**Red Wine Quality Prediction**

**Problem**

The dataset is related to red and white variants of the Portuguese "Vinho Verde" wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

This dataset can be viewed as a classification task. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods. Here, we are trying to predict the wine quality by using the input features

The quality of the wine is depended on the following 11 features.

FEATURES:

* Fixed acidity
* Volatile acidity
* Citric acid
* Residual sugar
* Chlorides
* Free sulfur dioxide
* Total sulfur dioxide
* Density
* pH
* Sulfates
* Alcohol

**Data Analysis and EDA**

There are 11 features that determine the wine quality in our dataset. All these dependent variables are continuous. The output is the wine quality which ranges from 3 to 8. The output variable or dependent variable is a categorical variable. So we will be using classification ML algorithms to make the prediction. We need to import necessary python libraries like NumPy, Pandas, Seaborn, Matplotlib etc to perform data analysis. First, we have to read the data using Pandas.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

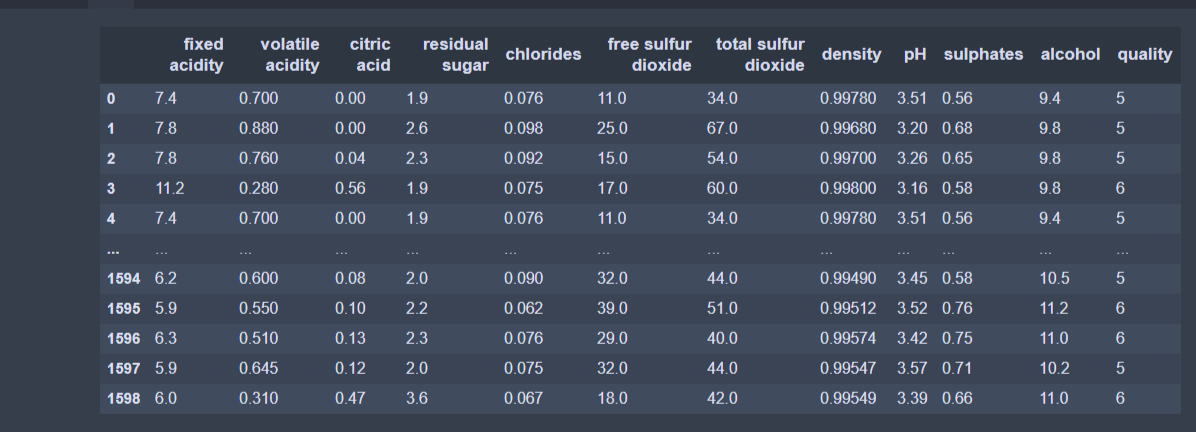
from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

import warnings

warnings.filterwarnings('ignore')

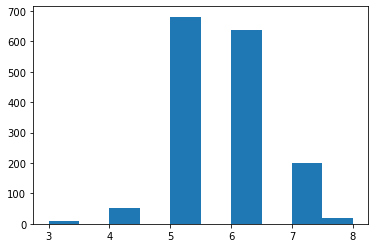
wine\_data = pd.read\_csv("<https://raw.githubusercontent.com/sandespalakat/Datascience/main/Wine-Quality/Data/wine-quality-red.csv>")



There are 1599 records in the dataset. Now we add an extra column that indicates the wine is of good quality or not. If the quality is above 7, that wine is considered as good quality otherwise, it is not good.

red\_wine["goodquality"] = np.where(red\_wine["quality"] >= 7,1,0)

Now we are checking the plot of wine quality.

