WINE QUALITY PREDICTOR

DONE BY: PRATIK SHRIKANT RAO, KUNAL B YELLURKAR, MOHAMMED SHAKEEL, MANJUNATH

ABSTRACT

Wine classification is a difficult task since taste is the least understood of the human senses. A good wine quality prediction can be very useful in the certification phase, since currently the sensory analysis is performed by human tasters, being clearly a subjective approach. An automatic predictive system can be integrated into a decision support system, helping the speed and quality of the performance. Furthermore, a feature selection process can help to analyse the impact of the analytical tests. If it is concluded that several input variables are highly relevant to predict the wine quality, since in the production process some variables can be controlled, this information can be used to improve the wine quality.

Classification models used here are: -

- Logistic Regression.
- K Nearest Neighbour.
- Support Vector Machine (linear classifier).
- Gaussian Naïve Bayes.
- Decision Tree Classifier.
- Random Forest Classifier.

TABLE OF CONTENTS

INTRODUCTION

The aim of this project is to predict the quality of wine on a scale of 0–10 given a set of features as inputs. The dataset used is Red Wine Quality Data set from Kaggle Repository. Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol. And the output variable is quality (score between 0 and 10). We are dealing only with red wine. We have quality being one of these values: [3, 4, 5, 6, 7, 8]. The higher the value the better the quality. In this project we will treat each class of the wine separately and their aim is to be able and find decision boundaries that work well for new unseen data. These are the classifiers.

We will be dealing with different classifier like Logistic Regression, K Nearest Neighbour, Support Vector Machine (linear classifier), Gaussian Naïve Bayes, Decision Tree Classifier and Random Forest Classifier. We will analyse the wine quality using these classifiers to predict the best quality of wine for different values.

ABOUT THE DOMAIN

Data analytics is the science of analysing raw data in order to make conclusions about that information. The techniques and processes of data analytics have been automated into mechanical processes and algorithms that work over raw data for human consumption. Data analytics help a business optimize its performance. Data analysts typically analyse raw data for insights and trends. They use various tools and techniques to help organizations make decisions and succeed.

Steps Involved in Data Analytics: -

- **Understand the problem:** Understanding the wine quality problems, defining the organizational goals, and planning a lucrative solution is the first step in the analytics process. companies often encounter issues such as predicting the quality of wine, giving relevant product recommendations, cancellation of orders, identifying frauds, optimizing vehicle routing, etc.
- **Data Collection:** Next, you need to collect wine quality data, having columns like fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol.
- **Data Cleaning:** Now, all the data you collect will often be disorderly, messy, and contain unwanted missing values. Such data is not suitable or relevant for performing data analysis. Hence, you need to clean the data to remove unwanted, redundant, and missing values to make it ready for analysis.
- **Data Exploration and Analysis:** After you gather the right data, the next vital step is to execute exploratory data analysis. You can use data visualization and business intelligence tools, data mining techniques, and predictive modelling to analyse, visualize, and predict future outcomes from this data. Applying these methods can tell you the impact and relationship of a certain feature as compared to other variables.
- **Interpret the results:** The final step is to interpret the results and validate if the outcomes meet your expectations. You can find out hidden patterns and

future trends. This will help you gain insights that will support you with appropriate data-driven decision making.

DATA CLEANING

Before going to proceed to the visualization of our dataset, first we need to do the following things:

- We need to check the rows and columns for any empty cell in it .
- Next, we need to include any random data according to the information or we need to include dummy variables in it .

Now our data is completely filled and we can perform data visualization.

First and most important thing is to know the data of the dataset.

- The dataset contains 12 columns as follows:
- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulphur dioxide
- Total sulphur dioxide
- Density
- pH
- sulphates
- alcohol

The above list are the column data of the dataset which provides the characteristics of the wine quality, by using this data we will find:

Objectives:

The main objectives of the project are as follows:

(1) to experiment with different classification methods to see which yields the highest accuracy
(2) to determine the quality of the wine whether it is good or bad
(3) to determine the relation between each characteristic of the wine quality

Software libraries used:

- PANDAS
- NUMPY
- SEABORN
- MATPLOTLIB

PANDAS

Pandas is primarily used for data analysis, and it is one of the most commonly used Python libraries. It provides you with some of the most useful set of tools to explore, clean, and analyse your data. With Pandas, you can load, prepare, manipulate, and analyse all kinds of structured data. Machine learning libraries also revolve around Pandas Data Frames as an input.

NUMPY

NumPy is mainly used for its support for N-dimensional arrays. These multi-dimensional arrays are 50 times more robust compared to Python lists, making NumPy a favourite for data scientists.

NumPy is also used by other libraries such as TensorFlow for their internal computation on tensors. NumPy also provides fast precompiled functions for numerical routines, which can be hard to manually solve. To achieve better efficiency, NumPy uses array-oriented computations, so working with multiple classes becomes easy.

SEABORN

Built on the top of Matplotlib, Seaborn is an effective library for creating different visualizations.

One of the most important features of Seaborn is the creation of amplified data visuals. Some of the correlations that are not obvious initially can be displayed in a visual context, allowing Data Scientists to understand the models more properly.

