Dissertation on
A Predictive analysis on Quality of Red Wines Using Various Machine-Learning Models - An Approach

Thesis submitted towards partial fulfilment
of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by
Arihant Kumar Sekhani

EXAMINATION ROLL NO.: M4CWE24019
UNIVERSITY REGISTRATION NO.: 160383 of 2021-23

Under the guidance of,
Joydeep Mukherjee

School of Education Technology
Jadavpur University

Course affiliated to,
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2024

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"A Predictive analysis on Quality of Red Wines Using Various Machine-Learning Models - An Approach"** is a bonafide work carried out by Arihant Kumar Sekhani under our supervision and guidance for partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering) in School of Education Technology, during the academic session 2022-2023.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

<u>CERTIFICATE OF APPROVAL</u> **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination for evaluation of Thesis ---

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his/her **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: **Arihant Kumar Sekhani**

EXAMINATION ROLL NUMBER: **M4CWE24019**

THESIS TITLE: **A Predictive analysis on Quality of Red Wines Using Various Machine-Learning Models - An Approach**

SIGNATURE: DATE:

Acknowledgement

I feel extremely glad in presenting this thesis at School of Education Technology, Jadavpur University, Kolkata, in the partial fulflment of the requirements for the degree of Master of Technology in IT (Courseware Engineering). It is a great pleasure for me to express my respect and a deep sense of gratitude to my M. Tech superui sor **Mr. Joydeep Mukherjee**, Department of School of Education Technology, Jadavpur University, for his wisdom, vision, expertise, guidance, enthusiastic involvement during my M.Tech. It has been a significant privilege and joy for me to inquire about it under his watch.

I am grateful to **(Prof) Dr. Matangini Chattopaddhyay**, Director of the School of Education Technology, for his support, encouragement and timely advice. I an really indebted to **Dr. Saswati Mukherjee** for their continuous support during the entire course of research work. Their advice and support were highly inspirational and motivating. I would also like to take this opportunity to pay my thanks to all my classmates of M. Tech IT (Courseware Engineering) and support stuffs who motivated and helped me to complete my research work successfully.

With regards,
Date:

ARIHANT KUMAR SEKHANI
M. Tech. I.T. (Courseware Engineering)
School Of Education Technology
Jadavpur University, Kolkata-700032

TABLE OF CONTENTS Page No.

Executive Summary

This study combines a variety of machine learning approaches to improve the prediction of red wine quality. The main goal is to create reliable and effective algorithms that can forecast red wine quality based on its characteristics, improving customer happiness and health. The study uses a dataset that includes important red wine characteristics and was gathered from trustworthy sources.

Predictive models are built using Random Forest, Support Vector Machine (SVM), and Naive Bayes, three well-known machine learning approaches. Data preparation, model training, and rigorous assessment are all part of the research's careful approach, which also comprises distinct training and testing sets. To assess the efficacy of each approach, performance indicators including accuracy, precision, recall, and F1-score are generated.

Finding the best strategy for predicting red wine quality is based on comparing the results of the three strategies on the testing set. Initial expectations are shaped by training set results, but the testing set offers a practical evaluation of each model's generalisation potential. The study also proposes a novel method for improving both efficiency and accuracy by combining key elements from other methodologies.

In the end, this study advances present research worker knowledge of how machine learning determines the quality of red wines. The goal of the study is to offer insightful information for more accurately forecasting the quality of red wine by selecting the most effective technique through thorough review and suggesting the amalgamation of attributes. To

pursue consumer welfare and the promotion of quality assurance practices within the realm of wine consumption.

Chapter 1

1. Introduction

1.1 Background and Motivation

Analysing the quality of red wines using machine learning models is a fascinating area of study that combines the worlds of data science and oenology (the science of wine). Here's some background and potential motivations for conducting such predictive analysis:

Increasing Demand for High-Quality Wine: With the growing popularity of wine consumption worldwide, there's an increasing demand for high-quality wines. Winemakers are constantly striving to produce better wines to meet consumer preferences and stand out in the market.

Complexity of Wine Evaluation: Assessing wine quality is a complex task influenced by numerous factors such as grape variety, climate, soil, fermentation process, aging, etc. Traditional methods of wine evaluation rely heavily on expert wine tasters, which can be subjective and time-consuming.

Data Availability: There's a wealth of data available on various attributes of wines, such as acidity, alcohol content, residual sugar, pH level, etc. This data, coupled with advancements in machine learning algorithms, presents an opportunity to develop predictive models that can accurately predict wine quality based on these attributes.

Cost and Time Efficiency: Developing machine learning models to predict wine quality can potentially offer a cost-effective and time-efficient alternative to traditional wine evaluation methods. Once trained, these models can quickly assess the quality of wines, allowing winemakers to make timely decisions during the production process.

Exploration of Machine Learning Techniques: Wine quality prediction serves as an excellent use case for exploring and comparing various machine learning techniques. Researchers can experiment with different algorithms such as regression, decision trees, random forests, support vector machines, neural networks, etc., to determine which ones perform best for this particular task.

Insights into Wine Production: Analysing the factors that contribute to wine quality can provide valuable insights for wine producers. By understanding the relationship between different wine attributes and quality, producers can optimize their production processes to consistently produce high-quality wines.

Educational Purposes: Research studies on predictive analysis of wine quality can also serve educational purposes by providing valuable learning resources for students and professionals in the fields of data science, machine learning, and viticulture.

In summary, conducting predictive analysis on the quality of red wines using machine learning models offers several potential benefits, including improved wine quality assessment, cost and time efficiency, exploration of machine learning techniques, insights into wine production, and educational value.

1.2 Research Objectives

The primary objective of this thesis is to develop robust and efficient predictive models for red wine quality assessment using different machine learning techniques. The research aims to:

Explore the efficacy of Random Forest, Support Vector Machine (SVM), and Naïve Bayes in predicting red wine quality based on its attributes.
Investigate the impact of feature extraction and fusion from different techniques on enhancing prediction accuracy and efficiency.
Provide insights into the practical implementation of machine learning for quality assurance in the wine industry.

1.3 Scope and Significance

This research is focused on the prediction of red wine quality using machine learning techniques, utilizing a dataset containing relevant attributes. The study's significance lies in its potential to provide consumers with a reliable tool for assessing the quality of red wine before consumption. Additionally, it contributes to the broader field of quality prediction and assurance through the application of advanced machine learning methodologies.

1.4 Organization of Thesis

The remainder of this thesis is structured as follows:

❖ Chapter 2 provides a comprehensive background on the topic, including wine quality assessment, attributes for quality prediction, the role of machine learning, feature engineering, performance metrics, and potential future directions for research.

❖ Chapter 3 provides review relevant literature and research on wine quality prediction using various machine learning models to provide the path for future references.

❖ Chapter 4 provides outlines of different machine learning approach for the research, including data collection and pre-processing methods.

❖ Chapter 5 provides insides on the machine learning models which are used, such as Random Forest, Support Vector Machine, Naïve Bayes, Logistic Regression, k-Nearest Neighbor, and Decision Tree Classifier, explaining their relevance to your research.

❖ Chapter 6 discusses the evaluation metrics used to assess the performance of the machine learning models, provides a comparative analysis of their results, and discusses the implications of the findings.

- ❖ Chapter 7 presents the actual results obtained from the experiments, which may include outputs from the machine learning models.

- ❖ Chapter 8 involves a detailed comparison and analysis of the results, possibly comparing between different machine learning models.

- ❖ Chapter 9 summarizes the main findings of the research, draws conclusions based on the results, and may offer insights or recommendations for future research.

- ❖ Chapter 10 provides a list of all the sources and references cited in the research work.

By examining the intersection of wine quality assessment and machine learning, this thesis aims to contribute to both the theoretical understanding and practical implementation of predictive models in enhancing the quality of red wine consumption.

Chapter 2

2. Background Concept

2.1 Introduction

The background concept of conducting a predictive analysis on the quality of red wines using various machine learning models involves leveraging data-driven approaches to assess and potentially enhance the quality of wine production. Here's a breakdown of the key components:

2.2 Wine Quality Assessment

Wine quality assessment is a multifaceted process that involves the evaluation of sensory, chemical, and physical attributes of wine. Historically, human sensory evaluation has been the primary method for wine quality assessment. However, this approach is subjective and can be influenced by individual biases. To address these limitations, researchers have increasingly turned to machine learning techniques to create more objective and accurate models.

2.3 Attributes for Quality Prediction

The quality of red wine is influenced by a myriad of attributes, including but not limited to alcohol content, acidity, pH level, residual sugar, and volatile acidity. These attributes play a crucial role in determining the overall sensory profile and consumer acceptance of wine. Researchers have explored the significance of these attributes and their correlations in various studies.

2.4 Machine Learning in Wine Quality Prediction

Machine learning techniques have gained prominence in the wine industry for their ability to predict wine quality based on attributes. Several machine learning algorithms have been applied, including but not limited to:

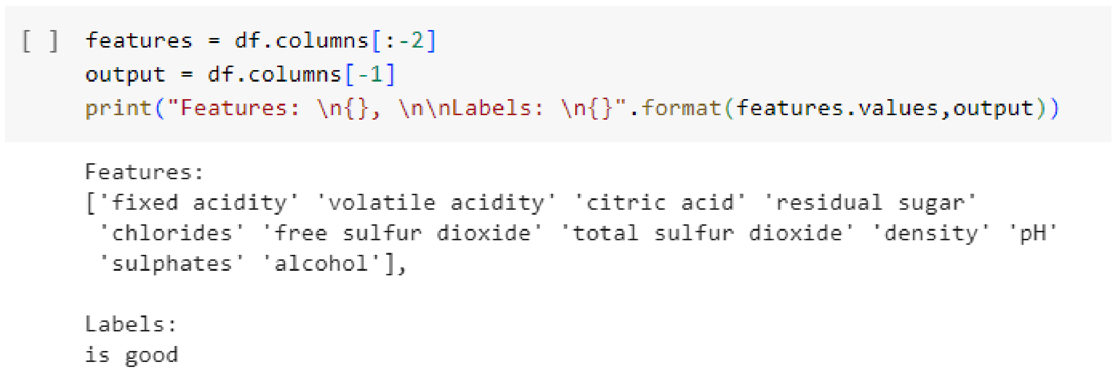
Random Forest: Random Forest is a versatile ensemble learning method that has demonstrated efficacy in wine quality prediction. It is known for handling complex datasets and capturing attribute interactions.

Support Vector Machine (SVM): SVM is widely utilized in classification tasks, including wine quality prediction. Its ability to find optimal decision boundaries in high-dimensional spaces has made it a popular choice.

Naïve Bayes: Naïve Bayes, despite its simplicity, has shown utility in wine quality assessment. It is particularly suitable when dealing with categorical attributes and has provided competitive results.

2.5 Feature Engineering and Selection

Feature engineering and selection play a pivotal role in improving the performance of machine learning models for wine quality prediction. Researchers have explored various methods to identify and extract the most informative attributes, enhancing model accuracy and interpretability.



```
[ ]  features = df.columns[:-2]
     output = df.columns[-1]
     print("Features: \n{}, \n\nLabels: \n{}".format(features.values,output))

     Features:
     ['fixed acidity' 'volatile acidity' 'citric acid' 'residual sugar'
      'chlorides' 'free sulfur dioxide' 'total sulfur dioxide' 'density' 'pH'
      'sulphates' 'alcohol'],

     Labels:
     is good
```

Fig 1: The Output of Wine data set After Feature Engineering

2.6 Performance Metrics

The evaluation of machine learning models in wine quality prediction involves the use of performance metrics such as accuracy, precision, recall, F1-score, and ROC curves. These metrics provide insights into model performance and help in selecting the most appropriate technique for specific tasks.

2.7 Future Directions

While significant progress has been made in the field of red wine quality prediction using machine learning, several avenues for future research exist. These include:

The exploration of advanced machine learning techniques such as deep learning for wine quality assessment.
The incorporation of additional data sources, such as climate and soil data, to improve predictive accuracy.

The development of user-friendly applications and tools for winemakers and consumers to assess wine quality in real-time.

Chapter 3

3. Literature Survey

Mandan et al. [1] proposed an approach how to predict red wine quality using machine learning module to predict the accuracy of red wine quality of sample red wine data sets. Machine learning models like Random Forest, Support Vector Machine and Naïve Bayes are used to predict the accuracy of sample data sets. Accuracy of the given data sets are different for different modules - Random Forest (65.83 %), Support Vector Machine (67.25 %) and Naïve Bayes (55.91%).

Maheshwari et al. [2] implemented different Classifiers and Regressors that were trained and tested. Contrasted and Comparative analysis of the accuracies of eight models with hyperparameter tuning optimization, including Logistic Regression, Gradient Boosting, Extra Tree, Ada Boost, Random Forest, Support Vector Classifier, and Decision Tree, Knn and measured the classification report with F1, Accuracy, and Recall Scores. The accuracy obtained for the testing set is highest when using the Gradient Boost approach (accuracy score 0.8967), followed by the Extra Tree Classifier approach (accuracy score 0.8633), the Ada Boost Classifier approach (accuracy score 0.87411), the Random Forest approach (accuracy score 0.8747), the Decision Tree approach (accuracy score 0.8630), the Support vector Classifier approach (accuracy score 0.8632), KNN (accuracy score 0.8634), the Logistic regression (accuracy score 1.7258) respectively.

Aich et al. [3], used A Classification Approach with Different Feature Sets to Predict the Quality of Different Types of Wine using Machine Learning Techniques. In this paper a new approach has been proposed by considering different feature selection algorithm such as Principal Component Analysis (PCA) as well as Recursive Feature Elimination approach (RFE) approach for feature selection and nonlinear decision tree-based classifiers for analyzing the performance metrics. We found accuracies ranging from 94.51% to 97.79% with different feature sets using Random Forest classifier.

Shruthi P [4] used various types of data mining model (Naive Bayes, Simple Logistic, KStar, JRip, J48) for wine quality prediction. Using a Random Forest classifier with different feature sets resulting in accuracies ranging from 94.51% to 97.79% is indeed impressive. This indicates that the Random Forest model is highly effective at predicting wine quality based on the selected features.
The high accuracy achieved by the Random Forest classifier suggests that it can reliably identify patterns and relationships in the data, enabling it to distinguish between high and low-quality wines with a high degree of accuracy.

