Wine Quality Prediction Using Machine Learning

Summer Training Report Submitted in Partial Fulfillment of the Requirement for the Degree of

B. Tech

In

Computer Science and Engineering

Training Coordinator

Mrs. Palak Girdhar

By

Ankit Gaur

Bhagwan Parshuram Institute of Technology

PSP-4, Sector-17, Rohini, Delhi – 89

July 2021 - August 2021

DECLARATION

This is to certify that the Report entitled "Wine Quality Prediction Using Machine Learning" which is submitted by me in partial fulfilment of the requirement for the award of degree B. Tech in Computer Engineering to BPIT, GGSIP University, Dwarka, Delhi comprises only my original work and due acknowledgement has been made in the text to all other material used.

Date:

Name of Student

Ankit Gaur

ACKNOWLEDGEMENT

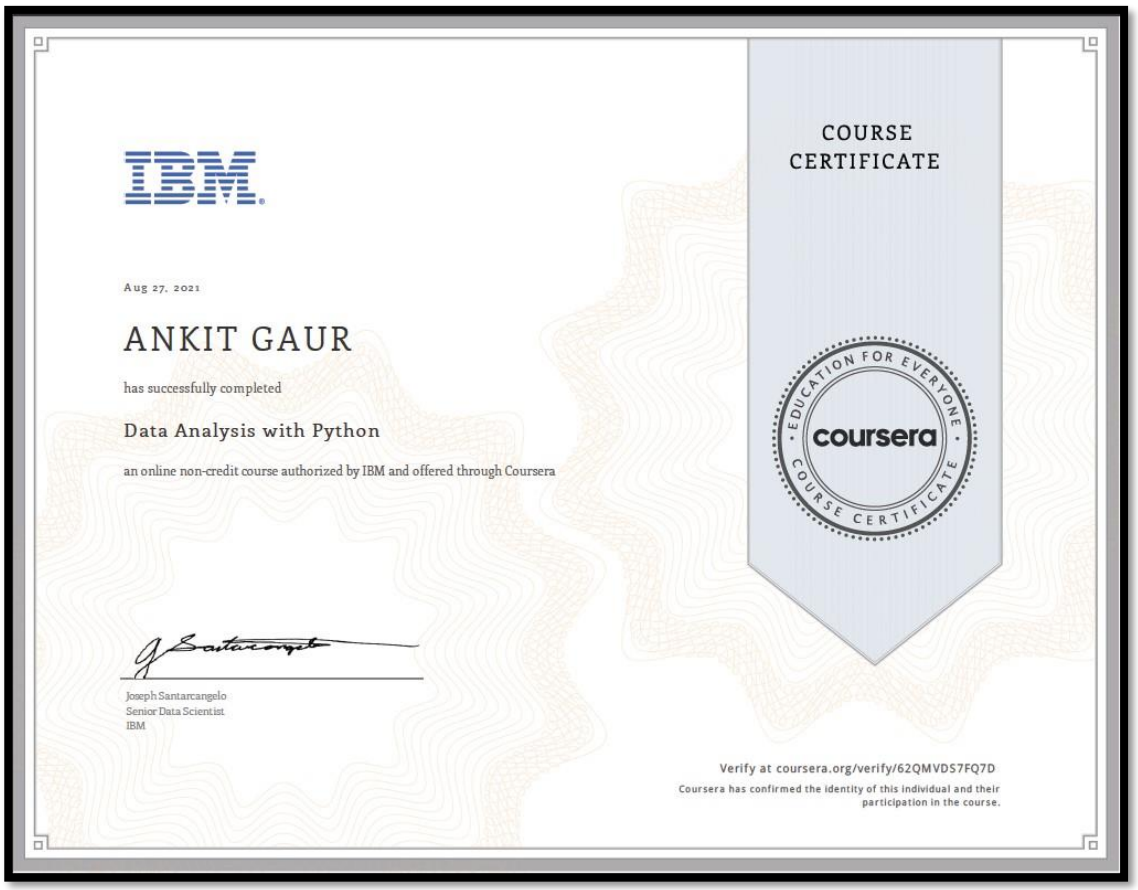
It is my pleasure to be indebted to various people, who directly or indirectly contributed to the development of this work and who influenced my thinking, behaviour, and acts during the study.

I am thankful to **Mrs Palak Girdhar** for providing me with constant inspiration with her presence and also guidance for my future endeavours, during the report drafting process.

I also extend my gratitude to **Dr Achal Kaushik** and **Prof. Payal Pahwa** who provided their valuable inputs and precious time in accomplishing our projects even during the Pandemic without fail

Lastly, I would like to thank the almighty and my parents for their moral support and my batchmates with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

COMPANY CERTIFICATE



Training Coordinator Certificate

This is to certify that the Report entitled “Wine Quality Prediction Using Machine Learning” which is submitted by Ankit Gaur in partial fulfilment of the requirement for the award of degree B.Tech in Computer Engineering to BPIT, GGSIP University, Dwarka, Delhi is a record of the candidate own work and the matter embodied in this report is adhered to the given format.

Date: **Coordinator**

Table of Contents

List of Figures & Tables

ABSTRACT

Wine is an alcoholic beverage made from grapes. 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 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.

This report has been generated from a perspective of a data scientist in a firm that is going to set up a wine factory. They asked for input for making better wine based on some chemical composition and parameters such as volatile acidity, citric acid, residual sugar etc.

The objective is to make a supervised machine learning system to predict the quality of the wine based on the data fed into it. For this, the used dataset is related to red variants of the Portuguese "Vinho Verde" wine. This dataset is available on Kaggle and can be viewed as a classification or regression task.

The methodology is fairly simple and straight forward first we collect labelled data from the internet (kaggle) then we analyze and make data more appropriate for our model to train. Then based on some analysis we select our random forest model or algorithm by using which we make our machine learn about the wine quality and what factors will be affecting the quality of the wine. Then we evaluate our model for new data and make some evaluation based on that we can say our model is more than 80% accurate hence we make a predictive system based on data entered to tell us whether wine quality is good or not with accuracy more than 80%.

In future, this can be more optimized for better accuracy and hence decreasing the time which is used by human tasters to evaluate the quality of wine as well as automating the process of wine quality analysis.

Chapter 1: INTRODUCTION

1.1 Project 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 increase 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 human experts it is a challenging task.

This report has been generated from a perspective of a data scientist in a firm that is going to set up a wine factory. They asked for input for making better wine based on some chemical composition and parameters such as volatile acidity, citric acid, residual sugar etc.

The project aims to predict whether the wine quality is good or not. Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol. And the output variable is quality.

In this report, we are explaining the steps we followed to build our models for predicting the quality of red wine in a simple non-technical way. We are dealing only with red wine. Classification is used to classify the wine as good or bad. Before examining the data it is often referred to as supervised learning because the classes are determined.

1.2 Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and predictive maintenance.

Machine learning starts with data — numbers, photos, or text, like bank transactions, pictures of people or even bakery items, repair records, time-series data from sensors, or sales reports. The data is gathered and prepared to be used as training data, or the information the machine learning model will be trained on. The more data, the better the program.

From there, programmers choose a machine learning model to use, supply the data, and let the computer model train itself to find patterns or make predictions. Over time the human programmer can also tweak the model, including changing its parameters, to help push it toward more accurate results There are three subcategories of machine learning:

Supervised machine learning models are trained with labelled data sets, which allow the models to learn and grow more accurate over time. For example, an algorithm would be trained with pictures of dogs and other things, all labelled by humans, and the machine would learn ways to identify pictures of dogs on its own. Supervised machine learning is the most common type used today.

In **unsupervised** machine learning, a program looks for patterns in unlabelled data. Unsupervised machine learning can find patterns or trends that people aren't explicitly looking for. For example, an unsupervised machine learning program could look through online sales data and identify different types of clients making purchases.

Reinforcement machine learning trains machines through trial and error to take the best action by establishing a reward system. Reinforcement learning can train models to play games or train autonomous vehicles to drive by telling the machine when it made the right decisions, which helps it learn over time what actions it should take.

