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 and Engineering

By

Paruchuri Sai Sumanth

(221710312039)

Under the guidance

Of

Mrs. K. Neha

Assistant Professor

GITAM (DEEMED TO BE UNIVERSITY)

VISAKHAPATNAM HYDERABAD BENGALURU

Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

June 2020

DECLARATION

I **Paruchuri Sai Sumanth** 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 and Engineering**”. I declare that it was carried out independently by me under the guidance of **Mrs. K. Neha**, 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 **Paruchuri Sai Sumanth**

Date: **221710312039**

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Date:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **“WINE QUALITY PREDICTION”** is being submitted by **Paruchuri Sai Sumanth(221710312039)** in partial fulfillment 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 2018-19

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.

Mrs.K. Neha	**Dr.S. Phani Kumar**
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

I would like to thank respected **Dr. S. Phani Kumar,** 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 a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mrs. K. Neha** 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.

Paruchuri Sai Sumanth

221710312039

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 strategically approach for solution of the client, I have adapted the view point 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 correlate 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

1.1 INTRODUCTION:

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence (AI).

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 Face book, 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

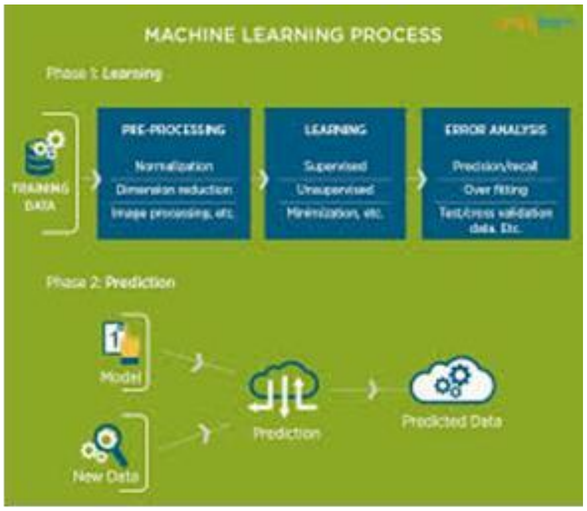


Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

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 labeled 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.

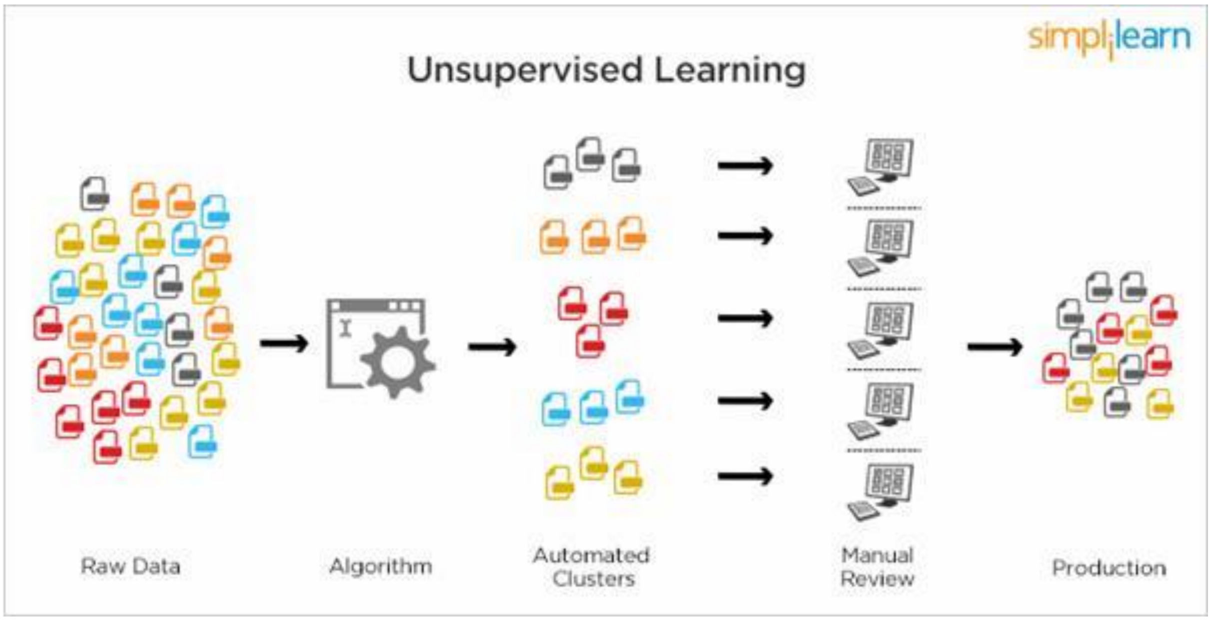


Figure 2 : Unsupervised Learning

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.

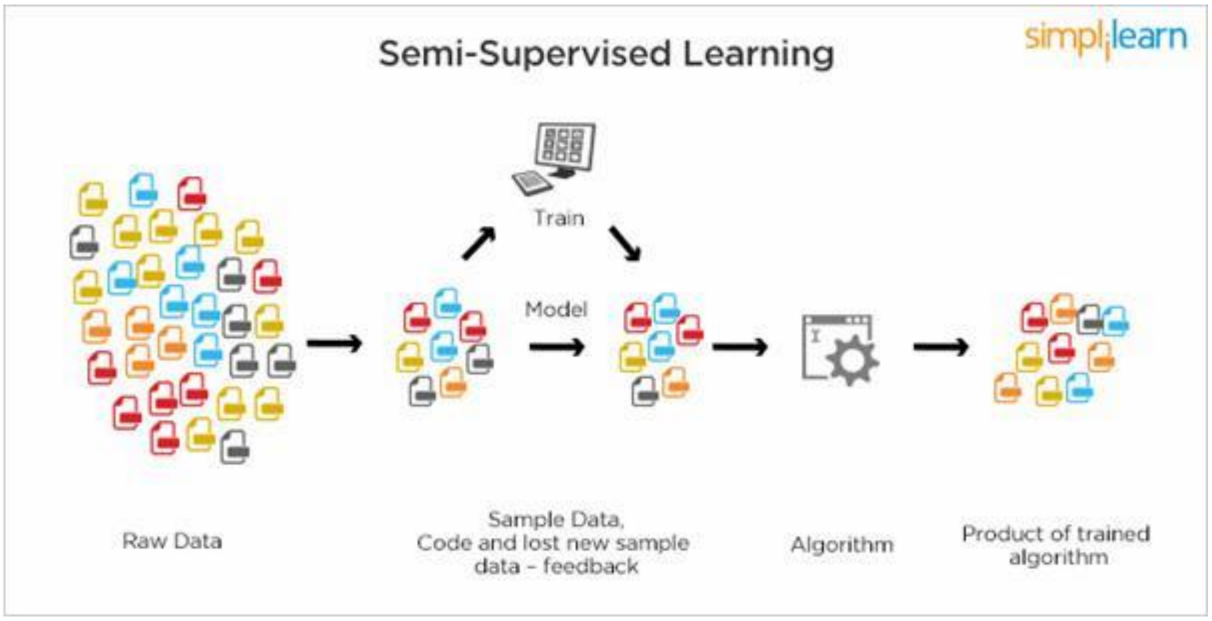


Figure 3 : Semi Supervised Learning

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 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 directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2HISTORY 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.

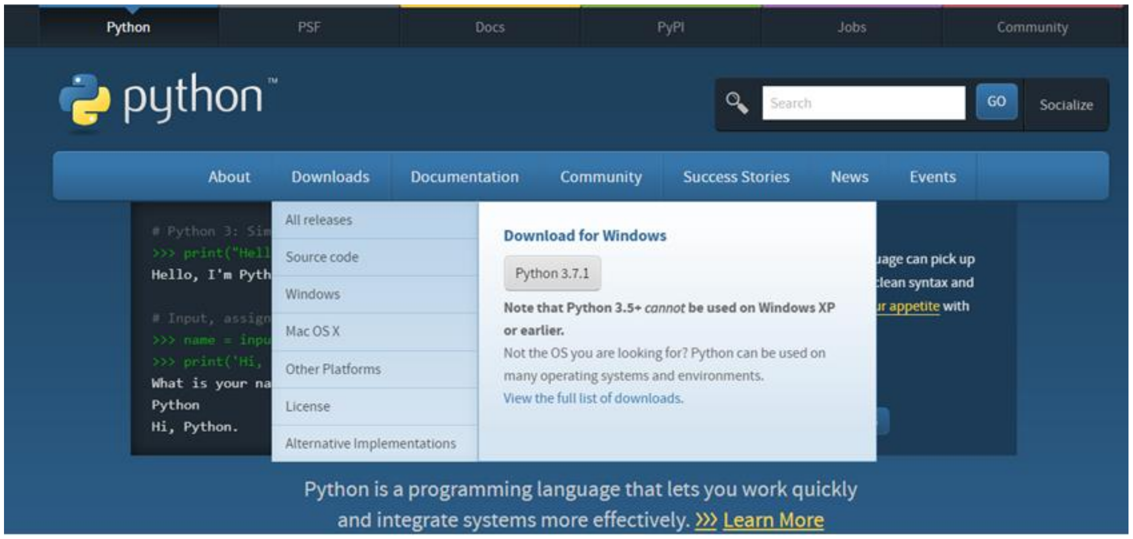
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.