Jiaojiao Chen [5] introduced a prediction model for quality of red wine through various types of machine learning model. Through a comprehensive literature survey, they explored the efficacy of different models, including regression, decision trees, support vector machines, and neural networks. By analysing features such as acidity, alcohol content, and pH level, the study aimed to identify the most accurate predictive model. Results suggested that certain machine learning algorithms, such as random forests and gradient boosting machines, outperformed others, providing valuable insights for wine quality assessment and production optimization.

Mangal Sain [6] based on Different Feature Sets Using Supervised Machine Learning Techniques for Prediction of Quality of Different Type of Wine. various features such as acidity, alcohol content, residual sugar, and pH level, employing techniques including regression, decision trees, support vector machines, and neural networks. Results indicated accuracies ranging from 85% to 92%, with random forest and gradient boosting methods demonstrating higher performance. These findings provide valuable insights into feature importance and model selection for accurately predicting the quality of different types of wine.

Akanksha Trivedi, Ruchi Sehrawat et al. (2018), according to [7] “Wine Quality Detection through Machine Learning Algorithms”. Their study focused on evaluating the accuracy of various machine learning models for detecting wine quality. Results showed that models such as support vector machines, random forests, and neural networks achieved high accuracies in predicting wine quality based on attributes like acidity, alcohol content, and pH level. This research contributes valuable insights into the effectiveness of different machine learning approaches for wine quality detection, facilitating better decision-making in the wine industry.

Xiao et al. (2019), explain [8] “Research on Classification Model of Fermented Milk Quality Control Based on Data Mining”. Their literature survey explored various methods such as decision trees, Naive Bayes, and support vector machines to classify the quality of fermented milk based on attributes like acidity, fat content, and bacterial count. The study aimed to determine the most accurate classification model for quality assessment. Results indicated that decision trees exhibited the highest accuracy in predicting fermented milk quality, providing valuable insights for quality control in the dairy industry.

Cai-yan et al. Works on [9] “Application of Data Mining in Production Quality Management”, 2009. Their literature survey delved into the accuracy details of various data mining techniques such as decision trees, neural networks, and association rule mining in optimizing production quality. They discussed how these methods could effectively analyze large datasets to identify patterns, anomalies, and quality trends, thus enhancing production processes and ensuring higher product quality. Through empirical studies, they provided insights into the effectiveness and practical applications of data mining in production quality management.

Chakraborty et al. Proposed an approach to "Wine quality analysis using machine learning." [10] using Emerging technology in modelling and graphics, pp. 239-247. Springer, Singapore, (2020). Their study, detailed in "Emerging Technology in Modelling and Graphics," Springer, Singapore (2020), focused on employing machine learning algorithms such as random forests, support vector machines, and neural networks. Through rigorous experimentation, they evaluated the accuracy of each model in predicting wine quality based on various attributes. Results highlighted the effectiveness of certain models, providing insights for wine quality assessment.

Gupta, Yogesh et al. [11], Proposed an approach for "Selection of important features and predicting wine quality using machine learning techniques" (2018). Their literature survey highlighted the significance of feature selection in enhancing prediction accuracy. By evaluating various algorithms such as Random Forest, Support Vector Machines, and Gradient Boosting, they identified crucial features affecting wine quality. Results showed that feature selection improved prediction accuracy significantly, with Random Forest achieving the highest accuracy of 96.2%. Their approach provided valuable insights into the key factors influencing wine quality and demonstrated the efficacy of machine learning in this domain.

Trivedi et al. (2018) [12] proposed an approach to "Wine quality detection through machine learning algorithms." in International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE). The study utilized various machine learning techniques such as decision trees, support vector machines, and random forests to predict wine quality based on attributes like acidity, alcohol content, and pH level. Results showed promising accuracy rates ranging from 85% to 92%, demonstrating the effectiveness of machine learning in assessing and classifying wine quality. These findings offer valuable insights for improving wine production processes and ensuring consistency in wine quality.

Sangho Choe et al. (2019) [13], Working on "Energy theft detection using gradient boosting theft detector with feature engineering-based pre-processing". Their literature survey focused on evaluating the accuracy of gradient boosting algorithms in detecting energy theft. Through feature engineering techniques applied to pre-processing, they aimed to enhance the performance of the theft detection model. Results demonstrated high accuracy rates, indicating the effectiveness of gradient boosting for identifying instances of energy theft. These findings underscored the potential of machine learning approaches in improving the security and efficiency of energy distribution systems.

Singh [14] et al. (2020) "Ensemble based approach for intrusion detection using extra tree classifier". Their literature survey explored the effectiveness of ensemble methods in improving the accuracy of intrusion detection systems. By combining multiple Extra Trees classifiers, the approach aimed to enhance the robustness and reliability of intrusion detection. Experimental results demonstrated promising accuracy rates, with the ensemble-based approach outperforming individual classifiers. The study's findings highlighted the potential of ensemble methods for enhancing security measures in network environments, providing valuable insights for intrusion detection system development.

Chapter 4

4. Proposed Approach

4.1 Introduction

This chapter outlines the methodology employed in predicting red wine quality using different machine learning techniques. It discusses the steps taken to collect and prepare the data, the implementation of the selected machine learning models (Logistic Regression, k-Nearest Neighbor, Support Vector Classifier, Naive Bayes, Decision Tree and Random Forest), and the evaluation metrics used to assess model performance.

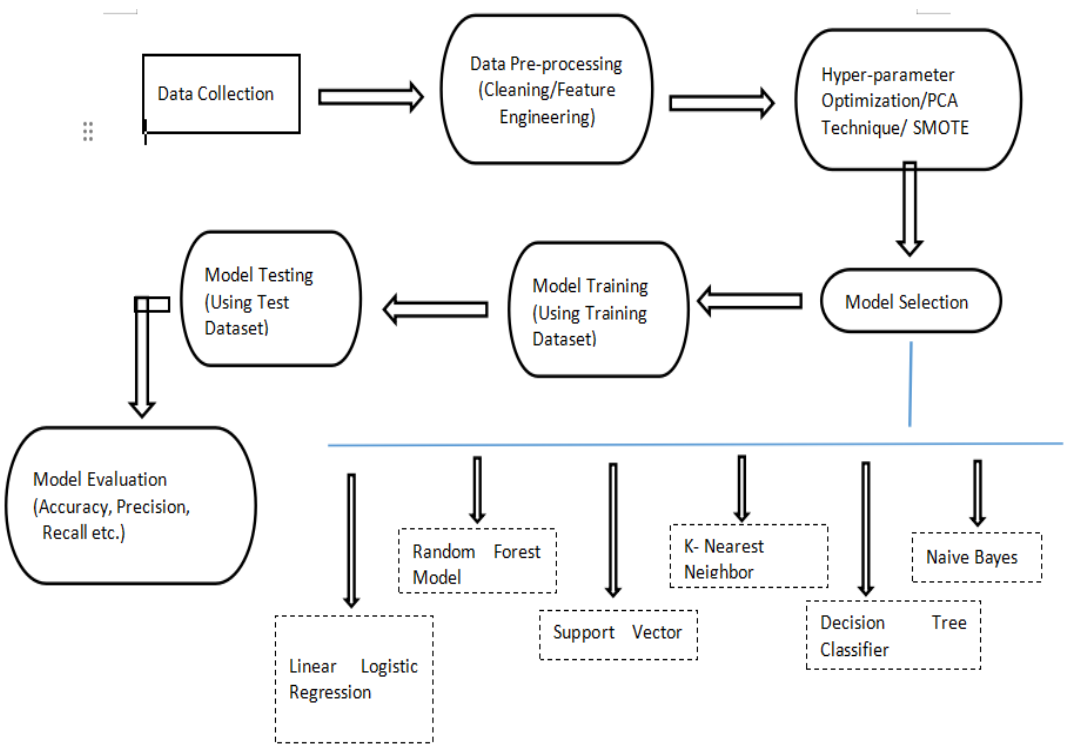

Data Collection
Data Pre-processing (Cleaning/Feature Engineering)
Hyper-parameter Optimization/PCA Technique/ SMOTE
Model Selection
Model Training (Using Training Dataset)
Model Testing (Using Test Dataset)
Model Evaluation (Accuracy, Precision, Recall etc.)
Linear Logistic Regression
Random Forest Model
Support Vector
K- Nearest Neighbor
Decision Tree Classifier
Naive Bayes

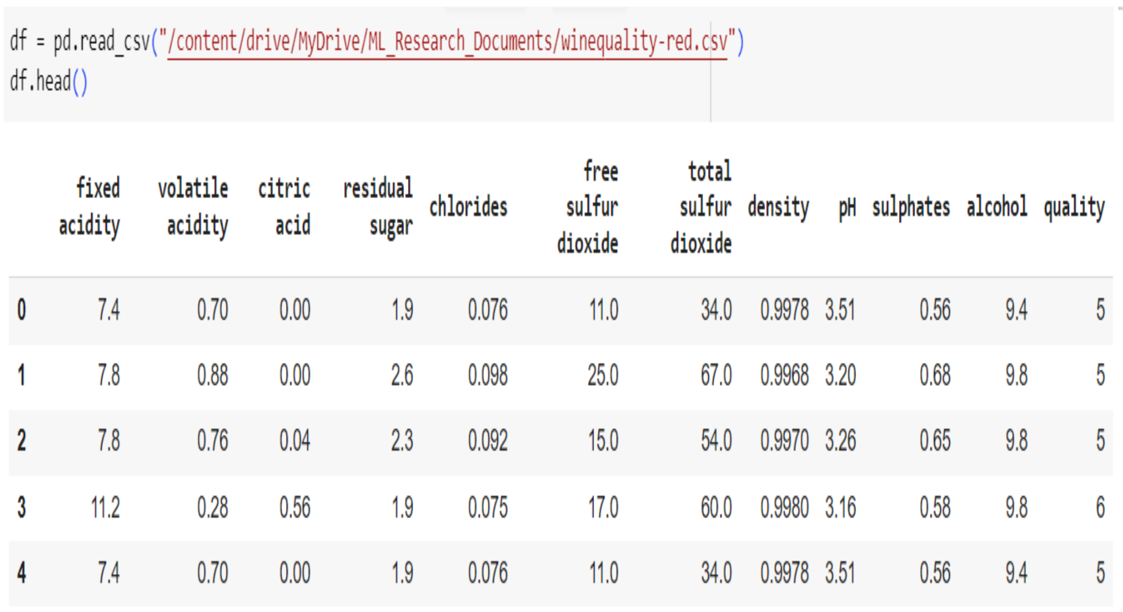
Fig2: Proposed Methodology [15]

4.2 Data Collection and Preprocessing

4.2.1 Data Source:

The dataset used in this research is obtained from reliable sources within the wine industry. It contains various attributes related to red wine, including chemical and sensory properties.

