MACHINE LEARNING

(WINE QUALITY PREDICTION)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE Engineering

By

RAHUL.k

221710306050

Under the Guidance
Of

GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM ❁ HYDERABAD ❁ BENGALURU

Department COMPUTER SCIENCE Engineering
GITAM school of technology
GITAM(Deemed to be University)
Hyderabad-50239
July 2020

DECLARATION

I submit this industrial training work entitled "**WINE QUALITY PREDICTION**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of **"Bachelor of Technology"** in "**COMPUTER SCIENCE Engineering**". I declare that it was carried out independently by me under the guidance of, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD Rahul.k

Date: 12-07-2020 221710306050

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:12-7-2020

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"WINE QUALITY PREDICTION"** is being submitted by Rahul.k (221710306050) in partial fulfillment of the requirement for the award of **Bachelor of Technology in COMPUTER SCIENCE ENGINEERING** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20

It is faithful record work carried out by the **COMPUTER SCIENCE ENGINEERING Department**, GITAM University Hyderabad Campus under my guidance and supervision.

DR.S.Phani kumar

Assistant professor

Professor and HOD

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. CH. Sanjay**, Principal, GITAM Hyderabad

I would like to thank respected **Dr. K. Manjunathachari**, Head of the Department of Electronics and Communication Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties_____who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

RAHUL.K
221710306050

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on Cereals data set My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategic approach for solution of the client, I have adapted the viewpoint of looking at ratings of the products and for further deep understanding of the problem, I have taken the stance of a consumer and reasoned out the various factors of choice of the products and they purchase , and my primary objective of this case study was to look up the factors which were dampening the sale of products and relate them to ratings of products and draft out an outcome report to client regarding the various accepts of a product manufacturing , marketing and sale point determination

Table of Contents:

CHAPTER 1:MACHINE LEARNING

CHAPTER 2:PYTHON

Chapter 3 :CASE STUDY

CHAPTER 4:WINE QUALITY PREDICTION:

CHAPTER 5. PLOTTING :

CHAPTER 6. CATEGORIZING:

CHAPTER 7. TRAINING THE MODEL:

CHAPTER 8. Modelling :

CHAPTER 9:

LIST OF FIGURES

DOCUMENTATION

CHAPTER 1:

MACHINE LEARNING

1.1 INTRODUCTION:

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

1.2 Importance of machine learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

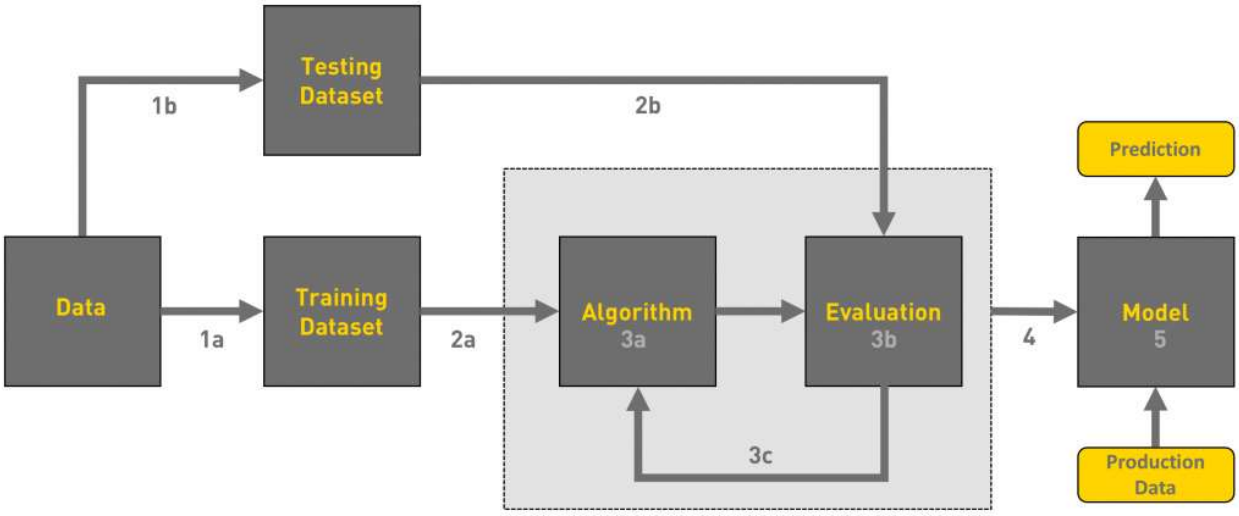


FIGURE 1.2.1: Process Flow

1.3 USES OF MACHINE LEARNING:

Machine learning (ML) equips computers to learn and interpret without being explicitly programmed to do so. Here, as the "computers", also referred as the "models", are exposed to sets of new data, they adapt independently and learn from earlier computations to interpret available data and identify hidden patterns. This involves data analysis and automation of analytical model-building using numerous ML algorithms. ML enables computers and computing machines to search for and identify hidden insights, without being programmed for where to look for, when exposed to new data sets.

Although this technology is not new, it is now gaining fresh momentum as there are numerous things to know about ML. The factors responsible for resurging interest in ML are powerful and affordable computational processing, continuously growing volumes of huge data sets, and affordable data storage options. Today, companies can make informed decisions by using ML algorithms to develop analytical models, which uncover connections, trends and patterns with minimal or no human intervention.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

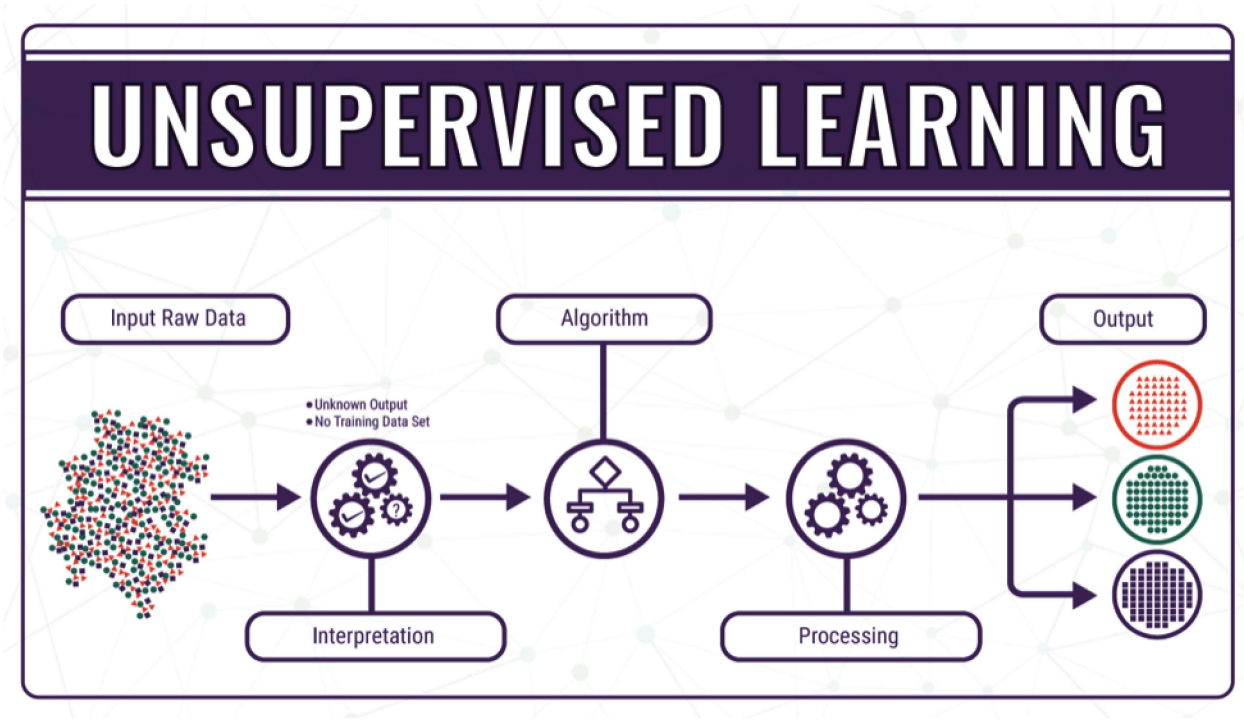


Figure 1.4.2 : unsupervised learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

