Capstone Project Report on

Wine Quality Prediction Using Machine Learning

Submitted towards the partial fulfillment of the criteria for award of Genpact Data Science Prodegree by Imarticus

Submitted By:

KISHOR S

ABSTRACT

The quality of a wine is important for the consumers as well as the wine industry.Product quality certification is used by industries to sell or advertise their products. 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. Classification models used here are 1) Random Forest 2)Decision Tree 3)Logistic Regression and Bagging classifier are the algorithms used to predict the quality of the Wine.

Dataset : https://github.com/Kishormsd/Capstone-Project-Predicting-The-Quality-Of-Wine-/blob/main/winequality-red.csv

Source code : https://github.com/Kishormsd/Capstone-Project-Predicting-The-Quality-Of-Wine-

ACKNOWLEDGEMENT

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of my capstone project. I am thankful for their aspiring guidance, invaluably constructive criticism, and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on several issues related to the project.

Further, I have fortunate to have Mr. PRASAD as my mentor. He has readily shared his immense knowledge in data science and guides me in a manner that the outcome resulted in enhancing my data skills.

I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: July 10, 2022 Name: KISHOR. S

CERTIFICATE OF COMPLETION

I hereby certify that the project titled “Wine Quality Prediction” was undertaken and completed the project (10th July, 2022).

Mentor : Mr. Prasad

Date : 10th July,2022

Place : Karur

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The quality of the wine is a very important part for the consumers as well as the manufacturing industries. Industries are increasing their sales using product quality certification. Nowadays, all over the world wine is a regularly used beverage and the industries are using the certification of product quality to increases their value in the market. Previously, testing of product quality will be done at the end of the production, this is time taking process and it requires a lot of resources such as the need for various human experts for the assessment of product quality which makes this process very expensive. Every human has their own opinion about the test, so identifying the quality of the wine based on humans experts it is a challenging task. There are several features to predict the wine quality but the entire features will not be relevant for better prediction.

Figure 1.1: wine sample

To improve product quality, testing is a key element that ensures product quality. Today, different types of companies are embracing and implementing new technology to verify and assess product quality. Testing the quality of a product with human expertise is an expensive and time-consuming operation that takes time to complete. For wine quality assurance, this study investigates various machine learning algorithms such Decision Tree, Random Forest Classifier. These strategies automate the quality assurance process by minimizing human interference and utilizing accessible product attributes. The research also highlights the key characteristics that can be used to forecast the values of dependent variables. One of the major factors that can be utilized for certification is wine quality assessment, and this sort of quality certification helps to ensure wine quality in the market

The aim of this project is to predict the quality of wine on a scale of 0–10 given a set of features as inputs. The dataset used is Wine Quality Data set from UCI Machine Learning Repository. Input variables are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulphur dioxide, total sulphur dioxide, density, pH, sulphates, alcohol. And the output variable is quality (score between 0 and 10). We are dealing only with red wine. We have quality being one of these values: [3, 4, 5, 6, 7, 8]. 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.

In this paper we are explaining the steps we followed to build our models for predicting the quality of red wine in a simple non-technical way. We are dealing only with red wine. We would follow similar process for white wine or we could even mix them together and include a binary attribute red/white, but our domain knowledge about wines suggests that we shouldn't. Classification is used to classify the wine as good or bad. Before examining the data it is often referred to as supervised learning because the classes are determined.

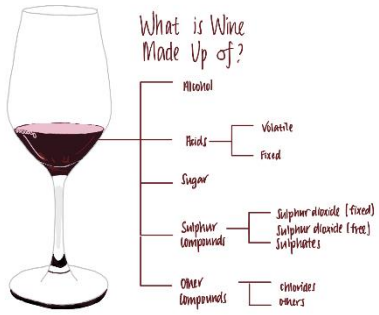


Figure 1.2: wine attributes

CHAPTER 2

DATA COLLECTION AND DATA PREPROOCESSING

2.1 Data collection:

The practice of cumulating and scrutinizing the data from the different sources is known as data collection. Data collection is way to keep track of past occurrence so that one can utilize data analysis to detect repetitive patterns.

Wine prediction dataset was collected from Kaggle, web and browser search. Finally, there are nearly 1600 datasets considered of 12 different variables,

Namely,

1. Fixed acidity
2. Volatile acidity
3. Citric acid
4. Residual Sugar
5. Chlorides
6. Free sulfur dioxide
7. Total sulfur dioxide
8. Density
9. pH
10. sulphates
11. alcohol

Output variables (based on Sensory data):

12. quality (score between 0 and 10)

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
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

Fig 2.1 Sample Dataset

2.2 Data preprocessing:

The process of modifying raw data into a form that analysts and data scientists can use in machine learning algorithms to find insights or forecast outcomes is called Data pre-processing. In this project the data processing method is to find missing values. Getting every data point for every record in a dataset is tough. Empty cells, values like null or a specific character, such as a question mark, might all indicate that data is missing. The dataset used in the project did not have any missing values.

CHAPTER 3

TRAIN VALIDATION SPLIT

It is a process of splitting the dataset into a training dataset and testing dataset using train_test_split () method of scikit learn module. 1600 data in the dataset has been divided as 70% of a dataset into training dataset-1120 and 30% of a dataset into testing dataset-480 data.

```
x = df.drop(['quality'],axis=1)
y = df['quality']

[36] from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(x_res,y_res, test_size=0.2, random_state=0)
```

Fig 3.1 Train Test split validation

x_train

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
170	7.900000	0.885000	0.030000	1.800000	0.058000	4.000000	8.000000	0.997200	3.360000	0.330000	9.100000
1630	9.710979	0.667548	0.336706	2.621959	0.094302	11.972552	87.945104	0.998043	3.264666	0.659451	10.132938
1482	7.200000	0.580000	0.540000	2.100000	0.114000	3.000000	9.000000	0.997190	3.330000	0.570000	10.300000
1553	7.300000	0.735000	0.000000	2.200000	0.080000	18.000000	28.000000	0.997650	3.410000	0.600000	9.400000
1493	7.700000	0.540000	0.260000	1.900000	0.089000	23.000000	147.000000	0.996360	3.260000	0.590000	9.700000
...	...	...	...	...	...	...	...	...	...	...	...
835	7.600000	0.665000	0.100000	1.500000	0.066000	27.000000	55.000000	0.996550	3.390000	0.510000	9.300000
1216	7.900000	0.570000	0.310000	2.000000	0.079000	10.000000	79.000000	0.996770	3.290000	0.690000	9.500000
1653	7.553679	0.415368	0.203679	3.807358	0.104000	28.463208	91.852833	0.997849	3.145368	0.574632	9.100000
559	13.000000	0.470000	0.490000	4.300000	0.085000	6.000000	47.000000	1.002100	3.300000	0.680000	12.700000
684	9.800000	0.980000	0.320000	2.300000	0.078000	35.000000	152.000000	0.998000	3.250000	0.480000	9.400000