2.4HOW 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.



```
# Python 3: Sim
>>> print("Hell
Hello, I'm Pyth

# Input, assign
>>> name = inpu
>>> print('Hi,
What is your na
Python
Hi, Python.
```

Download for Windows

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

uage can pick up
lean syntax and
ur appetite with

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 4 : Python download

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)
    - 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)

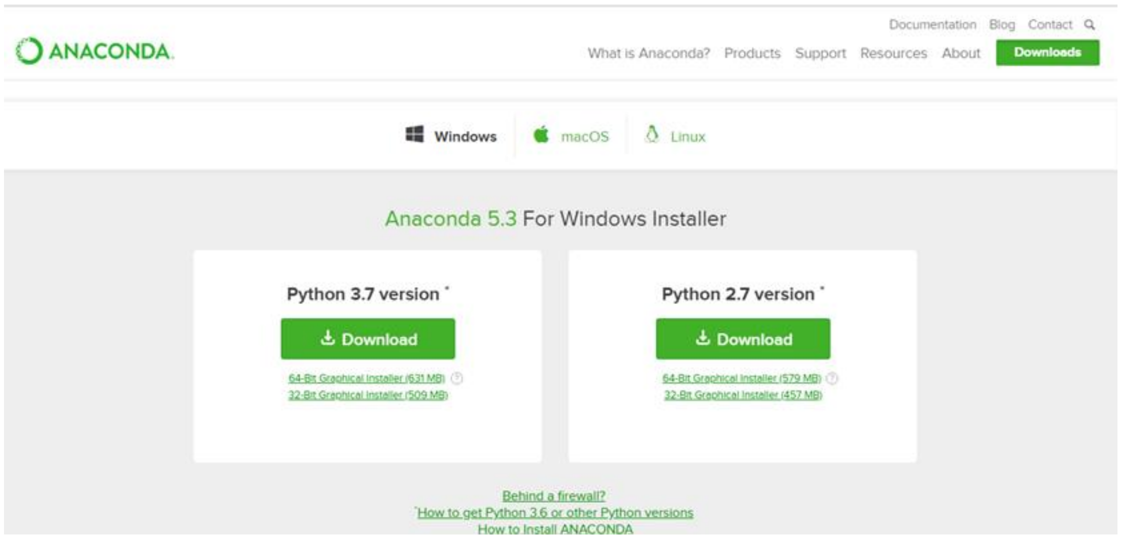

ANACONDA.
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 5 : Anaconda download

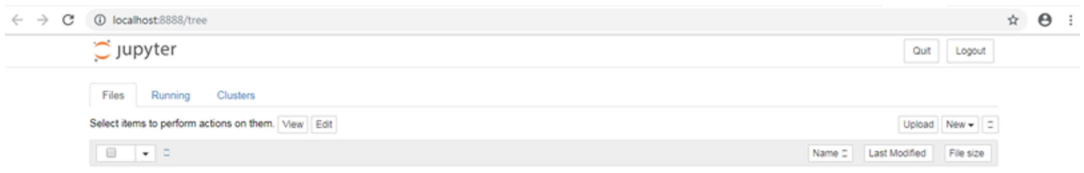

localhost:8888/tree
jupyter
Quit Logout
Files Running Clusters
Select items to perform actions on them. View Edit
Upload New
Name Last Modified File size

Figure 6: Jupiter 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–
    - 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.

2.5.3Python 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 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 appended 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 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:

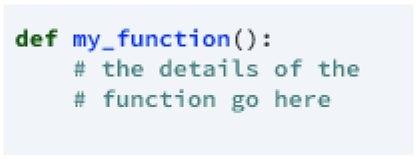
Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOP's CONCEPTS:

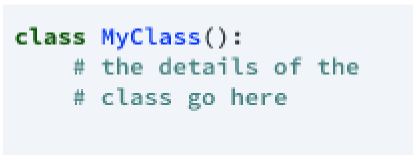
2.7.1 Class:

- 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 witha 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 indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```



```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : 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 under scores: __init__ ().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To predict the amount of purchases in Retail shop using Machine Learning algorithm called MULTIPLE LINEAR REGRESSION

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

M- Color

N- dtype: object

3.3 OBJECTIVE OF THE CASE STUDY:

To get a better understanding and chalking out a plan of action for solution of the client, we have adapted the view point of looking at product categories and for further deep understanding of the problem, we have also considered gender age of the customer and reasoned out the various factors of choice of the products and they purchase , and our primary objective of this case study was to look up the factors which were dampening the sale of products and correlate them to product categories and draft out an outcome report to client regarding the various accepts of a product purchases.

CHAPTER 4
MODEL BUILDING

4.1 PREPROCESSING OF DATA

Preprocessing of the data actually involves the following steps:

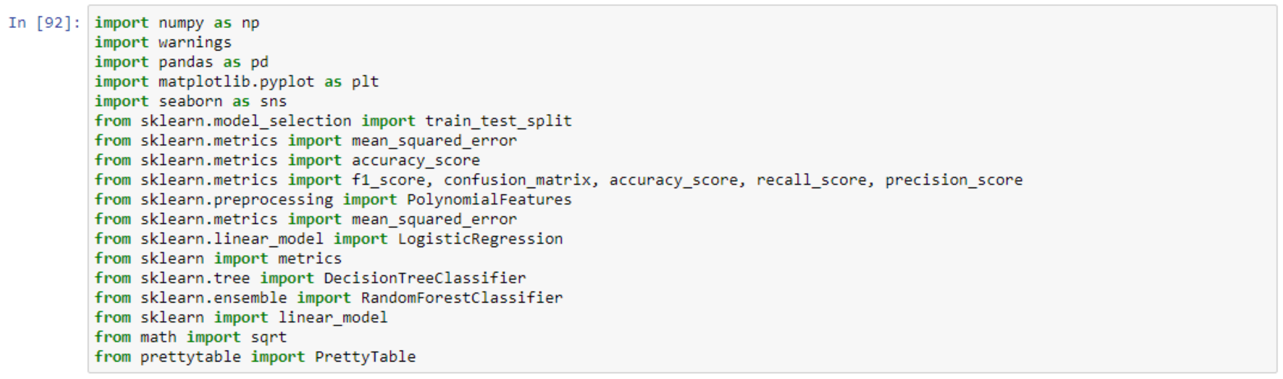
4.1.1: GETTING THE DATASET

We can get the dataset from the database or the client.

4.1.2: IMPORTING THE REQUIRED LIBRARIES

Here we can find the libraries we will use in order to develop a solution for this problem.

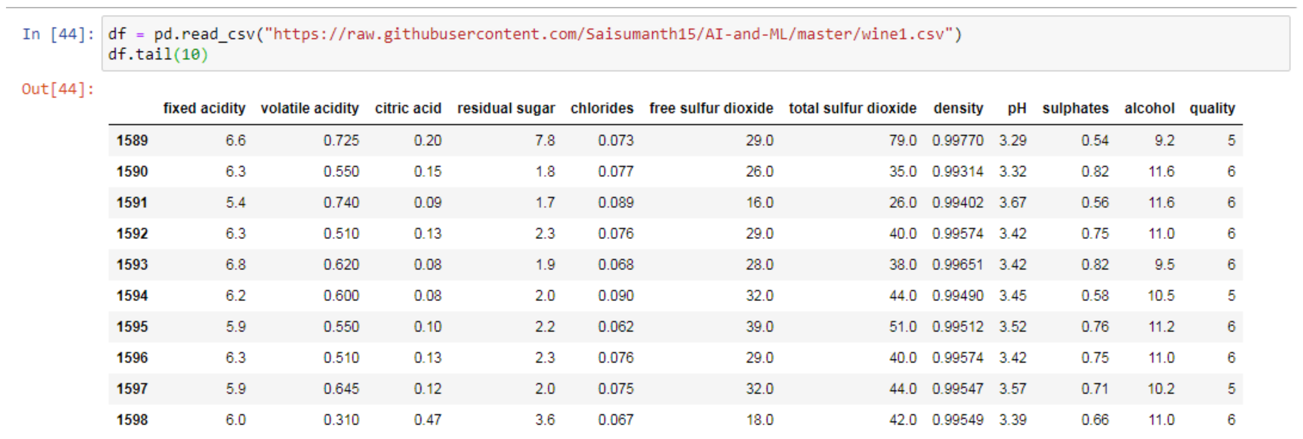
- **Numpy|Pandas**: Will help us treat the data.
- **Matplotlib|seaborn**: Will help us plot the information so we can visualize it in different ways and have a better understanding of it.
- **Sklearn**: Will provide all necessary tools to train our models and test them afterwards.
- **Math**: Will provide some functions we might want to use when testing our models (sqrt).
- **Prettytable**: Will allow us to plot simple ASCII tables.



```
In [92]: import numpy as np
         import warnings
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import f1_score, confusion_matrix, accuracy_score, recall_score, precision_score
         from sklearn.preprocessing import PolynomialFeatures
         from sklearn.metrics import mean_squared_error
         from sklearn.linear_model import LogisticRegression
         from sklearn import metrics
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn import linear_model
         from math import sqrt
         from prettytable import PrettyTable
```

4.1.3: DATA CLEANING AND DATA CHECKING

We load the .csv and visualize the last ten rows of it; we can also see the columns name.

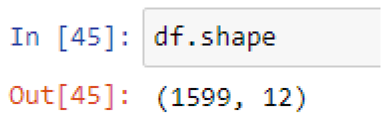


```
In [44]: df = pd.read_csv("https://raw.githubusercontent.com/Saisumanth15/AI-and-ML/master/wine1.csv")
         df.tail(10)
```

Out[44]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1589	6.6	0.725	0.20	7.8	0.073	29.0	79.0	0.99770	3.29	0.54	9.2	5
1590	6.3	0.550	0.15	1.8	0.077	26.0	35.0	0.99314	3.32	0.82	11.6	6
1591	5.4	0.740	0.09	1.7	0.089	16.0	26.0	0.99402	3.67	0.56	11.6	6
1592	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1593	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

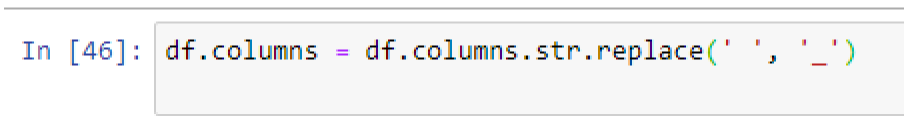
Now let's check the dataset shape so we can see the number of rows and columns.



