Fuse Machines

AI FELLOWSHIP 2023

A Final Project Report on:

Wine Quality Classification

Submitted By:

Nirajan Bekoju

Bishal Adhikari

Rijan Pokhrel

Manoj Khatri

Submitted To:

Fuse Machines

Submission Date:

1^{st} May, 2023

Contents

List of Figures

List of Tables

Chapter 1

Abstract

This report details a machine learning project focused on predicting the quality of red wines using their chemical properties. The dataset used in this project contained 11 features describing the chemical composition of wines, as well as a quality rating ranging from 0 to 10. To improve the performance of machine learning algorithms, various preprocessing techniques such as standard scaler, min max scaler, and logarithmic and boxcox transformation were applied to the data. Exploratory data analysis was also performed, visualizing the data distributions, box plots, and scatter plots to better understand the relationships between the features and the target variable. Several popular machine learning algorithms were trained on the preprocessed data, including logistic regression, SVM, Random forest, decision trees, and boosting algorithms, and their performance was compared. Finally, the best algorithm and preprocessing technique were identified based on performance metrics such as accuracy, precision, recall, and F1 score. The results and conclusions of this project are presented in detail in this report.

Keywords: Wine Quality, Data Analysis, Machine Learning, Random Forest, Django

Chapter 2

Introduction

2.1 Problem Statement

The dataset is related to red variants of the Portuguese "Vinho Verde" wine.The dataset describes the amount of various chemicals present in wine and their effect on it's quality. The datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).Task is to predict the quality of wine using the given data.

A simple yet challenging project, to anticipate the quality of wine.

The complexity arises due to the fact that the dataset has fewer samples, & is highly imbalanced.

Data Source : https://archive.ics.uci.edu/ml/datasets/wine+quality

2.2 About Dataset

The dataset contains the following columns:

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
12. quality (Targe Variable) : ranges from 0 to 10

The data information is as follow:

```
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64
 12  Id                    1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

Figure 2.1: Data Information

2.3 Dataset Statistics

The count plot of the whole dataset on the basis of quality of wine is shown below.

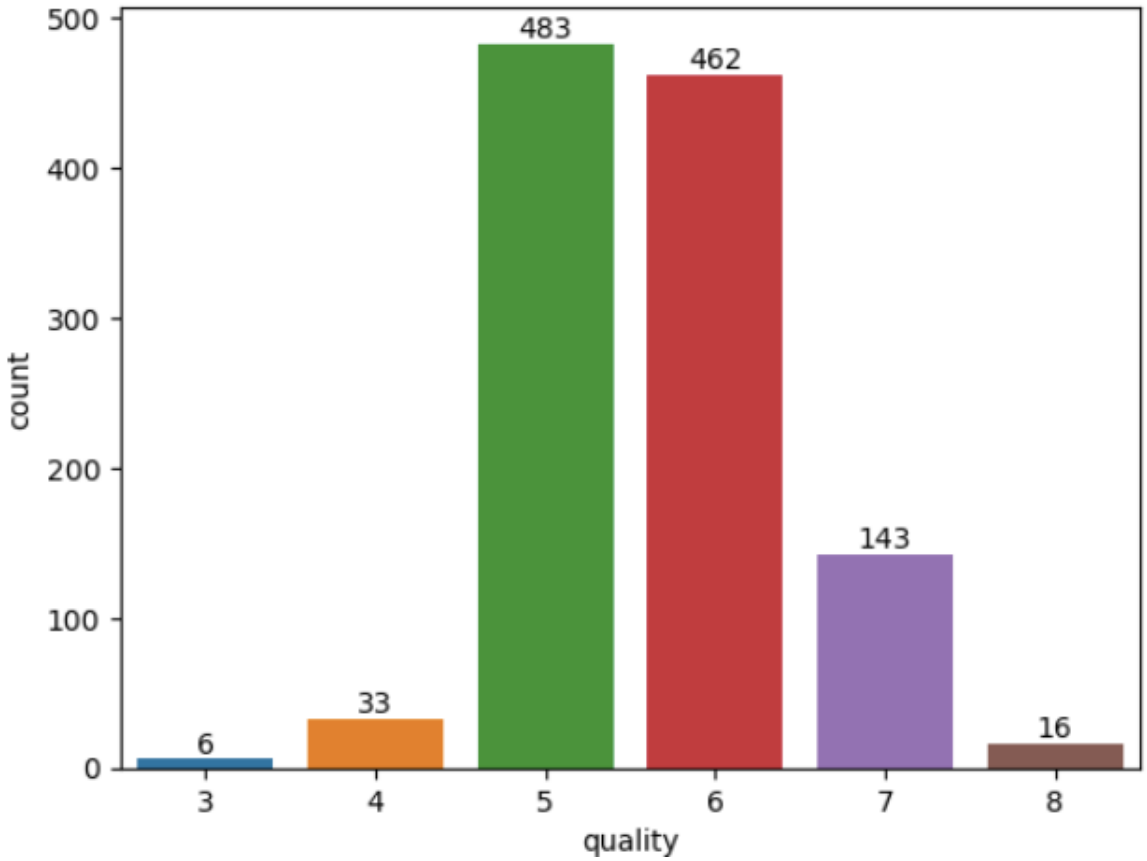


Figure 2.2: Quality Count plot

We can see from above countplot, that the data is highly imbalanced and quality of wine(0, 1, 2, 9, 10) are not present in the dataset.

In order to deal with this imbalanced, at first we are going to merge the quality labels (3 and 4) and (7 and 8) as shown below.

```
def wineQualityTransform(quality):
    wine_quality_transformation = {3: 0, 4: 0, 5:1, 6:2, 7: 3, 8: 3}
    return wine_quality_transformation[quality]