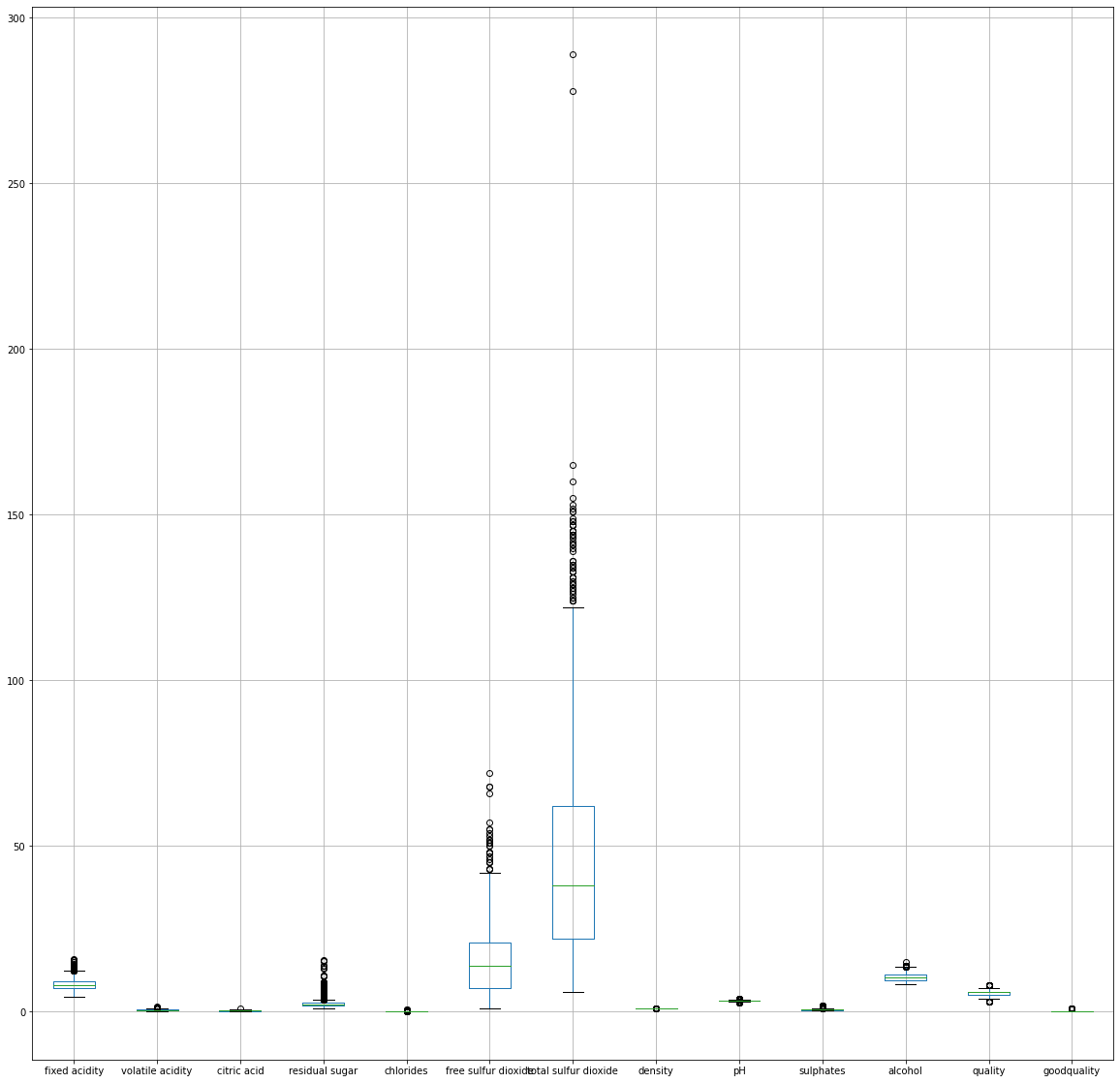


From the graph, we can see that there are wine qualities between 3 and 8. Most of the wines have a quality of 5 and 6. High quality and low-quality wine are very less. Now we can check the good quality wine. There is only about 15% of good quality wine.



