Contents

BONAFIDE CERTIFICATE

Certified that this project report "Predicting Wine Quality using Wine Quality Dataset" is the bonafide work of **Subhadeep Mandal** and **Arman Singh Grewal** who carried out the project work under my supervision.

Signature of the Supervisor	:	
Name of supervisor	:	Dr. Dhanpratap Singh
Academic Designation	:	Associate professor
ID of Supervisor	:	25706
Department of Supervisor	:	Intelligence System 1

Place : Phagwara, Punjab
Date : 08.11.2022

.

1 Introduction

The quality of the wine is a very important part for the consumers as well as the manufacturing industries. Industries are increasing their sales using product quality certification. Nowadays, all over the world wine is a regularly used beverage and the industries are using the certification of product quality to increases their value in the market. Previously, testing of product quality will be done at the end of the production, this is time taking process and it requires a lot of resources such as the need for various human experts for the assessment of product quality which makes this process very expensive. Every human has their own opinion about the test, so identifying the quality of the wine based on humans experts it is a challenging task.

2 Description of Project

This research aims to what wine features are important to get the promising result by implementing the machine learning classification algorithms such as Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree(DT) and Softmax Regression using the wine quality

The wine quality dataset is publically available on the UCI machine learning repository (Cortez et al., 2009). The dataset has two files red wine and white wine variants of the Portuguese "Vinho Verde" wine. In this project we are focusing on only the red wine dataset which contains 1599 instances. It contains 11 input features and 1 output feature. Input features are based on the physicochemical tests and output variable based on sensory data is scaled in 11 quality classes from 0 to 10 (0-very bad to 10-very good).

Feature selection is the popular data preprocessing step to build the model, it selects the subset of relevant features and reduces time complexity. Features are sorted according to the weighing of the relevance, and relatively low weighting features will be removed. This process will simplify the model and reduce the training time, and increase the performance of the model. We pay attention to feature selection is also the study direction. To evaluate our model, accuracy, precision, recall, and f1 score are good indicators to evaluate the performance of the model.

3 Contributions

- **Subhadeep Mandal** : Preparation of the Report, Data processing & plotting
- **Arman Singh Grewal** : Built various models and Generated Insights

3.1 Project Link

https://github.com/Ultrasubha/Predicting-Wine-Quality

4 Implementation

1. First we loaded necessary libraries
2. Then imported the dataset from UCI ML website
3. Then we did some exploratory data analysis on our data, steps written alongside code.
4. Separated features and labels. First 11 columns are the input to our model so we separate them.
   The final column (quality) is our target/label/output
5. Then we split the data into Training and Test sets. 20

5 Inner Workings

5.1 Conversion to binary classification problem

Our target label was a 6 class setup ranging from 3-8. 3 was a very low quality wine and 8 was good quality wine. We converted this multiclass classification problem into a binary classification problem by converting those 6 classes to only 2 - Bad quality or Good quality. Threshold selected was '7', if wine quality ¡ 7 then it is bad wine, and if it is around 7 or 8 then it is considered a good quality wine.

Then we compared the performance of 7 built-in classification models on our wine quality dataset.

5.2 Standardisation

The data standardization technique can scale the features among 0 and 1, it will be useful for learning the model, by applying it to all the numeric features and then separating data by standard derivation. So, we use this technique to standardize the data.

$$z_i = \frac{x_i - \mu}{\sigma}$$

σ is the standard derivation, x_i is each value, and μ is the mean value of the array of features x Sklearn.preprocessing.StandardScaler() does this job for us

5.3 Pipeline

We also used sklearn.pipeline utility which is used to make a composite estimator, a chain of transformers and estimators. Finally compared all models based on their sklearn.metrics.classification report

1 Importing necessary libraries

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

2 Loading the dataset

```
data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
                   , delimiter=";")
data
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

```
# Names of all the columns in features
print(data.columns)
```

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