1.3 Dataset

The dataset is related to the red wine of the Portuguese "Vinho Verde" wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.). These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

1) fixed acidity
2) volatile acidity
3) citric acid
4) residual sugar
5) chlorides
6) free sulfur dioxide
7) total sulfur dioxide
8) density
9) pH
10) sulphates
11) alcohol

Output variable (based on sensory data):

quality (score between 0 and 10)

Table 1: Dataset Attributes

Attributes	Description	Range
Fixed acidity	Impart sourness and resist microbial infection, measured in no. of grams of tartaric acid per dm^3	Numeric: 3.8-15.9
Volatile acidity	no. of grams of acetic acid per dm^3	Numeric: 0.1-1.6
Citric acid	no. of grams of citric acid per dm^3	Numeric: 0.0-1.7
Residual sugar	Remaining sugar after fermentation stops	Numeric: 0.6-65.8
Chlorides	no. of grams of sodium chlorides per dm^3	Numeric: 0.01-0.61
Free sulfur dioxide	no. of grams of free sulphites per dm^3	Numeric: 1-289
Total sulfur dioxide	no. of grams of total sulfite	Numeric: 6-440
Density	Density in gm per cm^3	Numeric: 0.987-1.039
pH	To measure ripeness	Numeric: 2.7-4
Sulphates	no. of grams of potassium sulphates per dm^3	Numeric: 0.2-2.0
Alcohol	Volume of alcohol in %	Numeric: 8.0-14.9
Quality	Target variable	Numeric: 0-10

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

1.4 Model Selection

1.4.1 Decision Tree

A Decision Tree is a typical portrayal, for instance, arrangement. It is Supervised Machine Learning where the information is constantly isolated by a particular boundary. A progression of preparing models is separated into more modest and more modest subsets while simultaneously steadily making a connected choice tree. A choice tree that covers the preparation set is returned toward the finish of the learning cycle. The key thought is to utilize a choice tree to segment the information space into (or thick) group areas and vacant (or scanty) districts. Another model is ordered in Decision Tree Classification, by sending it to a progression of tests that choose the model's class mark. In various levelled frameworks called a choice tree, such tests are requested. Choice Trees obey the Algorithm of Divide-and-Conquer. Decision trees are manufactured utilizing heuristic parcelling, called recursive apportioning. This strategy is additionally for the most part alluded to as separating and vanquishing since it partitions the information into subsets, which are then more than once isolated into much more modest sub-sets etc. until the cycle stops when the calculation concludes that the information in the subsets are adequately homogeneous or has another halting measure. Utilizing the choice calculation, we start from the base of the tree and split the information on the element that outcomes in the main increase of data (IG) (decrease of vulnerability towards the decision). Then we can rehash this parting strategy at every youngster hub in an iterative cycle until the leaves are unadulterated, which implies the examples at every hub of the leaf are the entirety of a similar class. By and by, to forestall overfitting, we can set a cut-off on the tree's profundity. Here we rather bargain on virtue, as the last leaves can, in any case, have some pollution

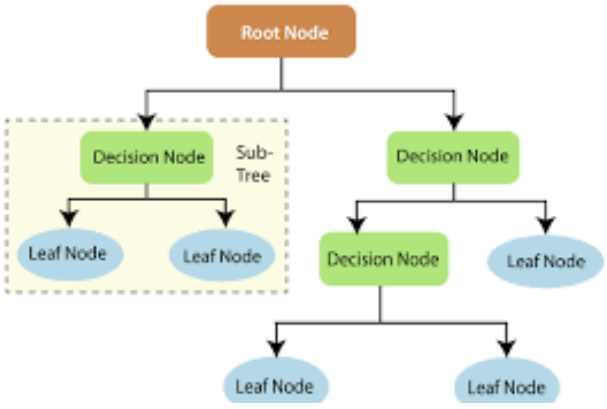


Figure 1: Decision tree diagram

1.4.2 Random Forest

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. Put simply: random forest builds multiple decision trees and merges them to get a more accurate and stable prediction.

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 the random forest in classification since classification is sometimes considered the building block of machine learning.

The 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 a 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).

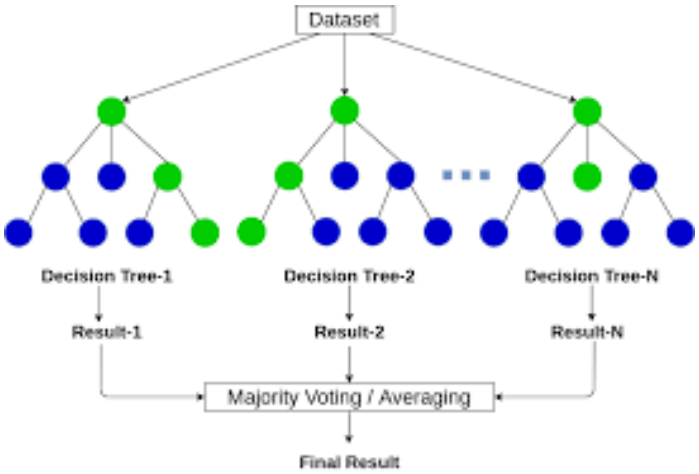


Figure 2: Random Forest diagram

Chapter 2: Software Requirements Specification (SRS)

2.1 Introduction

2.1.1 Purpose

This document is meant to delineate the feature of Wine Quality Prediction System so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

The project aims to predict whether the wine quality is good or not. Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol. And the output variable is quality

2.1.2 Scope

In this project we predict wine quality based on chemical composition of wine. 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 and time consuming. By using prediction system cost and time consumption can be decreased to a significant number. Also, effectiveness of the quality prediction can be increased. By using random forest classifier we as a data scientist can give a model that can predict the quality based on chemical parameters of wine.

2.2 Software requirement

- Operating System 8 or above
- Python 3.7 or above
- Jupyter notebook
- Wine Quality Dataset
- Python libraries:
    - Numpy
    - Pandas

- Matplotlib
- Seaborne
- Scikit learn

2.2 Hardware requirement

- Ram 1GB or more is required as it will provide fast reading and writing capabilities and will, in turn, support processing.
- Processor: x86 or x64 architecture, Pentium or later, and compatibles HDD: IDE/SATA with minimum 450 MB of free space on the primary partition

2.3 Functional requirement

- Opening jupyter notebook: The user should open any jupyter notebook having a python script.
- Running python script: go to cell ribbon of jupyter notebook and click on run all.
- Entering the input: At the end of the script, the user will be prompted to enter the data needed to predict the quality of the wine.

