Report On Wine Quality Prediction Using Machine Learning

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELORS OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

MAMIDIPAKA VENKATA SAI NIKHIL

12013998

Supervisor

Aishwarya Shukla

School of Computer Science and Engineering

Lovely Professional University

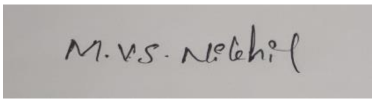
Phagwara, Punjab (India)

Month-May Year-2023

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled “WINE QUALITY PREDICTION” in partial fulfilment of the requirement for the award of Degree for Bachelors of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Aishwarya Shukla. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.



(Signature of Candidate)

Name: Mamidipaka Venkata Sai Nikhil
Registration No.: 12013998

Date: 27th April, 2023

ACKNOWLEDGEMENT

The success and final outcome of learning Machine Learning required a lot of guidance and assistance from many people. I am extremely privileged to have got this all along the completion of my course and few of the projects. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I would like to express my sincere gratitude to Upgrad, for providing me an opportunity to do the course and project work and giving me all support and guidance, which made me complete the course duty. I would especially thankful to the course advisor Mr. Aishwarya Shukla sir and Mr. Ved Prakash Chaubey sir.

I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance for choosing this course.

M.V.S. Nikhil

(Signature of Candidate)

Name: Mamidipaka Venkata Sai Nikhil

Registration no.: 12013998

Date: 27th April, 2023

TABLE OF CONTENTS

CONTENTS	PAGE NO.
Title Page	1
Declaration by candidates	2
Acknowledgement	3
Index No	4
Objective and scope of the Project	5
Introduction	6
Hardware & Software used	7
Working Of Code	8 - 19
Results and Discussion	20
Summary	21
Bibliography	22
Annexure	23-24

OBJECTIVE

The objective of a wine quality prediction dataset in machine learning is to provide a set of labelled data that can be used to train and evaluate machine learning models for predicting wine quality. The quality of wine can be rated on a scale of 1-10 or bad-good. The dataset typically includes a range of input features, such as volatile acidity, pH levels, alcohol content, and other chemical characteristics of the wine, as well as a corresponding quality rating. The goal of using this dataset is to develop a model that can accurately predict wine quality based on these input features.

- To experiment with different classification methods to see which yields the highest accuracy.

- To determine which features are the most indicative of a good quality wine.

Scope of the project

The scope of a wine quality prediction project can vary depending on the goals of the project and the available resources. However, some common aspects of such a project may include:

1. **Data Collection:** Collecting relevant data on different types of wine, such as their characteristics (e.g., acidity, sweetness, tannins, etc.), the region where they are produced, the grape varieties used, and the ratings given by wine experts or consumers.

2. **Data Preprocessing:** Cleaning and processing the collected data, such as handling missing values, outliers, and scaling the features.

3. **Feature Selection:** Selecting the most relevant features that contribute to the wine quality prediction.

4. **Model Selection:** Choosing an appropriate machine learning algorithm to train the model that can accurately predict the wine quality.

5. **Model Training:** Training the selected model on the preprocessed data.

6. **Model Evaluation:** Evaluating the performance of the trained model using various performance metrics such as accuracy, precision, recall, F1-score, etc.

7. Model Deployment: Deploying the model for real-time predictions, such as building a web application or a mobile app that can predict wine quality based on user input.

Overall, the scope of a wine quality prediction project can be challenging but rewarding, as it can help wine enthusiasts, producers, and consumers to make informed decisions about wine selection and production.

Introduction

The wine industry is highly competitive, with many different types of wines being produced worldwide. Wine quality prediction is a machine learning project that aims to predict the quality of wine based on its chemical and physical characteristics.

In recent years, machine learning techniques have become increasingly popular in the wine industry as they can help winemakers and wine enthusiasts to make better decisions about wine production and selection.

The wine quality prediction project involves the collection of relevant data on different types of wine, such as their chemical and physical characteristics, grape varieties, and ratings given by wine experts or consumers. This data is then preprocessed and used to train a machine learning model that can predict the quality of wine based on its features.

The trained model can be used to predict the quality of new wines or evaluate the quality of existing wines. This can help winemakers to identify the key factors that contribute to the quality of their wines and make necessary adjustments to improve the quality of their products. It can also help wine enthusiasts and consumers to make informed decisions about wine selection based on their preferences.

Overall, the wine quality prediction project in machine learning has the potential to revolutionize the wine industry by providing valuable insights into wine production and selection.

Software Used

We have used two different ML algorithms to predict the quality of wine: Logistic Regression and Random Forest Classifier. Both algorithms are widely used in supervised learning tasks and are well-suited for binary classification problems.

Logistic Regression:

Logistic regression is a popular machine learning algorithm that can be used for binary classification tasks, such as predicting whether a wine is of high or low quality based on its chemical properties. In wine quality prediction, logistic regression can be used to build a model that takes in inputs of chemical properties of wine, such as alcohol content, pH, volatile acidity, and residual sugar, and predicts whether the wine is of high quality or low quality.

Random Forest Classifier:

The Random Forest Classifier is an ensemble algorithm that builds multiple decision trees and combines their results to make a final prediction. This technique of combining multiple decision trees is called "bagging." One of the significant advantages of the Random Forest Classifier is that it can handle both categorical and continuous data. It also reduces overfitting by building multiple decision trees and combining their results.

