Wine Quality Prediction - Capstone Project

Submitted by Rajalakshmi

Context:

Wine sales is a million dollar business as social drinking is on rise. Wine quality ratings are important to build trust in customers, brand reputation and profitable sales. Hence, industry players are using product quality certifications to promote their products. But it is a time-consuming process and requires the assessment given by human experts, which makes this process very expensive. Machine learning model can be of great help in this complex assessment.

Data Description

-> Fixed acidity: are non-volatile acids that do not evaporate readily

-> Volatile acidity: are high acetic acid in wine which leads to an unpleasant vinegar taste

-> Citric Acid: acts as a preservative to increase acidity (small quantities add freshness and flavor to wines)

-> Residual Sugar: Remaining sugar after fermentation stops

-> Chlorides: the amount of salt in the wine

-> Free Sulfur Dioxide: it prevents microbial growth and the oxidation of wine

-> Total Sulfur Dioxide: is the amount of free + bound forms of SO2

-> Density: sweeter wines have a higher density

-> pH: the level of acidity

-> Sulphates: a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant

-> Alcohol: the amount of alcohol in wine

-> Quality : Target variable 1-10

Import Libraries

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn import svm
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
```

Read data

QualityPrediction csv file contains information on different parameters of wine. Let's read this file using pandas.

In [3]:
```
df = pd.read_csv('QualityPrediction.csv')
```

In [4]:
```
df.head()
```

Out[4]:

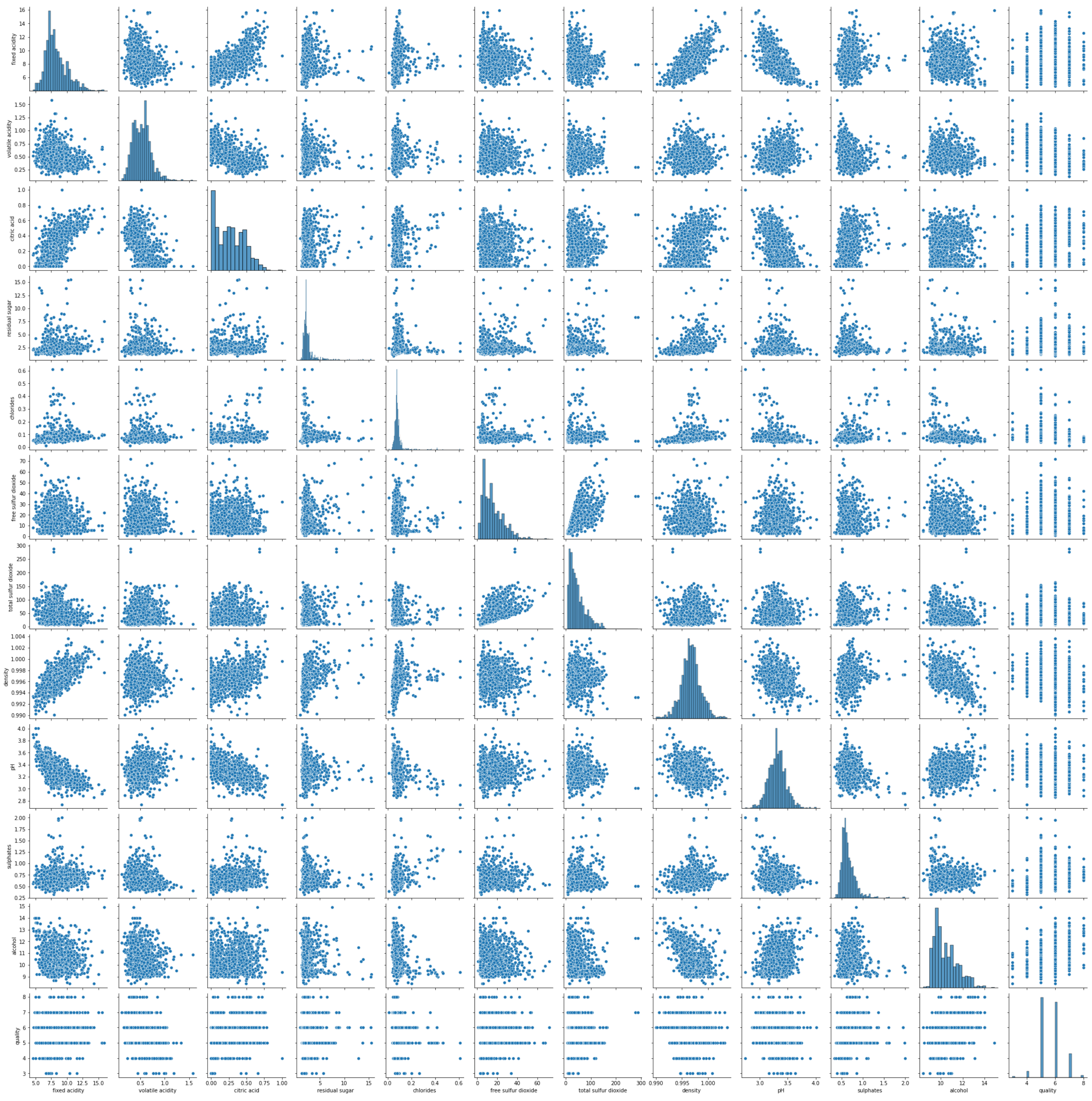
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

Data Exploration

Pairplot

In [5]:
```
sns.pairplot(data=df, kind='scatter')
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x21b315e1220>
```

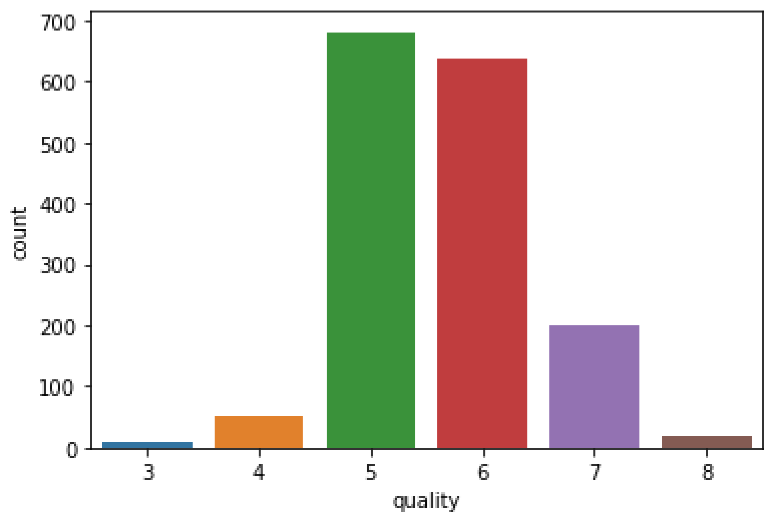


Countplot