Due to its customizable themes and high-level interfaces, it provides well-designed and extraordinary data visualizations, hence making the plots very attractive, which can, later on, be shown to stakeholders

MATPLOTLIB

- Matplotlib is a low-level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter
- Matplotlib is open source and we can use it freely
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and JavaScript for Platform compatibility

To use all these libraries, we need to install them by providing following code as:

- pip install pandas (for pandas)
- pip install NumPy (for NumPy)
- pip install matplotlib (for matplotlib)
- pip install seaborn (for seaborn)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Now that we have imported all the libraries necessary. By using pandas, we will assign the dataset to a variable because pandas can read and write the csv files.

```
dataset=pd.read_csv('winequality-red.csv')
dataset.head()
```

Python

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

dataset=pd.read_csv('winequality-red.csv')

this will be assigning the dataset to dataset as variable,

when we use dataset.head() as code as mentioned in the figure above we will get the information about the first 5 row and column data as above.

Like above we can print the last 5 row and column data by providing the code dataset.tail()

To get the datatype, index no. and column data and the space of memory used by the dataset we need to use code as shown in the following fig. and will get the output as shown.

Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string, it returns a different output which is shown in the examples below.

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

Pandas **describe()** is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string, it returns a different output which is shown in the figure below.

By using dataset.desciribe() we get the result as follows.

```
dataset.describe()
```

Python

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	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	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

Next, we need to check any null values in the data for confirmation we use the code as followed in figure:

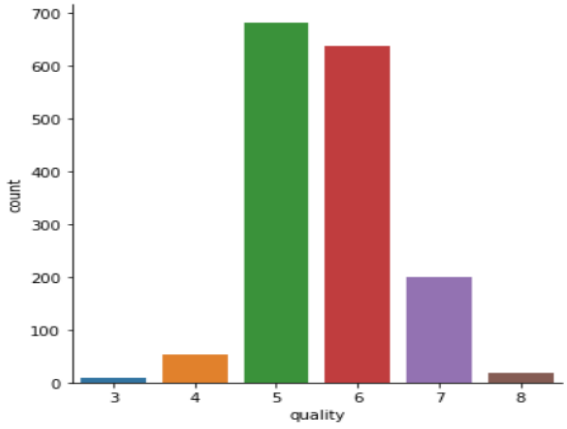
```
dataset.isnull().sum()
```

```
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
```

Data visualization

Data visualization is defined as a graphical representation that contains the information and the data. By using visual elements like charts, graphs, and maps, data visualization techniques provide an accessible way to see and understand trends, outliers, and patterns in data.

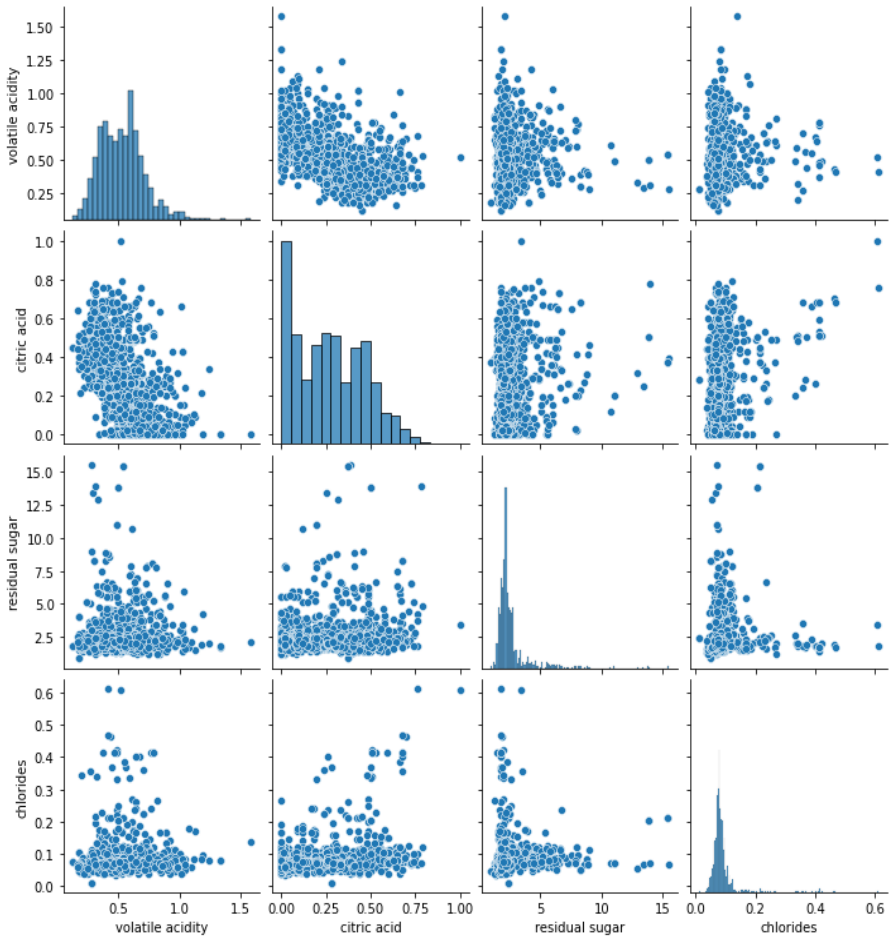
```
In [34]: #Data Visualization
         sns.catplot(x='quality',data=dataset,kind='count')

Out[34]: <seaborn.axisgrid.FacetGrid at 0x2c0b1cfffa0>
```



countplot() method is used to Show the counts of observations in each categorical bin using bars. Our wine scale ranges between 3 and 8. There are some really good wines (7-8), while most of them have an average quality between 5 and 6, and some of them are really poor (3-4).