Support Vector Classifier:

Support Vector Classifier (SVC) is one of the classifiers used for wine quality prediction in machine learning1. SVC is a supervised learning algorithm that can be used for classification or regression tasks. It works by finding the hyperplane that best separates the data into different classes. SVC has been used for wine quality prediction along with other machine learning algorithms such as Logistic Regression, Decision Tree, and Random Forest.

Working Of Code

Importing Libraries

In [26]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
from warnings import filterwarnings
filterwarnings(action='ignore')
```

- Converting into data frame by reading the wine quality csv file using pandas

In [29]:
```
# Importing the dataset
wine = pd.read_csv('winequality.csv')
wine
```

Out[29]:

	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

- Displaying the first five columns of data frame

In [30]:
```
wine.head()
```

Out[30]:

	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

- Displaying the last five columns of data frame

In [31]:
```
wine.tail()
```

Out[31]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
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

- Number of columns and rows in data frame

```
In [3]: wine.shape
Out[3]: (1599, 12)
```

The above winequality dataset contains 1599 rows and 12 columns.

- Datatype of each column in data frame

```
In [4]: wine.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
```

- Statistical summary of the wine DataFrame

```
In [5]: wine.describe(include='all')
Out[5]:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.6
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.8
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	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	0.996750	3.310000	0.620000	10.200000	6.0
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.0
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.0

It provides a statistical summary of the wine DataFrame, including information about its numerical variables.

The resulting summary will contain the following information:

count: the number of non-null values for each column
mean: the mean value for numeric variables
std: the standard deviation for numeric variables

min: the minimum value for numeric variables
25%: the 25th percentile for numeric variables
50%: the median (50th percentile) for numeric variables
75%: the 75th percentile for numeric variables
max: the maximum value for numeric variables

- Checking for the null values

Finding Null Values

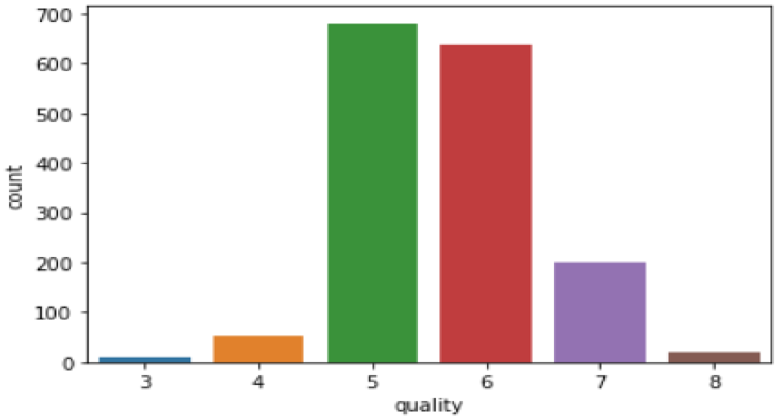
```
In [6]: print(wine.isna().sum())

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
```

The dataset doesnot contain any null values

Data Visualization

- Number of values for each quality

```
In [9]: sns.countplot(wine['quality'])
        plt.show()
```

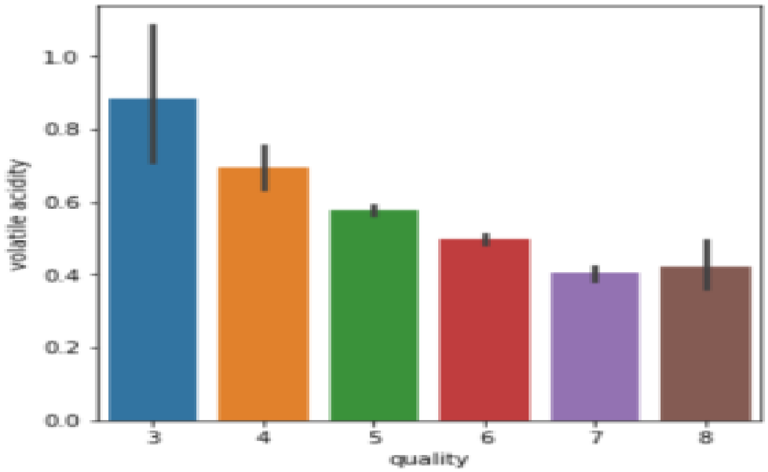


The above count plot describes that the quality feature values are heavily unbalanced. The dominant values are 5 and 6, and the values 3, 4 and 8 are almost absent from the set.

- Volatile acidity vs Quality

```
# volatile acidity vs Quality

plot = plt.figure(figsize=(5,5))
sns.barplot(x='quality', y = 'volatile acidity', data = df)
```