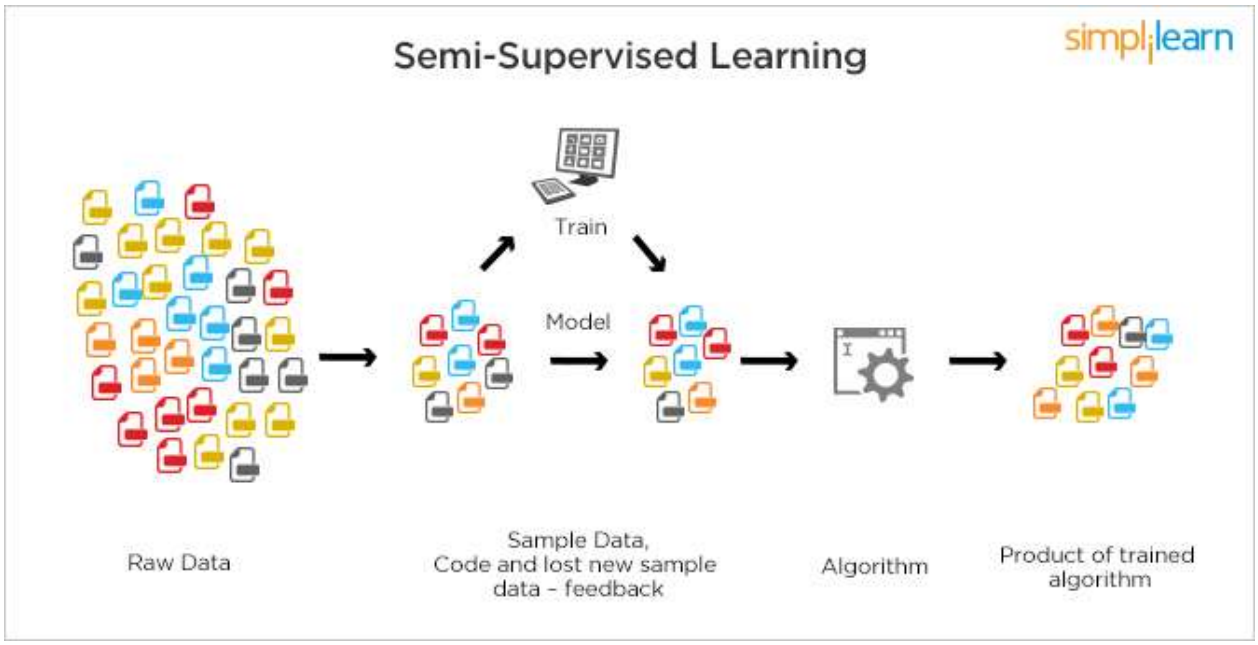


Figure 1.4.3 : semi-supervised learning algorithm

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovered previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special 5 types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning

CHAPTER 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYTHON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter direct to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.

- Easy-to-read: Python code is more clearly defined and visible to the eyes.

- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.

- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.

- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

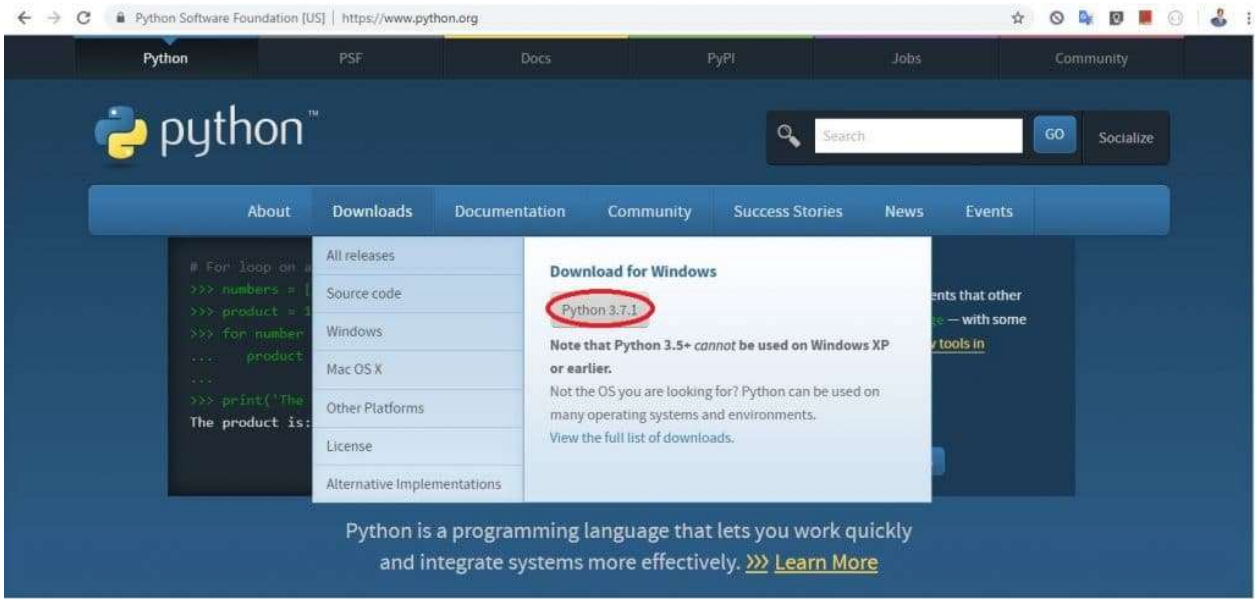


Figure 2.4.1 : installation

2.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.

- Conda is a package manager that quickly installs and manages packages.