data_df["quality"] = data_df["quality"].apply(wineQualityTransform)
print(data_df.quality.value_counts())
```

Figure 2.3: Code for transformation of quality attribute

The resulting count plot is as shown below:

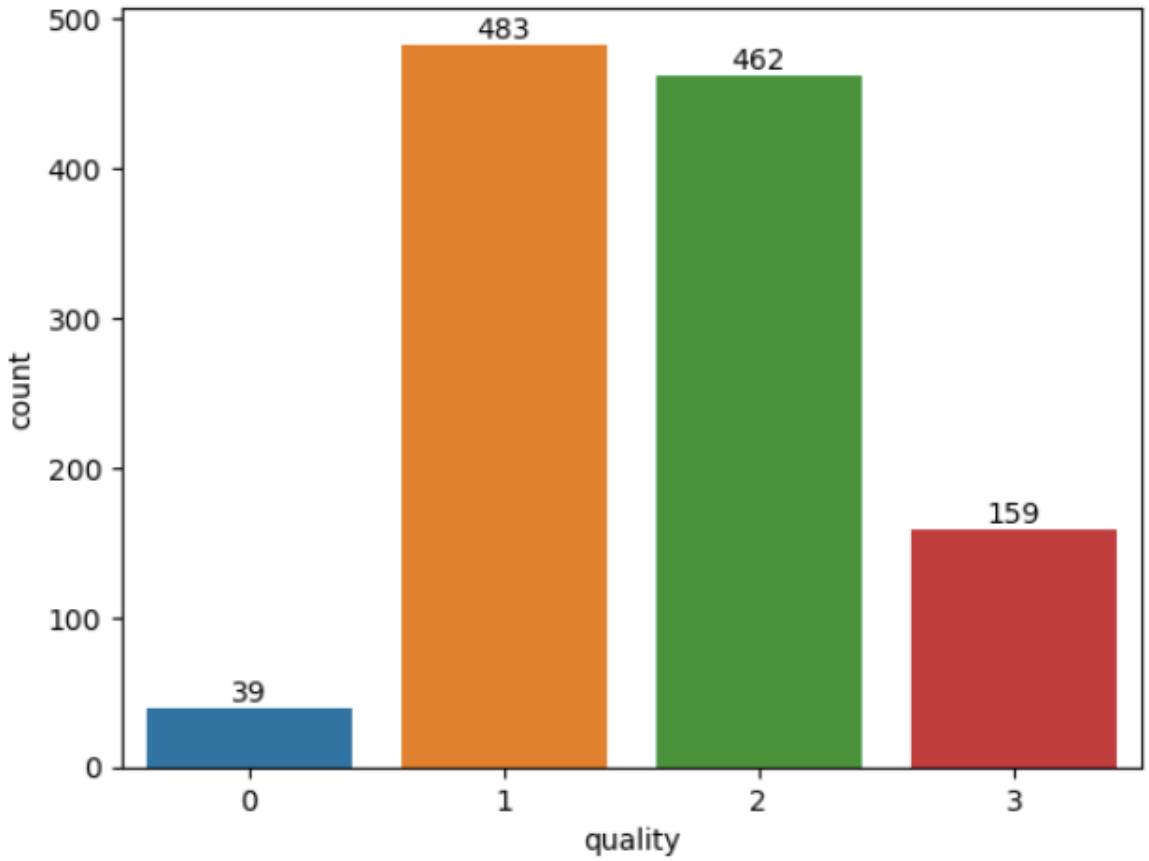


Figure 2.4: Quality Count plot after transformation

2.3.1 Train, validation and test dataset

We are going to use 80%, and 20% of the dataset as training, validation and test data. The countplot are as shown below:

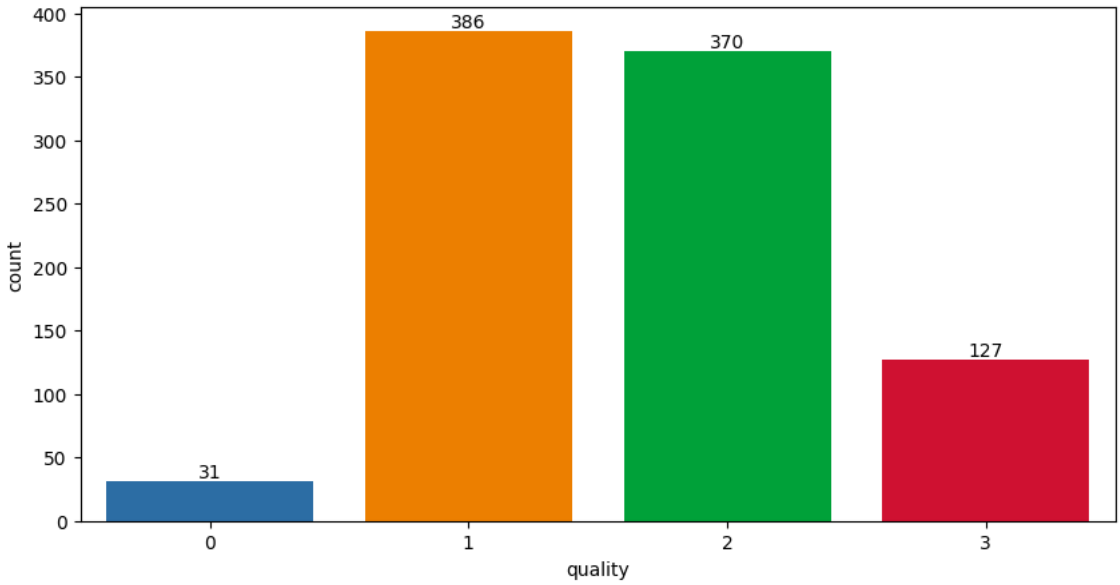


Figure 2.5: Train Quality Count plot after transformation

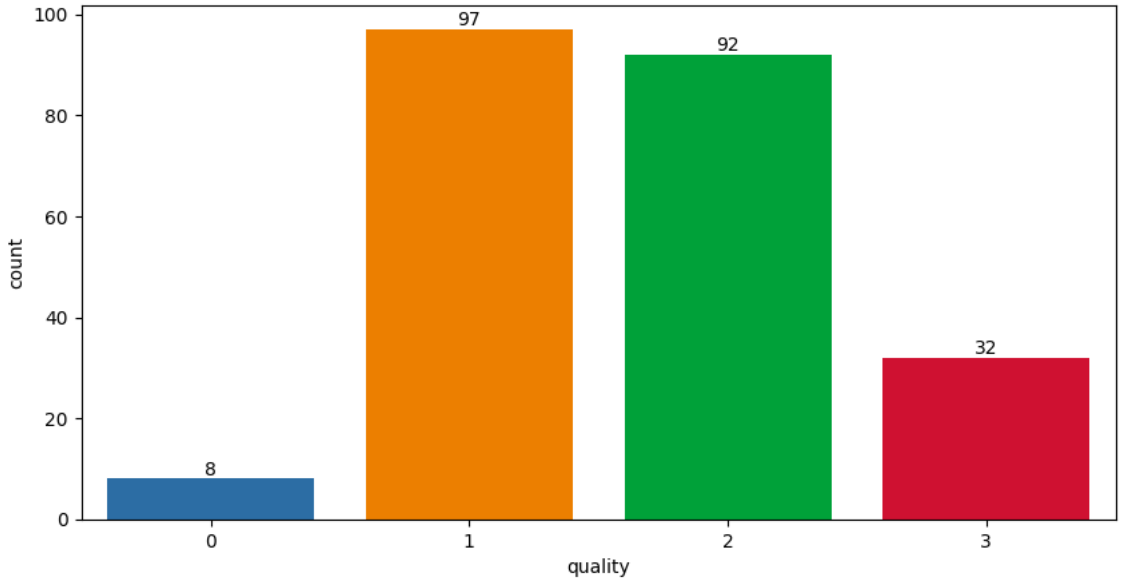


Figure 2.6: Test Quality Count plot after transformation

2.4 Literature Review

Several studies have been conducted to model wine preferences by data mining from physicochemical properties. For example, Cortez et al. (2009) used a dataset of 1599 red wines to develop a model that predicts wine quality based on 11 physicochemical properties. They used decision tree, support vector machine, and artificial neural network algorithms to model wine quality, and found that the artificial neural network algorithm had the highest accuracy. Similarly, Araújo and Juliano (2011) used a dataset of 649 Portuguese white wines to develop a model that predicts wine quality based on physicochemical properties. They used decision tree, logistic

regression, and artificial neural network algorithms to model wine quality, and found that the artificial neural network algorithm had the highest accuracy.

Other studies have focused on modeling wine preferences based on specific physicochemical properties. For example, Medina et al. (2017) used a dataset of 167 Spanish red wines to develop a model that predicts wine preference based on phenolic compounds. They used partial least squares regression and support vector machine algorithms to model wine preference, and found that the support vector machine algorithm had the highest accuracy. Similarly, Li et al. (2016) used a dataset of 80 Chinese red wines to develop a model that predicts wine preference based on aroma compounds. They used a fuzzy comprehensive evaluation method to model wine preference, and found that the method had a high accuracy.