1368 rows × 11 columns

y_train

```
170      bad
1630     bad
1482     bad
1553     bad
1493     bad
        ...
835      bad
1216    good
1653     bad
559     good
684      bad
Name: quality, Length: 1368, dtype: object
```

Fig 3.2 Training data

x_test

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
936	8.8	0.30	0.38	2.30	0.060	19.0	72.0	0.99543	3.39	0.72	11.8
6	7.9	0.60	0.06	1.60	0.069	15.0	59.0	0.99640	3.30	0.46	9.4
251	7.1	0.60	0.00	1.80	0.074	16.0	34.0	0.99720	3.47	0.70	9.9
840	11.1	0.42	0.47	2.65	0.085	9.0	34.0	0.99736	3.24	0.77	12.1
310	10.3	0.53	0.48	2.50	0.063	6.0	25.0	0.99980	3.12	0.59	9.3
...	...	...	...	...	...	...	...	...	...	...	...
616	9.6	0.60	0.50	2.30	0.079	28.0	71.0	0.99970	3.50	0.57	9.7
1168	6.5	0.34	0.27	2.80	0.067	8.0	44.0	0.99384	3.21	0.56	12.0
995	7.7	0.60	0.06	2.00	0.079	19.0	41.0	0.99697	3.39	0.62	10.1
315	7.1	0.35	0.29	2.50	0.096	20.0	53.0	0.99620	3.42	0.65	11.0
689	8.1	0.38	0.48	1.80	0.157	5.0	17.0	0.99760	3.30	1.05	9.4

342 rows × 11 columns

y_test

```
936     good
6        bad
251     good
840     good
310     good
        ...
616      bad
1168    good
995     good
315     good
689      bad
Name: quality, Length: 342, dtype: object
```

Fig 3.3 Testing data

CHAPTER 4

FITTING THE MODEL

Modifying the model's parameters to increase accuracy is referred to as fitting. To construct a machine learning model, an algorithm is performed on data for which the target variable is known. The model's accuracy is determined by comparing the model's outputs to the target variable's actual, observed values. Model fitting is the ability of a machine learning model to generalize data comparable to that with which it was trained. When given unknown inputs, a good model fit refers to a model that properly approximates the output.

4.1 Logistic Regression

Logistic Regression is a supervised learning technique that uses a machine learning algorithm. It's a method for predicting a categorical dependent variable from a set of independent variables. The output of a categorical dependent variable is predicted by logistic regression, and the outcome must be a categorical value. It can be 0 or 1, Yes or No, True or False, and so on, but instead of giving exact values, it delivers probabilistic values that fall between 0 and 1. The equation for the straight line can be written as

$$y=b0 + b1\ x1+b2\ x2+...bnxn$$

Because y in logistic regression can only be between 0 and 1, we can divide the above equation by (1-y) we will get equation

$$y/(1-y);\ 0 \text{ for } y=0 \text{ and infinity for } y=1$$

However, we require a range of -[infinity] to +[infinity], in which case the logarithm of the equation becomes

$$\text{Log } y/1-y=b0+b1\ x1 +b2\ x2+\ldots.\ bn\ xn$$

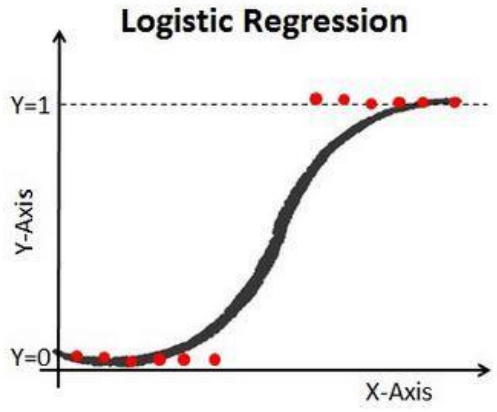


Fig 4.1.1 Logistic Regression

4.2 Random Forest Classifier

The Random Forest method consists of multiple decision tree classifiers to enhance the model's performance. Here ensemble learning is the principle used to resolve complicated problems. It is a supervised learning algorithm. Decision trees are created at random using the instances from the training set. Each of the decision trees gives out prediction as their outcome. The final prediction for the model is decided by majority voting. One of the reasons for its popularity as a machine learning approach is that it can handle the issue of overfitting and accuracy can be increased by using more trees.

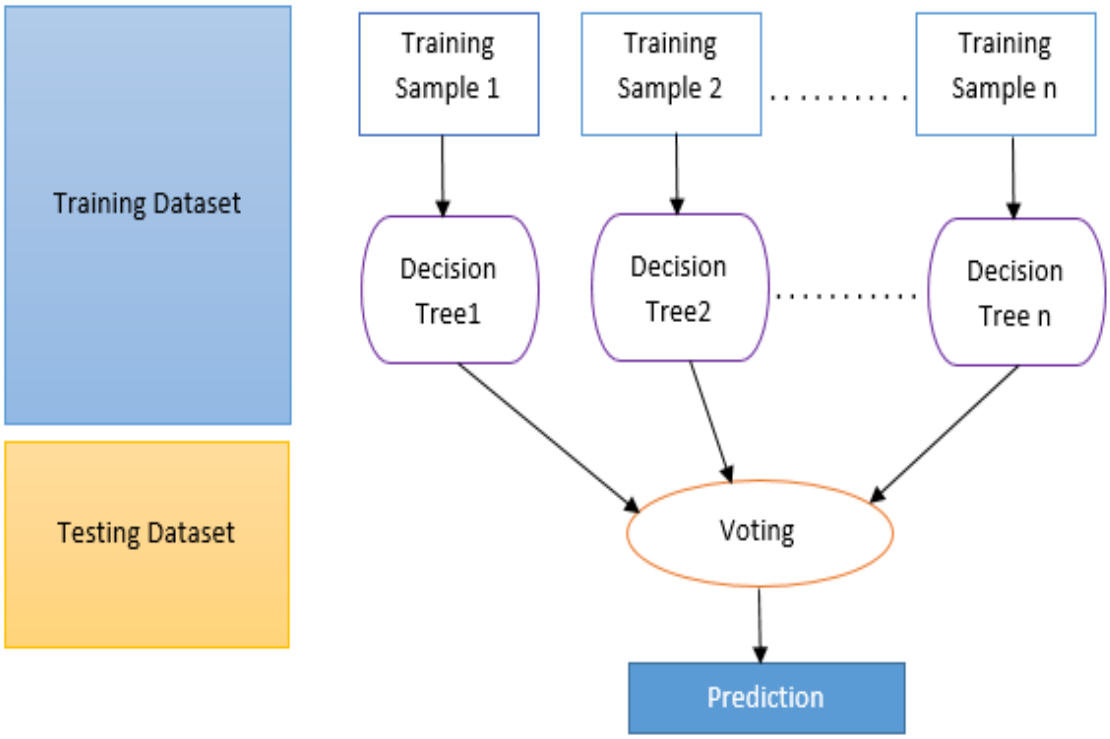


Fig 4.2.1 Random Forest Classifier

4.2.1 Random Forest Algorithm:

Step 1: K instances are chosen at random from the given training dataset.

