MACHINE LEARNING

(Wine Quality Prediction)

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

Bachelor of Technology
In
Computer Science and Engineering
By

M.S.Shreyas Reddy

221710313037

Under the Guidance of
Assistant Professor

GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)
VISAKHAPATNAM HYDERABAD BENGALURU

Department of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled "**Wine Quality Prediction**" to GITAM (Deemed to Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science and 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 M.S.Shreyas Reddy

Date: -12.07.20 221710313037

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated: **12.07.2020**

CERTIFICATE

This is to certify that the Industrial Training Report entitled "**Wine Quality Prediction**" is being submitted by M.S.Shreyas Reddy(221710313037) in partial fulfilment of the requirement for the award of Bachelor of Technology in **Computer Science and Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2020-21.

It is faithful record work carried out by her at the **Computer Science and Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Assistant Professor	Professor and HOD
Department of CSE	Department of CSE

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 respected **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 of what we study for.

I would like to thank the respected faculties **Mr. M. Venkateswarlu** 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.

M.S.Shreyas Reddy

221710313037

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 Wine data set.

Wine classification is a difficult task since taste is the least understood of the human senses. A good wine quality prediction can be very useful in the certification phase, since currently the sensory analysis is performed by human tasters, being clearly a subjective approach. An automatic predictive system can be integrated into a decision support system, helping the speed and quality of the performance. Furthermore, a feature selection process can help to analyse the impact of the analytical tests. If it is concluded that several input variables are highly relevant to predict the wine quality, since in the production process some variables can be controlled, this information can be used to improve the wine quality. Classification models used here are

1) Random Forest

2) Decision Tree classifier

The higher the value the better the quality. In this project we will treat each class of the wine separately and their aim is to be able and find decision boundaries that work well for new unseen data. These are the classifiers.

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER-1

MACHINE LEARNING

1.1 INTRODUCTION:

Artificial intelligence (AI) traditionally refers to an artificial creation of human-like intelligence that can learn, reason, plan, perceive, or process natural language.

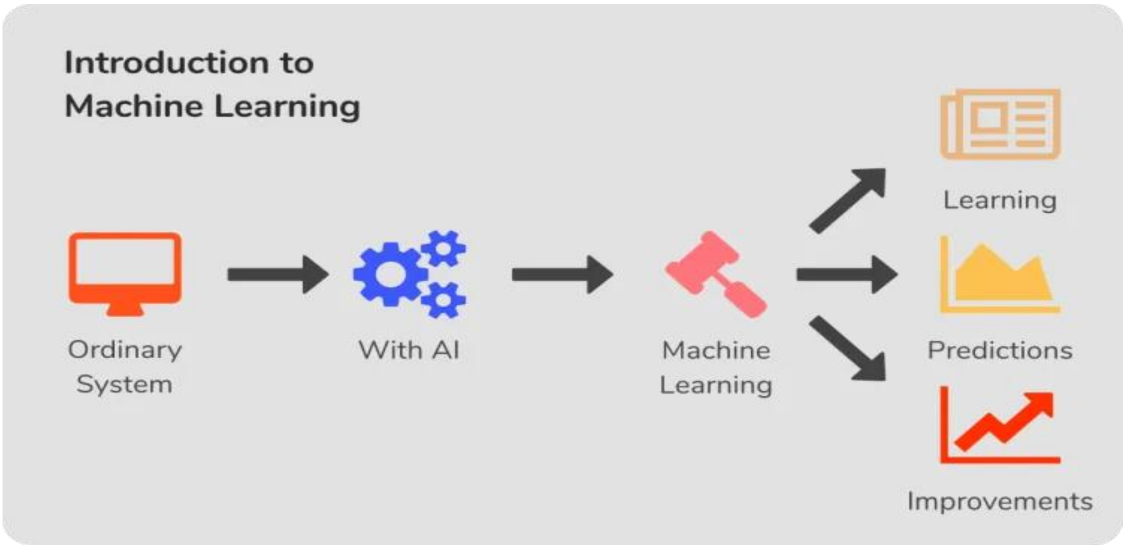
Artificial intelligence is further defined as “narrow AI” or “general AI”. Narrow AI, which we interact with today, is designed to perform specific tasks within a domain (e.g. language translation). General AI is hypothetical and not domain specific, but can learn and perform tasks anywhere. This is outside the scope of this paper. This paper focuses on advances in narrow AI, particularly on the development of new algorithms and models in a field of computer science referred to as *machine learning*.

1.2 IMPORTANCE OF MACHINE LEARNING:

Algorithms are a sequence of instructions used to solve a problem. Algorithms, developed by programmers to instruct computers in new tasks, are the building blocks of the advanced digital world we see today. Computer algorithms organize enormous amounts of data into information and services, based on certain instructions and rules. It’s an important concept to understand, because **in machine learning, learning algorithms – not computer programmers – create the rules**.

Instead of programming the computer every step of the way, this approach gives the computer instructions that allow it to learn from data without new step-by-step instructions by the programmer.

Figure 1: Introduction



The process flow depicted here represents how machine learning works

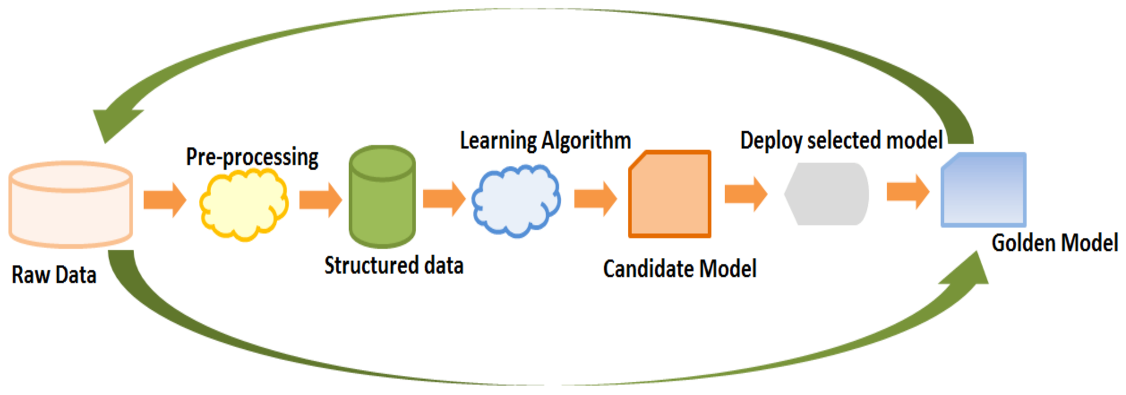


Figure 2: The Process Flow

1.3 USES OF MACHINE LEARNING: -

There are limitless applications of machine learning and there are a lot of machine learning algorithms are available to learn. They are available in every form from simple to highly complex. Top 10 Uses of machine learning are as follows:

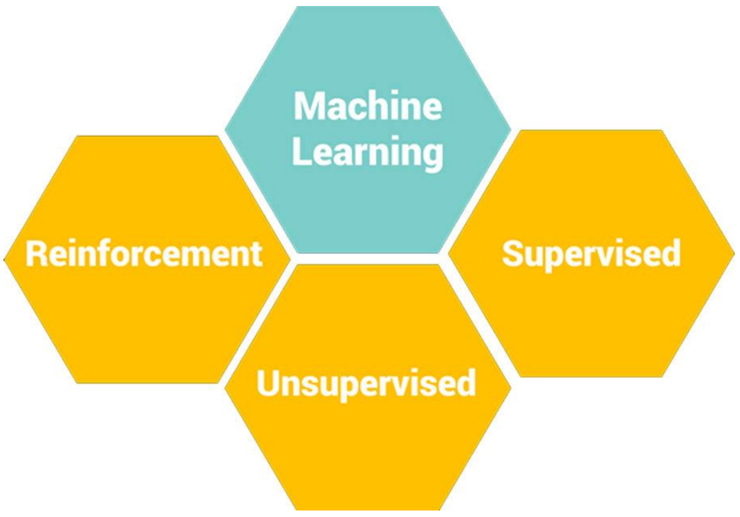
Image Recognition

The image recognition is one of the most common uses of machine learning applications. It can also be referred to as a digital image and for these images, the measurement describes the output of every pixel in an image. The face recognition is also one of the great features that have been developed by machine learning only. It helps to recognize the face and send the notifications related to that to people.

Voice Recognition

Machine learning (ML) also helps in developing the application for voice recognition. It also referred to as virtual personal assistants (VPA). It will help you to find the information when asked over the voice. After your question, that assistant will look out for the data or the information that has been asked by you and collect the required information to provide you with the best answer. There are many devices available in today's world of Machine learning for voice recognition that is Amazon echo and googles home is the smart speakers. There is one mobile app called Google allo and smartphones are Samsung S8 and Bixby.

1.4 TYPES OF LEARNING ALGORITHMS:



As with any method, there are different ways to train machine learning algorithms, each with their own advantages and disadvantages. To understand the pros and cons of each type of machine learning, we must first look at what kind of data they ingest. In ML, there are two kinds of data — labelled data and unlabelled data.

There are also some types of machine learning algorithms that are used in very specific use-cases, but three main methods are used today.

1.4.1 Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labelled data. Even though the data needs to be labelled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

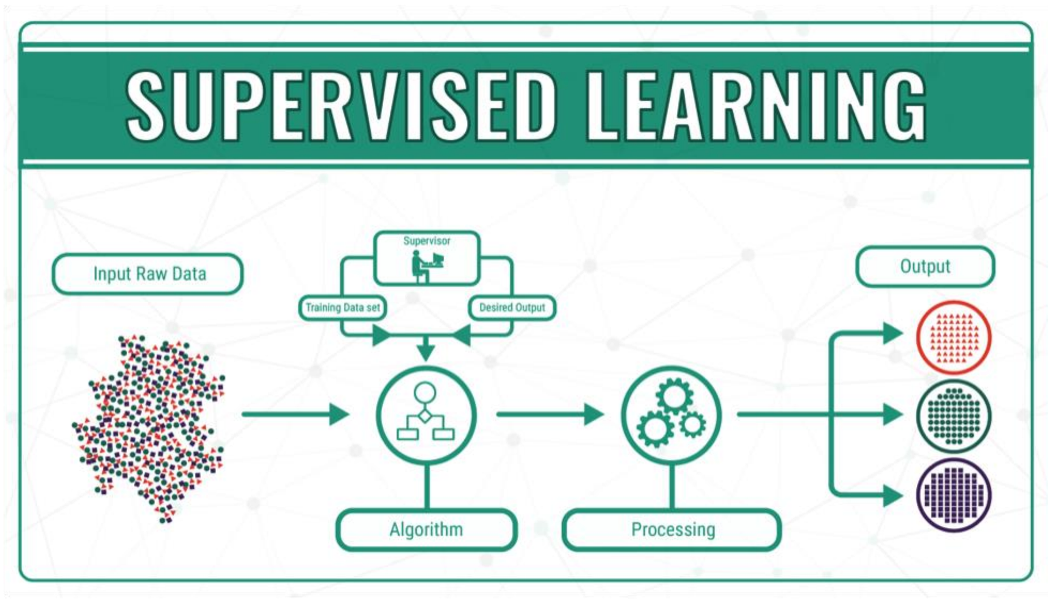


Figure 3: Supervised Learning

1.4.2 Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabelled data. This means that human labour is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