```
In [6]: sns.countplot(df['quality'])
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the fol
lowing variable as a keyword arg: x. From version 0.12, the only valid positional ar
gument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

```
Out[6]: <AxesSubplot:xlabel='quality', ylabel='count'>
```



We can infer that in the data that is collected most of the data lies in the quality 5-6. So most of the red wine data collected are in the medium quality.

Correlation with target variable

In [7]:
```
# There are only numerical variables.
# The correlation of each feature with our target variable - quality
correlations = df.corr()['quality'].drop('quality')
print("The correlation of each feature with our target variable - quality \n")
print(correlations)
```

```
The correlation of each feature with our target variable - quality

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

Correlation of independent variables:

In order of highest correlation with quality, these variables are:

-> Alcohol: the amount of alcohol in wine

-> Volatile acidity: are high acetic acid in wine which leads to an unpleasant vinegar taste

-> Sulphates: a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant

-> Citric Acid: acts as a preservative to increase acidity (small quantities add freshness and flavor to wines)

-> Total Sulfur Dioxide: is the amount of free + bound forms of SO2

-> Density: sweeter wines have a higher density

-> Chlorides: the amount of salt in the wine

-> Fixed acidity: are non-volatile acids that do not evaporate readily

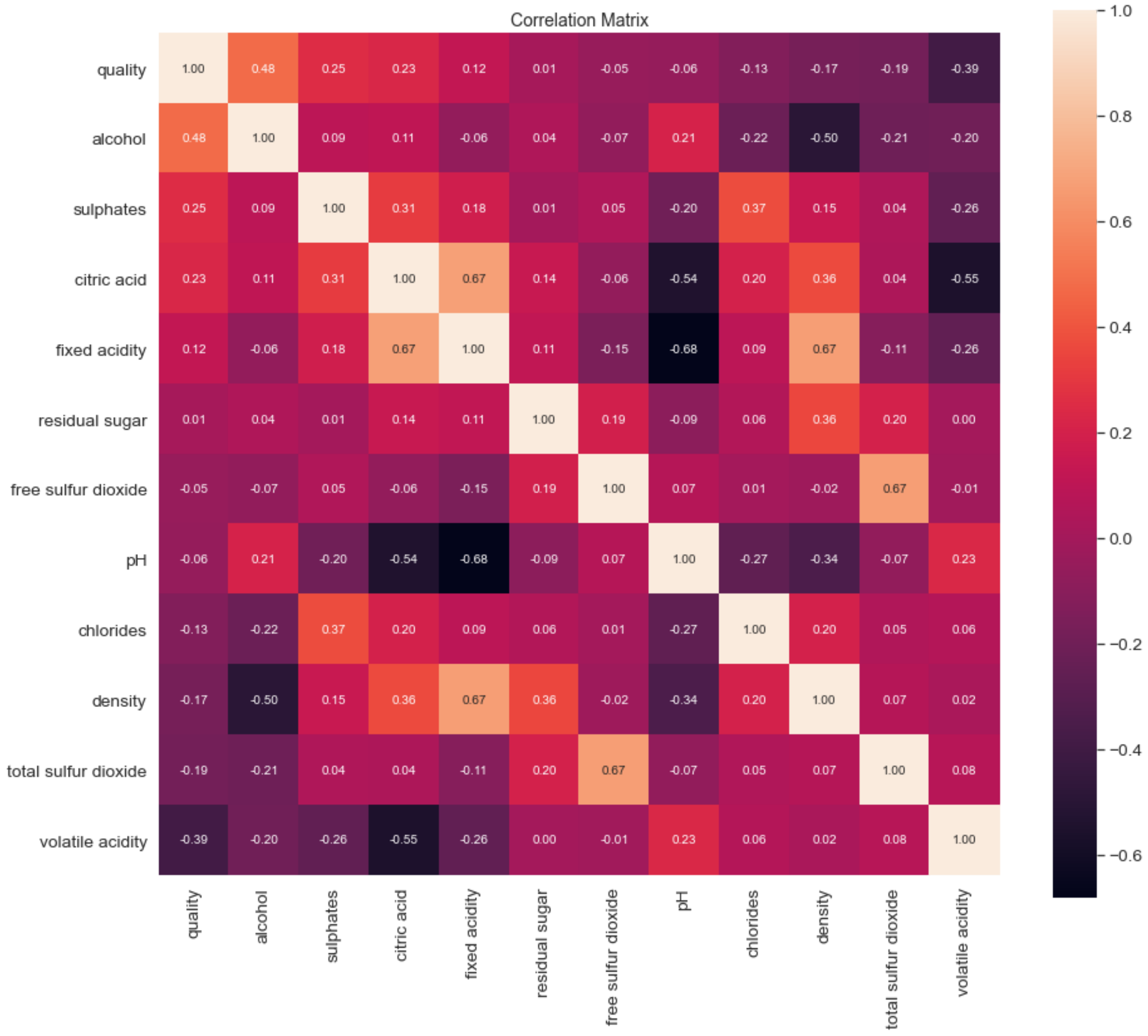
-> pH: the level of acidity

-> Free Sulfur Dioxide: it prevents microbial growth and the oxidation of wine

-> Residual sugar: is the amount of sugar remaining after fermentation stops. The key is to have a perfect balance between — sweetness and sourness (wines > 45g/ltrs are sweet)

Correlation matrix

In [8]:
```
f = plt.figure(figsize=(15, 13)) # setting figure size
corrmat = df.corr()
k = 15 #number of variables for heatmap
cols = corrmat.nlargest(k, 'quality')['quality'].index
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size
                 yticklabels=cols.values, xticklabels=cols.values)
plt.title('Correlation Matrix', fontsize=14);
plt.show()
```



Data Preparation

Checking Null Values

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

There aren't any null values in the dataset.

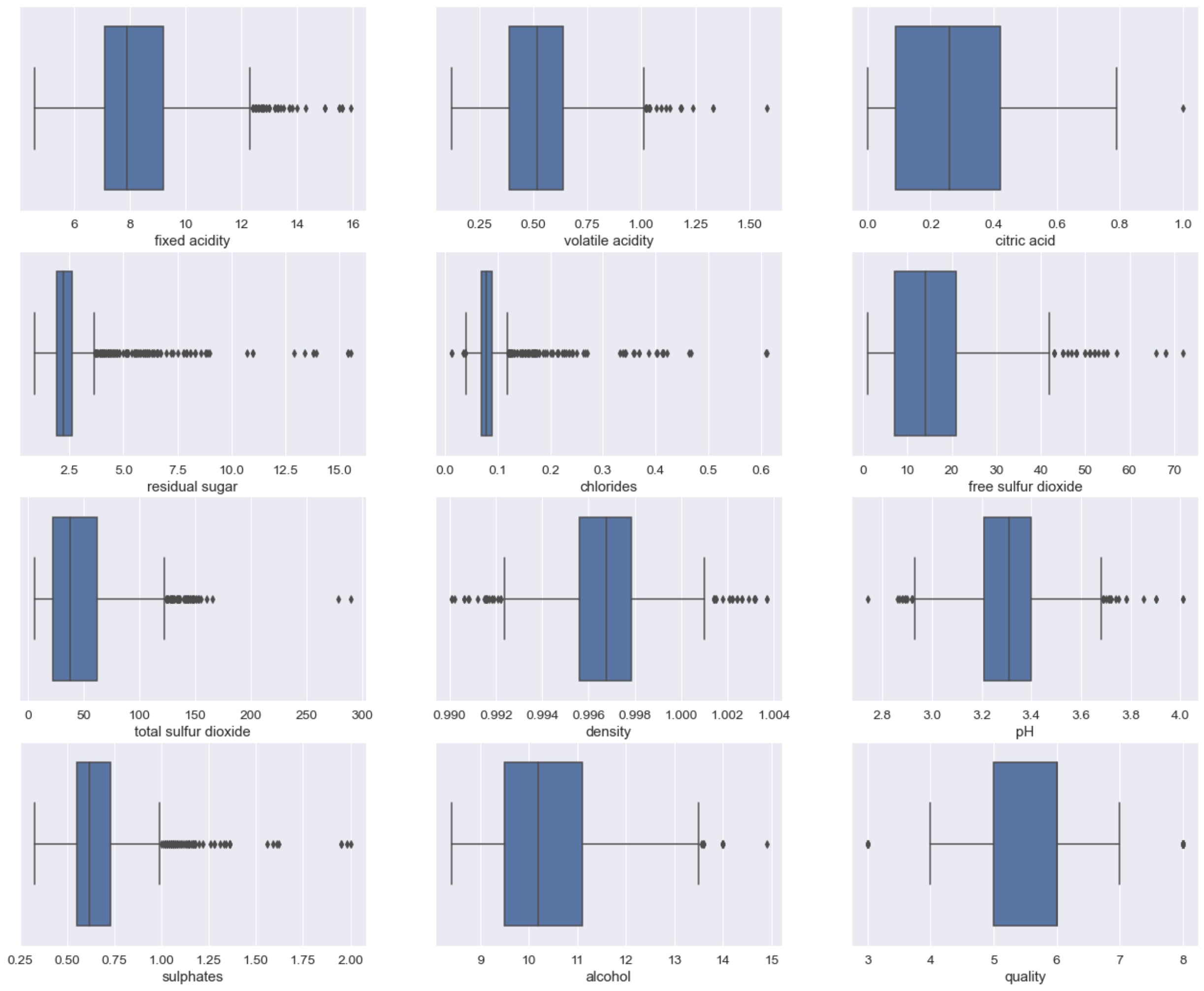
Checking Outliers

In [10]:
```
df.describe()
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000