Step 2: Decision trees are created for the chosen instances.

Step 3: The N is selected for the number of estimators to be created.

Step 4: Step 1 & Step 2 is repeated.

Step 5: For the new instance, the predictions of each estimator is determined, and the category with the highest vote is assigned.

4.3.1 Decision Tree Classifier

Decision Tree is a supervised learning technique used for both classification and regression problems where each path is a set of decision leading to a class. A sequence of questions is asked by taking an instance from the training set. The non-terminal node such as root and internal nodes has decision attributes. The decision is made by comparing the instance with the decision attribute, resulting in the split and jumping to the next node. This splitting continues, generating sub-trees until it reaches a leaf node which determines class labels for that instance. It divides the tree recursively, which is known as recursive partitioning. With high accuracy, decision trees are capable of handling high-dimensional data. It's a flowchart diagram-style representation that closely parallels human-level thinking. As a result, decision trees are simple to explain and apprehend.

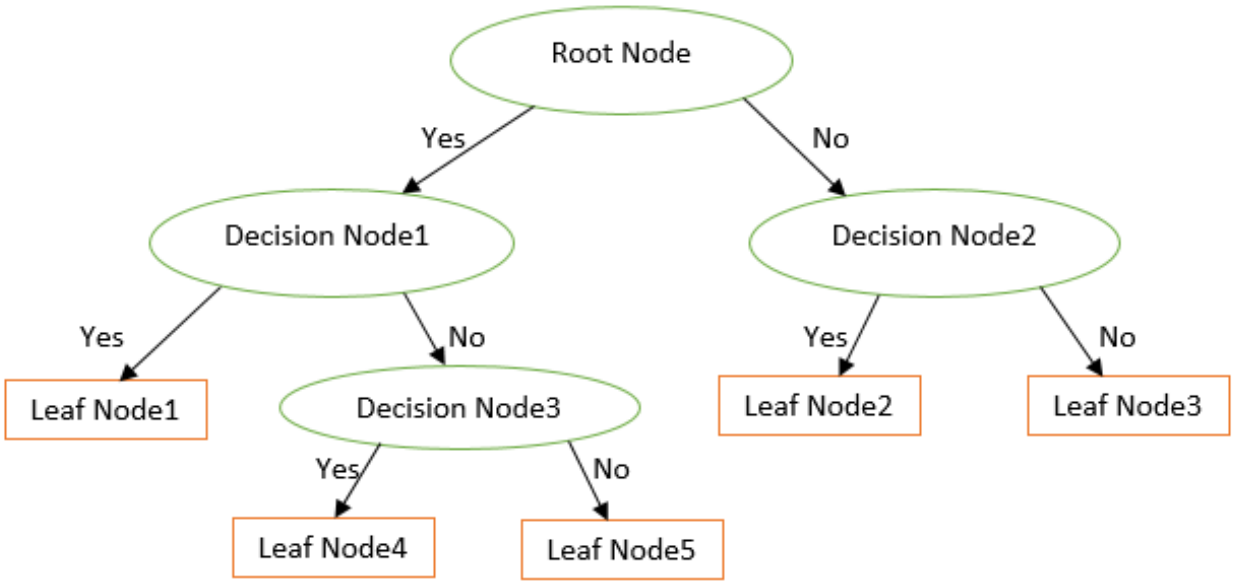


Fig :4.3.1 Decision Tree Classifier

4.3.1. Decision Tree Algorithm.

Step 1: Starting with the root node of the tree, which consists of the entire dataset, says S.

Step 2: The most appropriate attribute is obtained from the dataset by applying the Attribute Selection Measure (ASM).

Step 3: The S is divided into subdivisions that enclose feasible values for the most appropriate attributes.

Step 4: The node is formed in the decision tree with the most appropriate attributes

Step 5: The tree formation is setup by iteratively repeating this method for each child until one of the following requirements is met:

- The tuples are entirely correlated with the same attribute value.
- There are no further attributes accessible.
- There aren't any more instances.

4.4 Bagging Classifier

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples (or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

CHAPTER 5

PREDICTING NEW DATA

I have split the dataset into test part for the prediction of data the test data of size 480 data all models have fitted with the dataset. I have found the Random Forest Classifier with bagging classifier working well with the accuracy of 96% percentage of all other models.

```
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier

bag_classifier = BaggingClassifier(
      RandomForestClassifier(class_weight='balanced'),
    max_samples=0.5, max_features=0.5, bootstrap=False
)

bag_classifier.fit(x_train, y_train)
```

```
BaggingClassifier(base_estimator=RandomForestClassifier(class_weight='balanced'),
                  bootstrap=False, max_features=0.5, max_samples=0.5)
```

```
from sklearn.metrics import accuracy_score

def accuracy(model, data, labels):

    predictions = model.predict(data)
    acc = accuracy_score(labels, predictions)

    return acc
```

```
accuracy(bag_classifier, x_train, y_train)
```

```
0.9605263157894737
```

Fig 5.1 Prediction with new data

CHAPTER 6

CONFUSION MATRIX AND CLASSIFICATION REPORT

Confusion Matrix and Classification Report are the methods imported from the metrics module in the scikit learn library are calculated using the actual labels of test datasets and predicted values.

CONFUSION MATRIX

Confusion Matrix gives the matrix of frequency of true negatives, false negatives, true positives and false positives.

CLASSIFICATION REPORT

Classification Report is a metric used for evaluating the performance of a classification algorithm's predictions. It gives three things: Precision, Recall and f1-score of the model.

Precision refers to a classifier's ability to identify the number of positive predictions which are relatively correct. It is calculated as the ratio of true positives to the sum of true and false positives for each class.

$$Precision = \frac{TP}{TP + FP}$$

Where Precision-Positive Prediction Accuracy; TP-True Positive; FP-False Positive

Recall is the capability of a classifier to discover all positive cases from the confusion matrix. It is calculated as the ratio of true positives to the sum of true positives and false negatives for each class.

$$Recall = \frac{TP}{TP + FN}$$

Where Recall- The percentage of positives that were correctly identified; FN-False Negative

F1 score is a weighted harmonic mean of precision and recall, with 0.0 being the worst and 1.0 being the best. Since precision and recall are used in the computation, F1 scores are often lower than accuracy measurements.

