = confusion\_matrix(y\_test, y\_pred\_knn\_test)[1,0]

### \*\* Support Vector Machine (SVM - Linear) \*\*

# Fitting classifier to the Training set  
from sklearn.svm import SVC  
classifier\_svm\_linear = SVC(kernel = 'linear')  
classifier\_svm\_linear.fit(X\_train\_scaled, y\_train.ravel())

SVC(kernel='linear')

# Predicting Cross Validation Score  
cv\_svm\_linear = cross\_val\_score(estimator = classifier\_svm\_linear, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_svm\_linear.mean())  
  
y\_pred\_svm\_linear\_train = classifier\_svm\_linear.predict(X\_train\_scaled)  
accuracy\_svm\_linear\_train = accuracy\_score(y\_train, y\_pred\_svm\_linear\_train)  
print("Training set: ", accuracy\_svm\_linear\_train)  
  
y\_pred\_svm\_linear\_test = classifier\_svm\_linear.predict(X\_test\_scaled)  
accuracy\_svm\_linear\_test = accuracy\_score(y\_test, y\_pred\_svm\_linear\_test)  
print("Test set: ", accuracy\_svm\_linear\_test)

CV: 0.8670829232283465  
Training set: 0.8670836591086787  
Test set: 0.853125

confusion\_matrix(y\_test, y\_pred\_svm\_linear\_test)

array([[273, 0],  
 [ 47, 0]], dtype=int64)

tp\_svm\_linear = confusion\_matrix(y\_test, y\_pred\_svm\_linear\_test)[0,0]  
fp\_svm\_linear = confusion\_matrix(y\_test, y\_pred\_svm\_linear\_test)[0,1]  
tn\_svm\_linear = confusion\_matrix(y\_test, y\_pred\_svm\_linear\_test)[1,1]  
fn\_svm\_linear = confusion\_matrix(y\_test, y\_pred\_svm\_linear\_test)[1,0]

### \*\* Support Vector Machine (SVM - Kernel) \*\*

# Fitting classifier to the Training set  
from sklearn.svm import SVC  
classifier\_svm\_kernel = SVC(kernel = 'rbf', C = 10, tol = 0.001, gamma = 'scale')  
classifier\_svm\_kernel.fit(X\_train\_scaled, y\_train.ravel())

SVC(C=10)

# Predicting Cross Validation Score  
cv\_svm\_kernel = cross\_val\_score(estimator = classifier\_svm\_kernel, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_svm\_kernel.mean())  
  
y\_pred\_svm\_kernel\_train = classifier\_svm\_kernel.predict(X\_train\_scaled)  
accuracy\_svm\_kernel\_train = accuracy\_score(y\_train, y\_pred\_svm\_kernel\_train)  
print("Training set: ", accuracy\_svm\_kernel\_train)  
  
y\_pred\_svm\_kernel\_test = classifier\_svm\_kernel.predict(X\_test\_scaled)  
accuracy\_svm\_kernel\_test = accuracy\_score(y\_test, y\_pred\_svm\_kernel\_test)  
print("Test set: ", accuracy\_svm\_kernel\_test)

CV: 0.8999261811023622  
Training set: 0.9421422986708365  
Test set: 0.89375

confusion\_matrix(y\_test, y\_pred\_svm\_kernel\_test)

array([[261, 12],  
 [ 22, 25]], dtype=int64)

tp\_svm\_kernel = confusion\_matrix(y\_test, y\_pred\_svm\_kernel\_test)[0,0]  
fp\_svm\_kernel = confusion\_matrix(y\_test, y\_pred\_svm\_kernel\_test)[0,1]  
tn\_svm\_kernel = confusion\_matrix(y\_test, y\_pred\_svm\_kernel\_test)[1,1]  
fn\_svm\_kernel = confusion\_matrix(y\_test, y\_pred\_svm\_kernel\_test)[1,0]

### \*\* Naive Bayes \*\*

# Fitting classifier to the Training set  
from sklearn.naive\_bayes import GaussianNB  
classifier\_nb = GaussianNB()  
classifier\_nb.fit(X\_train\_scaled, y\_train.ravel())