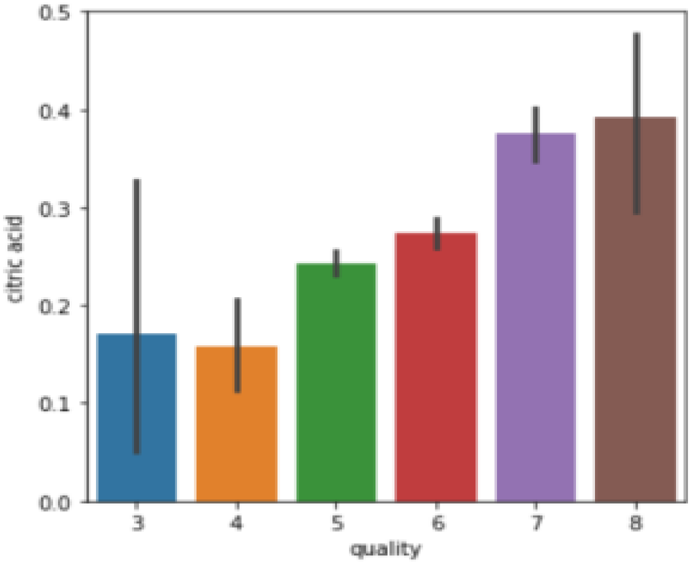
<AxesSubplot:xlabel='quality', ylabel='volatile acidity'>



- Citric acid vs Quality

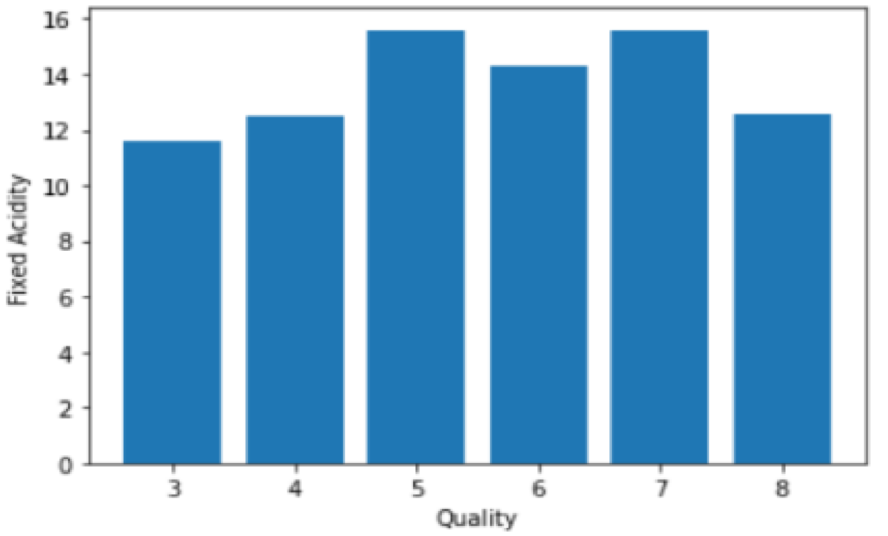
```
# citric acid vs Quality

plot = plt.figure(figsize=(5,5))
sns.barplot(x='quality', y = 'citric acid', data = df)
```

<AxesSubplot:xlabel='quality', ylabel='citric acid'>



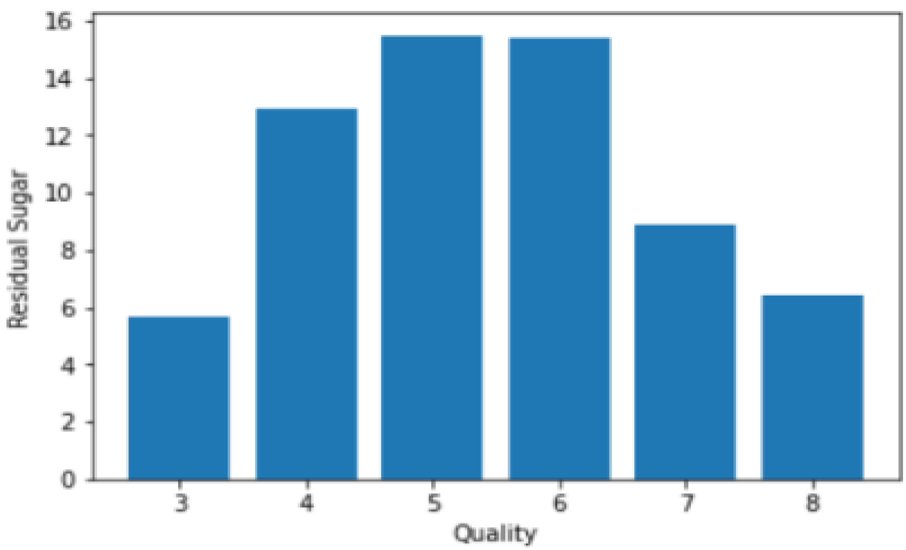
- Fixed acid vs Quality

```
# fixed acid vs quality

plt.bar(df['quality'],df['fixed acidity'])
plt.xlabel('Quality')
plt.ylabel('Fixed Acidity')
plt.show()
```