$F1\ Score = \frac{2*PR}{(P+R)}$ Where P is Precision and R is Recall

```
confusion_matrix(y_test, y_pred)
```

```
array([[122,  41],
       [ 60, 119]])
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
conf_matrix = confusion_matrix(y_test,pred)
print("CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL:\n",conf_matrix)
```

```
CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL:
 [[118  45]
 [ 55 124]]
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
conf_matrix = confusion_matrix(y_test,pred2)
print("CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER MODEL:\n",conf_matrix)
```

```
CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER MODEL:
 [[134  29]
 [ 46 133]]
```

Fig 6.1 Confusion matrix

```
Accuracy =  0.7807017543859649
Classification Report
              precision    recall  f1-score   support

         bad       0.74      0.82      0.78       163
        good       0.82      0.74      0.78       179

    accuracy                           0.78       342
   macro avg       0.78      0.78      0.78       342
weighted avg       0.78      0.78      0.78       342
```

Fig: 6.2 Classification report

CHAPTER 7

CODING

IMPORT LIBRARIES

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, precision_score, f1_score
from sklearn.preprocessing import StandardScaler, LabelEncoder

import warnings
warnings.filterwarnings("ignore")
```

GETTING KNOW ABOUT THE DATA SET

```
df = pd.read_csv('winequality-red.csv')
```

```
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
df.tail()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
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

```
df.shape
```

```
(1599, 12)
```

```
df.info()
```

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

```
df.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

DATA ANALYSIS

CENTRAL TENDENCY

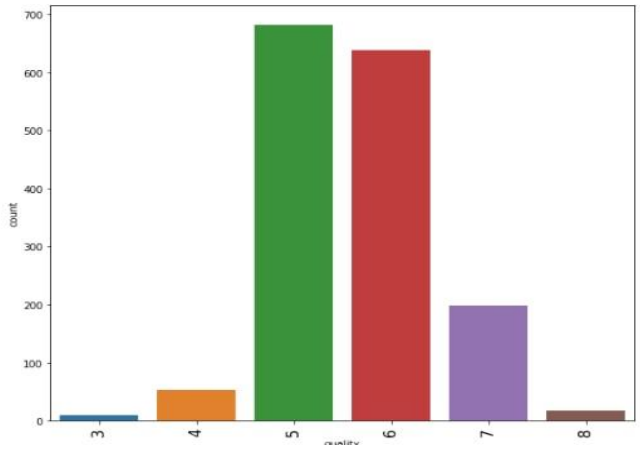
```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qu
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.63
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.80
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.00
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.00
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.00
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.00
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.00

```
df['quality'].value_counts()
```

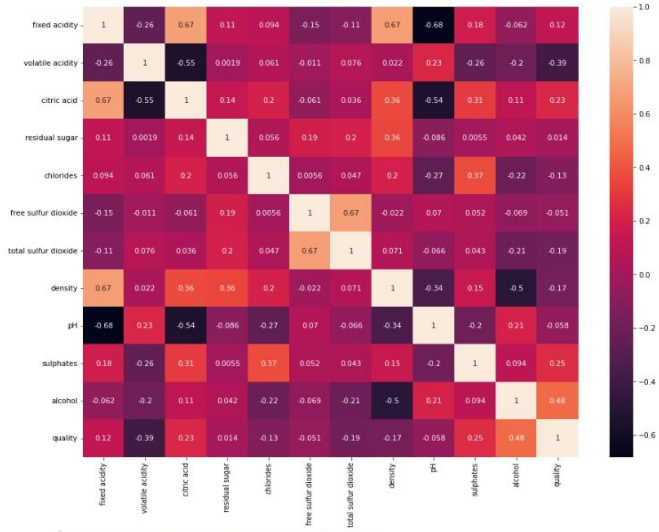
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
plt.figure(figsize=(10,8.5))
sns.countplot(df['quality'])
plt.xticks(rotation='vertical',size=15)
plt.show()
```



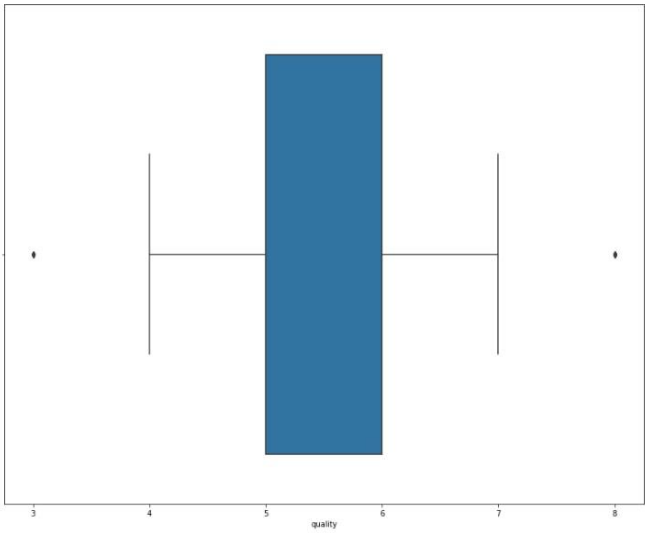
CORRELATION BETWEEN EACH FEATURES

```
plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

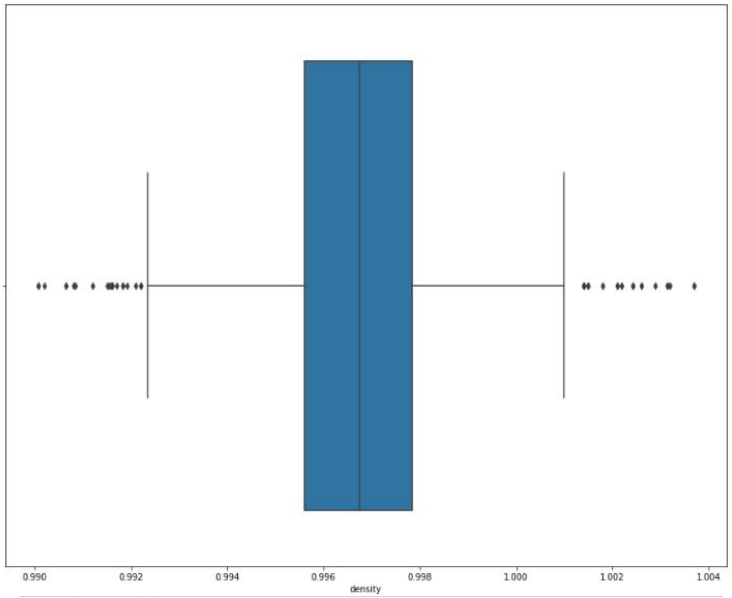


CHECKING OUTLIERS FOR EACH VARIABLE

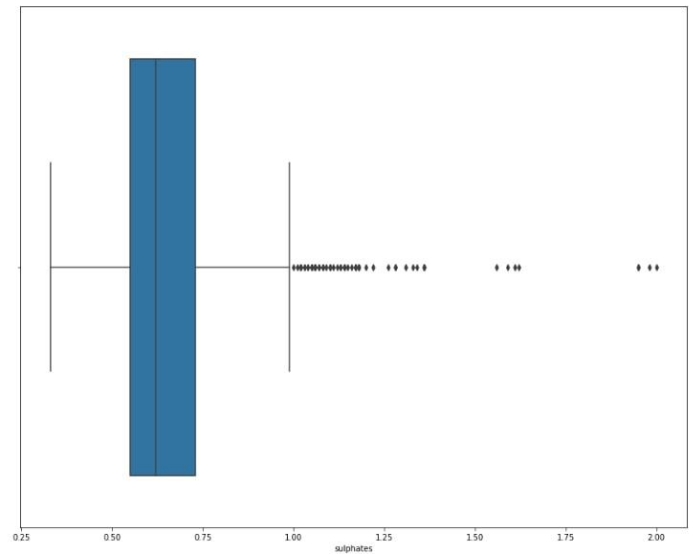
```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['quality'],data=df)
plt.show()
```