GaussianNB()

# Predicting Cross Validation Score  
cv\_nb = cross\_val\_score(estimator = classifier\_nb, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_nb.mean())  
  
y\_pred\_nb\_train = classifier\_nb.predict(X\_train\_scaled)  
accuracy\_nb\_train = accuracy\_score(y\_train, y\_pred\_nb\_train)  
print("Training set: ", accuracy\_nb\_train)  
  
y\_pred\_nb\_test = classifier\_nb.predict(X\_test\_scaled)  
accuracy\_nb\_test = accuracy\_score(y\_test, y\_pred\_nb\_test)  
print("Test set: ", accuracy\_nb\_test)

CV: 0.8373462106299213  
Training set: 0.8389366692728695  
Test set: 0.846875

confusion\_matrix(y\_test, y\_pred\_nb\_test)

array([[234, 39],  
 [ 10, 37]], dtype=int64)

tp\_nb = confusion\_matrix(y\_test, y\_pred\_nb\_test)[0,0]  
fp\_nb = confusion\_matrix(y\_test, y\_pred\_nb\_test)[0,1]  
tn\_nb = confusion\_matrix(y\_test, y\_pred\_nb\_test)[1,1]  
fn\_nb = confusion\_matrix(y\_test, y\_pred\_nb\_test)[1,0]

### \*\* Decision Tree Classification \*\*

# Fitting classifier to the Training set  
from sklearn.tree import DecisionTreeClassifier  
classifier\_dt = DecisionTreeClassifier(criterion = 'gini', max\_features=6, max\_leaf\_nodes=400, random\_state = 33)  
classifier\_dt.fit(X\_train\_scaled, y\_train.ravel())

DecisionTreeClassifier(max\_features=6, max\_leaf\_nodes=400, random\_state=33)

# Predicting Cross Validation Score  
cv\_dt = cross\_val\_score(estimator = classifier\_dt, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_dt.mean())  
  
y\_pred\_dt\_train = classifier\_dt.predict(X\_train\_scaled)  
accuracy\_dt\_train = accuracy\_score(y\_train, y\_pred\_dt\_train)  
print("Training set: ", accuracy\_dt\_train)  
  
y\_pred\_dt\_test = classifier\_dt.predict(X\_test\_scaled)  
accuracy\_dt\_test = accuracy\_score(y\_test, y\_pred\_dt\_test)  
print("Test set: ", accuracy\_dt\_test)

CV: 0.8960014763779528  
Training set: 1.0  
Test set: 0.878125

confusion\_matrix(y\_test, y\_pred\_dt\_test)

array([[252, 21],  
 [ 18, 29]], dtype=int64)

tp\_dt = confusion\_matrix(y\_test, y\_pred\_dt\_test)[0,0]  
fp\_dt = confusion\_matrix(y\_test, y\_pred\_dt\_test)[0,1]  
tn\_dt = confusion\_matrix(y\_test, y\_pred\_dt\_test)[1,1]  
fn\_dt = confusion\_matrix(y\_test, y\_pred\_dt\_test)[1,0]

### \*\* Random Forest Classification \*\*

# Fitting Random Forest Classification to the Training set  
from sklearn.ensemble import RandomForestClassifier  
classifier\_rf = RandomForestClassifier(criterion = 'entropy', max\_features = 4, n\_estimators = 800, random\_state=33)  
classifier\_rf.fit(X\_train\_scaled, y\_train.ravel())

RandomForestClassifier(criterion='entropy', max\_features=4, n\_estimators=800,  
 random\_state=33)

