INTRODUCTION TO DATA SCIENCE

Title :- Wine Quality Prediction

Project report by

Raghav R Sharma 19ucs204

Prabjot Singh 19ucs166

Viren Saroha 19ucs152

Manan Maheshwari 19ucs213

Course Coordinator

Dr. Sudheer Sharma

Dr. Aloke Datta

Dr. Indra Deep Mastan

LNMIIT
The LNM Institute of
Information Technology

Index:

1) Introduction:-

Paulo Cortez, University of Minho, Guimarães, Portugal, http://www3.dsi.uminho.pt/pcortez

A. Cerdeira, F. Almeida, T. Matos and J. Reis, Viticulture Commission of the Vinho Verde Region(CVRVV), Porto, Portugal @2009.The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine.

Goal :-

To predict the quality of Wine using 13 columns and 6497 rows

Datasets :-

For red wine dataset :- https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

For White wine dataset:-

https://www.kaggle.com/piyushagni5/white-wine-quality

Here,we merge both the datasets to have more data for training and for better predictions.For establishing the originality,we add a new column name Type containing labels (Red = 0) and (White=1).

Github link for code:-

https://github.com/raghav5102/IDS_Project_WineQualityPrediction

Description:-

1. **Fixed Acidity:** most acids involved with wine or fixed or non-volatile.
2. **Volatile Acidity:** the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste.
3. **Citric Acid:** found in small quantities, citric acid can add ‘freshness’ and flavour to wines.
4. **Residual Sugar:** the amount of sugar remaining after fermentation stops, it’s rare to find wines with less than 1 gram/litre and wines with greater than 45 grams/litre are considered sweet.
5. **Chlorides:** the amount of salt in the wine.
6. **Free Sulphur Dioxide:** the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine.
7. **Total Sulphur Dioxide:** amount of free and bound forms of S02; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine.
8. **Density:** the density of water is close to that of water depending on the percent alcohol and sugar content.
9. **pH:** describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale.

10. **Sulphates:** a wine additive which can contribute to sulphur dioxide gas (S02) levels, which acts as an antimicrobial and antioxidant.
11. **Alcohol:** the percent alcohol content of the wine.
12. **Quality:** score between 0 and 10.

2) Importing Necessary Libraries:-

Here,we import all the necessary python libraries which can be used throughout the project.

IMPORT NECESSARY LIBRARIES

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    from sklearn.model_selection import train_test_split
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import precision_recall_fscore_support
```

3) Importing and Merging Data:-

We have to import both the datasets:- Red Wine and White wine. Then we will merge them using the **pd.concat** function to make our final dataset for exploring.

Import the Datasets:-

```
df_rw = pd.read_csv('winequality-red.csv')
df_ww = pd.read_csv('wineQualityWhites.csv')
```

Add Type column in both Datasets to establish the originality of datasets . Value of Type columns in:-

Red dataset = Red

White dataset = White

```
[ ] df_rw['Type'] = 'red'
    df_ww['Type'] = 'white'
```

Merge both the dataset to make the final dataset

```
[ ] final_df = pd.concat([df_rw,df_ww])
```

Substitute Red = 0 and White = 1 in final dataset

```
[ ] Type={'red':1,'white':0}
```

```
[ ] final_df['Type']= [Type[item] for item in final_df['Type']]
```

4) Exploring the data

Now,we can look out our final dataset with the help of **head()** function.

```
final_df.head(10)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Type
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5	red
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5	red
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7	red
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7	red
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5	red

Shape of our dataset:- Using **shape** function

```
final_df.shape
```

```
(6497, 13)
```

Total no. of columns of Our dataset:- Using **columns** function

```
final_df.columns
```

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

It is important to describe our data in the form of Count,Mean,Standard Deviation etc for getting mathematical interpretation about our dataset.

For this we can use the **describe()** function.

```
[ ] final_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

s	alcohol	quality	Type
ɔ	6497.000000	6497.000000	6497.000000
ɜ	10.491801	5.818378	0.246114
ɔ	1.192712	0.873255	0.430779
ɔ	8.000000	3.000000	0.000000
ɔ	9.500000	5.000000	0.000000
ɔ	10.300000	6.000000	0.000000
ɔ	11.300000	6.000000	0.000000
ɔ	14.900000	9.000000	1.000000

Now,we have to find null values if present in the datasets because if null values are present,we can't use a classifier to predict the quality of wine. So we have to take care of null values if present.

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6497 entries, 0 to 4897
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64  
 12  Type                  6497 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 710.6 KB
```

Observations :-

1) No null values are present
2) All the columns are of int and float type.So we can go ahead with the dataset for analysis and prediction purposes.
3) Only Numerical data is present,no categorical data after converting red and white into 0 and 1 respectively.

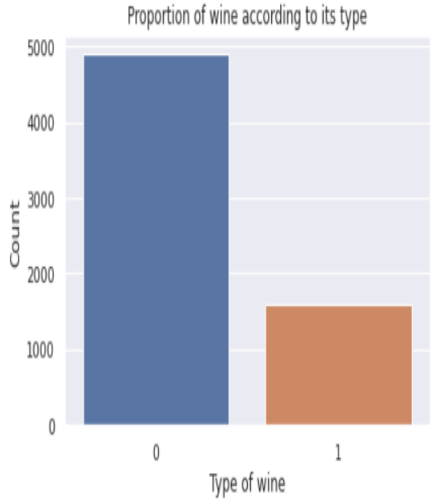
5) Data Visualization

1) Let us see the distribution of datasets into red and white wine using a bar plot.

Distribution of DataSet into Red and Wine

```
fig = plt.figure(figsize=(22,8))
plt.subplot2grid((2,3),(0,0))
sns.countplot(final_df['Type'])
plt.title("Proportion of wine according to its type"); plt.ylabel("Count"); plt.xlabel("Type of wine")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.
  FutureWarning
Text(0.5, 0, 'Type of wine')
```



Here, 0 = Red wine and 1 = White wine

Observation :-

1) No. of instances of Red wine is greater than White wine

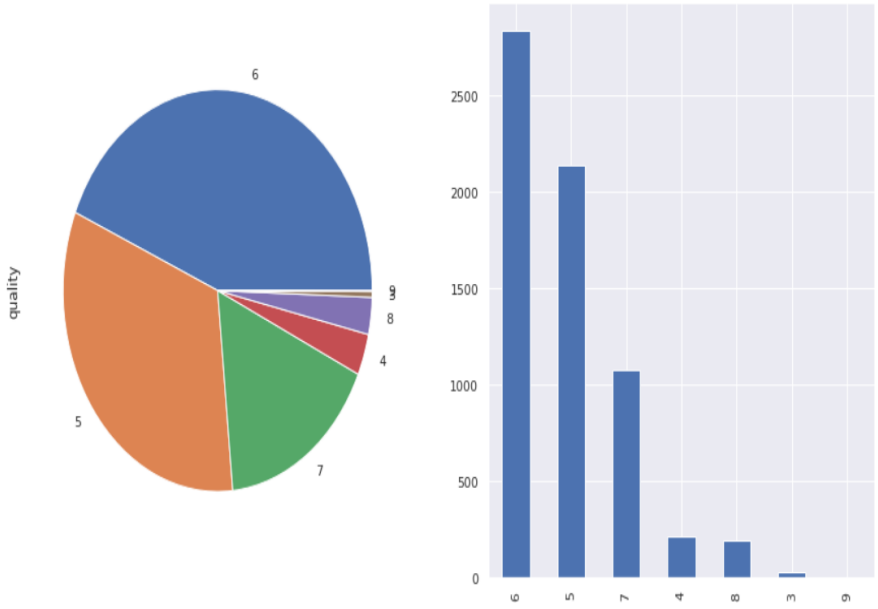
2) No. of instances of Red wine :- Between 4500 and 5000