- Residual sugar vs Quality

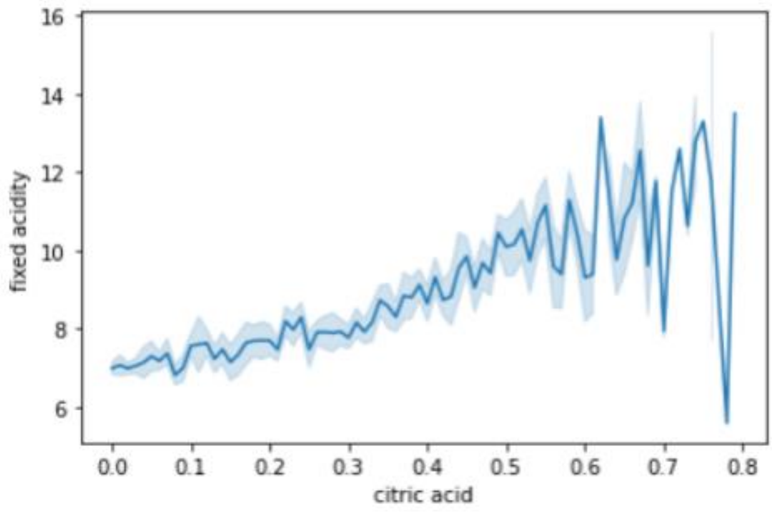
```
# residual sugar vs quality

plt.bar(df['quality'],df['residual sugar'])
plt.xlabel('Quality')
plt.ylabel('Residual Sugar')
plt.show()
```



- Citric acid vs fixed acid

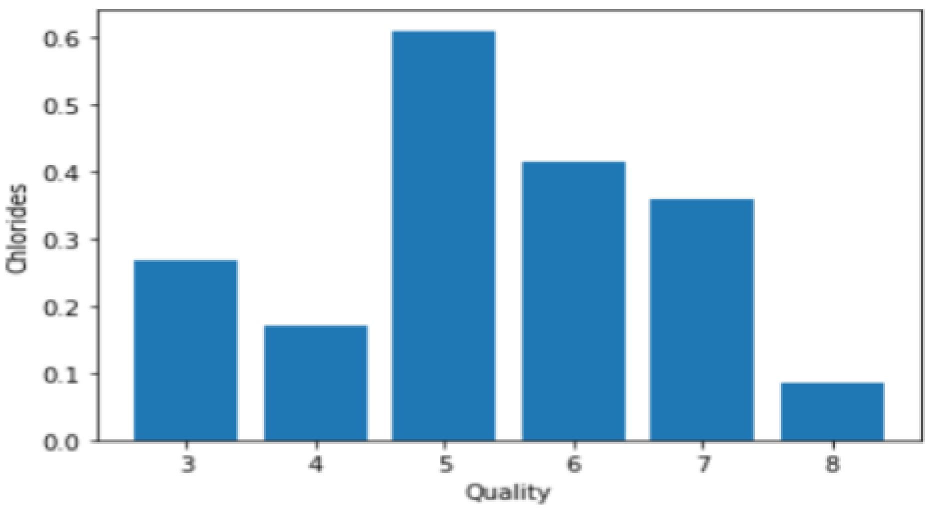
```
# line plot between citric acid vs fixed acid

sns.lineplot(x='citric acid',y='fixed acidity',data = df)
```

```
<AxesSubplot:xlabel='citric acid', ylabel='fixed acidity'>
```



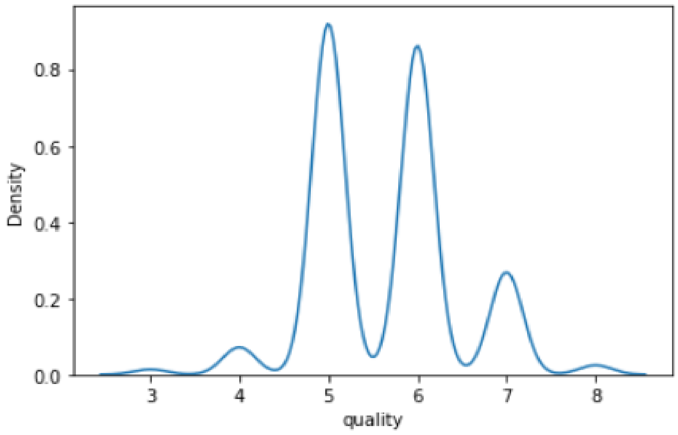
- Chlorides vs Quality

```
# Chlorides vs quality

plt.bar(df['quality'],df['chlorides'])
plt.xlabel('Quality')
plt.ylabel('Chlorides')
plt.show()
```



- **KDE plot**

```
In [10]: sns.kdeplot(wine.query('quality > 2').quality)
Out[10]: <AxesSubplot:xlabel='quality', ylabel='Density'>
```

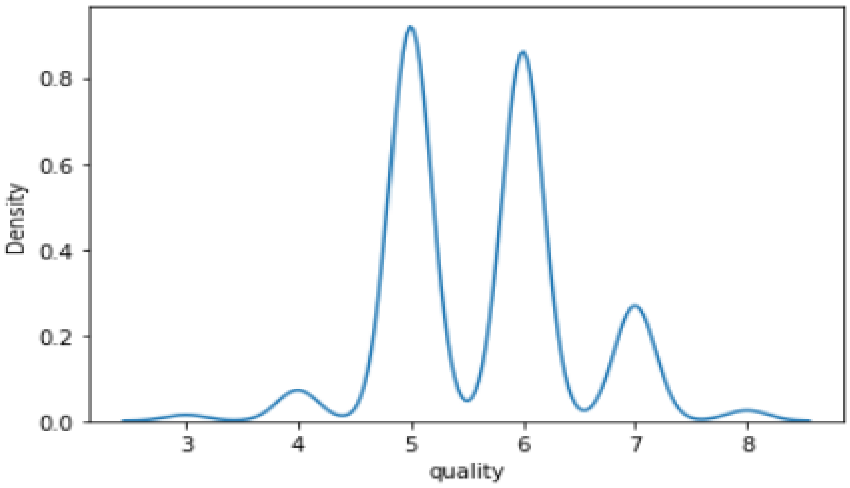


In this plot, the x-axis represents the wine quality scores, while the y-axis represents the estimated probability density of those scores.