Several studies have also investigated the relationship between physicochemical properties and wine sensory attributes. For example, Escudero et al. (2017) used a dataset of 100 Spanish red wines to study the relationship between physicochemical properties and wine sensory attributes. They found that several physicochemical properties, such as pH, alcohol content, and total phenolic content, were significantly correlated with wine sensory attributes, such as color intensity, aroma intensity, and astringency. Similarly, Jolliffe et al. (2016) used a dataset of 301 Australian white wines to study the relationship between physicochemical properties and wine sensory attributes. They found that several physicochemical properties, such as pH, titratable acidity, and residual sugar, were significantly correlated with wine sensory attributes, such as fruity, floral, and herbaceous aromas.

Data mining techniques have been widely used to model wine preferences based on physicochemical properties. Several studies have shown that these techniques can accurately predict wine quality and preference based on specific physicochemical properties, such as phenolic compounds and aroma compounds. Additionally, studies have found significant correlations between physicochemical properties and wine sensory attributes, suggesting that physicochemical properties play an important role in determining wine quality and consumer preferences. These findings have important implications for the wine industry, as they can be used to improve wine quality and tailor wines to specific consumer preferences.

2.5 Outcome

We expect the output to be the list of probabilites of belonging to a class[0, 1, 2 or 3]. Among these, the class with higher probability is out predicted class.

For the application of wine quality classification, we have developed a web app using django for backend and react for frontend whose screenshot is given below.

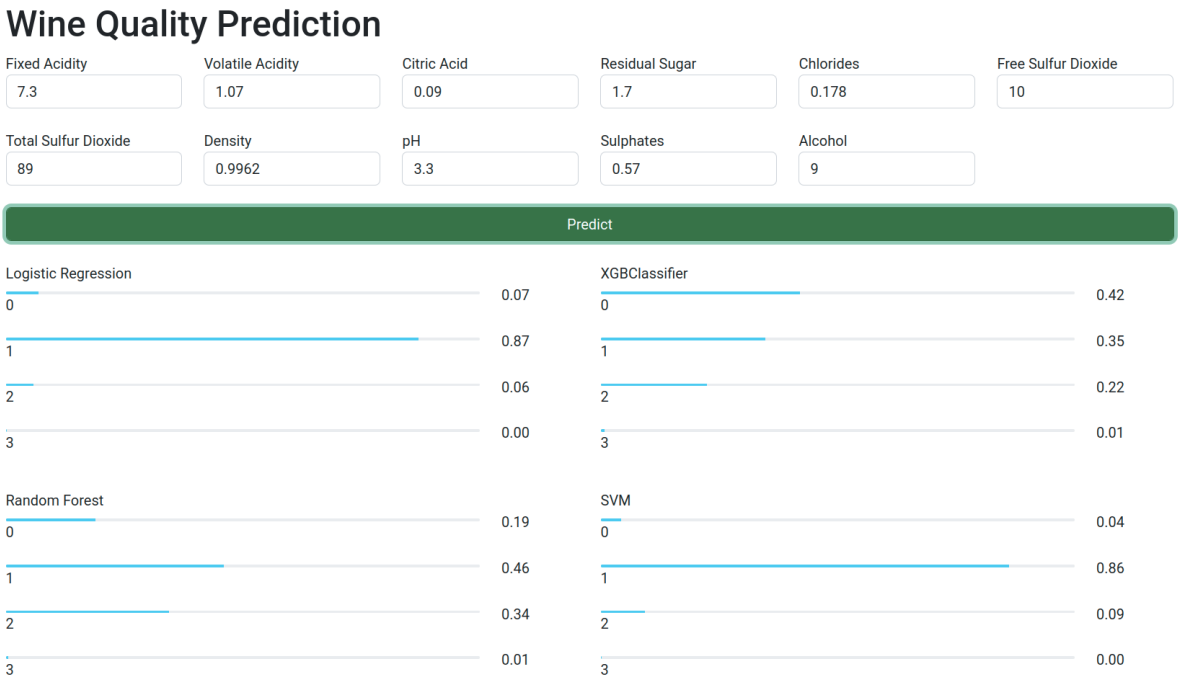


Figure 2.7: Frontend

Chapter 3

Project Management

3.1 Agile Development Methodology

To develop the wine quality classification, we decided to use Agile methodology. Agile methodology is an iterative approach to software development that emphasizes flexibility, collaboration, and rapid prototyping. We chose Agile methodology because it is well-suited to projects that require frequent feedback and adaptation to changing requirements.