Chapter 3: DIAGRAMS

3.1 DFD

DFD level 0

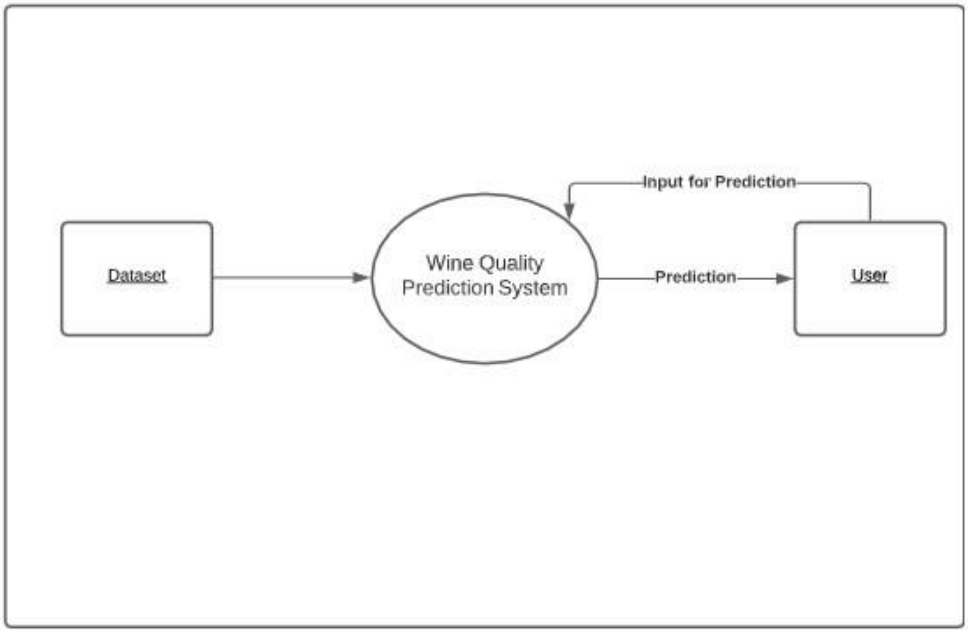


Figure 3: DFD level 0

DFD level 1

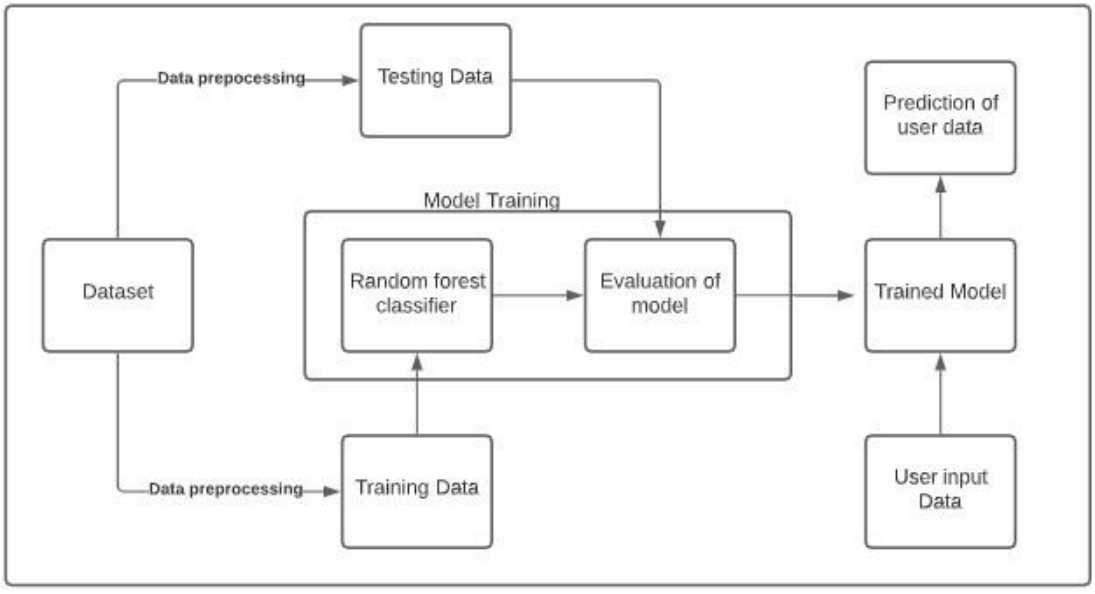


Figure 4: DFD level 1

3.2 Use case diagram

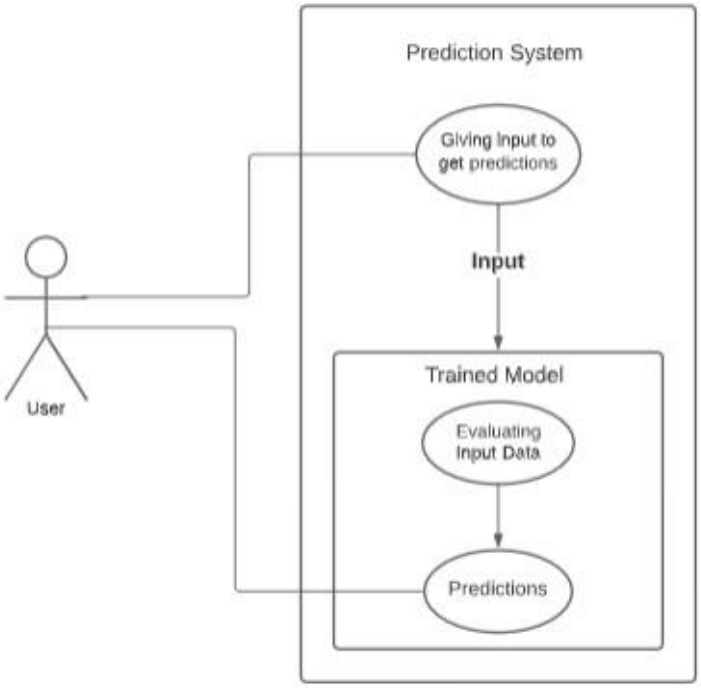


Figure 5: Use case diagram

Chapter 4: PROCESS SELECTION

4.1 Dependencies

4.1.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

There are many features in Python, some of which are discussed below –

- Easy to code: Python is a high-level programming language. Python is very easy to learn language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.
- Free and Open Source: Python language is freely available on the official website since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.
- Object-Oriented Language: One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.
- GUI Programming Support: Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.PyQt5 is the most popular option for creating graphical apps with Python.
- High-Level Language: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.
- Extensible feature: Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

- Python is a Portable language: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
- Python is an Integrated language: Python is also an Integrated language because we can easily integrate python with other languages like c, c++, etc.
- Interpreted Language: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.
- Large Standard LibraryPython has a large standard library that provides a rich set of modules and functions so you do not have to write your code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.
- Dynamically Typed Language: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

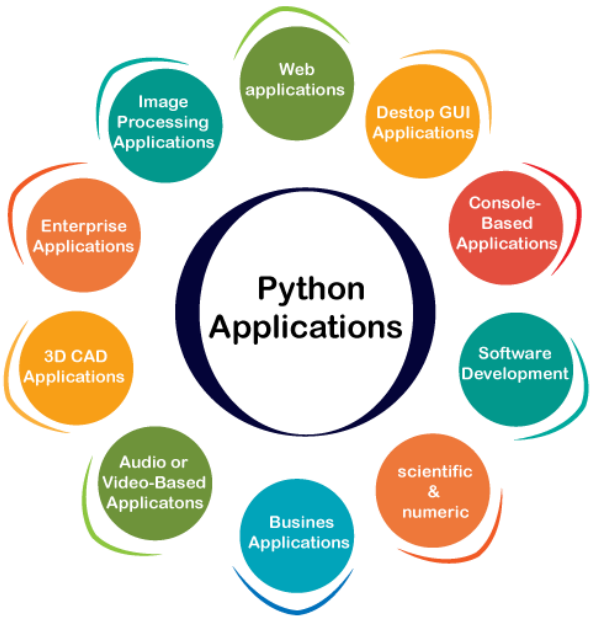


Figure 6: Python applications

4.1.2 Numpy

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into the Numeric package. There are many contributors to this open-source project.

4.1.2.1 Operations using NumPy

Using NumPy, a developer can perform the following operations −

• Mathematical and logical operations on arrays.

• Fourier transforms and routines for shape manipulation.

• Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

4.1.2.2 Use of NumPy

In Python, we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

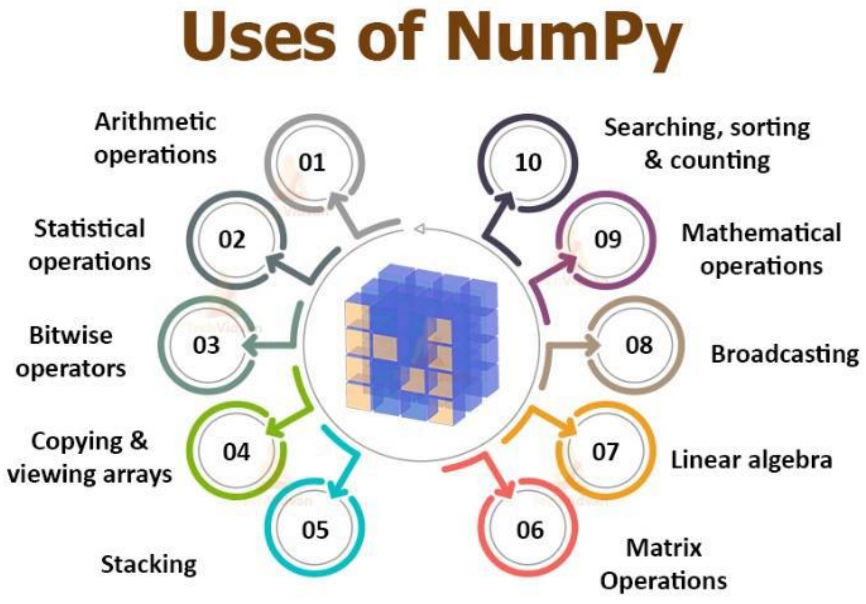


Figure 7: Uses of Numpy

4.1.3 Pandas

Pandas is an open-source Python library providing high-performance data manipulation and analysis tools using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tools for analysis of data.

Before Pandas, Python was majorly used for data munging and preparation. It had very little contribution to data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, statistics, analytics, etc.

4.1.3.1 Features of Pandas

- Fast and efficient DataFrame object with the default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Label-based slicing, indexing, and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High-performance merging and joining of data.

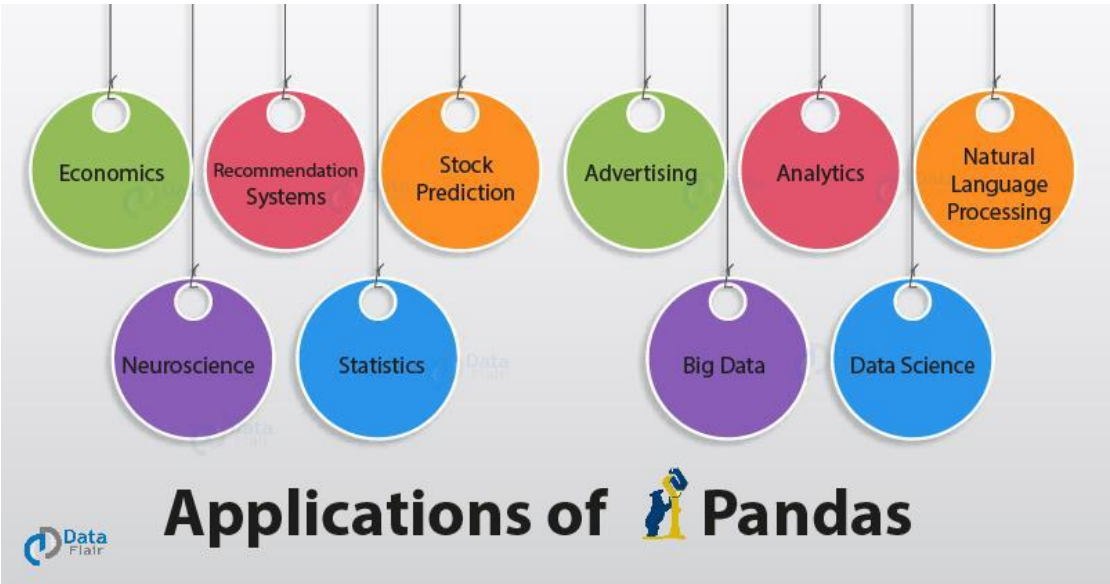


Figure 8: Application of Pandas

4.1.4 Matplotlib

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook, and web application servers also.