● In WINDOWS:

- Step 1: Open Anaconda.com/downloads in a web browser.
- Step 2: Download python 3.4 version for (32-bits graphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

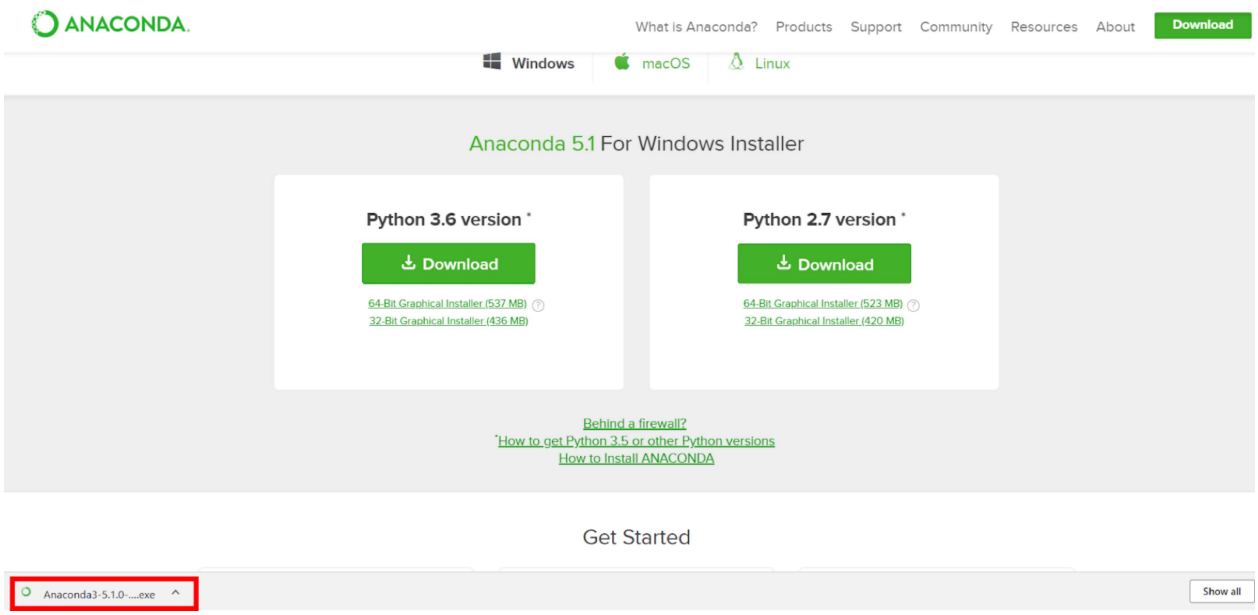


Figure 2.4.2 installation in window

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

- Variables are nothing but reserved memory locations to store values.

- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

- Python has five standard data types –
    - Numbers
    - Strings
    - Lists
    - Tuples
    - Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.

- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data types.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.

- Tuples can be thought of as read-only lists.

- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are a kind of hash table type. They work like associative arrays 12 or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOPs CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

- Data member: A class variable or instance variable that holds data associated with a class and its objects.

- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

- Defining a Class:

    - We define a class in a very similar way how we define a function.

    - Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is

2.7.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT :

The aim of the project is to find out what factors which affect the quality of the wine.

3.2 DATA SET:

The given data set consists of the following parameters:

A - **fixed_acidity**

B - **volatile_acidity**

C - **citric_acid**

D - **residual_sugar**

E - **chlorides**

F - **free_sulfur_dioxide**

G - **total_sulfur_dioxide**

H - **density**

I - **pH**

J -**sulphates**

K -**alcohol**

L -**quality**

3.3 OBJECTIVE OF THE CASE STUDY:

1. To experiment with different classification methods to see which yields the highest accuracy
2. To determine which features are the most indicative of a good quality wine.

CHAPTER 4

WINE QUALITY PREDICTION

4.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.1.1 GETTING THE DATASET:

We got the dataset from a website called kaggle.

4.1.2 IMPORTING THE LIBRARIES:

In this step we import all the libraries which are required to perform some machine learning algorithims

```
]: # importing the required libraries
   import matplotlib.pyplot as plt
   from scipy import stats
   import seaborn as sns
   import pandas as pd
   import numpy as np

   # importing the important models
   from sklearn import metrics
   from sklearn.ensemble import AdaBoostClassifier
   from sklearn.preprocessing import StandardScaler
   from sklearn.preprocessing import LabelEncoder
   from sklearn.neighbors import KNeighborsClassifier
   from sklearn.ensemble import GradientBoostingClassifier
   from sklearn.metrics import confusion_matrix
   from sklearn.metrics import accuracy_score
   from sklearn.metrics import classification_report
   from sklearn.ensemble import RandomForestClassifier
   from sklearn.tree import DecisionTreeClassifier
   from sklearn.linear_model import SGDClassifier
   from sklearn.model_selection import RandomizedSearchCV,cross_val_score
```

Figure 4.1.2 :libraries

4.1.3 version of the libraries which we are using :

We are checking the versions of the libraries which we are using

```
#checking the version of the libraries

print(pd.__version__)
print(sns.__version__)
print(np.__version__)
```

```
1.0.1
0.10.0
1.18.1
```

Figure 4.1.3 :version of the libraries

4.2 READING THE DATA SET :

In this step we read the dataset and gather some basic information

During this step we read the data

- and gather some basic information from the data set

```
# reading the data
wine=pd.read_csv("wine.csv")
wine
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	c
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	w
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	w
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	w
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	w
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	w

6497 rows × 13 columns

Figure 4.2.1:reading the dataset

4.3 CATEGORICAL DATA :

- Machine Learning models are based on equations, we need to replace the text by numbers. So that we can include the numbers in the equations.

- Categorical Variables are of two types:
  Nominal and Ordinal

- Nominal: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour

- Ordinal: The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium

- Categorical data can be handled by using dummy variables, which are also called as indicator variables.

- Handling categorical data using dummies: In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0’s and 1’s. Once these dummies have been created we have to concat this dummy set to our dataframe or we can add that dummy set to the dataframe.

4.4 CONVERTING categorical columns to variable columns :

This step is done to convert the categorical columns into variable columns as ,it will be much easier for the coming steps or for the algorithms which we gonna use

```
# here we are converting a categorical coloumn into variable coloumn for performing some actions in the coming steps
number=LabelEncoder()
wine["color"]=number.fit_transform(wine['color'].astype('string'))

wine
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	c
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	

6497 rows × 13 columns

Figure 4.4.1: converting the column

4.5 CHECKING FOR MISSING VALUES :

We check for missing values as ,the missing values can affect the dataset by giving us the wrong prediction ,when we are predicting the model

```
# checking for any missing values
wine.isnull()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	c
0	False	False	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...	...	
6492	False	False	False	False	False	False	False	False	False	False	False	False	F
6493	False	False	False	False	False	False	False	False	False	False	False	False	F
6494	False	False	False	False	False	False	False	False	False	False	False	False	F
6495	False	False	False	False	False	False	False	False	False	False	False	False	F
6496	False	False	False	False	False	False	False	False	False	False	False	False	F

6497 rows × 13 columns

Figure 4.5.1 : missing values

4.6 MINIMIZING THE DATA SET :

As the data which we have taken from the client or the data which we have ,can consists of various numbers of rows and columns which makes very hard to perform the operations as it time taking and space taking process

So for this we minimize the data.set by using the function (data.head()) in this way it will have the same properties of the whole data.set and it will easier to compute

As the dataset contains n no.of rows we can minimize the dataset and perform machine learning operations on them .cause it possess the same set of attributes of the overall dataset, and minimizing it can save us with compilation time

```
# limiting the dataset
wine.head()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0

Figure 4.6.1: minimize the dataset

4.7 UNIQUE VALUES OF QUALITY:

As wine quality prediction is directly related to the quality of the wine ,we are checking for the unique values in the columns “quality”

checking unique values of quality attribute

```
# checking for unique values
wine["quality"].unique()
```

```
array([5, 6, 7, 4, 8, 3, 9], dtype=int64)
```

