Title: Wine Quality Prediction using Random Forest Classifier

Author: Swaroop Suryakar (BE-01)

Nishant Wagh (BE-12)

Vaibhav Kale (BE-16)

Rohit Bhavsar (BE-17)

Date: 23/10/2023

Abstract:

The objective of this project is to predict the quality of wine based on its physicochemical properties using a Random Forest Classifier. Wine quality is a complex trait that depends on a variety of factors, including grape variety, terroir, winemaking techniques, and more. However, it has been observed that certain physicochemical properties of wine, such as acidity, sugar content, and alcohol level, can significantly influence its quality.

In this project, we propose a machine learning model based on the Random Forest Classifier to predict wine quality. The Random Forest Classifier is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes of the individual trees. This method is particularly suited for this task due to its ability to handle high dimensional data and its robustness against overfitting.

The dataset used in this project consists of red and white variants of Portuguese “Vinho Verde” wine. The dataset includes physicochemical properties such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol content and quality score (0-10).

The performance of the model will be evaluated using standard metrics such as accuracy, precision, recall and F1-score. The results of this project could provide valuable insights for winemakers and connoisseurs alike in understanding the factors that contribute to wine quality and in making informed decisions about wine production and selection.

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 analyze 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 can be used are

1) Random Forest 2) Stochastic Gradient Descent

3) SVC 4) Logistic Regression.

1. Introduction:

1.1 Background:

The quality of wine is a subject of great interest in the wine industry and among consumers. It is a complex trait that is influenced by a multitude of factors, including the type of grape, the climate and soil in which it is grown, and the techniques used in its fermentation. However, it has been observed that certain physicochemical properties of wine can significantly influence its quality.

In recent years, machine learning techniques have been increasingly applied to predict wine quality based on these physicochemical properties. These techniques offer the potential to provide objective, consistent, and cost-effective assessments of wine quality, which could be beneficial for both winemakers and consumers.

Among the various machine learning techniques available, the Random Forest Classifier has emerged as a particularly promising tool for this task. The Random Forest Classifier is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes of the individual trees. This makes it particularly suited for tasks such as wine quality prediction, where the target variable (wine quality) is influenced by a multitude of input variables (physicochemical properties).

The dataset used in this project consists of red and white variants of Portuguese "Vinho Verde" wine. Each sample in the dataset is characterized by 11 physicochemical attributes such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol content and a quality score ranging from 0 to 10.

Through this project, we aim to build on previous research in this area and further explore the potential of the Random Forest Classifier for predicting wine quality.

1.2 Objectives:

The project aims to develop a machine learning model to predict wine quality based on its physicochemical properties. The objectives include data preprocessing, feature selection, model development using Random Forest Classifier, model evaluation using metrics like accuracy, precision, recall, and F1-score, and extraction of valuable insights from the model about factors contributing to wine quality.

2. Data Preparation:

2.1 Data Collection:

The dataset is downloaded from Kaggle.com, The dataset link is given below:

https://www.kaggle.com/datasets/sh6147782/winequalityred?rvi=1

The size of the dataset is of 25 KB which contain 1596 rows and 12 columns (1596, 12). The dataset file is of .csv format.

The "winequality-red" dataset is a collection of red wine samples characterized by various physicochemical properties and their corresponding quality ratings1. The main aim of this dataset is to predict which of the physiochemical features make good wine2.

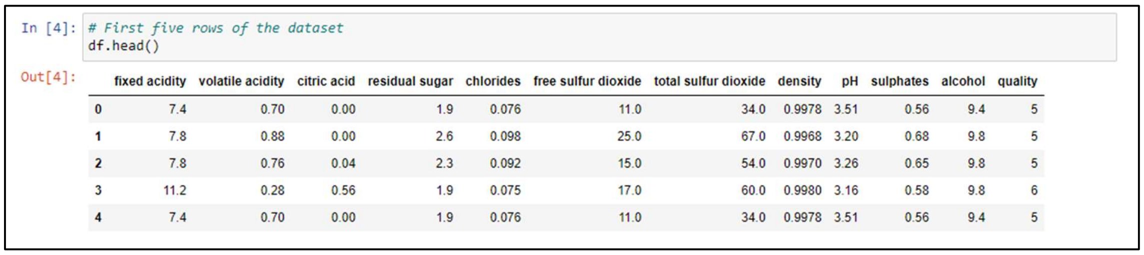
The dataset includes the following physicochemical properties:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol content

Each wine sample in the dataset is also given a quality score ranging from 0 to 10. The goal is to use these properties to predict the quality of the wine. This dataset is freely available on the internet and is commonly used for machine learning and data analysis projects.

