MACHINE LEARNING

(WINE QUALITY PREDICITION)

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

of

Bachelor of Technology

In

Computer Science Engineering

By

M.Rangabhagavan reddy

221710310043

Under the Guidance of

Mr. _____

Assistant Professor

Department Of Computer Science Engineering
GITAM School of Technology

GITAM (Deemed to be University)
Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled **"WINE QUALITY PREDICITION"** 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 Engineering"**. I declare that it was carried out independently by me under the guidance of Mr._____, 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.Rangabhagavan Reddy

Date: 221710310043

GITAM UNIVERSITY

GITAM (Deemed To Be University)

Hyderabad

Date:- ……

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"WINE QUALITY PREDICITION"** is being submitted by **M.Rangabhagavan Reddy (221710310043)** in partial fulfilment of the requirement for the award of **Bachelor of Technology** in **Computer Science Engineering at GITAM (Deemed To Be University)**, Hyderabad during the academic year **2019-2020**.

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

Assistant Professor
Department of CSE

Professor and HOD
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. _____Head of the Department of Computer Science 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. _____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.Rangabhagavan Reddy

221710310043

ABSTRACT

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 analyze the impact of the analytical tests. If it is concluded that several input variables are highly relevant to predict the wine quality, since in the production process some variables can be controlled, this information can be used to improve the wine quality.

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

Knn (K – nearest neighbour algorithm), Bernoulli naive bayes algorithm, DTC (Decision tree classifier algorithm), Random forest 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 CONTENT

List of Figures

1.INFORMATION ABOUT 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 Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

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

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

The process flow depicted here represents how machine learning works

Figure 1.2 : 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 MACHINE LEARNING :

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

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when

posed with new examples comes under the category of supervised learning.

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

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

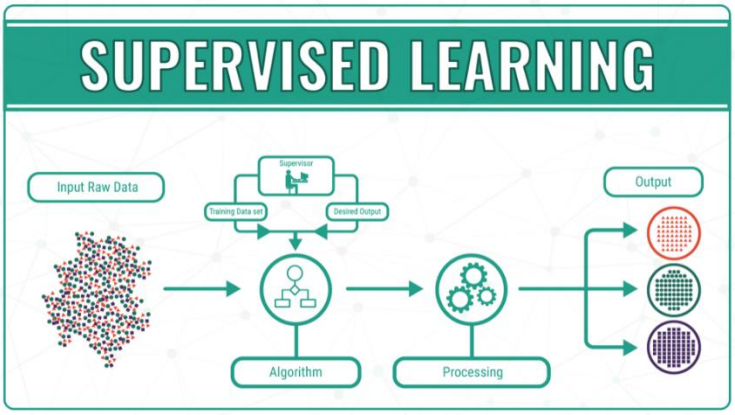


Figure 1.4.1: Supervised Learning

1.4.2 Unsupervised Learning:

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

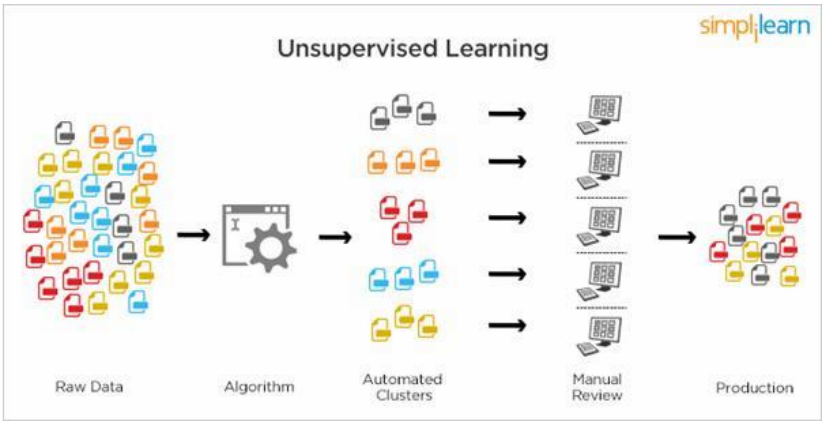


Figure 1.4.2: Unsupervised Learning

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

1.4.3 Semi Supervised Learning:

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

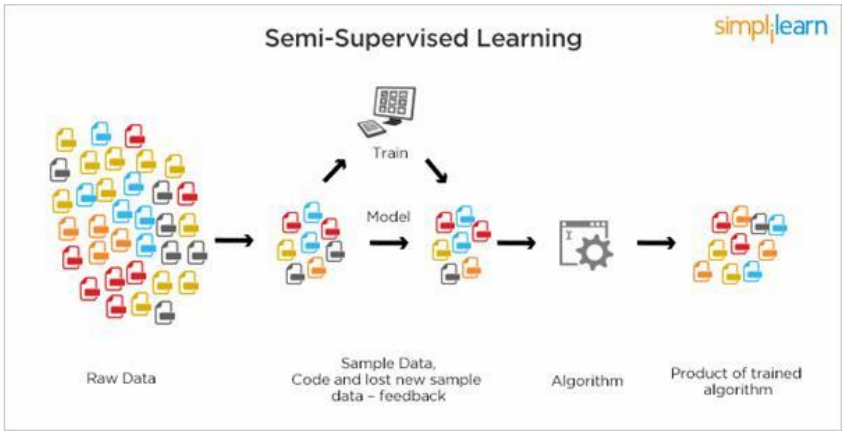


Figure 1.4.3: Semi Supervised Learning

2. INFORMATION ABOUT DEEP LEARNING

2.1 Importance of Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

If you are just starting out in the field of deep learning or you had some experience with neural networks some time ago, you may be confused. I know I was confused initially and so were many of my colleagues and friends who learned and used neural networks in the 1990s and early 2000s.

The leaders and experts in the field have ideas of what deep learning is and these specific and nuanced perspectives shed a lot of light on what deep learning is all about.