Matplotlib has a procedural interface named Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open-source equivalent of MATLAB.

Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018

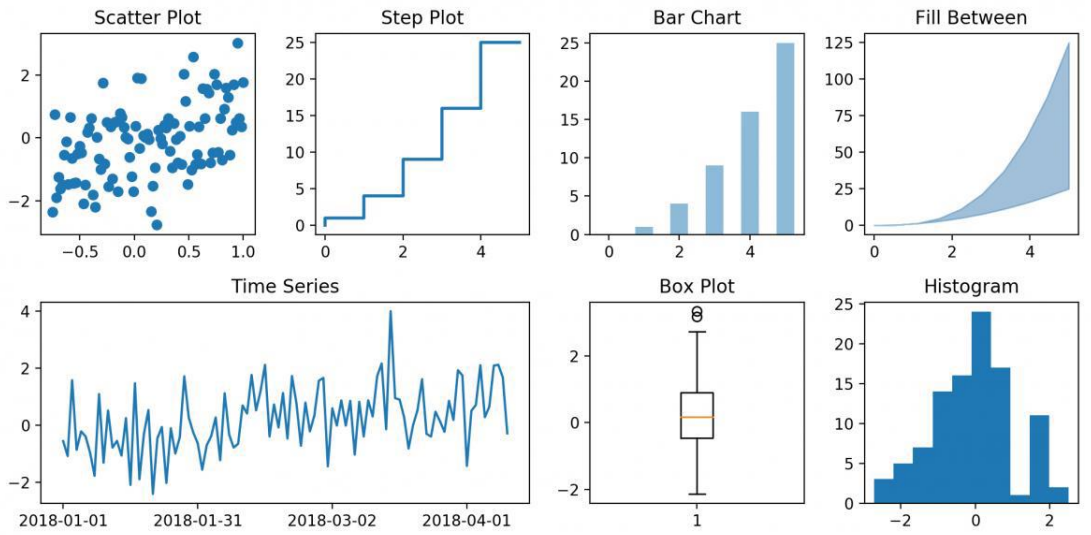


Figure 9: Example of Matplotlib graphs

4.1.5 Seaborne

In the world of Analytics, the best way to get insights is by visualizing the data. Data can be visualized by representing it as plots that are easy to understand, explore, and grasp. Such data helps in drawing the attention of key elements.

To analyze a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

4.1.6 Sci kit learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

It was originally called *scikit.learn* and was initially developed by David Cournapeau as a Google Summer of Code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA

4.2 Data collection

```
In [2]: #Loading Data to pandas Data frame
        Wine_Dataset = pd.read_csv("winequality-red.csv")
```

Figure 10: loading the dataset

Importing a preloaded Dataset from the Computer using Pandas function read_csv. It saves the data from the CSV in Pandas dataframe. This dataframe is stored in the variable Wine_Dataset.

```
In [3]: # Number of Rows and Columns in Dataset
        Wine_Dataset.shape

Out[3]: (1599, 12)
```

Figure 11: shape of dataset

Checking shape attribute of dataframe for getting the number of columns and rows. It confirms whether the Dataset is loaded correctly or not.

```
In [4]: # Printing first 5 rows of Dataset
        Wine_Dataset.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

Figure 12: first 5 value of dataset

Printing first 5 values of Dataset to get the overview of the Dataset

```
In [5]: Wine_Dataset.info()

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

Figure 13: info of dataset

Viewing type of value stored in the dataset. So that we can find which values are categorical type and which are of the value type.

```
In [6]: # printing names of all columns in Data set
        list(Wine_Dataset.columns)

Out[6]: ['fixed acidity',
         'volatile acidity',
         'citric acid',
         'residual sugar',
         'chlorides',
         'free sulfur dioxide',
         'total sulfur dioxide',
         'density',
         'pH',
         'sulphates',
         'alcohol',
         'quality']
```

Figure 14: list of columns in dataset

Printing the list of column names.

```
In [7]: # Checking for missing values in Dataset
        Wine_Dataset.isnull().sum()
        # from the output we can say there are no missing values in dataset

Out[7]: 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
```

Figure 15: handling null values in dataset

It is used to print the sum of null values for each column to check for the null values in it and here we get to know that there are no null values present in the dataset.

4.3 Data Analysis and visualization

```
In [8]: # statistical measures of the dataset
        Wine_Dataset.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000

Figure 16: statical analysis of dataset

It is a statistical analysis of the dataset column-wise. It gives the number of values, mean, standard deviation, minimum value stored, the maximum value stored, value at 25 percentile, 50 percentile, 75 percentile.

```
In [9]: # number of values for each quality
        sns.catplot(x='quality', data = Wine_Dataset, kind = 'count')

Out[9]: <seaborn.axisgrid.FacetGrid at 0x15435688850>
```

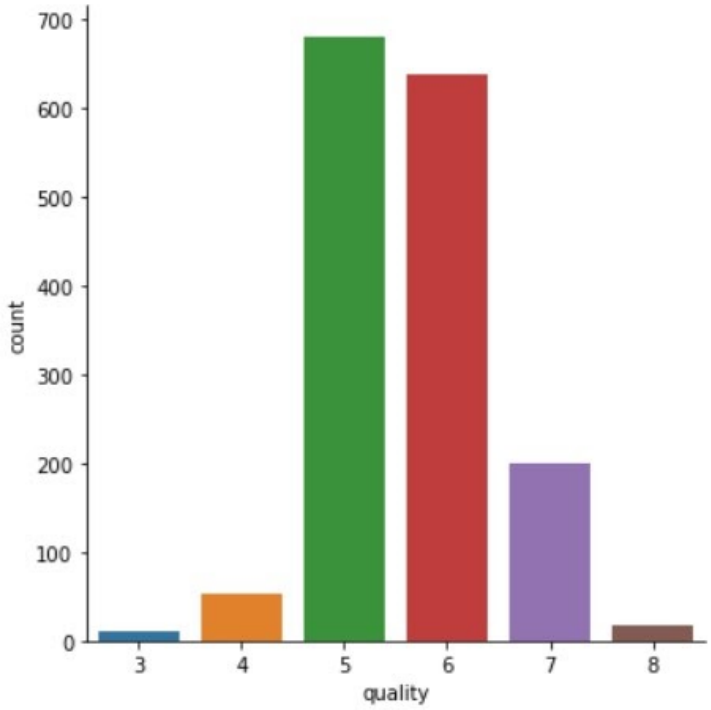


Figure 17: graph quality count

It is a categorical plot of quality. It gives how many different values are there in the quality column and what is count if each type of quality. From the graph, we can see it is very rare to have a wine having quality low as 3. At the same time, there is also very rare to have quality high as 7 or 8. Most of the wines in the dataset are average quality that is 5 and 6. So we can divide our quality column into 2 parts such as 6 or above and less than 6 these will show the quality of wine is good or bad respectively.

```
In [10]: Wine_Dataset.hist(bins=25,figsize=(10,10))