Figure 4.7.1 :unique values

4.8 NO.OF UNIQUE VALUES.:

finding out the number values ,which fall into the unique values of the quality attribute

```
]: # no.of values which fall into the quality attribute
   wine.quality.value_counts().sort_index()
```

```
]: 3      30
   4     216
   5    2138
   6    2836
   7    1079
   8     193
   9       5
   Name: quality, dtype: int64
```

Figure 4.8.1: unique values count

CHAPTER 5

PLOTTING

5.1 Plotting the values of quality:

We are trying to represent the unique values of the wine quality ,just so that we can understand the visual representation of the data

we are just plotting the values of quality

```
# plotting the values
sns.countplot(wine['quality'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234fd15c188>
```

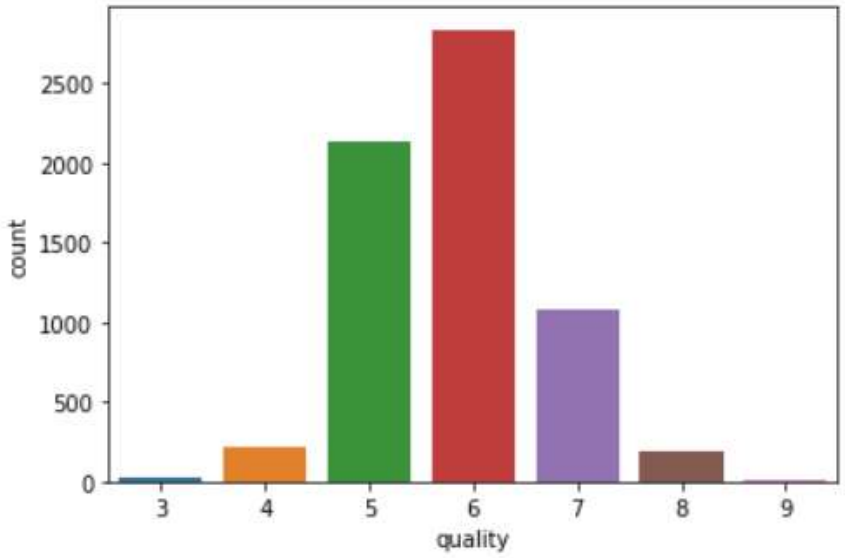


Figure 5.1.1: bar graph

5.2. HEATMAP :

A heatmap is a graphical representation of data that uses a system of color-coding to represent different values. Heatmaps are used in various forms of analytics but are most commonly used to show user behaviour on specific webpages or web page templates. Heatmaps can be used to show where users have clicked on a page, how far they have scrolled down a page, or used to display the results of eye-tracking tests.

We find the heatmap of the given data to find out the missing values and as a whole we can find out the correlation between all the values in one single representation

taking the heatmap of the data to get a better picture of dataset and to find the comparison of different values to different attributes ¶

```
]: #heatmap for the whole data set
   correlation=wine.corr()
   plt.figure(figsize=(14,8))
   sns.heatmap(correlation,annot=True,vmin=-2,cmap="plasma")
```

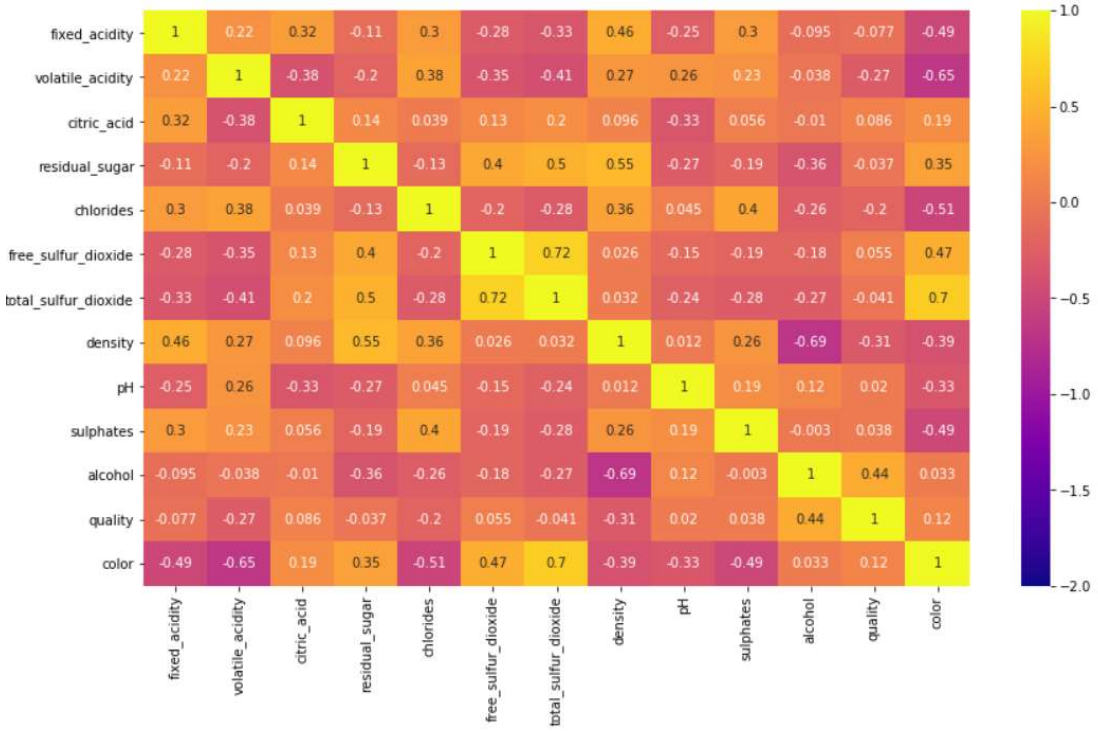


Figure 5.2.1:heatmap of the data

From this we can see that the quality of the wine is highly dependent on the alcohol percentage ,and the rest of the values gradually

5.3 COUNTPLOT :

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot(), so you can compare counts across nested variables.

Input data can be passed in a variety of formats, including:

- Vectors of data represented as lists, numpy arrays, or pandas Series objects passed directly to the x, y, and/or hue parameters.
- A “long-form” DataFrame, in which case the x, y, and hue variables will determine how the data are plotted.
- A “wide-form” DataFrame, such that each numeric column will be plotted.
- An array or list of vectors.

By this graphical representation we can see that the quality of the wine is divided by the color and which color of wine is more widely used

seeing the quality and the no.of users for each wine

```
# no.of users for each rating
sns.countplot(x="color", hue="quality",data=wine)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234fefcb748>
```

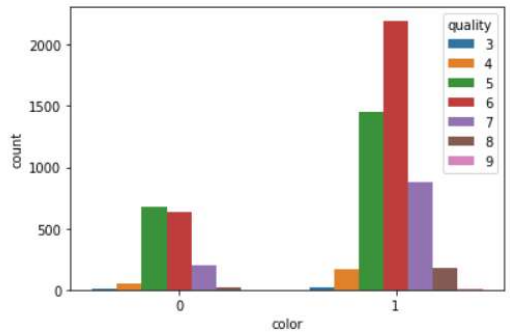


Figure 5.3.1: countplot

5.4 SCATTER PLOT:

A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

We are using scatter plot to find the distributive of the data with graphical representation right above it ,so that the user has clear understanding

plotting this to show the scatter plot with the rug plot as 2 dimensions

```
# scatter plot
sns.jointplot(x="alcohol", y="quality", data=wine);
```