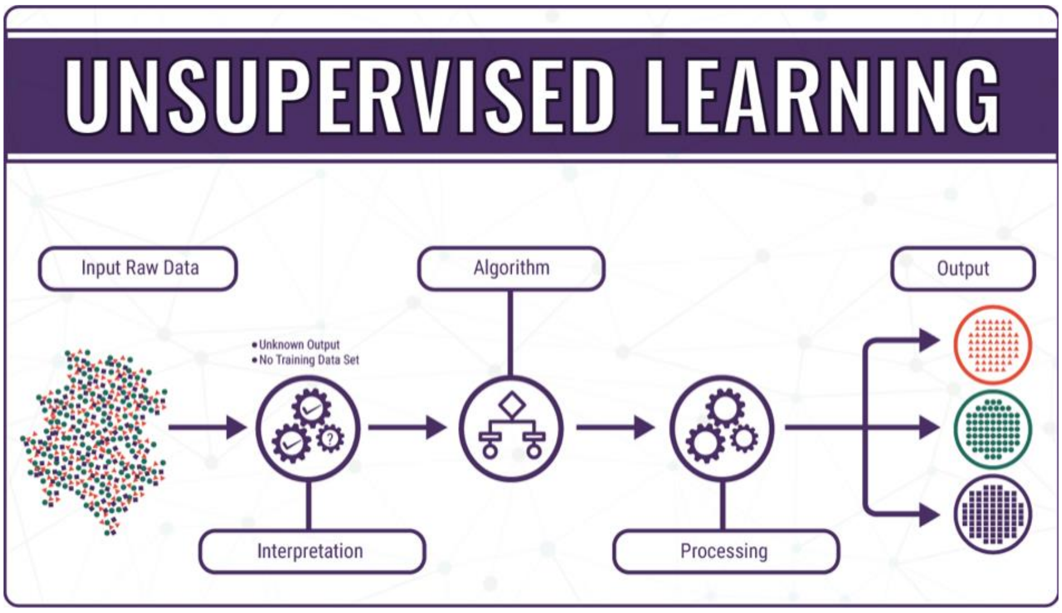


Figure 4: Unsupervised Learning

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 labelled and unlabelled data for training. In a typical scenario, the algorithm would use a small amount of labelled data with a large amount of unlabelled data.

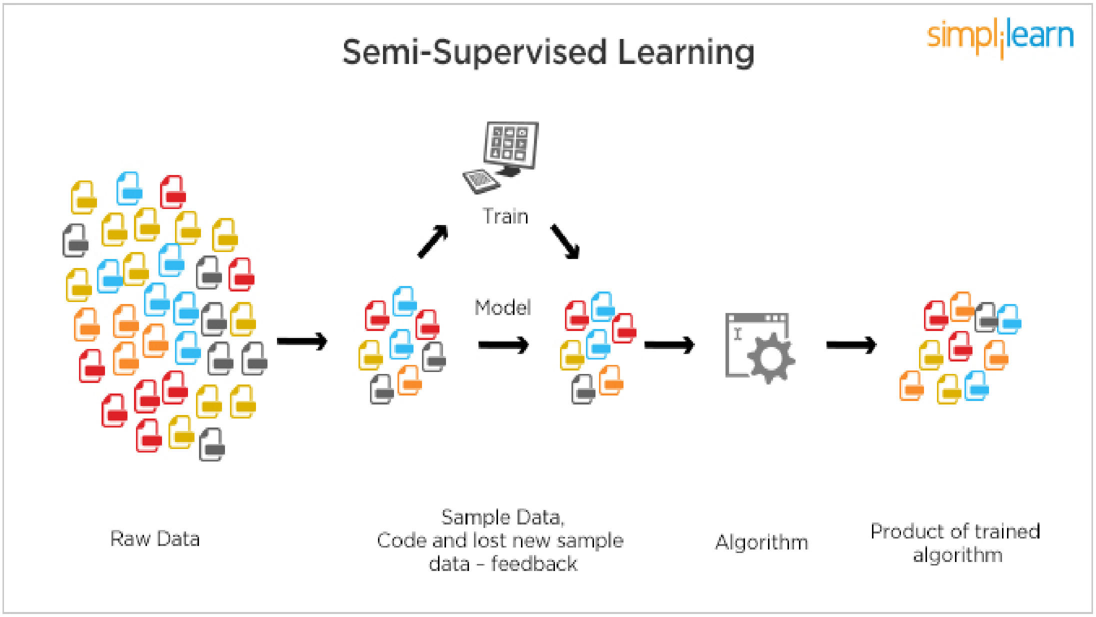

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 5: Semi Supervised Learning

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 discovers 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 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 PYHTON:

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

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:

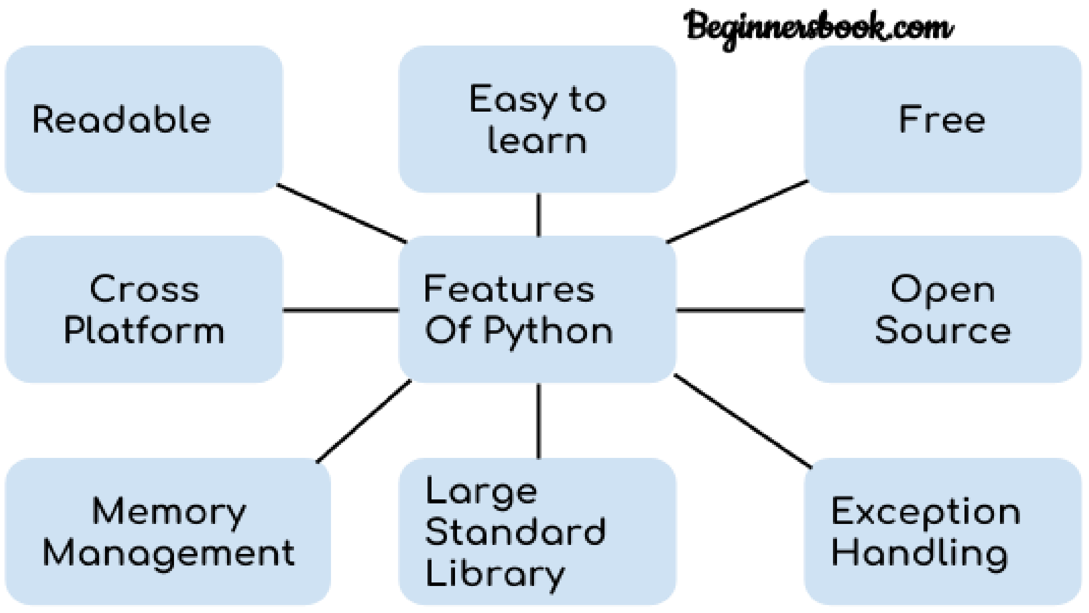


Figure 6: Features of python

- 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.

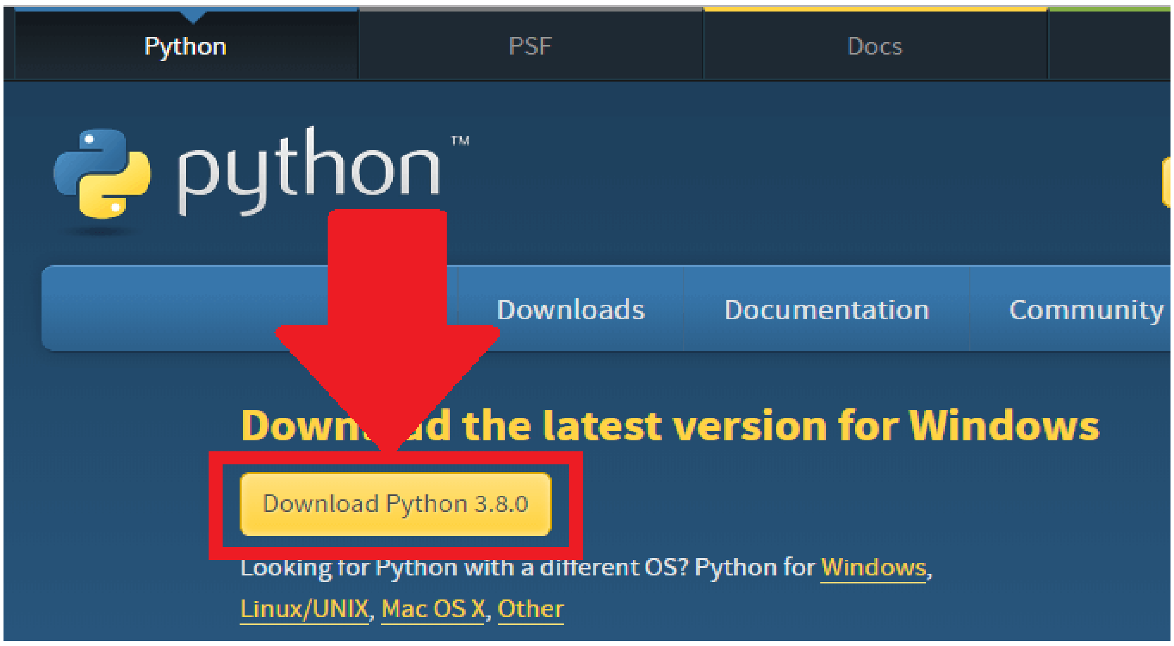
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
-

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



Down…ad the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Figure 7: Python download

- 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.

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 quickly installs and manages packages.

In WINDOWS:

In windows

- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer).
- Step 3: select installation type (all users).