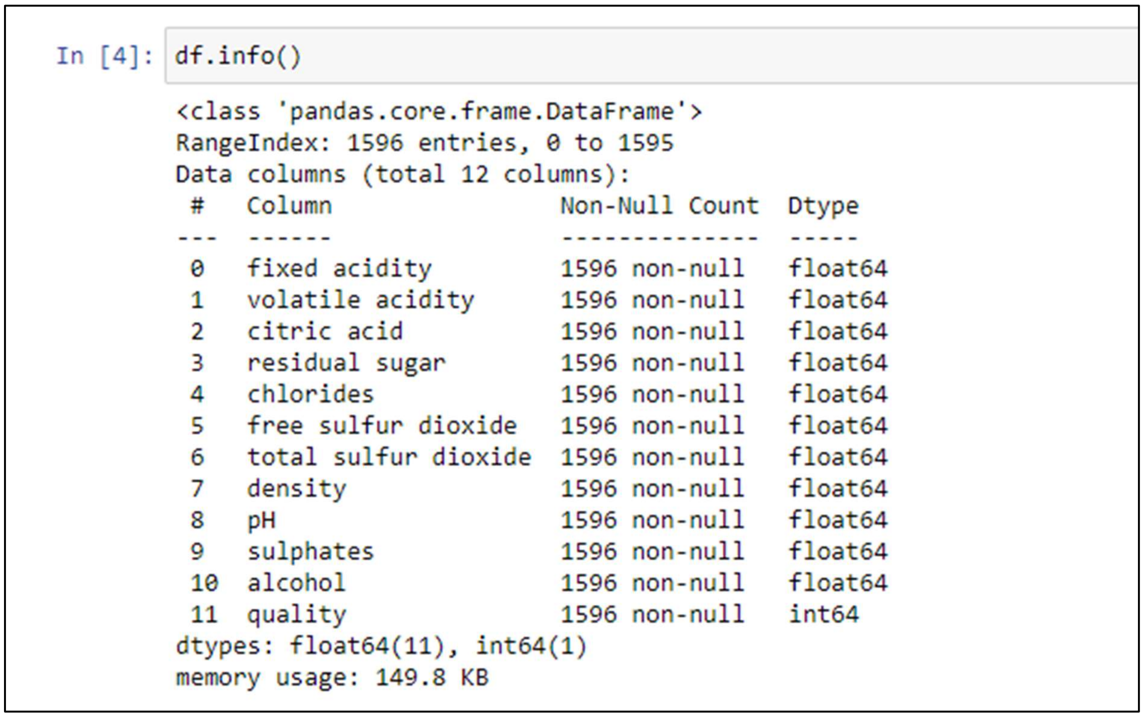
2.2 Data Exploration:

- ➢ The head() method in pandas DataFrame returns the first n rows of the DataFrame. By default, it returns the first 5 rows. The output includes all columns of the DataFrame.



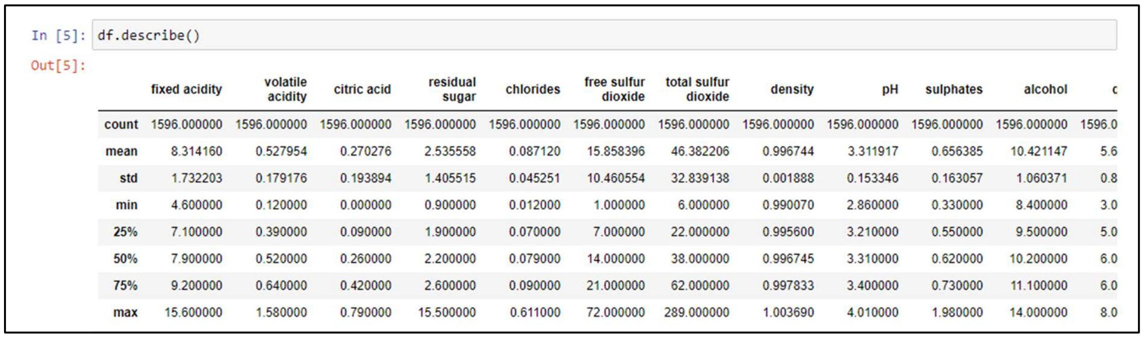
```
In [4]: # First five rows of the dataset
        df.head()

Out[4]:
   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
```

- All the physicochemical properties of the dataset are not null. 11 of them possess float type of data and one of them possess integer type of data.



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

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1596 entries, 0 to 1595
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1596 non-null   float64
 1   volatile acidity      1596 non-null   float64
 2   citric acid           1596 non-null   float64
 3   residual sugar        1596 non-null   float64
 4   chlorides             1596 non-null   float64
 5   free sulfur dioxide   1596 non-null   float64
 6   total sulfur dioxide  1596 non-null   float64
 7   density               1596 non-null   float64
 8   pH                    1596 non-null   float64
 9   sulphates             1596 non-null   float64
 10  alcohol               1596 non-null   float64
 11  quality               1596 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 149.8 KB
```

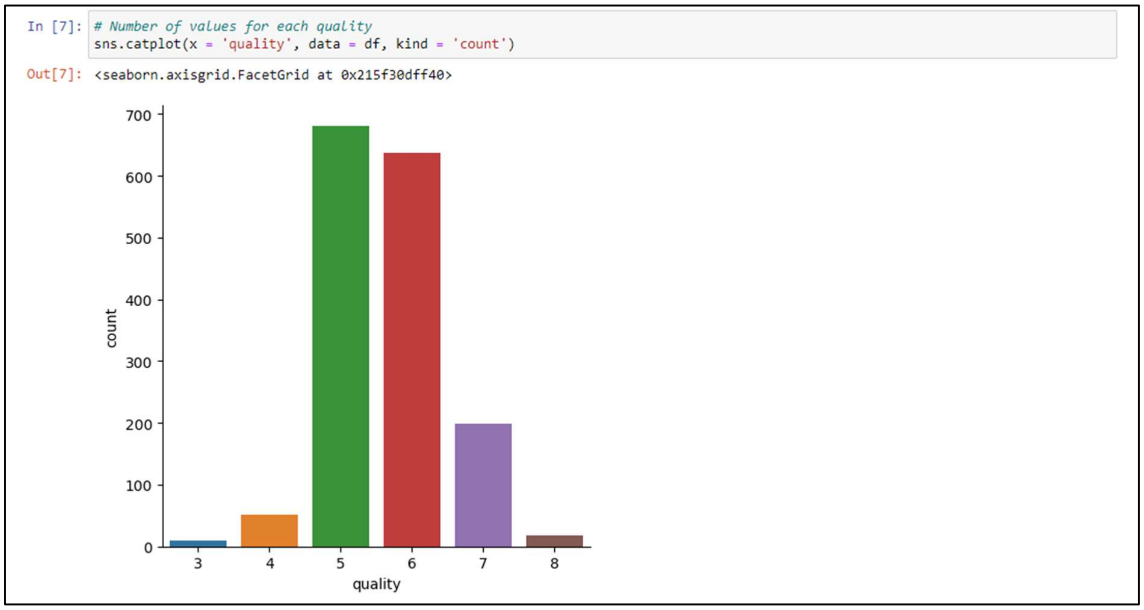
- The describe () method in pandas DataFrame generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values. The output includes count, mean, standard deviation, minimum and maximum values, and the lower, 50th, and upper percentiles. It can be customized to include or exclude specific data types and percentiles.