Out[10]: array([[<AxesSubplot:title={'center':'fixed acidity'}>,
                 <AxesSubplot:title={'center':'volatile acidity'}>,
                 <AxesSubplot:title={'center':'citric acid'}>],
                [<AxesSubplot:title={'center':'residual sugar'}>,
                 <AxesSubplot:title={'center':'chlorides'}>,
                 <AxesSubplot:title={'center':'free sulfur dioxide'}>],
                [<AxesSubplot:title={'center':'total sulfur dioxide'}>,
                 <AxesSubplot:title={'center':'density'}>,
                 <AxesSubplot:title={'center':'pH'}>],
                [<AxesSubplot:title={'center':'sulphates'}>,
                 <AxesSubplot:title={'center':'alcohol'}>,
                 <AxesSubplot:title={'center':'quality'}>]], dtype=object)
```

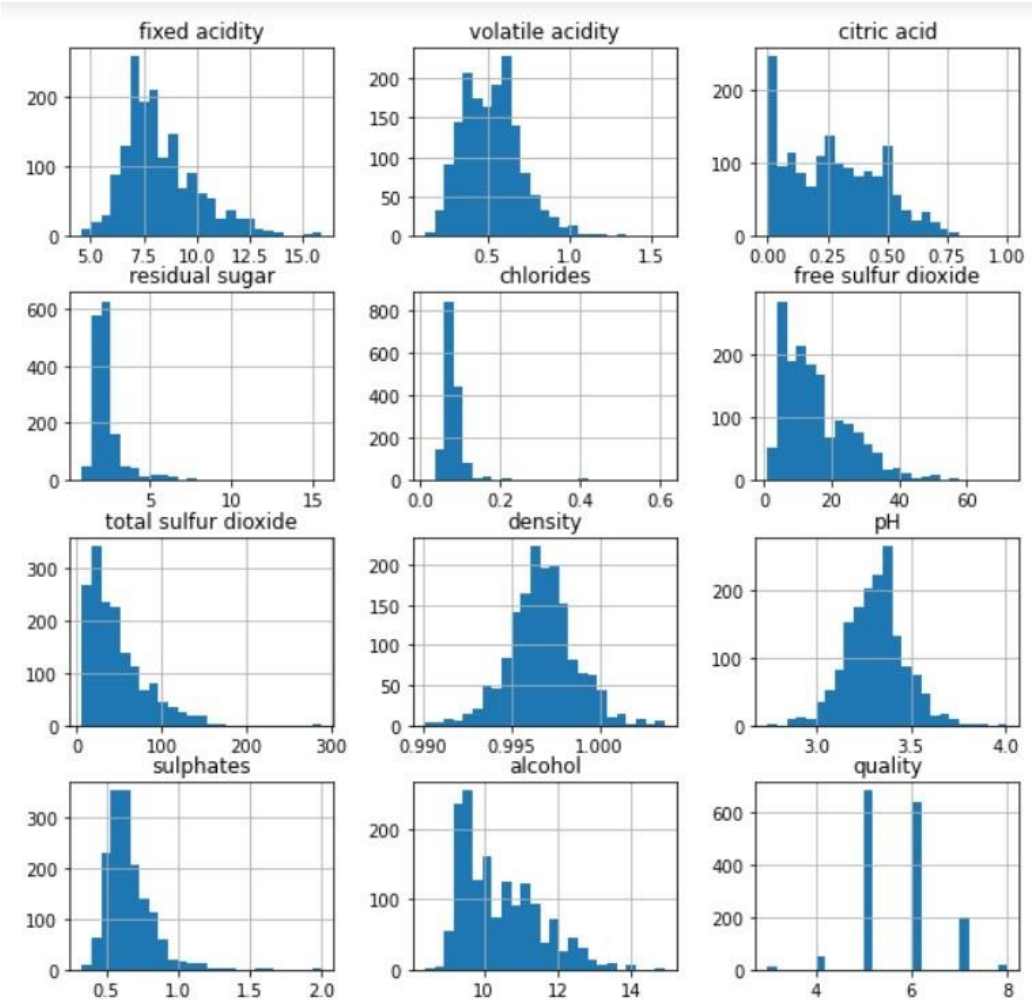


Figure 18: distribution graph

These are histograms showing the distribution of all values in each column of the dataset. From these graphs, we can see all values seem continuous but the quality is not continuous rather discrete. This shows all columns except the quality is having floating points due to which the graph seems to be continuous. Also, we can deduce that there are multiple outliers in the graphs.

```
In [11]: # fixed acidity vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'fixed acidity', data = Wine_Dataset)

Out[11]: <AxesSubplot:xlabel='quality', ylabel='fixed acidity'>
```

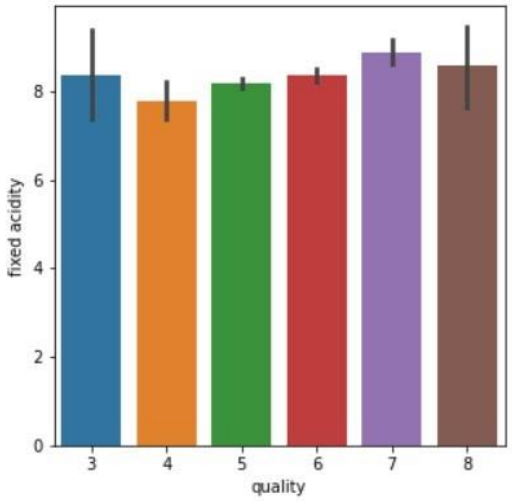


Figure 19: graph fixed acidity vs Quality

This graph is a bar plot showing the relation of fixed acidity with quality from the graph we can deduce as quality increase the fixed acidity seems to be changing but true relation can be found only by further analysis.

```
In [12]: # volatile acidity vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'volatile acidity', data = Wine_Dataset)

Out[12]: <AxesSubplot:xlabel='quality', ylabel='volatile acidity'>
```

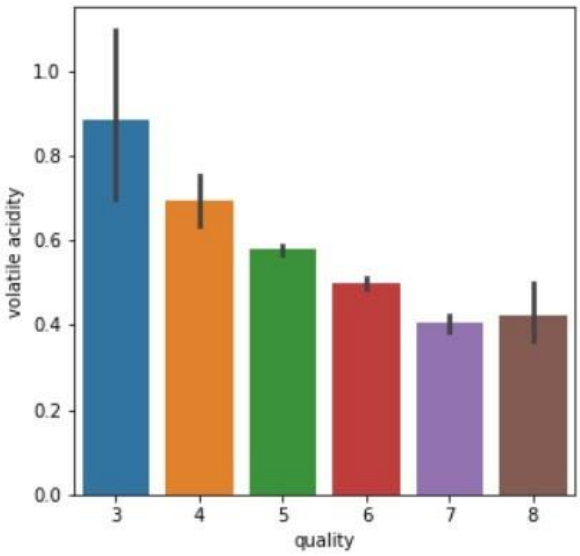


Figure 20: graph volatile acidity vs Quality

This graph is a bar plot showing the relation of volatile acidity with quality from the graph we can deduce as quality increases and the volatile acidity decreases.

```
In [13]: # citric acid vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'citric acid', data = Wine_Dataset)

Out[13]: <AxesSubplot:xlabel='quality', ylabel='citric acid'>
```

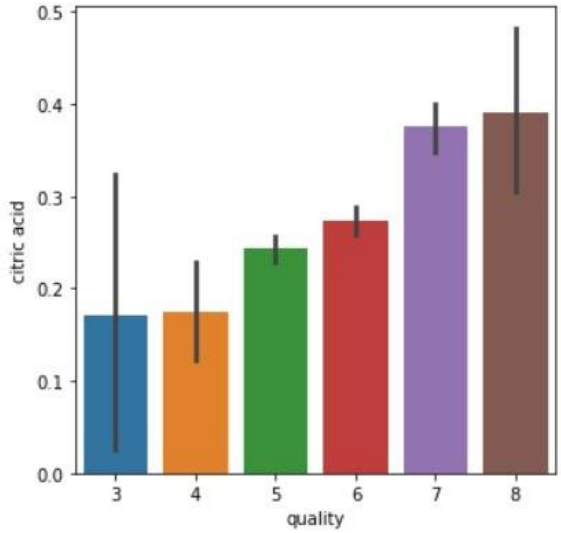


Figure 21: graph citric acid vs Quality

This graph is a bar plot showing the relation of citric acid with quality from the graph we can deduce as quality increases the citric acid increases.

```
In [14]: # residual sugar vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'residual sugar', data = Wine_Dataset)

Out[14]: <AxesSubplot:xlabel='quality', ylabel='residual sugar'>
```

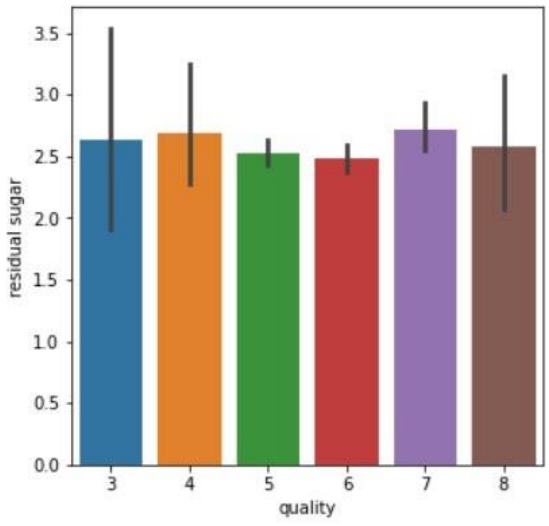


Figure 22: residual sugar vs Quality

This graph is a bar plot showing the relation of residual sugar with quality from the graph we can deduce as quality increases the residual sugar seems to be changing but the true relation can be found only by further analysis.

```
In [15]: # chlorides vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'chlorides', data = Wine_Dataset)

Out[15]: <AxesSubplot:xlabel='quality', ylabel='chlorides'>
```

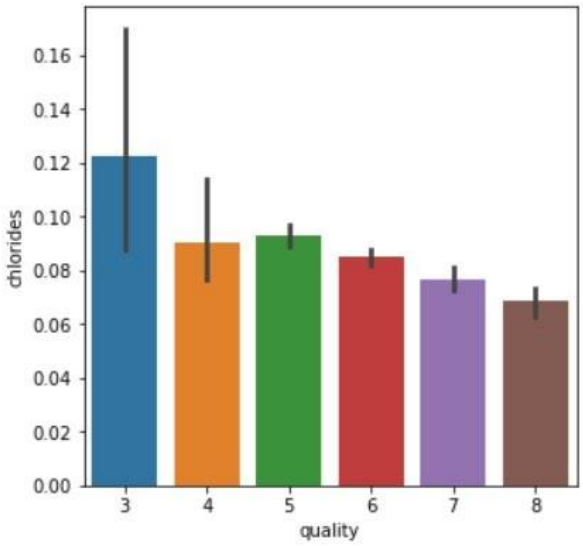


Figure 23: graph chlorides vs Quality

This graph is a bar plot showing the relation of chlorides with quality from the graph we can deduce as quality increases and the chlorides decrease.