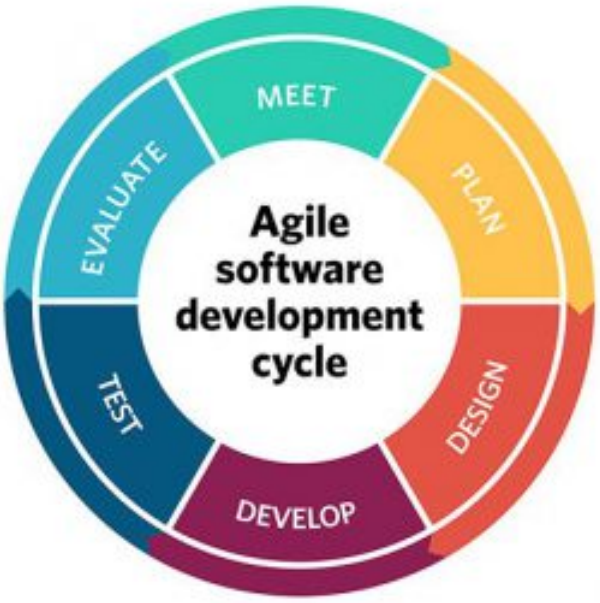


Figure 3.1: Agile Development Methodology

3.2 Trello

We have used Trello's default agile board for task management.

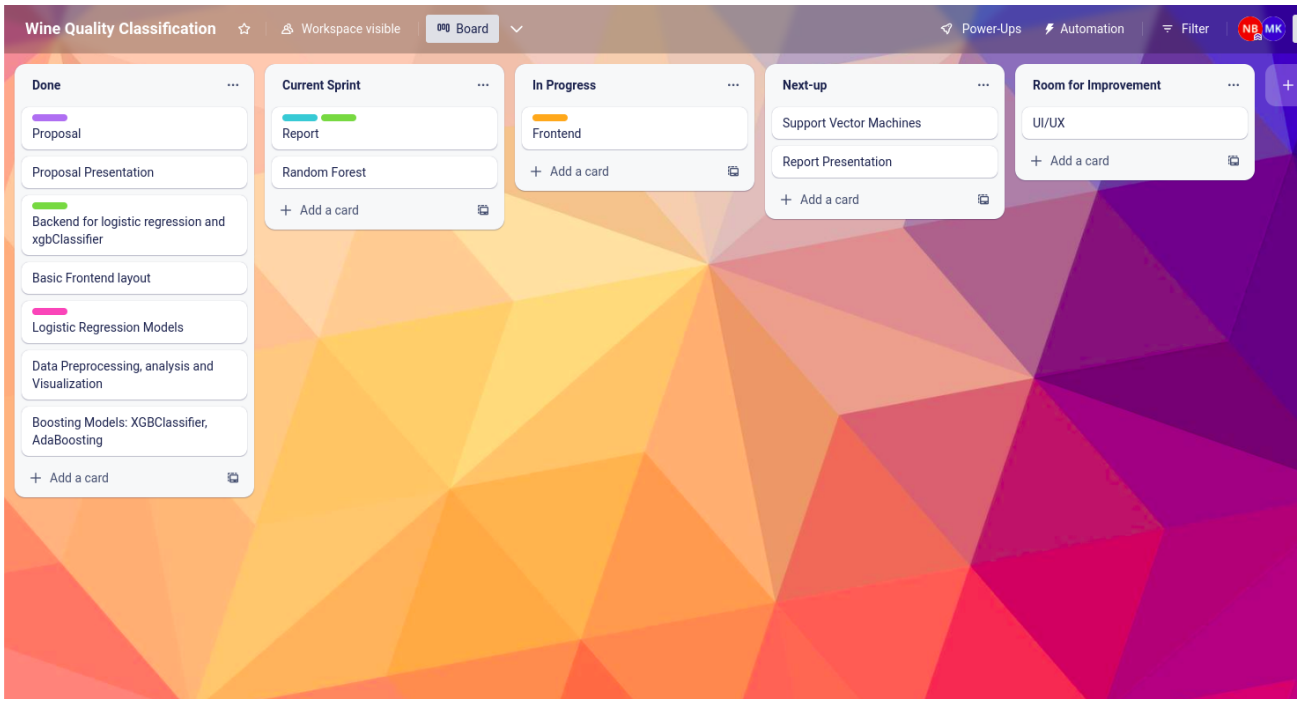


Figure 3.2: Trello

3.3 Discord

We have used Discord server for effective communication and team meetings.

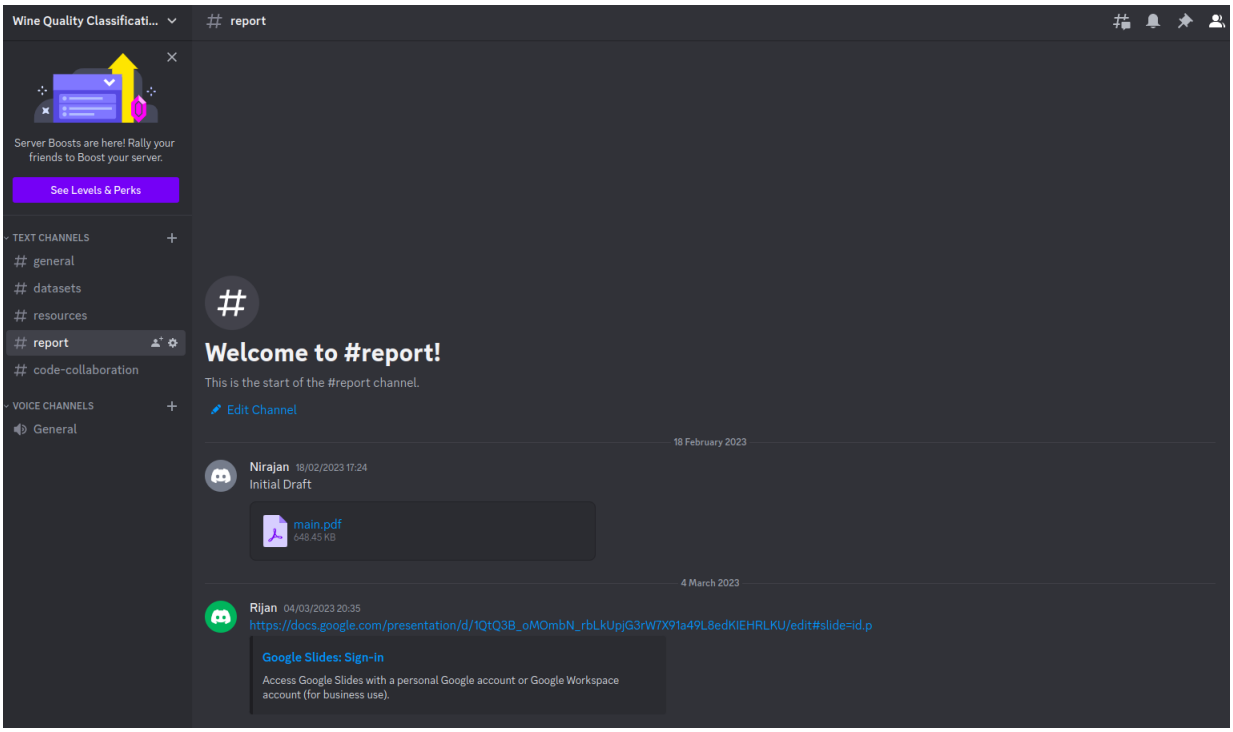


Figure 3.3: Discord

3.4 Excel

We have used Excel for experiment tracking.

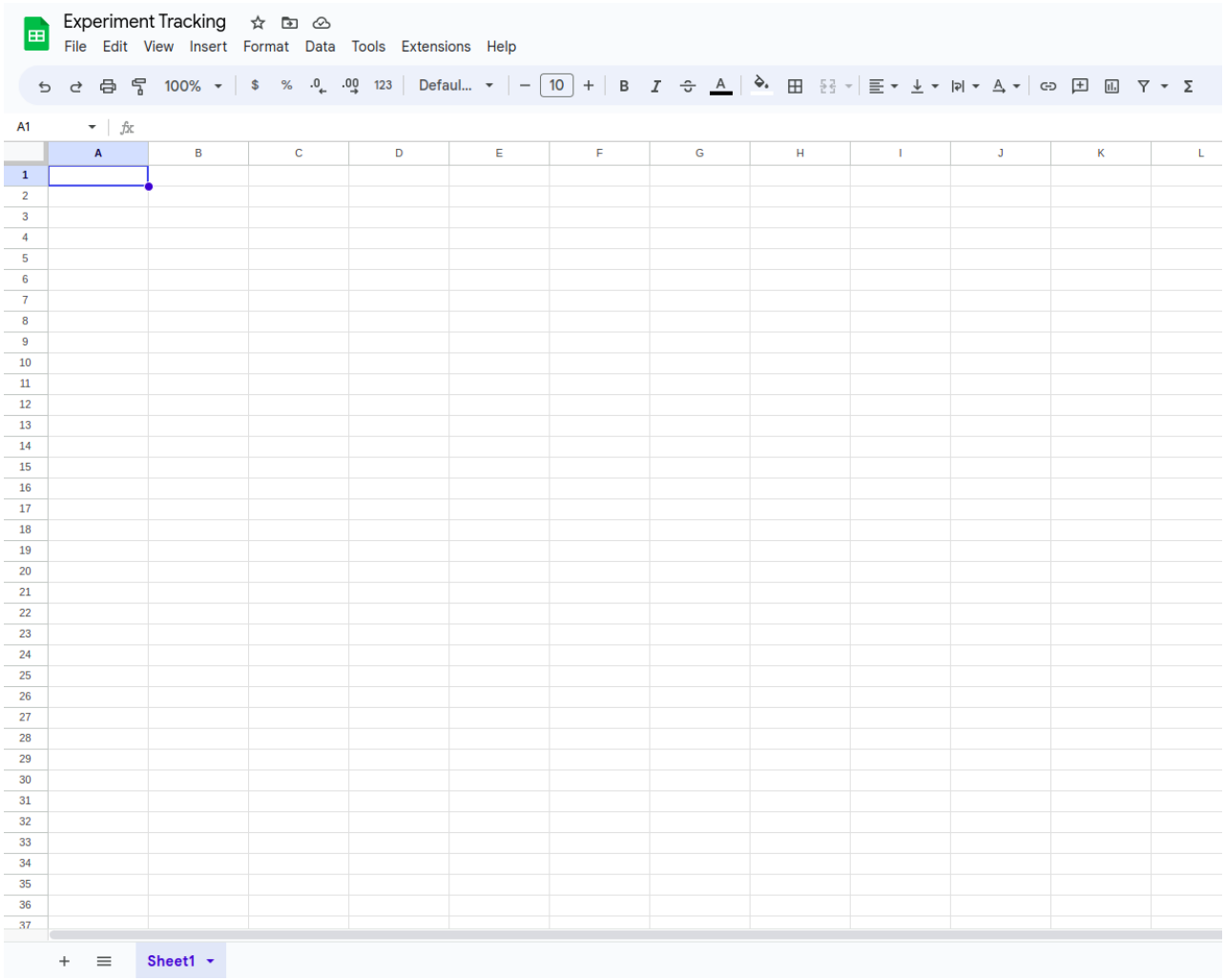


Figure 3.4: Google Sheet

Chapter 4

Feature Engineering and Exploratory Data Analysis

This chapter discusses the process of feature engineering, which involves techniques such as feature scaling and feature transformation are described, which can help to normalize the range and distribution of features. Data visualization is also emphasized as a tool for gaining insights into the data, with examples given of how boxplots, distribution plots, and heat maps can be used to identify patterns and relationships in the data.

4.1 Dataset Exploration

Basic statistics of each features are shown below:

stats	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
mean	8.311	0.5313	0.264	2.532	0.0869	15.615
std	1.747	0.179	0.196	1.355	0.0477	10.250
min	4.600	0.120000	0.000	0.900	0.0120	1.00
25%	7.100	0.39250	0.0900	1.900	0.070	7.000
50%	7.900	0.5200	0.2500	2.200	0.0790	13.000
75%	9.100	0.6400	0.4200	2.6000	0.090	21.000
max	15.900	1.580	1.00	15.50	0.6110	68.00

Table 4.1: Data Description 1

stats	total sulfur dioxide	density	pH	sulphates	alcohol
mean	45.914698	0.996730	3.311015	0.657708	10.442111
std	32.782130	0.001925	0.156664	0.170399	1.082196
min	6.000000	0.990070	2.740000	0.330000	8.400000
25%	21.000000	0.995570	3.205000	0.550000	9.500000
50%	37.000000	0.996680	3.310000	0.620000	10.200000
75%	61.000000	0.997845	3.400000	0.730000	11.100000
max	289.000000	1.003690	4.010000	2.000000	14.900000

Table 4.2: Data Description 2

The correlation between data is shown in following heat map.

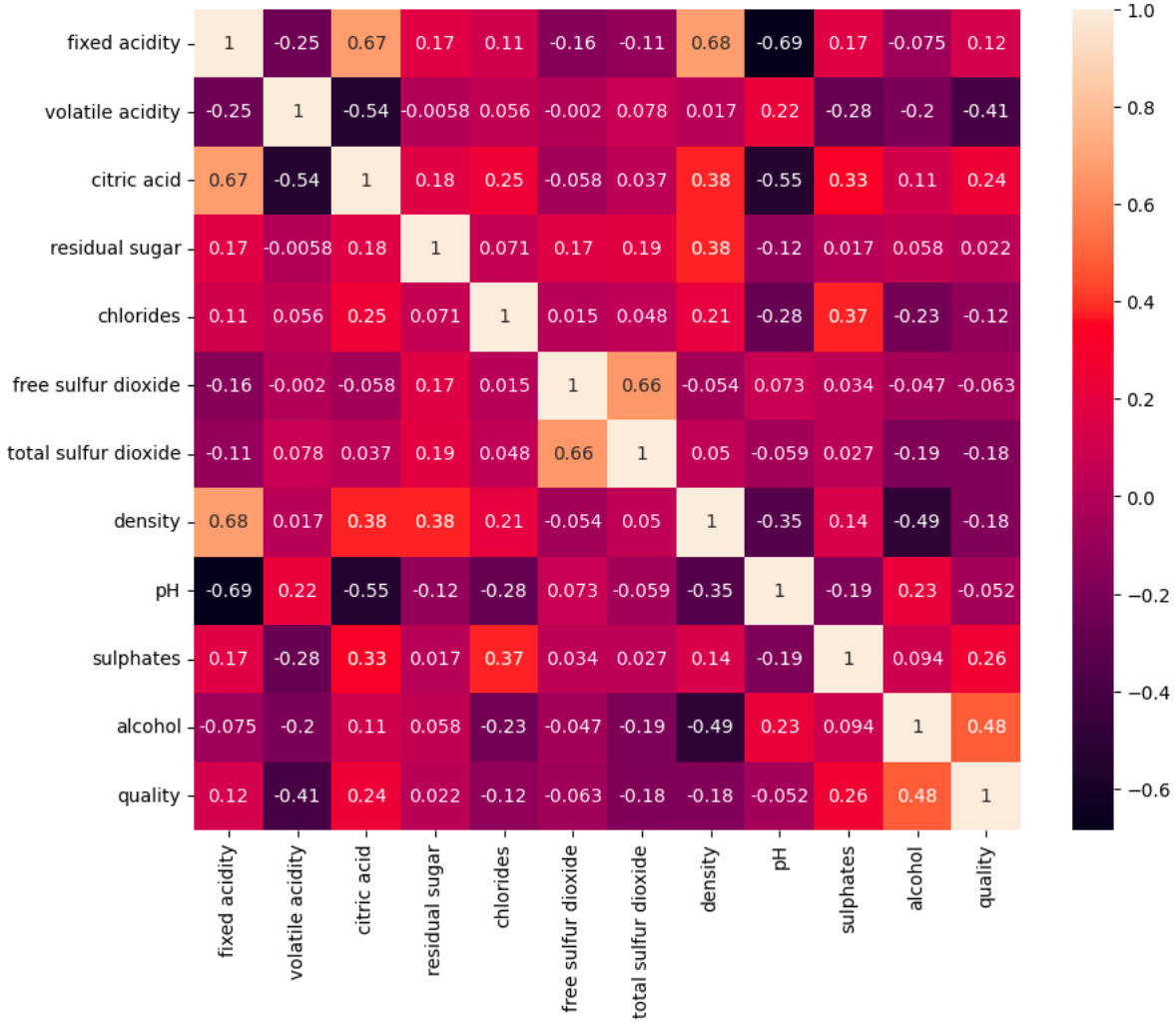


Figure 4.1: Correlation Heat Map

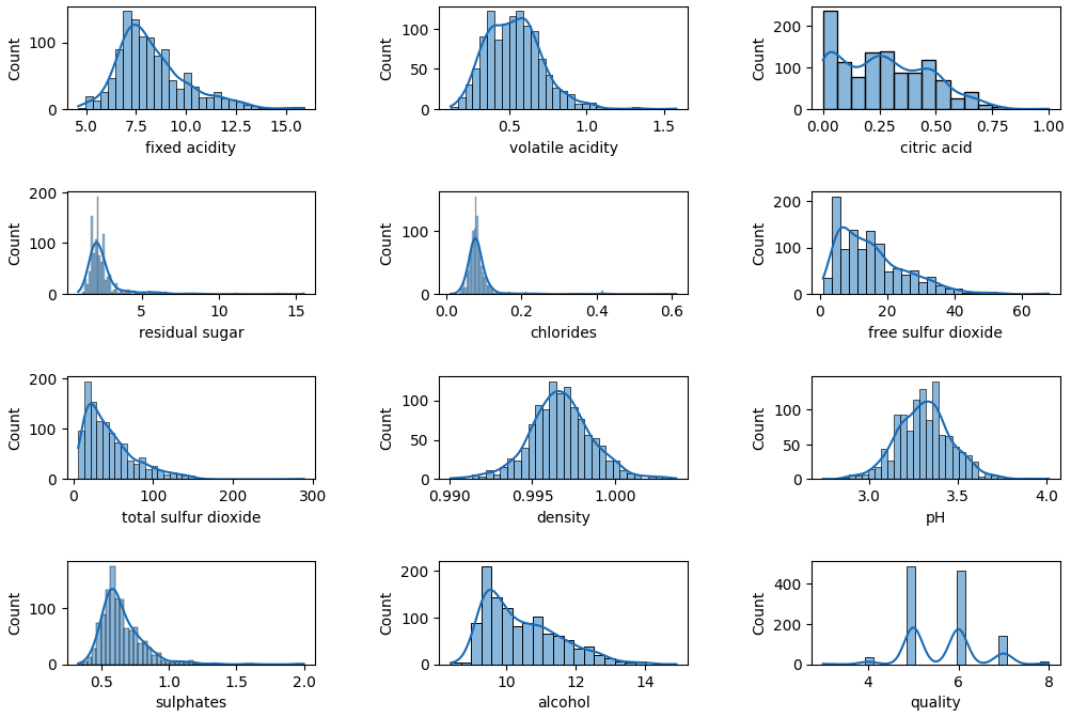


Figure 4.2: Distribution Plot of the entire datasets

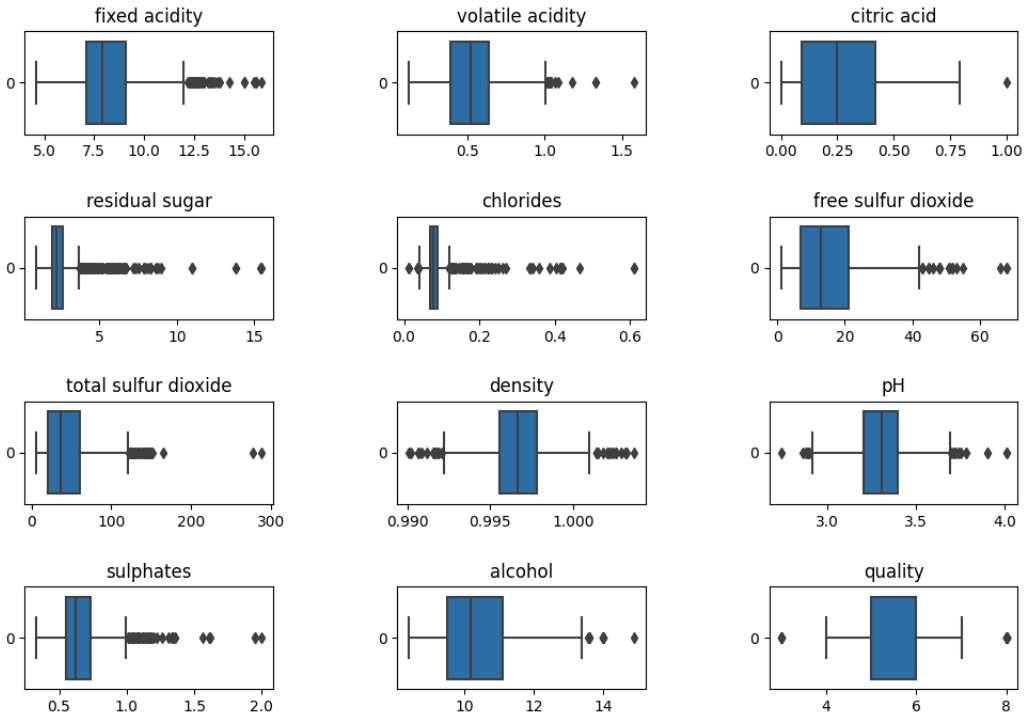


Figure 4.3: Box Plot of the entire datasets

4.2 Min Max Scaling