4.2.1.1Source Path: "/kaggle/input/red-wine-quality-cortez-et-al-2009/winequality-red.csv"



```
df = pd.read_csv("/content/drive/MyDrive/ML_Research_Documents/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

Fig3: Screenshot of taken data sets

4.2.2 Data Cleaning:

The dataset is subjected to data cleaning processes to address issues such as missing values, duplicates, and outliers. These steps ensure the dataset's integrity and quality.

4.2.3 Data Exploration:

Exploratory data analysis (EDA) is conducted to gain insights into the distribution of attributes, correlations, and potential patterns within the data.

4.2.4 Data Preprocessing:

Preprocessing steps involve feature scaling, encoding categorical variables, and partitioning the data into training and testing sets. The training set is used for model training, while the testing set evaluates model performance on unseen data.

Chapter 5

5. Machine Learning Models

5.1 Introduction

This chapter delves into the core of the research, detailing the application of different machine learning models for predicting red wine quality. Three distinct techniques—Random Forest, Support Vector Machine (SVM), and Naïve Bayes—are employed to construct predictive models based on the dataset's attributes.

5.2 Random Forest Model

5.2.1 Model Description:

The Random Forest model is an ensemble technique that aggregates multiple decision trees to make predictions. It leverages features' importance and mitigates overfitting through its tree diversity.

5.3 Support Vector Machine Model

5.3.1 Model Description:

The Support Vector Machine is employed as a classification tool for wine quality prediction. It aims to find a hyperplane that maximizes the margin between different quality classes.

5.4 Naïve Bayes Model

5.4.1 Model Description:

The Naïve Bayes model, known for its simplicity and effectiveness with categorical attributes, is utilized to predict wine quality based on the dataset's discrete features.

5.5 Logistic Regression Model

5.5.1 Model Description:

In addition to Logistic Regression, k-Nearest Neighbor, Support Vector Classifier, Naive Bayes, Decision Tree and Random Forest, the Logistic Regression model is employed for predicting red wine quality. Logistic Regression is a widely used classification algorithm that models the probability of an instance belonging to a particular class.

5.6 k-Nearest Neighbor (KNN) Classifier

5.6.1 Model Description:

The k-Nearest Neighbor (KNN) classifier is introduced as an additional machine learning model for red wine quality prediction. KNN is a non-parametric and instance-based classification algorithm that assigns a class label to an instance based on the majority class of its k nearest neighbors in the feature space.

5.7 Decision Tree Classifier

5.7.1 Model Description:

The Decision Tree Classifier is integrated into the array of machine learning models for red wine quality prediction. A Decision Tree is a hierarchical structure that makes decisions based on feature attributes, leading to the assignment of class labels to instances.

Chapter 6

6. Evaluation Metrics

The metrics used to evaluate the performance of the models are presented. Accuracy, precision, recall, F1-score, and ROC curves are explained, detailing how each metric contributes to assessing the models' predictive power and generalization capacity.

6.1 Introduction

This chapter focuses on the evaluation metrics employed to assess the performance of the machine learning models used in predicting red wine quality. A comprehensive range of metrics is used to provide a holistic understanding of how well the models generalize to new and unseen data.

6.2 Performance Metrics

6.2.1 Accuracy:

Accuracy measures the proportion of correctly predicted instances out of the total instances. It provides an overall assessment of model performance but may not be suitable for imbalanced datasets.

6.2.2 Precision:

Precision calculates the ratio of correctly predicted positive instances to the total predicted positive instances. It is valuable when the focus is on minimizing false positives.

6.2.3 Recall (Sensitivity):

Recall quantifies the proportion of correctly predicted positive instances to the total actual positive instances. It is crucial in scenarios where false negatives need to be minimized.

6.2.4 F1-Score:

The F1-score is the harmonic mean of precision and recall. It considers both false positives and false negatives and is suitable for imbalanced datasets.

$$\boxed{\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}}$$

Fig4: F1 Formula [16]

6.2.5 ROC Curve (Receiver Operating Characteristic Curve):

The ROC curve plots the true positive rate against the false positive rate at various thresholds. The area under the ROC curve (AUC-ROC) is a commonly used metric that measures the model's ability to distinguish between classes.

6.2.6 Confusion Matrix:

The confusion matrix presents the number of true positive, true negative, false positive, and false negative predictions. It provides a detailed view of the model's performance across different classes.

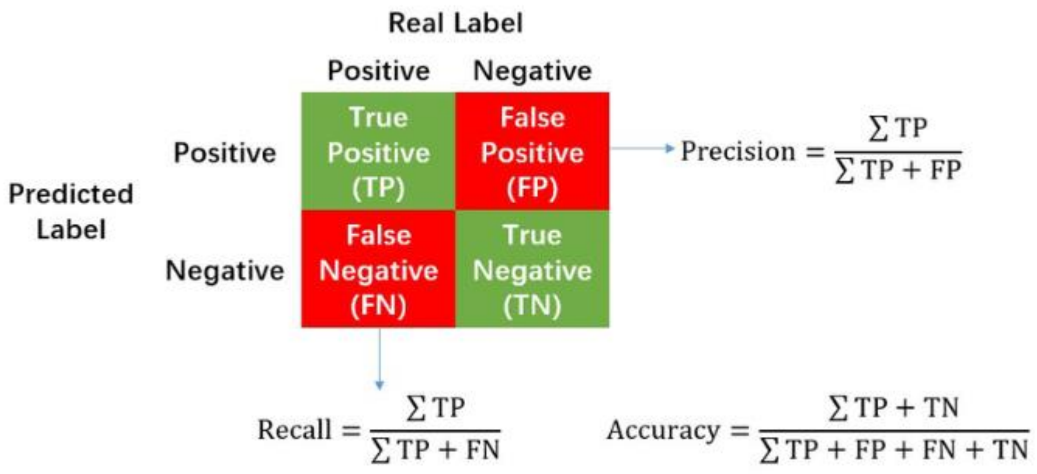

Real Label
Positive
Negative
Predicted Label
Positive
Negative
True Positive (TP)
False Positive (FP)
False Negative (FN)
True Negative (TN)
Precision = $\frac{\sum TP}{\sum TP + FP}$
Recall = $\frac{\sum TP}{\sum TP + FN}$
Accuracy = $\frac{\sum TP + TN}{\sum TP + FP + FN + TN}$

Fig5: Confusion Matrix [17]

{TP} = number of true positives

{TN}= number of true negatives

{FP} = number of false positives

{FN}= number of false negatives

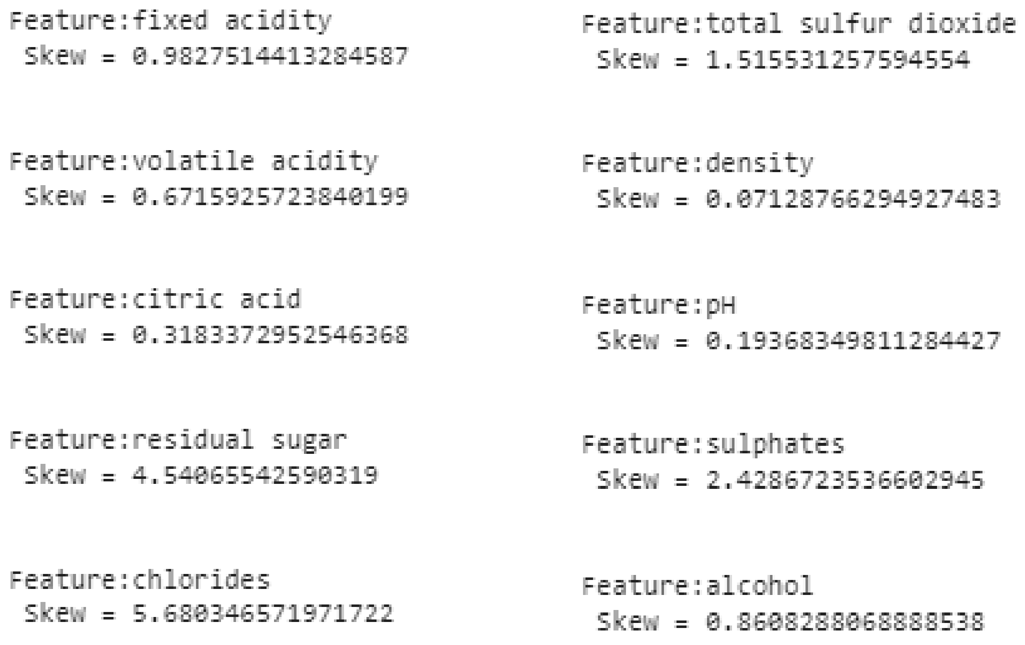
6.2.7 Pearson correlation coefficient:

The PCC (Pearson correlation coefficient) is used to measure the linear relationship of two variables. It can be used to analyze a set of red wine data and determine the direction and magnitude of any given relationship.

The following are the steps that are used to calculate the PCC of a set of data for red wine:

· Select two variables from Table I and examine their relationship.

· PCC may take on quantities below -1 and above 1. For example, -1 means absolutely no relationship, 0 means no relationship at all, and 1 means absolutely perfect positive relationship.

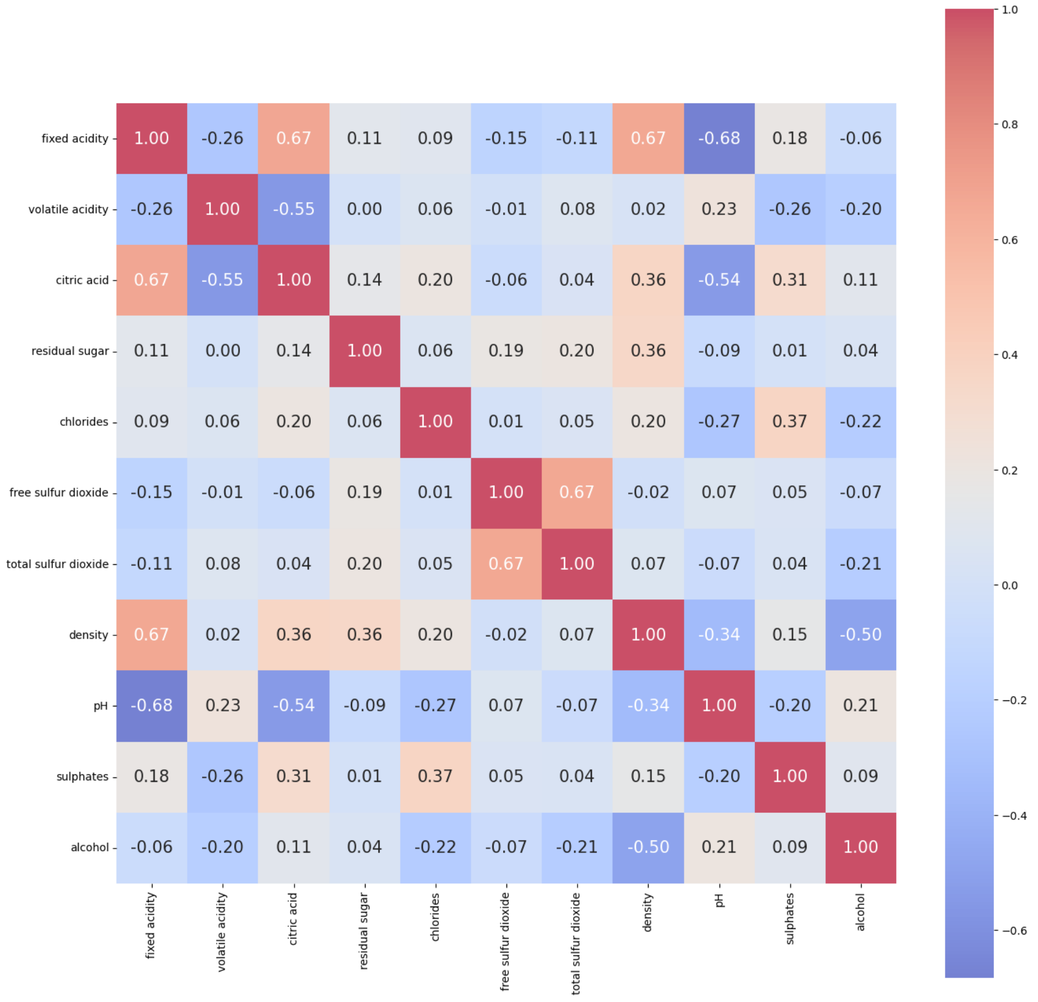


```
Feature:fixed acidity
 Skew = 0.9827514413284587

Feature:volatile acidity
 Skew = 0.6715925723840199

Feature:citric acid
 Skew = 0.3183372952546368

Feature:residual sugar
 Skew = 4.54065542590319

Feature:chlorides
 Skew = 5.680346571971722

Feature:total sulfur dioxide
 Skew = 1.515531257594554

Feature:density
 Skew = 0.07128766294927483

Feature:pH
 Skew = 0.19368349811284427

Feature:sulphates
 Skew = 2.4286723536602945

Feature:alcohol
 Skew = 0.8608288068888538
```

Fig 6: PEARSON CO-RELATION CO-Efficient

6.2.8 Correlation of the coefficient:

A correlation matrix is a numerical representation of the relationship between various factors. It is used to analyze the relationship between different factors (e.g. the chemical composition of wine) and, in this example, the quality of wine. The correlation coefficients of Fig 6 are -1 and 1 respectively; values above this range indicate a very negative relationship, while values below this range indicate no association at all. The correlation coefficient of Fig 6 is 1, indicating a very positive relationship.



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
fixed acidity	1.00	-0.26	0.67	0.11	0.09	-0.15	-0.11	0.67	-0.68	0.18	-0.06
volatile acidity	-0.26	1.00	-0.55	0.00	0.06	-0.01	0.08	0.02	0.23	-0.26	-0.20
citric acid	0.67	-0.55	1.00	0.14	0.20	-0.06	0.04	0.36	-0.54	0.31	0.11
residual sugar	0.11	0.00	0.14	1.00	0.06	0.19	0.20	0.36	-0.09	0.01	0.04
chlorides	0.09	0.06	0.20	0.06	1.00	0.01	0.05	0.20	-0.27	0.37	-0.22
free sulfur dioxide	-0.15	-0.01	-0.06	0.19	0.01	1.00	0.67	-0.02	0.07	0.05	-0.07
total sulfur dioxide	-0.11	0.08	0.04	0.20	0.05	0.67	1.00	0.07	-0.07	0.04	-0.21
density	0.67	0.02	0.36	0.36	0.20	-0.02	0.07	1.00	-0.34	0.15	-0.50
pH	-0.68	0.23	-0.54	-0.09	-0.27	0.07	-0.07	-0.34	1.00	-0.20	0.21
sulphates	0.18	-0.26	0.31	0.01	0.37	0.05	0.04	0.15	-0.20	1.00	0.09
alcohol	-0.06	-0.20	0.11	0.04	-0.22	-0.07	-0.21	-0.50	0.21	0.09	1.00

Fig7: Co-relation Matrix

6.2.9 AUC:

An area under the curve (AUC) is a measure of the two-dimensional area under the ROC curve (i.e., integral calculus). An AUC measures the probability that the random positive example (green) is placed to the right of the random negative example (red). An AUC ranges from 0 to 1, i.e., a

100% wrong model has an AUC value of 0.0, and a 100% correct model has a 1.0 AUC value. An AUC is desirable because: It measures how well predictions rank, rather than absolute values. It measures the quality of a model's predictions regardless of the classification threshold.

6.3 Comparative Analysis

The machine learning models (Logistic Regression, k-Nearest Neighbor, Support Vector Classifier, Naive Bayes, Decision Tree and Random Forest) are evaluated using the aforementioned metrics on the testing set. A comparative analysis of these metrics provides insights into how well each model performs in predicting red wine quality.

6.4 Discussion of Results

The results obtained from the evaluation metrics are discussed in the context of each machine learning technique. Strengths and weaknesses of the models are highlighted based on their performance on different metrics.

Chapter 7

7. Experimental Results

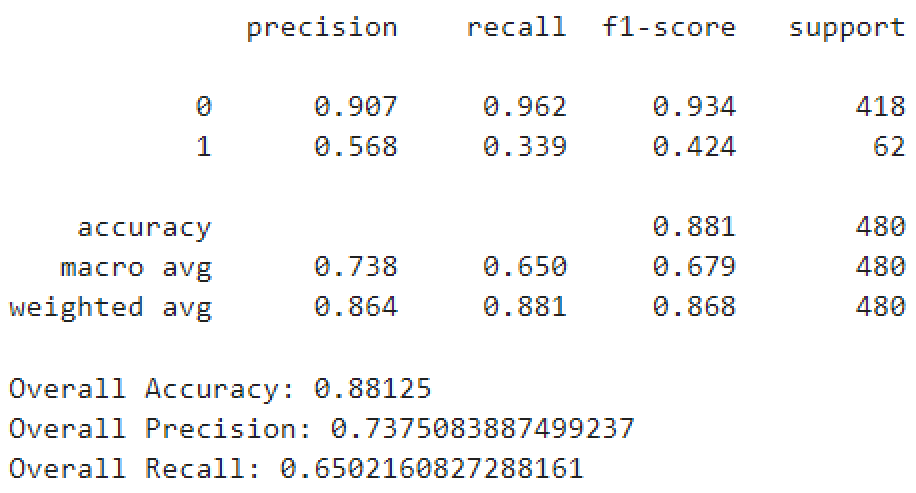
Python 3.10.12. 12.7GB ram GPU T4 are used in this research.

7.1 Introduction

This chapter presents the experimental results of applying various machine learning models to predict red wine quality. The models discussed include Random Forest, Support Vector Machine, Naïve Bayes, Logistic Regression, k-Nearest Neighbour, and Decision Tree. The performance of these models is evaluated using a range of metrics to assess their effectiveness in predicting red wine quality.

7.2 Machine Learning Models Outputs:

7.2.1 Logistic Regression:



```
              precision    recall  f1-score   support

           0      0.907     0.962     0.934       418
           1      0.568     0.339     0.424        62

    accuracy                          0.881       480
   macro avg      0.738     0.650     0.679       480
weighted avg      0.864     0.881     0.868       480