The plot shows a smooth curve that represents the probability density function of the quality scores, with higher peaks indicating where the scores are more concentrated. By using the query method with the condition quality > 2, we filter out wines with a quality score less than or equal to 2, and only consider wines with higher quality scores.

- **Distplot**

```
In [10]: sns.kdeplot(wine.query('quality > 2').quality)
Out[10]: <AxesSubplot:xlabel='quality', ylabel='Density'>
```

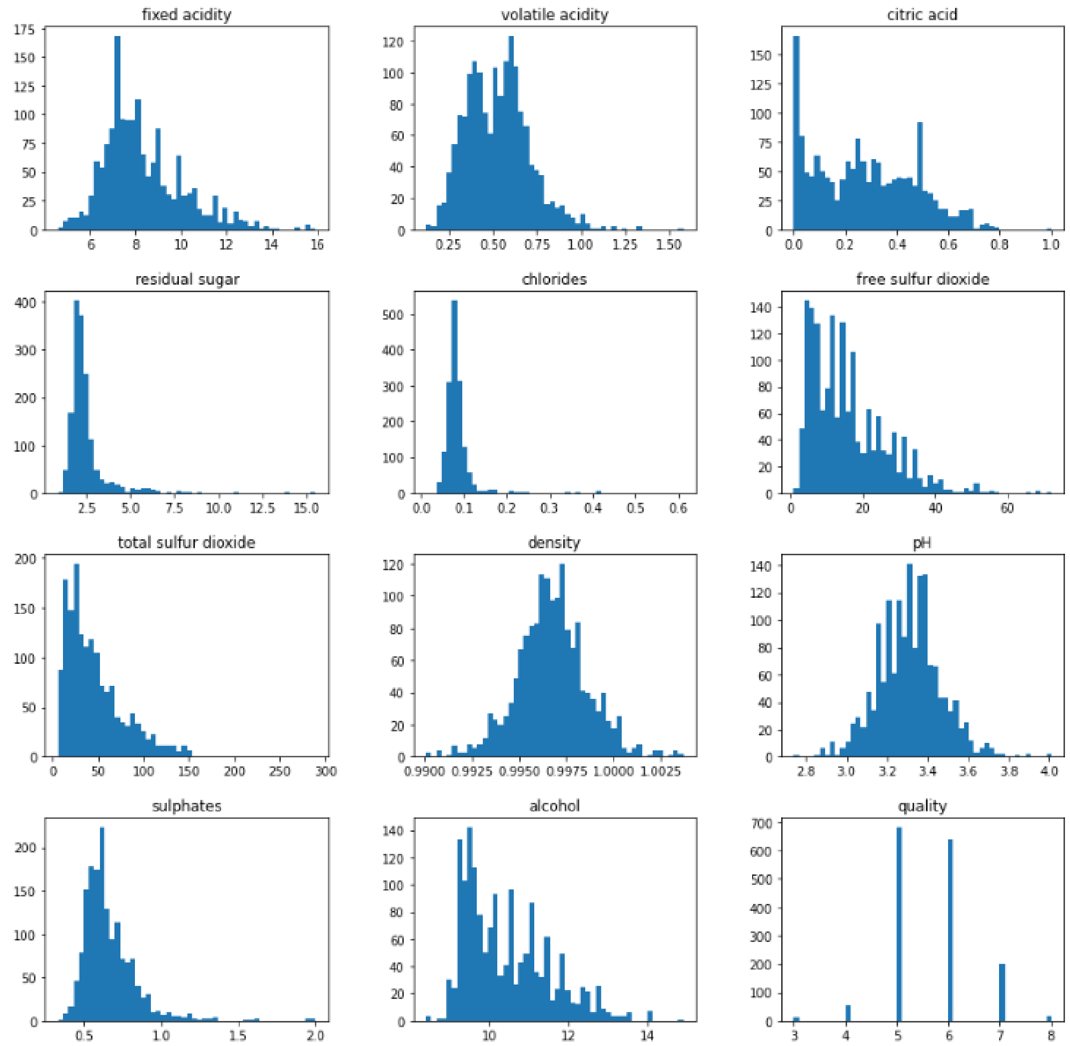


The resulting plot will show the frequency distribution of the "alcohol" values in the dataset, with the x-axis representing the alcohol values and the y-axis representing the density or frequency of occurrence of those values.

The histogram will display the counts of the different alcohol values, while the kernel density plot will display a smoothed estimate of the probability density function of the data.