In [11]:
```
fig, ax = plt.subplots(4, 3, figsize=(22, 18))
for var, subplot in zip(df.columns, ax.flatten()):
    sns.boxplot(x=var, data=df, ax=subplot)
```



Except for citric acid,alcohol and quantity all the other columns have significantly more outliers. Let's try removing them.

Removing Outlier

In [12]:
```
#Selecting the columns which has outliers
df_without_outlier = df[['fixed acidity', 'volatile acidity', 'residual sugar', 'chl
                'total sulfur dioxide', 'density', 'pH', 'sulphates']]

#Removing Outliers for these columns
from scipy import stats
z = np.abs(stats.zscore(df))
df_without_outlier=df_without_outlier[((z > -3)&(z < 3)).all(axis=1)]

print("Shape of data with outlier & without outlier are :",df.shape[0],"and",df_with
```

```
Shape of data with outlier & without outlier are : 1599 and 1451
```

Outliers have been removed from nine columns (namely 'fixed acidity', 'volatile acidity', 'residual sugar', 'chlorides', 'free sulfur dioxide','total sulfur dioxide', 'density', 'pH', 'sulphates') in the dataset which had more outliers.

Now the remaining three columns namely citric acid,alcohol and quantity are added to the new dataset.

In [13]:
```
# Adding three columns namely citric acid,alcohol and quantity to the new dataset.
df_without_outlier['alcohol']= df['alcohol']
df_without_outlier['quality']= df['quality']
df_without_outlier.insert(2, "citric acid", df['citric acid'], True)

df_without_outlier.head()
```

Out[13]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

Separate feature variables and target variable

In [14]:
```
# Separate feature variables and target variable
X = df_without_outlier.drop(['quality'], axis = 1)
y = df_without_outlier['quality']
```

Standardizing the data

In [15]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_features = X
X = scaler.fit_transform(X)
```

Machine Learning Model

Split Data into Train and Test data sets

We are splitting the train and test data in 80:20 ratio.

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20, random_stat

# y_train = y_train.values.ravel()
# y_test = y_test.values.ravel()
```

Linear Regression

Fitting to train data

In [17]:
```
# fitting linear regression to training data
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

Out[17]:
```
LinearRegression()
```

Predicting to train data

In [18]:
```
train_pred = regressor.predict(X_train)
test_pred = regressor.predict(X_test)
```

R2 Score

In [19]:
```
print('R2-score is %s'%regressor.score(X_test,y_test))
```

```
R2-score is 0.2545143857419855
```

```
In [20]: from sklearn.metrics import r2_score
         r2 = r2_score(y_test,test_pred)
         print(r2)

0.2545143857419855
```

Cross Validation

```
In [21]: scores = cross_val_score(regressor, X_train, y_train, scoring='r2', cv=10)
         print(scores.mean())

0.37931232360229705
```

Coeffecient

```
In [22]: feature_colns=df.columns.drop(['quality'])
         coeffecients = pd.DataFrame(regressor.coef_,feature_colns)
         coeffecients.columns = ['Coeffecient']
         print(coeffecients)

                      Coeffecient
fixed acidity            0.032361
volatile acidity        -0.140061
citric acid             -0.039090
residual sugar          -0.002426
chlorides               -0.024625
free sulfur dioxide      0.036346
total sulfur dioxide    -0.103264
density                 -0.036380
pH                      -0.080410
sulphates                0.183259
alcohol                  0.315323
```

Decision Tree Regression

Fitting to train data

```
In [23]: # fitting random foreest regression to training data
         regressor = DecisionTreeRegressor(random_state = 1)
         regressor.fit(X_train, y_train)

Out[23]: DecisionTreeRegressor(random_state=1)
```

Hyperparameters Tuning Using GridSearchCV

```
In [24]: np.random.seed(1)
         param_dist = {'max_depth':[2,3,4,5,6],
                       'max_features':['auto','sqrt','log2',None]}

         cv_rf = GridSearchCV(regressor,cv=10,param_grid=param_dist,n_jobs=3)
         cv_rf.fit(X_train,y_train)
         print('Best Parameters using Grid search: \n',cv_rf.best_params_)