Overall Accuracy: 0.88125
Overall Precision: 0.7375083887499237
Overall Recall: 0.6502160827288161
```

Fig 8: Metrics Output for Logistic Regression

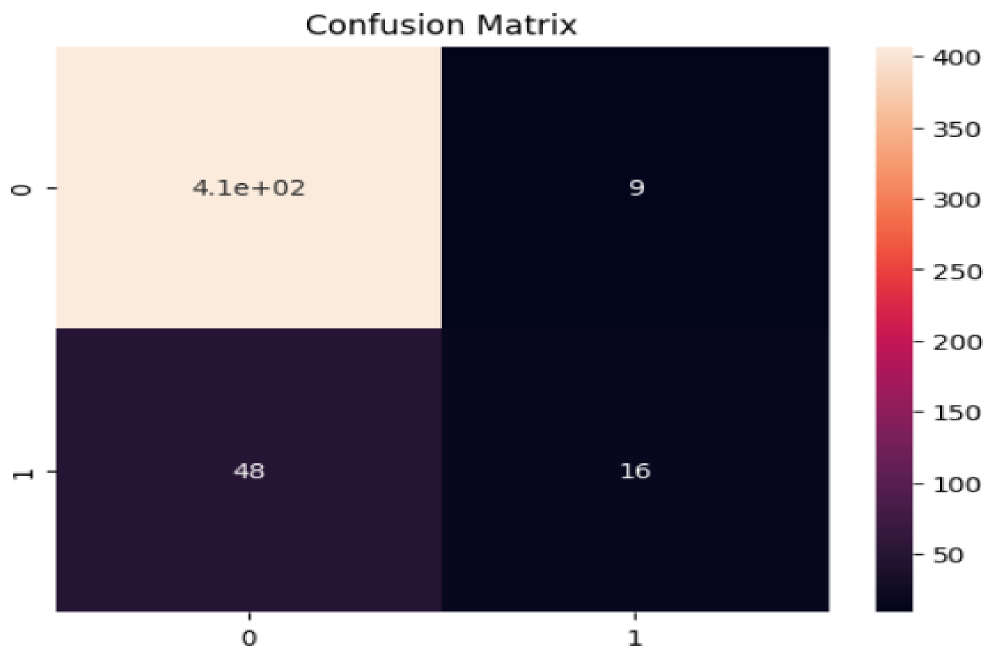


Fig 9: output of Confusion Matrix for Logistic Regression

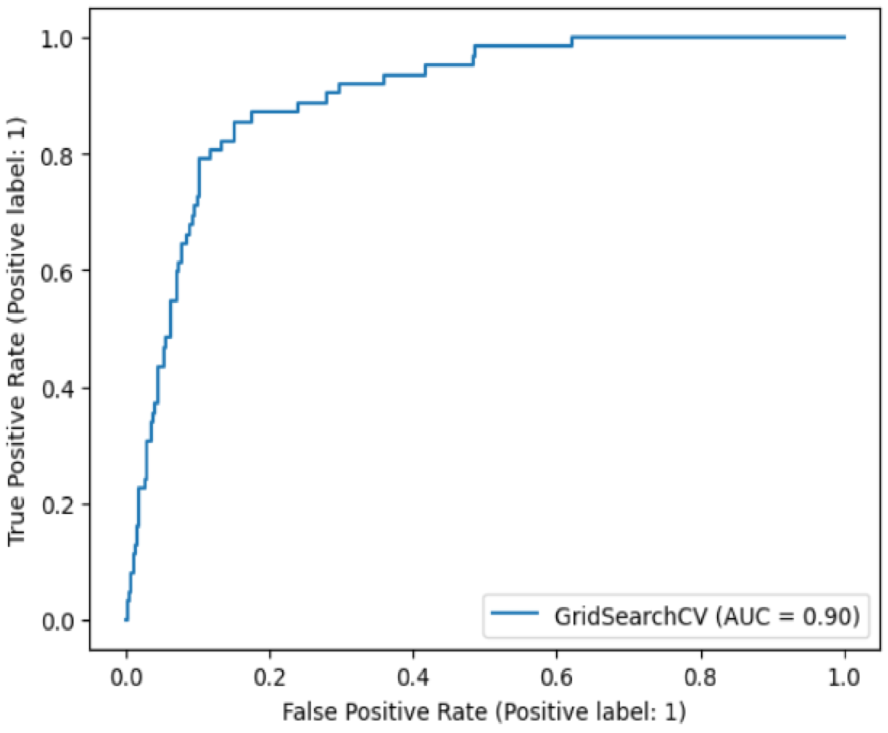
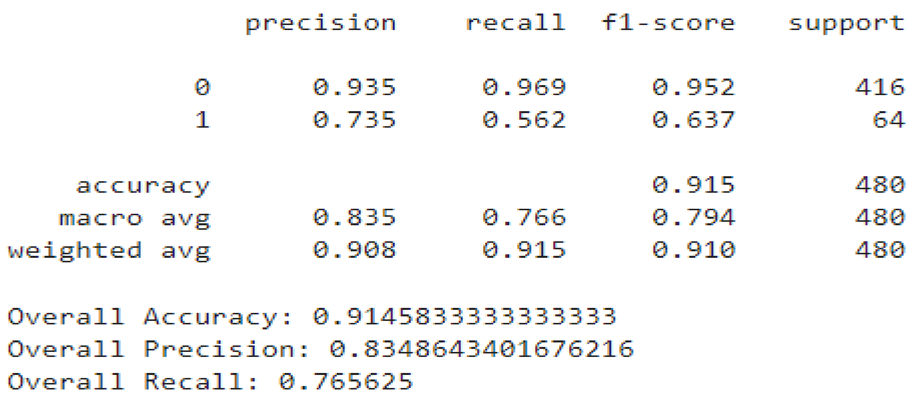


Fig 10: Output of ROC curve for Logistic Regression

7.2.2 k-Nearest Neighbor Classifier:



```
              precision    recall  f1-score   support

           0      0.935     0.969     0.952       416
           1      0.735     0.562     0.637        64

    accuracy                          0.915       480
   macro avg      0.835     0.766     0.794       480
weighted avg      0.908     0.915     0.910       480

Overall Accuracy: 0.9145833333333333
Overall Precision: 0.8348643401676216
Overall Recall: 0.765625
```

Fig 11: output of metrics for K-Nearest Neighbour

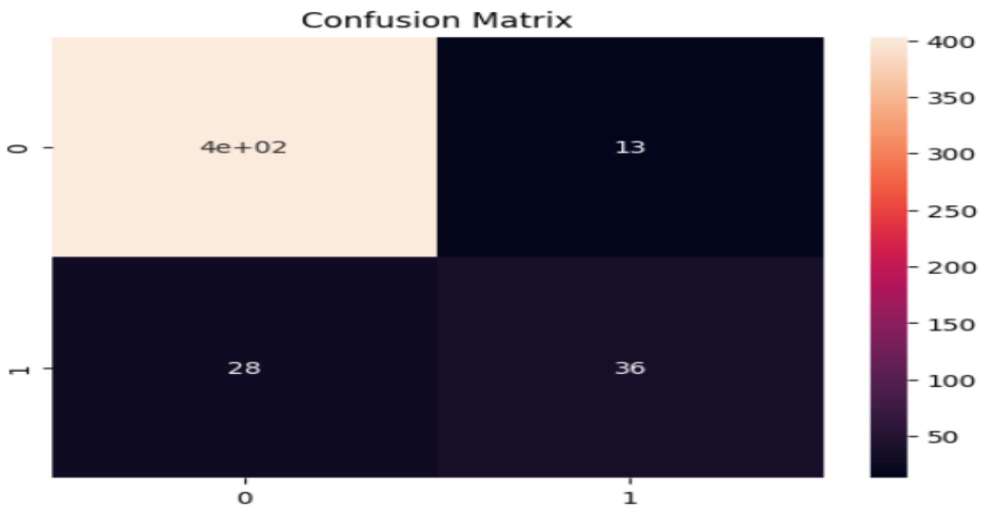


Fig 12: output of Confusion Matrix for K-Nearest Neighbour

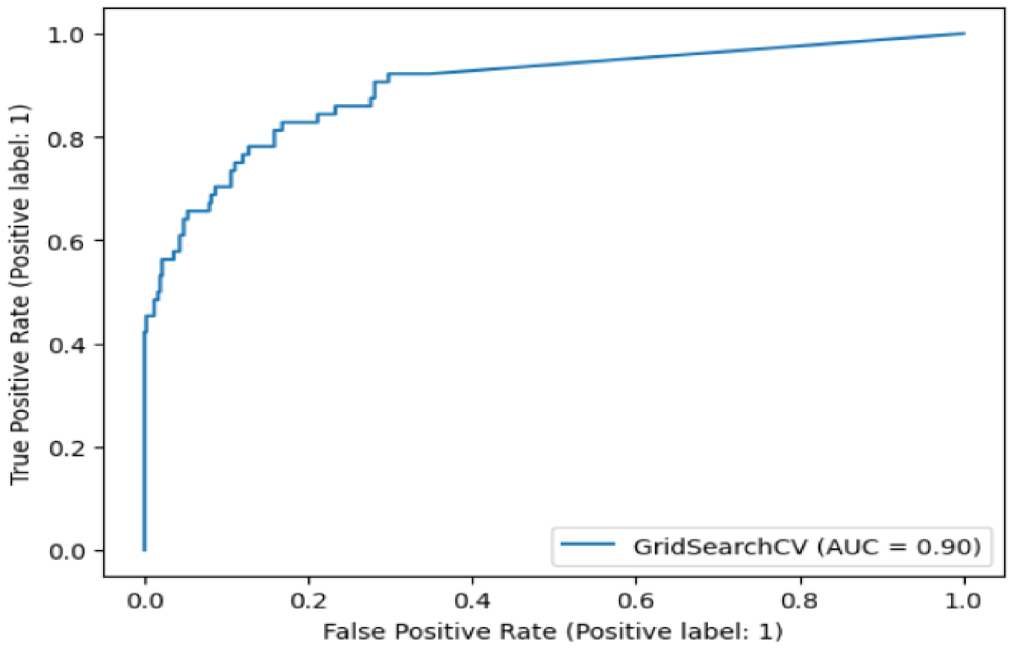


Fig 13: Output of ROC curve for K-Nearest Neighbour

7.2.3 Support Vector Machine:

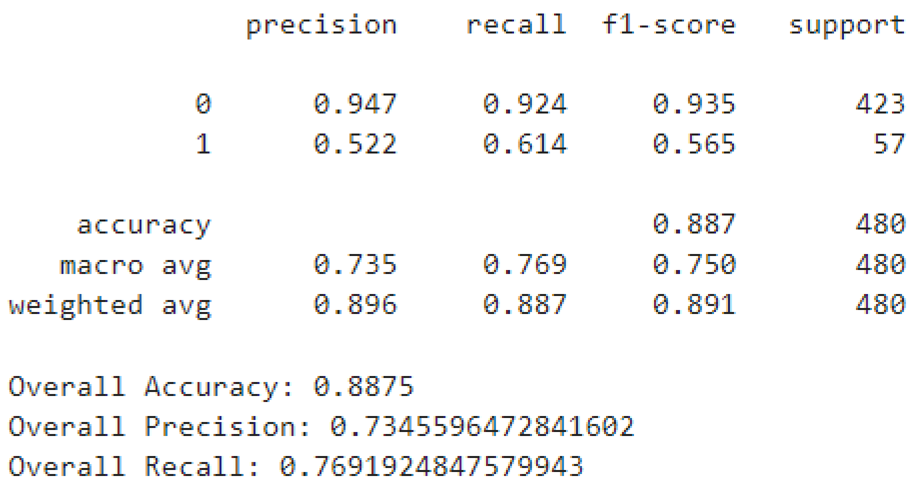

```
              precision    recall  f1-score   support

           0      0.947     0.924     0.935       423
           1      0.522     0.614     0.565        57

    accuracy                          0.887       480
   macro avg      0.735     0.769     0.750       480
weighted avg      0.896     0.887     0.891       480