In this post, you will discover exactly what deep learning is by hearing from a range of experts and leaders in the field.

2.2 Uses of Deep Learning :

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in decision making. Deep learning AI is able to learn from data that is both unstructured and unlabeled. Deep learning, a machine learning subset, can be used to help detect fraud or money laundering.

Top Applications of Deep Learning Across Industries, Self-Driving Cars, News Aggregation and Fraud News Detection, Natural Language Processing, Virtual Assistants, Entertainment, Visual Recognition, Fraud Detection, Healthcare.

2.3 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.

Basic programming language used for machine learning is: PYTHON

3 INFORMATION ABOUT PYTHON

3.1INTRODUCTION 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.
- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

3.2 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.

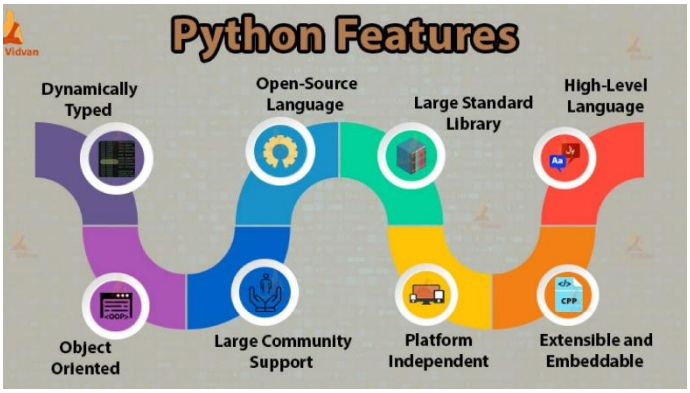


Figure 3.2: Features of python

3.3 SETUP OF 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.

3.3.1 Installation (using python IDLE):

Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.

Download python from www.python.org

When the download is completed, double click the file and follow the instructions to install it.

When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

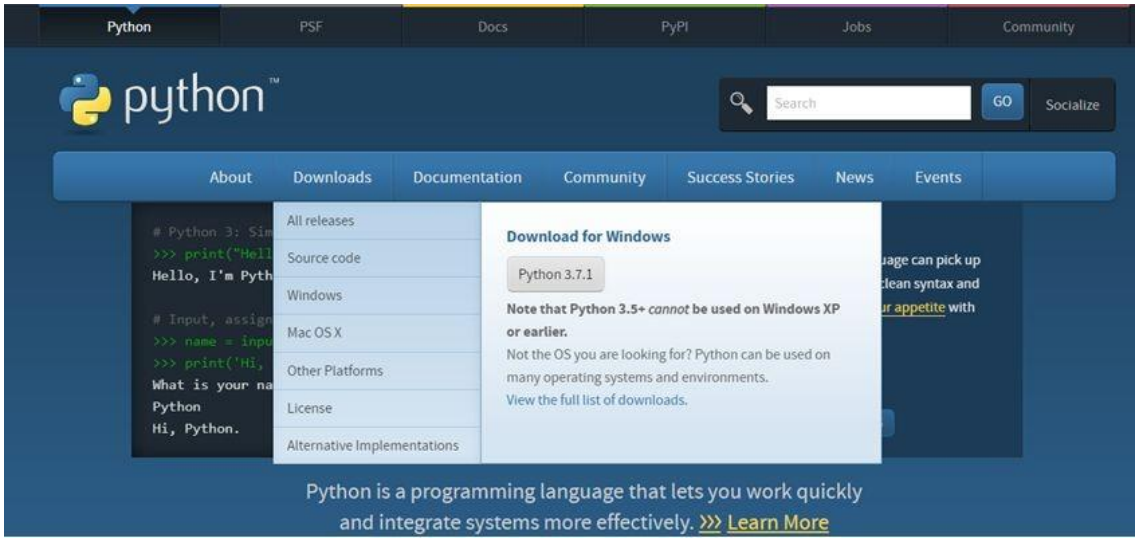


Figure 3.3.1: Python download

3.3.2 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:

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 Jupiter notebook (it opens in default browser)

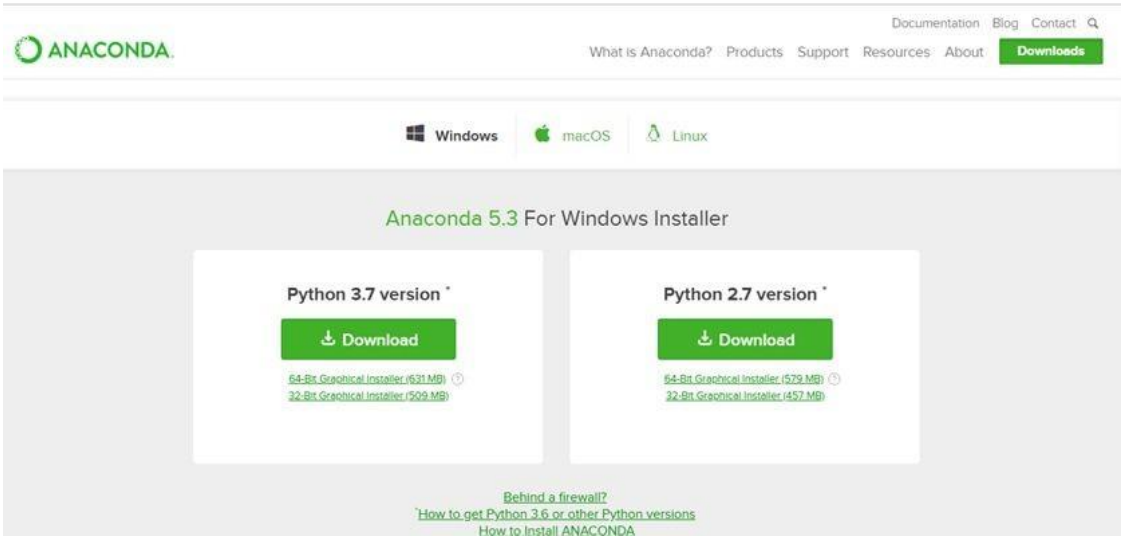


Figure 3.3.2: Anaconda download

Figure 3.3.2: Jupyter notebook

3.4 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

3.4.1 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).