Pair plot: -

```
In [35]: sns.pairplot(dataset.iloc[:,1:5],hue=None)
         plt.show()
```



The pair plot builds on two basic figures, the histogram and the scatter plot. The histogram on the diagonal allows us to see the distribution of a single variable while the scatter plots on the upper and lower triangles show the relationship (or lack thereof) between two variables.

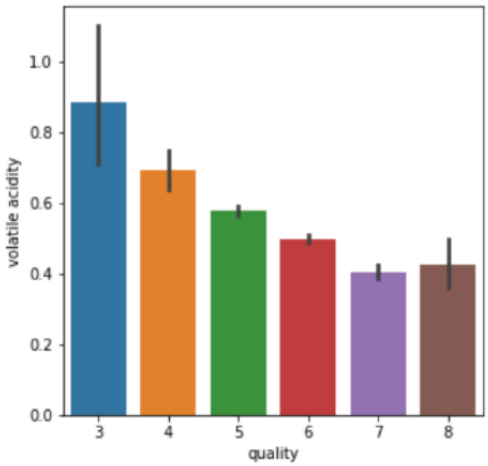
It plots every numerical attribute against every other.

In the above pair plot we create a grid of axes such that first five numeric variables in dataset will be shared across the y- axes across a single row and the x-axes across a single column.

Here we are comparing each 5 columns with one another to get a pairwise relationships in a dataset.

Bar plot (Volatile acidity Vs Quality): -

```
In [36]: plot=plt.figure(figsize=(5,5))
         sns.barplot(x='quality',y='volatile acidity',data=dataset)
Out[36]: <AxesSubplot:xlabel='quality', ylabel='volatile acidity'>
```

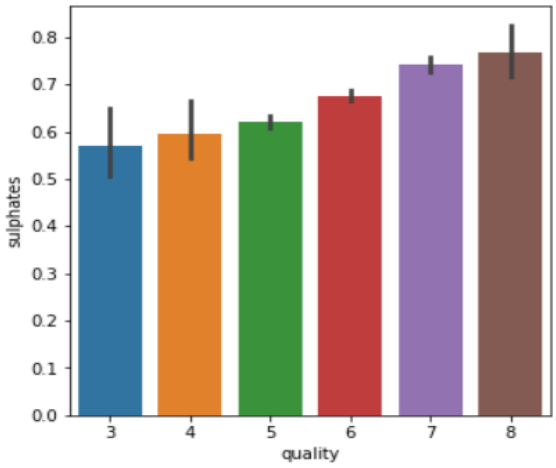


bar plot shows categorical data as rectangular bars with heights proportional to the value they represent. It is often used to compare between values of different categories in the data.

So, the above graph is the relation between volatile and quality when quality increases from 3 to 7 the volatile acidity decreases while there is a gradual increase in volatile acidity at 8 (quality).

Bar plot (Sulphates Vs Quality): -

```
In [37]: plot=plt.figure(figsize=(5,5))
         sns.barplot(x='quality',y='sulphates',data=dataset)
Out[37]: <AxesSubplot:xlabel='quality', ylabel='sulphates'>
```

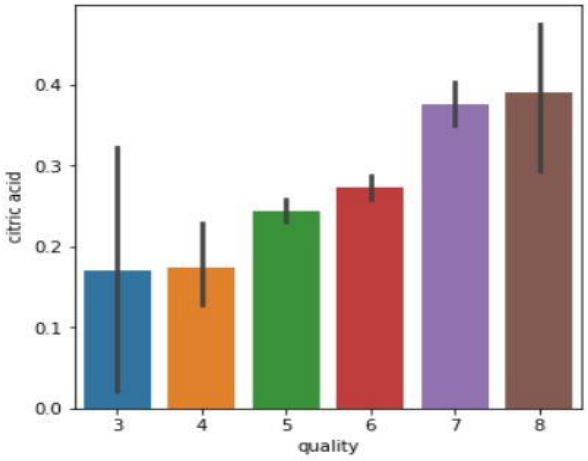


So, in this bar plot it shows the relation between quality and sulphates, so when the quality increases from 3 to 8 there is an increase in sulphates too. Thus, its directly proportional to each other, so more the sulphates better are the quality of our wine.