- **Histogram**

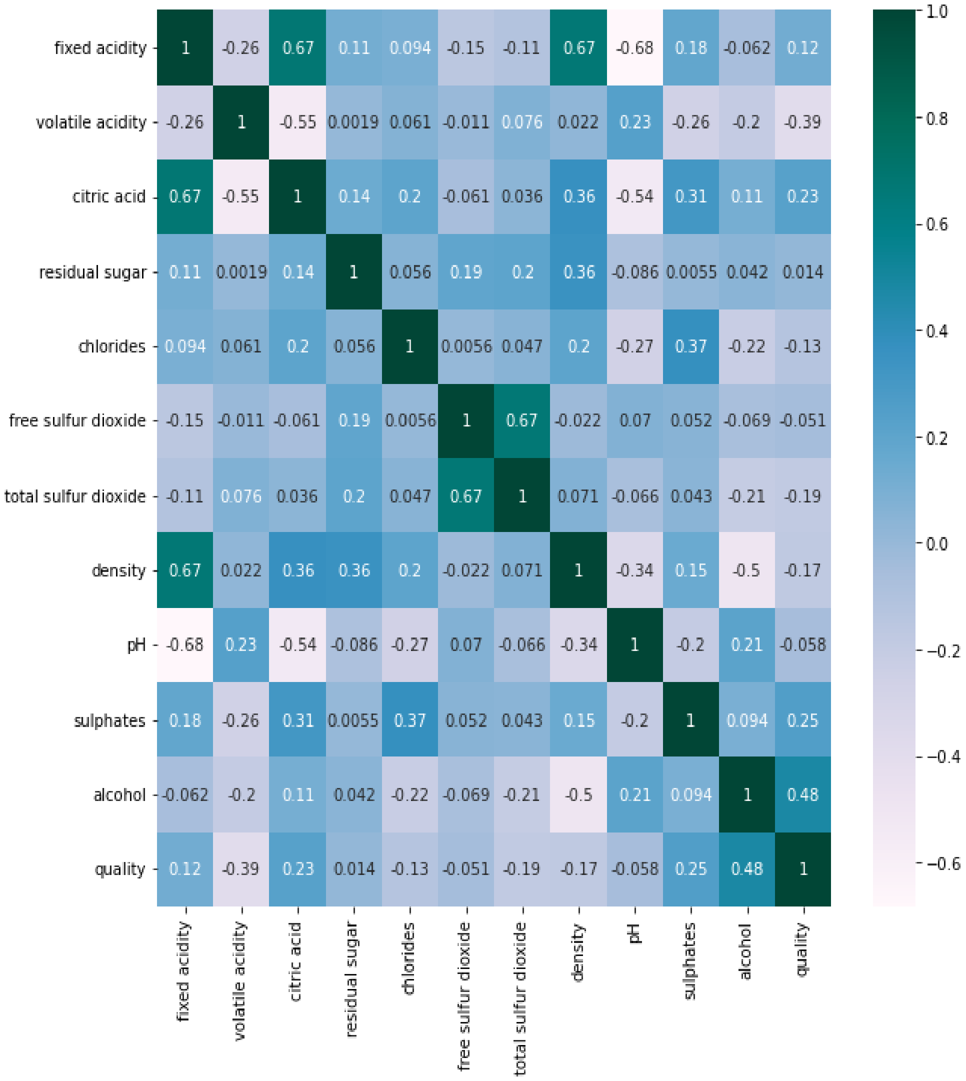
In [12]:
```
wine.hist(figsize=(15,15),bins=50,grid=False)
plt.show()
```



For a wine dataset, a histogram could show the distribution of a particular attribute or variable such as alcohol content, pH level, or quality rating. The x-axis of the histogram would represent the range of values for that attribute or variable, while the y-axis would represent the frequency of occurrence of values within each bin.

- Constructing a **heatmap** to understand the relationship between the columns

```
In [13]: plt.figure(figsize=(10,10))
         sns.heatmap(wine.corr(),annot=True,cmap='PuBuGn')
         plt.show()
```



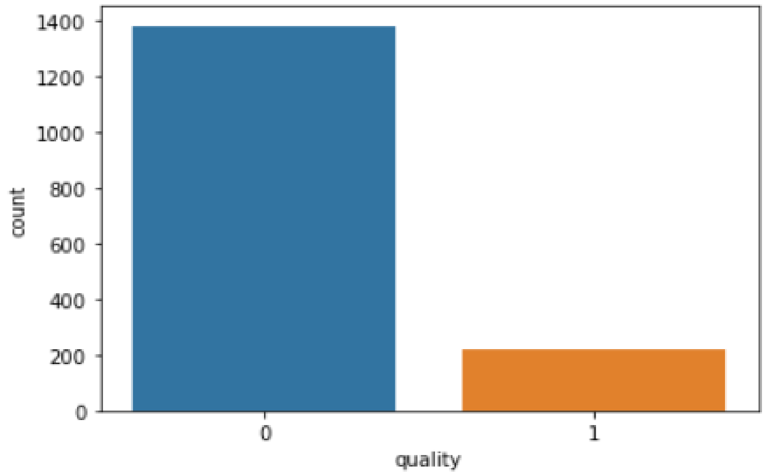
- Binarization of target variable

```
In [25]: wine['quality'].unique()
Out[25]: array([5, 6, 7, 4, 8, 3], dtype=int64)

In [26]: wine['quality'] = [1 if x >= 7 else 0 for x in wine['quality']]

In [27]: wine['quality'].unique()
Out[27]: array([0, 1], dtype=int64)
```

```
In [28]: wine['quality'].value_counts()
```

```
Out[28]: 0    1382
         1     217
         Name: quality, dtype: int64
```

```
In [30]: sns.countplot(x='quality',data=wine)
         plt.show()
```