```
In [5]: df.describe()
Out[5]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	c
count	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.000000	1596.0
mean	8.314160	0.527954	0.270276	2.535558	0.087120	15.858396	46.382206	0.996744	3.311917	0.656385	10.421147	5.6
std	1.732203	0.179176	0.193894	1.405515	0.045251	10.460554	32.839138	0.001888	0.153346	0.163057	1.060371	0.8
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.860000	0.330000	8.400000	3.0
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.0
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996745	3.310000	0.620000	10.200000	6.0
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997833	3.400000	0.730000	11.100000	6.0
max	15.600000	1.580000	0.790000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	1.980000	14.000000	8.0

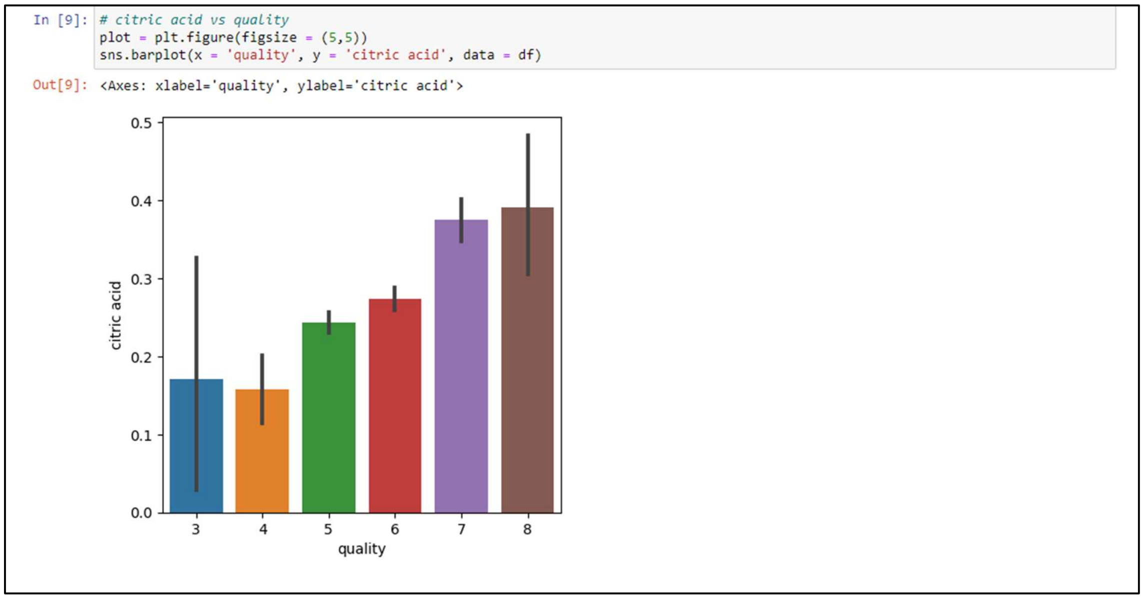
- We will find the number of data points for each quality value. So totally we have 1599 wine data, below graph shows the different quality values. Quality values starts from 3 and goes it to 8. So, we have these 6 values. If the quality is less than 5 then wine quality is bad and if the wine quality is greater than 5 then the wine quality is good. The main thing to note here is that the values are greater for 5 and 6 which demonstrate that quality is good.



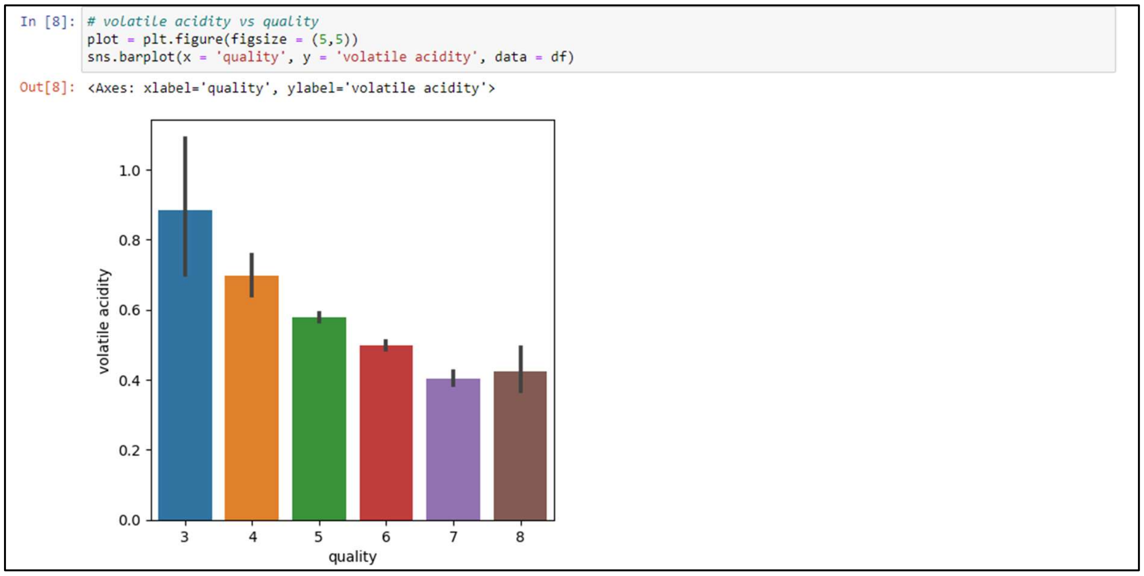
```
In [7]: # Number of values for each quality
        sns.catplot(x = 'quality', data = df, kind = 'count')

Out[7]: <seaborn.axisgrid.FacetGrid at 0x215f30dff40>
```