```
# Checking if there are any non-null values in dataset
# and also confirming that columns have right datatype
data.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
```

Conclusions ->

Sr. no.	Name	Description
1	**fixed acidity**	Primary fixed acids found in wine are tartaric, succinic, citric, and malic

Sr. no.	Name	Description
2	**volatile acidity**	Volatile acidity is the gaseous acids present in wine.
3	**citric acid**	It is weak organic acid, found in citrus fruits naturally.
4	**residual sugar**	Amount of sugar left after fermentation.
5	**chlorides**	Amount of salt present in wine.
6	**free sulphur dioxide**	So2 used for prevention of wine by oxidation and microbial spoilage.
7	**total sulphur dioxides**	Sum of free and combined SO2
8	**density**	thickness of wine
9	**pH**	pH is used for checking acidity
10	**sulphates**	Added sulfites preserve freshness and protect wine from oxidation, and bacteria.
11	**alcohol**	Percent of alcohol present in wine.
12	**quality**	Target label with 6 classes in range 3-8

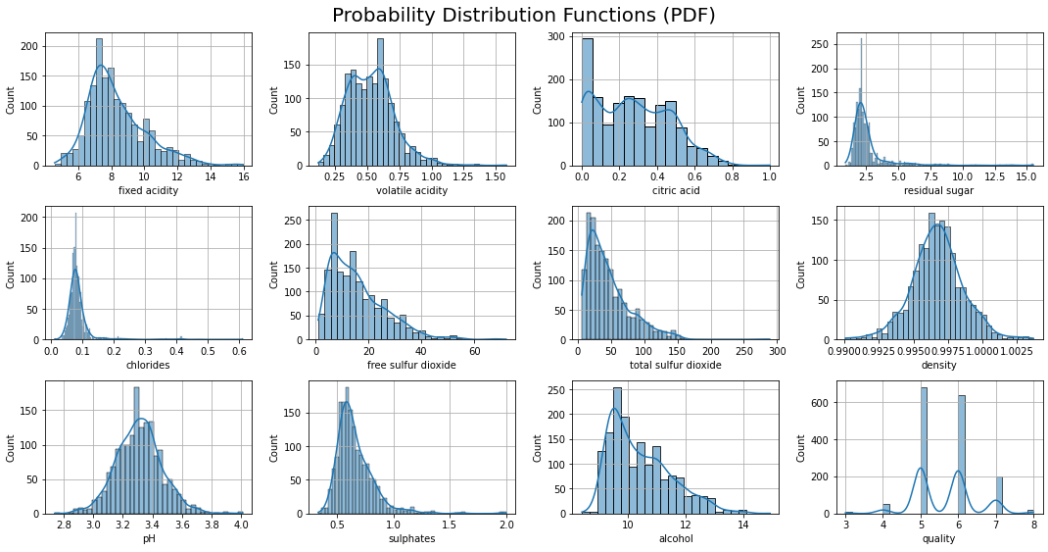
- This is a multiclass classification problem, as labels are discrete.
- There are 11 features (attributes).
- There are no null values, so we don't need to do Data manipulation
-

3 Exploratory data analysis

Exploratory data analysis (EDA) is an approach to analyze data sets to summarize their main characteristics, often with visual methods.

- It helps us to uncover the underlying structure of data and its dynamics through which we can maximize the insights.
- EDA is also critical to extract important variables and detect outliers and anomalies.
- EDA is considered one of the most critical parts to understand the data.

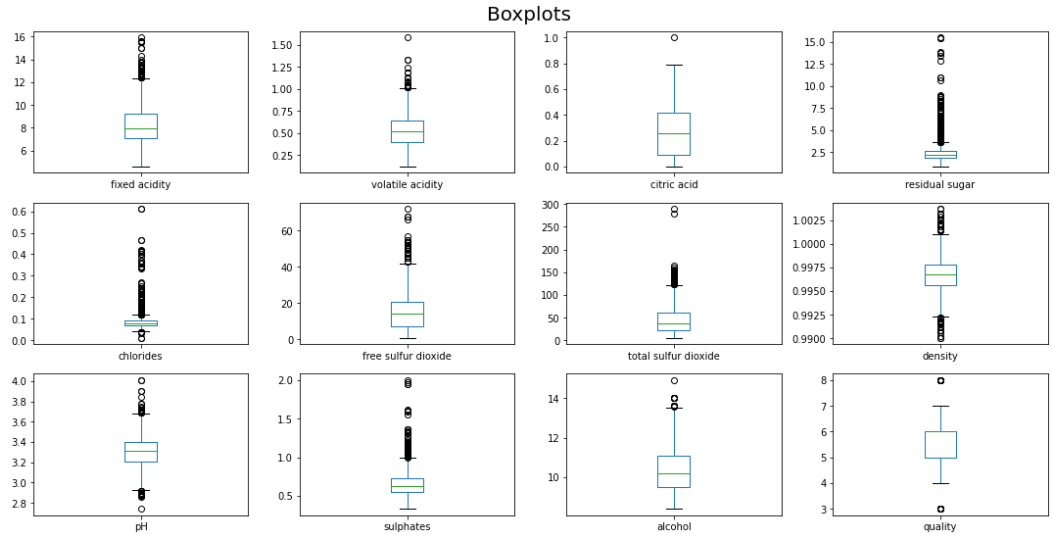
```
# features.hist(figsize=(15,10), bins=20);