- **Feature Engineering**

```
In [31]: #Assigning and dividing the dataset
         X = wine.drop('quality',axis=1)
         Y=wine['quality']
```

```
In [32]: X
```

Out[32]:

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

1599 rows × 11 columns

```
In [33]: Y
```

```
Out[33]: 0       0
         1       0
         2       0
         3       0
         4       0
                ..
         1594    0
         1595    0
         1596    0
         1597    0
         1598    0
         Name: quality, Length: 1599, dtype: int64
```

```
In [34]: wine.columns[:11]
```

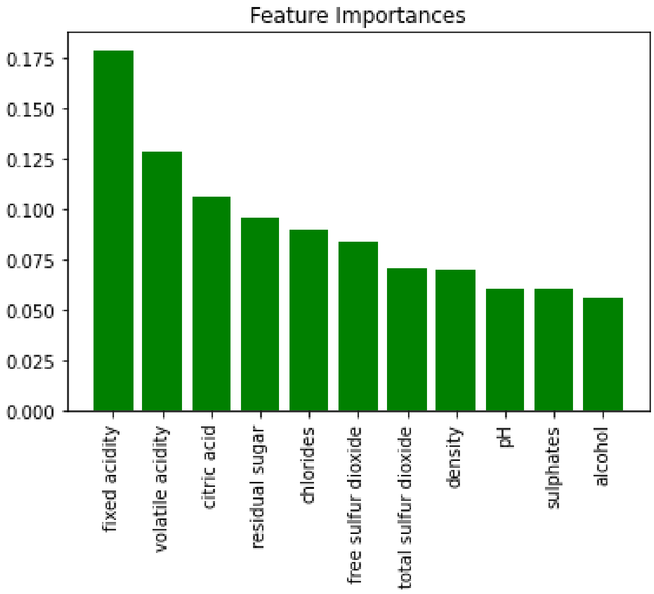
```
Out[34]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
               'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
               'pH', 'sulphates', 'alcohol'],
              dtype='object')
```

```
In [19]: features_label = wine.columns[:11]
```

```
In [20]: #Fitting Random Forest Classification to the Training set
         from sklearn.ensemble import RandomForestClassifier
         classifier = RandomForestClassifier(n_estimators = 200, criterion = 'entropy', random_state = 0)
         classifier.fit(X, Y)
         importances = classifier.feature_importances_
         indices = np. argsort(importances)[::-1]
         for i in range(X.shape[1]):
             print ("%2d) %-*s %f" % (i + 1, 30, features_label[i],importances[indices[i]]))
```

```
 1) fixed acidity                  0.178690
 2) volatile acidity               0.128748
 3) citric acid                    0.106496
 4) residual sugar                 0.095718
 5) chlorides                      0.089280
 6) free sulfur dioxide            0.083482
 7) total sulfur dioxide           0.070669
 8) density                        0.070104
 9) pH                             0.060459
10) sulphates                      0.060048
11) alcohol                        0.056307
```

```
In [21]: plt.title('Feature Importances')
         plt.bar(range(X.shape[1]),importances[indices], color="green", align="center")
         plt.xticks(range(X.shape[1]),features_label, rotation=90)
         #plt.xlim([-1, X.shape[1]])
         plt.show()
```



- Splitting the dataset into Training set and Testing set

```
In [22]: # Splitting the dataset into the Training set and Test set
from sklearn.model_selection  import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 5)
```

- **Logistic Regression**

Using Logistic Regression

```
In [23]: from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, Y_train)

# Predicting Test Set
Y_pred = classifier.predict(X_test)
from sklearn.metrics import  accuracy_score, f1_score, precision_score, recall_score
acc = accuracy_score(Y_test, Y_pred)
prec = precision_score(Y_test, Y_pred)
rec = recall_score(Y_test, Y_pred)
f1 = f1_score(Y_test, Y_pred)

results = pd.DataFrame([['Logistic Regression', acc, prec, rec, f1]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])
print(results)
```

```
                 Model  Accuracy  Precision   Recall  F1 Score
0  Logistic Regression   0.86875   0.611111  0.23913   0.34375
```

- **Support Vector Classifier**

```
In [24]: from sklearn.svm import SVC
classifier = SVC(random_state = 0)
classifier.fit(X_train, Y_train)

# Predicting Test Set
Y_pred = classifier.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
prec = precision_score(Y_test, Y_pred)
rec = recall_score(Y_test, Y_pred)
f1 = f1_score(Y_test, Y_pred)

model_results = pd.DataFrame([['SVC', acc, prec, rec, f1]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

results = results.append(model_results, ignore_index = True)
print(results)
```

```
                 Model  Accuracy  Precision   Recall  F1 Score
0  Logistic Regression   0.86875   0.611111  0.23913   0.34375
1                  SVC   0.85625   0.000000  0.00000   0.00000
```

- **Random Forest Classifier**

Using Random Forest

In [25]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(random_state = 0, n_estimators = 100,
                                    criterion = 'entropy')
classifier.fit(X_train, Y_train)