Min max scaling was used as it scales all the data features in the range of 0 and 1 due to which it becomes easier to transform the data using log transformation and boxcox transformation. We can see how data have been normalized and outliers have been handled from following plots

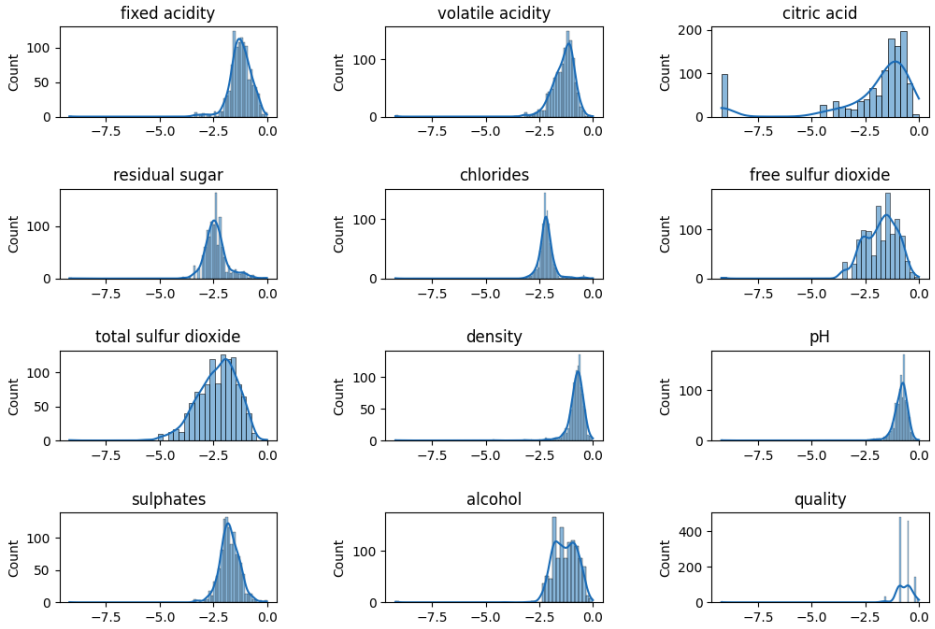


Figure 4.4: Distribution Plot of log transformed data after minmax scaling

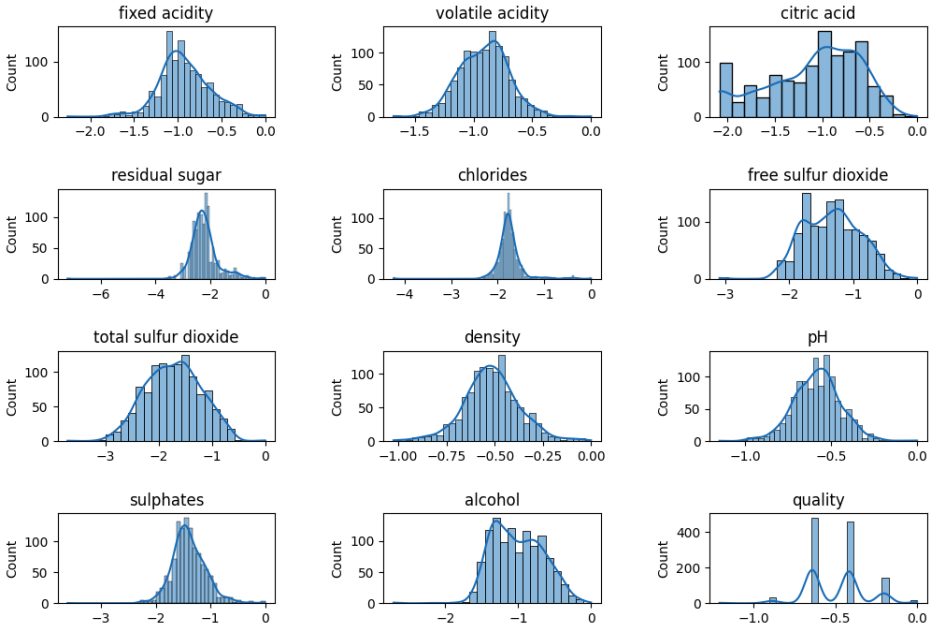


Figure 4.5: Distribution Plot of boxcox transformed data after minmax scaling

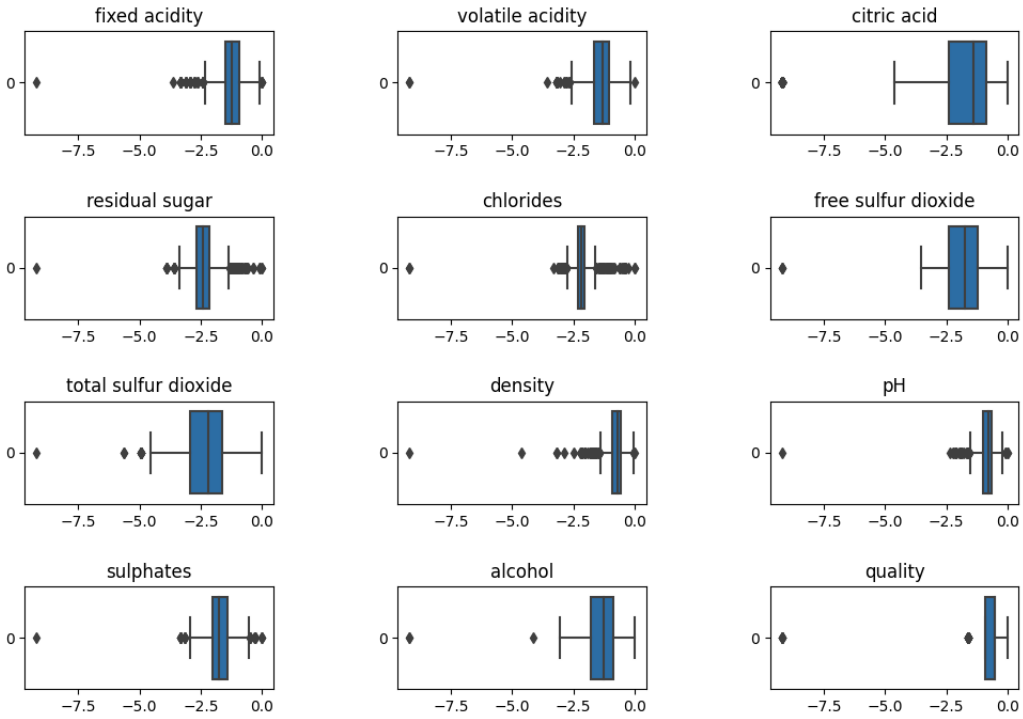


Figure 4.6: Box Plot of log transformed data after minmax scaling

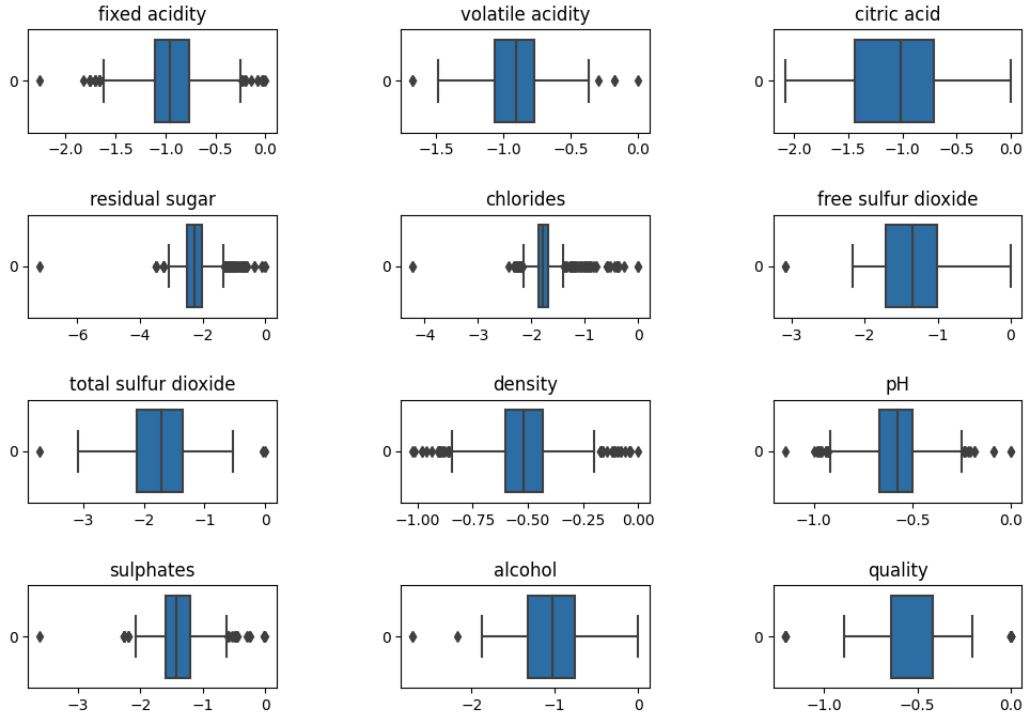


Figure 4.7: Box Plot of boxcox transformed data after minmax scaling

Chapter 5

Model Development and Evaluation

This section will cover the techniques used for developing models and evaluating their performance, including preprocessing methods such as scaling and transformation. It will also compare different machine learning algorithms for model development and training, including Logistic Regression, Support Vector Machines (SVM), Random Forest, and Boosting algorithms.

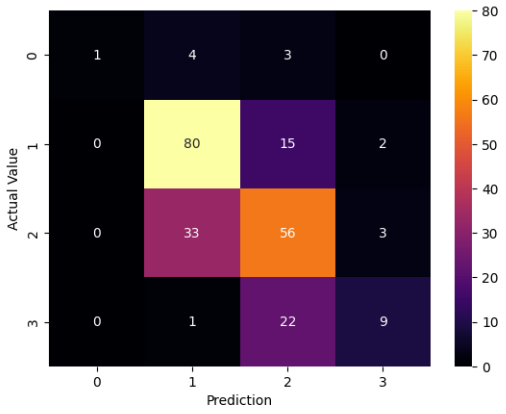
5.1 Logistic Regression

The data was initially split into training and testing sets, and then a log transformation was applied to the data followed by standard scaling. The resulting transformed data was used as input to a logistic regression model. During model development, different parameter and hyperparameter combinations were tested for the logistic regression model.

The hyperparameters and evaluation metrics of the model are given below:

multi_class	solver	micro avg	macro avg	weighted avg
ovr	newton-cg	0.64	0.49	0.62
multinomial	newton-cg	0.66	0.53	0.65

Table 5.1: Logistic regression hyperparameter and evaluation

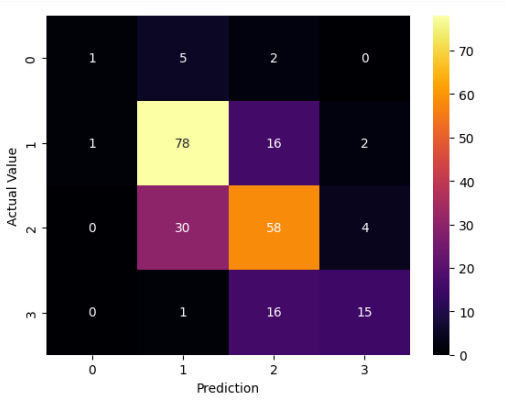


(a) Confusion matrix

	precision	recall	f1-score	support
0	1.00	0.12	0.22	8
1	0.68	0.82	0.74	97
2	0.58	0.61	0.60	92
3	0.64	0.28	0.39	32
accuracy			0.64	229
macro avg	0.73	0.46	0.49	229
weighted avg	0.65	0.64	0.62	229

(b) Classification report

Figure 5.1: Confusion matrix and classification report for logistic regression model with multi_class = "ovr"



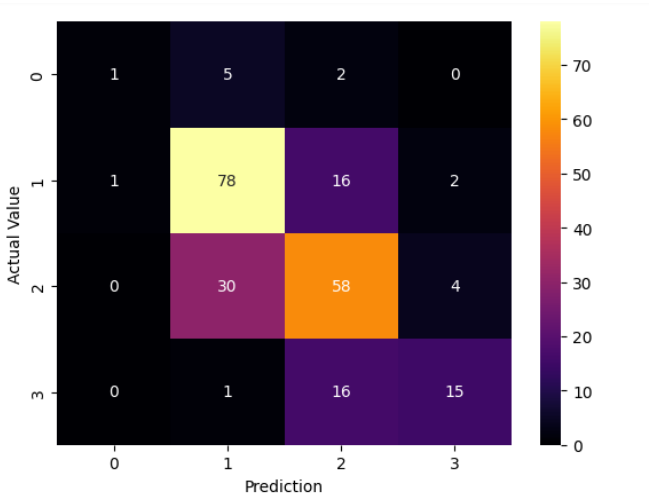
(a) Confusion matrix

	precision	recall	f1-score	support
0	0.50	0.12	0.20	8
1	0.68	0.80	0.74	97
2	0.63	0.63	0.63	92
3	0.71	0.47	0.57	32