No. of instances of White wine :- Between 1500 and 2000

2) Now let us look at our target variable i.e Quality Prediction.We will find the value counts of quality attributes with the help of Pie chart and Bar plot.

```
f,ax = plt.subplots(1,2,figsize=(15,8))
pd.Series(final_df['quality'].value_counts()).plot(kind='pie',ax=ax[0])
final_df['quality'].value_counts().plot.bar(ax=ax[1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc50a46a850>
```



Observations:- Wine with Quality :-

6 :- Most no. of instances(Between 2800-2900)

5 :- No. of instances between 2000 - 2200

7 :- No. of instances between 1000 - 1200

4 :- No. of instances between 200-300

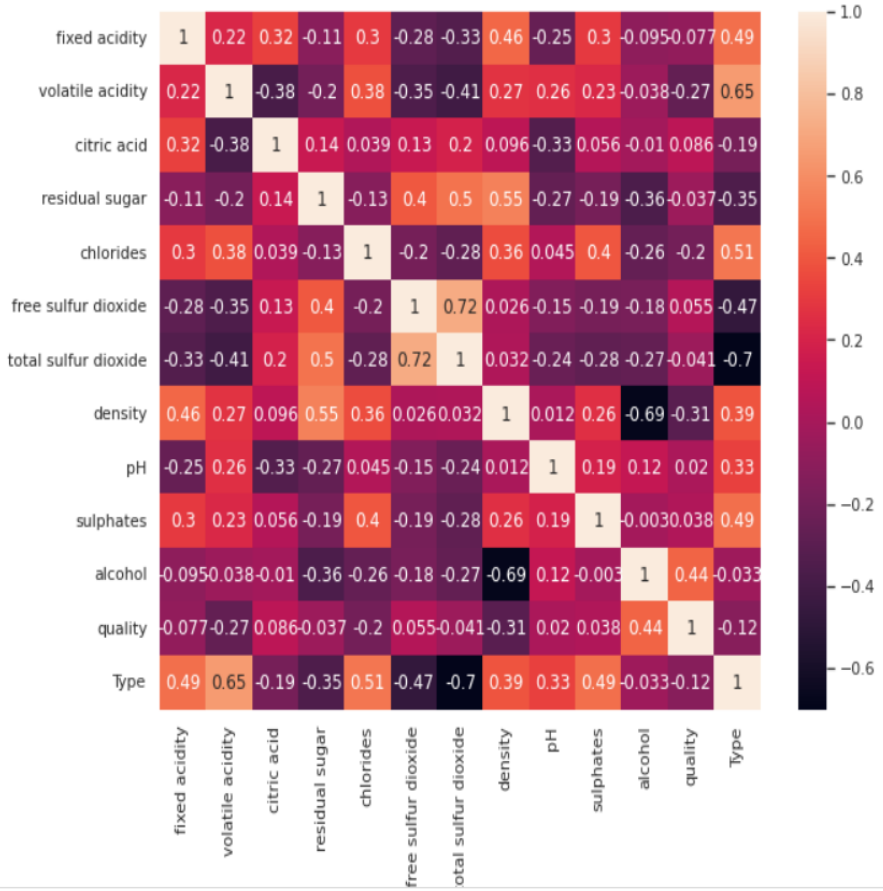
8 :- No. of instances between 150 - 250

2 :- No. of instances between 20 - 50

9 :- Least no. of instances(Between 5- 10)

3) Now we will see the correlation between all the attributes using correlation matrix with the help of seaborn library.

```
[51] sns.set(rc={'figure.figsize':(10,8)})
     Correlation = sns.heatmap(final_df.corr(), annot=True)
```



Observations : -

1) Correlation matrix can describe the whole scenario between the attributes. 3 scenarios are possible :-

1) Values greater than 0:- Positively correlated

Moves in same direction

Examples :-

1) Fixed acidity vs Volatile acidity

2) Chlorides vs Density

3) Quality vs Alcohol

2) Values less than 0 :- Negatively correlated

Moves in opposite direction

Examples:-

1) Citric Acid vs Volatile acidity

2) Sulphate vs Residual Sugar

3) Quality vs chlorides

3) Values equal to 0 :- No relation

Here,no combination is present

2) Volatile acidity, chlorides and pH are negatively correlated to quality -- hence our statement was right that quality increases with decrease in value of these features; and vice versa for other features.