# Plotting the histplot + kde of all the featues to check for outliers
plt.figure(figsize=(15, 10))
for i, col in enumerate(data.columns.values):
    plt.subplot(4, 4, i+1)
    sns.histplot(data[col], kde=True)
    plt.grid(); plt.tight_layout()
    plt.suptitle("Probability Distribution Functions (PDF)", size=20)
```



> We can conclude from above KDE plots that **pH** and **density** attributes have near normal disribuiton.

> All other attributes are positively skewed

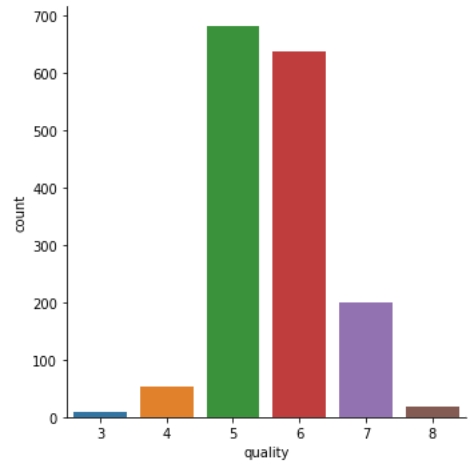
```
# Plotting the boxplot of all the featues to check for outliers
plt.figure(figsize=(15, 10))
for i, col in enumerate(data.columns.values):
    plt.subplot(4, 4, i+1)
    data.boxplot(col)
    plt.grid(); plt.tight_layout()
    plt.suptitle("Boxplots", size=20)
```



> We can conclude form the above boxplots that all columns have outliers

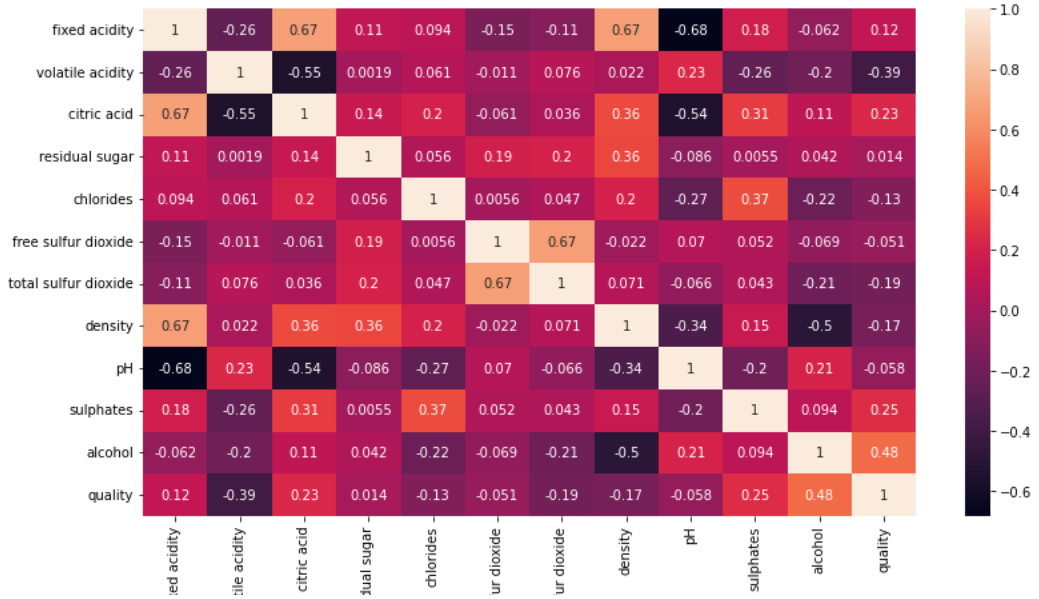
```
# Visualizing the count of all 6 classes of target label
sns.catplot(x='quality', data = data, kind = 'count')
```