```
In [45]: df.shape
Out[45]: (1599, 12)
```

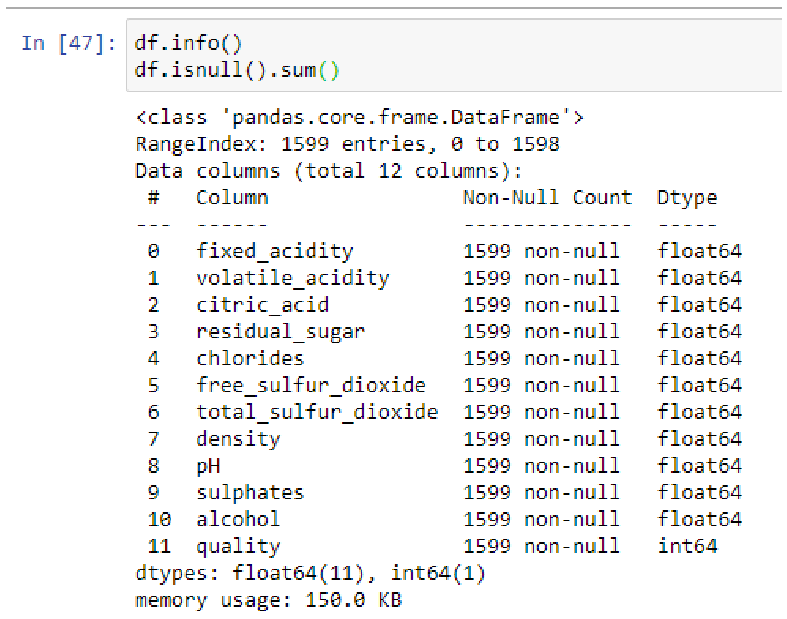
As we can see, the name of some columns contains spaces, which is something we do not really want when treating data; this is why we are going to replace these spaces with "_".

Replace spaces with _ for each column.



```
In [46]: df.columns = df.columns.str.replace(' ', '_')
```

We check there are no missing values and no object data types.



```
In [47]: df.info()
         df.isnull().sum()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed_acidity         1599 non-null   float64
 1   volatile_acidity      1599 non-null   float64
 2   citric_acid           1599 non-null   float64
 3   residual_sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free_sulfur_dioxide   1599 non-null   float64
 6   total_sulfur_dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

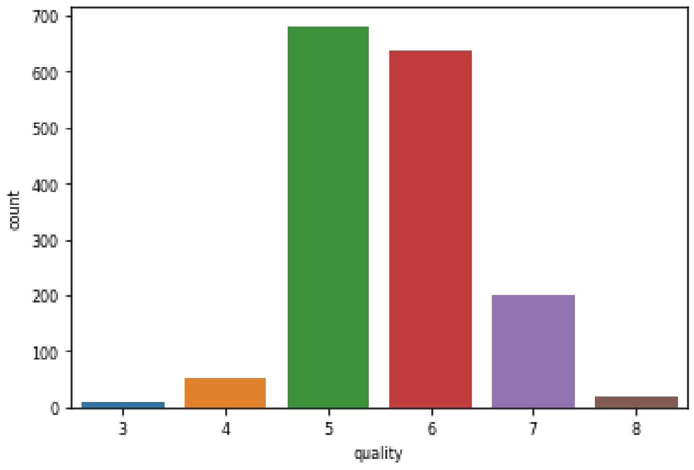
```
Out[47]: fixed_acidity          0
         volatile_acidity       0
         citric_acid            0
         residual_sugar         0
         chlorides              0
         free_sulfur_dioxide    0
         total_sulfur_dioxide   0
         density                0
         pH                     0
         sulphates              0
         alcohol                0
         quality                0
         dtype: int64
```

Fortunately for us, we find there are no null values and all data types seem to be right, since there are no object data types that must be converted to numerical values.

4.1.4: DATA EXPLORATION:

- After checking our dataset is fine and "ready to go" we are going to explore the data a little bit more, we are going to plot important information that will help us check how features behave and how they are correlated. We will also try to extract as much information as we can from it to help us understand the dataset better.
- Knowing our target variable is "quality", we are now going to plot some information about it. Let's see which values this column contains and how many of them there are.
- Let’s start visualizing the different quality values and how many wines have that rating in our dataset.

```
In [48]: sns.countplot(df['quality'])
         df['quality'].value_counts()

Out[48]: 5    681
         6    638
         7    199
         4     53
         8     18
         3     10
         Name: quality, dtype: int64
```



- Now that we got information about our target variable we are going to study the correlation between quality and other features and see which are the ones that play an important role in deciding the quality of a wine.

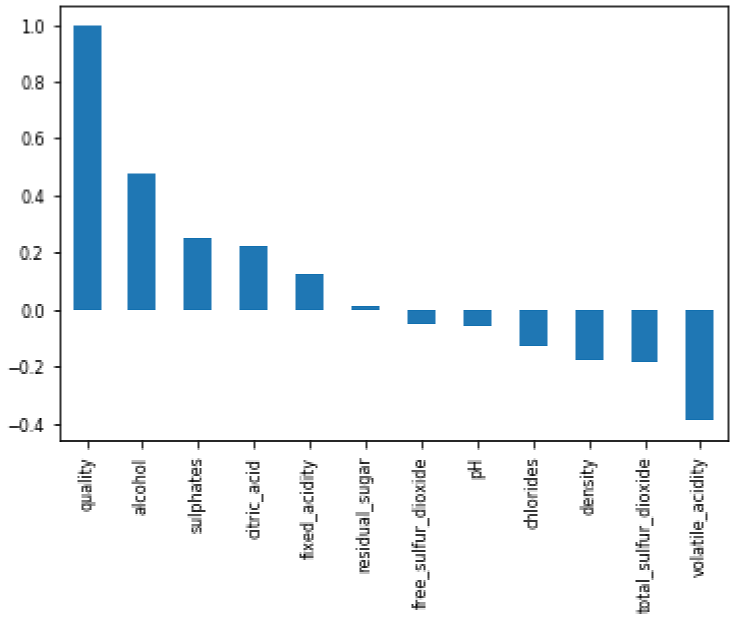
Calculate and order correlations

```
In [49]: correlations = df.corr()['quality'].sort_values(ascending=False)
         print(correlations)
```

```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric_acid             0.226373
fixed_acidity           0.124052
residual_sugar          0.013732
free_sulfur_dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total_sulfur_dioxide   -0.185100
volatile_acidity       -0.390558
Name: quality, dtype: float64
```

```
In [50]: correlations.plot(kind='bar')
```

```
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0xfcc8e08>
```

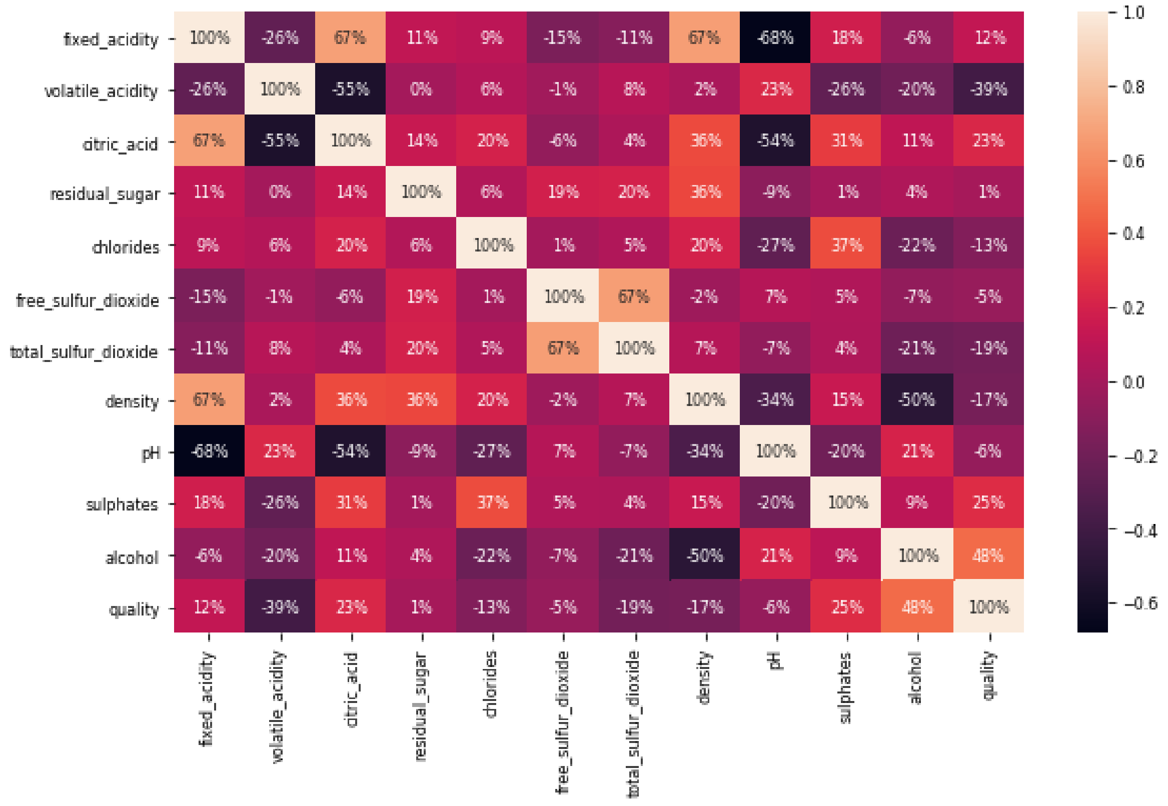


- Having now the correlation values between quality and the other features, let's have a look at the correlation matrix to have a better understanding of how features correlate with each other.