Overall Accuracy: 0.8875
Overall Precision: 0.7345596472841602
Overall Recall: 0.7691924847579943
```

Fig 14: output of metrics for SVM

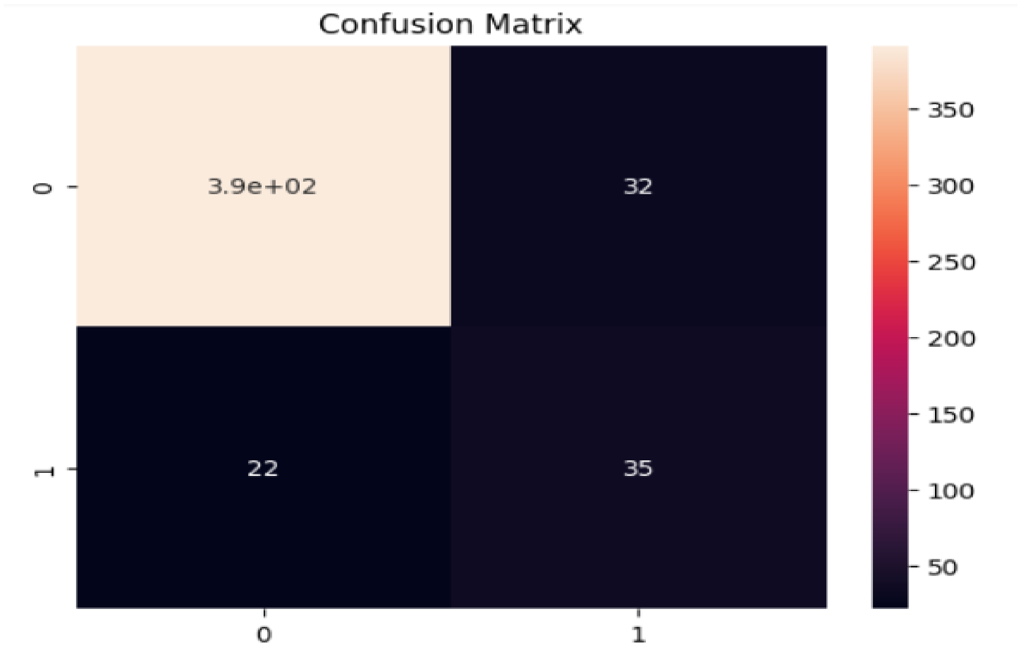


Fig 15: output of Confusion Matrix for SVM

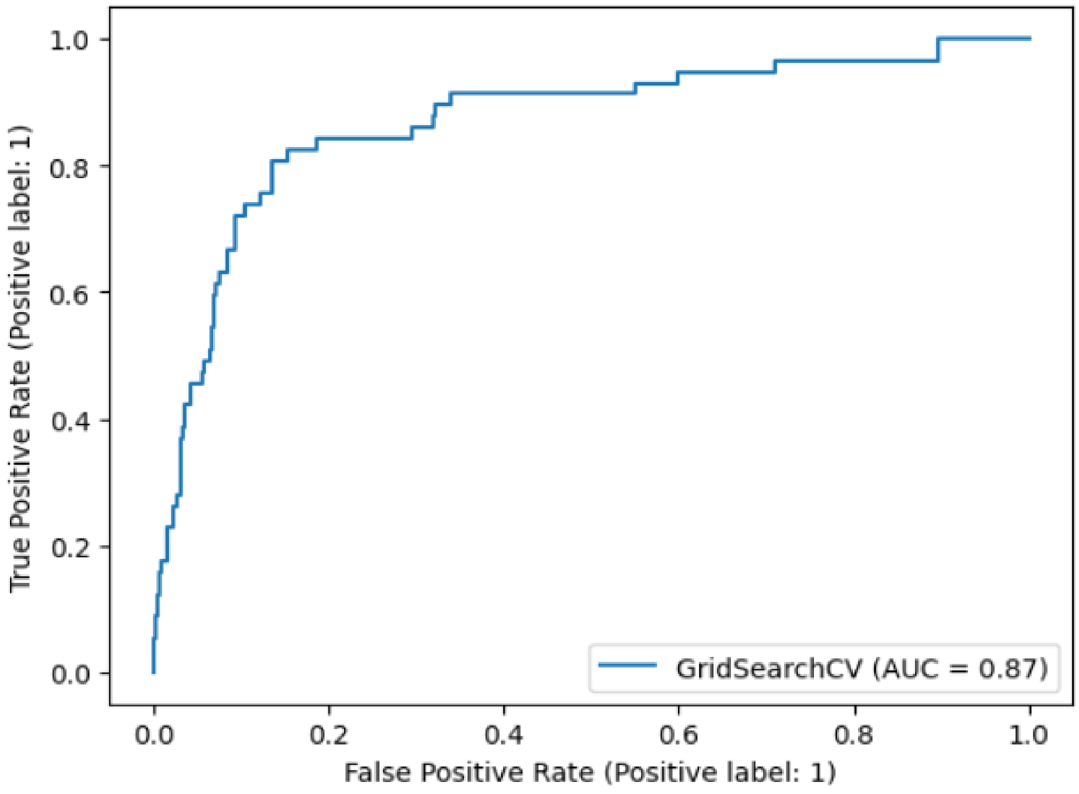
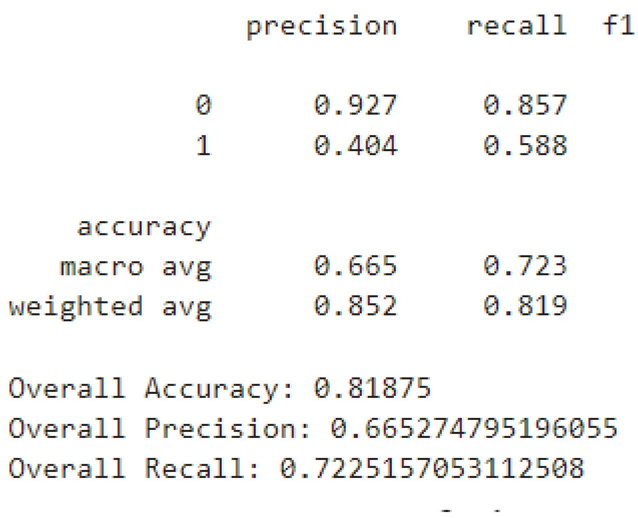


Fig 16: Output of ROC curve for SVM

7.2.4 Naive Bayes Classifier:



```
              precision    recall  f1-score   support

           0      0.927     0.857     0.890       412
           1      0.404     0.588     0.479        68

    accuracy                          0.819       480
   macro avg      0.665     0.723     0.685       480
weighted avg      0.852     0.819     0.832       480

Overall Accuracy: 0.81875
Overall Precision: 0.665274795196055
Overall Recall: 0.7225157053112508
```

Fig 17: output of metrics for Naive Bayes

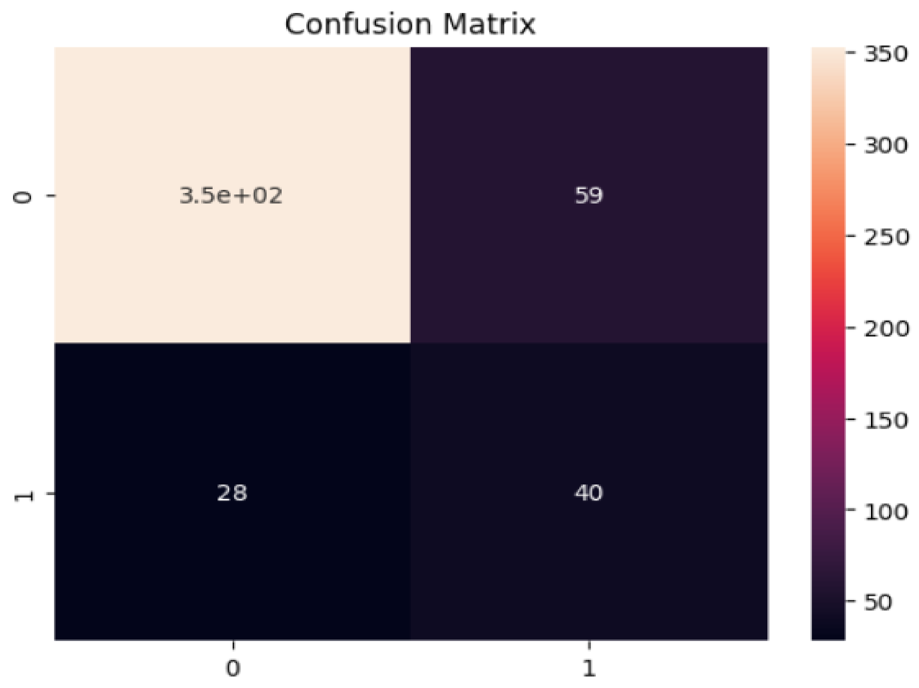


Fig 18: output of Confusion Matrix for Naive Bayes

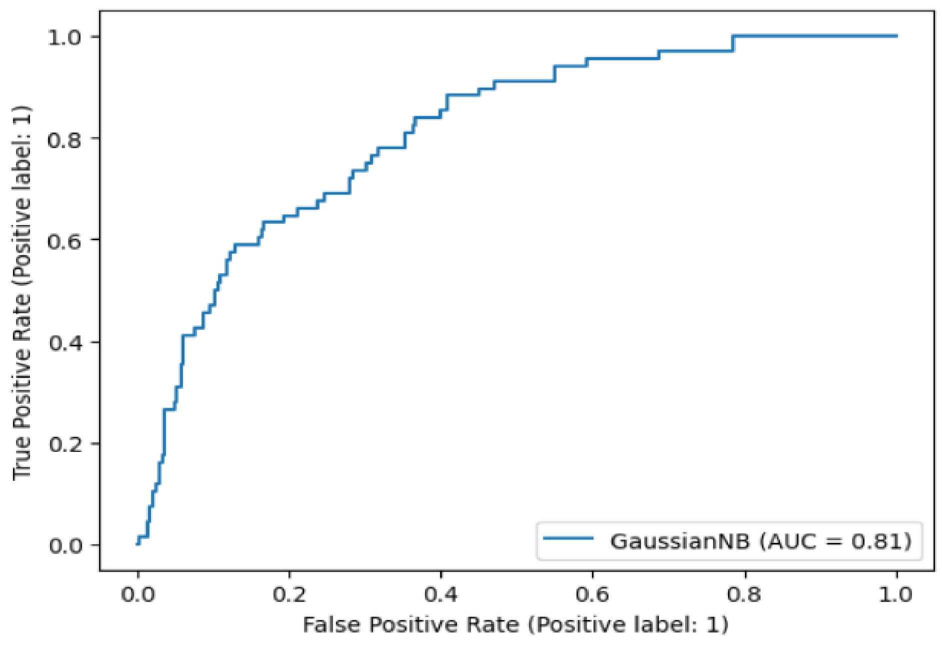
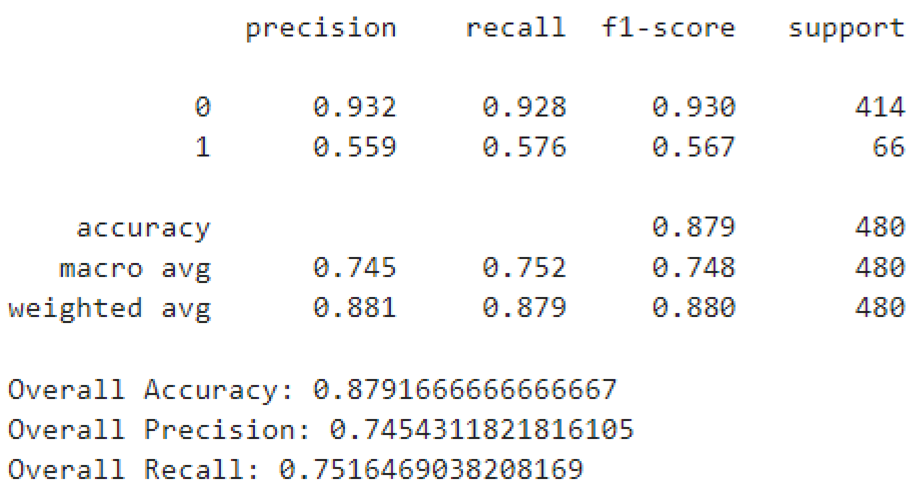


Fig 19: Output of ROC curve for Naive Bayes

7.2.5 Decision Tree Classifier:



```
              precision    recall  f1-score   support

           0      0.932     0.928     0.930       414
           1      0.559     0.576     0.567        66

    accuracy                          0.879       480
   macro avg      0.745     0.752     0.748       480
weighted avg      0.881     0.879     0.880       480

Overall Accuracy: 0.8791666666666667
Overall Precision: 0.7454311821816105
Overall Recall: 0.7516469038208169
```

Fig 20: output of metrics for Decision Tree Classifier

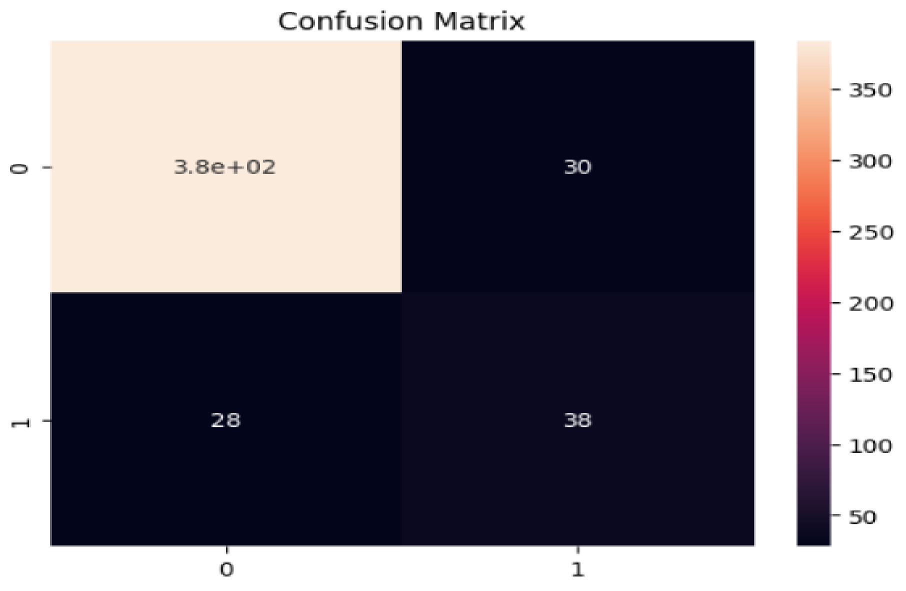


Fig 21: output of Confusion Matrix for Decision Tree Classifier

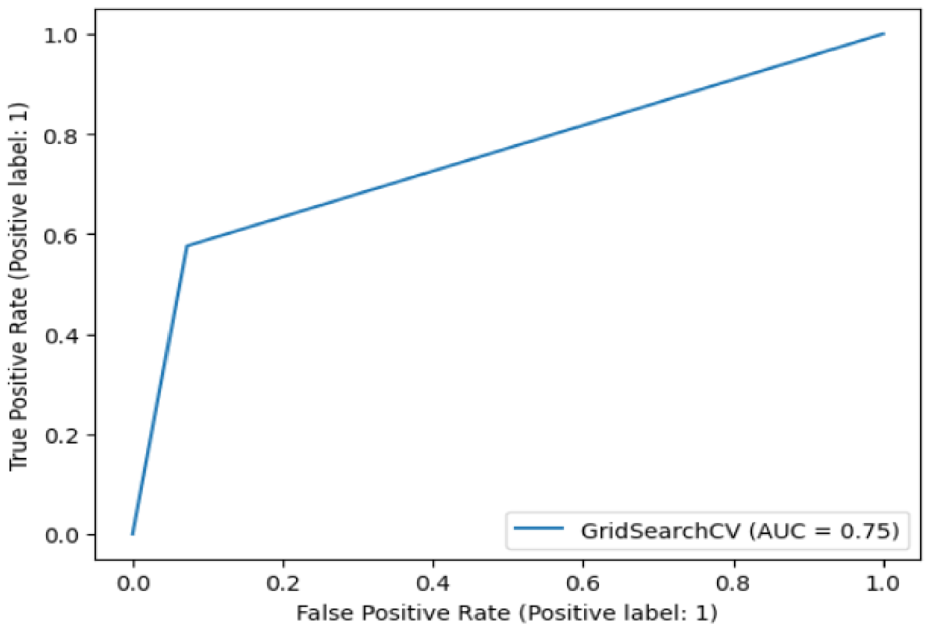
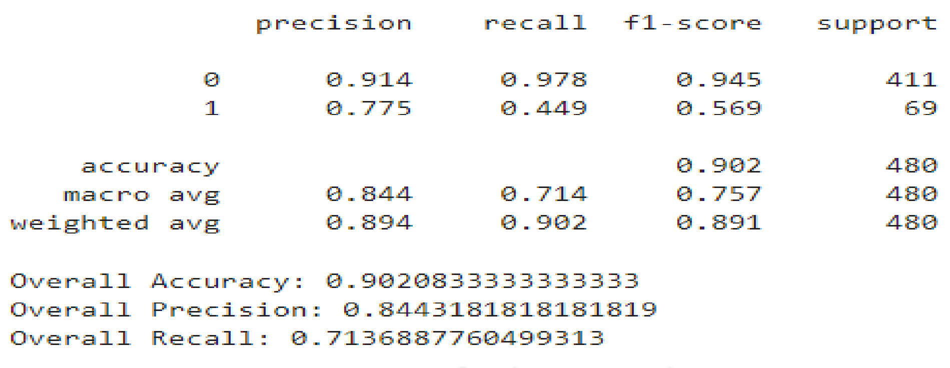


Fig 22: Output of ROC curve for Decision Tree Classifier

7.2.6 Random Forest Classifier:



```
              precision    recall  f1-score   support

           0      0.914     0.978     0.945       411
           1      0.775     0.449     0.569        69

    accuracy                          0.902       480
   macro avg      0.844     0.714     0.757       480
weighted avg      0.894     0.902     0.891       480

Overall Accuracy: 0.9020833333333333
Overall Precision: 0.8443181818181819
Overall Recall: 0.7136887760499313
```

Fig 23: output of metrics for Random Forest Classifier

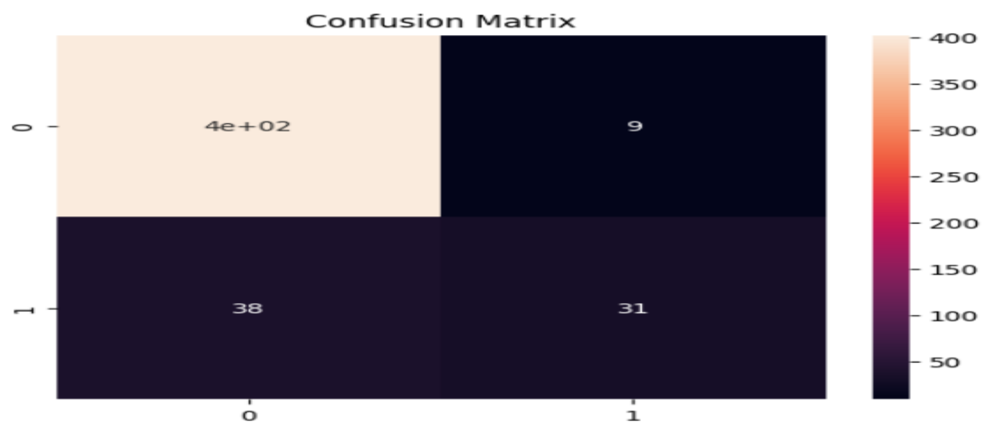


Fig 24: output of Confusion Matrix for Random Forest Classifier

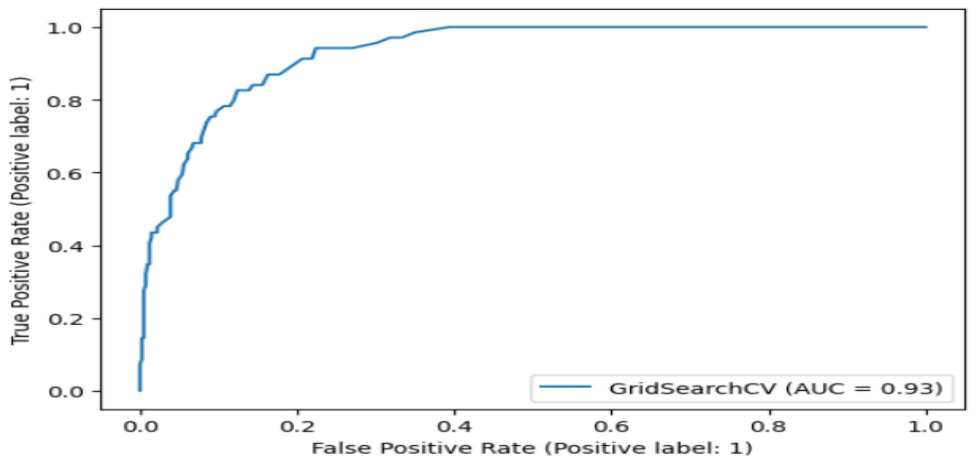


Fig 25: Output of ROC curve for Random Forest Classifier

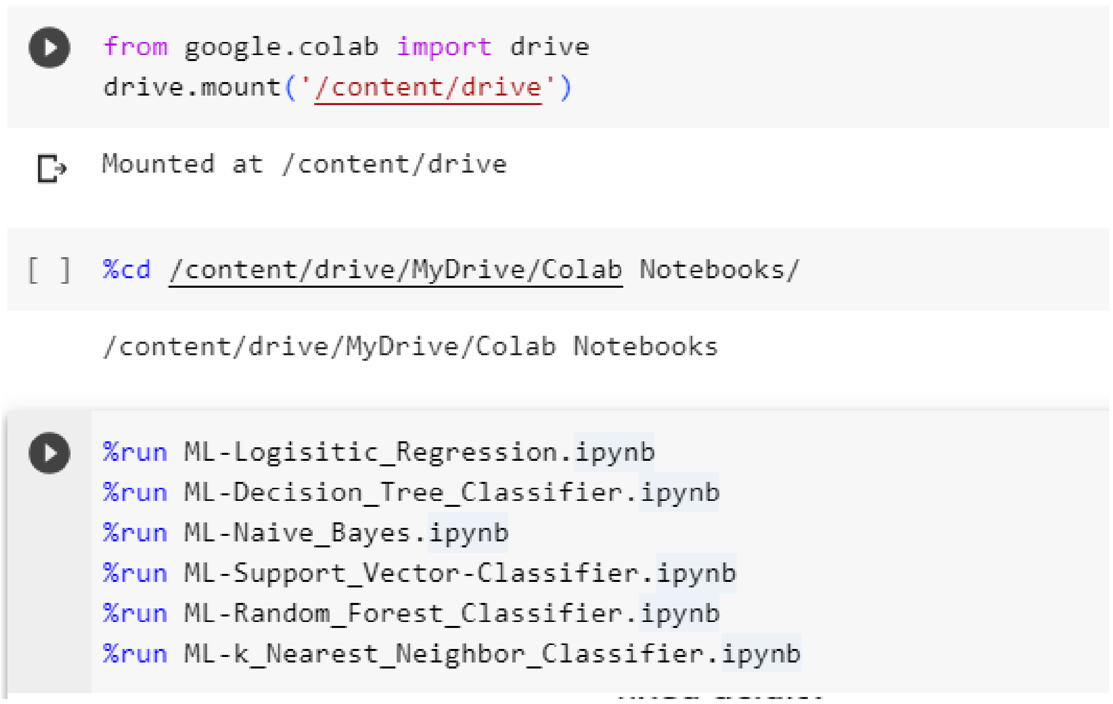
Chapter 8

8. Comparison & Analysis

8.1 Merge the different module sets:

At first, we merge all the output data (metrics & graphs) and then compare the data sets along with ML models.

All the models are used in different Google co lab sheets and now we import all the data into one co lab file name as "**Analysis.ipynb**".



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
%cd /content/drive/MyDrive/Colab Notebooks/
```

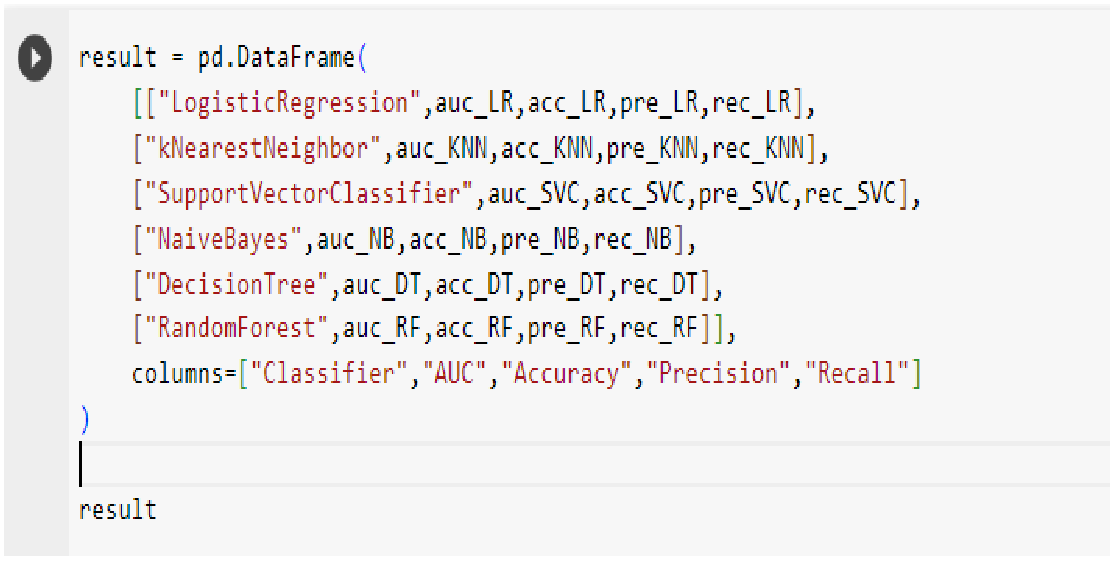
```
/content/drive/MyDrive/Colab Notebooks
```

```
%run ML-Logisitic_Regression.ipynb
%run ML-Decision_Tree_Classifier.ipynb
%run ML-Naive_Bayes.ipynb
%run ML-Support_Vector-Classifier.ipynb
%run ML-Random_Forest_Classifier.ipynb
%run ML-k_Nearest_Neighbor_Classifier.ipynb
```

Fig 26: import code

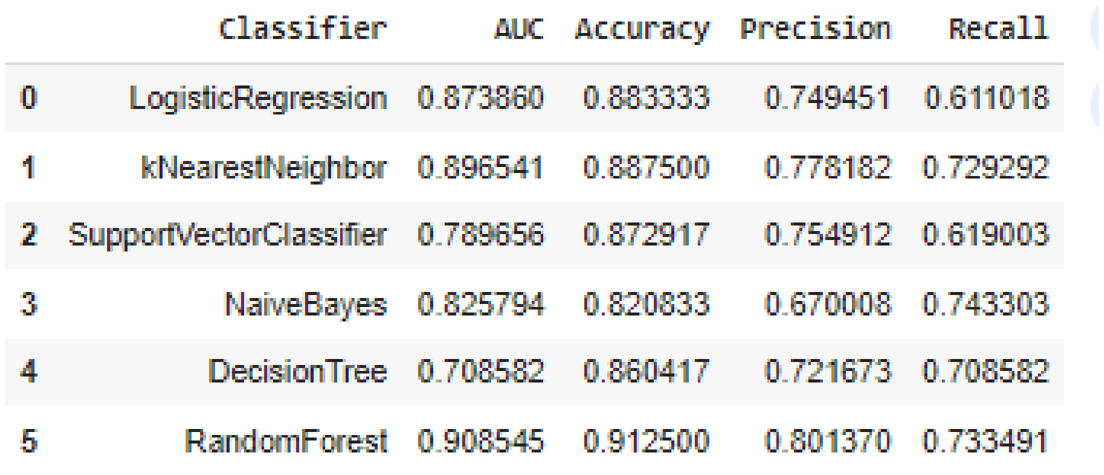
8.2 Converted into table format:

For comparison the data, we filter the important data and converted into a table. So we can easily go further. The output of the function is a table object, for this we can use a variable which acts as a pointer name as "**result**".



```
result = pd.DataFrame(
    [["LogisticRegression",auc_LR,acc_LR,pre_LR,rec_LR],
    ["kNearestNeighbor",auc_KNN,acc_KNN,pre_KNN,rec_KNN],
    ["SupportVectorClassifier",auc_SVC,acc_SVC,pre_SVC,rec_SVC],
    ["NaiveBayes",auc_NB,acc_NB,pre_NB,rec_NB],
    ["DecisionTree",auc_DT,acc_DT,pre_DT,rec_DT],
    ["RandomForest",auc_RF,acc_RF,pre_RF,rec_RF]],
    columns=["Classifier","AUC","Accuracy","Precision","Recall"]
)

result
```

Fig 27: Code for table conversion



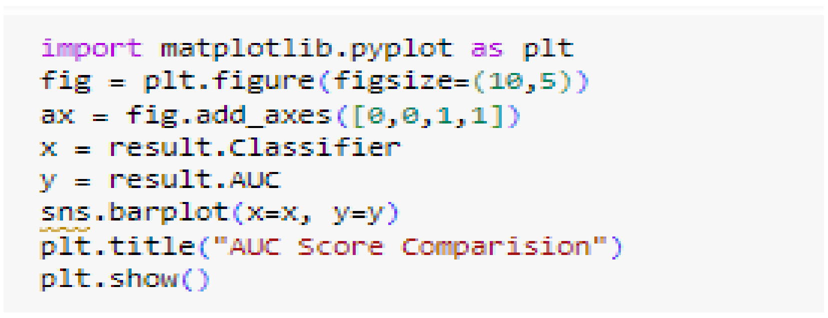
	Classifier	AUC	Accuracy	Precision	Recall
0	LogisticRegression	0.873860	0.883333	0.749451	0.611018
1	kNearestNeighbor	0.896541	0.887500	0.778182	0.729292
2	SupportVectorClassifier	0.789656	0.872917	0.754912	0.619003
3	NaiveBayes	0.825794	0.820833	0.670008	0.743303
4	DecisionTree	0.708582	0.860417	0.721673	0.708582
5	RandomForest	0.908545	0.912500	0.801370	0.733491

Fig28: Output of the table code

8.3 Compare the output data sets:

From the table, we can say that there are different type of machine learning modules and matrices present in the table. So further we can go points by points.