3.4.2 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.

3.4.3 Python Lists:

3.4.3.1 Lists are the most versatile of Python's compound data types.

3.4.3.2 A list contains items separated by commas and enclosed within square ([]).

- 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.

3.4.4 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.

3.4.5 Python Dictionary:

- 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.

3.5 PYTHON FUNCTION:

3.5.1 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.

3.5.2 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.

3.6 PYTHON USING OOP's CONCEPTS:

3.6.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

- Defining a Class:
- We define a class in a very similar way how we define a function.
- Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is
- 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 3.6.1: Defining a Class

3.6.2 __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 methodhas a special name that starts and ends with two underscores: init ().

4. PROJECT NAME (WINE QUALITY PREDICTION)

4.1 PROJECT REQUIREMENTS:

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

4.1.1 PACKAGES USED:

IMPORTING THE LIBRARIES

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import preprocessing
```

Figure 4.1.1: Importing the libraries

4.1.2 Versions of the Packages

In [73]:
```
import pkg_resources
# list packages to be checked
root_packages = [
    'geoviews', 'geopandas', 'pandas', 'numpy',
    'matplotlib', 'shapely', 'cartopy', 'holoviews',
    'mapclassify', 'fiona', 'bokeh']
# print versions, but check if package is imported first
for m in pkg_resources.working_set:
    if m.project_name.lower() in root_packages:
        print(f"{m.project_name}=={m.version}")
```

```
pandas==0.25.1
numpy==1.16.5
matplotlib==3.1.1
bokeh==1.3.4
```

Figure 4.1.2: Versions of the Packages

4.1.3 Algorithms used

I have used 4 algorithms for Wine quality prediction

1. Knn (K – nearest neighbour algorithm)

2. Bernoulli naive bayes algorithm

3. DTC (Decision tree classifier algorithm)

4. Random forest classifier

1. Knn (K – nearest neighbour algorithm):

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method proposed by Thomas Cover used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. We are getting less Accuracy so that's why I am avoiding this knn algorithm.

2. Bernoulli naïve bayes algorithm:

Applying the naïve bayes algorithm and importing Bernoulli naïve bayes algorithm

Creating an object

Applying the algorithm to the data

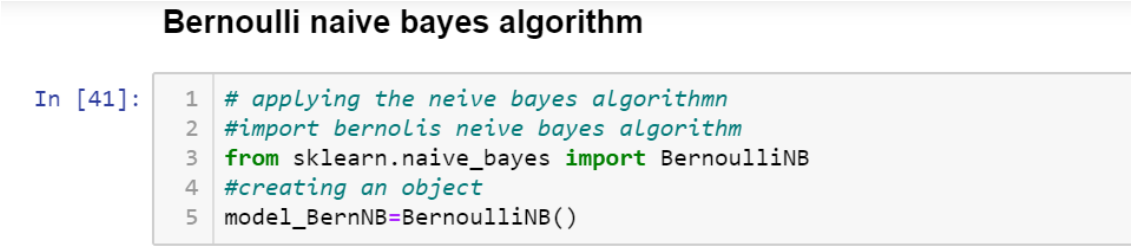


Figure 4.1.3(2): Importing and applying bayes

```
In [42]:
1  # applying the algorithm to the data
2  #object.fit(input,output)
3  model_BernNB.fit(X_train,y_train)

C:\Users\Vardhan reddy\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

Out[42]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Figure 4.1.3(2): Applying the algorithm

Predicting the final_train_pred from x_train and final_test_pred from x_test

```
In [43]:
1  final_train_pred=final_model.predict(scaled_X_train)
2  final_train_pred

Out[43]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
       'Medium'],
      dtype=object)

In [44]:
1  y_test_pred=model_BernNB.predict(X_test)
```

Figure 4.1.3(2): Predicting using train and test

Printing the classification report on trained, by this we get High, Low, Medium, Accuracy, macro avg, Weighted avg, Precision, Recall, f1-Score, Support , Checking f1 score

```
In [45]: 1 print(classification_report(y_train,final_train_pred))
```

```
              precision    recall  f1-score   support

        High       0.74      0.62      0.67       970
         Low       0.69      0.10      0.17       180
      Medium       0.87      0.94      0.90      3722

    accuracy                           0.85      4872
   macro avg       0.77      0.55      0.58      4872
weighted avg       0.84      0.85      0.83      4872
```

Figure 4.1.3(2): Checking the f1 score in train

I didn't get correct value in f1-score

Printing the classification report on test, by this we get High, Low, Medium, Accuracy, macro avg, Weighted avg, Precision, Recall, f1-Score, Support

Checking the f1 score

```
In [46]: 1 print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

        High       0.00      0.00      0.00       307
         Low       0.00      0.00      0.00        66
      Medium       0.77      1.00      0.87      1252

    accuracy                           0.77      1625
   macro avg       0.26      0.33      0.29      1625
weighted avg       0.59      0.77      0.67      1625

C:\Users\Vardhan reddy\Anaconda3\lib\site-packages\sklearn\metr
ics\classification.py:1437: UndefinedMetricWarning: Precision a
nd F-score are ill-defined and being set to 0.0 in labels with
no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Figure 4.1.3(2): checking the f1 score in test

In f1-score I didn't get proper values, So I am using another algorithm

DTC (Decision tree classifier algorithm), Random forest classifier algorithm

3. DTC (Decision tree classifier algorithm):

Applying the Decision tree classifier algorithm and importing Decision tree classifier algorithm

```
In [47]:
1 from sklearn import tree
2 DTC=tree.DecisionTreeClassifier(random_state=15)
3 DTC.fit(X_train, y_train)
4