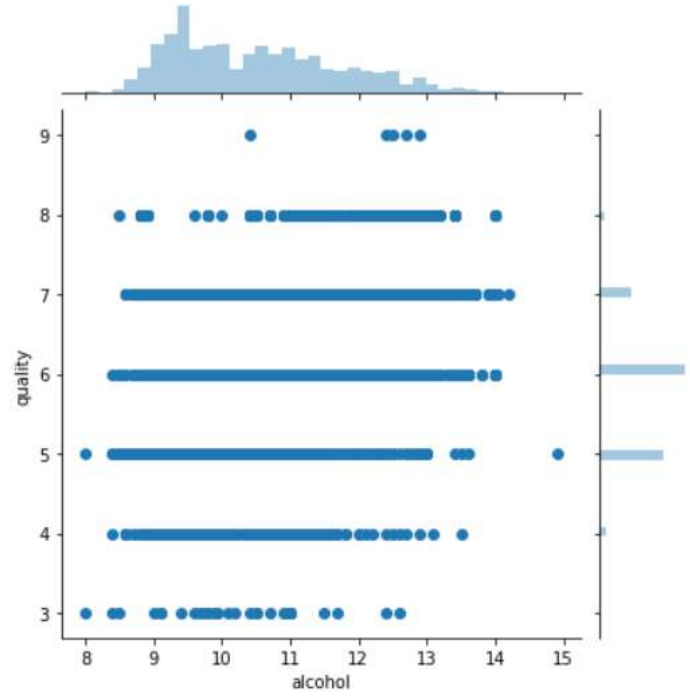


Figure 5.4.1:scatter plot

5.5 DENSITY ESTIMATION:

By this we can see that the density in which the data has reached and it useful to proceed with further explanations

this is used to show the concentration of the values in the dataset with respect to quality and alcohol levels

```
# denisty estimation
sns.jointplot(x="alcohol", y="quality", data=wine, kind="kde");
```

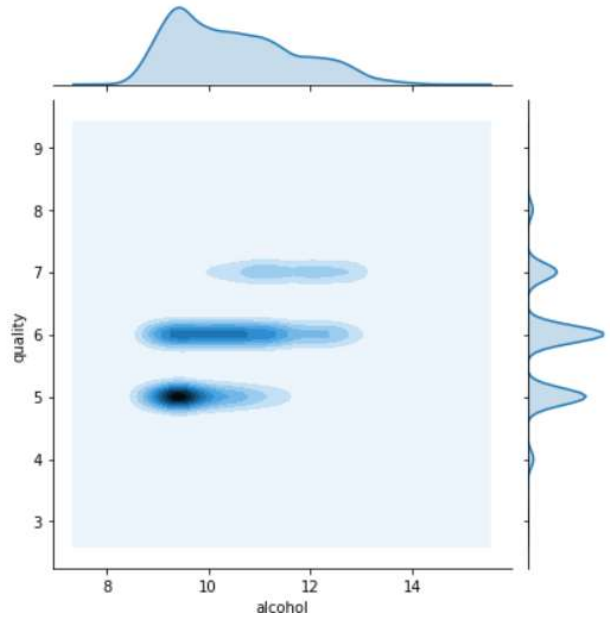


Figure 5.5.1:density estimation

5.6 HISTOGRAM :

Matplotlib can be used to create histograms. A histogram shows the frequency on the vertical axis and the horizontal axis is another dimension. Usually it has bins, where every bin has a minimum and maximum value. Each bin also has a frequency between x and infinite.

a histogram for all the datasets and by the diagram we can find out the data is not organised and it difficult to predict any proper information

```
: # histogram for the complete data set
  wine.hist(figsize=(10,10))
```

```
: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023482DCFC88>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x0000023482F51548>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x0000023482F85D88>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x0000023482FC1EC8>],
         [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023482FFE048>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x00000234830360C8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x00000234830451C8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x000002348307E308>],
         [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023483083EC8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x00000234830C40C8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x00000234834FDE48>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x000002348352F6C8>],
         [<matplotlib.axes._subplots.AxesSubplot object at 0x00000234835677C8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x000002348359F908>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x00000234835DAAC8>,
          <matplotlib.axes._subplots.AxesSubplot object at 0x0000023483614C88>]],
        dtype=object)
```

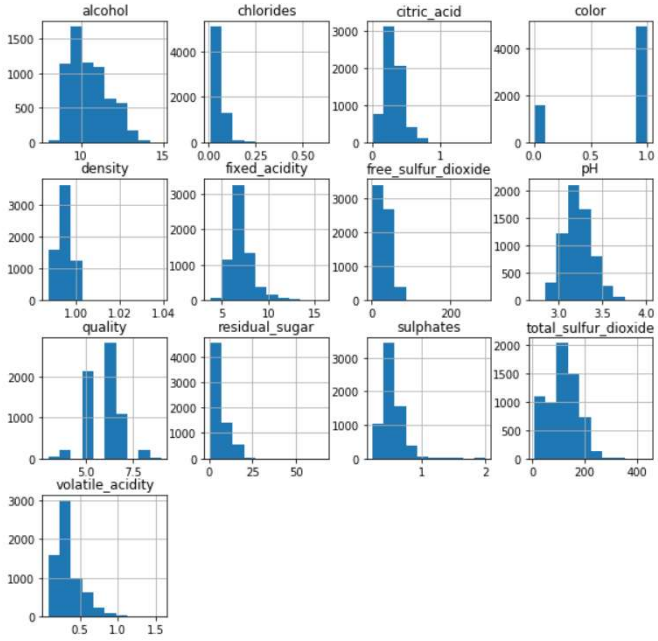


Figure 5.6.1:histogram

From the histogram we can depict that the values are not uniform as they are either at max values or to the extreme corners of the graph. And we can say that the data is not organised and difficult to predict any useful information.

CHAPTER 6

CATEGORIZING

6.1 Categorizing The DATA(QUALITY) :

We are categorizing the data for to make it much simpler for the user model,we are categorizing the data into 3 parts:

1. GOOD QUALITY: this in the range between (5-6)
2. INFERIOR QUALITY: this is in the range of (0-4)
3. SUPERIOR QUALITY: this in the range of (7-9)

in this we are grouping the quality attribute into 3 parts:

- if greater then 7 :superior
- if between 4-7:good
- if less then 4:inferior

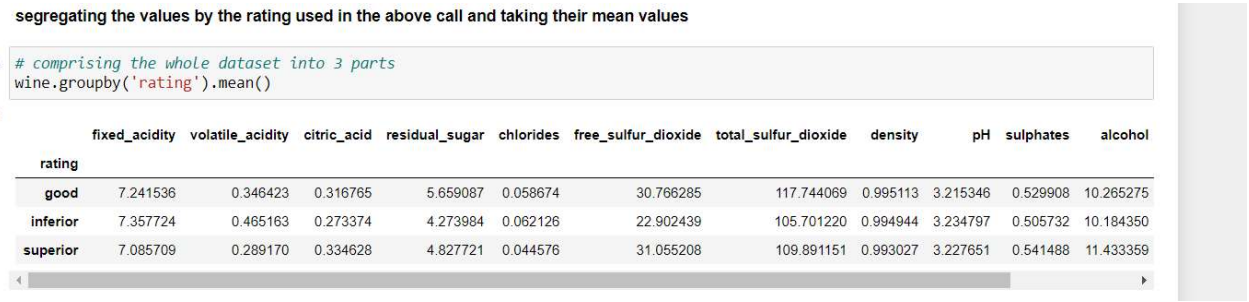
```
# dividing the values into 3 parts
con=[
    wine["quality"]>=7, wine["quality"]<=4
]
rating=['superior','inferior']
wine['rating']=np.select(con,rating,default='good')
wine.rating.value_counts()
```

```
good        4974
superior    1277
inferior     246
Name: rating, dtype: int64
```