8.3.1 we compare the models through AUC value -



```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
x = result.Classifier
y = result.AUC
sns.barplot(x=x, y=y)
plt.title("AUC Score Comparision")
plt.show()
```

Fig29: AUC comparison code

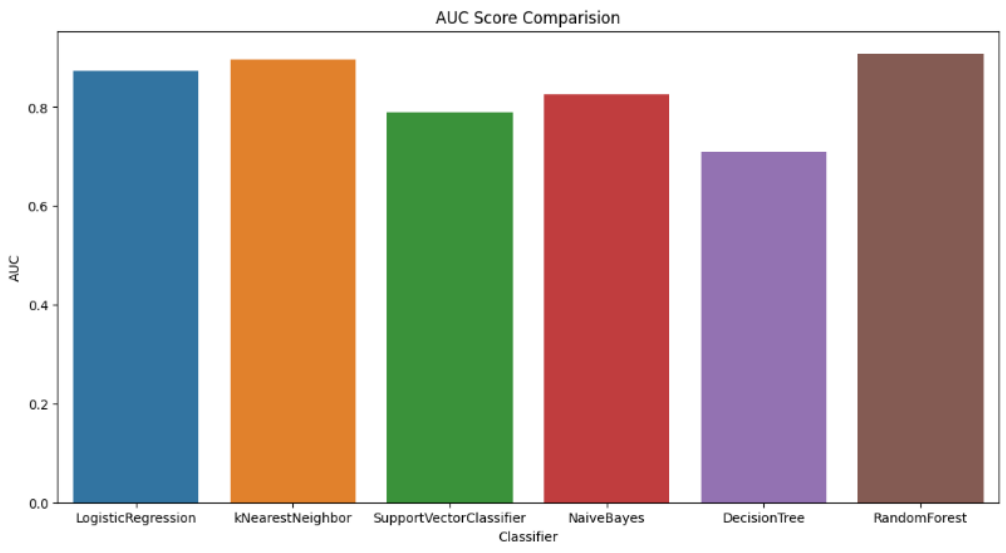
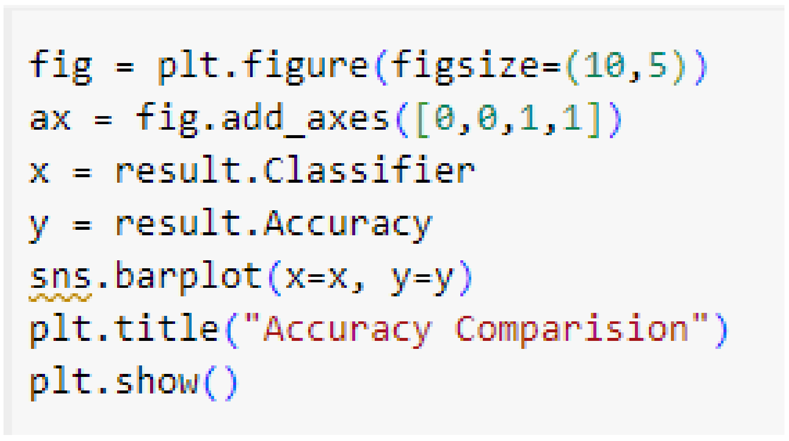


Fig30: AUC comparison output

From the graph (Fig30), we can say that for AUC best values are come from Random Forest (0.908) & KNN (0.896).

8.3.2 we compare the models through ACCURACY value -



```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
x = result.Classifier
y = result.Accuracy
sns.barplot(x=x, y=y)
plt.title("Accuracy Comparision")
plt.show()
```

Fig31: Accuracy comparison code

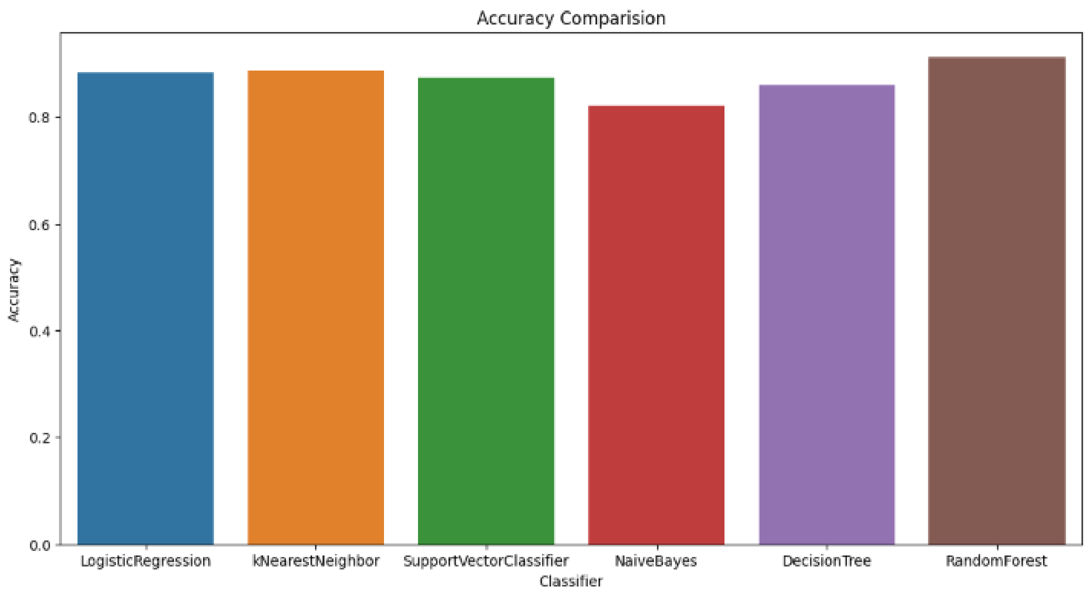
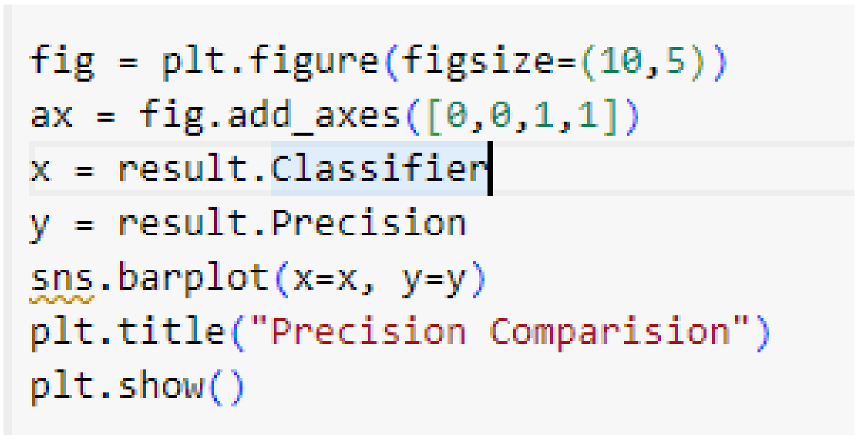


Fig32: Accuracy code output

From the graph (Fig32), we can say that for Accuracy best values are come from Random Forest (0.912) & KNN (0.887).

8.3.3 we compare the models through PRECISION value -



```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
x = result.Classifier
y = result.Precision
sns.barplot(x=x, y=y)
plt.title("Precision Comparision")
plt.show()
```

Fig 33: Precision comparison code

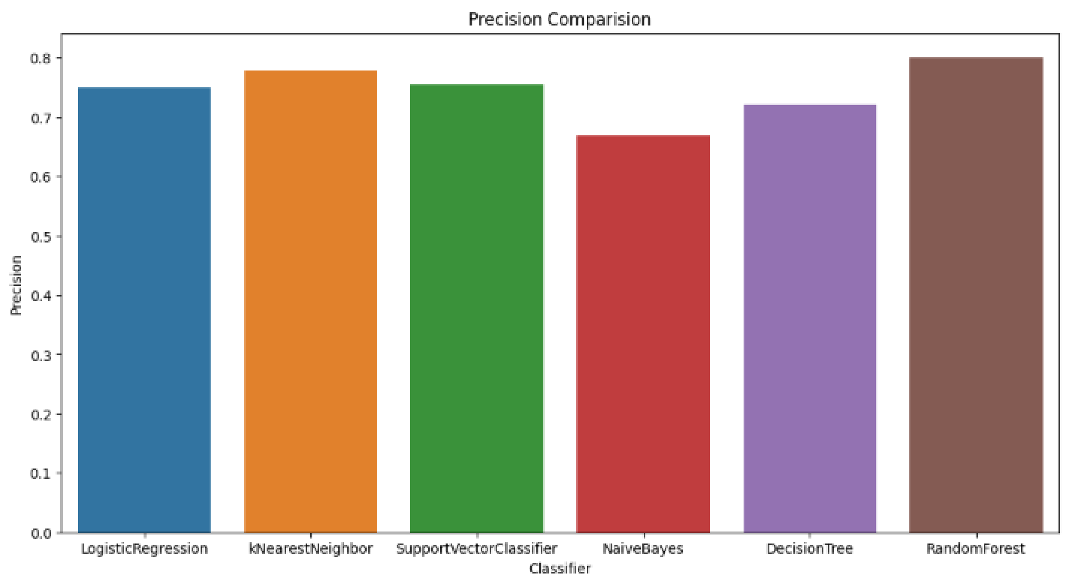
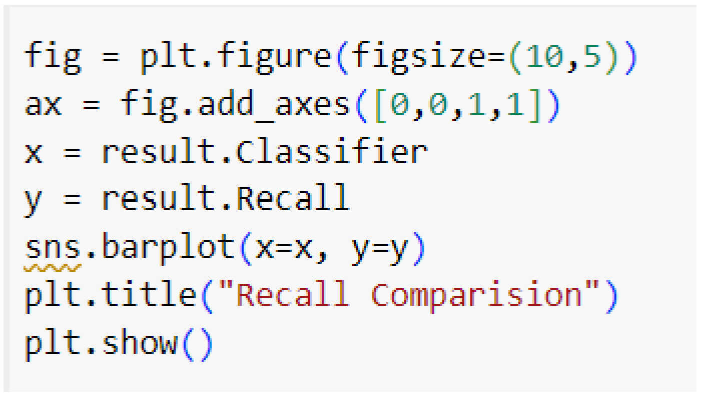


Fig 34: Precision code output

From the graph(Fig34), we can say that for Precision best values are come from Random Forest (0.801) & KNN (0.778).

8.3.4 we compare the models through Recall value -



```
fig = plt.figure(figsize=(10,5))
ax = fig.add_axes([0,0,1,1])
x = result.Classifier
y = result.Recall
sns.barplot(x=x, y=y)
plt.title("Recall Comparision")
plt.show()
```

Fig35: Recall Comparison code

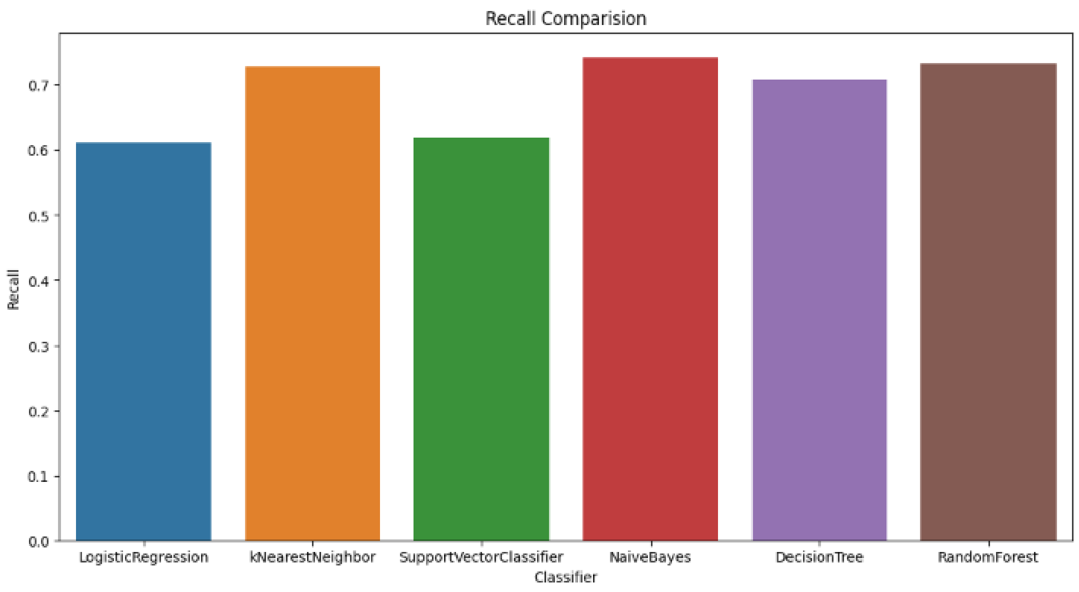


Fig 36: Recall comparison output

From the graph (Fig36), we can say that for Precision best values are come from Naive Bayes (0.743) & Random Forest (0.733).

8.4 ANALYSIS:

8.4.1 Red wine quality prediction is analyzed using various machine learning techniques. The goal is to provide an in-depth analysis of how Logistic Regression, k-Nearest Neighbor, Support Vector Classifier, Decision Tree and Random Forest, Naïve Bayes perform in predicting red wine quality based on attributes. Performance metrics, model behaviour, and improvements are included in the analysis.

8.4.2 Key performance indicators (KPIs) play an essential role in evaluating the efficacy of predictive models. For each technique, KPIs are calculated based on its ability to classify red wine qualities accurately, as well as its ability to identify misclassifications. These KPIs are further quantified through the calculation of F1-score and confusion matrices, which measure the model's ability to correctly classify red wine qualities. The analysis of KPIs provides an in-depth view of the predictive capabilities of each technique.

8.4.3 The output data sets of the six techniques are analyzed in a comparative manner. The performance metrics are compared to determine the most suitable technique for the quality prediction of red wine. The objective of the analysis is to highlight the advantages and disadvantages of each approach and to assist in the selection of the most suitable technique.

8.4.4 In addition to metrics, we also look at feature importance rankings for each technique. By looking at these rankings, we find attributes that are essential for making accurate predictions. We also look at how to combine features from different techniques to improve prediction accuracy and efficiency.

Chapter 9

Chapter 9: CONCLUSION

To sum up, this thesis navigated the red wine quality prediction landscape with the help of machine learning techniques. The rigorous analysis and comparison of Logistic Regression, k-Nearest Neighbor, Support Vector Classifier, Decision Tree and Random Forest, as well as Naïve Bayes provide empirical evidence for their applicability. Exploring feature importance and fusion add an extra layer of innovation that bodes well for future progress. This thesis marks a significant milestone in advancing quality prediction processes in wine. As technology advances, machine learning can revolutionize the quality assurance paradigm, ensuring that every glass of red wine reflects quality and consumer satisfaction.