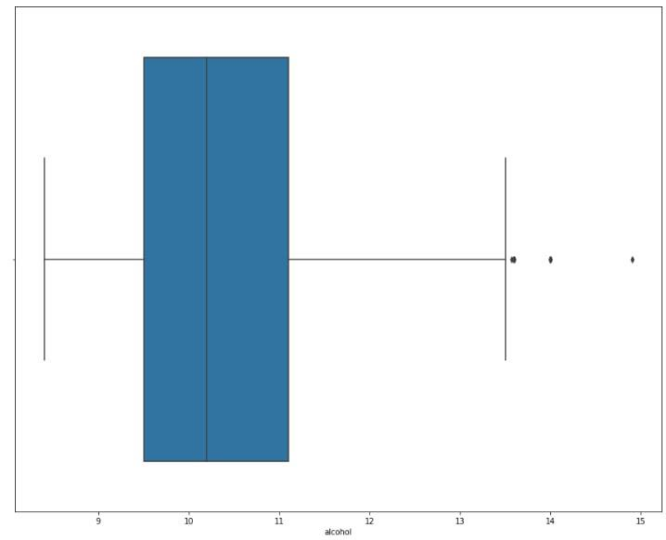
```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['density'],data=df)
plt.show()
```



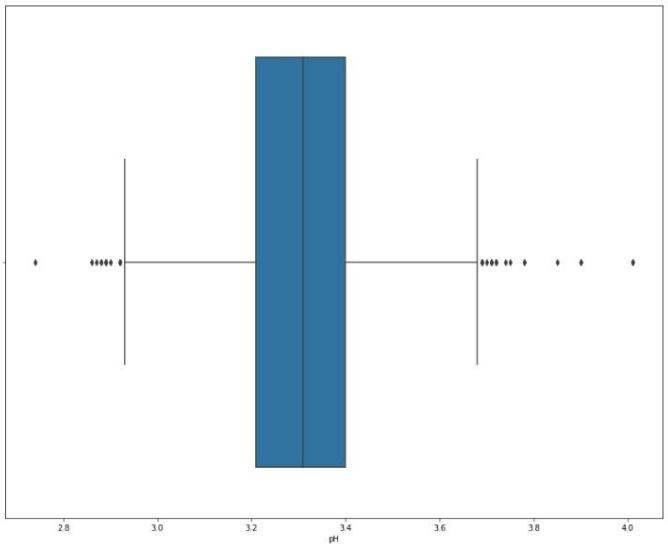

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['sulphates'],data=df)
plt.show()
```



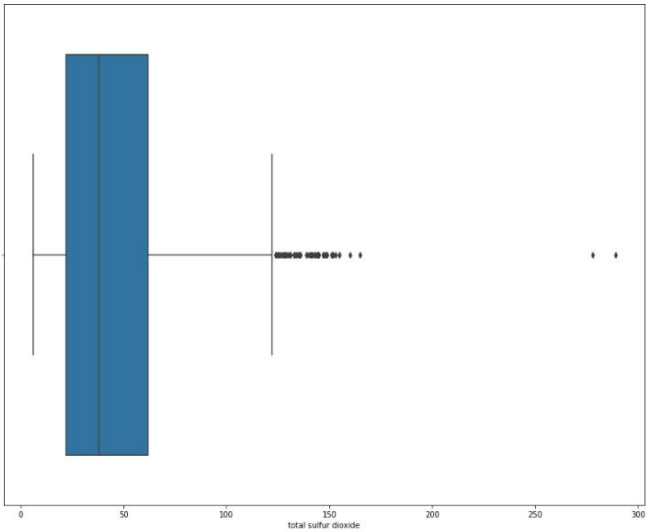

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['alcohol'],data=df)
plt.show()
```



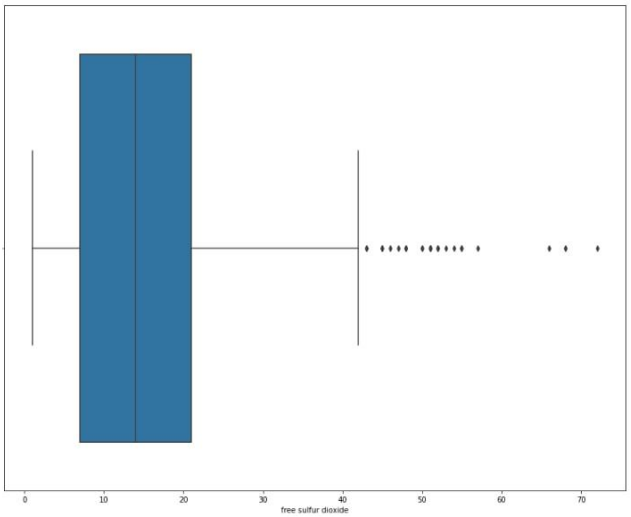
```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['pH'],data=df)
plt.show()
```



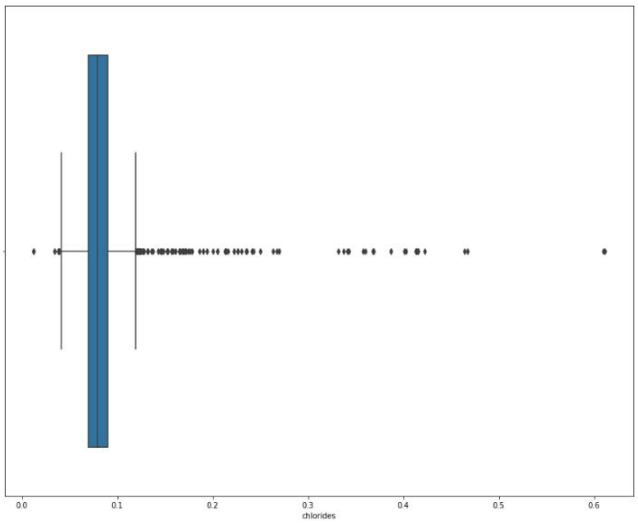

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['total sulfur dioxide'],data=df)
plt.show()
```



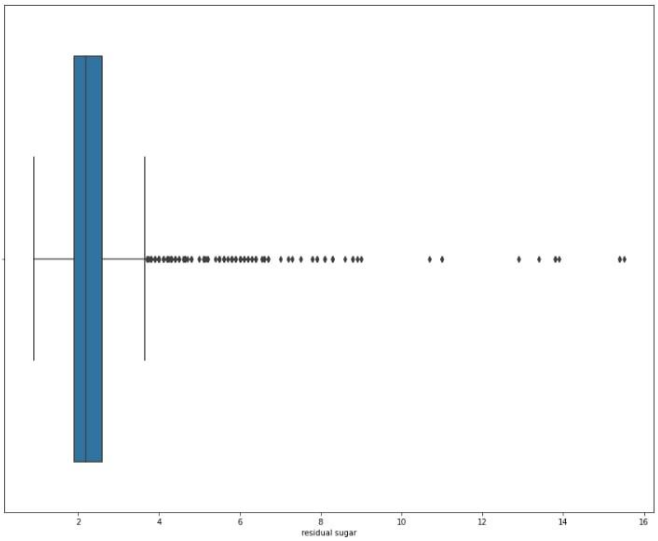

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['free sulfur dioxide'],data=df)
plt.show()
```