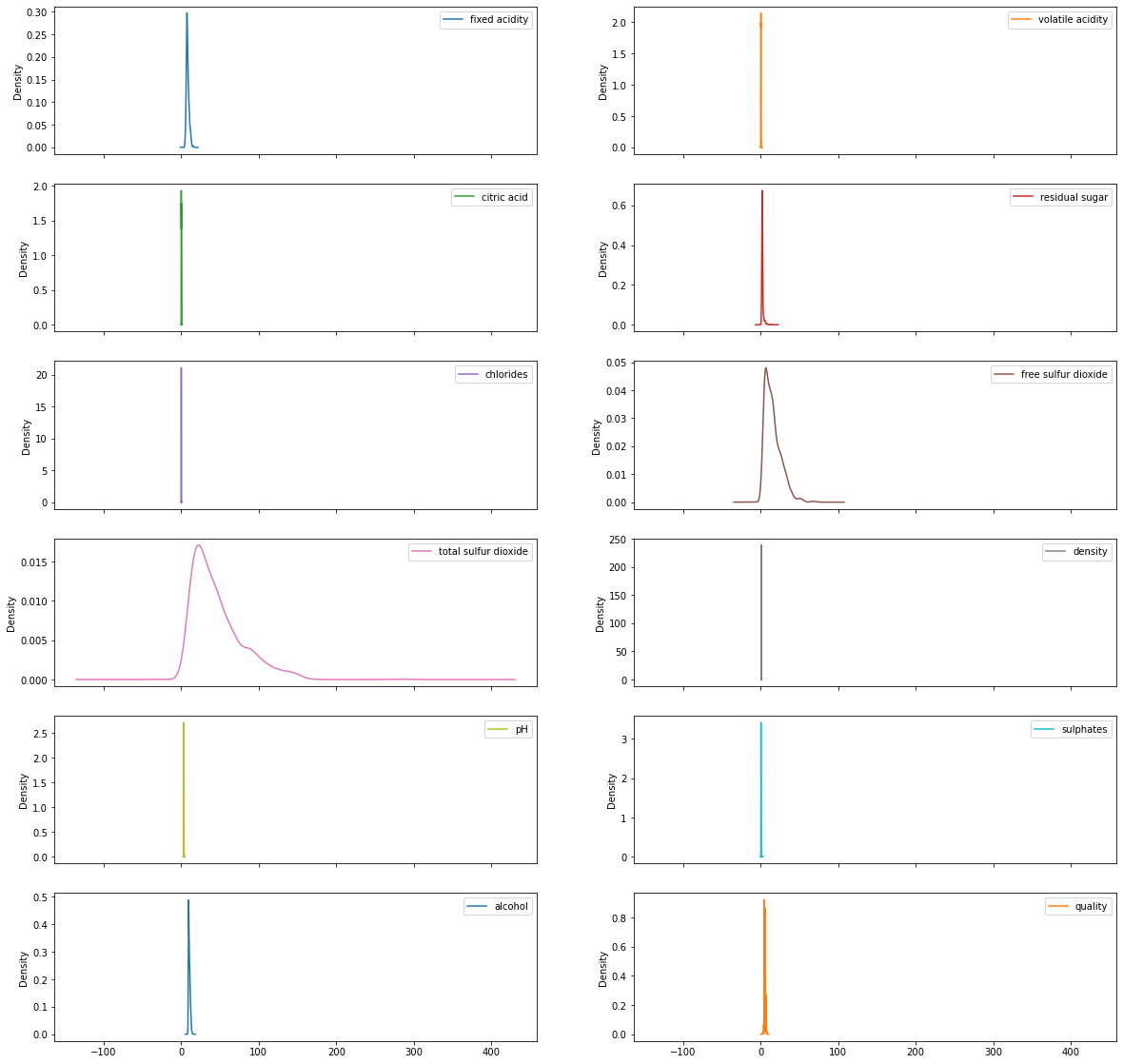
There are no null values in the dataset. So there is no need to act to fill the NaN values in the dataset. We can check this with the help of a heatmap.

Next we are going to plot each feature as boxplot.