Heatmap to plot all correlations between features

```
In [51]: plt.figure(figsize=(10,6))
         sns.heatmap(df.corr(), annot=True, fmt='.0%')
```

```
Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0xfd57308>
```



- From this matrix we can observe, apart from the information we had before, some obvious feature correlations such as pH and acidity. Apart from that, we get to know the percentage of the correlations we obtained before. We can also observe that approximately half of these features correlate positively with quality while the other half correlate negatively.

- From all these features, we are going to select the ones with bigger numbers since these are the ones that will give us more information. To do so we are going to establish a minimum threshold of correlation approximately around 0.2 (absolute value) since we do not have to take into account features whose values might be redundant and not provide information at all.

```
In [52]: print(abs(correlations) > 0.2)
```

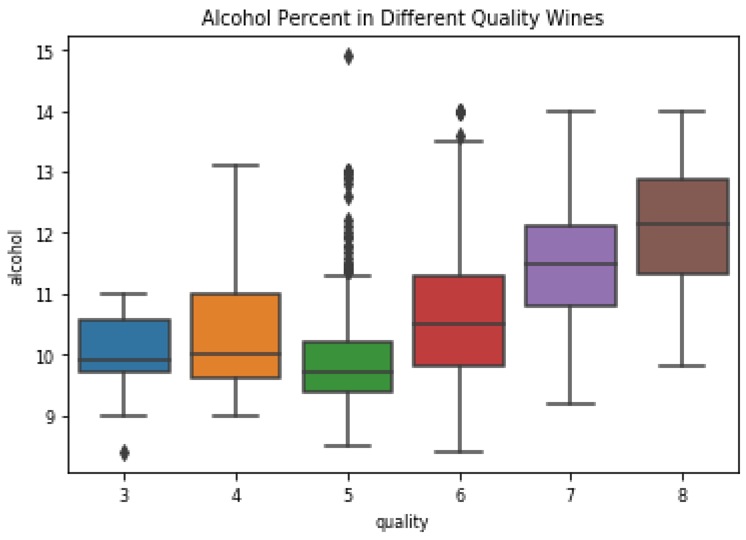
```
quality                  True
alcohol                  True
sulphates                True
citric_acid              True
fixed_acidity           False
residual_sugar          False
free_sulfur_dioxide     False
pH                      False
chlorides               False
density                 False
total_sulfur_dioxide    False
volatile_acidity         True
Name: quality, dtype: bool
```

- From all the values, we are selecting alcohol, sulphates, citric acid and volatile acidity in order to study them better and see the distribution of values that separate the different qualities.

Alcohol Percent

```
In [53]: bp = sns.boxplot(x='quality',y='alcohol', data=df)
         bp.set(title="Alcohol Percent in Different Quality Wines")
```

```
Out[53]: [Text(0.5, 1.0, 'Alcohol Percent in Different Quality Wines')]
```



- On this box plot we can see how wines that contain less alcohol seem to be getting worse ratings while the ones with higher alcohol % are getting better quality ratings. However, we can observe how "mid quality" wines with rating 5 or 6 are presenting some strange values. Let's do some further investigation now:
- In order to see why we do have these different values, we are going to select a subset of the dataset which will only contain rows whose quality column value is 5 or 6, and we will calculate correlation coefficients for this subset.

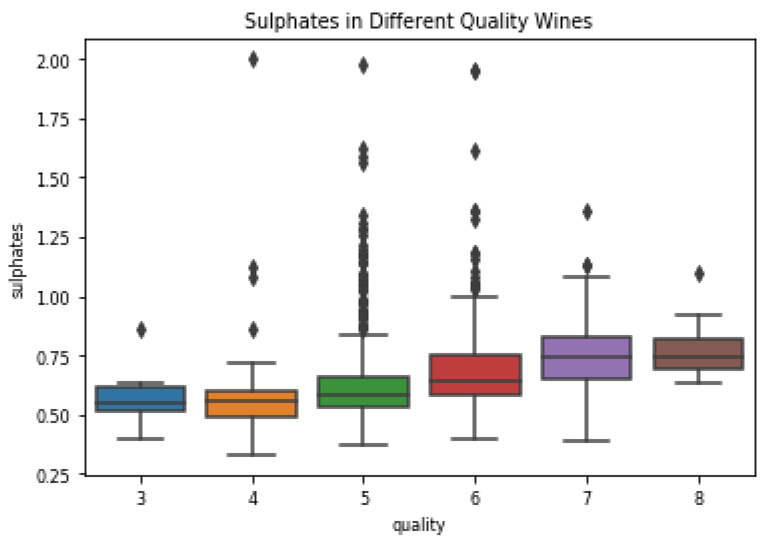
```
In [54]: df_quality_five_six = df.loc[(df['quality'] >= 5) & (df['quality'] <= 6)]
         df_quality_five_six['quality'].value_counts()

Out[54]: 5    681
         6    638
         Name: quality, dtype: int64
```

```
In [55]: correlations_subset = df_quality_five_six.corr()['quality'].sort_values(ascending=False)
         print(correlations_subset)

quality                 1.000000
alcohol                 0.375224
sulphates               0.162405
citric_acid             0.080146
fixed_acidity           0.053447
pH                      0.043065
residual_sugar         -0.018452
free_sulfur_dioxide    -0.060618
chlorides              -0.081813
density                -0.134559
volatile_acidity       -0.237193
total_sulfur_dioxide   -0.239067
Name: quality, dtype: float64
```

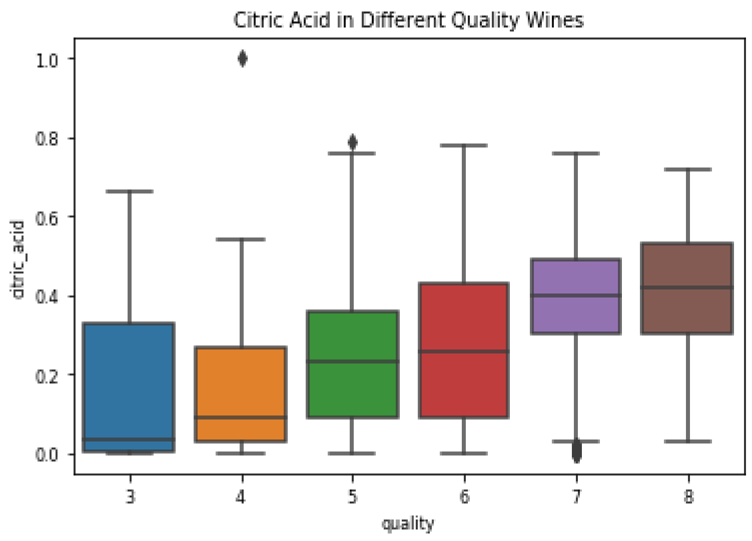
- After calculating the correlations for wines with quality of 5 and 6 we find, that features that correlate the most with quality are the same as we obtained before. However, the difference of values shown on the box plot can be explained due to having (although the highest) a poor correlation with quality.

Sulphates and Citric Acid Presence

```
In [56]: bp = sns.boxplot(x='quality',y='sulphates', data=df)
         bp.set(title="Sulphates in Different Quality Wines")

Out[56]: [Text(0.5, 1.0, 'Sulphates in Different Quality Wines')]
```



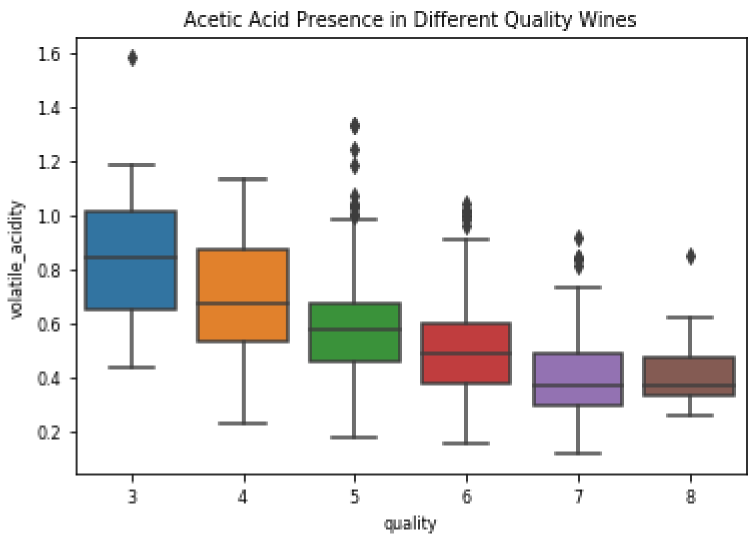
```
In [57]: bp = sns.boxplot(x='quality',y='citric_acid', data=df)
         bp.set(title="Citric Acid in Different Quality Wines")

Out[57]: [Text(0.5, 1.0, 'Citric Acid in Different Quality Wines')]
```



- In the case of the features "sulphates" and "citric acid" the relation between them and quality seem to be linearly positive, showing our correlation coefficients calculated before were right: adding higher amounts of sulphates and citric acid to these wines seem to get them higher quality ratings.