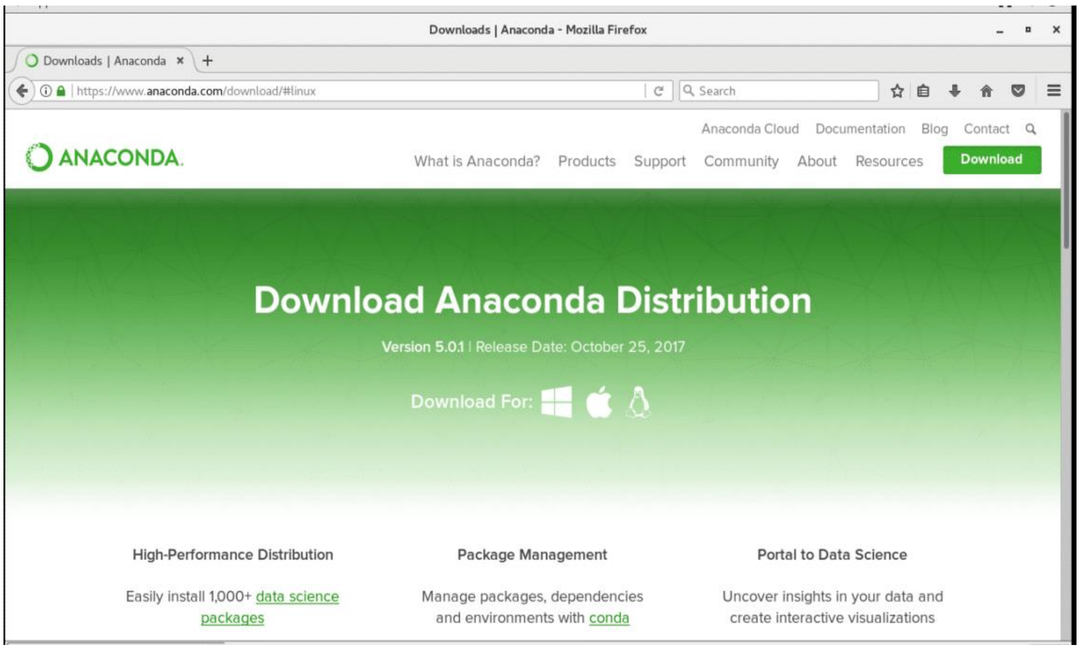
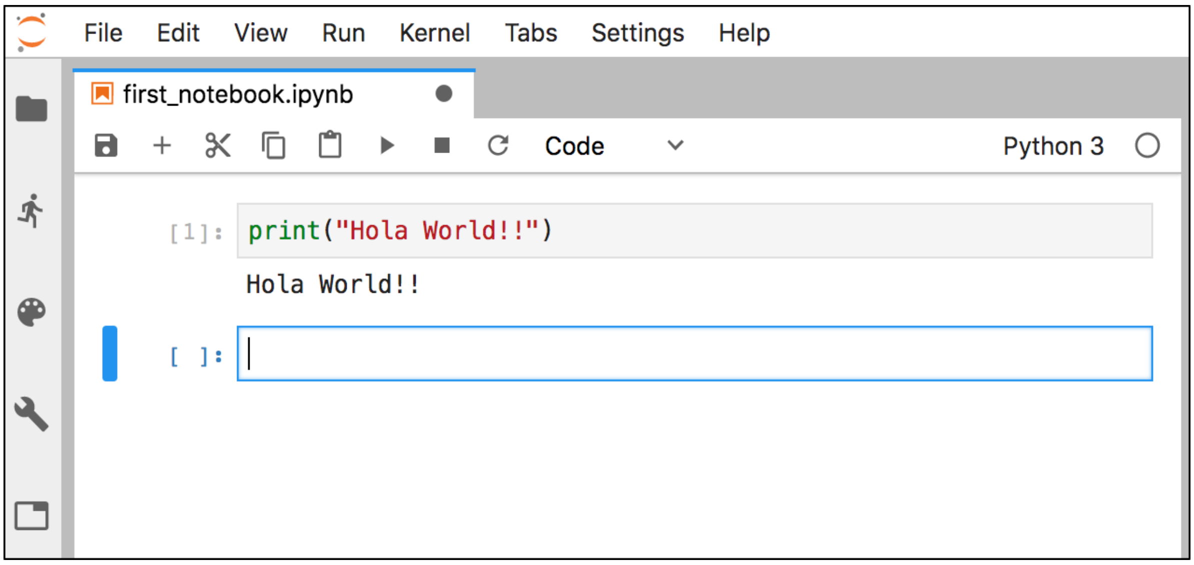

Downloads | Anaconda - Mozilla Firefox
https://www.anaconda.com/download/#linux
ANACONDA
What is Anaconda? Products Support Community About Resources Download
Download Anaconda Distribution
Version 5.0.1 | Release Date: October 25, 2017
Download For:
High-Performance Distribution
Easily install 1,000+ data science packages
Package Management
Manage packages, dependencies and environments with conda
Portal to Data Science
Uncover insights in your data and create interactive visualizations

Figure 8: Anaconda download

- 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).



```
[1]: print("Hola World!!")
Hola World!!
[ ]:
```

Figure 9: Jupyter notebook

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 –

1. Numbers
2. Strings
3. Lists
4. Tuples
5. 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 11 ([]).

● 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 type.

● 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 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 duct 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 OOP's CONCEPTS:

2.7.1 Class:

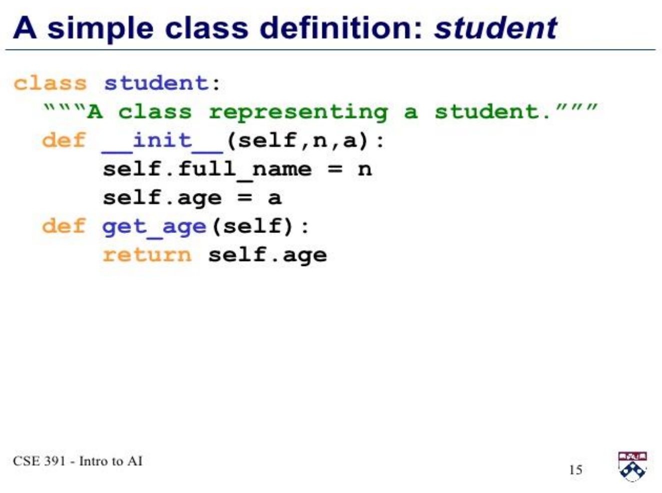
● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● 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: o We define a class in a very similar way how we define a function. o 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 14 indented like a functions body



A simple class definition: *student*

```
class student:
  """A class representing a student."""
  def __init__(self,n,a):
      self.full_name = n
      self.age = a
  def get_age(self):
      return self.age
```

CSE 391 - Intro to AI 15

Figure 10: Defining a Class

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

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To predict the quality of wine using machine algorithm using random forest classifier and decision tree classifier

3.2 DATA SET:

The given data set consists of the following parameters:

input variables (based on physicochemical tests):

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
M. Colour

3.3 OBJECTIVE OF THE CASE STUDY:

The task here is to predict the quality of wine on a scale of 0–10 given a set of features as inputs. I have solved it as a machine learning using random forest classifier and decision tree classifier. 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 (based on sensory data) is quality (score between 0 and 10). There are 11 columns describing chemical properties as follows.

CHAPTER 4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

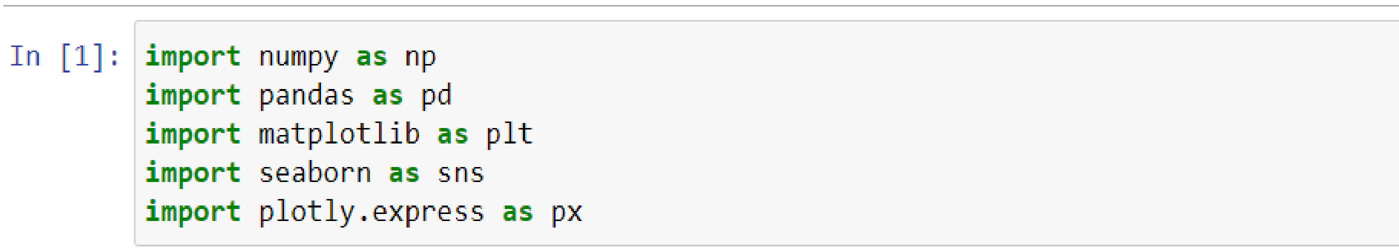
Pre-processing of the data actually involves the following steps:

- ### 4.1.1 GETTING THE DATASET:

We can get the data set from the database or we can get the data from client.

- ### 4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib as plt
        import seaborn as sns
        import plotly.express as px
```

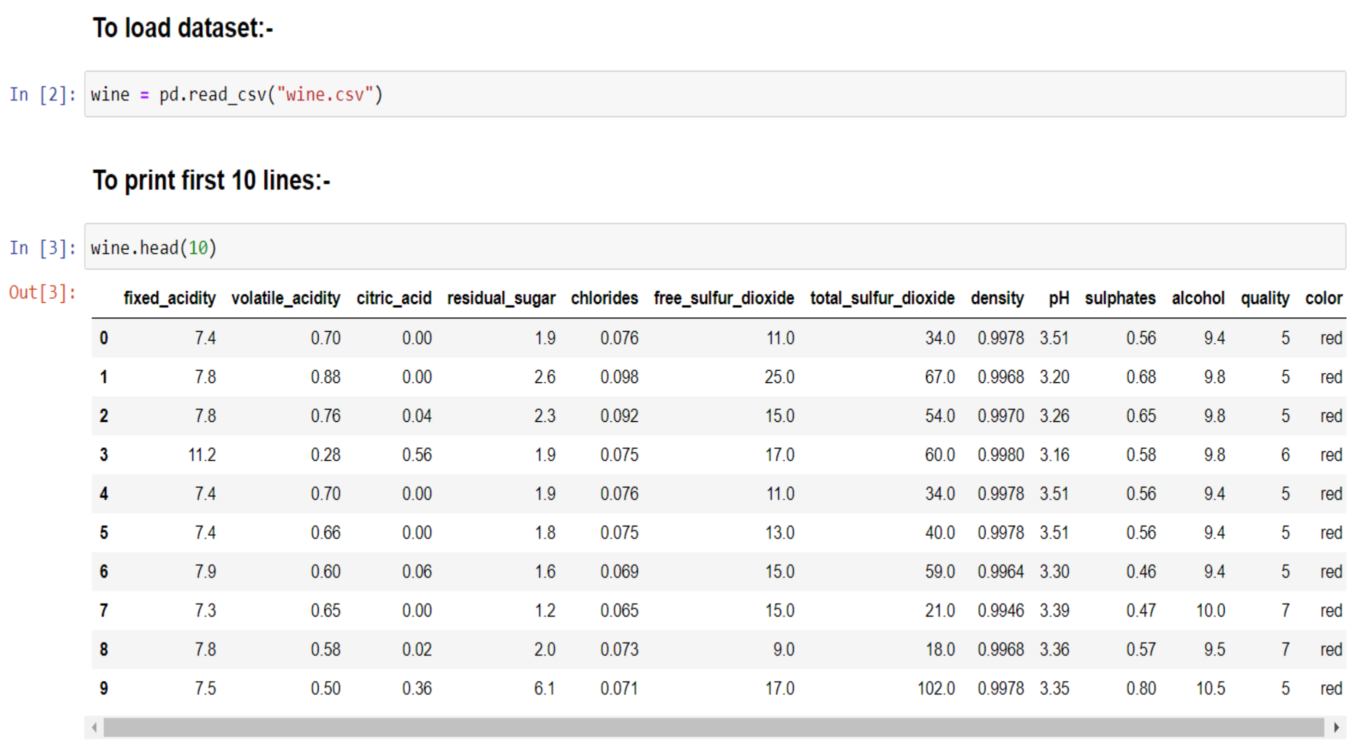
Figure 11: Importing Libraries

4.1.3 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the data frame. Any missing value or NaN value have to be cleaned.

Load CSV files to Python Pandas

1. Load the Pandas libraries with alias 'pd'
2. import pandas as pd.
3. **Read** data from file 'filename.csv'



To load dataset:-

