DATA SCIENCE FOUNDATIONS IN-COURSE ASSESSMENT

A REPORT ON MACHINE LEARNING MODELLING APPROACH TO PREDICT WINE QUALITY

BOWALE ADEDAMOLA
APPLIED DATA SCIENCE(MSc.)
A0353496

January 12, 2022

Abstract

Machine Learning and data analysis can be used to solve the most important problems in this project by training, predicting, testing and evaluating a model based on the wine quality dataset. The dataset used to analyze Wine Quality represents the quality of wines using various physiochemical characteristics (density, volatile acidity, residual sugar, chlorides, free Sulphur dioxide, alcohol, citric acid, total Sulphur dioxide, pH, fixed acidity, sulphates). Each wine combination in the dataset has a quality score ranging from 0 to 10 (from lowest to highest). There are 4898 instances of white wine 1599 instances of red wine in these data sets, and 4898 instances of white wine in them. There are 7 quality classes of wine in the dataset which were later grouped into 2 classes ("0" and "1") where values less than 6 are grouped as "0" and values greater than 6 are grouped as 1.

To compare different feature sets and supervised machine learning models, different performance metrics such as sensitivity, accuracy, Recall specificity, F1 Score,. Etc. were used, and in terms of Accuracy, Random Forest was the highest compared to Logistic regression and to contrast various feature sets and supervised machine learning algorithm we have used different performance measures such as accuracy, sensitivity, specificity (SVM) with an accuracy of 73.89% while logistic regression has 72.32% and Random Forest 73.63%.

Preliminary analysis using statistical methods reveals which elements have the greatest influence on wine's quality. This will assist wine producers in monitoring the quality of their wine before it is actually made.

Specification of Problem

A fine wine If taken properly, prevents heart disease, heart attack, stroke, arterial hardening (atherosclerosis), and chest pain (angina). However, a bad wine can exacerbate the already-known adverse effects of drinking more than a moderate amount of wine, including flushing, confusion, or rapid mood changes, blackouts, difficulty walking, seizures, vomiting, and diarrhea, among other serious problems.

I chose this dataset in order to use Machine Learning to address the issues caused by subpar wines. Similar to other applications, the goal of machine learning is to develop models from data in order to predict the quality of wine. If the features of the physicochemical values are known, machine learning can assist us in predicting the outcome of a wine.

Table of Contents

List of Figures and Table

1. Introduction and Background

Wine consumption has risen slightly in recent years due to research showing a link between wine consumption and heart rate variability. [1]. Wine manufacturers are in search for cost-effective ways to make high-quality wine in order to meet growing demand. Wines come in a variety of varieties and are used for a variety of purposes. Chemical tests have revealed that the majority of chemical compounds are similar for other types of wine, however, the magnitude of chemicals embedded in each chemical varies according to the type of wine. [2].

Historically, wine quality was determined through post-production testing; attaining that level already requires considerable time and money. If the quality is poor, numerous procedures must be implemented from the start, which is extremely costly. It's difficult to identify a product's quality based on a person's taste because everyone has a different take on what tastes good [3].

The wine industry is investing in cutting-edge technology to appreciate its rapid growth. In this context, assessment of quality and certification of wine are critical components. Certification guides the public health by not allowing wines from being illegally adulterated and by guaranteeing the quality of wines sold in the market [4]. Quality evaluation is frequently included in the accredited process, and it could be used to both enhance wine production (by categorising the most important variables) and to classify wines, such as luxury brand, into different groupings that are useful for pricing. [4].

Wine certification is generally determined by sensory attributes tests [5]. pH values, Density, and alcohol are all examples of physical and chemical tests conducted routinely used to characterize wine, whereas sensory assessments rely primarily on human specialists. As aforementioned, taste is the smallest comprehended of the human senses, which complicates wine classification.

Additionally, the relationships between sensory attributes assessments are complex and researchers are still working to understand fully. [6].

Massive amounts of complex data can now be collected, stored, and processed thanks to technological advances. You can use these trends and patterns to improve your decision-making and give the best results in various situations by analyzing this data. [7]. Classifying wines and determining which chemical analysis parameters are most important can now be done using machine learning techniques, which have been around for a while now [8].