Acetic Acid Presence

```
In [58]: bp = sns.boxplot(x='quality',y='volatile_acidity', data=df)
         bp.set(title="Acetic Acid Presence in Different Quality Wines")

Out[58]: [Text(0.5, 1.0, 'Acetic Acid Presence in Different Quality Wines')]
```



- For the acetic acid presence, we can clearly observe how less acetic acid presence in wine seems positive while having higher values contribute to having a lower rating.

More data visualization

In the last subsection we plotted some box graphics in order to see which values belong to each quality. For further investigations we are now going to plot histograms for each of those important features so we can see better the correlation between the distribution of values from each feature and quality. To do so, we are first going to separate the quality values in three different groups, so we can do things a little bit easier:

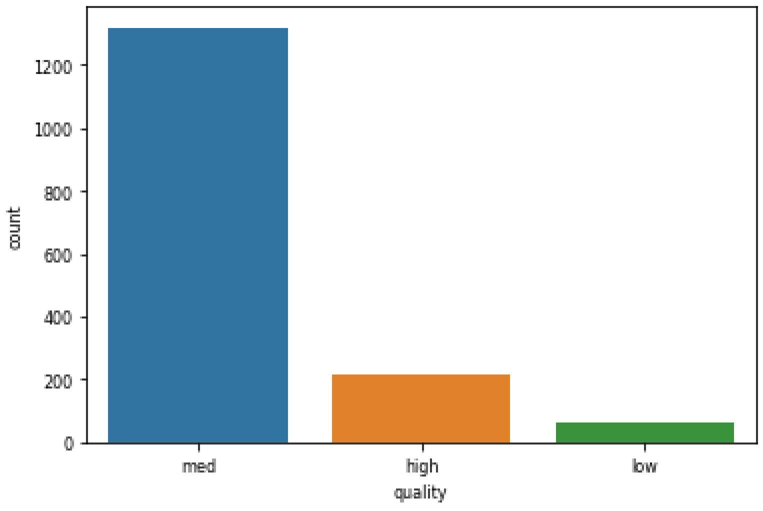
1. **Low**: contains wines whose quality is 3 or 4.
2. **Medium**: contains wines whose quality is 5 or 6.
3. **High**: contains wines whose quality is 7 or 8.

We make a copy of our data frame and group quality in different groups

```
In [59]: df_aux = df.copy()
         df_aux['quality'].replace([3,4],['low','low'],inplace=True)
         df_aux['quality'].replace([5,6],['med','med'],inplace=True)
         df_aux['quality'].replace([7,8],['high','high'],inplace=True)
```

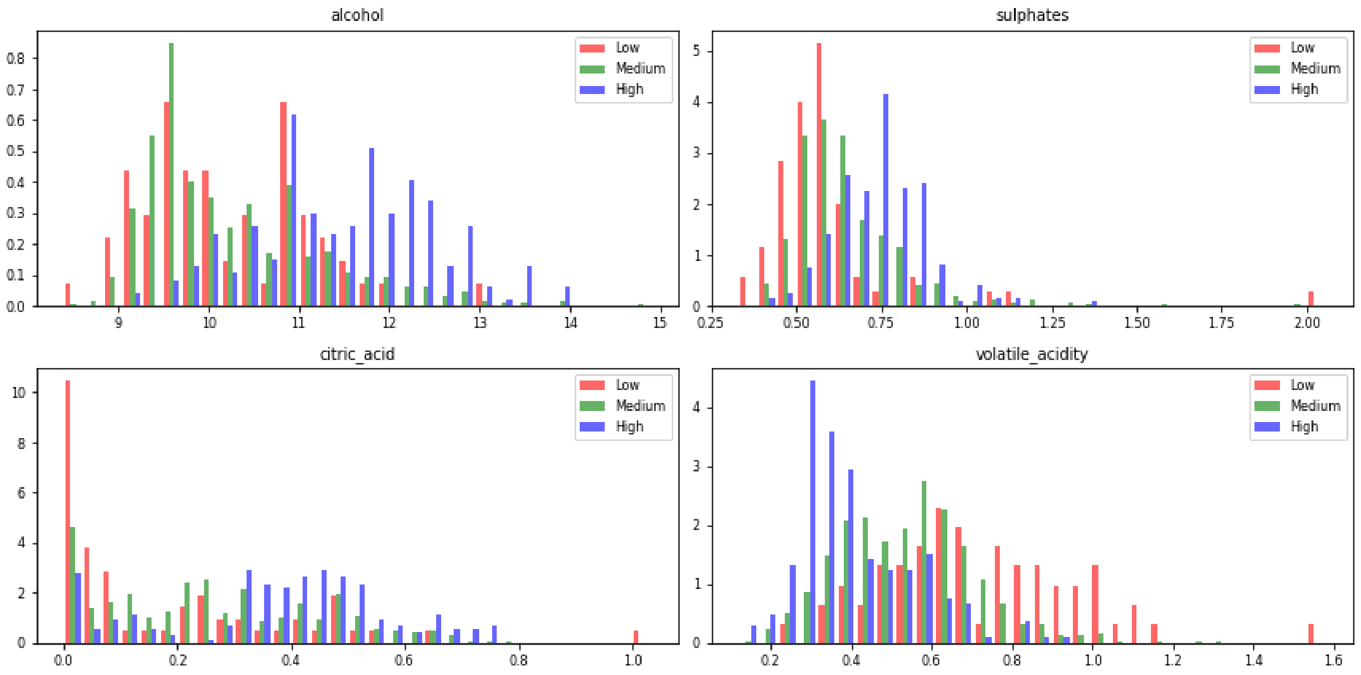
```
In [60]: sns.countplot(df_aux['quality'])
```

```
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0xcca8e88>
```



We plot some histograms that show the values of features selected

```
In [61]: flistt = ['alcohol','sulphates','citric_acid','volatile_acidity']
         low = df_aux[df_aux['quality'] == 'low']
         medium = df_aux[df_aux['quality'] == 'med']
         high = df_aux[df_aux['quality'] == 'high']
         plt.rcParams.update({'font.size': 8})
         plot, graphs = plt.subplots(nrows= 2, ncols= 2, figsize=(12,6))
         graphs = graphs.flatten()
         for i, graph in enumerate(graphs):
             graph.figure
             binwidth= (max(df_aux[flistt[i]]) - min(df_aux[flistt[i]]))/30
             bins = np.arange(min(df[flistt[i]]), max(df_aux[flistt[i]]) + binwidth, binwidth)
             graph.hist([low[flistt[i]],medium[flistt[i]],high[flistt[i]]], bins=bins, alpha=0.6, normed=True, label=['Low','Medium','Hig
             graph.legend(loc='upper right')
             graph.set_title(flistt[i])
         plt.tight_layout()
```



- As we can see in the histograms, higher values of alcohol, sulphates and citric acid seem to belong to higher quality wines while higher values of volatile acity are present in lower quality wines.

4.2: FEATURE SELECTION

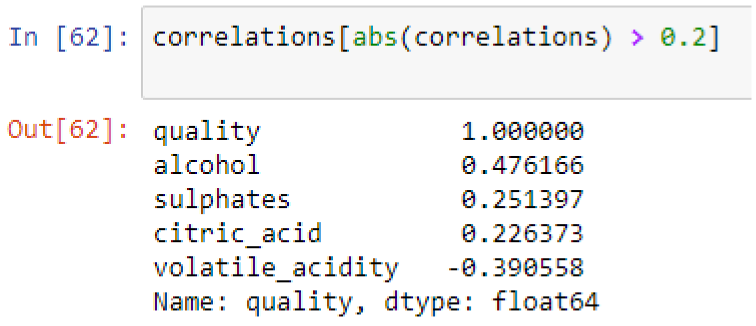
Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Feature selection and Data cleaning should be the first and most important step of your model designing.

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

How to select features and what are Benefits of performing feature selection before modeling your data?

- **Reduces Over fitting**: Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy:** Less misleading data means modeling accuracy improves.
- **Reduces Training Time:** fewer data points reduce algorithm complexity and algorithms train faster.

Now that we have already studied our dataset through histograms and different graphics it's time to select some features we will use in our machine learning algorithms. In this specific case, what we are going to do is use the same columns we studied before, since those are the four ones that give us the most information between features and quality.



```
In [62]: correlations[abs(correlations) > 0.2]

Out[62]: quality             1.000000
         alcohol             0.476166
         sulphates           0.251397
         citric_acid         0.226373
         volatile_acidity   -0.390558
         Name: quality, dtype: float64
```

CHAPTER 5

CLASSIFIER MODELS

A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset. For example, when filtering emails “spam” or “not spam” (also known as “ham”, < seriously, look it up if you don’t believe me), when looking at transaction data, “fraudulent”, or “authorized”.

There are two approaches to machine learning: supervised and unsupervised. In a supervised model, a training dataset is fed into the classification algorithm. That lets the model know what “authorized” transactions is, for example,. Then the test data sample is compared with that to determine if there is a “fraudulent” transaction. This type of learning falls under “Classification”.

Unsupervised models on the other hand, are fed a dataset that is not labeled and looks for clusters of data points. It can be used to search data for similarities, detect patterns, or identify outliers within a dataset. A typical use case would be finding similar images. Unsupervised models can also be used to find “fraudulent” transactions by looking for anomalies within a dataset. This type of learning falls under “Clustering”

Anti-spam uses the Naive Bayes classification algorithm. As people get junk mail, when they mark it as spam the words in that email get put into a database called spam. Good mail goes into the ham (aka not spam) database. Over time the list of spam words and phrases gets built up. Then the anti-spam algorithm can calculate the probability of an email being spam or not spam and make its determination based on that.

There are a number of classification models. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs.-rest, and Naive Bayes.