accuracy			0.66	229
macro avg	0.63	0.51	0.53	229
weighted avg	0.66	0.66	0.65	229

(b) Classification report

Figure 5.2: Confusion matrix and classification report for logistic regression model with multi_class = "multinomial"

Smote oversampling was used to handle the imbalanced data. But the accuracy got worse instead. The results after the oversampling was as follow:



(a) Confusion matrix

	precision	recall	f1-score	support
0	0.50	0.12	0.20	8
1	0.68	0.80	0.74	97
2	0.63	0.63	0.63	92
3	0.71	0.47	0.57	32
accuracy			0.66	229
macro avg	0.63	0.51	0.53	229
weighted avg	0.66	0.66	0.65	229

(b) Classification report

Figure 5.3: Confusion matrix and classification report for logistic regression model after smote oversampling

5.2 Boosting

To develop models using boosting algorithms, the initial step was to split the data into training and testing sets. After that, the training dataset was oversampled using the SMOTE technique.

The next step involved fitting a min-max scaler to the training data, and then scaling both the training and testing data using this scaler. A log transformation was then applied to the preprocessed data.

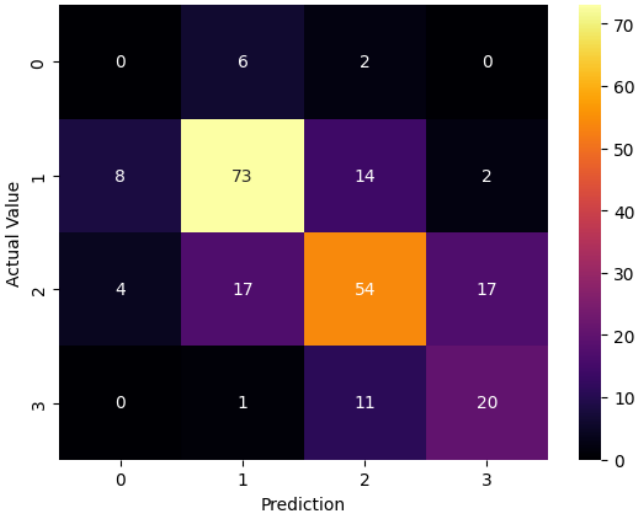
Various boosting algorithms such as XGBClassifier and AdaBoost were utilized to train models on the preprocessed training data. The performance of these models was then assessed, and the best performing algorithm was determined.

5.2.1 XGBClassifier

The RandomizedSearchCV was used to tune the hyperparameters of the XGBClassifier. The results were as follow:

id	learning_rate	max_depth	n_estimators	subsample	micro	macro	weighted
xgb_v1	0.0824	16	113	0.5394	0.64	0.49	0.65
xgb_v2	0.0903	19	127	0.6520	0.63	0.48	0.63

Table 5.2: XGBClassifier hyperparameter and evaluation metrics

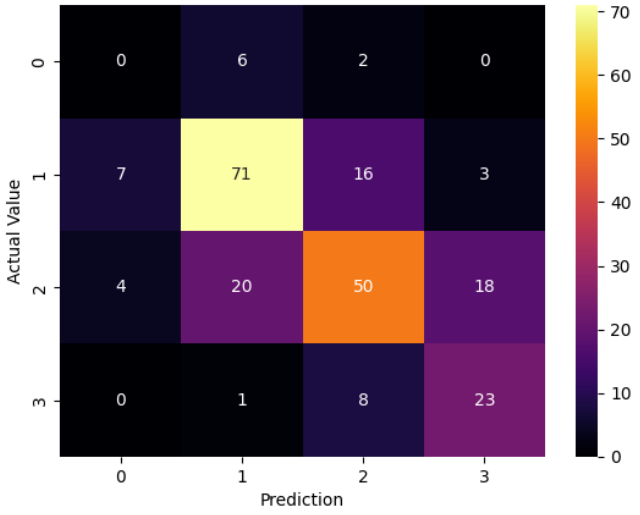


(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.75	0.75	0.75	97
2	0.67	0.59	0.62	92
3	0.51	0.62	0.56	32
accuracy			0.64	229
macro avg	0.48	0.49	0.49	229
weighted avg	0.66	0.64	0.65	229

(b) Classification report

Figure 5.4: Confusion matrix and classification report for XGBClassifier xgb_v1



(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.72	0.73	0.73	97
2	0.66	0.54	0.60	92
3	0.52	0.72	0.61	32
accuracy			0.63	229
macro avg	0.48	0.50	0.48	229
weighted avg	0.64	0.63	0.63	229

(b) Classification report