Bar plot (Sulphates Vs Quality): -

```
In [38]: plot=plt.figure(figsize=(5,5))
         sns.barplot(x='quality',y='citric acid',data=dataset)
```

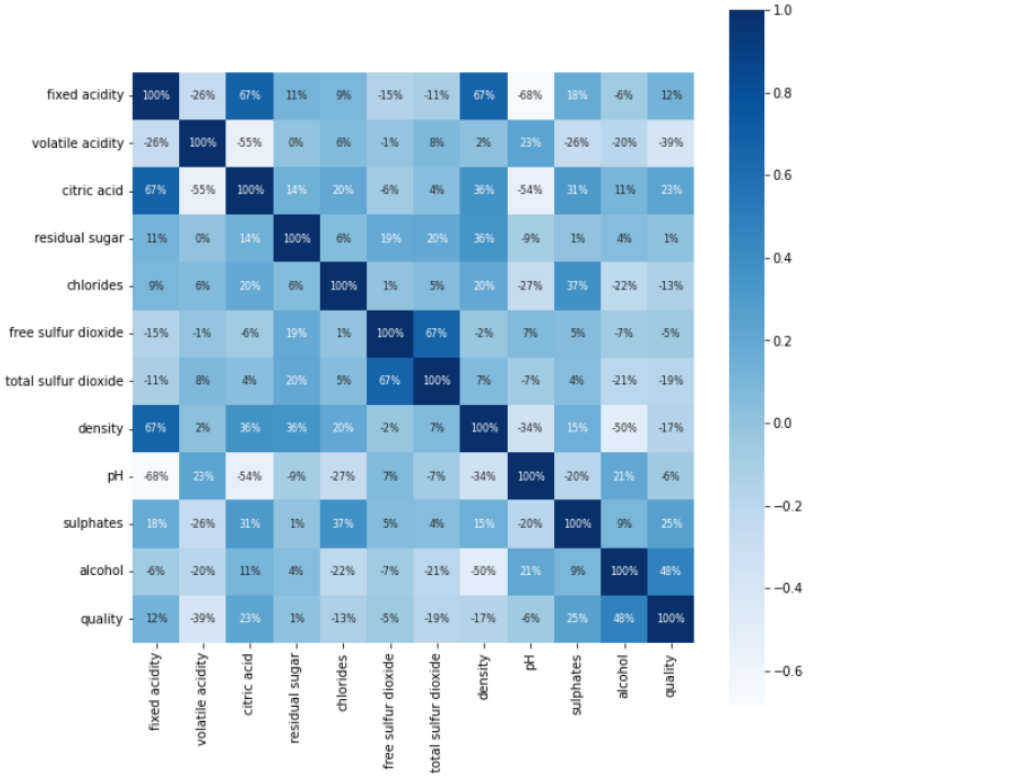
```
Out[38]: <AxesSubplot:xlabel='quality', ylabel='citric acid'>
```



So, in this bar plot it shows the relation between quality and citric acid, so when the quality increases from 3 to 8 there is an increase in citric acid too. Thus, its directly proportional to each other, so more the citric acid better is the quality of our wine.

Heatmap

```
In [51]: plt.figure(figsize=(10,10))
         sns.heatmap(dataset.corr(),cbar=True,square=True,annot=True,annot_kws={'size':8},cmap='Blues',fmt='.0%')
         plt.show()
```



Heatmap is a way to show some sort of matrix plot. To use a heatmap the data should be in a matrix form. By matrix we mean that the index name and the column name must match in some way so that the data that we fill inside the cells are relevant.

From this matrix we can observe, apart from the information we had before, some obvious feature correlations such as pH and acidity. Apart from that, we get to know the percentage of the correlations we obtained before. We can also observe that approximately half of these features correlate positively with quality while the other half correlate negatively.

From all these features, we are going to select the ones with bigger numbers since these are the ones that will give us more information. To do so we are going to stablish a minimum threshold of correlation approximately around 0.2 (absolute value) since we do not have to take into account features whose values might be redundant and not provide information at all.

Data processing: -

Data Processing is the task of converting data from a given form to a much more usable and desired form i.e., making it more meaningful and informative. -+

```
In [39]: #Data Preprocessing
         x=dataset.drop('quality',axis=1)
         y=dataset['quality'].apply(lambda y_values: 1 if y_values>=6 else 0)

In [40]: y.value_counts()

Out[40]: 1    855
         0    744
         Name: quality, dtype: int64
```

In data processing we can see that, for the x variable we have taken quality column and in y variable we have all the columns except the quality column and we are converting all the values that are above 6 in the quality column to 1 and the rest to 0 (1 indicating good quality and 0 indicating bad quality).

Splitting our data: -

We are keeping 20% of our dataset to treat it as testing data and 80% of our dataset as training data and be able to test the performance of our models. We are splitting our dataset in a way such that all of the wine qualities are represented proportionally equally in both training and testing dataset.

Other than that, the selection is being done randomly with uniform distribution. Various classification and regression algorithms are used to fit the model.

```
#Training and test split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
```

✓ 0.2s Python

Normalization: -

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

To normalize your data, you need to import the *Minmax Scalar* from the sklearn library and apply it to our dataset.

```
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
scalar.fit(x_train)
x_train=scalar.transform(x_train)
x_test=scalar.transform(x_test)
```

✓ 0.1s Python

Modelling: -

In this project we will be using different ml model for predicting the quality of the wine, like Logistic Regression, K Nearest Neighbour, Support Vector Machine (linear classifier), Gaussian Naïve Bayes, Decision Tree Classifier and Random Forest Classifier.