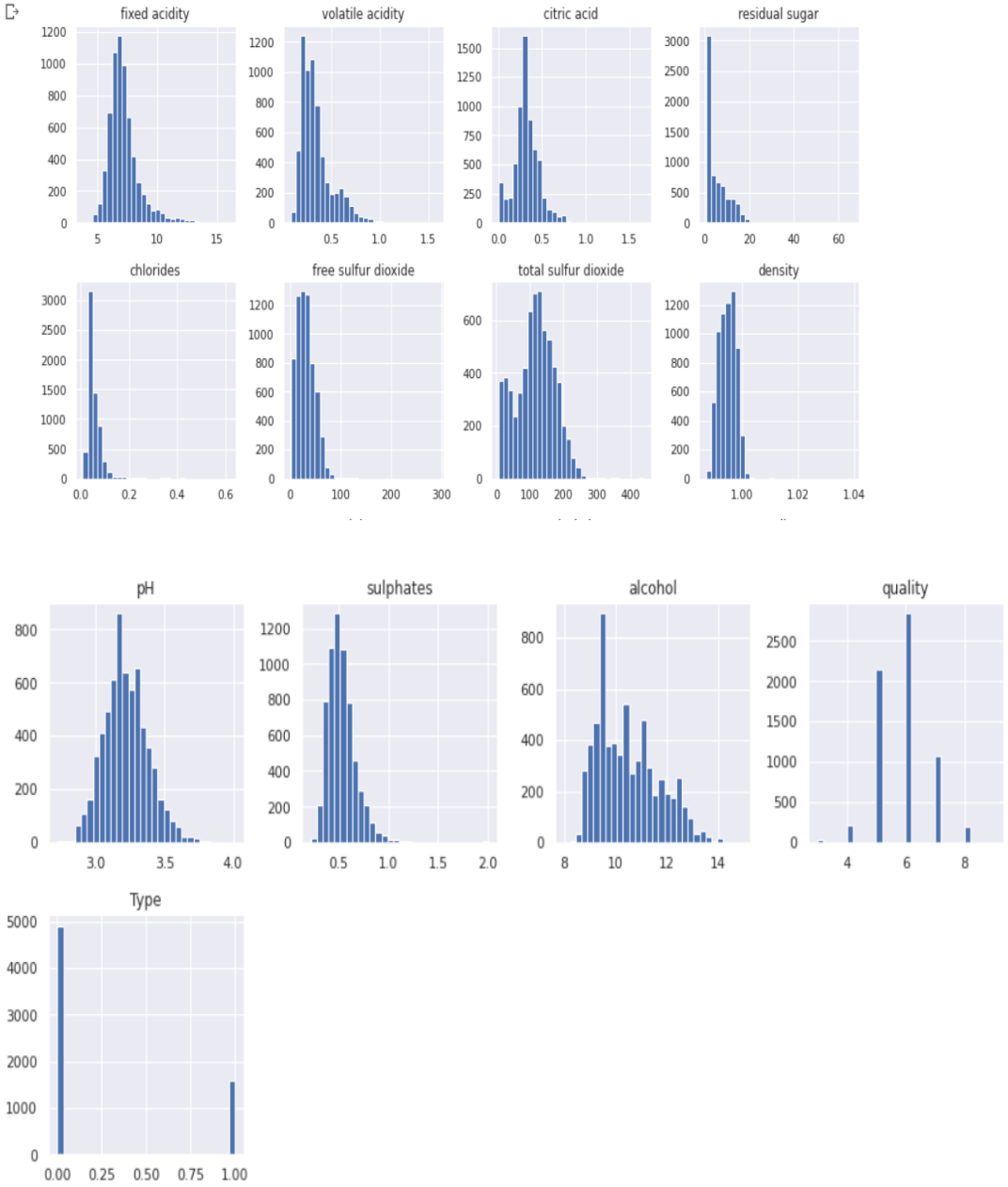
3) Free sulfur dioxide and total sulfur dioxide are highly correlated to each other.

4) There are many features with correlation < 0.5 to quality, and may be removed from the dataset

4) Now let us visualize each attribute and analyze their characteristics using Histogram for better understanding of each feature and to know their concentration.

```
##Histogram

final_df.hist(figsize=(15,15),bins=30)
plt.show()
```



OBSERVATIONS:-

1) Fixed Acidity :- Most of the values lies between 6-8

(around 70%)

2) Volatile Acidity :- Most of the values lies between 0.2 - 0.4

3) Citric Acid : - Most of the values lies between 0.1 - 0.4

4) Residual Sugar :- Most of the values lies between 0 - 5

5) Chlorides :- Most of the values lies between 0-0.2(around 90%)

6) Free Sulphur Dioxide :- Most of the values lies between 20 - 50

7) Total Sulphur Dioxide :- Most of the values lies between 130-170

8) Density :- Most of the values lies between 0.4 - 0.8

9) pH :- Most of the values lies between 3.2-3.4

10) Sulphates :- Most of the values lies between 0.6-0.8

11) Alcohol :- Equally distributed throughout the histogram

12) Quality :- 6 has most values

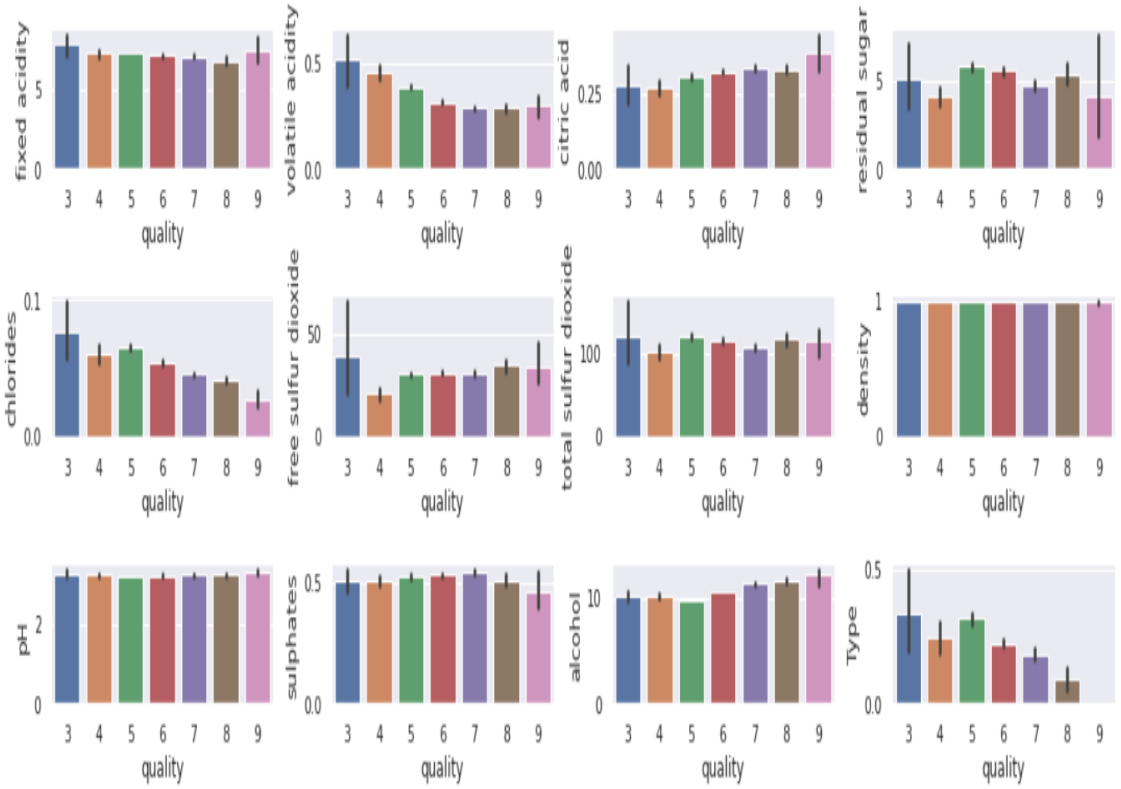
5) Finally, we will visualize the relationship between the dependent variable and independents variable using a bar plot.Here,

Dependent variable :- Quality

Independent variable :- All other features in dataset

```
[85]
fig, ax = plt.subplots(ncols=4, nrows=3, figsize=(15, 5))
ax = ax.flatten()
index=0
for i in final_df.columns:
  if i != 'quality':
    sns.barplot(x='quality', y=i, data=final_df, ax=ax[index])
    index+=1
plt.tight_layout(pad=0.4)
plt.show
```

```
<function matplotlib.pyplot.show>
```



Observations :-

1) In cases of Fixed acidity, density, pH,Sulphates, Alcohol, no. of values for each quality is approximately the same, so we can not differentiate quality of the wine on the basis of these features.

2) In case of Citric acid,Residual Sugar,Free sulphur dioxide,total sulphur dioxide no. of values for each quality is significantly different,so it can be used for wine quality prediction.