```
<seaborn.axisgrid.FacetGrid at 0x14a4d3f40>
```



> “quality” has a high number of values in categories 5 and 6.

```
# Checking the correlation between features
plt.figure(figsize=(13, 7))
sns.heatmap(data.corr(), annot=True, color='k');
```



4 Data Preprocessing

Dividing the data into test and training sets

```
# separate the feature and Label
X = data.drop('quality',axis=1)
y = data['quality'].copy()
```

```
# from sklearn.preprocessing import LabelBinarizer
# y = LabelBinarizer().fit_transform(y)
y = data['quality'].apply(lambda y_value: 1 if y_value>=7 else 0)
# y
```

```
# Splitting the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)
```

Building various models

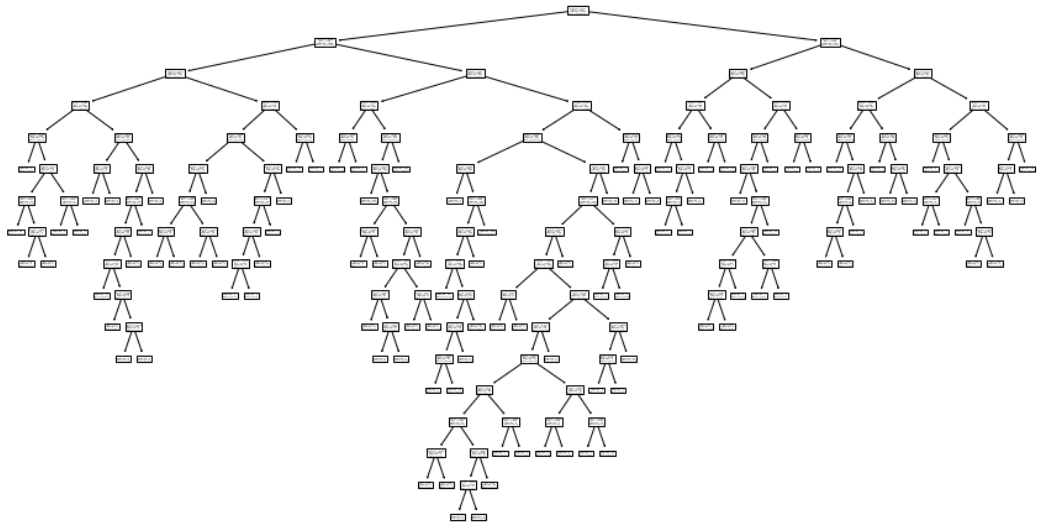
```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
final_scores = {}
```

1 Decision tree

```
from sklearn.tree import DecisionTreeClassifier

pipe1 = make_pipeline(StandardScaler(), DecisionTreeClassifier(random_state=42))
pipe1.fit(X_train, y_train)
y_pred1 = pipe1.predict(X_test)
print(classification_report(y_test, y_pred1))
final_scores['DecisionTree']  = pipe1.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.92      0.93      0.93       273
               1       0.57      0.51      0.54        47

        accuracy                           0.87       320
       macro avg       0.74      0.72      0.73       320
    weighted avg       0.87      0.87      0.87       320
```