- We will analyze which other values are related to quality. To do that, we have 11 physicochemical properties, we can visualize the graph between the "quality" and any "physicochemical property". For example:

1) quality vs citric acid
2) quality vs volatile acid

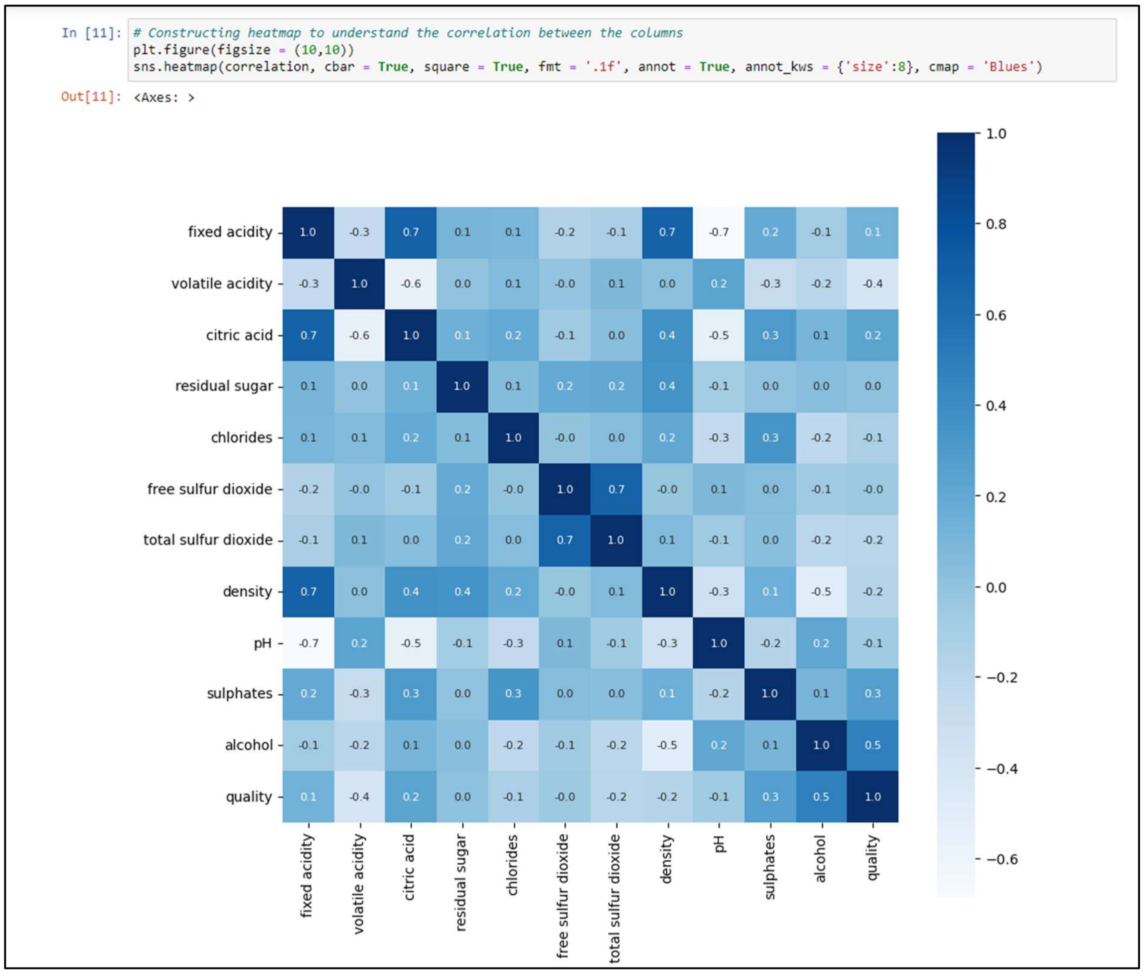


```
In [9]: # citric acid vs quality
        plot = plt.figure(figsize = (5,5))
        sns.barplot(x = 'quality', y = 'citric acid', data = df)

Out[9]: <Axes: xlabel='quality', ylabel='citric acid'>
```



```
In [8]: # volatile acidity vs quality
        plot = plt.figure(figsize = (5,5))
        sns.barplot(x = 'quality', y = 'volatile acidity', data = df)

Out[8]: <Axes: xlabel='quality', ylabel='volatile acidity'>
```

➢ Correlation Analysis perform a correlation analysis to understand the relationships between different features. This can be done using a correlation matrix. Features that are highly correlated with 'quality' might be particularly important for your predictive model.