3) In the case of Volatile acid,Chlorides,Type,it is best to distinguish between the wine quality predictions.

Here, Our Visualization of data is finish.We look almost all the aspects of our data using bar plot, pie chart, histogram etc and gain many inferences which we can use for further analysis.

6) Data Pre-Processing

1) Here, No null values are present as we seen above so we don't need to touch our dataset for that purpose

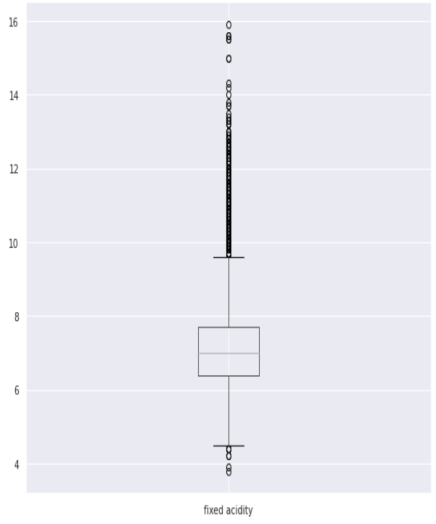
2) No categorical values present other than Type which we deal above.

Now only outliers are present in our dataset for each continuous variable.So we have to remove that using Box-Whisker plot for making our final dataset which we will train in future.

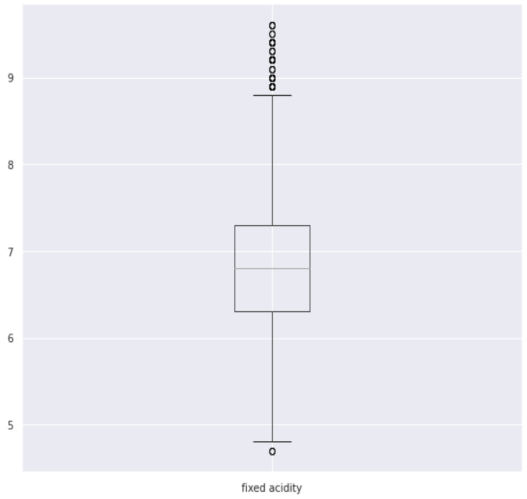
```
q1 = final_df.quantile(0.25)
q3 = final_df.quantile(0.75)
iqr = q3-q1
l = q1-1.5*iqr
h = q3+1.5*iqr
final_data = final_df[((final_df >= (q1 - 1.5 * iqr))& (final_df <= (q3 + 1.5 * iqr))).all(axis=1)]
final_data.describe()
```