Figure 6.1.1:categorizing the data

6.2. SEGREGATING THE VALUES :

In this we segregate the values based on the rating of the quality ,in that we can see how many values belong to the rating which we have divided in the above step



segregating the values by the rating used in the above call and taking their mean values

```
# comprising the whole dataset into 3 parts
wine.groupby('rating').mean()
```

rating	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol
good	7.241536	0.346423	0.316765	5.659087	0.058674	30.766285	117.744069	0.995113	3.215346	0.529908	10.265275
inferior	7.357724	0.465163	0.273374	4.273984	0.062126	22.902439	105.701220	0.994944	3.234797	0.505732	10.184350
superior	7.085709	0.289170	0.334628	4.827721	0.044576	31.055208	109.891151	0.993027	3.227651	0.541488	11.433359

Figure 6.2.1:segregating

6.3. CORRELATION :

It is important to discover and quantify the degree to which variables in your dataset are dependent upon each other. This knowledge can help you better prepare your data to meet the expectations of machine learning algorithms, such as linear regression, whose performance will degrade with the presence of these interdependencies.

From the below correlation we can depict 3 observations:

1. **Rating good**: we can see that most of the values are between 5-6 and the line is gradually increasing from 5 to 6 ,which shows a positive correlation

2. **Rating superior**:here we can see that the values range from 7-9 and we can also see that most of the values are close to predicted correlation

3. **Rating inferior** : the values range from 4 or less ,and the data is not close to related model which is not good correlation

finding the correlation for between the quality and here we can see that the rating of the values as

- good(5-6)
- superior(7-9)
- inferior(3-4)

```
4]: # correlation of the data
    sns.lmplot(y="alcohol" ,x="quality",col="rating",data=wine)
```

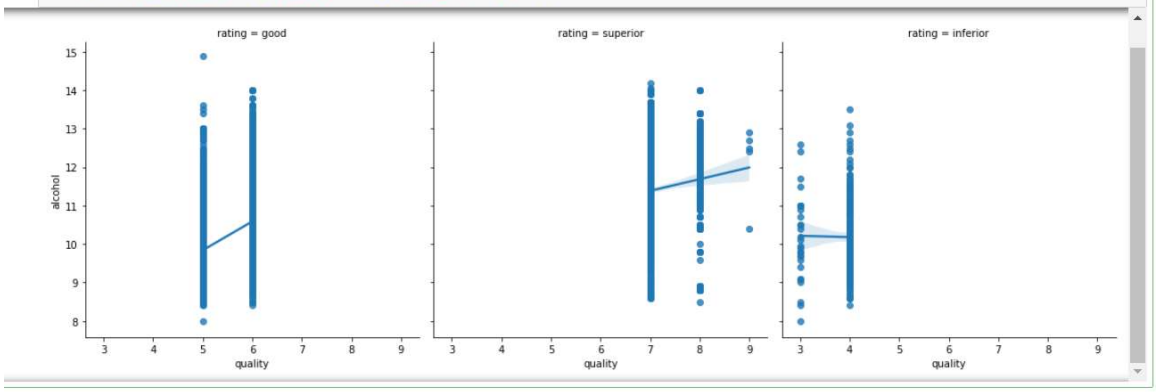


Figure 6.3.1:correlation

CHAPTER 7

TRAINING THE MODEL:

7.1 Train and Test split:

- **Splitting the data :** after the preprocessing is done then the data is split into train and test sets

- In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.

- training set - a subset to train a model.(Model learns patterns between Input and Output)

- test set - a subset to test the trained model.(To test whether the model has correctly learnt)

- The amount or percentage of Splitting can be taken as specified (i.e. train data = 75% , test data =25% or train data = 80% , test data= 20%)

- First we need to identify the input and output variables and we need to separate the input set and output set

- In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method

- This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables)

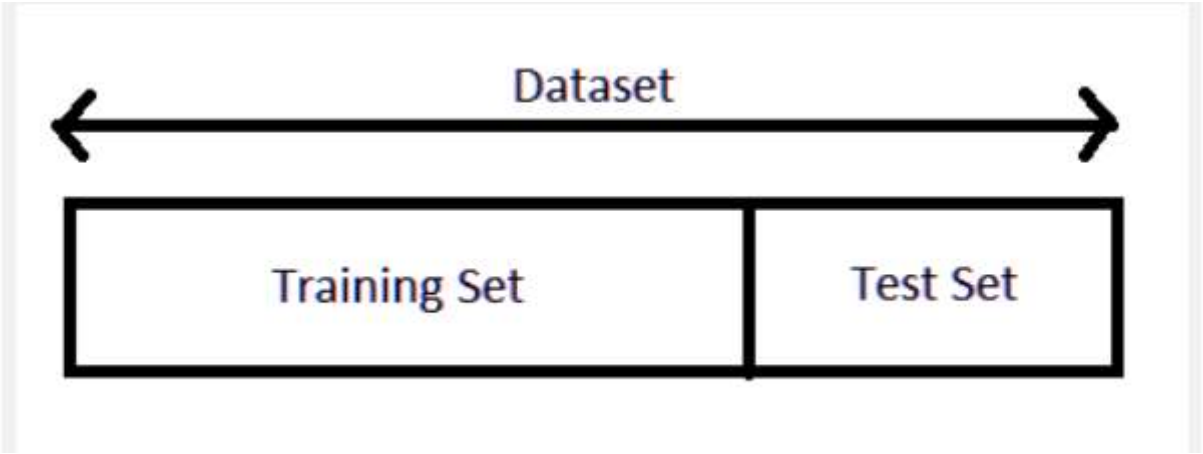


Figure 7.1.1: train and test

We are taking one set without the quality value and the other set which includes the quality set ,this splitting of the dataset gives us the optimal solution if we apply the models

here we are dividing the data into training and testing to perform machine learning algorithms

- we divide the data into 80% into training and the other into testing
- this helps us in getting the optimal solution

```
# dividing into training and test
from sklearn.model_selection import train_test_split
x=wine[wine.columns[:-1]]
y=wine["rating"]

x_train , x_test , y_train , y_test =train_test_split(x,y ,test_size=.2)
```

```
# loading the data set
sc=StandardScaler()
x_train_std = sc.fit_transform(x_train)
x_test_std = sc.transform(x_test)
```

Figure 7.1.2:splitting the data

7.2 Describing the train and test split data:

After this we are describing the data which has been divided into train and test ,by this we can actually see all the values between the train and test dataset

we are describing the values in y_test and y_train

```
: # describing the values in test and train
for wine in [y_train,y_test]:
    print(wine.describe())
```

```
count     5197
unique       3
top       good
freq      3990
Name: rating, dtype: object
count     1300
unique       3
top       good
freq       984
Name: rating, dtype: object
```

Figure 7.2.1:description

CHAPTER 8

Modelling

For this project, I wanted to compare five different machine learning models: ,K-Neighbors classifier, decision trees, random forests, AdaBoost, Gradient Boost, and Gradient Boost. For the purpose of this project, I wanted to compare these models by their accuracy.