Best Parameters using Grid search:
 {'max_depth': 3, 'max_features': 'auto'}
```

```
In [25]: regressor.set_params(max_features = 'auto',max_depth = 3)

Out[25]: DecisionTreeRegressor(max_depth=3, max_features='auto', random_state=1)
```

Prediting to train data

```
In [26]: train_pred = regressor.predict(X_train)
         test_pred = regressor.predict(X_test)
```

R2 Score

```
In [27]: print('R2-score is %s'%regressor.score(X_test,y_test))
```

```
R2-score is 0.040068075807874664
```

```
In [28]: from sklearn.metrics import r2_score
         r2 = r2_score(y_test,test_pred)
         print(r2)
```

```
0.040068075807874664
```

Feature Importance

```
In [29]: feature_colns=df.columns.drop(['quality'])
         coeffecients = pd.DataFrame(regressor.feature_importances_,feature_colns)
         coeffecients.columns = ['Feature Importance']
         print(coeffecients)
```

```
                      Feature Importance
fixed acidity                   0.044973
volatile acidity                0.086014
citric acid                     0.075822
residual sugar                  0.043225
chlorides                       0.051761
free sulfur dioxide             0.027100
total sulfur dioxide            0.056284
density                         0.041412
pH                              0.082804
sulphates                       0.136690
alcohol                         0.353913
```

Cross Validation

```
In [30]: scores = cross_val_score(regressor, X_train, y_train, scoring='r2', cv=10)
         print(scores.mean())
```

```
0.2681232062611908
```

Random Forest Regression

Fitting to train data

```
In [31]: # fitting random foreest regression to training data
         regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
         regressor.fit(X_train, y_train)
```

```
Out[31]: RandomForestRegressor(n_estimators=10, random_state=0)
```

Hyperparameters Tuning Using GridSearchCV

```
In [32]: np.random.seed(1)
         param_dist = {'max_depth':[2,3,4,5,6],
                       'max_features':['auto','sqrt','log2',None]}

         cv_rf = GridSearchCV(regressor,cv=10,param_grid=param_dist,n_jobs=3)
         cv_rf.fit(X_train,y_train)
         print('Best Parameters using Grid search: \n',cv_rf.best_params_)
```

```
Best Parameters using Grid search: 
 {'max_depth': 6, 'max_features': 'auto'}
```

```
In [33]: regressor.set_params(max_features = 'auto',max_depth = 6)
```

```
Out[33]: RandomForestRegressor(max_depth=6, n_estimators=10, random_state=0)
```

Prediting to train data

```
In [34]: train_pred = regressor.predict(X_train)
         test_pred = regressor.predict(X_test)
```

R2 Score

```
In [35]: print('R2-score is %s'%regressor.score(X_test,y_test))
```

```
R2-score is 0.3121506912636045
```

```
In [36]: from sklearn.metrics import r2_score
         r2 = r2_score(y_test,test_pred)
         print(r2)
```

```
0.3121506912636045
```

Feature Importance

```
In [37]: feature_colns=df.columns.drop(['quality'])
         coeffecients = pd.DataFrame(regressor.feature_importances_,feature_colns)
         coeffecients.columns = ['Feature Importance']
         print(coeffecients)
```

```
                      Feature Importance
fixed acidity                   0.037448
volatile acidity                0.090049
citric acid                     0.059106
residual sugar                  0.057132
chlorides                       0.052428
free sulfur dioxide             0.035871
total sulfur dioxide            0.081828
density                         0.048270
pH                              0.065020
sulphates                       0.151995
alcohol                         0.320853
```

Cross Validation

```
In [38]: scores = cross_val_score(regressor, X_train, y_train, scoring='r2', cv=10)
         print(scores.mean())
```

```
0.40685911692484444
```

```
In [39]: with_target_data_set = df.iloc[:,df.columns == 'quality']
         with_target_data_set = with_target_data_set.values.ravel()

         without_target_data_set = df.iloc[:, df.columns != 'quality']

         predictions_date_sets = regressor.predict(without_target_data_set)
```

```
In [42]: df['Predicted_values']=np.where(with_target_data_set,abs(predictions_date_sets),with
```