```
In [2]: wine = pd.read_csv("wine.csv")
```

To print first 10 lines:-

```
In [3]: wine.head(10)
```

Out[3]:

	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	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5	red
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5	red
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7	red
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7	red
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5	red

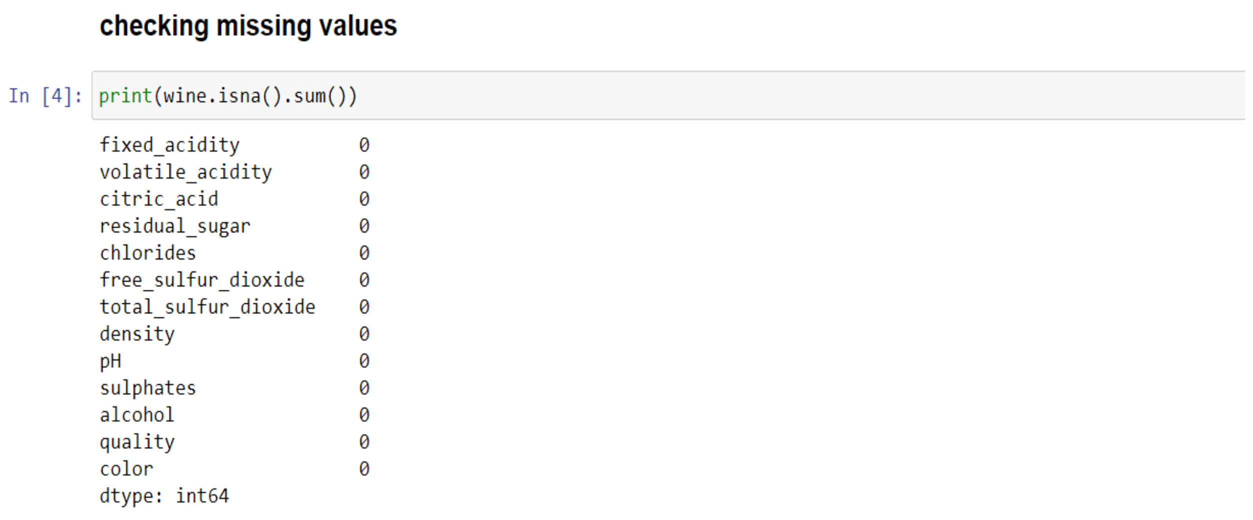
Figure 12: Reading the Data-Set

4.1.4 HANDLING MISSING VALUES:

- Using fillna(): -

Data Cleaning with Python and Pandas: Detecting Missing Values: -
In Pandas missing data is represented by two value:

- None: None is a Python singleton object that is often used for missing data in Python code.
- NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation



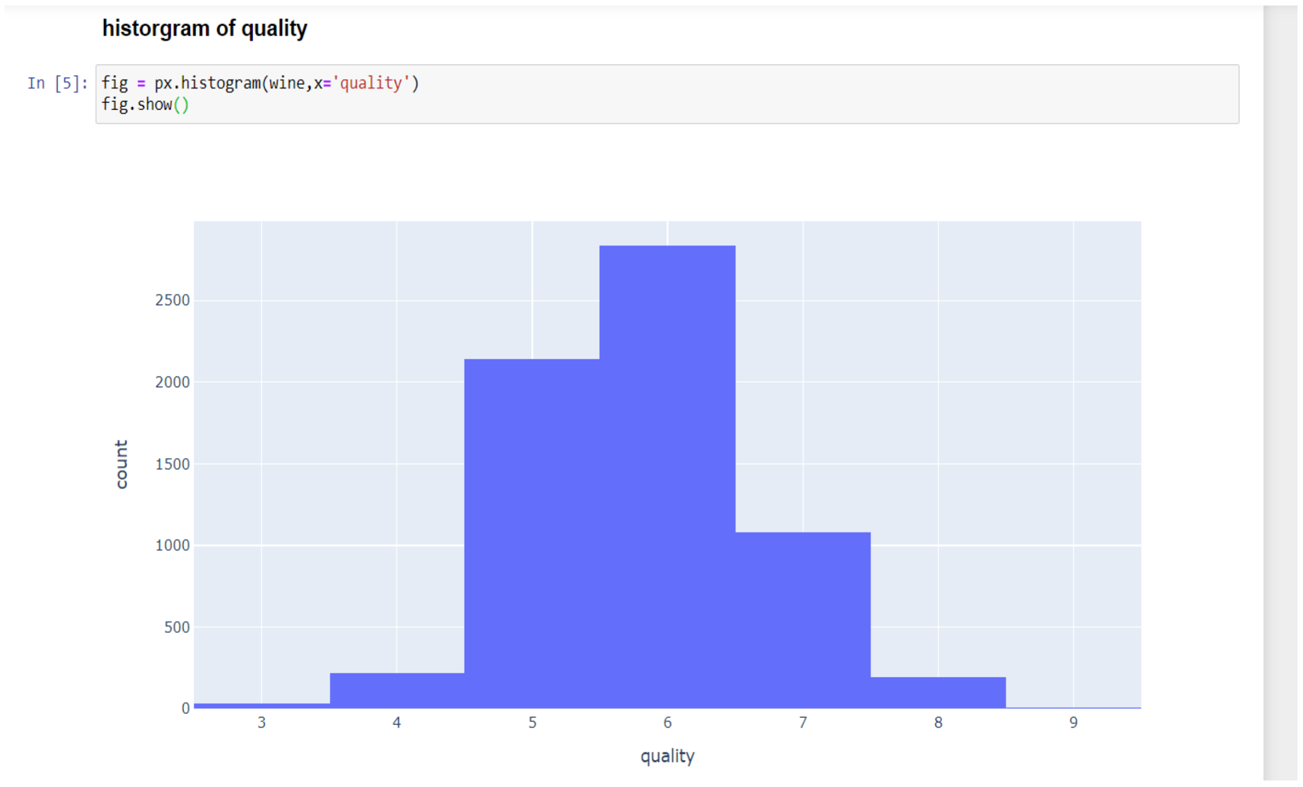
checking missing values

```
In [4]: print(wine.isna().sum())
```

```
fixed_acidity           0
volatile_acidity        0
citric_acid             0
residual_sugar          0
chlorides               0
free_sulfur_dioxide     0
total_sulfur_dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
color                   0
dtype: int64
```

Figure 13: There are no missing values

Representation of Histogram of Quality: -



historgram of quality

```
In [5]: fig = px.histogram(wine,x='quality')
        fig.show()
```

Figure 14: Histogram of quality

Correlation Matrix: -

Next, I wanted to see the correlations between the variables that I'm working with. This allows me to get a much better understanding of the relationships between my variables in a quick glimpse.

Correlation Matrix to check features

```
In [6]:  plt.pyplot.subplots(figsize=(13, 9))
         sns.heatmap(wine.corr(), annot=True)

Out[6]:  <matplotlib.axes._subplots.AxesSubplot at 0x1e43fe41808>
```

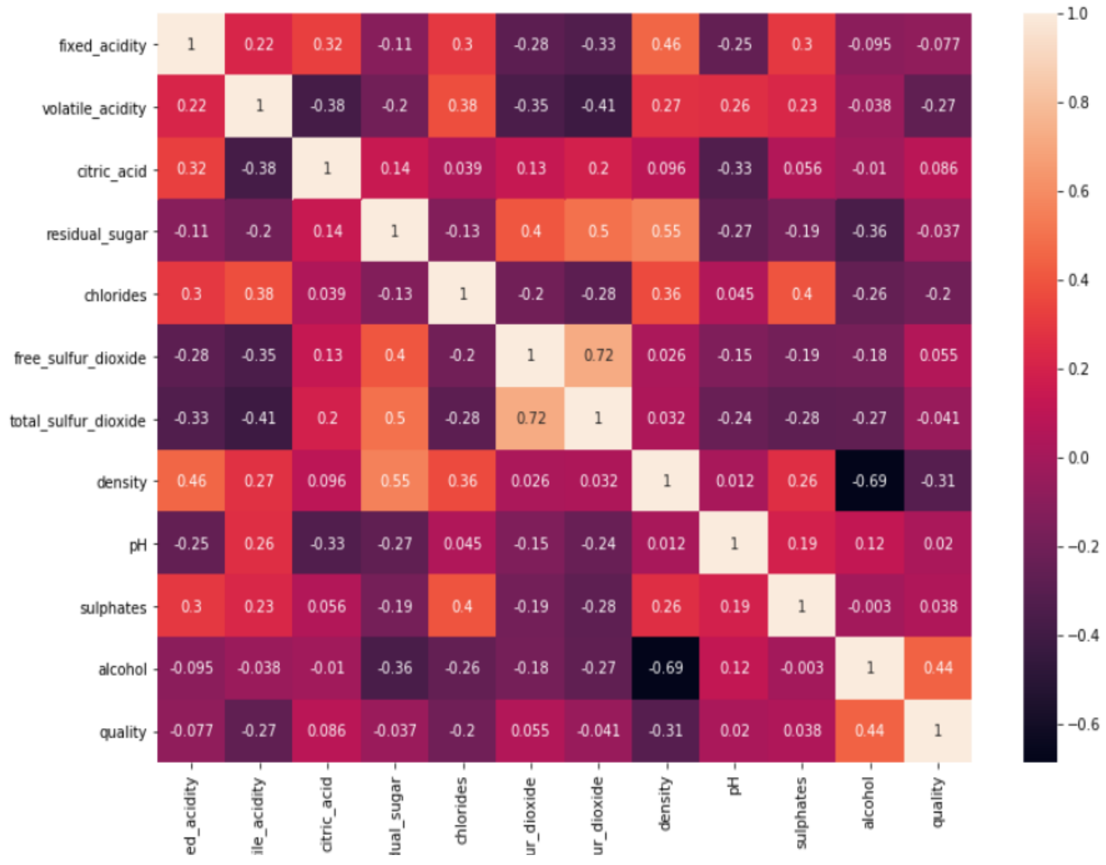
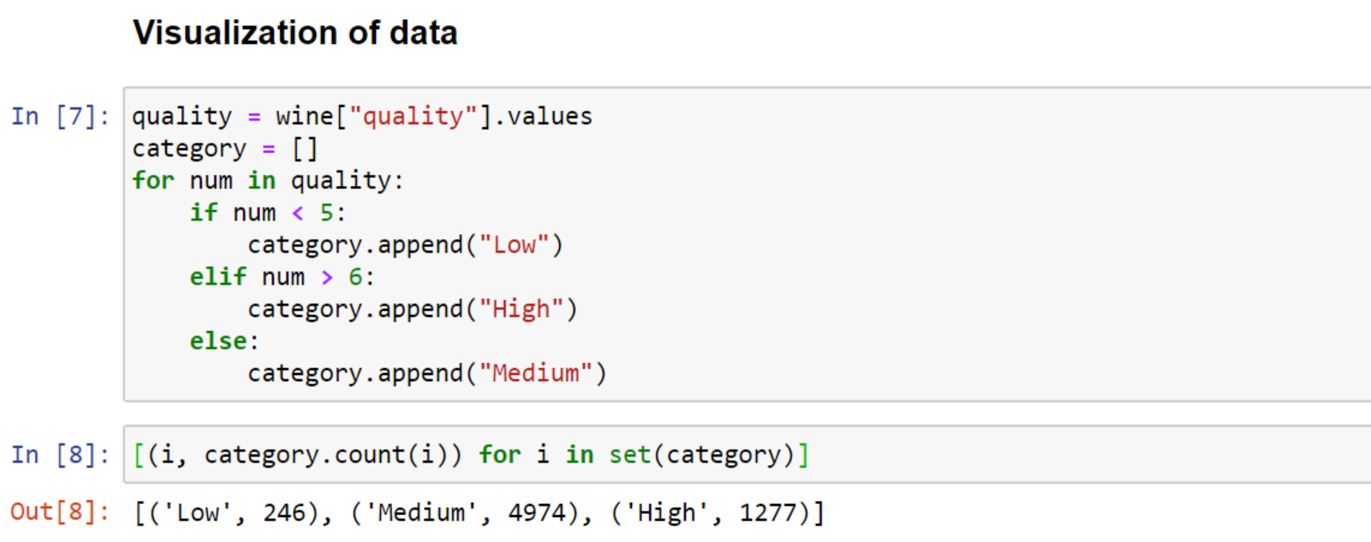


Figure 15:-Correlation Matrix

Convert to a Classification Problem: -

For this problem, I defined a bottle of wine as 'high' if it had a quality score of 7 or higher, and if it had a score of less than 7, it was deemed 'low', or else 'medium

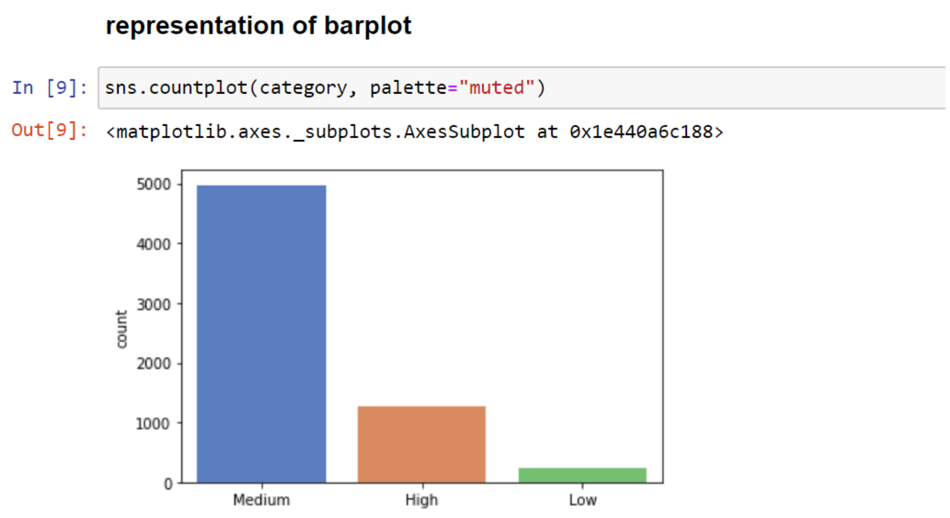


Visualization of data