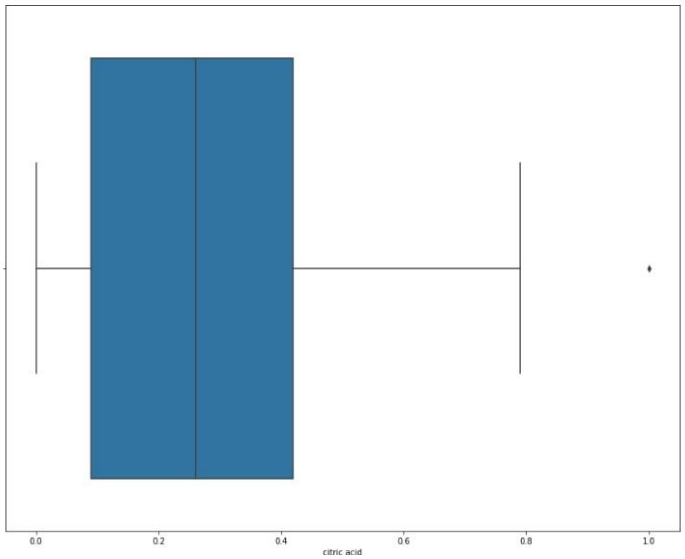
```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['chlorides'],data=df)
plt.show()
```



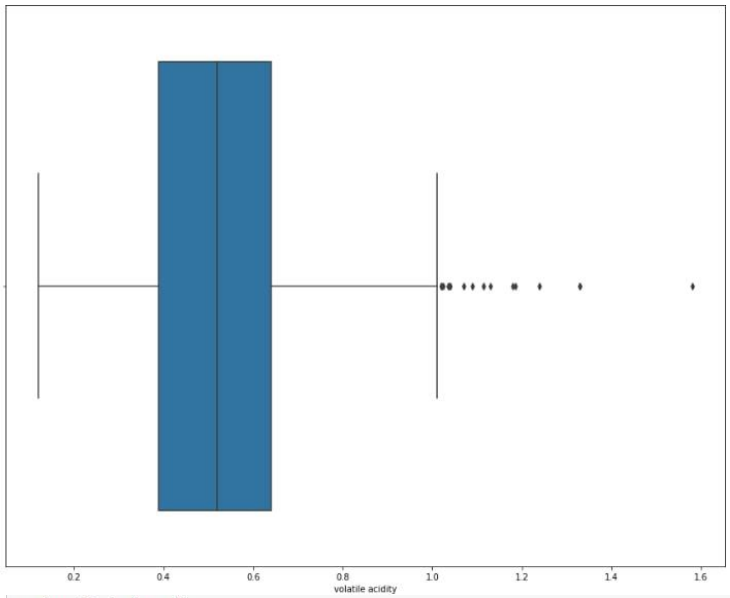

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['residual sugar'],data=df)
plt.show()
```



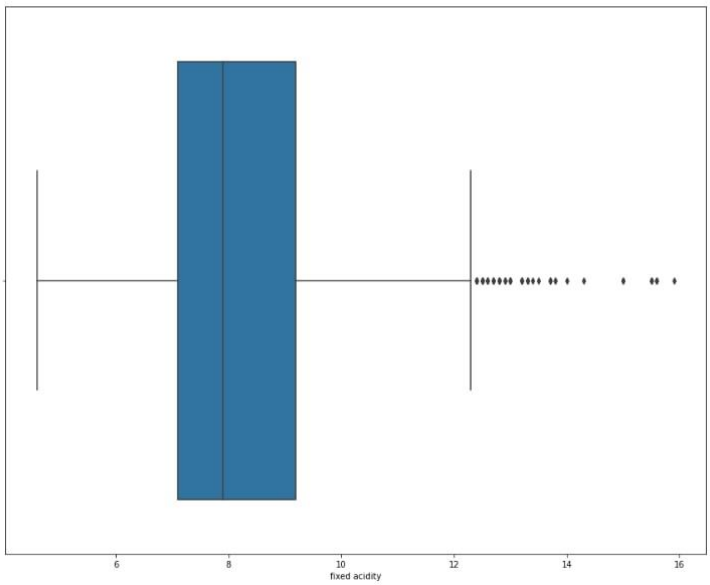

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['citric acid'],data=df)
plt.show()
```



```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['volatile acidity'],data=df)
plt.show()
```



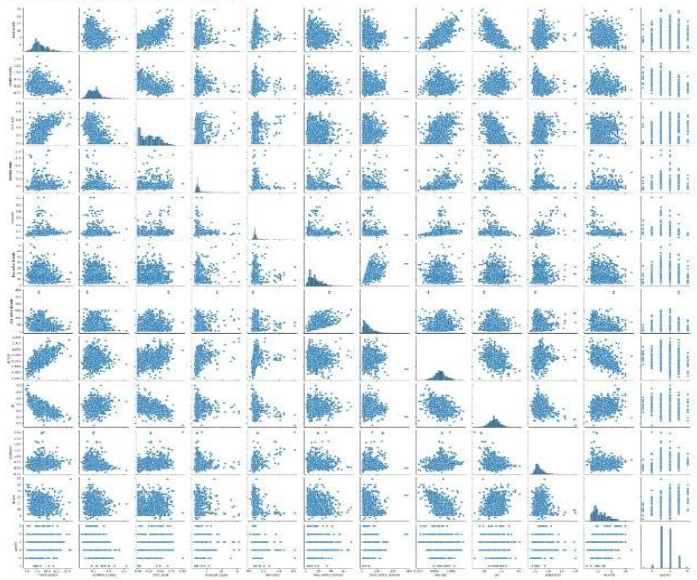

```
plt.figure(figsize=(15,12))
sns.boxplot(x=df['fixed acidity'],data=df)
plt.show()
```