Model 1: Logistic Regression

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

```
#logistic regression
from sklearn.linear_model import LogisticRegression
log=LogisticRegression(random_state=0)
log.fit(x_train,y_train)
```

Model 2: K Nearest Neighbour

The k-nearest neighbours (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

```
#Kneighbors classifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
knn.fit(x_train, y_train)
```

Model 3: Support vector machine

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

```
#Svc linear
from sklearn.svm import SVC
svc_lin=SVC(kernel='linear',random_state=0)
svc_lin.fit(x_train, y_train)
```

Model 4: Gaussian Naïve Bayes

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = P(B/A) * P(A)/P(B).$$

```
#GaussianNB
from sklearn.naive_bayes import GaussianNB
gauss=GaussianNB()
gauss.fit(x_train, y_train)
```

Model 5: Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g., whether a coin flip comes up heads or tails), each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric **supervised learning** method used for both **classification** and **regression** tasks.

```
#decision Tree
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(criterion='entropy',random_state=0)
tree.fit(x_train, y_train)
```

Model 6: Random Forest Classifier

Random forests are an ensemble learning technique that builds off of decision trees. Random forests involve creating multiple decision trees using bootstrapped datasets of the original data and randomly selecting a subset of variables at each step of the decision tree. The model then selects the mode of all of the predictions of each decision tree. What's the point of this? By relying on a "majority wins" model, it reduces the risk of error from an individual tree.

```
#random forest
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)
forest.fit(x_train, y_train)
```

Training Accuracy: -

```
[0]Logistic Regression Training Accuracy: 0.7414512093411176
[1]K Nearest Neighbor Training Accuracy: 0.8148457047539617
[2]Support Vector Machine(Linear Classifier) Training Accuracy: 0.7397831526271893
[3]Gaussian Naive Bayes Training Accuracy: 0.7264386989157632
[4]Decision Tree Classifier Training Accuracy: 1.0
[5]Random Forest Classifier Training Accuracy: 0.9899916597164303
```

From the above results Decision Tree is having the highest training accuracy (100%) followed by Random Forest (98%).

Confusion Matrix: -

We import confusion matrix from sklearn library and print the confusion matrix for all the model and we also print the testing accuracy.

Here, TN is True Negative, TP-True Positive, FN-False Negative, FP-False Positive.

And Testing Accuracy is given by: -

(TP+TN)/(TP+TN+FN+FP)

```
from sklearn.metrics import confusion_matrix
for i in range(len(model)):
    cm=confusion_matrix(y_test,model[i].predict(x_test))
    TN=cm[0][0]
    TP=cm[1][1]
    FN=cm[1][0]
    FP=cm[0][1]
    print(cm)

    print('model[{}] Testing Accuracy = "{}!"'.format(i,(TP+TN)/(TP+TN+FN+FP)))
    print()
```

✓ 0.1s Python

Testing Accuracy: -

```
[[137  48]
 [ 52 163]]
model[0] Testing Accuracy = "0.75!"

[[112  73]
 [ 52 163]]
model[1] Testing Accuracy = "0.6875!"

[[143  42]
 [ 61 154]]
model[2] Testing Accuracy = "0.7425!"

[[129  56]
 [ 49 166]]
model[3] Testing Accuracy = "0.7375!"

[[129  56]
 [ 47 168]]
model[4] Testing Accuracy = "0.7425!"

[[156  29]
 [ 40 175]]
model[5] Testing Accuracy = "0.8275!"
```

Random forest classifier has the highest testing accuracy about 82% compared to different model. Then it is followed by decision tree and support machine vector about 74%.

CLASSIFICATION REPORT AND ACCURACY SCORE

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

for i in range(len(model)):
    print('Model',i)
    #check precision, recall, f1-score
    print(classification_report(y_test, model[i].predict(x_test)))
    #Another way to get the models accuracy on the test data
    print(accuracy_score(y_test,model[i].predict(x_test)))
    print()
```

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Support - Support is nothing but the number of test samples available for testing.

Macro Average - Compute the average without considering the proportion.
Weighted Average - Compute the average considering the proportion

Let’s have a look at each Model:

MODEL 0: Logistic regression

```
Model 0
              precision    recall  f1-score   support

           0       0.72      0.74      0.73       185
           1       0.77      0.76      0.77       215

    accuracy                           0.75       400
   macro avg       0.75      0.75      0.75       400
weighted avg       0.75      0.75      0.75       400

0.75
```

The logistic regression model has an accuracy of 0.75.

MODEL 1: K Nearest Neighbour

```
Model 1
              precision    recall  f1-score   support

           0       0.68      0.61      0.64       185
           1       0.69      0.76      0.72       215

    accuracy                           0.69       400
   macro avg       0.69      0.68      0.68       400
weighted avg       0.69      0.69      0.69       400

0.6875
```

K Nearest Neighbour model has an accuracy of 0.6875.

MODEL 2: Support Vector Machine (Linear Classifier)