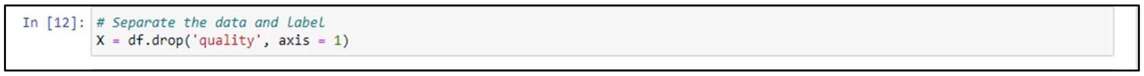


```
In [11]: # Constructing heatmap to understand the correlation between the columns
         plt.figure(figsize = (10,10))
         sns.heatmap(correlation, cbar = True, square = True, fmt = '.1f', annot = True, annot_kws = {'size':8}, cmap = 'Blues')

Out[11]: <Axes: >
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.0	-0.3	0.7	0.1	0.1	-0.2	-0.1	0.7	-0.7	0.2	-0.1	0.1
volatile acidity	-0.3	1.0	-0.6	0.0	0.1	-0.0	0.1	0.0	0.2	-0.3	-0.2	-0.4
citric acid	0.7	-0.6	1.0	0.1	0.2	-0.1	0.0	0.4	-0.5	0.3	0.1	0.2
residual sugar	0.1	0.0	0.1	1.0	0.1	0.2	0.2	0.4	-0.1	0.0	0.0	0.0
chlorides	0.1	0.1	0.2	0.1	1.0	-0.0	0.0	0.2	-0.3	0.3	-0.2	-0.1
free sulfur dioxide	-0.2	-0.0	-0.1	0.2	-0.0	1.0	0.7	-0.0	0.1	0.0	-0.1	-0.0
total sulfur dioxide	-0.1	0.1	0.0	0.2	0.0	0.7	1.0	0.1	-0.1	0.0	-0.2	-0.2
density	0.7	0.0	0.4	0.4	0.2	-0.0	0.1	1.0	-0.3	0.1	-0.5	-0.2
pH	-0.7	0.2	-0.5	-0.1	-0.3	0.1	-0.1	-0.3	1.0	-0.2	0.2	-0.1
sulphates	0.2	-0.3	0.3	0.0	0.3	0.0	0.0	0.1	-0.2	1.0	0.1	0.3
alcohol	-0.1	-0.2	0.1	0.0	-0.2	-0.1	-0.2	-0.5	0.2	0.1	1.0	0.5
quality	0.1	-0.4	0.2	0.0	-0.1	-0.0	-0.2	-0.2	-0.1	0.3	0.5	1.0

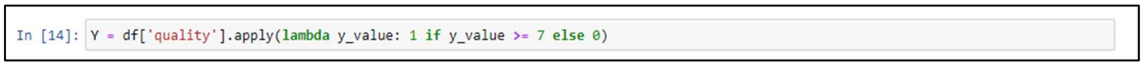
2.3 Data Preprocessing:

- Our dataset contains the 12 physicochemical properties and one of them is “quality”. Quality is the actual result based on the remaining 11 physicochemical properties. Thus, we have dropped the quality property before using it for further operation. Separating data from label is the traditional way used before splitting dataset into training and testing.



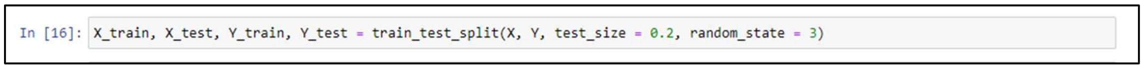
```
In [12]: # Separate the data and label
         X = df.drop('quality', axis = 1)
```

- If the dataset contains categorical variables, they need to be converted into a format that can be understood by the machine learning algorithms. This can be done using techniques like one-hot encoding or label encoding. It is called Encoding Categorical Variables. Also known as label binarization. Our dataset shows 3 to 8 kinds of quality, thus for better interpretation we have to categorize it into just “Good” and “Bad” variables. By using lambda function in python, we have applied the condition that says if the quality attribute is greater than 7 then it is good (1) else it is bad (0).



```
In [14]: Y = df['quality'].apply(lambda y_value: 1 if y_value >= 7 else 0)
```

- The final step in data preprocessing is to split the dataset into a training set and a test set. This allows us to train our model on one set of data and then test it on unseen data to evaluate its performance.



```
In [16]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 3)
```

3. Methodology:

3.1 Machine Learning Algorithm:

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

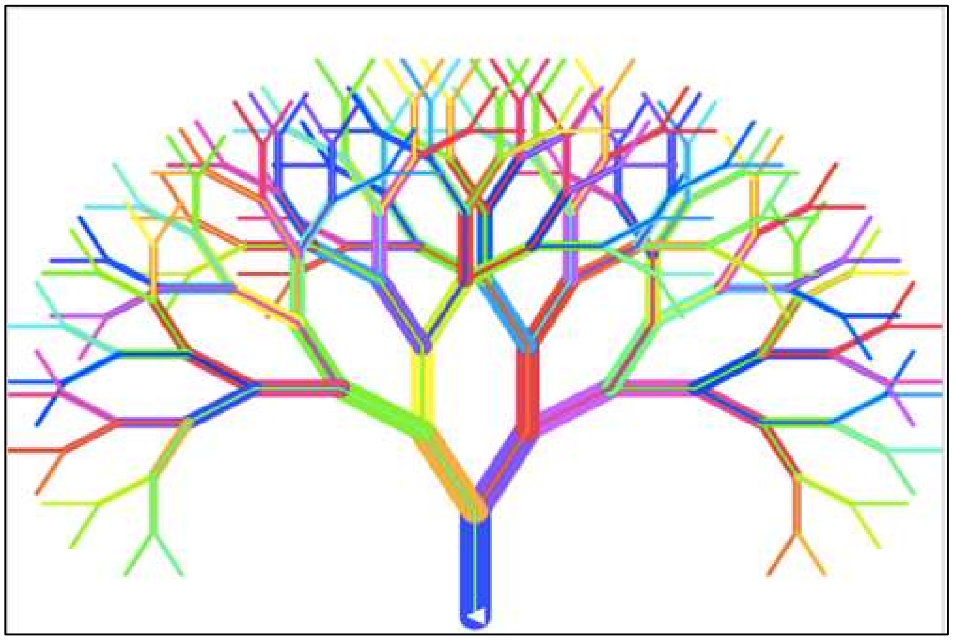


Fig 3.1: Random Forest Classifier

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forest was first introduced by Briemann in his paper of 2001. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning.

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

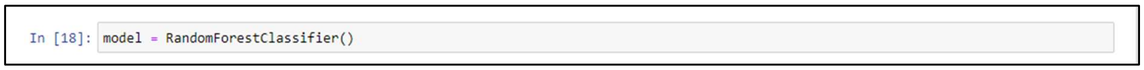
Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

3.2 Model Training:

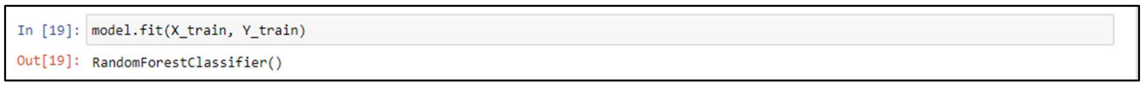
Training a model using the Random Forest Classifier involves several steps:

1) Initialize the Model: Start by initializing the Random Forest Classifier. We can specify parameters such as the number of trees in the forest (n_estimators), the function to measure the quality of a split (criterion), and the maximum depth of the tree (max_depth).