# Predicting Test Set
Y_pred = classifier.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
prec = precision_score(Y_test, Y_pred)
rec = recall_score(Y_test, Y_pred)
f1 = f1_score(Y_test, Y_pred)

model_results = pd.DataFrame([['Random Forest (n=100)', acc, prec, rec, f1]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

results = results.append(model_results, ignore_index = True)
print(results)
```

```
                   Model  Accuracy  Precision    Recall  F1 Score
0    Logistic Regression   0.86875   0.611111  0.239130  0.343750
1                    SVC   0.85625   0.000000  0.000000  0.000000
2  Random Forest (n=100)   0.91250   0.846154  0.478261  0.611111
```

Result

As I used two machine learning algorithms: - Logistic Regression
Random forest classifier
Support Vector Classifier

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.86875	0.611111	0.239130	0.343750
1	SVC	0.85625	0.000000	0.000000	0.000000
2	Random Forest (n=100)	0.91250	0.846154	0.478261	0.611111

The Support Vector Classifier model, despite achieving an accuracy of 0.85625(85.625%), still has room for improvement. Further, the logistic regression model, despite achieving an accuracy of 0.86875(86.875%), still has room for improvement. It is possible that further feature engineering or model tuning may increase the accuracy of the model. At last Random Forest Classifier Model achieving an accuracy of 0.91250(91.25%).

Hence, I will prefer Random Forest Classifier algorithm for training my model as it has more accuracy.

This suggests that the random forest classifier may be a more suitable option when accuracy is of utmost importance, such as in medical diagnosis or fraud detection. Overall, these results highlight the importance of selecting appropriate classification techniques based on the specific context and goals of the predictive modeling project.

Summary

The Wine Quality Prediction project aimed to use Machine Learning algorithms to predict the quality of wine. We used three popular algorithms, Logistic Regression, Support Vector Classifier and Random Forest Classifier, to analyze various factors such as acidity, sugar content, pH, etc. Wine quality is a subjective measure that can vary from person to person, but it is generally based on the overall taste and aroma of the wine.

One of the most commonly used supervised learning techniques for wine quality prediction is regression analysis. Regression models aim to find a mathematical relationship between the input features and the quality ratings. The trained model can then be used to predict the quality of new wines based on their features.

After analyzing the data and comparing our predictions to the actual results, we found that the Random Forest Classifier algorithm was the most effective, achieving an accuracy of 91.25%. This high level of accuracy suggests that our model can be a valuable tool for winemakers looking to make informed predictions.

The project's significance lies in its ability to leverage data and Machine Learning algorithms to predict the outcome of wine quality. Feature selection and engineering are critical components of wine quality prediction. It is important to choose the most relevant features that contribute to wine quality and to engineer new features that capture important information. Feature selection can be done using techniques such as correlation analysis, while feature engineering can involve transforming existing features or creating new ones.

Overall, the Wine Quality Prediction project has demonstrated the potential of Machine Learning algorithms in predicting the quality of wine. We hope that this project will inspire further research in using ML algorithms for predicting the outcome of wine for winemakers and wine enthusiasts.

Bibliography

- https://www.kaggle.com/datasets/yasserh/wine-quality-dataset
- https://www.towardsdatascience.com
- https://learn.upgrad.com/
- https://www.javatpoint.com/machine-learning
- https://www.turing.com/
- https://builtin.com/data-science/

Annexure

Here are some possible details that could be included in an annexure for a wine quality prediction dataset in ML:

1. Data source: The wine quality prediction dataset may come from various sources, such as public databases, industry partners, or research institutions. It is important to document the origin of the dataset for transparency and reproducibility.

2. Data description: The wine quality prediction dataset should include a detailed description of the attributes and quality ratings. This may include the units of measurement, range of values, and any missing or outlier data. The description should also specify whether the quality ratings are based on expert or consumer evaluations.

3. Data preprocessing: The wine quality prediction dataset may require preprocessing steps to clean, transform, or normalize the data. This may involve handling missing or outlier values, scaling or standardizing the features, or encoding categorical variables. It is important to document these preprocessing steps for reproducibility.

4. Data split: The wine quality prediction dataset may need to be split into training and test sets for machine learning model training and evaluation. The split should be randomized and stratified if necessary to ensure that the training and test sets are representative of the overall dataset.

5. Data storage: The wine quality prediction dataset may be stored in various formats, such as CSV, Excel. It is important to document the storage format and location for easy access and retrieval.

6. Data usage: The wine quality prediction dataset may have restrictions on its usage, such as licensing agreements or privacy regulations. It is important to document these restrictions and obtain the necessary permissions before using the dataset for machine learning model training and evaluation.

7. Data citation: The wine quality prediction dataset should be properly cited in any publications or reports that use the dataset. The citation should include the data source, description, preprocessing steps, and any other relevant information.

8. Model Selection and Evaluation: We experimented with two popular Machine Learning algorithms, Logistic Regression, Support Vector Classifier and Random Forest Classifier, to predict the outcome of wine quality. We evaluated the performance of each model using metrics such as accuracy, precision, recall, and F1-score.

9. Conclusion and Future Work: Our project has demonstrated the potential of Machine Learning algorithms in predicting the quality of wine. In the future, we plan to explore other ML algorithms and incorporate more features to improve the accuracy of our model. Additionally, we hope to apply our methodology to other winemakers and contribute to the wine enthusiasts.

Overall, an annexure for a wine quality prediction dataset in ML can provide important information about the data source, description, preprocessing, storage, usage, and citation. This information can help ensure that the dataset is properly documented and used for transparent and reproducible machine learning model training and evaluation.