```
Model 2
              precision    recall  f1-score   support

           0       0.70      0.77      0.74       185
           1       0.79      0.72      0.75       215

    accuracy                           0.74       400
   macro avg       0.74      0.74      0.74       400
weighted avg       0.75      0.74      0.74       400

0.7425
```

Support Vector Machine (Linear Classifier) model has an accuracy of 0.7425.

MODEL 3: Gaussian Naive Bayes

```
Model 3
              precision    recall  f1-score   support

           0       0.72      0.70      0.71       185
           1       0.75      0.77      0.76       215

    accuracy                           0.74       400
   macro avg       0.74      0.73      0.74       400
weighted avg       0.74      0.74      0.74       400

0.7375
```

Gaussian Naive Bayes model has an accuracy of 0.7375

MODEL 4: Decision Tree Classifier

```
Model 4
              precision    recall  f1-score   support

           0       0.73      0.70      0.71       185
           1       0.75      0.78      0.77       215

    accuracy                           0.74       400
   macro avg       0.74      0.74      0.74       400
weighted avg       0.74      0.74      0.74       400

0.7425
```

The Decision Tree Classifier model has an accuracy of 0.7425.

MODEL 5: Random Forest Classifier

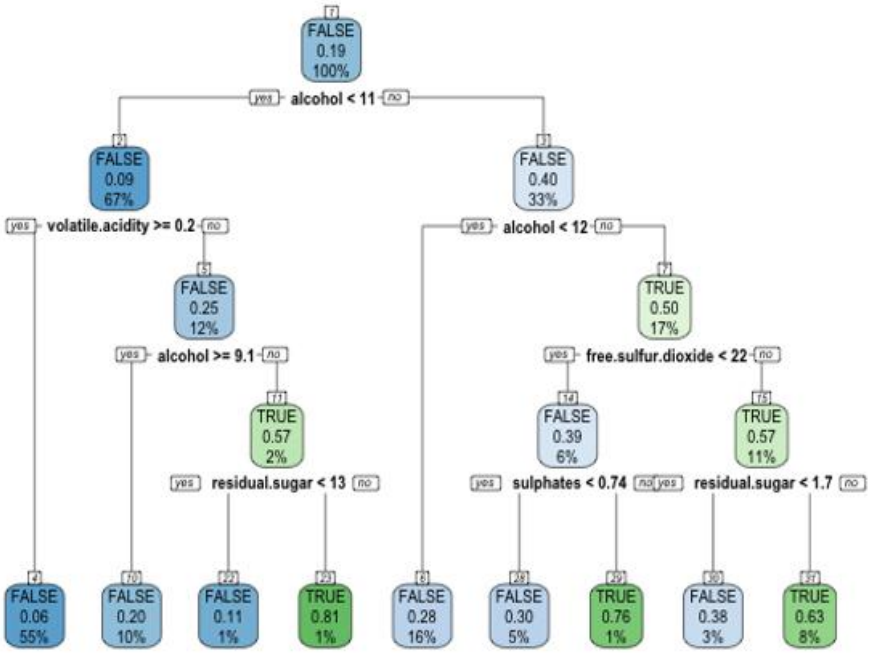
```
Model 5
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       185
           1       0.86      0.81      0.84       215

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400

0.8275
```

Random Forest Classifier model has an accuracy of 0.8275

Now by comparing all the models with each other, we see that the Random Forest Classifier model has the most accuracy which is (0.8275).

Prediction Random Forest Classifier Model

Random Forest is used to improve the performance of Decision trees by aggregating many decision trees to give an optimal model that is not susceptible to overfitting. In Decision trees, splitting is done using a complete set of features but in Random Forest only a random subset of features is used for splitting. As it builds many trees with less correlation between the trees this would reduce the variance of the optimal model tree.



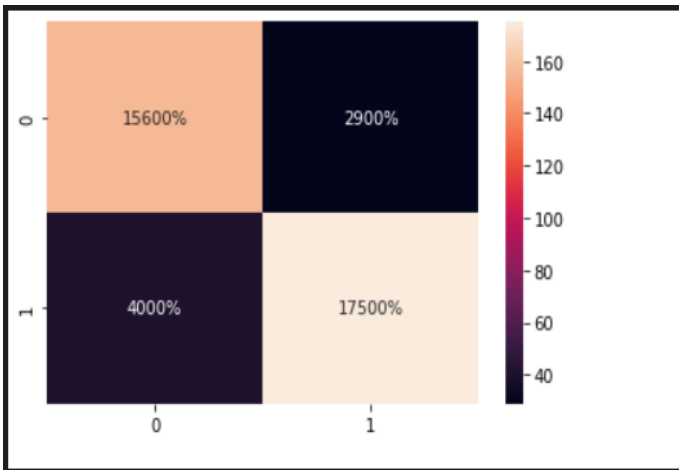
From the above decision tree, if the percent of alcohol level is greater than 11, residual sugar greater than 1.7, free sulphur dioxide greater than 22 and sulphates level greater than 0.74 then the quality of the wine is said to be higher. Random Forest takes this to the next level by combining hundreds of decision trees.