FEATURES DISTRIBUTION TO EACH OTHER

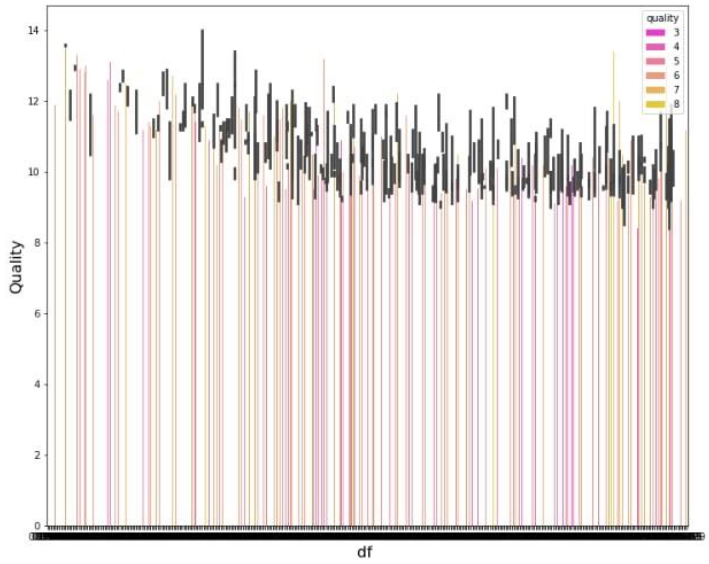
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fee18afdb10>
```



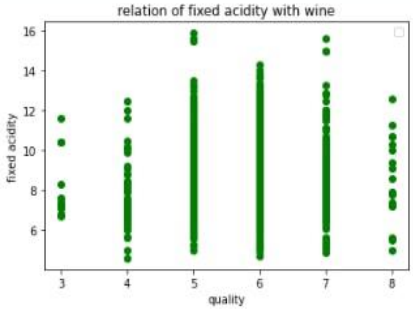
```
plt.figure(figsize=(12,10))
sns.barplot(x=df.density , y=df.alcohol, hue=df.quality,palette='spring')
plt.xlabel('df ',fontsize=16)
plt.ylabel('Quality',fontsize=16)
```

```
Text(0, 0.5, 'Quality')
```



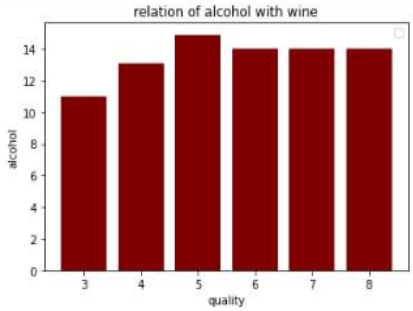

```
# checking the variation of fixed acidity in the different qualities of wine

plt.scatter(df['quality'], df['fixed acidity'], color = 'green')
plt.title('relation of fixed acidity with wine')
plt.xlabel('quality')
plt.ylabel('fixed acidity')
plt.legend()
plt.show()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```



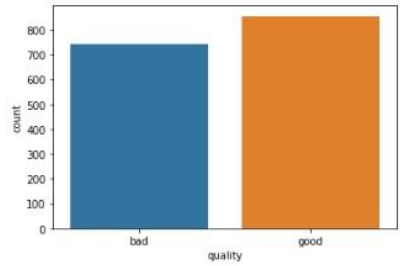

```
# checking the variation of fixed acidity in the different qualities of wine

plt.bar(df['quality'], df['alcohol'], color = 'maroon')
plt.title('relation of alcohol with wine')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.legend()
plt.show()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```



```
sns.countplot(df['quality'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee10cd7ad0>
```



MODEL BULDING

PRE PROCCESSING

```
df.columns
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

```
# converting the response variables(3-7) as binary response variables that is either good or bad

#names = ['bad', 'good']
#bins = (2, 6.5, 8)

#data['quality'] = pd.cut(data['quality'], bins = bins, labels = names)

df['quality'] = df['quality'].map({3 : 'bad', 4 :'bad', 5: 'bad',
                                   6: 'good', 7: 'good', 8: 'good'})
```

```
#analyzing the different values present in the dependent variable(quality column)
df['quality'].value_counts()
```

```
good    855
bad     744
Name: quality, dtype: int64
```

```
x = df.drop(['quality'],axis=1)
y = df['quality']
```

```
from imblearn.over_sampling import SMOTE
```

```
os=SMOTE()
x_res,y_res = os.fit_resample(x,y)
```

```
x_train, x_test, y_train, y_test = train_test_split(x_res,y_res, test_size=0.2, random_state=0)
```

```
x_train
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
170	7.900000	0.885000	0.030000	1.800000	0.058000	4.0	8.000000	0.997200	3.360000	0.330000	9.100000
1630	7.173610	0.703137	0.126041	2.008797	0.075560	12.0	20.000000	0.995500	3.300556	0.540880	10.091203
1482	7.200000	0.580000	0.540000	2.100000	0.114000	3.0	9.000000	0.997190	3.330000	0.570000	10.300000
1553	7.300000	0.735000	0.000000	2.200000	0.080000	18.0	28.000000	0.997650	3.410000	0.600000	9.400000
1493	7.700000	0.540000	0.260000	1.900000	0.089000	23.0	147.000000	0.996360	3.260000	0.590000	9.700000
...	...	...	...	...	...	...	...	...	...	...	...
835	7.600000	0.665000	0.100000	1.500000	0.066000	27.0	55.000000	0.996550	3.390000	0.510000	9.300000
1216	7.900000	0.570000	0.310000	2.000000	0.079000	10.0	79.000000	0.996770	3.290000	0.690000	9.500000
1653	8.066115	0.597639	0.086454	2.581110	0.056299	3.0	6.984258	0.992924	3.218583	0.374093	12.959072
559	13.000000	0.470000	0.490000	4.300000	0.085000	6.0	47.000000	1.002100	3.300000	0.680000	12.700000
684	9.800000	0.980000	0.320000	2.300000	0.078000	35.0	152.000000	0.998000	3.250000	0.480000	9.400000

1368 rows × 11 columns

```
x_test
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
936	8.8	0.30	0.38	2.30	0.060	19.0	72.0	0.99543	3.39	0.72	11.8
6	7.9	0.60	0.06	1.60	0.069	15.0	59.0	0.99640	3.30	0.46	9.4
251	7.1	0.60	0.00	1.80	0.074	16.0	34.0	0.99720	3.47	0.70	9.9
840	11.1	0.42	0.47	2.65	0.085	9.0	34.0	0.99736	3.24	0.77	12.1
310	10.3	0.53	0.48	2.50	0.063	6.0	25.0	0.99980	3.12	0.59	9.3
...	...	...	...	...	...	...	...	...	...	...	...
616	9.6	0.60	0.50	2.30	0.079	28.0	71.0	0.99970	3.50	0.57	9.7
1168	6.5	0.34	0.27	2.80	0.067	8.0	44.0	0.99384	3.21	0.56	12.0
995	7.7	0.60	0.06	2.00	0.079	19.0	41.0	0.99697	3.39	0.62	10.1
315	7.1	0.35	0.29	2.50	0.096	20.0	53.0	0.99620	3.42	0.65	11.0
689	8.1	0.38	0.48	1.80	0.157	5.0	17.0	0.99760	3.30	1.05	9.4

```
y_train
```

```
170      bad
1630     bad
1482     bad
1553     bad
1493     bad
        ...
835      bad
1216    good
1653     bad
559     good
684      bad
Name: quality, Length: 1368, dtype: object
```

```
y_test
```

```
936     good
6        bad
251     good
840     good
310     good
        ...
616      bad
1168    good
995     good
315     good
689      bad
Name: quality, Length: 342, dtype: object
```