Out[47]: DecisionTreeClassifier(class_weight=None, criterion='gini', max
_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_
split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=Fa
lse,
                       random_state=15, splitter='best')
```

Figure 4.1.3(3): importing and applying the DTC

Predicting the y_train_pred from x_train and y_test_pred from x_test

```
In [48]:
1 y_train_pred=DTC.predict(X_train)
2 y_train_pred
3

Out[48]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
'Medium'],
      dtype=object)

In [49]:
1 y_test_pred=DTC.predict(X_test)
2 y_test_pred

Out[49]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
'High'],
      dtype=object)
```

Figure 4.1.3(3): Predicting the train and test

Printing the classification report on trained, by this we get High, Low, Medium, Accuracy, macro avg, Weighted avg, Precision, Recall, f1-Score, Support
Checking f1 score

```
In [50]:
1 from sklearn.metrics import classification_report,f1_score
2 print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

        High       1.00      1.00      1.00       970
         Low       1.00      1.00      1.00       180
      Medium       1.00      1.00      1.00      3722

    accuracy                           1.00      4872
   macro avg       1.00      1.00      1.00      4872
weighted avg       1.00      1.00      1.00      4872
```

Figure 4.1.3(3): checking the f1 score of train

Printing the classification report on trained, by this we get High, Low, Medium, Accuracy, macro avg, Weighted avg, Precision, Recall, f1-Score, Support

Checking f1 score

```
In [51]:
1 print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

        High       0.57      0.64      0.61       307
         Low       0.32      0.35      0.33        66
      Medium       0.88      0.85      0.86      1252

    accuracy                           0.79      1625
   macro avg       0.59      0.61      0.60      1625
weighted avg       0.80      0.79      0.79      1625
```

Figure 4.1.3(3): checking the f1 score of test

By comparing with previous 2 algorithms we got some best f1 score in train Accuracy

4.Random forest classifier:

Importing Random forest and applying Random forest

```
In [52]:
1 from sklearn.ensemble import RandomForestClassifier
2 rfc=RandomForestClassifier(random_state=15)
3 rfc.fit(X_train,y_train)
```

```
C:\Users\Vardhan reddy\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estima
tors will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Vardhan reddy\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Out[52]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=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=10,
                       n_jobs=None, oob_score=False, random_state=15, verbose=0,
                       warm_start=False)
```

Figure 4.1.3(4): Importing and applying with random forest

Predicting the y_pred_train from X_train

```
In [53]:
1 # predicting the y_pred_train from X_train
2 y_predict_train=rfc.predict(X_train)
3 y_predict_train
```

```
Out[53]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
       'Medium'],
      dtype=object)
```

Figure 4.1.3(4): Predicting with train

Predicting the y_predict_test from X_test

```
In [54]:
1 # predicting the y_predict_test from X_test
2 y_predict_test=rfc.predict(X_test)
3 y_predict_test
```

```
Out[54]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
       'Medium'],
      dtype=object)
```

Figure 4.1.3(4): Predicting with test

Checking the f1 score of train

```
In [55]:
1 # checking the f1 score
2 print(classification_report(y_train,y_predict_train))
```

```
              precision    recall  f1-score   support

        High       0.99      0.98      0.99       970
         Low       1.00      0.93      0.96       180
      Medium       0.99      1.00      0.99      3722

    accuracy                           0.99      4872
   macro avg       0.99      0.97      0.98      4872
weighted avg       0.99      0.99      0.99      4872
```

Figure 4.1.3(4): Checking f1 score of train

Checking the f1 score of test

```
In [56]:
1 print(classification_report(y_test,y_predict_test))
```

```
              precision    recall  f1-score   support

        High       0.67      0.60      0.63       307
         Low       0.86      0.18      0.30        66
      Medium       0.87      0.93      0.90      1252

    accuracy                           0.84      1625
   macro avg       0.80      0.57      0.61      1625
weighted avg       0.83      0.84      0.82      1625
```

Figure 4.1.3(4): checking f1 score of test

Now we are taking different values in Data set for prediction

```
In [57]: 1 print(rfc.predict([[7.8,0.760,0.04,2.3,0.092,15.0,54.0,0.9970,3.26,0.65,9.8,0]]))
['Medium']

In [59]: 1 print(rfc.predict([[5.4,0.835,0.08,1.2,0.046,13.0,93.0,0.9924,3.57,0.85,13.0,0]]))
['High']

In [60]: 1 print(rfc.predict([[6.9,1.090,0.06,2.1,0.061,12.0,31.0,0.9948,3.51,0.43,11.4,0]]))
['Medium']

In [61]: 1 print(rfc.predict([[11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.9980,3.16,0.58,9.8,0]]))
['Medium']

In [62]: 1 print(rfc.predict([[5.4,0.835,0.08,1.2,0.046,13.0,93.0,0.9924,3.57,0.85,13.0,0]]))
['High']
```

Figure 4.1.3(4): Data set for prediction

4.2 Problem Statement :

To predict the quality of wine based on wine quality fixed acidity, volatile acidity, citric acid, residual sugar, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, Ph, sulpates, alcohol

4.3 Dataset Description

My data set consist quality fixed acidity, volatile acidity, citric acid, residual sugar, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, Ph, sulpates, alcohol and it has 6497 rows X 13 Columns

I have taken this Dataset from Kaggle website (Wine quality prediction)

Out[2]:

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

6497 rows × 13 columns

Figure 4.3: Dataset descriptions

4.4 Objective of the Case Study:

To predicting the wine quality based on the unknown users

To predict the quality of wine to the unknown users based on various components

5.DATA PREPROCESSING / FEATURES ENGINEERING AND EDA

5.1 Statistical Analysis

df.describe()

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000

Figure 5.1: Statistical Analysis

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types

5.2 Generating Plots:

Visualizing the data plots on all features which is there in data set

Visualizing the fixed_acidity feature in data set, in this fixed acid low is greater than medium, high.