8.1 MODEL 1 - K-Nearest Neighbors:

The K-nearest neighbors (KNN) algorithm is a type of supervised machine learning algorithm. KNN is extremely easy to implement in its most basic form, and yet performs quite complex classification tasks. It is a lazy learning algorithm since it doesn't have a specialized training phase. Rather, it uses all of the data for training while classifying a new data point or instance. KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data. This is an extremely useful feature since most of the real world data doesn't really follow any theoretical assumption e.g. linear-separability, uniform distribution, etc.

Now if we apply the model on the dataset ,we can see the following observations

kneighbour classifier : here we can see that the accuracy is 96%

```
]: # keighbbour classifier
   n5=KNeighborsClassifier(n_neighbors=5)
   n5.fit(x_train_std,y_train)
   predict_n5=n5.predict(x_test_std)
   print(classification_report(y_test,predict_n5))
   cross_val=cross_val_score(estimator=n5 , X=x_train_std,y=y_train,cv=4)
   print(cross_val.mean())
```

```
              precision    recall  f1-score   support

        good       0.96      1.00      0.98       984
    inferior       1.00      0.51      0.67        61
    superior       0.98      0.94      0.96       255

    accuracy                           0.96      1300
   macro avg       0.98      0.81      0.87      1300
weighted avg       0.96      0.96      0.96      1300

0.9622857819624563
```

Figure 8.1.1:kneighbour classifier.

8.1.1 Confusion matrix :

Confusion matrix is a matrix which is often used to describe the performance of a classification model a set of test data of which the true values are known .

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

```
# confusion matrix to know the true values
confusion_matrix(y_test,predict_n5)
```

```
array([[980,   0,   4],
       [ 30,  31,   0],
       [ 16,   0, 239]], dtype=int64)
```

Figure 8.1.2 confusion matrix

8.1.2 HEATMAP FOR CONFUSION MATRIX :

Here we are just plotting the confusion matrix on a heatmap for just a visual representation

heatmap for the confusion matrix

```
#graphical representation of confusion matrix
sns.heatmap(confusion_matrix(y_test,predict_n5),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f06ad122c8>
```

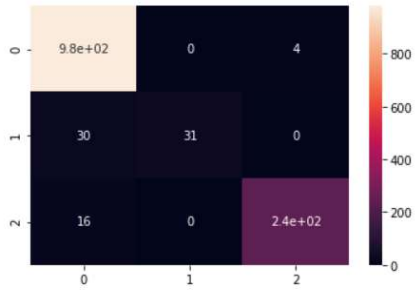


Figure 8.1.3 :heatmap for confusion matrix

8.2 Model 2: Random Forest

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

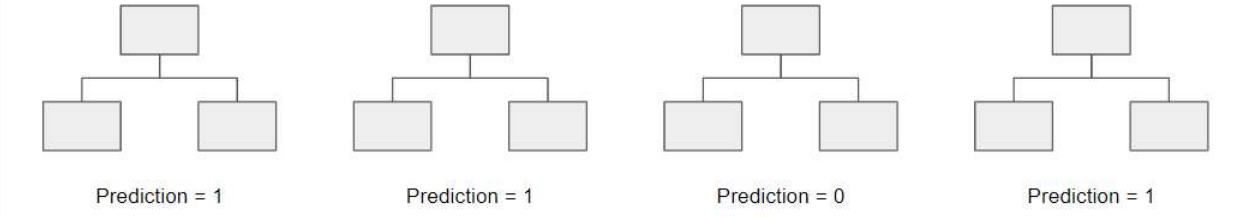


Figure 8.2.1:random forest

For example, if we created one decision tree, the third one, it would predict 0. But if we relied on the mode of all 4 decision trees, the predicted value would be 1. This is the power of random forests.

Random forests also offer a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

random forest classifier : this has got accuracy of 100% ,but we check other classifiers if they have got better accuracy and less recall value

```
#random forest classifier
rf=RandomForestClassifier()
rf.fit(x_train_std,y_train)
predict_rf=rf.predict(x_test_std)
print(classification_report(y_test,predict_rf))
cross_val=cross_val_score(estimator=rf,X=x_train_std , y=y_train,cv=4)
print(cross_val)
```

```
              precision    recall  f1-score   support

        good       1.00      1.00      1.00       984
    inferior       1.00      1.00      1.00        61
    superior       1.00      1.00      1.00       255

    accuracy                           1.00      1300
   macro avg       1.00      1.00      1.00      1300
weighted avg       1.00      1.00      1.00      1300

[1. 1. 1. 1.]
```

Figure 8.2.2: random forest classification

Here we can see that ,the accuracy of the model is 100% but we the recall value is less so therefore we try and see if any other model is better than the random forest classifier

8.2.1 Confusion matrix :

Confusion matrix is a matrix which is often used to describe the performance of a classification model on a set of test data of which the true values are known .

```
# confusion matrix to know the true values
confusion_matrix(y_test,predict_rf)
```

```
array([[984,   0,   0],
       [  0,  61,   0],
       [  0,   0, 255]], dtype=int64)
```

Figure 8.2.3:confusion matrix

8.2. HEATMAP FOR CONFUSION MATRIX :

Here we are just plotting the confusion matrix on a heatmap for just a visual representation

```
#graphical representation of confusion matrix
sns.heatmap(confusion_matrix(y_test,predict_rf),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f06cb7e7c8>
```

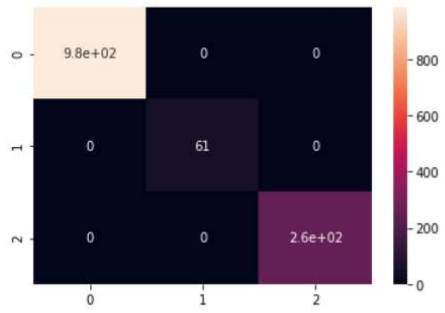


Figure 8.2.4 heatmap for confusion matrix

8.3 Model 3: Decision Tree:

Decision trees are a popular model, used in operations research, strategic planning, and machine learning. Each square above is called a node, and the more nodes you have, the more accurate your decision tree will be (generally). The last nodes of the decision tree, where a decision is made, are called the leaves of the tree. Decision trees are intuitive and easy to build but fall short when it comes to accuracy.