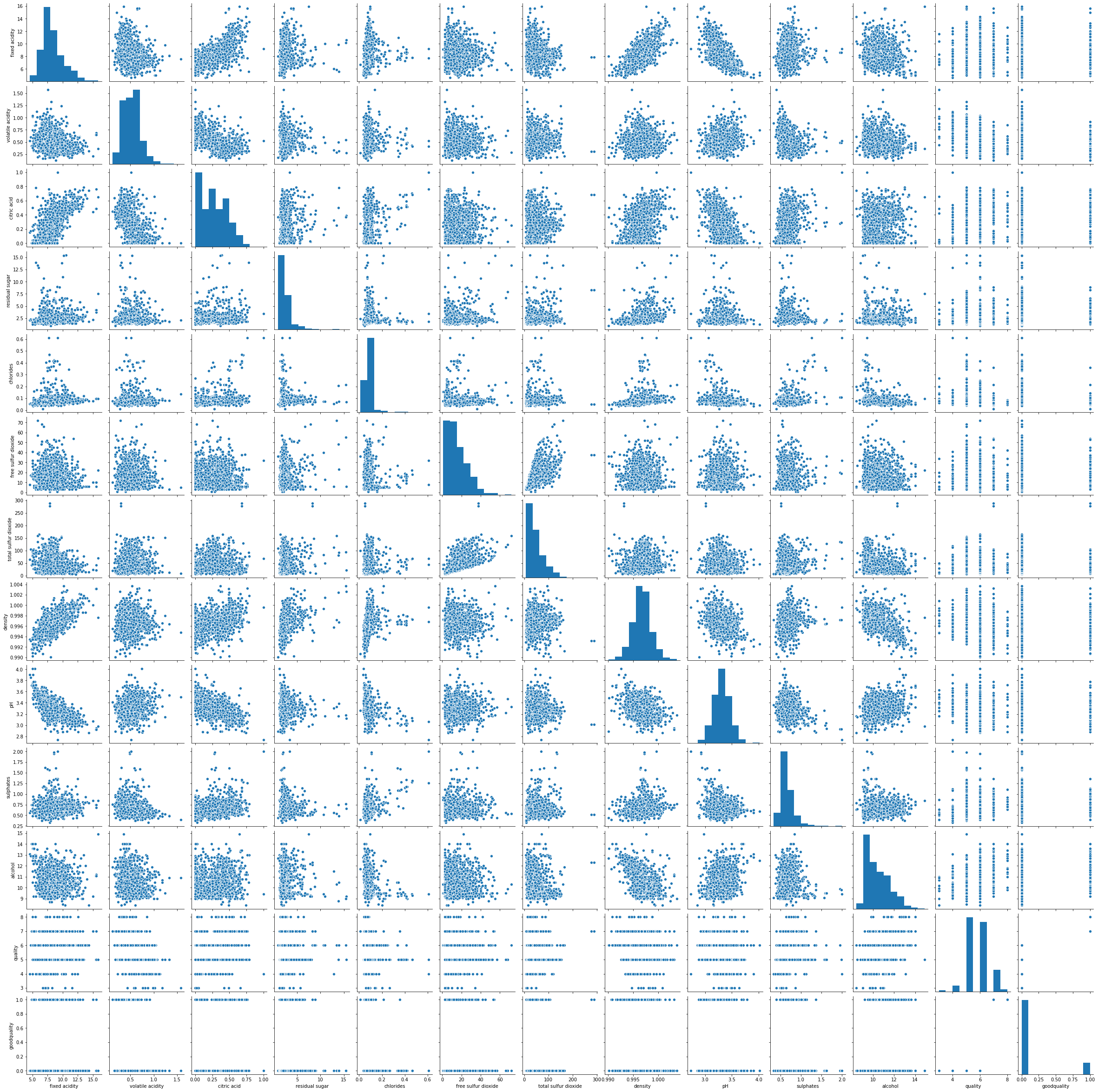


From the plot, we can see that there are outliers in some columns.

Next, we are going to plot some density graphs to check the skewness of the variables



We will use a pair plot to see the different relationships between the variables.