```
#Print Prediction of Random Forest Classifier Model
y_pred=model[5].predict(x_test)
print(y_pred)
print()
print(y_test)
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt='.0%')
```

```
[1 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 1 1 0
 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0
 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0 1
 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1
 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1
 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0 1 1 0 0 1
 1 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0
 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 0 1
 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0
 1 0 1 1 0 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0
 1 0 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 0 1 1 1 0 0 0 0]
```

```
1109    1
1032    0
1002    1
487     1
979     0
       ..
489     1
362     0
526     0
442     1
1229    0
Name: quality, Length: 400, dtype: int64
```

This the heatmap from this we get to know that 15600% values were predicted correct for 1 and 2900% were predicted falsely and 17500% values were predicted correct and 40 were predicted falsely.

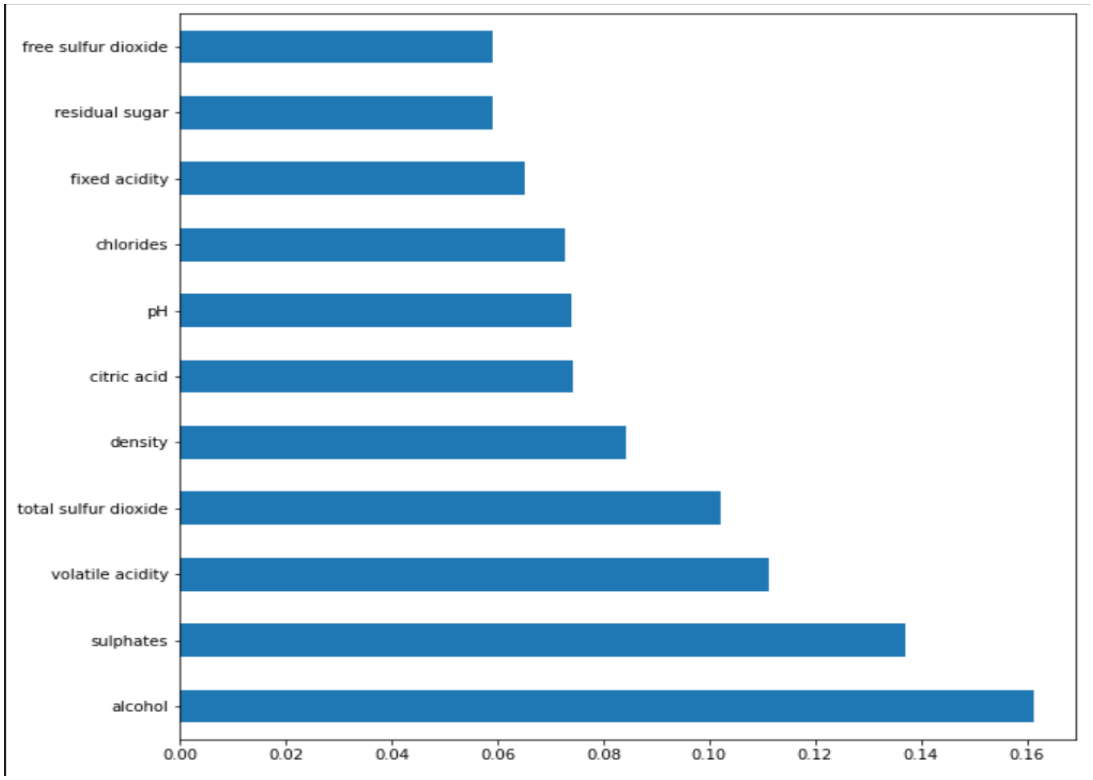


Feature Importance: -

Below, I graphed the feature importance based on the Random Forest model and the Decision Tree model. While they slightly vary, the top 3 features are the same: alcohol, volatile acidity, and sulphates. If you look below the graphs, I split the dataset into good quality and bad quality to compare these variables in more detail.

Random Forest: -

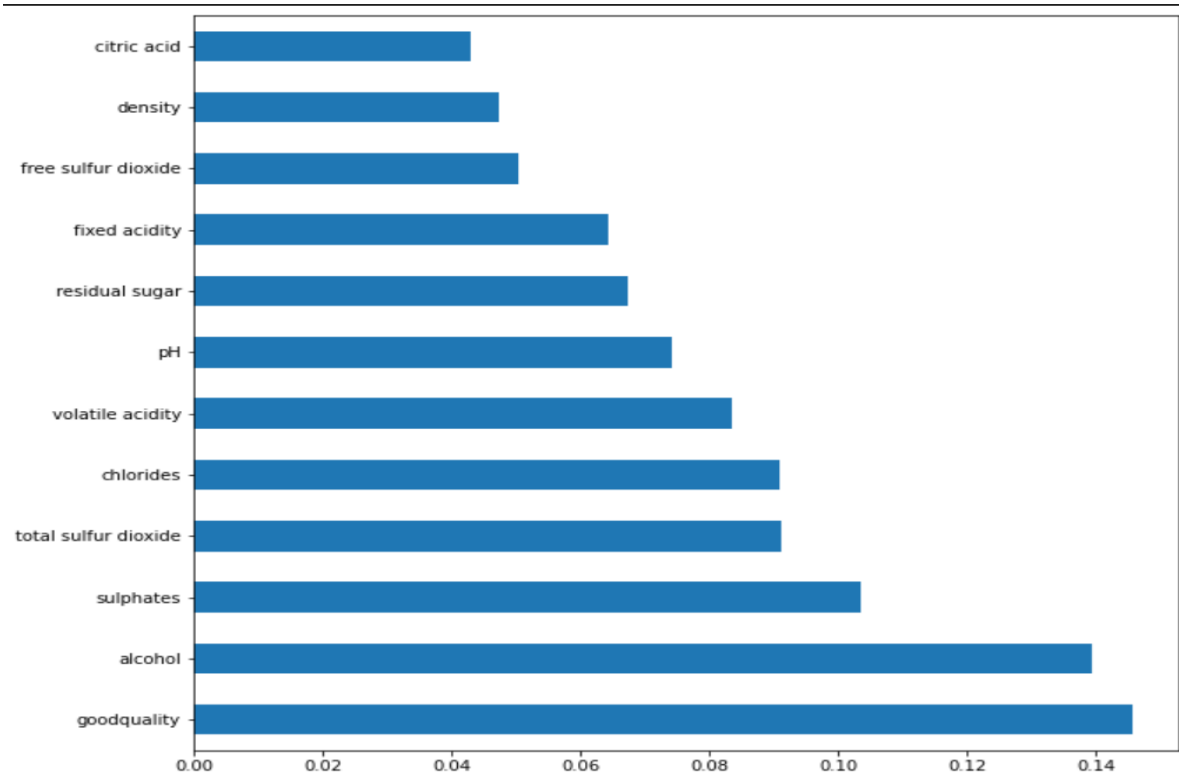
```
feat_importances = pd.Series(model[5].feature_importances_, index=x.columns)
feat_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```