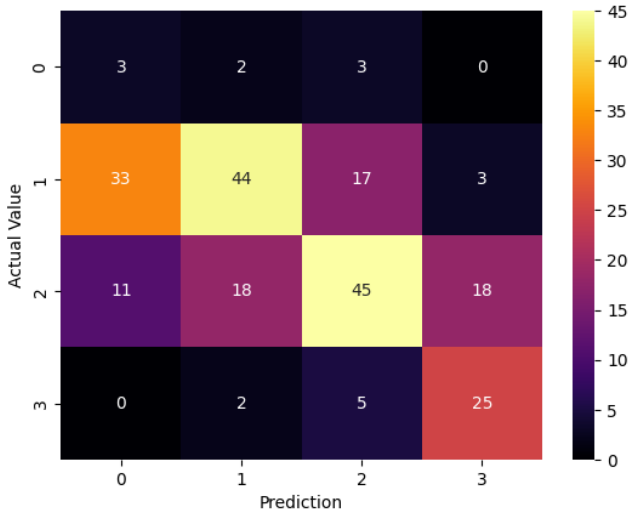
Figure 5.5: Confusion matrix and classification report for XGBClassifier xgb_v2

5.2.2 AdaBoost, Gradient Boosting and LGBMClassifier

The same preprocessing step were followed and the results were as follow:

model	micro	macro	weighted
AdaBoost	0.51	0.46	0.55
Gradient Boosting	0.61	0.50	0.61
xgb_v1	0.64	0.49	0.65

Table 5.3: Boosting algorithms evaluation metrics

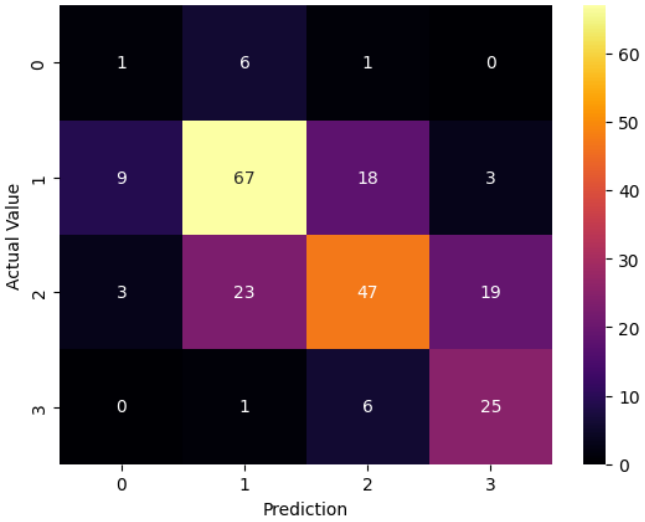


(a) Confusion matrix

	precision	recall	f1-score	support
0	0.06	0.38	0.11	8
1	0.67	0.45	0.54	97
2	0.64	0.49	0.56	92
3	0.54	0.78	0.64	32
accuracy			0.51	229
macro avg	0.48	0.52	0.46	229
weighted avg	0.62	0.51	0.55	229

(b) Classification report

Figure 5.6: Confusion matrix and classification report for AdaBoost

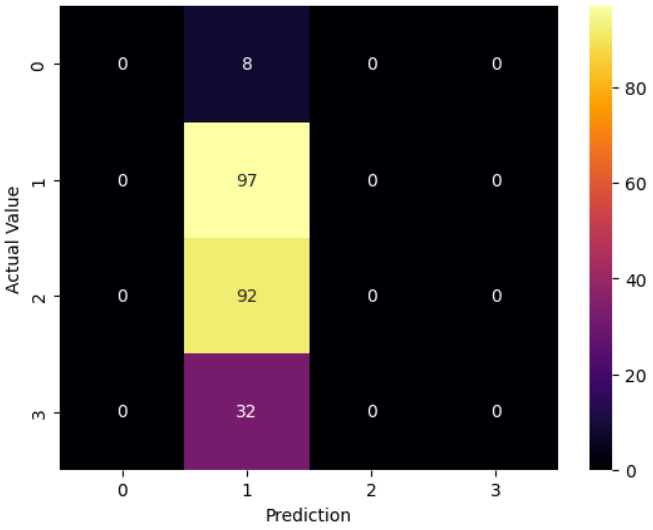


(a) Confusion matrix

	precision	recall	f1-score	support
0	0.08	0.12	0.10	8
1	0.69	0.69	0.69	97
2	0.65	0.51	0.57	92
3	0.53	0.78	0.63	32
accuracy			0.61	229
macro avg	0.49	0.53	0.50	229
weighted avg	0.63	0.61	0.61	229

(b) Classification report

Figure 5.7: Confusion matrix and classification report for Gradient Boosting Classifier



(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.42	1.00	0.60	97
2	0.00	0.00	0.00	92
3	0.00	0.00	0.00	32
accuracy			0.42	229
macro avg	0.11	0.25	0.15	229
weighted avg	0.18	0.42	0.25	229

(b) Classification report

Figure 5.8: Confusion matrix and classification report for LGBM Classifier

5.3 Random Forest and SVM

The first step in developing models using random forest and SVM classifiers involved splitting the data into training and testing sets. Next, a log transformation was applied to the data to normalize it.

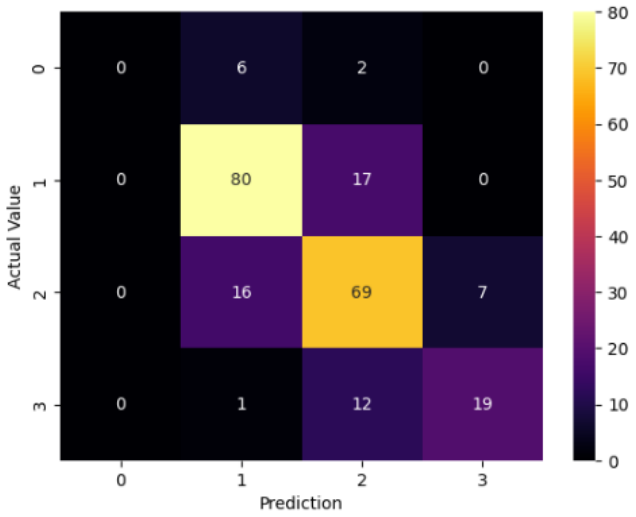
After normalization, the data was scaled using the Robust Scaler, which was specifically chosen for its ability to handle outliers in the data. The preprocessed data was then used to train models using the random forest and SVM classifiers.

5.3.1 Random Forest

The hyperparameter tuning result and evaluation metrics for each model are shown below.

id	n_estimators	max_depth	min_samples_split	min_samples_leaf	micro	macro	weighted
1	100	None	None	1	0.73	0.54	0.72
2	200	20	2	1	0.69	0.51	0.68

Table 5.4: Boosting algorithms evaluation metrics

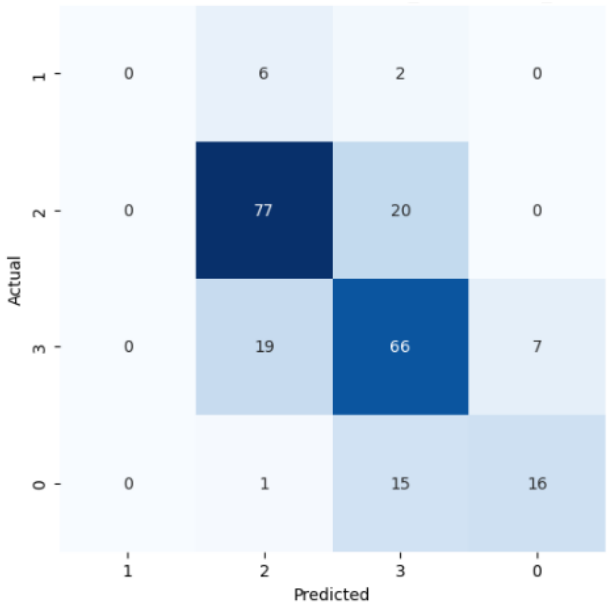