```
In [7]: quality = wine["quality"].values
        category = []
        for num in quality:
            if num < 5:
                category.append("Low")
            elif num > 6:
                category.append("High")
            else:
                category.append("Medium")

In [8]: [(i, category.count(i)) for i in set(category)]

Out[8]: [('Low', 246), ('Medium', 4974), ('High', 1277)]
```

Figure 16: - Representation of Data

To understand the above classified data, we represented a bar plot: -

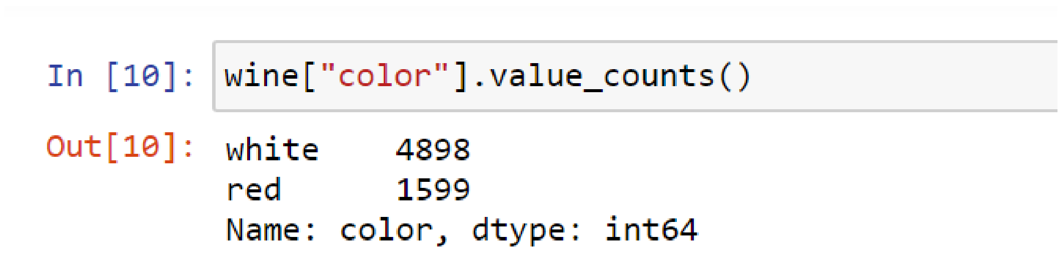


representation of barplot

```
In [9]: sns.countplot(category, palette="muted")

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1e440a6c188>
```

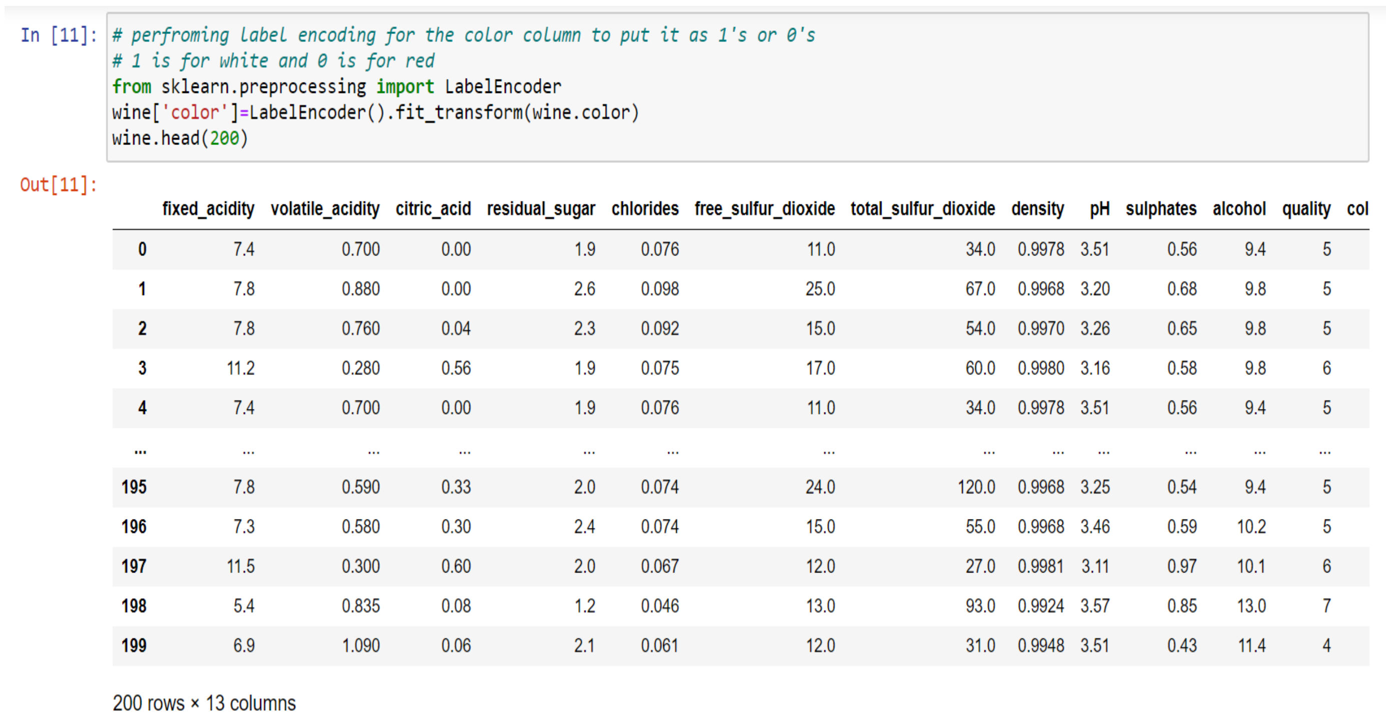
Figure 17: -Representation of Bar plot

Next, we changed the values of colour of wine to 0’s and 1’s



```
In [10]: wine["color"].value_counts()

Out[10]: white    4898
         red      1599
         Name: color, dtype: int64
```

- performing label encoding for the colour column to put it as 1's or 0's
- 1 is for white and 0 is for red



```
In [11]: # perfroming label encoding for the color column to put it as 1's or 0's
         # 1 is for white and 0 is for red
         from sklearn.preprocessing import LabelEncoder
         wine['color']=LabelEncoder().fit_transform(wine.color)
         wine.head(200)
```

Out[11]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	col
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	
195	7.8	0.590	0.33	2.0	0.074	24.0	120.0	0.9968	3.25	0.54	9.4	5	
196	7.3	0.580	0.30	2.4	0.074	15.0	55.0	0.9968	3.46	0.59	10.2	5	
197	11.5	0.300	0.60	2.0	0.067	12.0	27.0	0.9981	3.11	0.97	10.1	6	
198	5.4	0.835	0.08	1.2	0.046	13.0	93.0	0.9924	3.57	0.85	13.0	7	
199	6.9	1.090	0.06	2.1	0.061	12.0	31.0	0.9948	3.51	0.43	11.4	4	

200 rows × 13 columns

Figure 18: -Performing Label Encoding

Preparing Data for Modelling: -

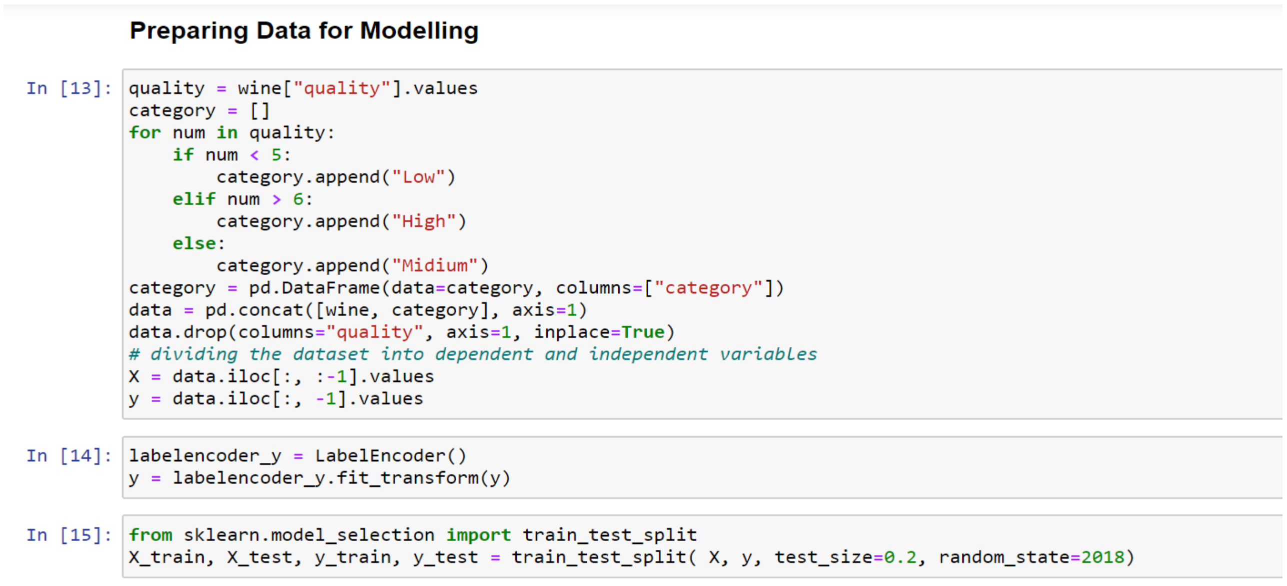
train_test_split is a function in **Sklearn model selection** for splitting data arrays into **two subsets**: for training data and for testing data. With this function, you don't need to divide the dataset manually.

By default, Sklearn **train_test_split** will make random partitions for the two subsets. However, you can also specify a random state for the operation.

Sklearn test_train_split has several **parameters**. A basic example of the syntax would look like this:

```
train_test_split(X, y, train_size=0.*,test_size=0.*, random_state=*)
```

- X, y. The first parameter is the **dataset** you're selecting to use.
- train_size. This parameter sets the **size of the training dataset**. There are three options: None, which is the default, Int, which requires the exact number of samples, and float, which ranges from 0.1 to 1.0.
- test_size. This parameter specifies the **size of the testing dataset**. The default state suits the training size. It will be set to **0.20-0.25** if the training size is set to default.

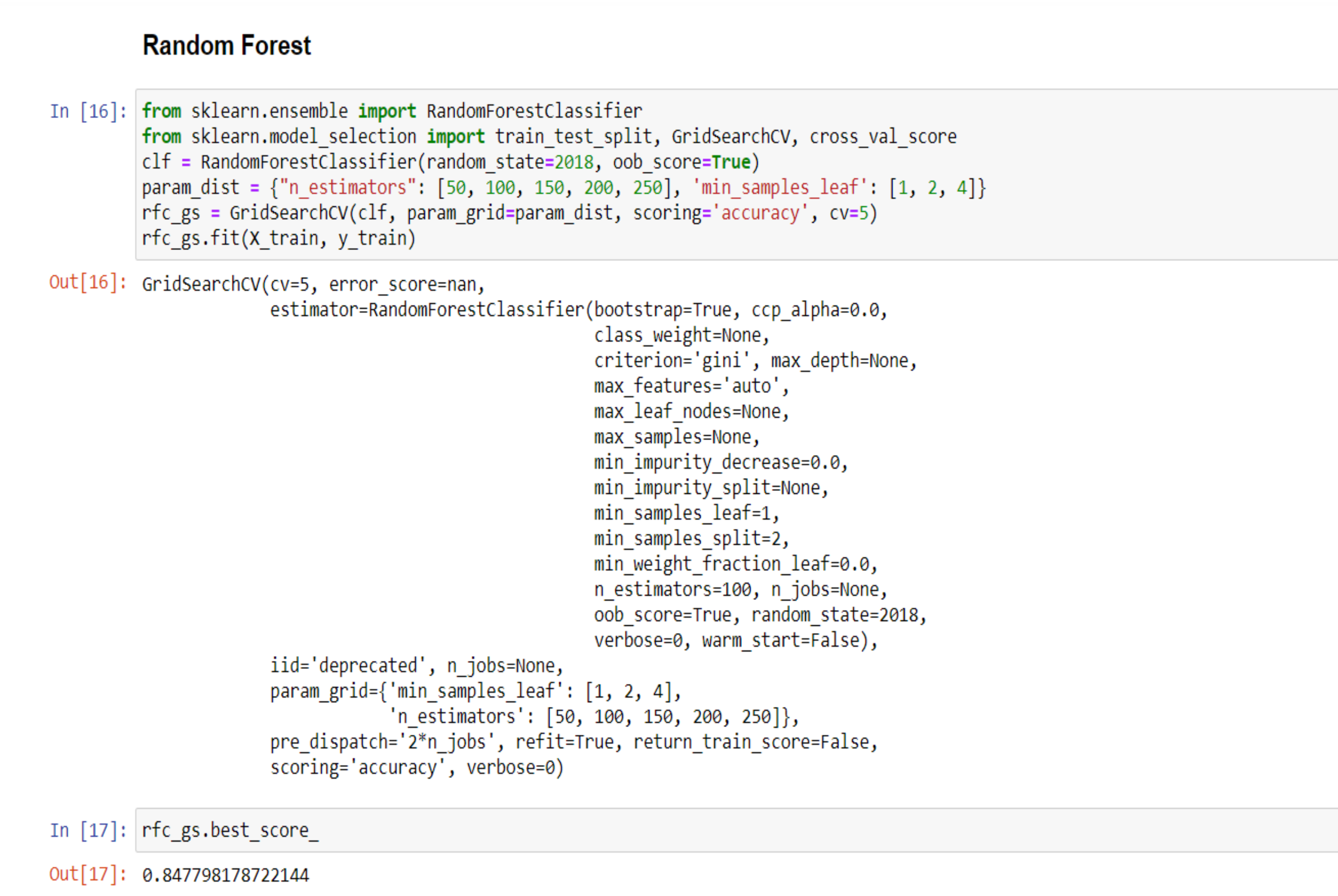


Preparing Data for Modelling

```
In [13]: quality = wine["quality"].values
         category = []
         for num in quality:
             if num < 5:
                 category.append("Low")
             elif num > 6:
                 category.append("High")
             else:
                 category.append("Midium")
         category = pd.DataFrame(data=category, columns=["category"])
         data = pd.concat([wine, category], axis=1)
         data.drop(columns="quality", axis=1, inplace=True)
         # dividing the dataset into dependent and independent variables
         X = data.iloc[:, :-1].values
         y = data.iloc[:, -1].values

In [14]: labelencoder_y = LabelEncoder()
         y = labelencoder_y.fit_transform(y)

In [15]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=2018)
```

Figure 19: -Preparing Data

Performing Algorithm (Random forest classifier): -

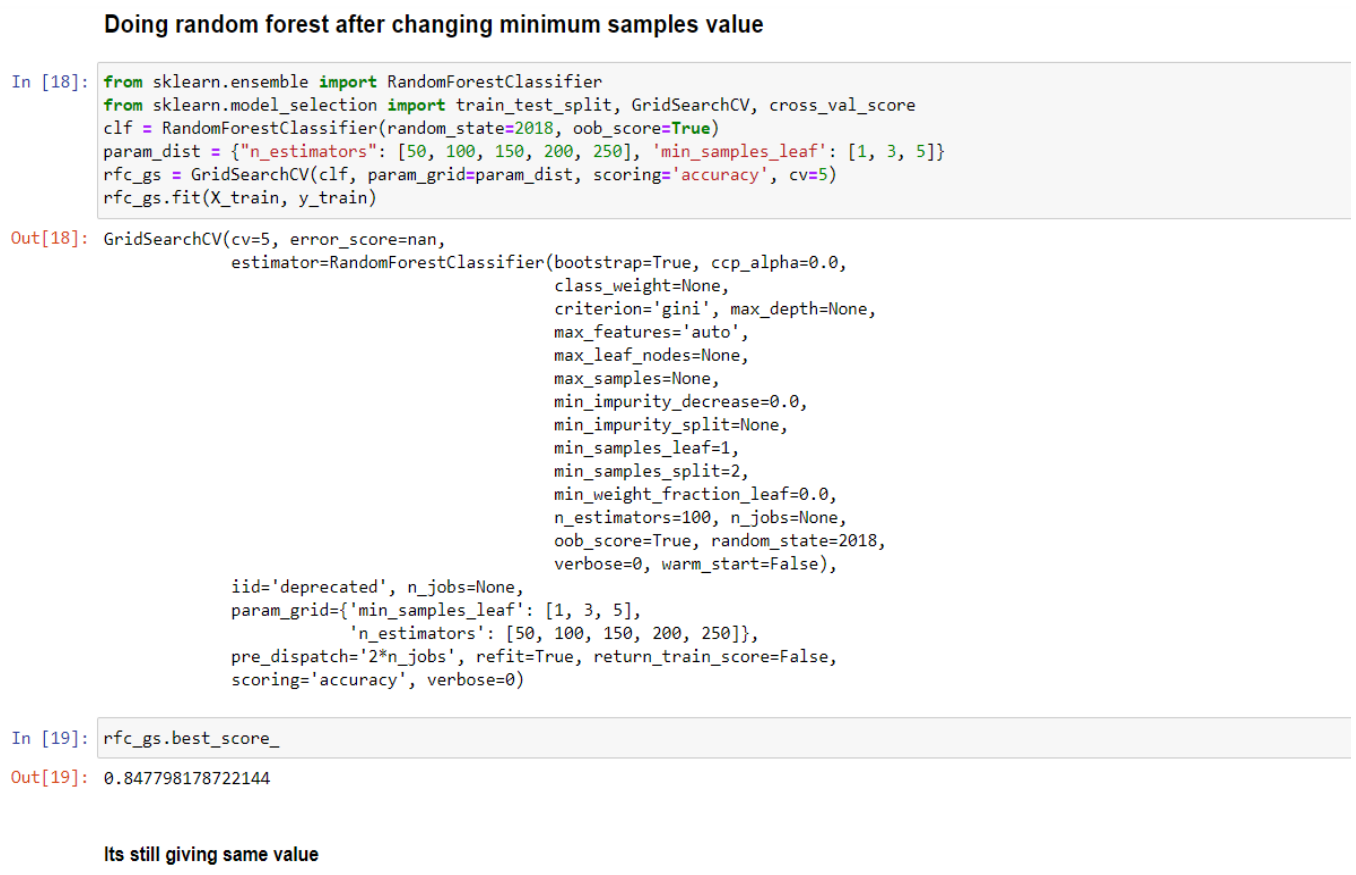

Random Forest

```
In [16]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
         clf = RandomForestClassifier(random_state=2018, oob_score=True)
         param_dist = {"n_estimators": [50, 100, 150, 200, 250], 'min_samples_leaf': [1, 2, 4]}
         rfc_gs = GridSearchCV(clf, param_grid=param_dist, scoring='accuracy', cv=5)
         rfc_gs.fit(X_train, y_train)

Out[16]: GridSearchCV(cv=5, error_score=nan,
                      estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                                       class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features='auto',
                                                       max_leaf_nodes=None,
                                                       max_samples=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       n_estimators=100, n_jobs=None,
                                                       oob_score=True, random_state=2018,
                                                       verbose=0, warm_start=False),
                      iid='deprecated', n_jobs=None,
                      param_grid={'min_samples_leaf': [1, 2, 4],
                                  'n_estimators': [50, 100, 150, 200, 250]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring='accuracy', verbose=0)