```
In [14]: 1 sns.barplot(x='category',y='fixed_acidity',data=df,palette="bright")
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c6dbdc8>
```

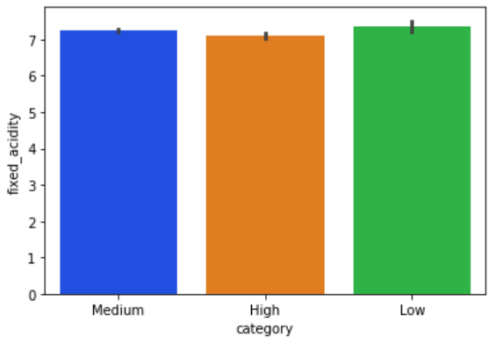


Figure 5.2 Fixed_acidity

Visualizing the Citric_acid feature in data set, In this Citric acid high is greater than medium, low

```
In [15]: 1 sns.barplot(x='category',y='citric_acid',data=df,palette="bright")
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c732908>
```

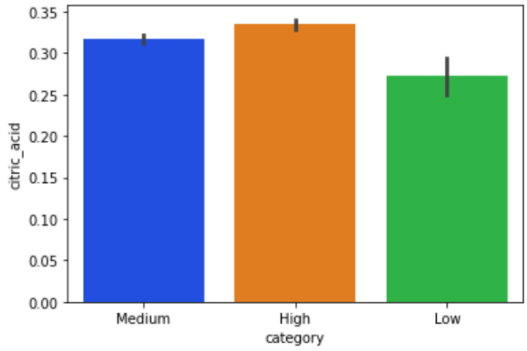


Figure 5.2 Citric acid

Visualizing the residual_sugar feature in data set, In this residual sugar medium greater than High, low

```
In [16]: 1 sns.barplot(x='category',y='residual_sugar',data=df,palette="bright")
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c6cdec8>
```

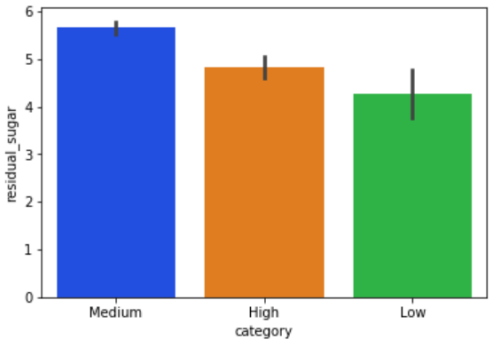


Figure 5.2 residual sugar

Visualizing the chlorides feature in data set, in this chlorides low is greater than medium, high.

```
In [17]: 1 sns.barplot(x='category',y='chlorides',data=df,palette="bright")
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c7ffa08>
```

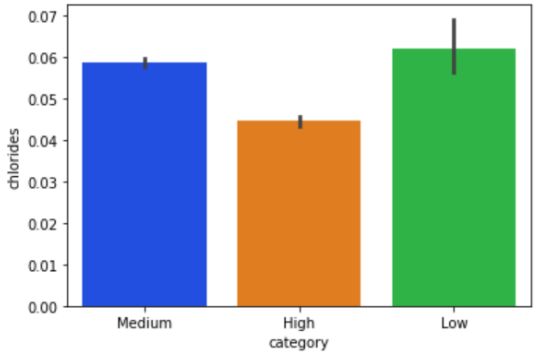


Figure 5.2 chlorides

Visualizing the free_sulfur_dioxide feature in data set in this high is greater than medium, low.

```
In [18]: 1 sns.barplot(x='category',y='free_sulfur_dioxide',data=df,palette="bright")
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c86da48>
```

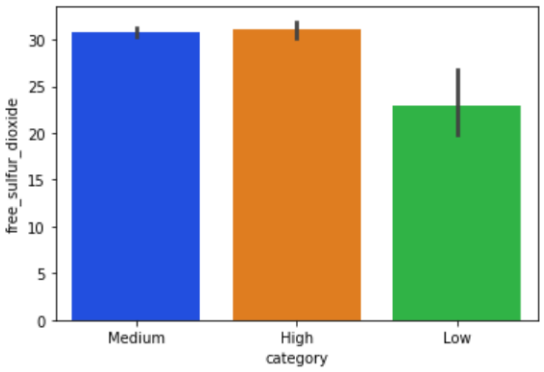


Figure 5.2 free sulfur dioxide

Visualizing the Total_sulfur_dioxide feature in data set in this medium is greater than high, low.

```
In [19]: 1 sns.barplot(x='category',y='total_sulfur_dioxide',data=df,palette="bright")
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1be1e788>
```

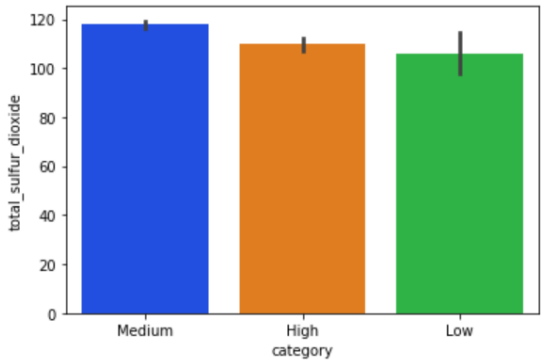


Figure 5.2: total sulfur dioxide

Visualizing the density feature in data set in these all three categories are equal

```
In [20]: 1 sns.barplot(x='category',y='density',data=df,palette="bright")
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c947e88>
```

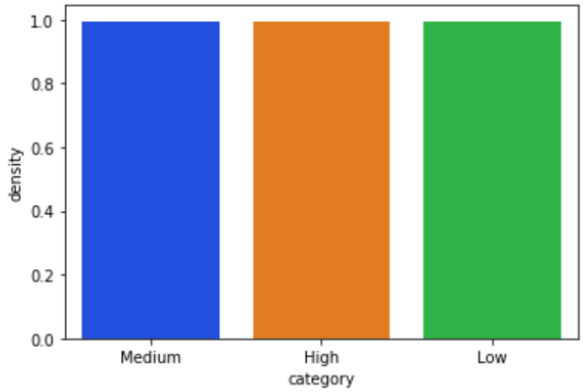


Figure 5.2: density

Visualizing the pH feature in data set in these all three categories are equal in Ph value 3.2