```
In [16]: # free sulfur dioxide vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'free sulfur dioxide', data = Wine_Dataset)

Out[16]: <AxesSubplot:xlabel='quality', ylabel='free sulfur dioxide'>
```

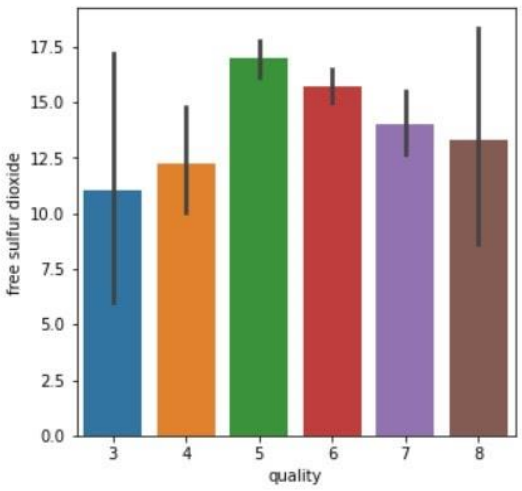


Figure 24: graph free sulfer dioxide vs Quality

This graph is a bar plot showing the relation of free sulfur dioxide with quality from the graph we can deduce as quality increases the free sulfur dioxide seems to be changing but the true relationship can be found only by further analysis.

```
In [17]: # total sulfur dioxide vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'total sulfur dioxide', data = Wine_Dataset)
```

```
Out[17]: <AxesSubplot:xlabel='quality', ylabel='total sulfur dioxide'>
```

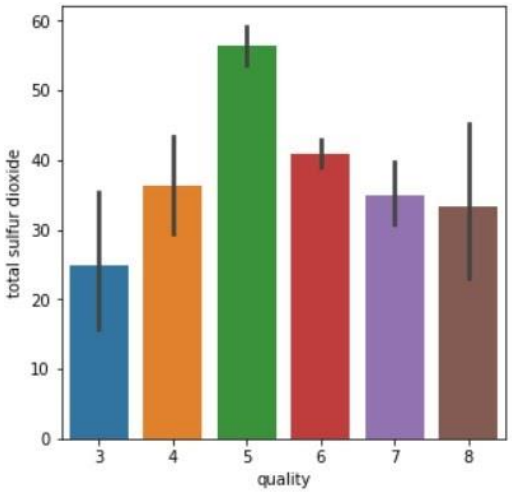


Figure 25: Total sulfer dioxide vs Quality

This graph is a bar plot showing the relation of total sulfur dioxide with quality from the graph we can deduce as quality increases the total sulfur dioxide decrease.

```
In [18]: # density vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'density', data = Wine_Dataset)
```

```
Out[18]: <AxesSubplot:xlabel='quality', ylabel='density'>
```

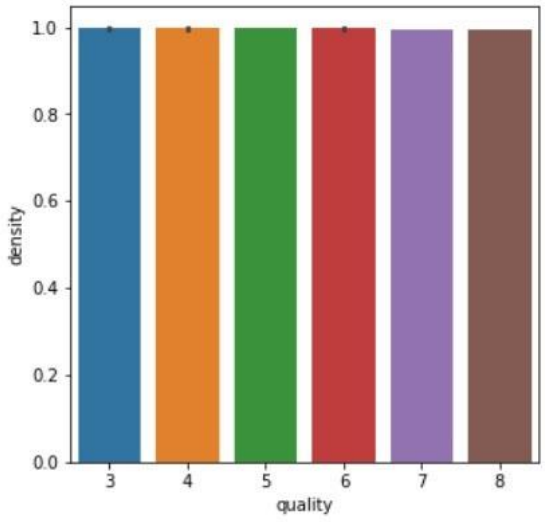


Figure 26: density vs Quality

This graph is a bar plot showing the relation of density with quality from the graph we can deduce as quality increases the density seems to be changing but the true relationship can be found only by further analysis.

```
In [19]: # pH vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'pH', data = Wine_Dataset)
```

```
Out[19]: <AxesSubplot:xlabel='quality', ylabel='pH'>
```

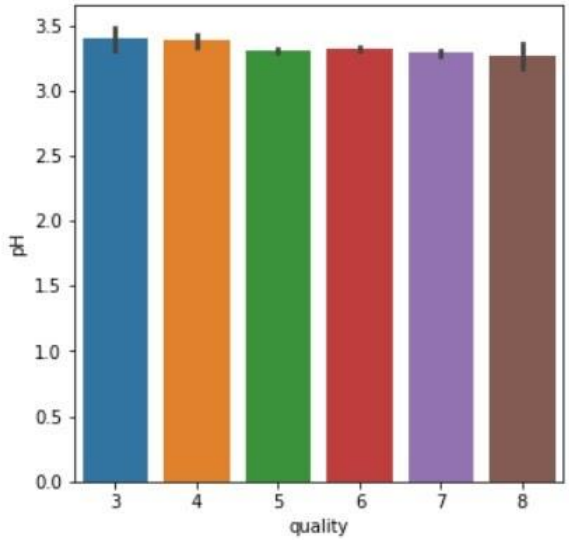


Figure 27: Graph pH vs Quality

This graph is a bar plot showing the relation of pH with quality from the graph we can deduce as quality increases the pH seems to be changing but the true relationship can be found only by further analysis.

```
In [20]: # sulphates vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'sulphates', data = Wine_Dataset)
```

```
Out[20]: <AxesSubplot:xlabel='quality', ylabel='sulphates'>
```

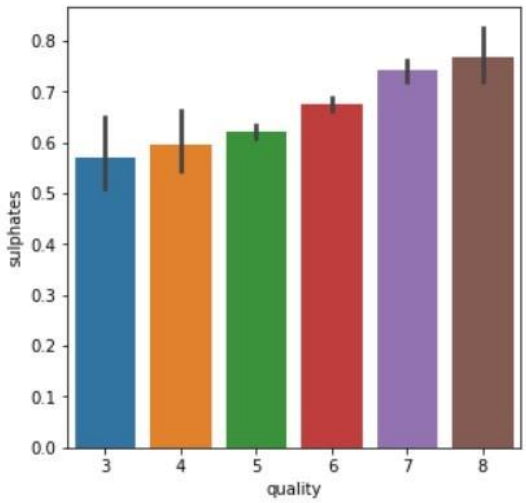


Figure 28: graph sulphates vs Quality

This graph is a bar plot showing the relation of sulphates with quality from the graph we can deduce as quality increases the sulphates increase.

```
In [21]: # alcohol vs Quality
         plot = plt.figure(figsize=(5,5))
         sns.barplot(x='quality', y = 'alcohol', data = Wine_Dataset)
```

```
Out[21]: <AxesSubplot:xlabel='quality', ylabel='alcohol'>
```

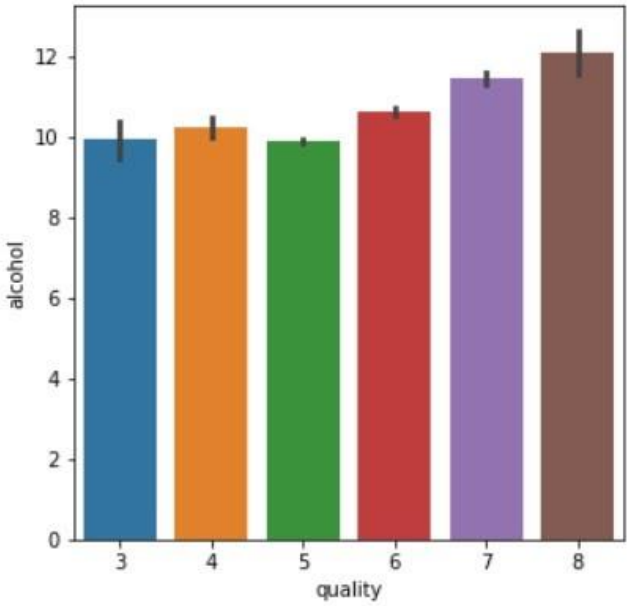


Figure 29: graph alcohol vs Quality

This graph is a bar plot showing the relation of alcohol with quality from the graph we can deduce as quality increases the alcohol increase.

4.4 Correlation

Correlation explains how one or more variables are related to each other. These variables can be input data features that have been used to forecast our target variable. If two variables are closely correlated, then we can predict one variable from the other. Correlation plays a vital role in locating the important variables on which other variables depend. It's used as the foundation for various modelling techniques. Proper correlation analysis leads to a better understanding of data.

There are three categories of correlation.

- Positive Correlation: Two features (variables) can be positively correlated with each other. It means that when the value of one variable increases then the value of the other variable(s) also increases.
- Negative Correlation: Two features (variables) can be negatively correlated with each other. It means that when the value of one variable increases then the value of the other variable(s) decreases.
- No Correlation: Two features (variables) are not correlated with each other. It means that when the value of one variable increase or decrease then the value of the other variable(s) doesn't increase or decrease.

We use the Pearson coefficient correlation matrices to calculate the correlation between the features. A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable in the table is correlated with each of the other values in the table. This allows you to see which pairs have the highest correlation. The diagonal of the table is always a set of ones because the correlation between a variable and itself is always 1.