In [17]: rfc_gs.best_score_

Out[17]: 0.847798178722144
```

Figure 20: - Performing Random Forest Algorithm

- **By performing Random Forest, we are getting 84% value**

We are performing again changing minimum sample values: -



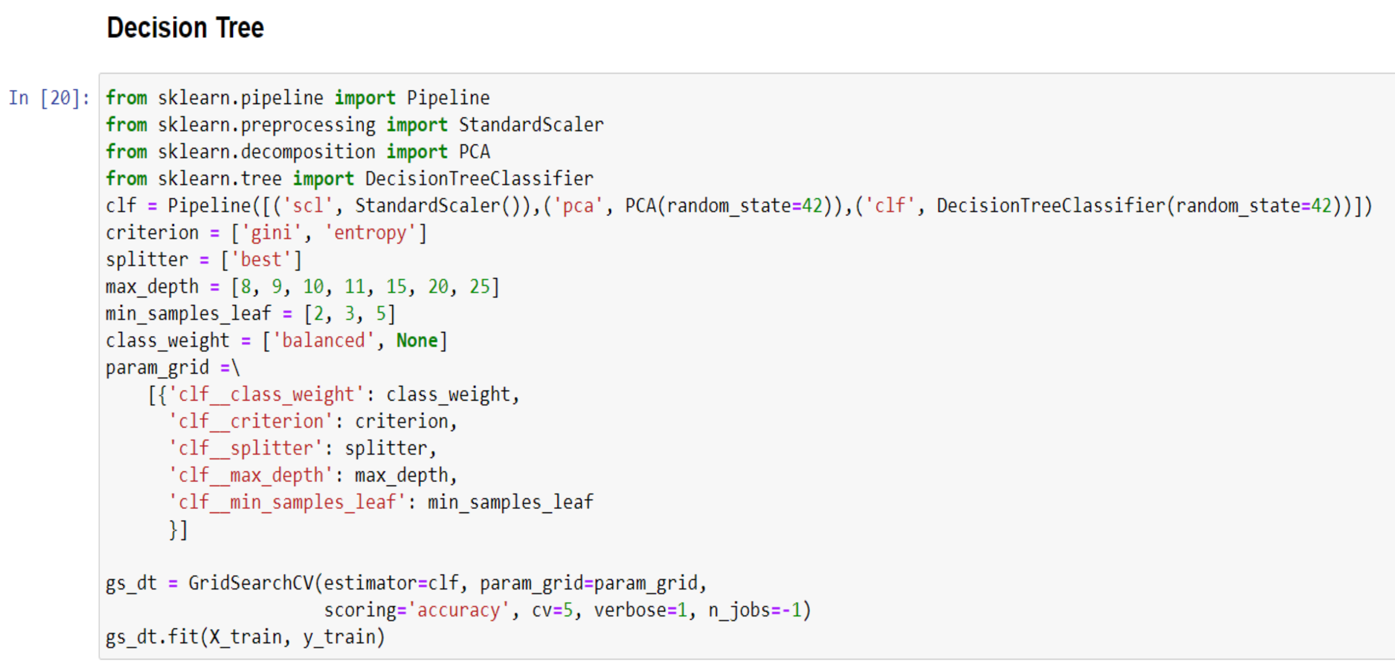
Doing random forest after changing minimum samples value

```
In [18]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
         clf = RandomForestClassifier(random_state=2018, oob_score=True)
         param_dist = {"n_estimators": [50, 100, 150, 200, 250], 'min_samples_leaf': [1, 3, 5]}
         rfc_gs = GridSearchCV(clf, param_grid=param_dist, scoring='accuracy', cv=5)
         rfc_gs.fit(X_train, y_train)

Out[18]: GridSearchCV(cv=5, error_score=nan,
                      estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                                       class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features='auto',
                                                       max_leaf_nodes=None,
                                                       max_samples=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       n_estimators=100, n_jobs=None,
                                                       oob_score=True, random_state=2018,
                                                       verbose=0, warm_start=False),
                      iid='deprecated', n_jobs=None,
                      param_grid={'min_samples_leaf': [1, 3, 5],
                                  'n_estimators': [50, 100, 150, 200, 250]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring='accuracy', verbose=0)

In [19]: rfc_gs.best_score_

Out[19]: 0.847798178722144
```

Its still giving same value

Figure 21: -Random forest Result (After changing values)

Performing Another machine algorithm (Decision tree classifier): -

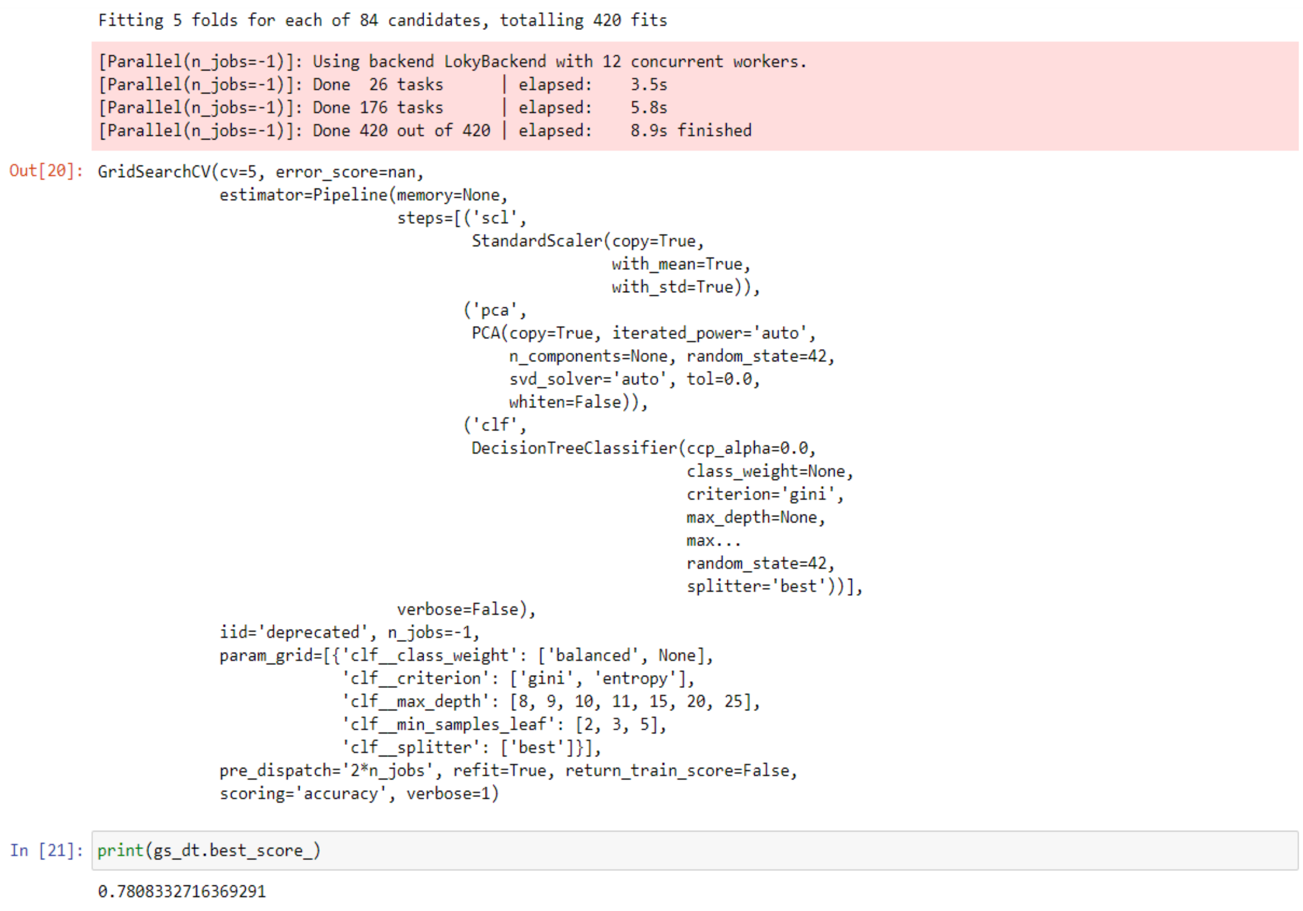


Decision Tree

```
In [20]: from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler
         from sklearn.decomposition import PCA
         from sklearn.tree import DecisionTreeClassifier
         clf = Pipeline([('scl', StandardScaler()),('pca', PCA(random_state=42)),('clf', DecisionTreeClassifier(random_state=42))])
         criterion = ['gini', 'entropy']
         splitter = ['best']
         max_depth = [8, 9, 10, 11, 15, 20, 25]
         min_samples_leaf = [2, 3, 5]
         class_weight = ['balanced', None]
         param_grid =\
             [{'clf__class_weight': class_weight,
               'clf__criterion': criterion,
               'clf__splitter': splitter,
               'clf__max_depth': max_depth,
               'clf__min_samples_leaf': min_samples_leaf
               }]

         gs_dt = GridSearchCV(estimator=clf, param_grid=param_grid,
                              scoring='accuracy', cv=5, verbose=1, n_jobs=-1)
         gs_dt.fit(X_train, y_train)
```

Figure 22: -Performing Decision Tree Classifier

Result of Decision Tree classifier



```
Fitting 5 folds for each of 84 candidates, totalling 420 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done 176 tasks      | elapsed:    5.8s
[Parallel(n_jobs=-1)]: Done 420 out of 420 | elapsed:    8.9s finished

Out[20]: GridSearchCV(cv=5, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('scl',
                                        StandardScaler(copy=True,
                                                       with_mean=True,
                                                       with_std=True)),
                                       ('pca',
                                        PCA(copy=True, iterated_power='auto',
                                            n_components=None, random_state=42,
                                            svd_solver='auto', tol=0.0,
                                            whiten=False)),
                                       ('clf',
                                        DecisionTreeClassifier(ccp_alpha=0.0,
                                                               class_weight=None,
                                                               criterion='gini',
                                                               max_depth=None,
                                                               max...
                                                               random_state=42,
                                                               splitter='best'))],
                                verbose=False),
             iid='deprecated', n_jobs=-1,
             param_grid=[{'clf__class_weight': ['balanced', None],
                          'clf__criterion': ['gini', 'entropy'],
                          'clf__max_depth': [8, 9, 10, 11, 15, 20, 25],
                          'clf__min_samples_leaf': [2, 3, 5],
                          'clf__splitter': ['best']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=1)

In [21]: print(gs_dt.best_score_)