```
# Visualizing the decision tree
from sklearn import tree
fig = plt.figure(figsize=(15,8))
tree.plot_tree(pipe1['decisiontreeclassifier']);
# fig.savefig('dtree.png')
```



2 Random Forest

```
from sklearn.ensemble import RandomForestClassifier

pipe2 = make_pipeline(StandardScaler(), RandomForestClassifier(random_state=1))
pipe2.fit(X_train, y_train)
y_pred2 = pipe2.predict(X_test)
print(classification_report(y_test, y_pred2))
final_scores['RandomForest'] = pipe2.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.93      0.97      0.95       273
               1       0.79      0.55      0.65        47

        accuracy                           0.91       320
       macro avg       0.86      0.76      0.80       320
    weighted avg       0.91      0.91      0.91       320
```

3 AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier

pipe3 = make_pipeline(StandardScaler(), AdaBoostClassifier(random_state=1))
pipe3.fit(X_train, y_train)
y_pred3 = pipe3.predict(X_test)
print(classification_report(y_test, y_pred3))
final_scores['AdaBoost'] = pipe3.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.89      0.93      0.91       273
               1       0.45      0.32      0.38        47

        accuracy                           0.84       320
       macro avg       0.67      0.63      0.64       320
    weighted avg       0.82      0.84      0.83       320
```

4 Gradient Boost

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
pipe4 = make_pipeline(StandardScaler(), GradientBoostingClassifier(random_state=1))
pipe4.fit(X_train, y_train)
y_pred4 = pipe4.predict(X_test)
print(classification_report(y_test, y_pred4))
final_scores['GradBoost'] =  pipe4.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.90      0.95      0.93       273
               1       0.59      0.40      0.48        47

        accuracy                           0.87       320
       macro avg       0.75      0.68      0.70       320
    weighted avg       0.86      0.87      0.86       320
```

5 Naive Bayes classifier

```
from sklearn.naive_bayes import GaussianNB

pipe5 = make_pipeline(StandardScaler(), GaussianNB())
pipe5.fit(X_train, y_train)
y_pred5 = pipe5.predict(X_test)
print(classification_report(y_test, y_pred5))
final_scores['NaiveBayes'] =  pipe5.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.96      0.86      0.91       273
               1       0.49      0.79      0.60        47

        accuracy                           0.85       320
       macro avg       0.72      0.82      0.75       320
    weighted avg       0.89      0.85      0.86       320
```

6 Support Vector Classifier

```
from sklearn.svm import SVC

pipe6 = make_pipeline(StandardScaler(), SVC())
pipe6.fit(X_train, y_train)
y_pred6 = pipe6.predict(X_test)
print(classification_report(y_test, y_pred6))
final_scores['SupportVector'] = pipe6.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.88      0.98      0.93       273
               1       0.71      0.26      0.37        47

        accuracy                           0.88       320
       macro avg       0.80      0.62      0.65       320
    weighted avg       0.86      0.88      0.85       320
```

7 Logistic regression

```
from sklearn.linear_model import LogisticRegressionCV

pipe7 = make_pipeline(StandardScaler(), LogisticRegressionCV(random_state=1))
pipe7.fit(X_train, y_train)
y_pred7 = pipe7.predict(X_test)
print(classification_report(y_test, y_pred7))
final_scores['LogisticRegression'] = pipe7.score(X_test, y_test)
```