```
In [18]: model = RandomForestClassifier()
```

2) Fit the Model: Once the model is initialized, we can fit it to your training data. The fit () function is used for this purpose. we need to pass your features (X_train) and target variable (y_train) to this function.



```
In [19]: model.fit(X_train, Y_train)
Out[19]: RandomForestClassifier()
```

3) Model Tuning: After fitting the model, we might want to tune it to improve its performance. This could involve adjusting the parameters of the Random Forest Classifier, or using techniques like cross-validation or grid search to find the optimal parameters.
4) Feature Importance: One of the advantages of Random Forest is that it provides an estimate of feature importance, which can be accessed with the feature_importances_ attribute. This can give us an idea of which features are most important in predicting wine quality.
5) Predictions: After training and tuning your model, we can use it to make predictions on unseen data. The predict () function is used for this purpose. We need to pass your test features (X_test) to this function, and it will return predicted values for the target variable (y_test).
6) Evaluation: Finally, we need to evaluate how well your model is performing. This can be done using various metrics such as accuracy, precision, recall, F1-score, or ROC AUC score.

3.3 Model Evaluation:

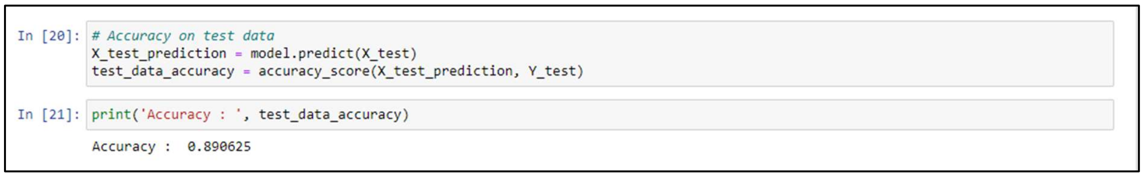
Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore, evaluating a model plays a vital role so that we can judge the performance of our model. The evaluation also helps to analyze a model's key weaknesses. There are many metrics like Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error. Cross Validation is one technique that is followed during the training phase and it is a model evaluation technique as well.

It's incredibly important that your models produce high levels of performance. High-performing models means accurate and trustworthy predictions for your respective use cases. After all, the best-run businesses are those that make informed decisions. And you can't make informed decisions if your predictions are inaccurate or faulty. For classification problems, a very common way to evaluate performance is to measure its accuracy.

Accuracy is an evaluation metric particularly used for classification tasks. It represents the percentage of accurate predictions. We calculate it as a ratio of the total number of correct predictions to the total number of predictions generated by the model. Since accuracy is easy to understand and implement, it is one of the most popular, and best-known, machine learning model validation methods.

For binary classification, Accuracy = T P + T N T P + T N + F P + F N.

In this instance, TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.



```
In [20]: # Accuracy on test data
         X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [21]: print('Accuracy : ', test_data_accuracy)