Methods for developing models that accurately predict the quality of red and white wine using three machine learning algorithms will be described in this paper (Support Vector Machine [SVM]), Logistic Regression, and Random Forest).

2. Literature Review

Y. Subba Reddy et al. [9] developed an algorithm to determine how similar products are to each other. With the help of the well-known "Red wine quality" dataset, researchers have been able to classify and grade individual wines based on users. Conventional techniques, which do not come into play customers' expressed preferences, yielded quite distinctive and intriguing results when compared to the user-centric approach. The query types presented in this project required longer execution times as the number of consumers and their preferences grew. The planned framework's scalability may be compromised as a result. There are ways of reducing the execution time of the proposed framework by integrating R-tree-like data structures for browsing and indexing. Making this kind of system is aimed at helping wine drinkers make better choices and winemakers make better wines is the point of it all.

Additionally, it provides an in-depth examination of recent research patterns in wine quality and consumer similarity metrics. The Red Wine dataset, a popular Wine data set, is being evaluated using a new user-centric similarity measure in product clustering. The findings of this study are capable of recommending better products to consumers than current systems. It is possible to use the Red wine dataset to group users' preferred wine variants into ordered groups, and then use these user preference groups to assess the wine quality [9].

Priority-based clustering is proposed for the wine dataset records. In order to assign a recommended voting label to the test data records, we clustered the data using classification. The vast majority of prior wine data research focused on traditional clustering and classification methods that relied on data from tasters, while the innovative hybrid model proposed here might suggest superior wine pairings without depending on taster detecting data at all [9].

In a survey of almost 3,000 people from three countries, Tim et al. [10] discovered that polyphenolic compounds in grape skin may be responsible for many of the health benefits associated with moderate consumption of red wine. The study also found that red wine consumption was linked to lower rates of obesity and "poor" cholesterol., in part because of gut microbiota. "Red wine consumption and gut microbiota diversity have been found to be associated, but it appears that consuming red wine infrequently (once every two weeks) is sufficient to detect an effect. Even so, Dr. Le Roy [10] cautioned, "moderation in alcohol consumption is still recommended.

Physicochemical data, according to Yesim Eret al. [11], will be used to predict wine quality. There were two large sets of data, both from the UCI Machine Learning Repository, used in this study. There are 4898 white wine samples and 1599 red wine samples in these data sets, each with 11 physicochemical characteristics, such as chlorides, total sulphur dioxide, alcohol, free sulphur dioxide, and pH, to name a few. Using Random Forests technique, 99.523 percent of the instances were correctly classified as white or red wine. Three different data mining algorithms were then used to classify white and red wine quality: random forests, k-nearest neighbor, , and support vector machines (SVM). Red wines are divided into six distinct classifications, while white wines are divided into seven distinct classifications. The Random Forests Algorithm was used to achieve the most accurate classification.

Random Forests Algorithm's classification accuracy improves when principal component analysis (PCA) is used in the feature selection process. For each classification algorithm, we examined how the results are different when the test mode is modified. ' There are two types of wine data used in this study: red and white. The percentage of correct classifications, precision, recall, F-measure, and ROC are presented following percentage split mode. Classifiers such as random forests, k-

nearest-neighborhood, and SVMs are all put to the test on large datasets in order to determine which is the most effective. Random Forests Algorithm outperforms the SVM classifier and k nearest neighbors in classification tasks. According to the results of our experiments. PCA reduced the cross-validation success rate for white wine quality classification from 70.3757 percent to 69.9061 percent. According to the percentage split method, the success rate for white wine quality classification has dropped from 68% to 67% [11].

3. Technical Implementation (Developed System or Model)

3.1. Dataset

The data set is a freely accessible dataset on wine quality that can be found at https://www.kaggle.com/ishanoze15/winequality-eda/data [13]. The dataset includes 1599 cases of red wine with eleven distinct characteristics and 4898 cases of white wine with much the same eleven distinct characteristics. Objective assessments (for instance, pH values) are used as inputs, while sensory data is used as output. Each expert assigned a score between 0 (very poor) and 10 to the wine's quality (very excellent). The dataset contains information about red and white varieties of Portugal's "Vinho Verde" wine. Citric acid, Fixed acidity, volatile acidity, total Sulphur dioxide, residual sugar, chlorides, density, free Sulphur dioxide, pH, sulphates, and alcohol are just some of the characteristics. On a scale of 0 to 14, pH indicates how acidic or basic a wine is (. The pH of the majority of wines lies between 3 and 4.