# Predicting Cross Validation Score  
cv\_rf = cross\_val\_score(estimator = classifier\_rf, X = X\_train\_scaled, y = y\_train.ravel(), cv = 10)  
print("CV: ", cv\_rf.mean())  
  
y\_pred\_rf\_train = classifier\_rf.predict(X\_train\_scaled)  
accuracy\_rf\_train = accuracy\_score(y\_train, y\_pred\_rf\_train)  
print("Training set: ", accuracy\_rf\_train)  
  
y\_pred\_rf\_test = classifier\_rf.predict(X\_test\_scaled)  
accuracy\_rf\_test = accuracy\_score(y\_test, y\_pred\_rf\_test)  
print("Test set: ", accuracy\_rf\_test)

CV: 0.9140194389763779  
Training set: 1.0  
Test set: 0.9125

confusion\_matrix(y\_test, y\_pred\_rf\_test)

array([[267, 6],  
 [ 22, 25]], dtype=int64)

tp\_rf = confusion\_matrix(y\_test, y\_pred\_rf\_test)[0,0]  
fp\_rf = confusion\_matrix(y\_test, y\_pred\_rf\_test)[0,1]  
tn\_rf = confusion\_matrix(y\_test, y\_pred\_rf\_test)[1,1]  
fn\_rf = confusion\_matrix(y\_test, y\_pred\_rf\_test)[1,0]

## \*\* 6. Measuring The Error \*\*

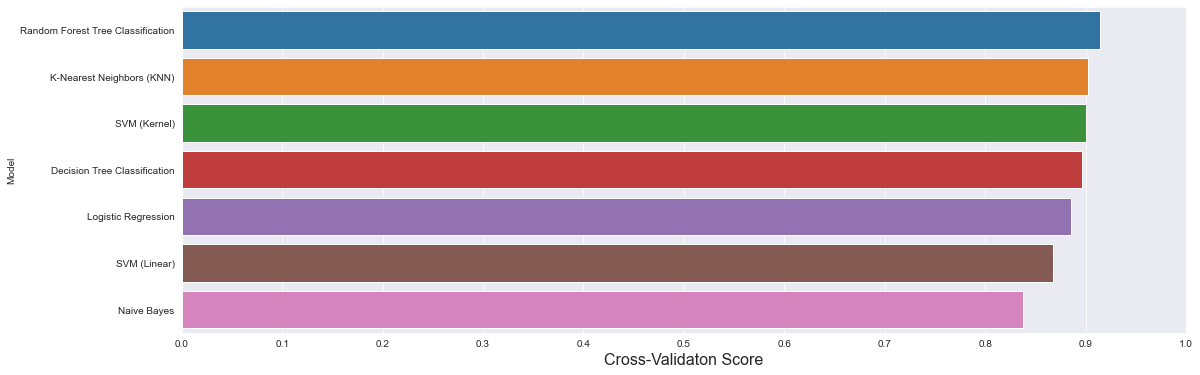
models = [('Logistic Regression', tp\_lr, fp\_lr, tn\_lr, fn\_lr, accuracy\_lr\_train, accuracy\_lr\_test, cv\_lr.mean()),  
 ('K-Nearest Neighbors (KNN)', tp\_knn, fp\_knn, tn\_knn, fn\_knn, accuracy\_knn\_train, accuracy\_knn\_test, cv\_knn.mean()),  
 ('SVM (Linear)', tp\_svm\_linear, fp\_svm\_linear, tn\_svm\_linear, fn\_svm\_linear, accuracy\_svm\_linear\_train, accuracy\_svm\_linear\_test, cv\_svm\_linear.mean()),  
 ('SVM (Kernel)', tp\_svm\_kernel, fp\_svm\_kernel, tn\_svm\_kernel, fn\_svm\_kernel, accuracy\_svm\_kernel\_train, accuracy\_svm\_kernel\_test, cv\_svm\_kernel.mean()),  
 ('Naive Bayes', tp\_nb, fp\_nb, tn\_nb, fn\_nb, accuracy\_nb\_train, accuracy\_nb\_test, cv\_nb.mean()),  
 ('Decision Tree Classification', tp\_dt, fp\_dt, tn\_dt, fn\_dt, accuracy\_dt\_train, accuracy\_dt\_test, cv\_dt.mean()),  
 ('Random Forest Tree Classification', tp\_rf, fp\_rf, tn\_rf, fn\_rf, accuracy\_rf\_train, accuracy\_rf\_test, cv\_rf.mean())  
 ]

predict = pd.DataFrame(data = models, columns=['Model', 'True Positive', 'False Positive', 'True Negative',  
 'False Negative', 'Accuracy(training)', 'Accuracy(test)',  
 'Cross-Validation'])  
predict