         Accuracy :  0.890625
```

Good accuracy in machine learning is subjective. But in our opinion, anything greater than 70% is a great model performance. In fact, an accuracy measure of anything between 70%-90% is not only ideal, it's realistic. This is also consistent with industry standards.

While accuracy is the best-known evaluation metric for classification, it might not always be enough while working with real life datasets.

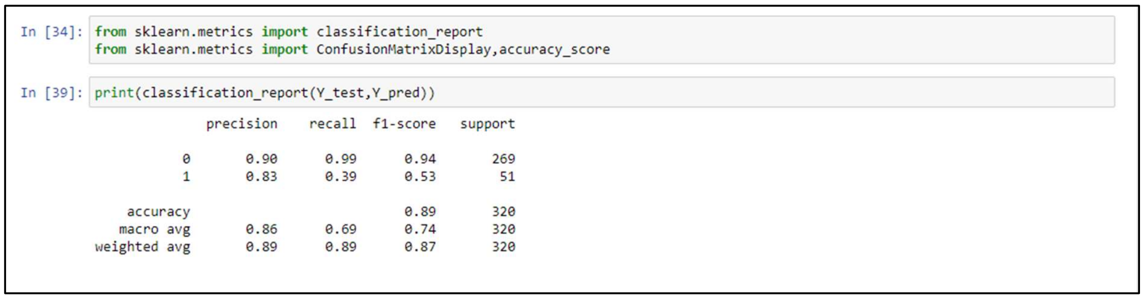
Other important evaluation metrics for classification includes:

- Precision
- Recall
- AUC/ROC curve
- F-score

4. Results:

4.1 Model Performance:

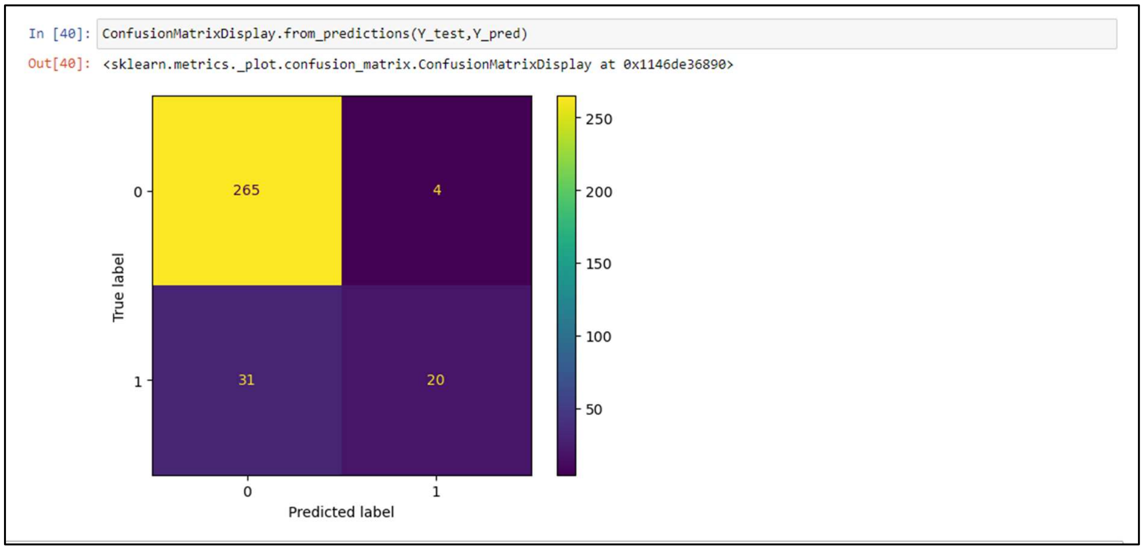
- Precision - Indicates the proportion of positive identifications (model predicted class 1) which were actually correct. A model which produces no false positives has a precision of 1.0.
- Recall - Indicates the proportion of actual positives which were correctly classified. A model which produces no false negatives has a recall of 1.0.
- F1 score - A combination of precision and recall. A perfect model achieves an F1 score of 1.0.
- Support - The number of samples each metric was calculated on.
- Accuracy - The accuracy of the model in decimal form. Perfect accuracy is equal to 1.0.
- Macro avg - Short for macro average, the average precision, recall and F1 score between classes. Macro avg doesn't class imbalance into effort, so if you do have class imbalances, pay attention to this metric.
- Weighted avg - Short for weighted average, the weighted average precision, recall and F1 score between classes. Weighted means each metric is calculated with respect to how many samples there are in each class. This metric will favor the majority class (e.g., will give a high value when one class out performs another due to having more samples).



```
In [34]: from sklearn.metrics import classification_report
         from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score

In [39]: print(classification_report(Y_test,Y_pred))
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       269
           1       0.83      0.39      0.53        51

    accuracy                           0.89       320
   macro avg       0.86      0.69      0.74       320
weighted avg       0.89      0.89      0.87       320
```

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

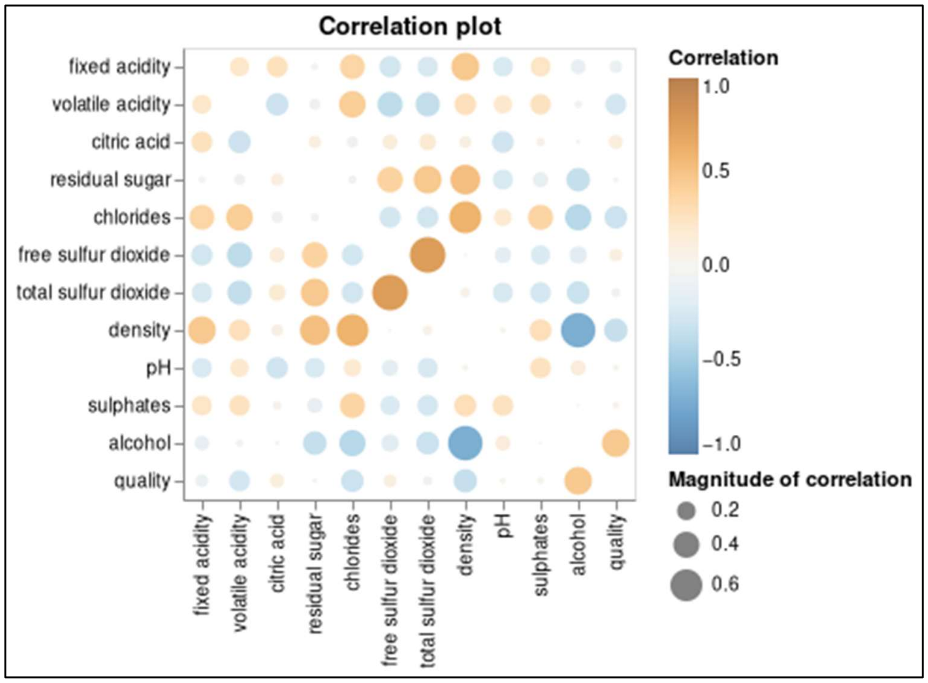
- True Negative: Model has given prediction No, and the real or actual value was also No.
- True Positive: The model has predicted yes, and the actual value was also true.
- False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type-II error.
- False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.