```
                  precision    recall  f1-score   support

               0       0.89      0.97      0.92       273
               1       0.59      0.28      0.38        47

        accuracy                           0.87       320
       macro avg       0.74      0.62      0.65       320
```

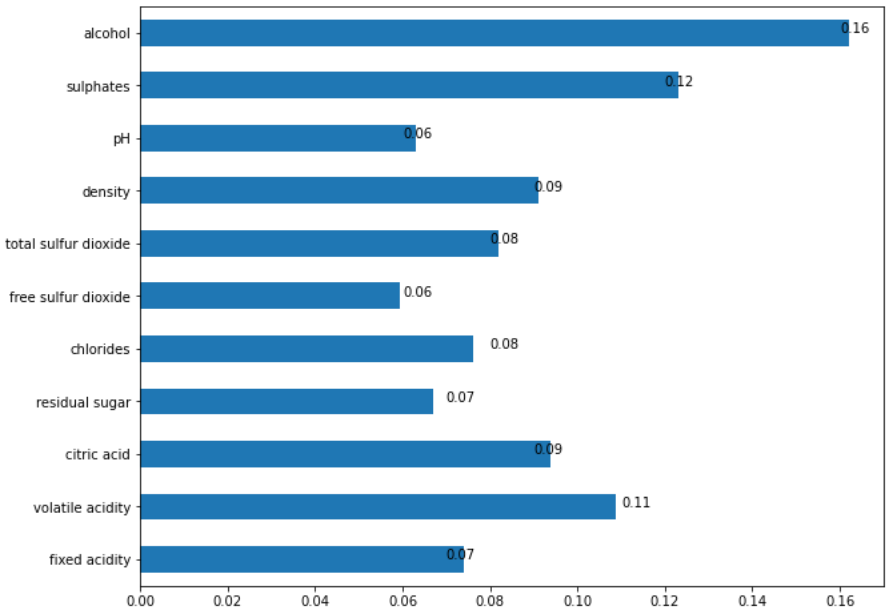
```
weighted avg       0.84      0.87      0.84       320
```

```
# Comparing 7 of above models -
final_scores = dict(sorted(final_scores.items(), key=lambda item: item[1], reverse=True))
final_scores
```

```
{'RandomForest': 0.9125,
 'SupportVector': 0.875,
 'DecisionTree': 0.871875,
 'GradBoost': 0.871875,
 'LogisticRegression': 0.865625,
 'NaiveBayes': 0.846875,
 'AdaBoost': 0.84375}
```

Selecting best features

```
feat_importances = pd.Series(pipe2['randomforestclassifier'].feature_importances_,
                             index= X_train.columns)
feat_importances.plot(kind='barh',figsize=(10,8))