```
In [22]: # finding correlation in between all columns
         correlation = Wine_Dataset.corr()

In [23]: # making heatmap for visualizing correlation
         plt.figure(figsize=(10,10))
         sns.heatmap(correlation, cbar=True, square=True, fmt = '.2f', annot = True, annot_kws={'size':8}, cmap = 'Blues')
Out[23]: <AxesSubplot:>
```

Figure 30: making correlation

Finding a correlation between all the columns in the dataset. The correlation matrix is stored in variable correlation. Then that matrix is plotted using a heatmap.

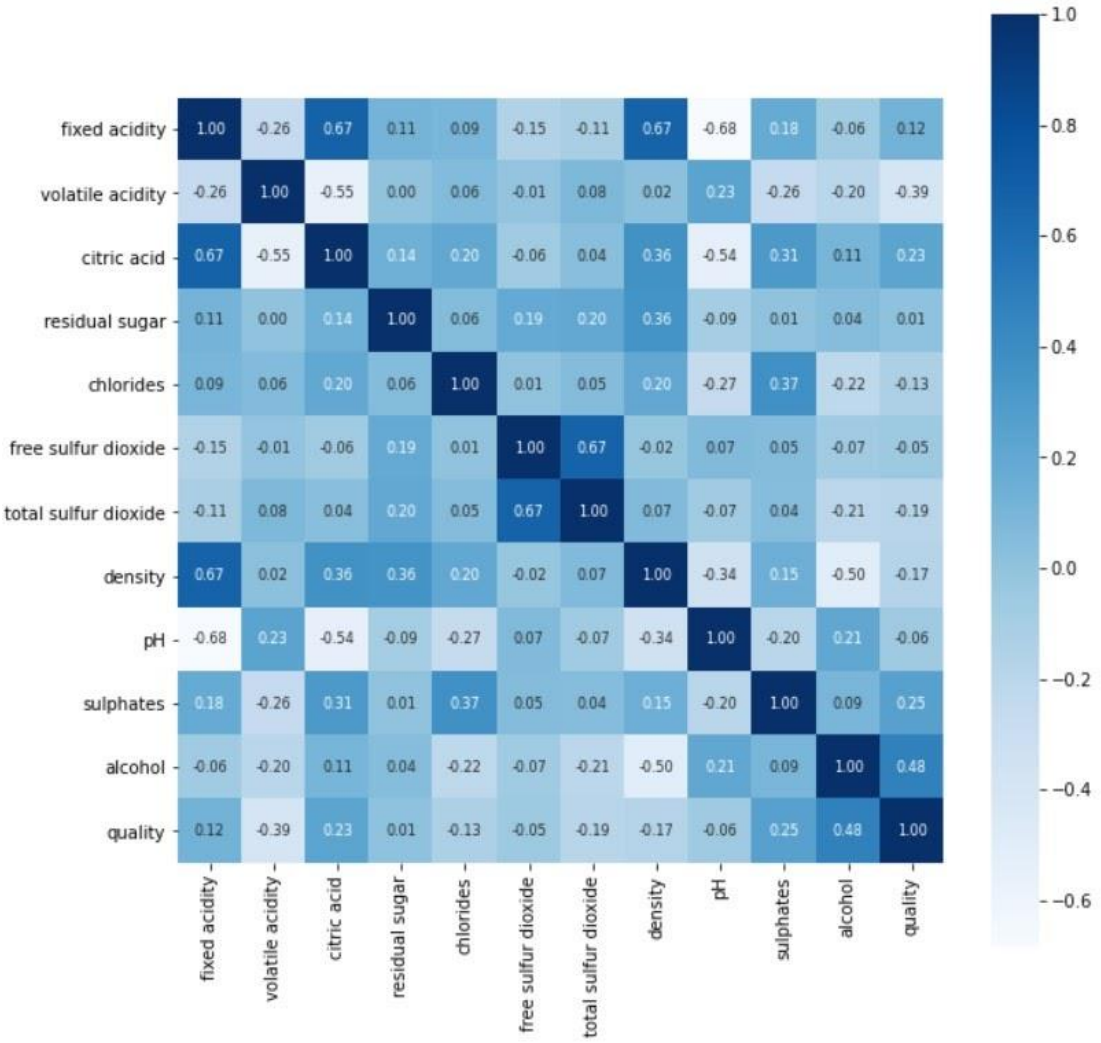


Figure 31: Heatmap of correlation Matrix

From the heat map, we can see the value of correlation between all the columns. From the heat map, we can deduce some attributes have a positive effect on quality such as alcohol, sulphates etc. Some have negative effects on correlation such as volatile acidity, chlorides etc.

4.5 Data preprocessing

Sometimes, the data collected is in the raw form and needs to be pre-processed.
Example: Some tuples may have missing values for certain attributes, and, in this case, it has to be filled with suitable values to perform machine learning or any form of data mining.
Missing values for numerical attributes such as the price of the house may be replaced with the mean value of the attribute whereas missing values for categorical attributes may be replaced with the attribute with the highest mode. This invariably depends on the types of filters we use. If data is in the form of text or images then converting it to numerical form will be required, be it a list or array or matrix. Simply, Data is to be made relevant and consistent. It is to be converted into a format understandable by the machine.
It is a crucial part of any machine learning project. In this step, we divide the data into two parts namely data and label. Label is the column that is predicted for a classification model most of the time the values are 0 and 1. Data are the columns that are used to predict the label. These columns have some kind of effect on the label column.

```
In [24]: # Seprating Data and Label
         X = Wine_Dataset.drop('quality',axis=1) # droping quality
```

Figure 32:Sepration of data and label

Separating data from the label so that we don't get confused about what to predict.

```
In [25]: # checking if Data is seprated from Label or not
         X.head()
```

Out[25]:

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

Figure 33: printing data

By printing the first 5 values we can see that the data is completely separated from the label.

```
In [26]: X.shape
Out[26]: (1599, 11)
```

Figure 34: shape of data

This confirms that data is separated from the label as the dataset had 12 columns now we separated the label so only 11 columns remain.

```
In [27]: list(X.columns)
Out[27]: ['fixed acidity',
          'volatile acidity',
          'citric acid',
          'residual sugar',
          'chlorides',
          'free sulfur dioxide',
          'total sulfur dioxide',
          'density',
          'pH',
          'sulphates',
          'alcohol']
```

Figure 35: list of columns in data

Printing the list of column names. Through that, we can say we got all the necessary columns in the X variable.

Now its time for the label

Label Encoding or label Binarization is used to convert the labels into the numeric form to convert it into the machine-readable form. It is an important preprocessing step for the structured dataset in supervised learning. We have used label encoding to label the quality of data as good or bad. Assigning 1 to good and 0 to bad.

```
In [28]: Y = Wine_Dataset['quality'].apply(lambda y_value: 1 if y_value>=6 else 0)
```

Figure 36: Label Binarization

Using the lambda function we can convert the quality label to 0 and 1. As we discussed earlier if the quality is greater than equal to 6 quality is good that is 1 and if it's less than 6 then it's bad that is 0.

```
In [29]: # number of values for each quality good and bad
         #sns.catplot(x='quality', data = Y, kind = 'count')
         Y.value_counts().plot.pie(figsize=(8,8),title="Quality",fontsize=12,legend=True)
Out[29]: <AxesSubplot:title={'center':'Quality'}, ylabel='quality'>
```

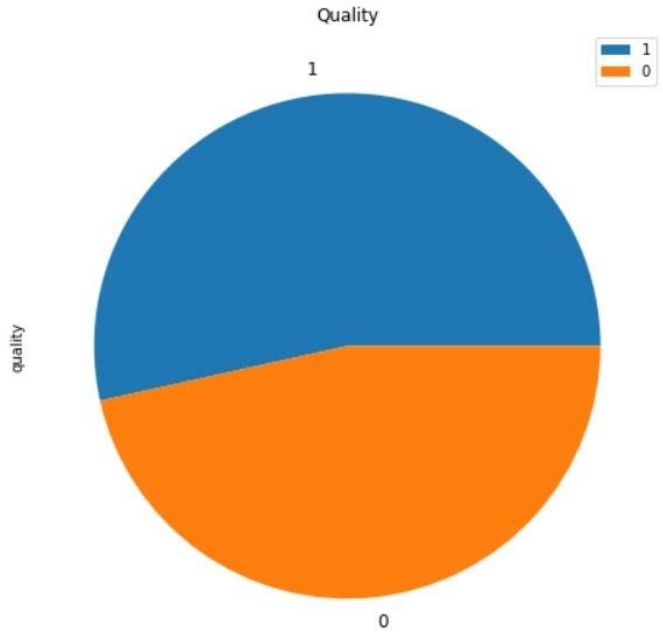


Figure 37: ploting distribution of good and bad wine in label

By plotting a pie chart we can see we successfully converted all values to 0 and 1. Also, we can deduce that more than 50% of the wine is good in our dataset according to our definition of good quality wine.