Decision tree classifiers provide a readable classification model that is potentially accurate in many different application contexts, including energy-based applications. The decision tree classifier (Pang-Ning et al., 2006) creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute. Each leaf represents class labels associated with the instance. Instances in the training set are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path. Starting from the root node of the tree, each node splits the instance space into two or more sub-spaces according to an attribute test condition

decision tree classifier: this classifier has got 100% accuracy and this is the best model compared to the other models

```
#decision tree classifier
dt=DecisionTreeClassifier()
dt.fit(x_train_std,y_train)
predict_dt=dt.predict(x_test_std)
print(classification_report(y_test,predict_dt))
cross_val=cross_val_score(estimator=dt,X=x_train_std , y=y_train,cv=4)
print(cross_val)
```

```
              precision    recall  f1-score   support

        good       1.00      1.00      1.00       984
    inferior       1.00      1.00      1.00        61
    superior       1.00      1.00      1.00       255

    accuracy                           1.00      1300
   macro avg       1.00      1.00      1.00      1300
weighted avg       1.00      1.00      1.00      1300

[1. 1. 1. 1.]
```

Figure 8.3.1 :decision tree classifier

In this we got an accuracy of 100% ,but we try to check other models just to make sure

8.3.1 Confusion matrix :

Confusion matrix is a matrix which is often used to describe the performance of a classification model on a set of test data of which the true values are known .

```
# confusion matrix to know the true values
confusion_matrix(y_test,predict_dt)
```

```
array([[984,   0,   0],
       [  0,  61,   0],
       [  0,   0, 255]], dtype=int64)
```

Figure 8.3.2:confusion matrix

8.3.2HEATMAP FOR CONFUSION MATRIX :

Here we are just plotting the confusion matrix on a heatmap for just a visual representation

```
#graphical representation of confusion matrix
sns.heatmap(confusion_matrix(y_test,predict_dt),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f06d0063c8>
```

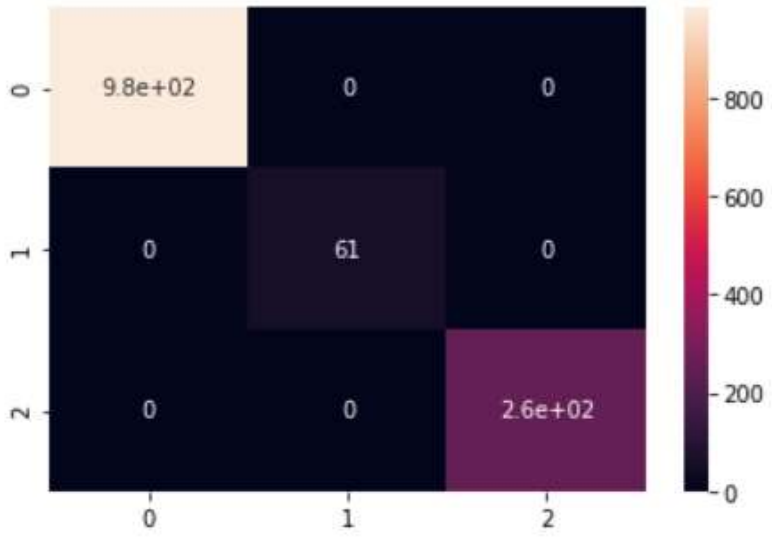


Figure 8.3.3:heatmap for confusion matrix

8.4 Model 4: AdaBoost:

The next three models are boosting algorithms that take weak learners and turn them into strong ones. I don't want to get sidetracked and explain the differences between the three because it's quite complicated and intricate. That being said, I'll leave some resources where you can learn about AdaBoost, Gradient Boosting, and XGBoosting.

AdaBoost classifier

```
#AdaBoost classifier
ad=AdaBoostClassifier()
ad.fit(x_train_std,y_train)
predict_ad=ad.predict(x_test_std)
print(classification_report(y_test,predict_ad))
cross_val=cross_val_score(estimator=ad,X=x_train_std , y=y_train,cv=2)
print(cross_val)
```

```
              precision    recall  f1-score   support

        good       1.00      1.00      1.00       984
    inferior       1.00      1.00      1.00        61
    superior       1.00      1.00      1.00       255

    accuracy                           1.00      1300
   macro avg       1.00      1.00      1.00      1300
weighted avg       1.00      1.00      1.00      1300

[1. 1.]
```

Figure 8.4.1:Adaboost classifier

In this we got an accuracy of 100% ,which is less compared to the other models

8.4.1 Confusion matrix :

Confusion matrix is a matrix which is often used to describe the performance of a classification model on a set of test data of which the true values are known .

```
# confusion matrix to know the true values
confusion_matrix(y_test,predict_ad)
```

```
array([[984,   0,   0],
       [  0,  61,   0],
       [  0,   0, 255]], dtype=int64)
```

Figure 8.4.2:confusion matrix

8.4.2 HEATMAP FOR CONFUSION MATRIX :

Here we are just plotting the confusion matrix on a heatmap for just a visual representation

```
#graphical representation of confusion matrix
sns.heatmap(confusion_matrix(y_test,predict_rf),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f06cb7e7c8>
```

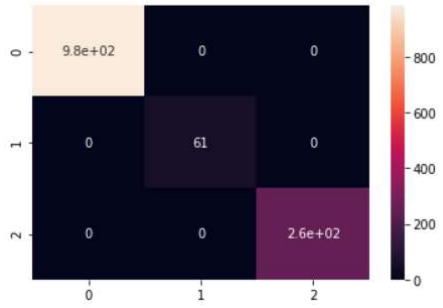


Figure 8.4.3:heatmap for confusion matrix

8.5 MODEL 5: GRADIENT BOOSTING

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

GradientBoosting classifier: this classifier has got 100% accuracy and this is the best model compared to the other model

```
: #GradientBoosting classifier
gd=GradientBoostingClassifier()
gd.fit(x_train_std,y_train)
predict_gd=gd.predict(x_test_std)
print(classification_report(y_test,predict_gd))
cross_val=cross_val_score(estimator=gd,X=x_train_std , y=y_train,cv=2)
print(cross_val)
```

```
              precision    recall  f1-score   support

        good       1.00      1.00      1.00       984
    inferior       1.00      1.00      1.00        61
    superior       1.00      1.00      1.00       255

    accuracy                           1.00      1300
   macro avg       1.00      1.00      1.00      1300
weighted avg       1.00      1.00      1.00      1300

[1. 1.]
```

Figure 8.5.1:gradientboosting classifier

In this model as well we got 100% accuracy

8.5.1 Confusion matrix :

Confusion matrix is a matrix which is often used to describe the performance of a classification model on a set of test data of which the true values are known .

```
# confusion matrix to know the true values
confusion_matrix(y_test,predict_dt)
```

```
array([[984,   0,   0],
       [  0,  61,   0],
       [  0,   0, 255]], dtype=int64)
```

Figure 8.5.2 :confusion matrix

8.5.2 HEATMAP FOR CONFUSION MATRIX :

Here we are just plotting the confusion matrix on a heatmap for just a visual representation

```
#graphical representation of confusion matrix
sns.heatmap(confusion_matrix(y_test,predict_rf),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f06cb7e7c8>
```

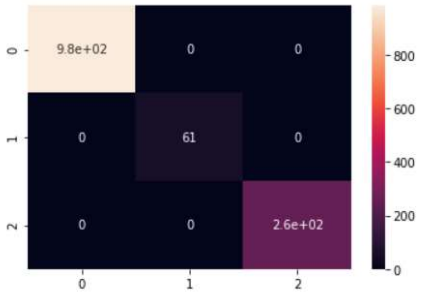


Figure 8.5.3:heatmap for confusion matrix

CHAPTER 9

Conclusion

From the above various models ,we can see that we have got different accuracy values for different models but they are 2 models which have resulted in 100% accuracy

1. Decision tree classifier
2. Gradient boosting classifier

By looking into the details, we can see that good quality wines have higher levels of alcohol on average, have a lower volatile acidity on average, higher levels of sulphates on average, and higher levels of residual sugar on average.

REFERENCES:

1. **https://www.kaggle.com/search?q=wine+quality+dataset**
2. **https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434**
3. Github-**https://github.com/rahul624/wine-quality-prediciton**