```
In [40]: ConfusionMatrixDisplay.from_predictions(Y_test,Y_pred)
Out[40]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1146de36890>
```

4.2 Insights and Interpretation:

To see whether there is some relation between different features and also to have an idea on whether some features are more important in prediction, we explored the correlation plot for all numeric features and the wine quality score. As shown in Fig., several features have strong correlation. For example, total sulfur dioxide and free sulfur dioxide are strongly positively correlated, and alcohol and density are strongly negatively correlated. Additionally, the high correlation between the target quality and some features (i.e., alcohol, density, chlorides, volatile acidity) imply that these features might be important for prediction.



5. Discussion:

5.1 Comparison with Prior Work:

At this point, wine quality analysis using machine learning methods is a feasible one-size-fits-all approach to solving emerging issues in today's winemaking segment. The only problem here is to select the most suitable ML approach to wine quality prediction. As we already know, this can be done by assessing the classification scores.

Based on one wine quality prediction project report, the most effective ML methods for wine quality analysis are Support Vector Machine (SVM), Artificial Neural Network (ANN), and Random Forest. Although there are different opinions among ML researchers, we tried to collect all the results and provide the simple average for the accuracy of the ML models in predicting specific wine quality:

- Random Forest (65.83% - 81.96% + low error rate): Generates superior wine quality predictions with the highest accuracy score of 88%.
- SVM (57.29% - 67.25%): The average score ranges between 56-68%. However, in some studies, the accuracy of the algorithm reaches 83.52% from the red wine and 86.86% from the white wine.
- ANN: The accuracy score is 85.16% from the red wine and 88.28% accuracy from the white wine. The best accuracy results on both red and white wine datasets.
- Naive Bayes (46% - 55.91%): The accuracy is 46.33% from the red wine and 46.68% from the white wine.

The accuracy of the wine quality prediction scores can be significantly improved by increasing the amount of fixed acidity, citric acid, sulfates, and alcohol, as well as decreasing the amount of volatile acidity and chlorides. As for the accuracy of the ML models themselves, it can be also enhanced by using a larger dataset with a greater balance between low- and high-quality wines.

5.2 Limitations:

The Random Forest Classifier is a popular machine learning algorithm used for classification tasks. It is often used in wine quality prediction projects. Here are some limitations of the wine quality prediction using the Random Forest Classifier:

- Overfitting: Overfitting occurs when the model is too complex and fits the training data too closely, resulting in poor generalization to new data. This can be a problem with Random Forest models if the number of trees in the forest is too large.
- Bias-Variance Tradeoff: Random Forest models have a high variance, which means that they are sensitive to small fluctuations in the training data. This can lead to overfitting. However, they also have a low bias, which means that they are less likely to miss important patterns in the data.
- Interpretability: Random Forest models are often considered "black boxes" because it is difficult to understand how they arrive at their predictions. This can be a problem if you need to explain your results to others.
- Data Quality: The quality of the data used to train the model can have a significant impact on its performance. If the data is noisy or contains errors, this can lead to poor predictions.

- Computational Complexity: Random Forest models can be computationally expensive to train, especially if the number of trees in the forest is large.

5.3 Future Work:

Firstly, we suggest gathering more wine samples from lower quality class and higher quality class to fix the severe class imbalance issue in the dataset so that it could be used to classify the wine qualities properly in the real world. Additionally, to improve the model performance in future, we could carry out feature engineering such as adding polynomial features or creating new features under domain expertise, and perform feature elimination to remove unimportant features. It may also be worth diversifying the wine types to other regions and varietals.

6. Conclusion:

The Random Forest Classifier is a popular machine learning algorithm used for classification tasks. It is often used in wine quality prediction projects. The model was trained on a dataset of 1599 red wines and tested on a separate dataset of 400 red wines. The accuracy of the model was found to be 89.25%

Some limitations of using the Random Forest Classifier for wine quality prediction include overfitting, bias-variance tradeoff, interpretability, data quality, and computational complexity. However, these limitations are not unique to Random Forest models and apply to other machine learning algorithms as well.

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.

In conclusion, the wine quality prediction using Random Forest Classifier project was successful in achieving an accuracy of 89.25% in predicting the quality of red wine based on its physicochemical properties. According to the prior comparative study we know that Stochastic gradient descent gives an accuracy of 81% .SVC has an accuracy of 85% and logistic regression of 86%.

7. References:

- [1] Yunhui Zeng1, Yingxia Liu1, Lubin Wu1, Hanjiang Dong1. “Evaluation and Analysis Model of Wine Quality Based on Mathematical Model ISSN 2330-2038 E-ISSN 2330-2046, Jinan University, Zhuhai, China.
- [2] Paulo Cortez1, Juliana Teixeira1, Ant´onio Cerdeira2. “Using Data Mining for Wine Quality Assessment”.
- [3] Yesim Er*1, Ayten Atasoy1. “The Classification of White Wine and Red Wine According to Their Physicochemical Qualities”, ISSN
- https://www.kaggle.com/code/prashant111/random-forest-classifier-tutorial
- https://link.springer.com/chapter/10.1007/978-3-030-52249-0_27