Decision Tree: -

```
feat_importances = pd.Series(model[4].feature_importances_, index=x.columns)
feat_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```

✓ 0.4s



Comparing the Top 4 Features: -

```
#Filtering df for only good quality
df_temp = dataset[dataset['quality']>=6 ]
df_temp.describe()
# Filtering df for only bad quality
df_temp2 = dataset[dataset['quality']<=5]
df_temp2.describe()
```

✓ 0.2s

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	
count	855.000000	855.000000	855.000000	855.000000	855.000000	855.000000	855.000000	855.000000	855.000000	855.000000	8
mean	8.474035	0.474146	0.299883	2.535965	0.082661	15.272515	39.352047	0.996467	3.310643	0.692620	
std	1.862795	0.161999	0.199889	1.424835	0.037258	10.038538	27.253280	0.002067	0.154554	0.155558	
min	4.700000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.860000	0.390000	
25%	7.100000	0.350000	0.115000	1.900000	0.067000	7.000000	20.000000	0.995185	3.210000	0.590000	
50%	8.000000	0.460000	0.310000	2.200000	0.077000	13.000000	33.000000	0.996400	3.310000	0.660000	
75%	9.650000	0.580000	0.460000	2.600000	0.087500	20.500000	50.000000	0.997685	3.400000	0.770000	
max	15.600000	1.040000	0.780000	15.400000	0.415000	72.000000	289.000000	1.003690	4.010000	1.950000	

Good Quality

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
count	744.000000	744.000000	744.000000	744.00000	744.000000	744.000000	744.000000	744.000000	744.000000	744.000000
mean	8.142204	0.589503	0.237755	2.54207	0.092989	16.567204	54.645161	0.997068	3.311653	0.618535
std	1.572396	0.177956	0.183368	1.39355	0.055781	10.890291	36.720468	0.001598	0.154296	0.176194
min	4.600000	0.180000	0.000000	1.20000	0.039000	3.000000	6.000000	0.992560	2.740000	0.330000
25%	7.100000	0.460000	0.080000	1.90000	0.074000	8.000000	23.750000	0.996120	3.200000	0.520000
50%	7.800000	0.590000	0.220000	2.20000	0.081000	14.000000	45.000000	0.996935	3.310000	0.580000
75%	8.900000	0.680000	0.360000	2.60000	0.094000	23.000000	78.000000	0.997900	3.400000	0.650000
max	15.900000	1.580000	1.000000	15.50000	0.611000	68.000000	155.000000	1.003150	3.900000	2.000000

Bad Quality

CONCLUSION

Based on the bar plots plotted we come to a conclusion that not all input features are essential and affect the data, for example from the bar plot against quality and residual sugar we see that as the quality increases residual sugar is moderate and does not have change drastically. So, this feature is not so essential as compared to others like alcohol and citric acid, so we can drop this feature while feature selection. For classifying the wine quality, we have implemented multiple algorithms, namely 1) Logistic Regression 2) K Nearest Neighbour 3) Support Vector Classifier 4) Gaussian Naïve Bayes 5) Decision Tree Classifier 6) Random Forest Classifier. We were able to achieve maximum accuracy for training using random forest of 98% and testing accuracy of 82% followed by decision tree which gave us training accuracy of 100% and testing accuracy of 74%.