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[122,  41],
       [ 60, 119]])
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import GridSearchCV, cross_val_score



# creating the model
model = LogisticRegression()

# feeding the training set into the model
model.fit(x_train, y_train)

# predicting the results for the test set
y_pred = model.predict(x_test)

# calculating the training and testing accuracies
print("Training accuracy :", model.score(x_train, y_train))
print("Testing accuracy :", model.score(x_test, y_test))

# classification report
print(classification_report(y_test, y_pred))

# confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
Training accuracy : 0.7602339181286549
Testing accuracy : 0.7076023391812866
              precision    recall  f1-score   support

         bad       0.68      0.74      0.71       163
        good       0.74      0.68      0.71       179

    accuracy                           0.71       342
   macro avg       0.71      0.71      0.71       342
weighted avg       0.71      0.71      0.71       342

[[120  43]
 [ 57 122]]
```

DECISION TREE CLASSIFIER

```
model = DecisionTreeClassifier(random_state=0,criterion='entropy',max_depth=None)
```

```
model.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
pred = model.predict(x_test)
```

```
df_predict = pd.DataFrame({'Actual': y_test, 'Predicted': pred})
df_predict.head(10)
```

	Actual	Predicted
936	good	good
6	bad	bad
251	good	good
840	good	good
310	good	good
408	good	bad
1015	good	good
620	bad	bad
1275	good	good
641	bad	bad

MODEL EVALUATION

```
acc_per = accuracy_score(y_test,pred)
```

```
print('Model Accuracy : '+ str(round(acc_per*100))+'%')
```

```
Model Accuracy : 71%
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
conf_matrix = confusion_matrix(y_test,pred)
print("CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL:\n",conf_matrix)
```

```
CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL:
 [[118  45]
 [ 55 124]]
```

```
print('Accuracy = ',accuracy_score(y_test,pred))
print('Classification Report\n',classification_report(y_test,pred))
```

```
Accuracy =  0.7076023391812866
Classification Report
               precision    recall  f1-score   support

          bad       0.68      0.72      0.70       163
         good       0.73      0.69      0.71       179

     accuracy                           0.71       342
    macro avg       0.71      0.71      0.71       342
 weighted avg       0.71      0.71      0.71       342
```

RANDOM FOREST CLASSIFIER

```
model2 = RandomForestClassifier(n_estimators=120,random_state=0)
```

```
model2.fit(x_train,y_train)
```

```
RandomForestClassifier(n_estimators=120, random_state=0)
```

```
pred2 = model2.predict(x_test)
```

```
df_predict2 = pd.DataFrame({'Actual': y_test, 'Predicted': pred2})
df_predict2.head(10)
```

	Actual	Predicted
936	good	good
6	bad	bad
251	good	bad
840	good	good
310	good	good
408	good	good
1015	good	good
620	bad	bad
1275	good	good
641	bad	bad

MODEL EVALUATION

```
acc_per2 = accuracy_score(y_test,pred2)
```

```
print('Model Accuracy : '+ str(round(acc_per2*100,4)))
```

```
Model Accuracy : 78.0702
```

```
cross_validation = cross_val_score(model2,x_res,y_res,cv=5)
print('Cross validations mean score ',round(np.mean(cross_validation)*100,4))
```

```
Cross validations mean score  74.4444
```

```
print("Recall Score :",recall_score(y_test, pred2, average='weighted'))
print("Percision Score :",precision_score(y_test, pred2, average='weighted'))
print("F1 Score :",f1_score(y_test, pred2, average='weighted'))
```

```
Recall Score : 0.7807017543859649
Percision Score : 0.7845065338242726
F1 Score : 0.7806698804585226
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
conf_matrix = confusion_matrix(y_test,pred2)
print("CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER MODEL:\n",conf_matrix)
```

```
CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER MODEL:
 [[134  29]
 [ 46 133]]
```

```
print('Accuracy = ',accuracy_score(y_test,pred2))
print('Classification Report\n',classification_report(y_test,pred2))
```

```
Accuracy =  0.7807017543859649
Classification Report
              precision    recall  f1-score   support

         bad       0.74      0.82      0.78       163
        good       0.82      0.74      0.78       179

    accuracy                           0.78       342
   macro avg       0.78      0.78      0.78       342
weighted avg       0.78      0.78      0.78       342
```

```
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier

bag_classifier = BaggingClassifier(
      RandomForestClassifier(class_weight='balanced'),
    max_samples=0.5, max_features=0.5, bootstrap=False
)

bag_classifier.fit(x_train, y_train)
```

```
BaggingClassifier(base_estimator=RandomForestClassifier(class_weight='balanced'),
                  bootstrap=False, max_features=0.5, max_samples=0.5)
```

```
from sklearn.metrics import accuracy_score

def accuracy(model, data, labels):

    predictions = model.predict(data)
    acc = accuracy_score(labels, predictions)

    return acc
```

```
accuracy(bag_classifier, x_train, y_train)
```

```
0.9605263157894737
```

CHAPTER 8

CONCLUSION

In recent years, there has been an increase in interest in the wine sector, necessitating its expansion. As a result, companies are investing in innovative technologies to boost wine production and sales. Wine quality certification is crucial for a product's marketability, and it necessitates human wine testing. This research looks into several machine learning techniques for predicting wine quality. This study shows how the results alter when the test mode is changed for each categorization model. The analysis of classifiers on red wine datasets is part of the research. The percentage of correctly identified cases, precision, recall, and F measure are all used to explain the results. Different classifiers are tested on datasets, including Logistic Regression, Decision Tree, Random Forest. The results of the studies lead us to believe that the Random Forests Algorithm outperforms other classifiers in classification tasks. The Random Forest Algorithm predicts wine quality with a maximum accuracy of 78 percent with using a bagging classifier 96%, Logistic Regression with the accuracy of 70% and, Decision tree with the accuracy of 71%. Therefore, in the classification algorithms by selecting the appropriate features and balancing the data can improve the performance of the model.

CHAPTER 9

REFERENCE

[1] Gupta, Y. (2018), Selection of important features and predicting wine quality using machine learning techniques, Procedia Computer Science, 125, 305–312, doi:10.1016/j.procs.2017.12.041.

[2] Shin, T. (2020, May 8), Predicting Wine Quality with Several Classification Techniques Medium,https://towardsdatascience.com/predicting-wine-qualitywith-several-classification-techniques-179038ea6434.

[3] Devika Pawar, Aakanksha Mahajan, Sachin Bhoithe (2019), Wine Quality Prediction using Machine Learning Algorithms, International Journal of Computer Applications Technology and Research, Volume 8–Issue 09, 385-388, ISSN:-2319–8656.