Classification is a form of data analysis that extracts models describing data classes. A classifier, or classification model, predicts categorical labels (classes). Numeric prediction models continuous-valued functions. Classification and numeric prediction are the two major types of prediction problems.

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails have to be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email.

Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

5.1 LOGISTIC REGRESSION

Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types –

1. **Binary and Binomial**
   - In such a kind of classification, a dependent variable will have only two possible types either 1 or 0. For example, these variables may represent success or failure, yes or no, win or loss etc.
2. **Multinomial**
   - In such a kind of classification, dependent variable can have 3 or more possible **unordered** types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.
3. **Ordinal**
   - In such a kind of classification, dependent variable can have 3 or more possible **ordered** types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

LOGISTIC REGRESSION ASSUMPTIONS

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same –

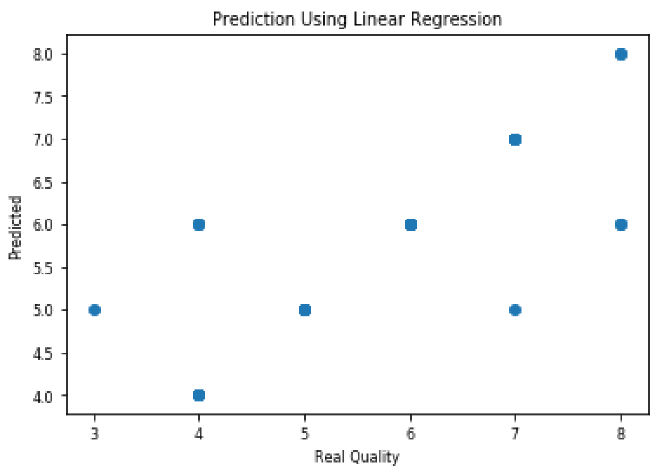
- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-co linearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

Fit the model and make prediction

```
In [64]: logreg = LogisticRegression()
         logreg.fit(X_train, y_train)

         E:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

Out[64]: LogisticRegression()
```

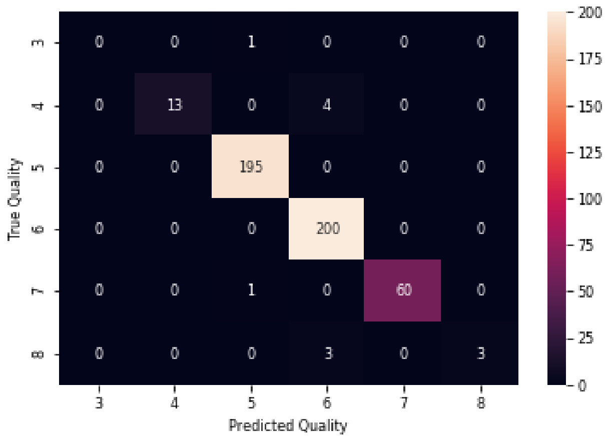
- Let's plot a scatter plot for the above model.

```
In [65]: plt.scatter(y_test,y_prediction_lr)
         plt.title("Prediction Using Linear Regression")
         plt.xlabel("Real Quality")
         plt.ylabel("Predicted")
         plt.show()
```



- **Confusion matrix for the Logistic Regression Model**

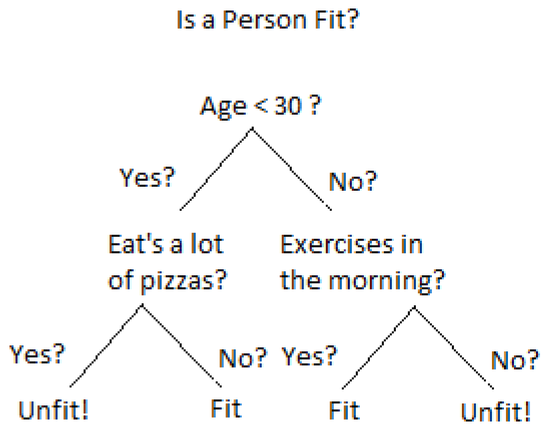
```
In [89]: label_aux = plt.subplot()
         cm_logistic_regression = confusion_matrix(y_test,y_prediction_lr)
         cm_lr = pd.DataFrame(cm_logistic_regression,
                             index = ['3','4','5','6','7','8'],
                             columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_lr,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	13	0	4	0	0
5	0	0	195	0	0	0
6	0	0	0	200	0	0
7	0	0	1	0	60	0
8	0	0	0	3	0	3

5.2 DECISION TREE CLASSIFIER

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.



An example of a decision tree can be explained using above binary tree. Let’s say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like ‘What’s the age?’, ‘Does he exercise?’, and ‘Does he eat a lot of pizzas’? And the leaves, which are outcomes like either ‘fit’, or ‘unfit’. In this case this was a binary classification problem (a yes no type problem).

Decision Tree Analysis is a general, predictive modeling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

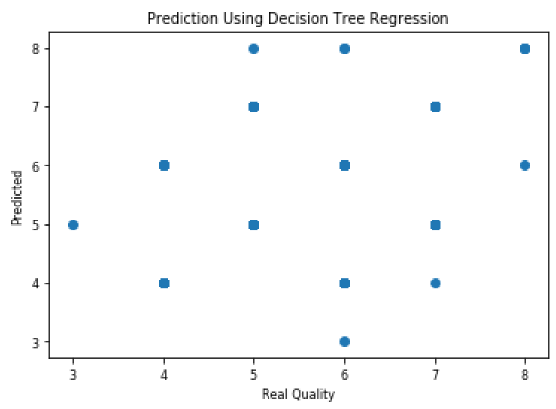
Let's illustrate this with help of an example. Let's assume we want to play badminton on a particular day — say Saturday — how will you decide whether to play or not. Let's say you go out and check if it's hot or cold, check the speed of the wind and humidity, how the weather is, i.e. is it sunny, cloudy, or rainy. You take all these factors into account to decide if you want to play or not.

Fit the model and make prediction

```
In [67]: clf = DecisionTreeClassifier()
         clf = clf.fit(X_train,y_train)
         y_pred = clf.predict(X_test)
```

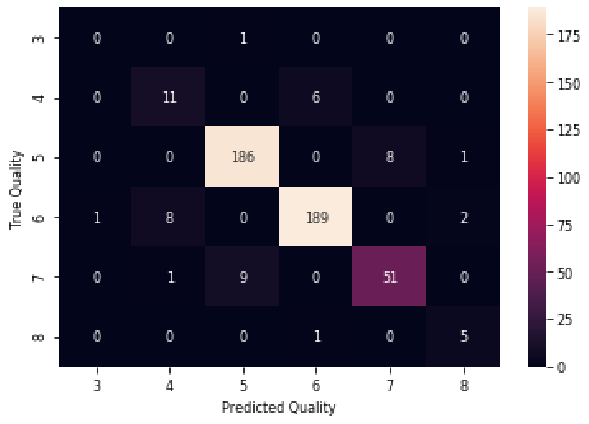
- Let’s plot a scatter plot for the above model.

```
In [68]: plt.scatter(y_test,y_prediction_dt)
         plt.title("Prediction Using Decision Tree Regression")
         plt.xlabel("Real Quality")
         plt.ylabel("Predicted")
         plt.show()
```



- **Confusion matrix for the Decision Tree Classifier Model**

```
In [88]: label_aux = plt.subplot()
         cm_decision_tree_classifier = confusion_matrix(y_test,y_prediction_dt)
         cm_dt = pd.DataFrame(cm_decision_tree_classifier,
                              index = ['3','4','5','6','7','8'],
                              columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_dt,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



True Quality \ Predicted Quality	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	11	0	6	0	0
5	0	0	186	0	8	1
6	1	8	0	189	0	2
7	0	1	9	0	51	0
8	0	0	0	1	0	5

5.3 RANDOM FOREST CLASSIFIER

This approach to classification is similar to the decision tree, except the questions that are posed include some randomness. The goal is to push out bias and group outcomes based upon the most likely positive responses. These collections of positive responses are called bags.

An example application of the Random Forest model is the algorithm Netflix uses to recommend movies. It looks at people who have similar tastes and then recommends movies that way. It tosses out outlier answers by using randomness to avoid skewing the response in an incorrect direction.

The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees (which we could call a “forest”), this model uses two key concept that gives it the name random:

Random sampling of training observations

When training, each tree in a random forest learns from a random sample of the data points. The samples are drawn with replacement, known as bootstrapping, which means that some samples will be used multiple times in a single tree. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias.

At test time, predictions are made by averaging the predictions of each decision tree. This procedure of training each individual learner on different bootstrapped subsets of the data and then averaging the predictions is known as bagging, short for bootstrap aggregating.

Random Subsets of features for splitting nodes

The other main concept in the random forest is that only a subset of all the features are considered for splitting each node in each decision tree. Generally this is set to sqrt (n_features) for classification meaning that if there are 16 features, at each node in each tree, only 4 random features will be considered for splitting the node. (The random forest can also be trained considering all the features at every node as is common in regression. These options can be controlled in the Scikit-Learn Random Forest implementation).