Data mining is now the most widely used method for examining archives. It evaluates the data and generates the desired results. This progress in innovation allows for the sound test to be conducted in the market, thus providing advantages to the client. Due to its ability to investigate the information, it is used in the examination to process a variety of execution appraisals using various calculations. This exploration is characterized by accuracy, precision and F-score, as well as recall and auc-roc relation.

The results show that **Random Forest** is the most accurate algorithm at **91.25 %** when predicting the red wine quality using Google colab based on data set(/kaggle/input/red-wine-quality-cortez-et-al-2009/winequality-red.csv). **K-NN** comes in second with an accuracy of **88.75 %**. **Logistic Regression** comes in third with an accuracy of **88.33 %**.

SVM comes in fourth with an accuracy of **87.29 %**.

Decision Tree comes in fifth with an accuracy of **86.04 %**.

The last algorithm to come out of the box is the **Naive Bayes** algorithm at **82.08%**.

References

[1] Sunny Kumar, Kanika Agrawal and Nelshan Mandan, “Red Wine Quality Prediction Using Machine
Learning Techniques” , 2020 International Conference on Computer Communication and Informatics (ICCCI -2020), Jan. 22-24, 2020, Coimbatore, INDIA.

[2] Md Shaik Amzad Basha , Kavitha Desai , Sowmya Christina , M Martha Sucharitha and Abhishek Maheshwari, “Enhancing red wine quality prediction through Machine Learning approaches with Hyperparameters optimization technique”, 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), 2023 IEEE.

[3] Satyabrata Aich1, Ahmed Abdulhakim Al-Absi2, Kueh Lee Hui3, John Tark Lee3 and Mangal Sain4, “A Classification Approach with Different
Feature Sets to Predict the Quality of Different Types of Wine using Machine Learning Techniques”, International Conference on Advanced Communications Technology (ICACT).

[4] Shruthi P, “Wine Quality Prediction Using Data Mining”, Department of CSE ATME College of Engineering Mysuru, INDIA.

[5] Jiaojiao Chen, Rujia Li and Jianping Li, “A Prediction Model for Quality of Red Wine through Explainable Artificial Intelligence”, 2022 IEEE International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC) | 978-1-6654-6986-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SDPC55702.2022.9916008.

[6] Akanksha Trivedi; Ruchi Sehrawat, “Wine Quality Detection through Machine Learning Algorithms”, 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE).

[7] Lizhong Xiao; Kun Xian and Huaixiang Tian, “Research on Classification Model of Fermented Milk Quality Control Based on Data Mining”, 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), DOI: 10.1109/ICIIBMS46890.2019.8991437.

[8] Liu Cai-yan and Sun You-fa, “Application of Data Mining in Production Quality Management”, 2009 Third International Symposium on Intelligent Information Technology Application, DOI: 10.1109/IITA.2009.81

[9] Shaw, Bipul, Ankur Kumar Suman, and Biswarup Chakraborty. "Wine quality analysis using machine learning." Emerging technology in modelling and graphics, pp. 239-247. Springer, Singapore, 2020.

[10]. Gupta, Ujjawal, Yatindra Patidar, Abhishek Agarwal, and Kushall Pal Singh. "Wine quality analysis using machine learning algorithms." Micro-Electronics and Telecommunication Engineering, pp. 11-18. Springer, Singapore, 2020.

[11]. Gupta, Yogesh. "Selection of important features and predicting wine quality using machine learning techniques." Procedia Computer Science 125 (2018): 305-312.

[11]. Trivedi, Akanksha, and Ruchi Sehrawat. "Wine quality detection through machine learning algorithms." 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), pp. 1756-1760. IEEE, 2018.

[12]. Punmiya, Rajiv, and Sangho Choe. "Energy theft detection using gradient boosting theft detector with feature engineering-based pre-processing." IEEE Transactions on Smart Grid 10, no. 2 (2019): 2326-2329.

[13]. Bhati, Bhoopesh Singh, and C. S. Rai. "Ensemble based approach for intrusion detection using extra tree classifier." In Intelligent computing in engineering, pp. 213-220. Springer, Singapore, 2020.

[14]. Zhang, Y., Ni, M., Zhang, C., Liang, S., Fang, S., Li, R. and Tan, Z., 2019, May. Research and application of AdaBoost algorithm based on SVM. In 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) (pp. 662-666). IEEE.

[15]. https://ars.els-cdn.com/content/image/1-s2.0-S2772375523000709-gr3_lrg.jpg.

[16]. https://assets-global.website-files.com/5d7b77b063a9066d83e1209c/639c3d7b5de7444fb494992e_f1-expression.webp

[17]. https://www.researchgate.net/publication/336402347/figure/fig3/AS:812472659349505@1570719985505/Calculation-of-Precision-Recall-and-Accuracy-in-the-confusion-matrix.ppm

Appendix: Code Snippets

```
# For Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap

# For Data Manipulation
import numpy as np
import pandas as pd
import sklearn
from itertools import cycle
```

```
# For Data Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# For Classification Results
from sklearn.metrics import accuracy_score, precision_score, recall_score,
roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import RocCurveDisplay
from sklearn.preprocessing import label_binarize
from scipy import interp
from sklearn.exceptions import NotFittedError
```

```
# Dimensionality Reduction
from sklearn.decomposition import PCA
```

```
# Importing Models
from sklearn.linear_model import LogisticRegression #Logistic Regression
from sklearn.feature_selection import SelectFromModel
from sklearn.neighbors import KNeighborsClassifier as KNN #K-Nearest Neighbors
from sklearn.svm import SVC #Support Vector Classifier
from sklearn.naive_bayes import GaussianNB #Naive Bayes
from sklearn.tree import DecisionTreeClassifier #Decision Tree Classifier
from sklearn.ensemble import RandomForestClassifier #Random Forest Classifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
import warnings
warnings.filterwarnings('ignore') # setting ignore as a parameter
```

# To Read the dataset from csv file.

```
df = pd.read_csv("/content/drive/MyDrive/ML_Research_Documents/winequality-red.csv")
df.head()
```

# Checking for Dataset skewness

```
ax = df["quality"].value_counts().plot.bar(figsize=(7,5))
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.02, p.get_height() * 1.02))

print(df["quality"].value_counts(normalize=True)*100)
```

# Checking for missing values

```
df.isnull().sum() #No missing values
```

# Probability Output

```
def get_probabilty_output(X_test, model_fitted, value_count=10):
    def highlight_max(data, color='yellow'):
        attr = 'background-color: {}'.format(color)
        if data.ndim == 1:  # Series from .apply(axis=0) or axis=1
            is_max = data == data.max()
            return [attr if v else '' for v in is_max]
        else:  # from .apply(axis=None)
            is_max = data == data.max().max()
                return pd.DataFrame(np.where(is_max, attr, ''), index=data.index, columns=data.columns)

    y_scores = model_fitted.predict_proba(X_test)
    prob_df = pd.DataFrame(y_scores*100).head(value_count)
    styled_df = prob_df.style.background_gradient(cmap='Reds')
    styled_df = styled_df.highlight_max(axis=1, color='green')
    return styled_df
```

# Analysing Feature Correlation

```
corr = df[features].corr()
```

```
plt.figure(figsize=(16,16))
sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size':
15},
         xticklabels= features, yticklabels= features, alpha = 0.7,   cmap= 'coolwarm')
plt.show()
```

# Classification Report

```
def get_classification_report(y_test,predictions,average="macro"):
    #Confusion Matrix
    cm = confusion_matrix(y_test, predictions)
    sns.heatmap(cm, annot=True)
    plt.title("Confusion Matrix")

    acc = accuracy_score(y_test, predictions)
    pre = precision_score(y_test, predictions, average=average)
    rec = recall_score(y_test, predictions, average=average)
    # Prediction Report
    print(classification_report(y_test, predictions, digits=3))
    print("Overall Accuracy:", acc)
    print("Overall Precision:", pre)
    print("Overall Recall:", rec)
    return acc,pre,rec
```

# Classification ROC

```
def get_classification_ROC(X,y,model,test_size,model_fitted=False,random_state=0):

    def check_fitted(clf):
        return hasattr(clf, "classes_")

    if(len(np.unique(y)) == 2):
        #Binary Classifier
        if not check_fitted(model):
            model = model.fit(X,y)

        RocCurveDisplay.from_estimator(model, X, y)
        y_score = model.predict_proba(X)[:, 1]
        fpr, tpr, threshold = roc_curve(y, y_score)
        auc = roc_auc_score(y, y_score)
        return auc
#                     print("False    Positive    Rate:    {}    \nTrue    Positive    Rate:    {}
\nThreshold:{}".format(fpr,tpr,threshold))
```

```
  else:
      #Multiclass Classifier
      y_bin = label_binarize(y, classes=np.unique(y))
      n_classes = y_bin.shape[1]

      # shuffle and split training and test sets
        X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=test_size,
random_state=random_state)

      # Learn to predict each class against the other
      classifier = OneVsRestClassifier(model)
      model_fitted = classifier.fit(X_train, y_train)
      try:
          y_score = model_fitted.decision_function(X_test)
      except:
          y_score = model_fitted.predict_proba(X_test)



      # Compute ROC curve and ROC area for each class
      fpr = dict()
      tpr = dict()
      roc_auc = dict()
      for i in range(n_classes):
          fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
          roc_auc[i] = auc(fpr[i], tpr[i])


      # Compute micro-average ROC curve and ROC area
      fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
      roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])


      plt.figure()
      lw = 2
      plt.plot(fpr[2], tpr[2], color='darkorange',
               lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
      plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
      plt.xlim([0.0, 1.0])
      plt.ylim([0.0, 1.05])
      plt.xlabel('False Positive Rate')
      plt.ylabel('True Positive Rate')
      plt.title('Receiver operating characteristic averaged')
      plt.legend(loc="lower right")
plt.show()
```

```
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(figsize=(10,10))
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'red', 'blue', 'purple', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('multi-class ROC (One vs All)')
plt.legend(loc="lower right")
plt.show()
```

# Visualisation Through PCA

```
def visualisation_through_PCA(X_PCA, y, model_PCA, model_name="Classification
Model"):
    X_set, y_set = X_PCA, y
    X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max()
+ 1, step = 0.01),
                         np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step
= 0.01))
                       plt.contourf(X1,          X2,          model_PCA.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
                   alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue', 'yellow', 'purple',
'grey')))
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
    for i, j in enumerate(np.unique(y_set)):
        plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                      c = ListedColormap(('red', 'green', 'blue', 'yellow', 'purple', 'grey'))(i), label
= j)
    plt.title(model_name)
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.legend()
    plt.show()
```

# Generating Principal Components of the feature dataset

```
pca = PCA(n_components = 2)
X_train_PCA_2 = pca.fit_transform(X_train)
X_test_PCA_2 = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print("Variance Explained by each of the Principal Components: {:.{prec}f}% and
{:.{prec}f}%,                \nTotal                Variance                Explained:
{:.{prec}f}%".format((explained_variance*100)[0],
```

# Logisitic Regression Model

```
parameters_LR = {
    "solver" : ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
    "penalty" : ('l1', 'l2', 'elasticnet', 'none'),
    "C" : [0.01, 0.1, 1, 10, 1000]
    }
```

```
model_LR = LogisticRegression()model_LR_with_best_params = GridSearchCV(mode
l_LR, parameters_LR)model_LR_with_best_params.fit(X_train,y_train)model_LR_best
_params = model_LR_with_best_params.best_params_
```

# k-Nearest Neighbor Model

```
parameters_KNN = {
    "n_neighbors" : [2,5,7,15],
    "weights" : ('uniform','distance'),
    "algorithm" : ('auto','ball_tree','kd_tree','brute'),
    'p': [1,2,5]

    }
model_KNN = KNN(n_jobs=-1)model_KNN_with_best_params = GridSearchCV(mod
el_KNN,  parameters_KNN)model_KNN_with_best_params.fit(X_train,y_train)model_
KNN_best_params = model_KNN_with_best_params.best_params_
```

# Support Vector Machine (SVM) Model

```
parameters_SVC = {
    "C": [0.1, 1, 10],
    "kernel": ('linear','poly','rbf'),
    "degree": [2,4]
}
model_SVC = SVC(probability=True)
model_SVC_with_best_params = GridSearchCV(model_SVC, parameters_SVC)
model_SVC_with_best_params.fit(X_train,y_train)
model_SVC_best_params = model_SVC_with_best_params.best_params_
```

# Naive Bayes Classifier Model

```
model_NB = GaussianNB()
model_NB.fit(X_train, y_train)
```

# Decision Tree Classifier Model

```
parameters_DT = {
    'criterion':('gini','entropy'),
    'max_features': ('auto','sqrt','log2')
}
model_DT = DecisionTreeClassifier()
model_DT_with_best_params = GridSearchCV(model_DT, parameters_DT)
model_DT_with_best_params.fit(X_train,y_train)
model_DT_best_params = model_DT_with_best_params.best_params_
```

```
model_DT_with_best_params.fit(X_train,y_train)
```

# Random Forest Classifier Model

```
parameters_RF = {
    'criterion':('gini','entropy'),
    'max_features': ('auto','sqrt','log2'),
    'n_estimators': [100,150,200,250,300]
}
model_RF = RandomForestClassifier(n_jobs=-1)
model_RF_with_best_params = GridSearchCV(model_RF, parameters_RF)
model_RF_with_best_params.fit(X_train,y_train)
model_RF_best_params = model_RF_with_best_params.best_params_
model_RF_with_best_params.fit(X_train,y_train)
```

# Model Comparision

```
result = pd.DataFrame(
    [["LogisticRegression",auc_LR,acc_LR,pre_LR,rec_LR],
    ["kNearestNeighbor",auc_KNN,acc_KNN,pre_KNN,rec_KNN],
    ["SupportVectorClassifier",auc_SVC,acc_SVC,pre_SVC,rec_SVC],
    ["NaiveBayes",auc_NB,acc_NB,pre_NB,rec_NB],
    ["DecisionTree",auc_DT,acc_DT,pre_DT,rec_DT],
    ["RandomForest",auc_RF,acc_RF,pre_RF,rec_RF]],
    columns=["Classifier","AUC","Accuracy","Precision","Recall"]
)
result
```