Fixed Acidity :-

Before :-

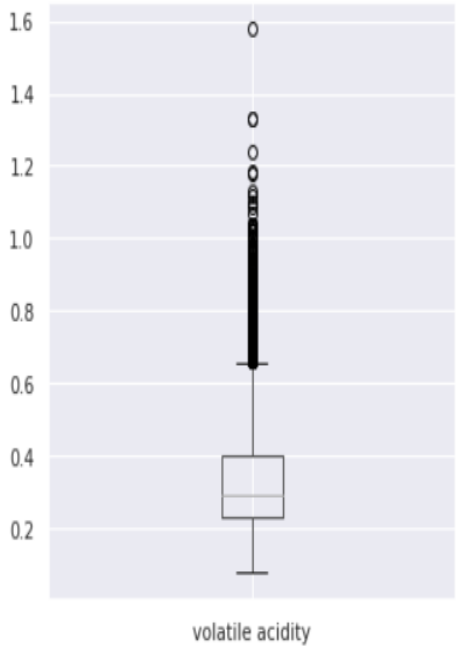


After :-

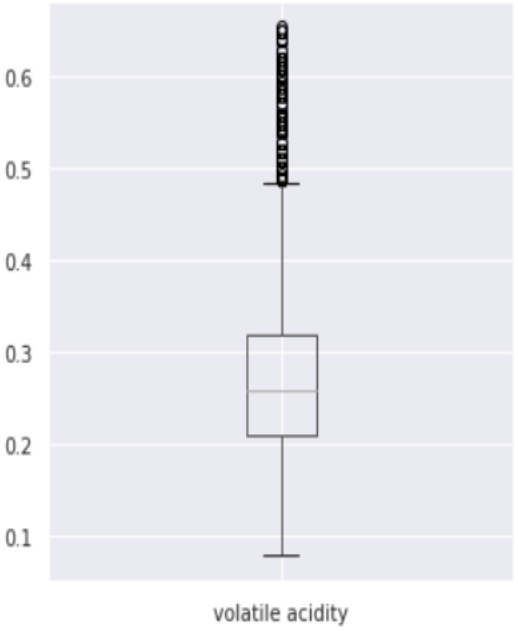


Volatile Acidity :-

Before:-

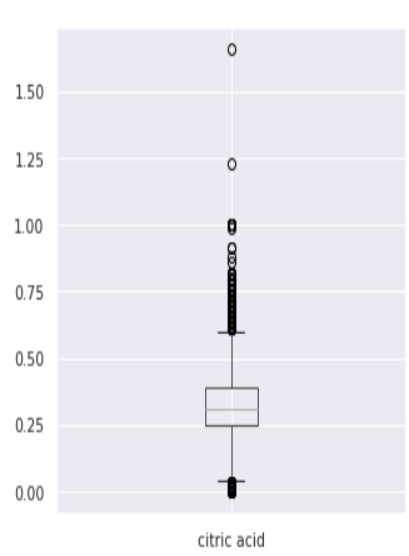


After:-

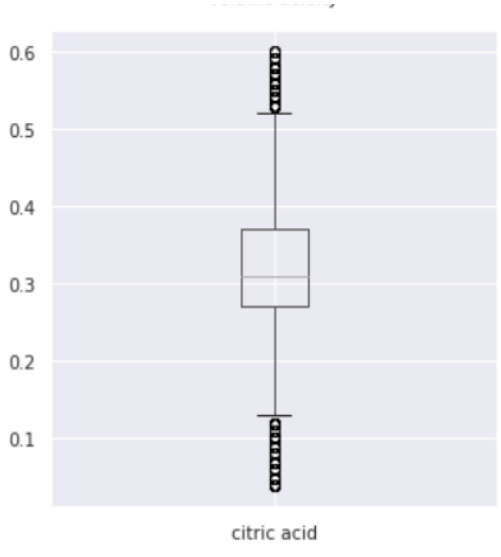


Citric Acidity:-

Before :-

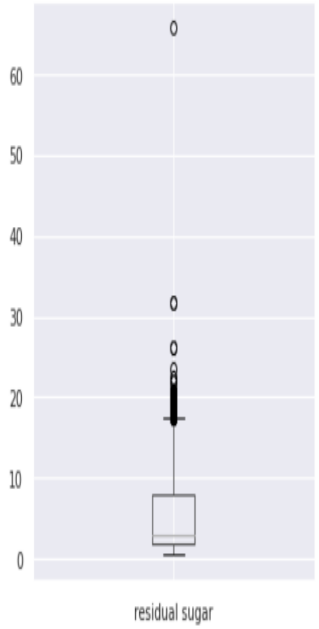


After:-

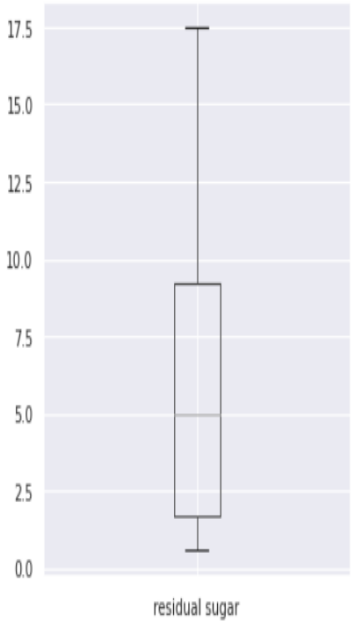


Residual Sugar :-

Before:-

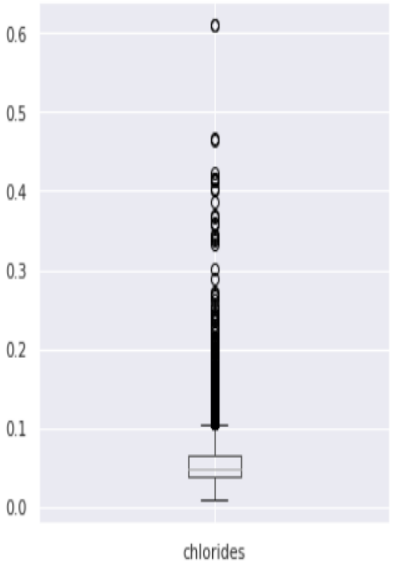


After:-

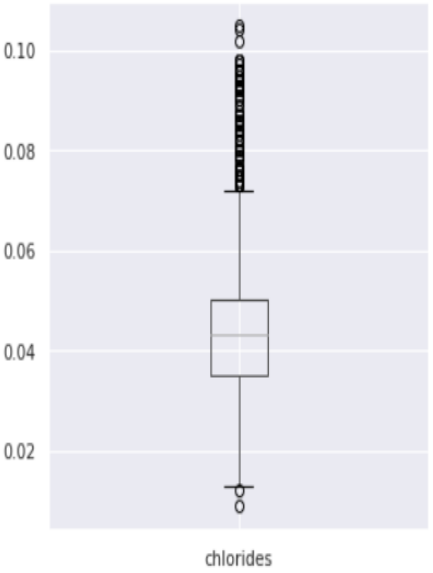


Chlorides :-

Before:-

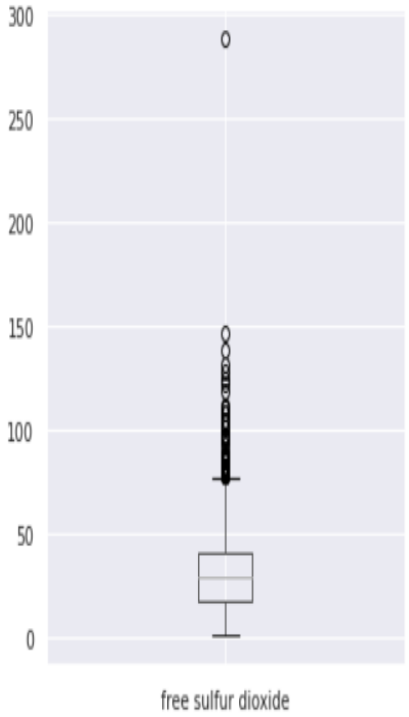


After:-

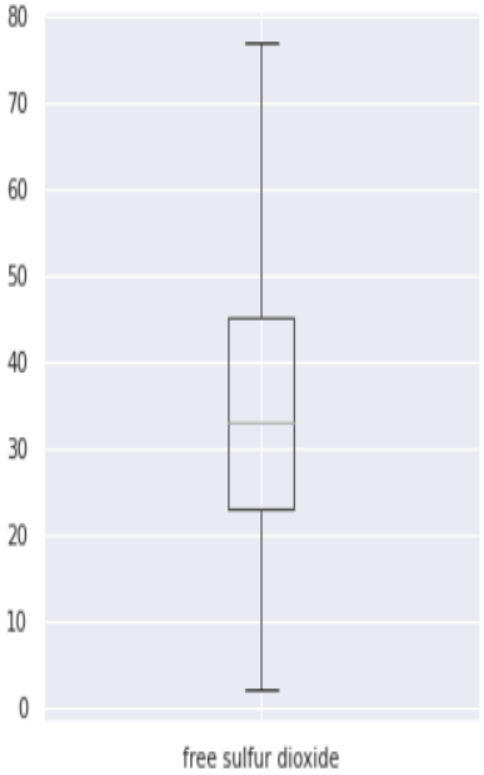


Free Sulphur Dioxide :-

Before :-

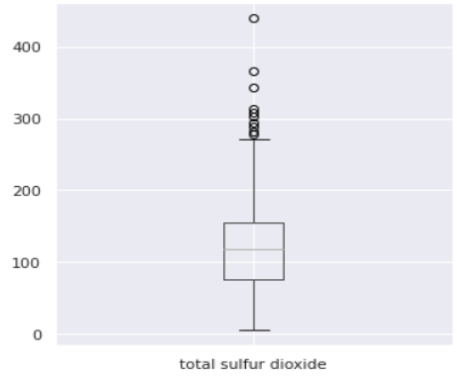


After :-

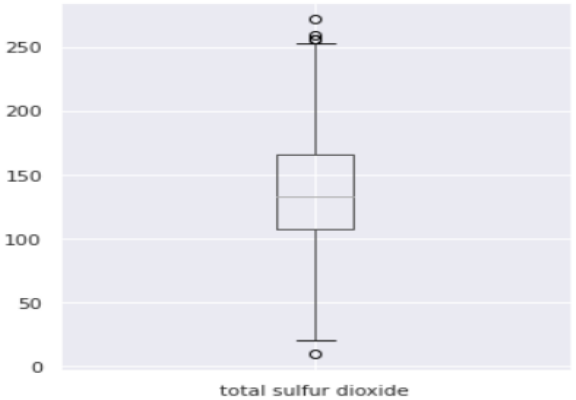


Total Sulphur Dioxide :-

Before:-

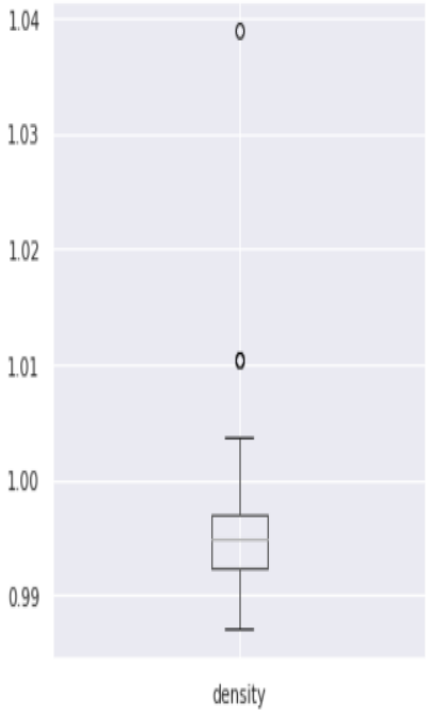


After:-

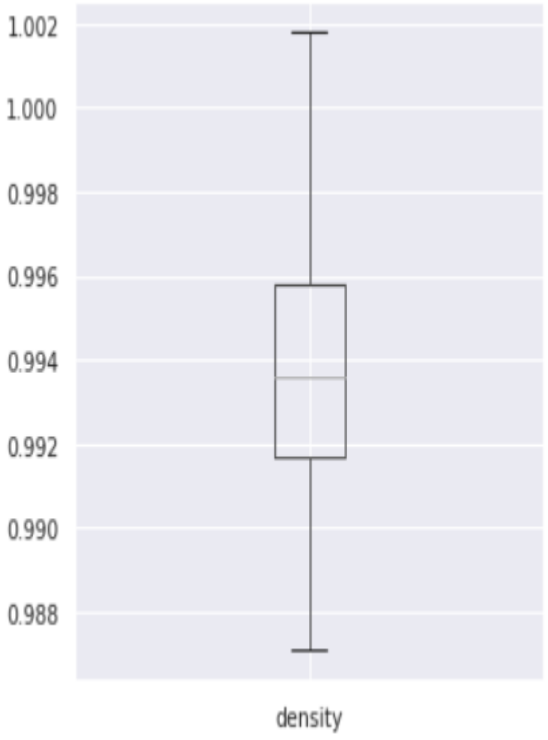


Density:-

Before:-

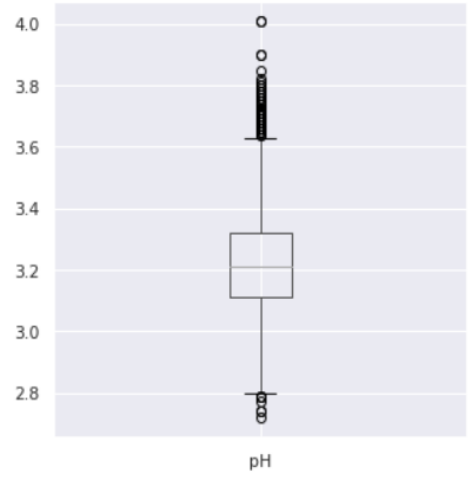


After:-

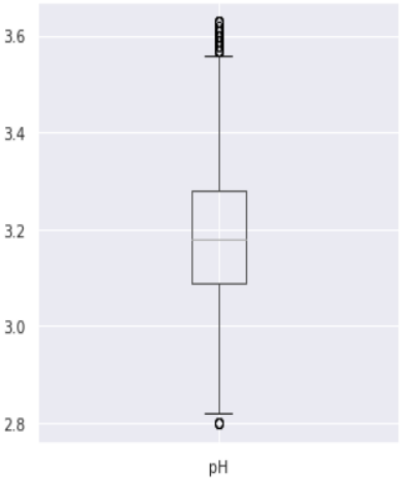


pH:-

Before:-

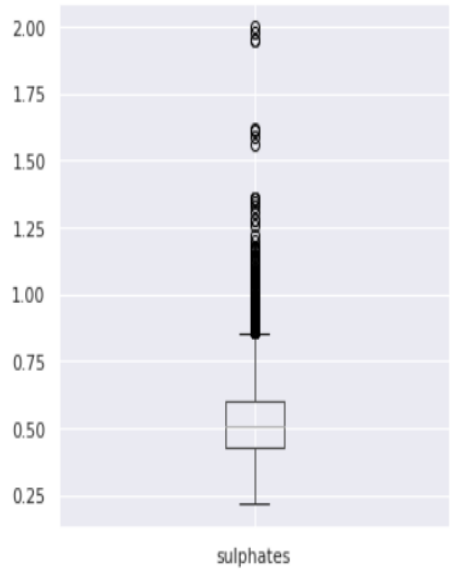


After:-

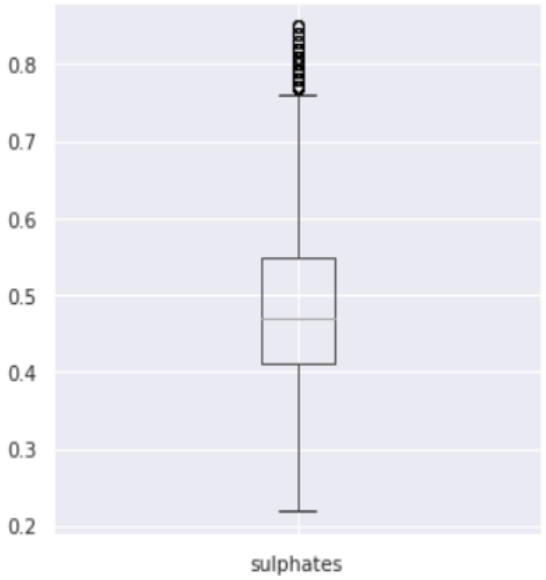


Sulphates:-

Before:-



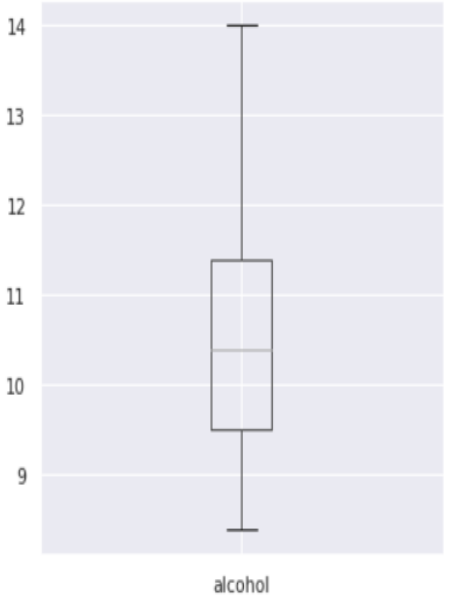
After:-



Alcohol :-

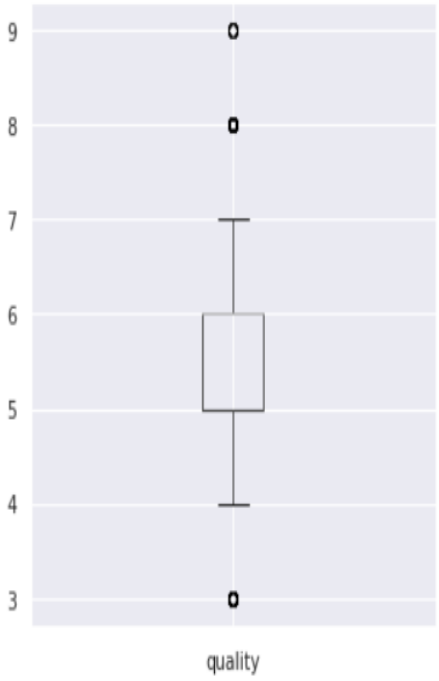
Before:-

After :-

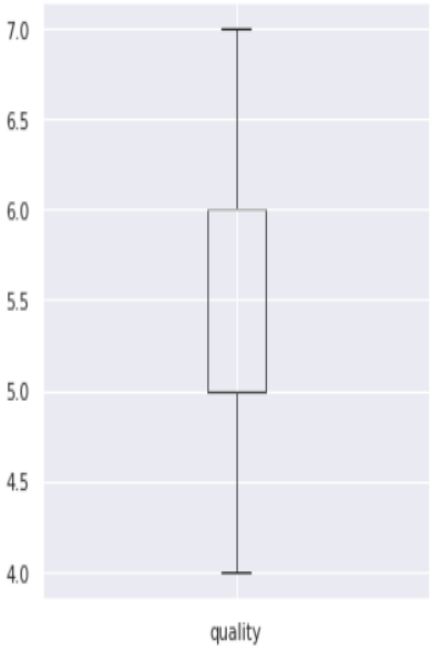


Quality :-

Before :-



After:-



Here after removing outliers from quality attributes, We only have 4 classes left :- 4,5,6,7. It produces better results in future.

We remove all the outliers from our dataset.Now our final dataset is ready for Machine learning.

Name of final dataset :- final_data

Shape of our final dataset:-

```
[83] final_data.shape

(4158, 13)
```

7) Training The Dataset

Before training the model, we need to separate the label from the data. That is separating X and y matrices.

```
[ ] y = final_data['quality']
    X = final_data.drop('quality',axis = 1)
```

In the next step, we will split the dataset into training and testing data using train_test_split.Here we take 30% of dataset as a test set

```
[ ] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,shuffle=True)
```

Shape of X_train,X_test :-

```
[ ] X_train.shape

    (2910, 12)

    X_test.shape

    (1248, 12)
```

Here, We also use Principal Component Analysis(PCA) for dimensionality reduction and also train the PCA model and compare it with the original dataset to get the best outcome for a certain model. We reduce dimensionality to **5 using n_components**.

PCA Implementation:-

```
[ ] pca = PCA(n_components = 5)
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = pca.transform(X_test)

[ ] X_train_pca[0]

    array([37.77277074, 10.87078854, -2.23190018,  0.7663796 ,  0.32065161])