(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.78	0.82	0.80	97
2	0.69	0.75	0.72	92
3	0.73	0.59	0.66	32
accuracy			0.73	229
macro avg	0.55	0.54	0.54	229
weighted avg	0.71	0.73	0.72	229

(b) Classification report

Figure 5.9: Confusion matrix and classification report for Random Forest Classifier 1



(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.75	0.79	0.77	97
2	0.64	0.72	0.68	92
3	0.70	0.50	0.58	32
accuracy			0.69	229
macro avg	0.52	0.50	0.51	229
weighted avg	0.67	0.69	0.68	229

(b) Classification report

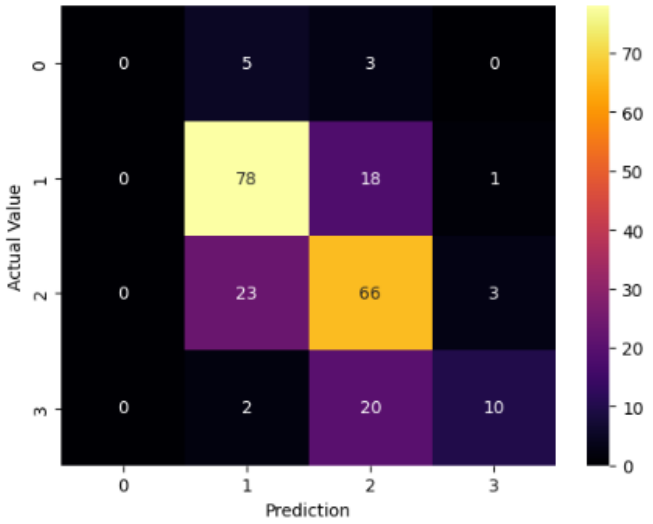
Figure 5.10: Confusion matrix and classification report for Random Forest Classifier 2

5.4 SVM

The hyperparameter tuning result and evaluation metrics for each model are shown below.

id	C	kernel	micro	macro	weighted
1	1	ovr	0.67	0.46	0.65
2	1	linear	0.62	0.44	0.61

Table 5.5: Boosting algorithms evaluation metrics

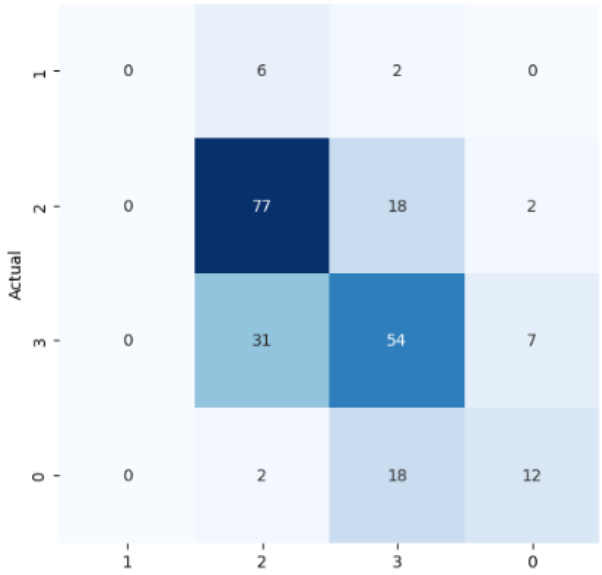


(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.72	0.80	0.76	97
2	0.62	0.72	0.66	92
3	0.71	0.31	0.43	32
accuracy			0.67	229
macro avg	0.51	0.46	0.46	229
weighted avg	0.65	0.67	0.65	229

(b) Classification report

Figure 5.11: Confusion matrix and classification report for SVM id = 1



(a) Confusion matrix

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.66	0.79	0.72	97
2	0.59	0.59	0.59	92
3	0.57	0.38	0.45	32
accuracy			0.62	229
macro avg	0.46	0.44	0.44	229
weighted avg	0.60	0.62	0.61	229

(b) Classification report

Figure 5.12: Confusion matrix and classification report for SVM id = 2

Chapter 6

Conclusion

In conclusion, this project involved exploring various machine learning algorithms for wine quality classification. We also learned several data preprocessing techniques, such as scaling using different scalers, and applying log and boxcox transformations to normalize the data.

After comparing the performance of several models, we found that the random forest model with id = 1 performed the best, with a micro F1 score of 0.73. This suggests that the random forest algorithm is effective for classifying wine quality and can be used in practical applications.

Overall, this project highlights the importance of selecting appropriate preprocessing techniques and evaluating various machine learning algorithms to develop an accurate and effective model for wine quality classification.