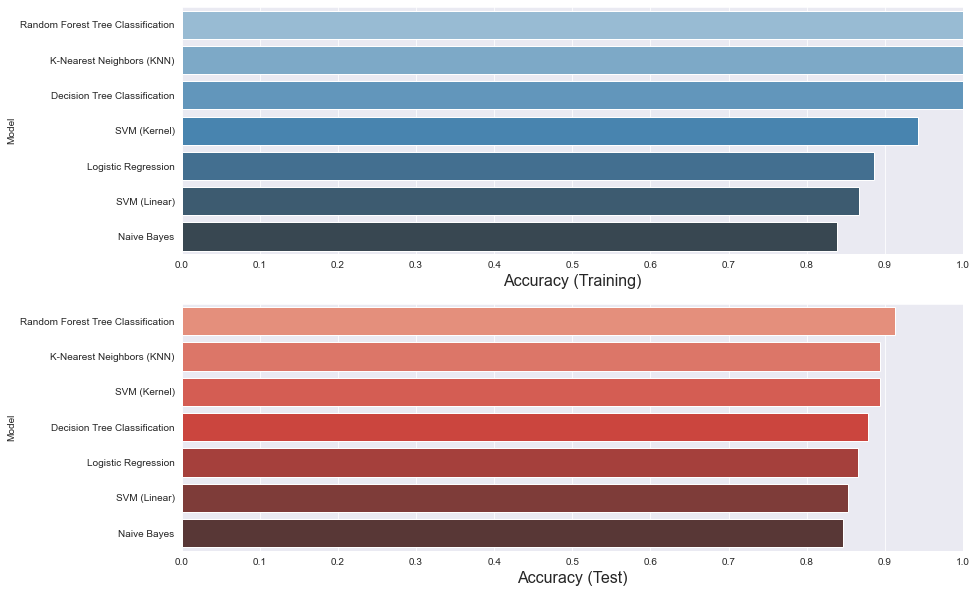
Model True Positive False Positive \  
0 Logistic Regression 264 9   
1 K-Nearest Neighbors (KNN) 264 9   
2 SVM (Linear) 273 0   
3 SVM (Kernel) 261 12   
4 Naive Bayes 234 39   
5 Decision Tree Classification 252 21   
6 Random Forest Tree Classification 267 6   
  
 True Negative False Negative Accuracy(training) Accuracy(test) \  
0 13 34 0.885848 0.865625   
1 22 25 1.000000 0.893750   
2 0 47 0.867084 0.853125   
3 25 22 0.942142 0.893750   
4 37 10 0.838937 0.846875   
5 29 18 1.000000 0.878125   
6 25 22 1.000000 0.912500   
  
 Cross-Validation   
0 0.885858   
1 0.902270   
2 0.867083   
3 0.899926   
4 0.837346   
5 0.896001   
6 0.914019