[ ] X_test_pca[0]

    array([18.54107007, 20.71266101, -2.8466092 , -1.07207672,  0.02795737])

[ ] X_train_pca.shape

    (2910, 5)

[ ] X_test_pca.shape

    (1248, 5)
```

MODELS:-

1) K-Nearest Neighbours

1) Import K-Neighbour Classifier
2) Take different values of neighbours
3) Train the classifier on both X_train and X_train_pca and predict the output for both and compare the results with the help of accuracy.

Code:-

```
[130] from sklearn.neighbors import KNeighborsClassifier
      from sklearn.metrics import accuracy_score,classification_report
```

```
##Using Original Dataset

neighbour = [1,3,5,10,20]
for i in neighbour:
    clf = KNeighborsClassifier(n_neighbors=i, weights='uniform')
    clf.fit(X_train,y_train)
    prediction = clf.predict(X_test)
    accuracy = accuracy_score(y_test,prediction)
    print('Accuracy for K nearest neighbour = ',i,'is',accuracy)
```

```
Accuracy for K nearest neighbour =  1 is 0.5913461538461539
Accuracy for K nearest neighbour =  3 is 0.5072115384615384
Accuracy for K nearest neighbour =  5 is 0.48717948717948717
Accuracy for K nearest neighbour =  10 is 0.4823717948717949
Accuracy for K nearest neighbour =  20 is 0.46955128205128205
```

```
[132] ##Using Principal Component Analysis

neighbour = [1,3,5,10,20]
for i in neighbour:
    clf = KNeighborsClassifier(n_neighbors=i, weights='uniform')
    clf.fit(X_train_pca,y_train)
    prediction = clf.predict(X_test_pca)
    accuracy = accuracy_score(y_test,prediction)
    print('Accuracy for K nearest neighbour = ',i,'is',accuracy)
```

```
Accuracy for K nearest neighbour =  1 is 0.592948717948718
Accuracy for K nearest neighbour =  3 is 0.5080128205128205
Accuracy for K nearest neighbour =  5 is 0.484775641025641
Accuracy for K nearest neighbour =  10 is 0.4791666666666667
Accuracy for K nearest neighbour =  20 is 0.4703525641025641
```

Here Best model is Principal Component Analysis with K-Neighbours = 1

So,Confusion matrix and Classification Report for that model is :-

```
[133] clf = KNeighborsClassifier(n_neighbors=1, weights='uniform')
      clf.fit(X_train_pca,y_train)
      prediction = clf.predict(X_test_pca)
      confusion_matrix(y_test,prediction)
```

```
array([[  7,  13,   7,   3],
       [  6, 220, 116,  37],
       [ 10, 109, 377,  94],
       [  4,  31,  78, 136]])
```

```
[135] print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           4       0.26      0.23      0.25        30
           5       0.59      0.58      0.59       379
           6       0.65      0.64      0.65       590
           7       0.50      0.55      0.52       249