```
In [21]: 1 sns.barplot(x='category',y='pH',data=df,palette="bright")
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1c9b4c88>
```

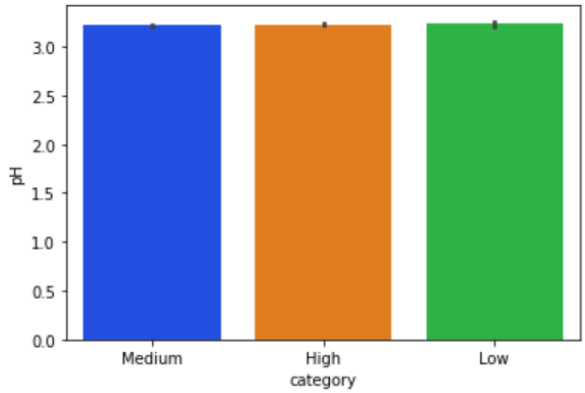


Figure 5.2: pH

Visualizing the sulphates feature in data set in this high has more sulphates than medium, low

```
In [22]: 1 sns.barplot(x='category',y='sulphates',data=df,palette="bright")
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1ca16108>
```

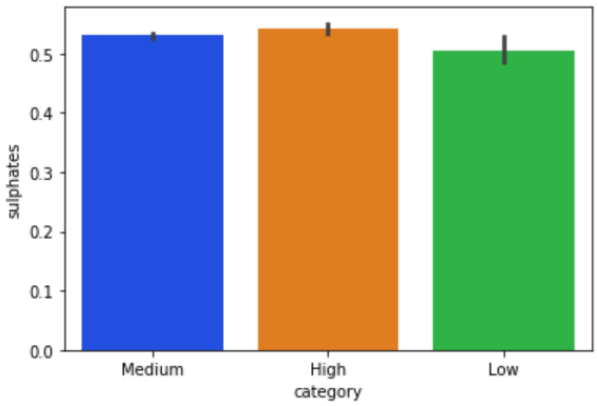


Figure 5.2: sulphates

Visualizing the alcohol feature in data set in, this alcohol has high comparing with medium, low.

```
In [23]:  1  sns.barplot(x='category',y='alcohol',data=df,palette="bright")
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1ca7a3c8>
```

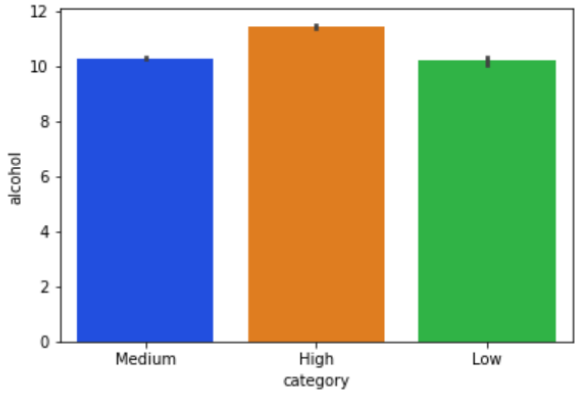


Figure 5.2: alcohol

Visualizing the color feature in data set in, this color high is greater than medium and low.

```
In [24]:  1  sns.barplot(x='category',y='color',data=df,palette="bright")
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x15f1dac7248>
```

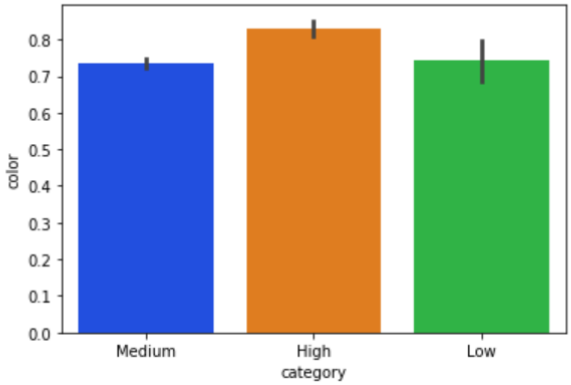
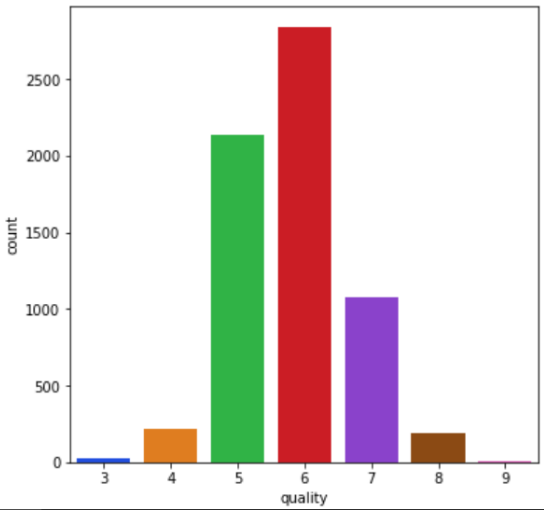


Figure 5.2: color

5.3 Data Type Conversions

```
In [11]:
plt.figure(figsize=(6,6))
sns.countplot(df["quality"],palette="bright")
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x21d9a124dc8>
```



I have convert the continues values into 3 types Low, Medium, High

Low – less than 5

Medium – 5

High – greater than 6

```
In [11]:
quality = df["quality"].values
category = []
for num in quality:
    if num < 5:
        category.append("Low")
    elif num > 6:
        category.append("High")
    else:
        category.append("Medium")
category = pd.DataFrame(data=category, columns=["category"])
df = pd.concat([df, category], axis=1)
df.drop(columns="quality", axis=1, inplace=True)
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Figure 5.3: data type conversion

```
In [13]:  1  df["category"].value_counts()

Out[13]: Medium    4974
         High      1277
         Low        246
         Name: category, dtype: int64
```

Figure 5.3: data type conversion

5.4 Handling Missing Values :

There is no Missing values.