Next, we are going to find out the correlation values using panda’s corr() function

wine\_cor = red\_wine.corr()

The seaborn heatmap of the correlation values as follows



From the above equations and graphs, we can understand that there is a skew in total sulfur dioxide. Also, some columns like residual sugar, volatile acidity, free sulfur dioxide are approximately zero correlated with quality.

**Data Preprocessing**

Data preprocessing consist of the removal of outliers, adjusting the skewness

Our next step is to find and remove outliers. Outliers are the elements that lie beyond the standard deviation on the normal distribution. We are going to find and remove the outliers using Z-score. If the Z-score value is greater than 3 or less than -3 is considered as an oulier. We will perform this for all elements in our dataframe. We are using zscore() method from scipy.stats library.

z\_wine = np.abs(zscore(red\_wine))

wine = red\_wine[(z\_wine<3).all(axis=1)]

The new dataset has no outliers in it. We must consider the amount of data loss when removing outliers.If the data loss would be high, we need to find some other solutions.

Earlier we found that the dataset has some skewed columns. To remove that, we need to import boxcox from scipy.stats library.

from scipy.stats import boxcox

wine["total sulfur dioxide"] = boxcox(wine["total sulfur dioxide"], 0)

wine["free sulfur dioxide"] = boxcox(wine["free sulfur dioxide"], 0)

wine["alcohol"] = boxcox(wine["alcohol"], 0)

Now the data is cleaned and processed. The next part is to divide the data into x and y. The x is the group of dependant variables and y is the target variable.

x = wine.iloc[:,:-2]

y = wine.iloc[:,-2]

**Machine learning**

The upcoming processes are related to machine learning. First, we have to split the whole data to train data and test data. For this, we have to use a parameter called random state. For finding the best random state, we will iterate from 0 to 200 to find the random state which has the highest accuracy.

max\_acc = 0

max\_rs = 0

for i in range(1,200):

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = .22, random\_state = i)

dtr = DecisionTreeClassifier()

dtr.fit(x\_train, y\_train)

pred = dtr.predict(x\_test)

acc = accuracy\_score(y\_test, pred)

if acc > max\_acc:

max\_acc = acc

max\_rs = i

print("Best random state=",max\_rs)

We got that 33 is the best random state. So we divide the whole data set using randomstate= 33

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = .22, random\_state = 33)

Next, we are going to test different machine learning algorithms on the dataset to find which one is the best for this dataset. we are using the DecisionTree classifier, SVC, KNN and Random forest. We are using the metrics like accuracy score, confusion matrix, and classification report to find out the efficiency of the algorithm.

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(x\_train, y\_train)

predict = rf.predict(x\_test)

print("accuracy score=", accuracy\_score(y\_test, predict))

print("conf. matrix =", confusion\_matrix(y\_test, predict))

print("classi. report =", classification\_report(y\_test, predict))

Similarly, we do this procedure for all ML algorithms. Again we do cross-validation for all the models mentioned before. From these, we will find out the best ML algorithm by comparing the accuracy. Here, SVC is the best Ml algorithm. So we are going to use SVC for training.

#Support Vector Machine

svc = SVC()

svc.fit(x\_train, y\_train)

predict = svc.predict(x\_test)

print("accuracy score=", accuracy\_score(y\_test, predict))

print("conf. matrix =", confusion\_matrix(y\_test, predict))

print("classi. report =", classification\_report(y\_test, predict))

Now we have to perform hyper parameter tuning to improve the accuracy of the model. We are using GridSearchCV to find out the best parameters for the model to get the maximum accuracy.

from sklearn.model\_selection import GridSearchCV

parameters = {'C': [0.1, 1, 10, 100, 1000],

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

'kernel': ['rbf']}

GCV = GridSearchCV(SVC(), parameters, cv=5)

GCV.fit(x\_train, y\_train)

We got the best parameters as

{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

The accuracy has improved by around 10%.

The final step is to save the model for future prediction with the help of joblib library

import joblib

joblib.dump(GCV.best\_estimator\_, "WineMOdel.obj")