    accuracy                           0.59      1248
   macro avg       0.50      0.50      0.50      1248
weighted avg       0.59      0.59      0.59      1248
```

Here we use the K-neighbours model because it is the most simple model which works on similarity of the patterns. Here similarity is calculated using distance between 2 patterns.

2) Decision Tree

Code :-

```
[136] from sklearn.tree import DecisionTreeClassifier
```

Train on Original Dataset :-

```
[137] ##Using Original Dataset

      depth = [25,50,100,500,None]
      for i in depth:
          clf = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=i)
          clf.fit(X_train,y_train)
          prediction = clf.predict(X_test)
          accuracy = accuracy_score(y_test,prediction)
          print('Accuracy for max depth = ',i,'is',accuracy)
```

```
Accuracy for max depth =  25 is 0.592948717948718
Accuracy for max depth =  50 is 0.6049679487179487
Accuracy for max depth =  100 is 0.6073717948717948
Accuracy for max depth =  500 is 0.6001602564102564
Accuracy for max depth =  None is 0.6049679487179487
```

Train on PCA :-

```
[139] ##Using Principal Component Analysis

      depth = [25,50,100,500,None]
      for i in depth:
          clf = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=i)
          clf.fit(X_train_pca,y_train)
          prediction = clf.predict(X_test_pca)
          accuracy = accuracy_score(y_test,prediction)
          print('Accuracy for max depth = ',i,'is',accuracy)
```

```
Accuracy for max depth =  25 is 0.6185897435897436
Accuracy for max depth =  50 is 0.6169871794871795
Accuracy for max depth =  100 is 0.6217948717948718
Accuracy for max depth =  500 is 0.6169871794871795
Accuracy for max depth =  None is 0.6209935897435898
```

Here,Best accuracy can be achieved by using PCA with

Max_depth = 100

Confusion Matrix and Classification Report :-

```
clf = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=100)
clf.fit(X_train_pca,y_train)
prediction = clf.predict(X_test_pca)
confusion_matrix(y_test,prediction)
```

```
array([[  7,  13,   8,   2],
       [ 16, 230, 100,  33],
       [  9, 114, 386,  81],
       [  0,  20,  94, 135]])
```

```
[145] print(classification_report(y_test,prediction))
```

	precision	recall	f1-score	support
4	0.22	0.23	0.23	30
5	0.61	0.61	0.61	379
6	0.66	0.65	0.66	590
7	0.54	0.54	0.54	249
accuracy			0.61	1248
macro avg	0.51	0.51	0.51	1248
weighted avg	0.61	0.61	0.61	1248

Here We use Decision tree because it is an iterative method for classifying the labels with the help of one tree. So, it gives better accuracy.We can use multiple trees for classification with the help of Random Forest which we will see later.