5.5 Encoding Categorical Data:

I am importing the Label Encoder and converting the colors into 0,1.

```
In [4]:  1  from sklearn.preprocessing import LabelEncoder
         2  df['color']=LabelEncoder().fit_transform(df.color)
         3  df.head(200)
```

Out[4]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	0
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	0
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	0
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	7.8	0.590	0.33	2.0	0.074	24.0	120.0	0.9968	3.25	0.54	9.4	5	0
96	7.3	0.580	0.30	2.4	0.074	15.0	55.0	0.9968	3.46	0.59	10.2	5	0
97	11.5	0.300	0.60	2.0	0.067	12.0	27.0	0.9981	3.11	0.97	10.1	6	0
98	5.4	0.835	0.08	1.2	0.046	13.0	93.0	0.9924	3.57	0.85	13.0	7	0
99	6.9	1.090	0.06	2.1	0.061	12.0	31.0	0.9948	3.51	0.43	11.4	4	0

0 rows × 13 columns

Figure 5.5: label encoder converting the colors into 0,1

6 FEATURES SELECTION

6.1 Selecting relevant features for the analysis:

All the features are necessary

```
In [3]:
# Displaying top 10 in dataset
df.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 6.1: displaying top 10 in data set(features selection)

6.2 Drop irrelevant features:

As all the features are required relevant no need to drop.

6.3 Train-Test-Split

I had imported the Train-Test Split

There will be no change in rows and columns by using random state=1

```
In [17]:
from sklearn.model_selection import train_test_split
```

```
In [18]:
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=1)
```

```
In [19]:
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4872, 12)
(1625, 12)
(4872, 1)
(1625, 1)
```

Figure 6.3: import the Train-Test_Split

7. MODEL BUILDING AND EVALUATION

7.1 Brief about the algorithms used

I have used 4 algorithms for Wine quality prediction

1. Knn (K – nearest neighbour algorithm)
2. Bernoulli naive bayes algorithm
3. DTC (Decision tree classifier algorithm)
4. Random forest classifier

1. **Knn (K – nearest neighbour algorithm):**

   In pattern recognition, the **k-nearest neighbors algorithm** (**k-NN**) is a non-parametric **method** proposed by Thomas Cover used for classification and regression. In both cases, the input consists of the **k closest** training examples in the feature space.

2. **Bernoulli naïve bayes algorithm:**

   Applying the naïve bayes algorithm and importing Bernoulli naïve bayes algorithm
   Creating an object

3. **Decision tree classifier algorithm:**

   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

4. **Random forest classifier algorithm:**

   Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual

7.2 Train the Models

K-nearest neighbour algorithm:

```
In [25]:
# predictions on the data
# predict function
# syntax objectname.predict(input)
y_train_predict=knn.predict(scaled_X_train)
y_train_predict

Out[25]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
'Medium'],
      dtype=object)
```

Figure 7.2: train the model using knn algorithm

Bernoulli's neive bayes algorithm :

```
In [43]:
final_train_pred=final_model.predict(scaled_X_train)
final_train_pred

Out[43]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
'Medium'],
      dtype=object)
```

Figure 7.2: train the model using bernoulis neive bayes algorithm

DTC-Decision tree classifier training the model:

```
In [48]:
# predicting the y_train_pred from X_train
y_train_pred=DTC.predict(X_train)
y_train_pred


Out[48]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
'Medium'],
      dtype=object)
```

Figure 7.2: Train the model using DTC Algorithm

Random forest classifier training the Model:

```
In [53]:
1  # predicting the y_pred_train from X_train
2  y_predict_train=rfc.predict(X_train)
3  y_predict_train

Out[53]: array(['Medium', 'Medium', 'Medium', ..., 'Medium', 'Medium',
         'Medium'],
               dtype=object)
```

Figure 7.2: Train the model using Random forest algorithm

7.3 Validate the Models

1. Knn (K – nearest neighbour algorithm)

```
In [31]:
1  # classification report
2  # prediction-->PPV-->out of the positive values,how many tr
3  print(classification_report(y_train,final_train_pred))
```

	precision	recall	f1-score	support
High	0.74	0.62	0.67	970
Low	0.69	0.10	0.17	180
Medium	0.87	0.94	0.90	3722
accuracy			0.85	4872
macro avg	0.77	0.55	0.58	4872
weighted avg	0.84	0.85	0.83	4872

Figure 7.3: train f1 score of Knn

```
In [35]:
1  # checking the accuracy
2  from sklearn.metrics import classification_report
3  print(classification_report(y_test,final_test_pred))
```

	precision	recall	f1-score	support
High	0.60	0.49	0.54	307
Low	0.58	0.11	0.18	66
Medium	0.84	0.92	0.88	1252
accuracy			0.80	1625
macro avg	0.67	0.50	0.53	1625
weighted avg	0.79	0.80	0.79	1625

Figure 7.3: Test f1 score of knn

2. Bernoulli naive bayes algorithm

```
In [45]:
# Checking the f1 score of test
print(classification_report(y_train,final_train_pred))
```

	precision	recall	f1-score	support
High	0.74	0.62	0.67	970
Low	0.69	0.10	0.17	180
Medium	0.87	0.94	0.90	3722
accuracy			0.85	4872
macro avg	0.77	0.55	0.58	4872
weighted avg	0.84	0.85	0.83	4872