### \*\* Visualizing Models Performance \*\*

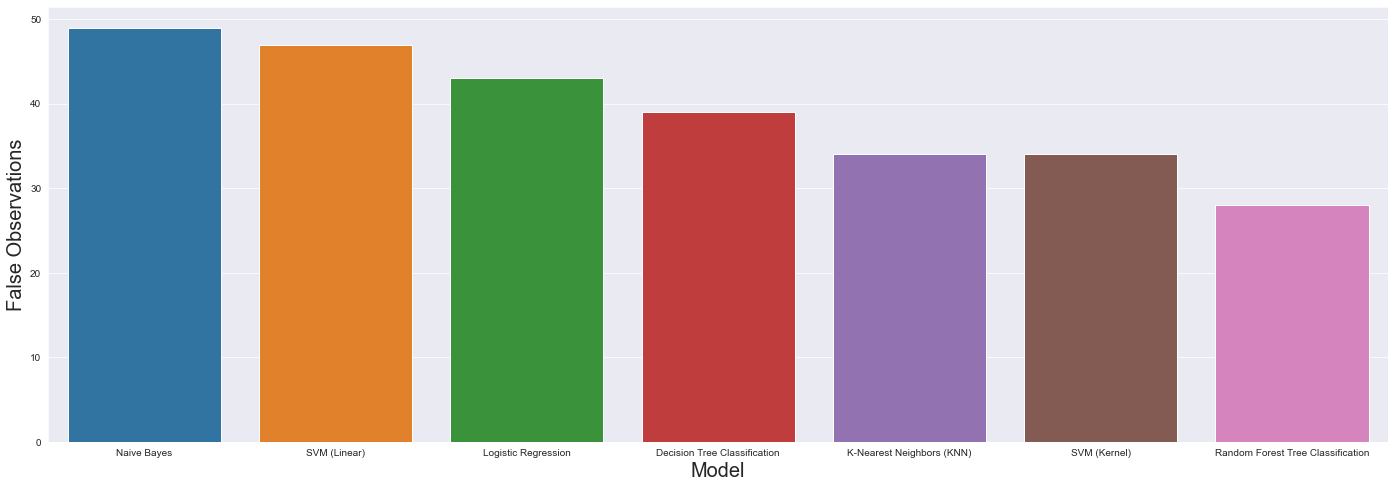
f, axe = plt.subplots(1,1, figsize=(18,6))  
  
predict.sort\_values(by=['Cross-Validation'], ascending=False, inplace=True)  
  
sns.barplot(x='Cross-Validation', y='Model', data = predict, ax = axe)  
#axes[0].set(xlabel='Region', ylabel='Charges')  
axe.set\_xlabel('Cross-Validaton Score', size=16)  
axe.set\_ylabel('Model')  
axe.set\_xlim(0,1.0)  
axe.set\_xticks(np.arange(0, 1.1, 0.1))  
plt.show()



f, axes = plt.subplots(2,1, figsize=(14,10))  
  
predict.sort\_values(by=['Accuracy(training)'], ascending=False, inplace=True)  
  
sns.barplot(x='Accuracy(training)', y='Model', data = predict, palette='Blues\_d', ax = axes[0])  
#axes[0].set(xlabel='Region', ylabel='Charges')  
axes[0].set\_xlabel('Accuracy (Training)', size=16)  
axes[0].set\_ylabel('Model')  
axes[0].set\_xlim(0,1.0)  
axes[0].set\_xticks(np.arange(0, 1.1, 0.1))  
  
predict.sort\_values(by=['Accuracy(test)'], ascending=False, inplace=True)  
  
sns.barplot(x='Accuracy(test)', y='Model', data = predict, palette='Reds\_d', ax = axes[1])  
#axes[0].set(xlabel='Region', ylabel='Charges')  
axes[1].set\_xlabel('Accuracy (Test)', size=16)  
axes[1].set\_ylabel('Model')  
axes[1].set\_xlim(0,1.0)  
axes[1].set\_xticks(np.arange(0, 1.1, 0.1))  
  
plt.show()



predict.sort\_values(by=(['Accuracy(test)']), ascending=True, inplace=True)  
  
f, axe = plt.subplots(1,1, figsize=(24,8))  
sns.barplot(x = predict['Model'], y=predict['False Positive'] + predict['False Negative'], ax = axe)  
axe.set\_xlabel('Model', size=20)  
axe.set\_ylabel('False Observations', size=20)  
  
plt.show()



y\_pred\_rf\_test

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,  
 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,  
 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,  
 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,  
 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,  
 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

#Let us take a random testcase and predict using random forests

classifier\_rf.predict([[7.4,0.76,0,4.4,0.106,15,67,0.99,3.7,0.5,9.3]])

array([0])

#This test case indicates the wine is of bad quality

# The End