3) Support Vector Machine (SVM)

Code :-

```
from sklearn import svm
```

Train on Original Dataset:-

```
##Using Original Dataset

clf = svm.SVC()
clf.fit(X_train,y_train)
prediction = clf.predict(X_test)
accuracy_svm = accuracy_score(y_test,prediction)
print(accuracy_svm)
```

```
0.47275641025641024
```

Train on PCA:-

```
[148] #Using Principal Component Analysis

clf = svm.SVC()
clf.fit(X_train_pca,y_train)
prediction = clf.predict(X_test_pca)
accuracy_svmpca = accuracy_score(y_test,prediction)
print(accuracy_svmpca)
```

```
0.48157051282051283
```

Here, Best model in when we apply PCA on dataset with

accuracy of = 0.4815

Confusion matrix and Classification Report :-

```
[ ] confusion_matrix(y_test,prediction)

array([[  0,   7,  45,   0],
       [  0,  33, 321,   0],
       [  0,  29, 554,   0],
       [  0,   3, 256,   0]])
```

```
✓ 0s  print(classification_report(y_test,prediction))

              precision    recall  f1-score   support

           4       0.00      0.00      0.00        30
           5       0.52      0.13      0.21       379
           6       0.48      0.93      0.63       590
           7       0.00      0.00      0.00       249

    accuracy                           0.48      1248
   macro avg       0.25      0.27      0.21      1248
weighted avg       0.38      0.48      0.36      1248
```

Here we use SVM because it is a multi-classification problem and SVM uses non-linear kernels for predicting so it captures much more complex relationships between your datapoints.

4) Naive Bayes

Code :-

```
[150] from sklearn.naive_bayes import GaussianNB
```

Train on Original Dataset:-

```
[151] ##Using Original Dataset

      GNB = GaussianNB()
      GNB.fit(X_train,y_train)
      prediction = GNB.predict(X_test)
      accuracy_nb = accuracy_score(y_test,prediction)
      print(accuracy_nb)

      0.48717948717948717
```

Train on PCA :-

```
[152] ##Using Principal Component Analysis

      GNB = GaussianNB()
      GNB.fit(X_train_pca,y_train)
      prediction = GNB.predict(X_test_pca)
      accuracy_nbpca = accuracy_score(y_test,prediction)
      print(accuracy_nbpca)

      0.5016025641025641
```

Here best model for Naive Bayes is with PCA on original dataset

(Accuracy :- 0.5016)

Confusion matrix and Classification Report :-

```
[156] confusion_matrix(y_test,prediction)

array([[  0,  15,  14,   1],
       [  1, 134, 237,   7],
       [  1, 100, 425,  64],
       [  0,  20, 162,  67]])
```

```
[157] print(classification_report(y_test,prediction))

              precision    recall  f1-score   support

           4       0.00      0.00      0.00        30
           5       0.50      0.35      0.41       379
           6       0.51      0.72      0.60       590
           7       0.48      0.27      0.35       249

    accuracy                           0.50      1248
   macro avg       0.37      0.34      0.34      1248
weighted avg       0.49      0.50      0.48      1248
```

Here, We use Naive Bayes Classifier with gaussian distribution because it can give us the best true positives for a given dataset because of less no. of misclassification and it gives good accuracy in multiclass classification.

5) Random Forest Classification :-

Code :-

```
[158] from sklearn.ensemble import RandomForestClassifier
```

Train on Original Dataset :-

```
[159] clf = RandomForestClassifier()
      clf.fit(X_train,y_train)
      prediction = clf.predict(X_test)
      accuracy_rf = accuracy_score(y_test,prediction)
      print(accuracy_rf)

      0.6971153846153846
```

Train On PCA :-

```
[160] clf = RandomForestClassifier()
      clf.fit(X_train_pca,y_train)
      prediction = clf.predict(X_test_pca)
      accuracy_rfpca = accuracy_score(y_test,prediction)
      print(accuracy_rfpca)

      0.6858974358974359
```

Here, we get the best model with the Original Dataset with the accuracy of 0.6971.

Confusion matrix and Classification Report :-

```
[ ]  confusion_matrix(y_test,prediction)
```

```
array([[  6,  28,  18,   0],
       [  1, 241, 108,   4],
       [  1,  63, 480,  39],
       [  0,   3, 121, 135]])
```

```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           4       0.88      0.23      0.37        30
           5       0.72      0.65      0.68       379
           6       0.66      0.78      0.72       590
           7       0.71      0.57      0.64       249

    accuracy                           0.69      1248
   macro avg       0.74      0.56      0.60      1248
weighted avg       0.69      0.69      0.68      1248
```

It is the most complex algorithm among the all,because it has multiple no. of trees which is used to classify labels in different categories.It is upgraded version of Decision tree.Because of its complexity,it learns more than any other algorithm and produces best Output.

Comparison between Models:-

We will compare the accuracy of best models of each algorithm to have an insight about the results :-

```
f,ax=plt.subplots(1,2,figsize=(15,6))
ax[0].plot(Model_keys,Accuracy_values)
ax[0].set_title('Accuracy Score Comparision using Scatter plot')
ax[1].bar(Model_keys,Accuracy_values)
ax[1].set_title('Accuracy Score Comparision using bar plot')
```

```
Text(0.5, 1.0, 'Accuracy Score Comparision using bar plot')
```

Here, Knn = K nearest neighbor

DT = Decision Tree

SVM = Support Vector Machine

NB = Naive Bayesian

RF = Random Forest classifier

8) Conclusions :-

After successfully completed all the steps of predicting the wine quality with the help of different models, we can conclude certain points:-

1) Random Forest is the best classifier among all the other algorithms due to its iterative method and fast-learning.

2) PCA works better in 4 algorithms rather than the original dataset,so we can conclude that it is the best technique for dimensionality reduction without losing essential features.

3) Naive Bayes can give us the best true positives, so always try to use it in prediction.

4) It is important to visualize each and every feature present in the dataset to get a better understanding of it.

5) Always check for null values and outliers,if present remove them in any possible way