If you can comprehend a single decision tree, the idea of bagging, and random subsets of features, then you have a pretty good understanding of how a random forest works:

The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

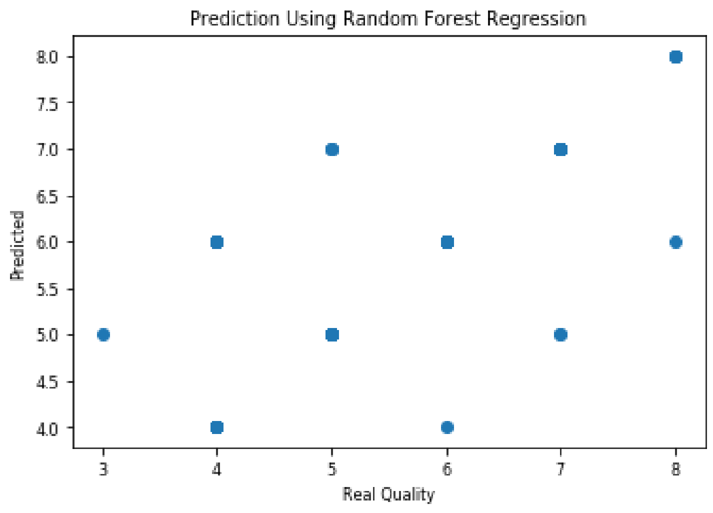
To understand why a random forest is better than a single decision tree imagine the following scenario: you have to decide whether Tesla stock will go up and you have access to dozen analysts who have no prior knowledge about the company. Each analyst has low bias because they don’t come in with any assumptions, and is allowed to learn from a dataset of news reports.

Fit the model and make prediction

```
In [70]: clf=RandomForestClassifier(n_estimators=100)
         clf.fit(X_train,y_train)
         y_pred=clf.predict(X_test)
```

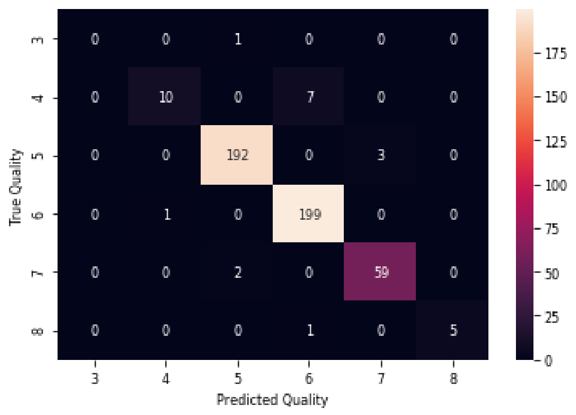
- Let’s plot a scatter plot for the above model.

```
In [71]: plt.scatter(y_test,y_prediction_rf)
         plt.title("Prediction Using Random Forest Regression")
         plt.xlabel("Real Quality")
         plt.ylabel("Predicted")
         plt.show()
```



- **Confusion matrix for the Random Forest Classifier Model**

```
In [87]: label_aux = plt.subplot()
         cm_random_forest_classifier = confusion_matrix(y_test,y_prediction_rf)
         cm_rf = pd.DataFrame(cm_random_forest_classifier,
                              index = ['3','4','5','6','7','8'],
                              columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_rf,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



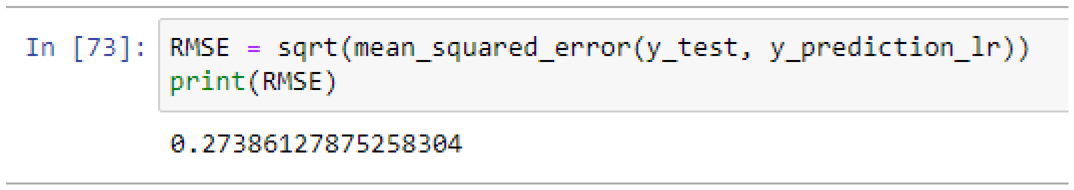
	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	10	0	7	0	0
5	0	0	192	0	3	0
6	0	1	0	199	0	0
7	0	0	2	0	59	0
8	0	0	0	1	0	5

5.4 Root Mean Square Deviation (RMSD) or Root Mean Square Error (RMSE) of the Models

The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. ... RMSD is the square root of the average of squared errors.

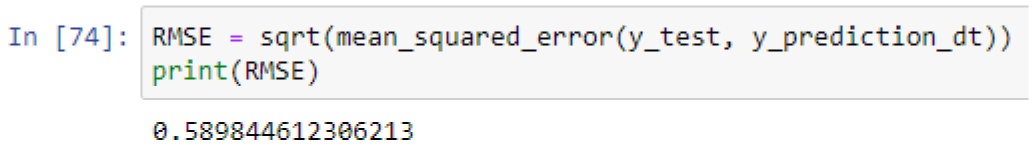
After having prepared our models, it's now time to evaluate them. To do so we are going to use RMSE (Root Mean Square Error) this is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are so RMSE is a measure of how spread out these residuals are.

RMSD or RMSE for Logistic Regression



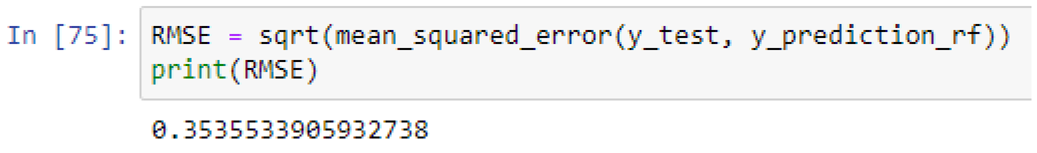
```
In [73]: RMSE = sqrt(mean_squared_error(y_test, y_prediction_lr))
         print(RMSE)

         0.27386127875258304
```

RMSD or RMSE for Decision Tree Classifier



```
In [74]: RMSE = sqrt(mean_squared_error(y_test, y_prediction_dt))
         print(RMSE)

         0.589844612306213
```

RMSD or RMSE for Random Forest Classifier



```
In [75]: RMSE = sqrt(mean_squared_error(y_test, y_prediction_rf))
         print(RMSE)

         0.3535533905932738
```

When deciding which classifier algorithm is better by looking at RMSE we would better chooses the one with smaller value, so for this problem, Logistic Regression seems to be the best fitting algorithm.

5.5: IMPROVING THE RESULTS WITH 1-OFF ACCURACY

As we can see from the confusion matrices we show above our predictions aren't bad at all but in order to "improve" them we are going to apply a concept called 1-off accuracy, which states that if the distance between our predicted quality and the true quality is 1 (in absolut value), we will accept it as a correct prediction.

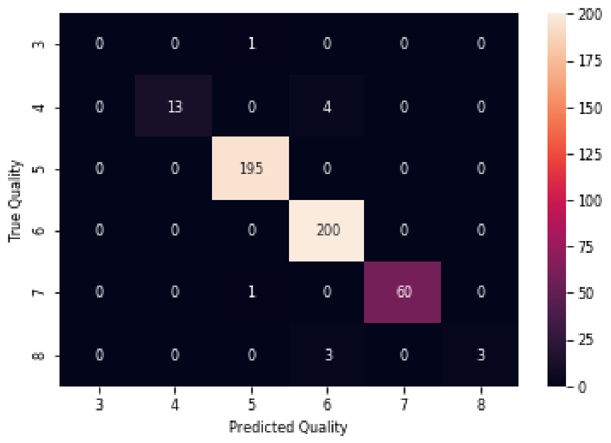
We will now create a function that will transform our predicted value into the true value if the distance between them is equal to 1. Afterwards we are going to plot the new correlation matrices and test the new values with some metrics.

```
In [76]: def one_accuracy(predicted, true):
             i = 0
             for x,y in zip(predicted,true):
                 if(abs(x-y)==1):
                     predicted[i] = y
                 i = i + 1

         one_accuracy(y_prediction_lr, y_test)
         one_accuracy(y_prediction_dt, y_test)
         one_accuracy(y_prediction_rf, y_test)
```

- Let’s check the improved RMSD and RMSE of the models starting with the confusion matrix.

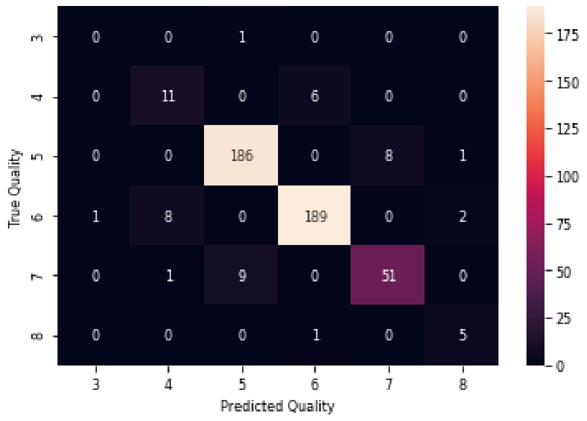
Confusion matrix for the Logistic Regression Model

```
In [89]: label_aux = plt.subplot()
         cm_logistic_regression = confusion_matrix(y_test,y_prediction_lr)
         cm_lr = pd.DataFrame(cm_logistic_regression,
                              index = ['3','4','5','6','7','8'],
                              columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_lr,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