```
In [30]: print(Y.shape)
         (1599,)

In [31]: print(Y)
         0       0
         1       0
         2       0
         3       1
         4       0
                ..
         1594    0
         1595    1
         1596    1
         1597    0
         1598    1
         Name: quality, Length: 1599, dtype: int64
```

Figure 38: Shape of label and values in table

Printing the shape and value of Y that label to see how it is saved in our system.

4.6 Train test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The train-test procedure is appropriate when there is a sufficiently large dataset available.

The idea of “sufficiently large” is specific to each predictive modelling problem. It means that there is enough data to split the dataset into train and test datasets and each

of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain. A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

There is a predefined function in the sci-kit learn library to split the data into training and testing. Ideally, 80% of data is used in training while 20% of data is used in testing.

```
In [32]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)

In [33]: print( X.shape, X_train.shape, X_test.shape, Y.shape, Y_train.shape, Y_test.shape)
         (1599, 11) (1279, 11) (320, 11) (1599,) (1279,) (320,)
```

Figure 39: train test split

We have saved training data in X_train, training label in Y_train, testing data in X_test, and testing label into Y_test. By using four variables we successfully divided all the datasets into training and testing. By summing the number of rows and columns we can produce a perfect shape of our dataset.

4.7 Model Training

It is the phase of a project where machines learn. Here we use the predefined function which contains certain algorithms through which we want to train our machine for the data we provide. This process is done in two steps. First, we create an object or instance of our algorithm.

```
In [34]: # Making Model
         Model = RandomForestClassifier()
```

Figure 40: making instance of model

For this project, we use a random forest algorithm so we make an instance of the random forest classifier. This class is pre-defined in the sci-kit-learn library.

The second step is to fit the data in the model.

```
In [35]: # training or Fitting the Model
         Model.fit(X_train,Y_train)

Out[35]: RandomForestClassifier()
```

Figure 41: fitting the data in model

Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A well-fitted model produces more accurate outcomes.

Once the model is fitted it is now ready to make predictions.

4.8 Model Testing and evaluation

Model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. The testing set consisting of a set of testing samples should be separated from both training and validation sets, but it should follow the same probability distribution as the training set.

In model testing, we use our test set to make predictions on that. Then we use those predictions to evaluate the accuracy and other matrices for the model. It gives us a view of how our model will perform in the real world.

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.

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\textit{True Positive + False Postive + False Negative +True Negative}}$$

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

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive+False Positive}}$$

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,

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

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.

$$\text{F1 score} = 2 * \frac{Recall * Precision}{Recall + Precision}$$

```
In [36]: # Accuracy on Test Data
         X_test_prediction = Model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [37]: print('Accuracy : ', test_data_accuracy)

         Accuracy :  0.8375
```

Figure 42: Accuracy score

X_test_prediction is the predictions made on the testing data. For evaluation, we compare the predicted values to our actual values.

We use different functions to calculate different scores. For accuracy, we use the accuracy_score function. We can see our model is 84% accurate for any new data provided to it.

Let's now calculate other evaluation matrices.

```
In [38]: # Printing Classification Report
         report=classification_report(Y_test, X_test_prediction)
         print(report)

                       precision    recall  f1-score   support

                    0       0.82      0.84      0.83       152
                    1       0.85      0.83      0.84       168

             accuracy                           0.84       320
            macro avg       0.84      0.84      0.84       320
         weighted avg       0.84      0.84      0.84       320
```

Figure 43: classification report

From this, we can see that all the evaluation scores are given such as precision, recall, f1-score, etc.

4.9 Predictive System

Last but not least we made a predictive system in which we ask the user to enter the attributes of his/her wine based on that data and model we trained we predict whether the wine is good or not.

```
In [39]: print("Lets predict the quality of our Wine")
         fixed_acidity = input("Enter the fixed acidity: ")
         volatile_acidity = input("Enter the volatile acidity: ")
         citric_acid = input("Enter the citric acid: ")
         residual_sugar = input("Enter the residual sugar: ")
         chlorides = input("Enter the chlorides: ")
         free_sulfur_dioxide = input("Enter the free sulfur dioxide: ")
         total_sulfur_dioxide = input("Enter the total sulfur dioxide: ")
         density = input("Enter the density: ")
         pH = input("Enter the pH: ")
         sulphates = input("Enter the sulphates: ")
         alcohol = input("Enter the alcohol: ")

         input_data=(fixed_acidity,volatile_acidity,citric_acid,residual_sugar,chlorides,free_sulfur_dioxide,
                     total_sulfur_dioxide,density,pH,sulphates,alcohol)

         # changing the input data to a numpy array
         input_data_as_numpy_array = np.asarray(input_data)

         # reshape the data as we are predicting the label for only one instance
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

         prediction = Model.predict(input_data_reshaped)
         print(prediction)

         if (prediction[0]==1):
           print('Good Quality Wine')
         else:
           print('Bad Quality Wine')
```

```
Lets predict the quality of our Wine
Enter the fixed acidity: 20
Enter the volatile acidity: 20
Enter the citric acid: 20
Enter the residual sugar: 20
Enter the chlorides: 20
Enter the free sulfur dioxide: 20
Enter the total sulfur dioxide: 023
Enter the density: 20
Enter the pH: 20
Enter the sulphates: 20
Enter the alcohol: 20
[0]
Bad Quality Wine
```

Figure 44: predictive system

Chapter 5: RESULT

5.1 Predictions

Now User can get the predictions from the model based on the input the user provides.

The predictions will be accurate to a percentage of more than 80%.

Users have to enter parameters of the wine such as volatile acidity, pH, density etc. then the model will predict the quality of that fine and return whether the wine has good quality or bad quality.

```
Lets predict the quality of our Wine
Enter the fixed acidity: 7.3
Enter the volatile acidity: 0.65
Enter the citric acid: 0.0
Enter the residual sugar: 1.2
Enter the chlorides: 0.065
Enter the free sulfur dioxide: 15.0
Enter the total sulfur dioxide: 21.0
Enter the density: 0.9946
Enter the pH: 3.39
Enter the sulphates: 0.47
Enter the alcohol: 10.0
[1]
Good Quality Wine
```

Figure 45: example prediction 1

```
Lets predict the quality of our Wine
Enter the fixed acidity: 7.5
Enter the volatile acidity: 0.5
Enter the citric acid: 0.36
Enter the residual sugar: 6.1
Enter the chlorides: 0.171
Enter the free sulfur dioxide: 17.0
Enter the total sulfur dioxide: 102.0
Enter the density: 0.9978
Enter the pH: 3.35
Enter the sulphates: 0.8
Enter the alcohol: 10.5
[0]
Bad Quality Wine
```

Figure 46: exapmple prediction 2

5.2 Evaluation Scores

	precision	recall	f1-score	support
0	0.82	0.88	0.84	152
1	0.88	0.82	0.85	168
accuracy			0.85	320
macro avg	0.85	0.85	0.85	320
weighted avg	0.85	0.85	0.85	320

Figure 47: classification score and evaluation

The precision of the model for predicting bad quality wine is 82% while for good quality wine it's 88%. While average precision and recall for bad quality wine is 84% and as for good quality is 85%

In total accuracy came out to be 84% to 85%.

Chapter 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

Based on the bar plots, we conclude 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 change drastically. So, this feature is not essential as compared to others like alcohol and citric acid, therefore, we can drop this feature while performing feature selection. For classifying the wine quality, we have implemented Random Forest Classifier. We can achieve maximum accuracy of 85% using this model.

6.2 Future Scope

In future, by using a more optimized model and better computation techniques we will be able to increase the accuracy of these predictions. Further, we can improve the user interface for a more user friendly and easy to use environment. We can make this system more accessible for users who don’t have needed system configurations and software.

Chapter 7: REFERENCES

1. Devika Pawar, Aakanksha Mahajan: Wine Quality Prediction using Machine Learning Algorithms; **International Journal of Computer Applications Technology and Research** Volume 8–Issue 09, 385-388, 2019, ISSN:-2319–8656

2. Mohit Gupta, Vanmathi C: A Study and Analysis of Machine Learning Techniques in Predicting Wine Quality; **International Journal of Recent Technology and Engineering** (IJRTE) ISSN: 2277-3878, Volume-10 Issue-1, May 2021

3. Nikita Sharma: Quality Prediction of Red Wine based on Different Feature Sets Using Machine Learning Techniques; **International Journal of Science and Research (IJSR)** ISSN: 2319-7064 ResearchGate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

4. https://www.kaggle.com/vaibhav333/starter-red-wine-quality-90efe1e5-f
5. https://www.slideshare.net/MahimaSaini2/wine-quality-analysis-using-machine-learning
6. https://www.kaggle.com/swati2212/red-wine-prediction
7. https://www.youtube.com/channel/UCG04dVOTmbRYPY1wvshBVDQ
8. https://pandas.pydata.org/docs/index.html
9. https://stackoverflow.com/questions/37514686/how-to-plot-a-bar-graph-from-a-pandas-series
10. https://www.javatpoint.com/machine-learning-random-forest-algorithm
11. www.geeksforgeeks.com
12. www.google.com