for i, v in enumerate(round(feat_importances, 2)):
    plt.text(v, i, str(v))
```



CONCLUSION -

So, alcohol content is the most important feature in predicting the quality of wine

6 Technologies & Frameworks used

- **DECISION TREE** - Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

  A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

- **RANDOM FOREST** - 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.

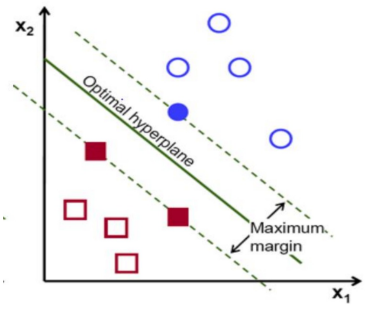
- **ADABOOST** - AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps.

  What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

- **NAIVE BAYES** -The naive Bayesian is the simple supervised machine learning classification algorithm based on the Bayes theorem. The algorithm assumes that the feature conditions are independent of the given class. The naive Bayes algorithm helps to build fast machine learning models that can make a fast prediction. The algorithm finds whether a particular portion has a spot by a particular class it utilizes the probability of likelihood.

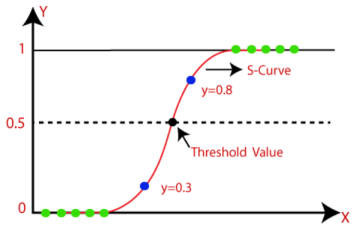
- **SUPPORT VECTOR CLASSIFIER** - The support vector machine (SVM) is the most popular and most widely used machine learning algorithm. It is a supervised learning model that can perform classification

and regression tasks. However, it is primarily used for classification problems in machine learning The SVM algorithm aims to create the best line or decision boundary that can separate n-dimensional space into classes. So we can put the new data points easily in the correct groups. This best decision boundary is called a hyperplane. The support vector machine selects the extreme data points that helping to create the hyperplane.



The SVM model is used for both non-linear and linear data. It uses a nonlinear mapping to convert the main preparing information into a higher measurement, and this is called Kernel SVM, although we didn't need to use it in this case. The model searches for the linear optimum splitting hyperplane in this new measurement. A hyperplane can split the data into two classes with an appropriate nonlinear mapping to suitably high measurements and for the finding, this hyperplane SVM uses the support vectors and edges. The SVM model is a representation of the models as a point in space, the different classes are isolated by the gap to mapped with the aim that instances are wide as would be careful. The model can perform out a nonlinear form of classification

- **LOGISTIC REGRESSION** - Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.



It predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or

No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

7 SWOT Analysis

(Metrics used to evaluate the models) - The performance measurement is calculated and evaluate the techniques to detect the effectiveness and efficiency of the model. There are four ways to check the predictions are correct or incorrect:

- **True Positive :** Number of samples that are predicted to be positive which are truly positive.
- **False Positive :** Number of samples that are predicted to be positive which are truly negative.
- **False Negative :** Number of samples that are predicted to be negative which are truly positive.
- **True Negative :** Number of samples that are predicted to be negative which are truly negative.

Below listed techniques, we use for the evaluation of the model.

1. **Accuracy** – Accuracy is defined as the ratio of correctly predicted observation to the total observation. The accuracy can be calculated easily by dividing the number of correct predictions by the total number of predictions.

$$Accuracy = \left(\frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + FalseNegative + TrueNegative} \right)$$

2. **Precision** – Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \left(\frac{TruePositive}{TruePositive + FalsePositive} \right)$$

3. **Recall** – Recall is defined as the ratio of correctly predicted positive observations to all observations in the actual class. The recall is also known as the True Positive rate calculated as,

$$Recall = \left(\frac{TruePositive}{TruePositive + FalseNegative} \right)$$

4. **F1 Score** – F1 score is the weighted average of precision and recall. The f1 score is used to measure the test accuracy of the model. F1 score is calculated by multiplying the recall and precision is divided by the recall and precision, and the result is calculated by multiplying two.

$$F1score = 2 \times \left(\frac{Recall \times Precision}{Recall + Precision} \right)$$

Accuracy is the most widely used evaluation metric for most traditional applications. But the accuracy rate is not suitable for evaluating imbalanced data sets, because many experts have observed that for extremely skewed class distributions, the recall rate for minority classes is typically 0, which means that no classification rules are generated for the minority class. Using the terminology in information retrieval, the precision and recall of the minority categories are much lower than the majority class. Accuracy gives more weight to the majority class than to the minority class, this makes it challenging for the classifier to implement well in the minority class.

For this purpose, additional metrics are coming into widespread usage

The F1 score is the popular evaluation metric for the imbalanced class problem. F1 score combines two matrices: precision and recall. Precision state how accurate the model was predicting a certain class and recall state that the opposite of the regrate misplaced instances which are misclassified. Since the multiple classes have multiple F1 scores. By using the unweighted mean of the F1 scores for our final scoring. We want our models to get optimized to classify instances that belong to the minority side, such as wine quality of 5 to 6 equally well with the rest of the qualities that are represented in a larger number.

8 Bibliography

References

[1] DECISION TREE :
https://www.geeksforgeeks.org/decision-tree/

[2] RANDOM FOREST :
https://www.javatpoint.com/machine-learning-random-forest-algorithm

[3] ADABOOST :
https://www.analyticsvidhya.com/adaboost-algorithm-a-complete-guide-for-beginners/

[4] NAIVE BAYES :
https://www.javatpoint.com/machine-learning-naive-bayes-classifier

[5] LOGISTIC REGRESSION :
https://www.javatpoint.com/logistic-regression-in-machine-learning