Confusion matrix for the Decision Tree Classifier Model

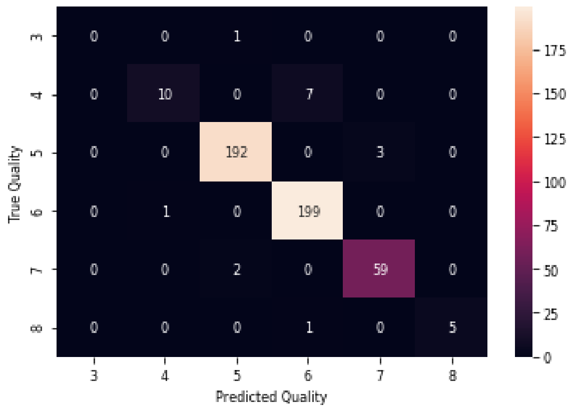
```
In [88]: label_aux = plt.subplot()
         cm_decision_tree_classifier = confusion_matrix(y_test,y_prediction_dt)
         cm_dt = pd.DataFrame(cm_decision_tree_classifier,
                              index = ['3','4','5','6','7','8'],
                              columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_dt,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	11	0	6	0	0
5	0	0	186	0	8	1
6	1	8	0	189	0	2
7	0	1	9	0	51	0
8	0	0	0	1	0	5

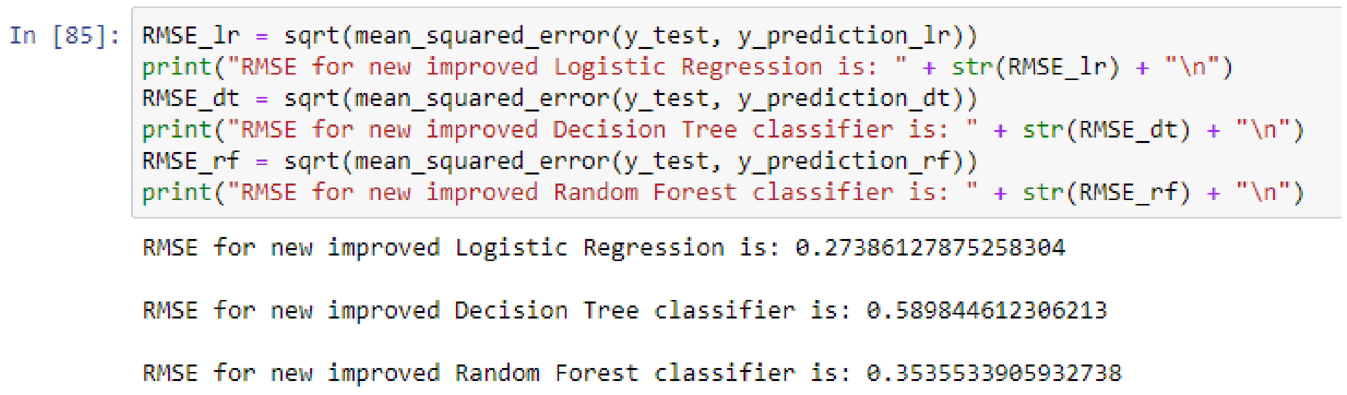
Confusion matrix for the Random Forest Classifier Model

```
In [87]: label_aux = plt.subplot()
         cm_random_forest_classifier = confusion_matrix(y_test,y_prediction_rf)
         cm_rf = pd.DataFrame(cm_random_forest_classifier,
                              index = ['3','4','5','6','7','8'],
                              columns = ['3','4','5','6','7','8'])
         sns.heatmap(cm_rf,annot=True,fmt="d")
         label_aux.set_xlabel('Predicted Quality');label_aux.set_ylabel('True Quality');
```



	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	10	0	7	0	0
5	0	0	192	0	3	0
6	0	1	0	199	0	0
7	0	0	2	0	59	0
8	0	0	0	1	0	5

As we can see, our results are far better than the ones we obtained before. Therefore we are now going to calculate the new RMSE for all the three models.



```
In [85]: RMSE_lr = sqrt(mean_squared_error(y_test, y_prediction_lr))
         print("RMSE for new improved Logistic Regression is: " + str(RMSE_lr) + "\n")
         RMSE_dt = sqrt(mean_squared_error(y_test, y_prediction_dt))
         print("RMSE for new improved Decision Tree classifier is: " + str(RMSE_dt) + "\n")
         RMSE_rf = sqrt(mean_squared_error(y_test, y_prediction_rf))
         print("RMSE for new improved Random Forest classifier is: " + str(RMSE_rf) + "\n")

         RMSE for new improved Logistic Regression is: 0.27386127875258304

         RMSE for new improved Decision Tree classifier is: 0.589844612306213

         RMSE for new improved Random Forest classifier is: 0.3535533905932738
```

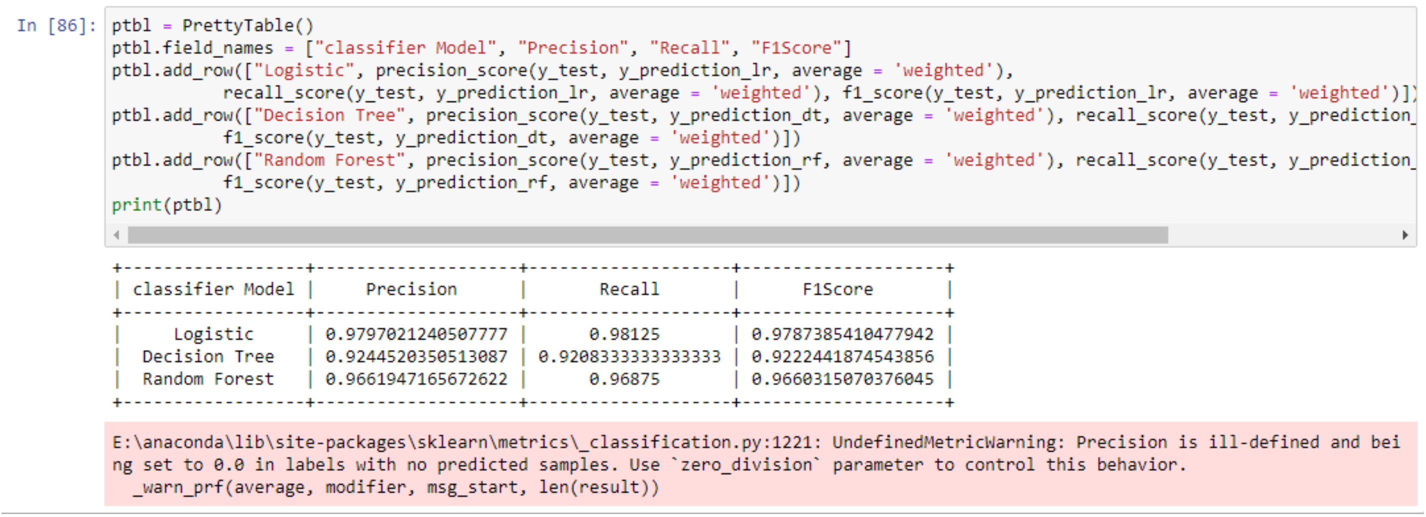
Improved RMSE for Logistic Regression: 0.27386127875258304

Improved RMSE for Decision Tree Classifier: 0.589844612306213

Improved RMSE for Random Forest Classifier: 0.3535533905932738

5.6 Testing Our Models with Precision, Recall and F1 Score

For further testing, we are plotting a table that shows the precision, recall and f1score of our three classifier models.



```
In [86]: ptbl = PrettyTable()
         ptbl.field_names = ["classifier Model", "Precision", "Recall", "F1Score"]
         ptbl.add_row(["Logistic", precision_score(y_test, y_prediction_lr, average = 'weighted'),
                       recall_score(y_test, y_prediction_lr, average = 'weighted'), f1_score(y_test, y_prediction_lr, average = 'weighted')]
         ptbl.add_row(["Decision Tree", precision_score(y_test, y_prediction_dt, average = 'weighted'), recall_score(y_test, y_prediction_
                       f1_score(y_test, y_prediction_dt, average = 'weighted')])
         ptbl.add_row(["Random Forest", precision_score(y_test, y_prediction_rf, average = 'weighted'), recall_score(y_test, y_prediction_
                       f1_score(y_test, y_prediction_rf, average = 'weighted')])
         print(ptbl)
```

```
+------------------+--------------------+--------------------+--------------------+
| classifier Model |     Precision      |       Recall       |      F1Score       |
+------------------+--------------------+--------------------+--------------------+
|     Logistic     | 0.9797021240507777 |      0.98125       | 0.9787385410477942 |
|  Decision Tree   | 0.9244520350513087 | 0.9208333333333333 | 0.9222441874543856 |
|  Random Forest   | 0.9661947165672622 |      0.96875       | 0.9660315070376045 |
+------------------+--------------------+--------------------+--------------------+
```

```
E:\anaconda\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision is ill-defined and bei
ng set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

- Here we got the Precision, Recall, and F1Score of all the three models.
- As we can see the different scores of the models and classifiers we can conclude by saying Random Forest Classifier and Logistic Regression seem to be the best fitting models when solving this problem using classifier models.

CHAPTER 6
CONCLUSION

After having obtained all the results through our models and plots, these are some things we can say about this problem and solution:

- The vast majority of wines get a quality rating of five or six, while having good and bad wines seems more unlikely. There seem not to be any excellent wines (>8) on this database.
- From the very first moment we saw there weren't strong correlations between features and quality, that's why it's hard to make an accurate prediction using regression algorithms. That said, alcohol, sulphates, citric acid features are the ones that correlate the most positively while volatile acidity is the one correlating the most negatively.
- Applying the concept 1-off Accuracy gives us much better results.
- Random Forest and Linear Regression seem to be the best fitting models when solving this problem using regression.
- Since there are only six different quality values in this dataset, it would be clever treating this problem as a multiclass classification problem (we can use the quality grouping approach we took earlier) and we might even get better results.

It is concluded after performing thorough Exploratory Data analysis which include Stats models which are computed to get accuracy and also Heat maps which are computed to get a clear understanding of the data set (which parameter has most abundant effect on the study case) and its come to point of getting the solution for the problem statement being , that the retail shopkeeper should strategically plan on the marketing in such a way that he could offer complementary products on the purchase of that particular product and if in possibility of bearing expenses offer more quantity of the product at the same price.

CHAPTER 7
REFERENCES

[1] https://www.kaggle.com/semakulapaul/cereals-dataset

[2] https://medium.com/code-heroku/introduction-to-exploratory-data-analysis-eda-c0257f888676

[3] https://en.wikipedia.org/wiki/Machine_learning