```
In [46]:
# Checking the f1 score of test
print(classification_report(y_test,y_test_pred))
```

	precision	recall	f1-score	support
High	0.00	0.00	0.00	307
Low	0.00	0.00	0.00	66
Medium	0.77	1.00	0.87	1252
accuracy			0.77	1625
macro avg	0.26	0.33	0.29	1625
weighted avg	0.59	0.77	0.67	1625

Figure 7.3: Train f1 score in Bernoulli navie and Test f1 score in Bernoulli navie

3.DTC (Decision tree classifier algorithm)

```
In [50]:
# Checking the f1 score of train
from sklearn.metrics import classification_report,f1_score
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

        High       1.00      1.00      1.00       970
         Low       1.00      1.00      1.00       180
      Medium       1.00      1.00      1.00      3722

    accuracy                           1.00      4872
   macro avg       1.00      1.00      1.00      4872
weighted avg       1.00      1.00      1.00      4872
```

```
In [51]:
# checking the f1 score of test
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

        High       0.57      0.64      0.61       307
         Low       0.32      0.35      0.33        66
      Medium       0.88      0.85      0.86      1252

    accuracy                           0.79      1625
   macro avg       0.59      0.61      0.60      1625
weighted avg       0.80      0.79      0.79      1625
```

Figure 7.3: Train f1 score in DTC and Test f1 score in DTC

4. Random forest classifier

```
In [55]:
# checking the f1 score in train
print(classification_report(y_train,y_predict_train))
```

```
              precision    recall  f1-score   support

        High       0.99      0.98      0.99       970
         Low       1.00      0.93      0.96       180
      Medium       0.99      1.00      0.99      3722

    accuracy                           0.99      4872
   macro avg       0.99      0.97      0.98      4872
weighted avg       0.99      0.99      0.99      4872
```

```
In [56]:
# Checking the f1 score in test
print(classification_report(y_test,y_predict_test))
```

```
              precision    recall  f1-score   support

        High       0.67      0.60      0.63       307
         Low       0.86      0.18      0.30        66
      Medium       0.87      0.93      0.90      1252

    accuracy                           0.84      1625
   macro avg       0.80      0.57      0.61      1625
weighted avg       0.83      0.84      0.82      1625
```

Figure 7.3: Train f1 score in random forest and Test f1 score in random forest

7.4 Make Predictions

Random forest classifier prediction

```
In [57]: 1 print(rfc.predict([[7.8,0.760,0.04,2.3,0.092,15.0,54.0,0.9970,3.26,0.65,9.8,0]]))
['Medium']

In [59]: 1 print(rfc.predict([[5.4,0.835,0.08,1.2,0.046,13.0,93.0,0.9924,3.57,0.85,13.0,0]]))
['High']

In [60]: 1 print(rfc.predict([[6.9,1.090,0.06,2.1,0.061,12.0,31.0,0.9948,3.51,0.43,11.4,0]]))
['Medium']

In [61]: 1 print(rfc.predict([[11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.9980,3.16,0.58,9.8,0]]))
['Medium']

In [62]: 1 print(rfc.predict([[5.4,0.835,0.08,1.2,0.046,13.0,93.0,0.9924,3.57,0.85,13.0,0]]))
['High']
```

Figure 7.4: Prediction using random forest classifier with same dataset values

7.5 Predictions from raw data

Predictions from raw data which is collected from Kaggle(Red wine)
https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009?select=winequality-red.csv

For prediction from raw data which is collected from Kaggle is used for predict the wine quality based on low, medium, high.

Predictions from raw data which is collected from kaggle (Red wine)

```
In [64]:
1 print(rfc.predict([[7.8,0.88,0,2.6,0.098,25,67,0.9968,3.2,0.68,9.8,0]]))
['Medium']

In [65]:
1 print(rfc.predict([[11.2,0.28,0.56,1.9,0.075,17,60,0.998,3.16,0.58,9.8,0]]))
['Medium']

In [66]:
1 print(rfc.predict([[6.7,0.58,0.08,1.8,0.097,15,65,0.9959,3.28,0.54,9.2,0]]))
2
['Medium']

In [69]:
1 print(rfc.predict([[6.9,0.605,0.12,10.7,0.073,40,83,0.9993,3.45,0.52,9.4,0]]))
['Medium']

In [70]:
1 print(rfc.predict([[6.9,0.84,0.21,4.1,0.074,16,65,0.99842,3.53,0.72,9.233333333,0]]))
2
['Medium']
```

Figure 7.5: prediction from raw data which is collected from Kaggle (redwine data values)

8. CONCLUSION

Based on the bar plots plotted we come to an conclusion that all input features are essential and affect the data, for example from the bar plot against quality and residual sugar we see that as the quality increases residual sugar is moderate and does not have change drastically. So this feature is not so essential as compared to others like alcohol and citric acid, so we can drop this feature while feature selection.

For classifying the wine quality, we have implemented multiple algorithms, namely

1) knn (k-nearest neighbour)

2) Bernoulli naive bayes classifier

3) Decision tree Classifier

4) Random Forest Classifier

Based on this below feature of data set I calculated f1 score for 4 algorithms and finally

We were able to achieve maximum accuracy using random forest and Decision tree Classifier.

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.

Based on this High, Low, Medium we can predict the quality of wine.

9. REFERENCES

1. https://en.wikipedia.org/wiki/Machine_learning

2. Wine quality data set which is download from Kaggle
https://drive.google.com/file/d/1Q8QrpmQJcLipoeoUROOg--6X8_ax28gf/view?usp=sharing

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

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

5.
https://en.wikipedia.org/wiki/Python_(programming_language)#:~:text=Python%20is%20an%20interpreted%2C%20high,notable%20use%20of%20significant%20whitespace.&text=Python%20is%20dynamically%20typed%20and%20garbage%2Dcollected.

6. Predictions from raw data which is collected from Kaggle(Red wine)

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

7. GitHub link: - https://github.com/Ranga440/Wine-Quality-Prediction-Project-