0.7808332716369291
```

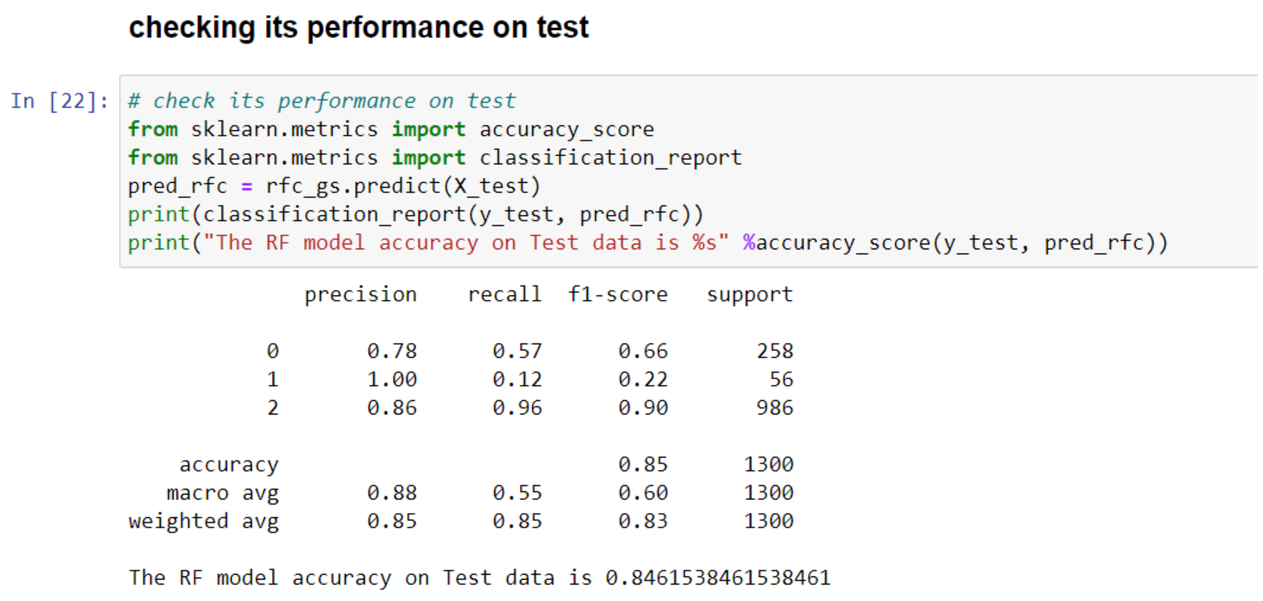
Figure 23: -Result of Decision tree classifier

- **By performing decision tree classifier, we are getting 78% accurate value**

So, after performing random forest and decision Tree classifier, comparing both we got random forest value 0.84 which is higher than decision tree classifier, we need to check performance of random forest.

Classification report: -

It is used to measure the quality of predictions from a **classification** algorithm. How many predictions are True and how many are False? More specifically, True Positives, False Positives, True negatives and False Negatives.



checking its performance on test

```
In [22]: # check its performance on test
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import classification_report
         pred_rfc = rfc_gs.predict(X_test)
         print(classification_report(y_test, pred_rfc))
         print("The RF model accuracy on Test data is %s" %accuracy_score(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.78      0.57      0.66       258
           1       1.00      0.12      0.22        56
           2       0.86      0.96      0.90       986

    accuracy                           0.85      1300
   macro avg       0.88      0.55      0.60      1300
weighted avg       0.85      0.85      0.83      1300

The RF model accuracy on Test data is 0.8461538461538461
```

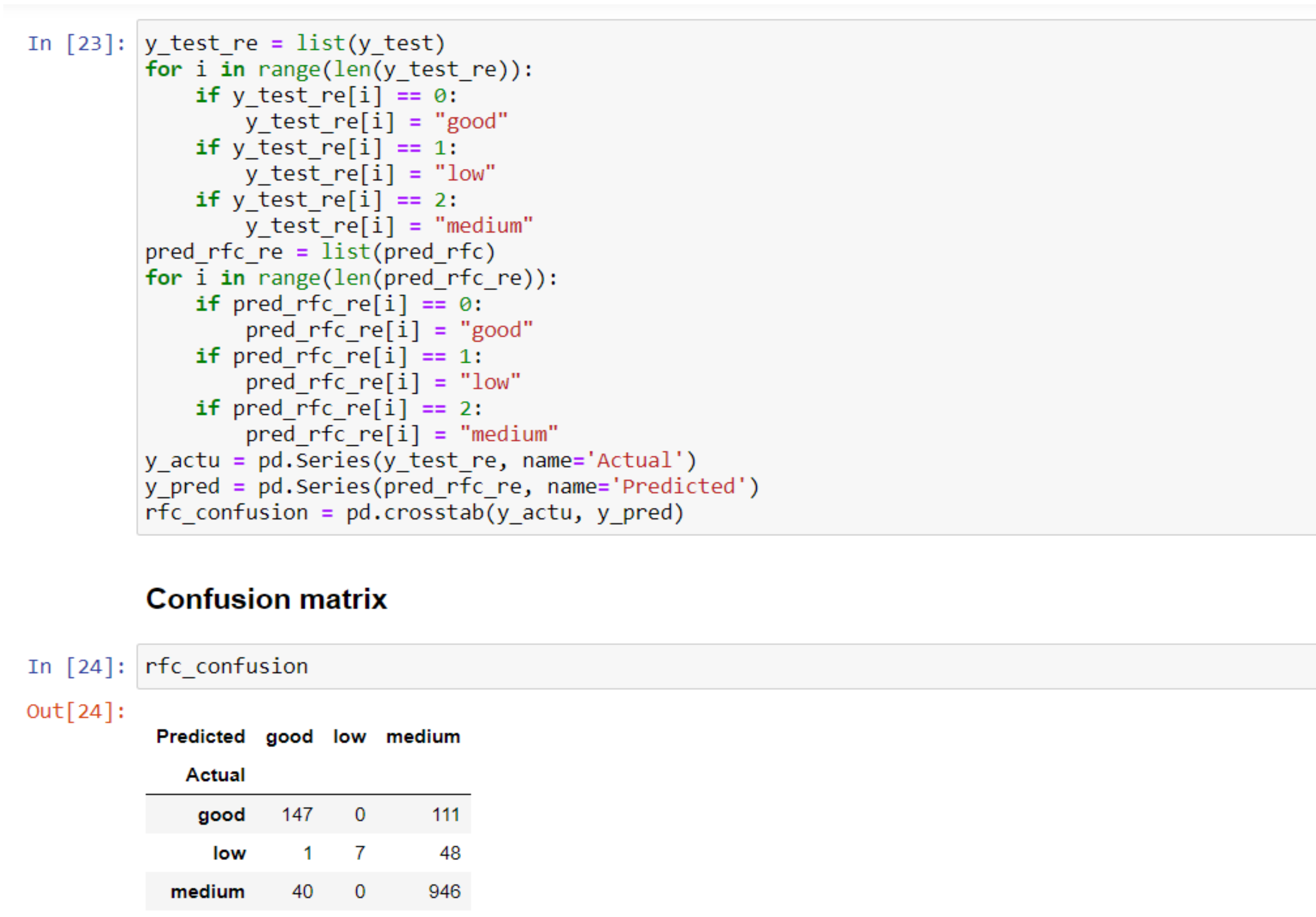
Figure 24: -Checking Performance test

By using performance, the value of random forest has slightly increased. This is known as Optimization of the code.

CONFUSION MATRIX: -

A **confusion matrix** is a tabular summary of the number of correct and incorrect predictions made by a classifier. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

Now we need to perform confusion matrix: -



```
In [23]: y_test_re = list(y_test)
         for i in range(len(y_test_re)):
             if y_test_re[i] == 0:
                 y_test_re[i] = "good"
             if y_test_re[i] == 1:
                 y_test_re[i] = "low"
             if y_test_re[i] == 2:
                 y_test_re[i] = "medium"
         pred_rfc_re = list(pred_rfc)
         for i in range(len(pred_rfc_re)):
             if pred_rfc_re[i] == 0:
                 pred_rfc_re[i] = "good"
             if pred_rfc_re[i] == 1:
                 pred_rfc_re[i] = "low"
             if pred_rfc_re[i] == 2:
                 pred_rfc_re[i] = "medium"
         y_actu = pd.Series(y_test_re, name='Actual')
         y_pred = pd.Series(pred_rfc_re, name='Predicted')
         rfc_confusion = pd.crosstab(y_actu, y_pred)
```

Confusion matrix

```
In [24]: rfc_confusion
```

Out[24]:

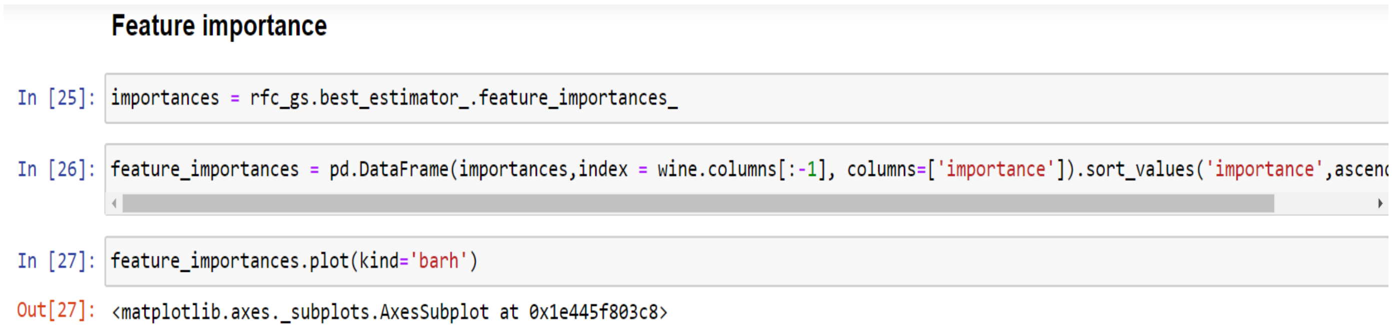
Predicted	good	low	medium
Actual			
good	147	0	111
low	1	7	48
medium	40	0	946

Figure 25: -Confusion Matrix

FEATURE IMPORTANCE: -

Random forest feature importance. Random forests are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use. They also provide two straightforward methods for **feature** selection: mean decrease impurity and mean decrease accuracy

Now we need to do feature performance test



Feature importance

```
In [25]: importances = rfc_gs.best_estimator_.feature_importances_

In [26]: feature_importances = pd.DataFrame(importances,index = wine.columns[:-1], columns=['importance']).sort_values('importance',ascend

In [27]: feature_importances.plot(kind='barh')

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1e445f803c8>
```

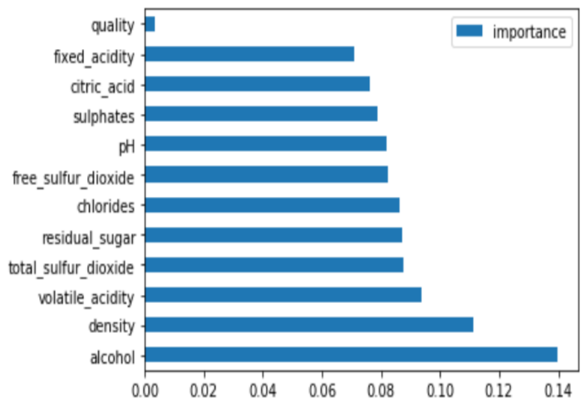


Figure 26: -Feature Importance bar plot

CONCLUSION: -

By looking above bar plot, we can say that good quality wines have higher levels of alcohol on average, have lower volatile acidity on average, higher levels of sulphates, and higher levels of residual sugar on average.

The two most important features among all 12 attributes are Sulphur dioxide (both free and total) and Alcohol. LAST Volatile acidity contributes to acidic tastes and have negative correlation to wine quality. SECOND The most important factor to decide the quality of wine is alcohol, higher concentration of alcohol leads to better quality of wine and lower density of wine.

REFERENCES: -

https://www.kaggle.com/vishalyo990/prediction-of-quality-of-wine

https://medium.com/themlblog/wine-quality-prediction-using-machine-learning-59c88a826789