According to this data set, a wine expert's rating can be predicted using a variety of chemical characteristics, such as acidity and alcohol content. Only input and output variables are available due to privacy and logistical constraints (e.g., data on grape varieties, wine brand, and wine selling price are not available).

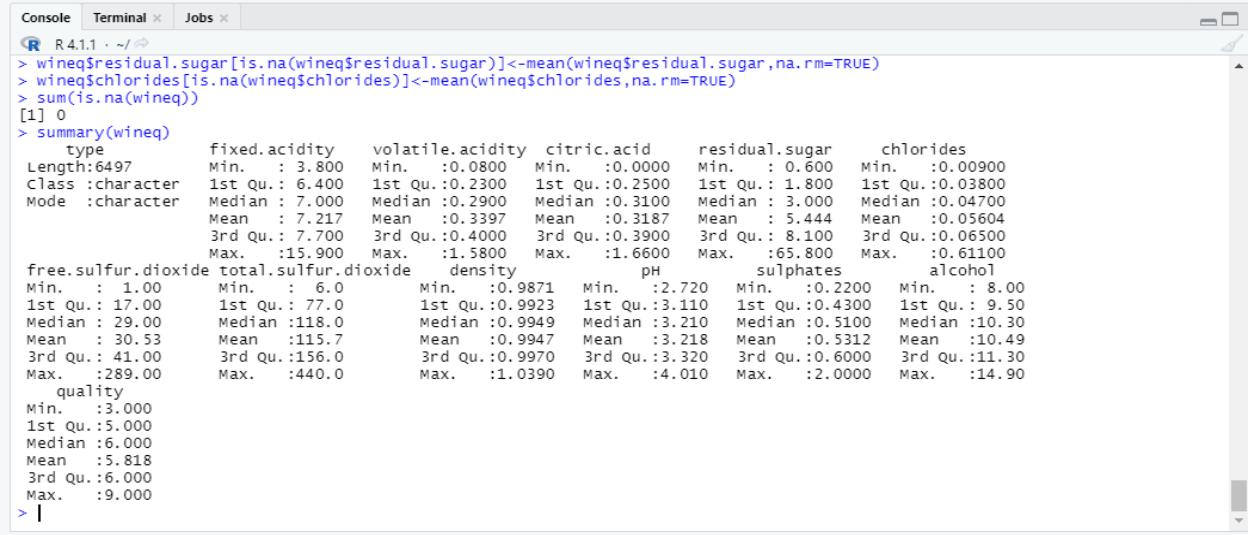

Figure 3. 1: Statistical summary of all attributes

	MIN	1ST QUARTILE	MEDIAN	MEAN	3RD QUARTILE	MAX
FIXED ACIDITY	3.800	6.400	7.000	7.217	7.700	15.900
VOLATILE ACIDITY	0.0800	0.2300	0.2900	0.3397	0.4000	1.5800
CITRIC ACID	0.0000	0.2500	0.3100	0.3187	0.3900	1.6600
RESIDUAL SUGAR	0.600	1.800	3.000	5.444	8.100	65.800
CHLORIDES	0.00900	0.03800	0.04700	0.05604	0.06500	0.61100
FREE SULPHUR DIOXIDE	1.00	17.00	29.00	30.53	41.00	289.00
TOTAL SULFUR DIOXIDE	6.0	77.0	118.0	115.7	156.0	440.0
DENSITY	0.9871	0.09923	0.9949	0.9947	0.9970	1.0390
PH	2.720	3.110	3.210	3.218	3.320	4.010
SULPHATES	0.2200	0.4300	0.5100	0.5312	0.6000	2.0000
ALCOHOL	8.00	9.50	10.30	10.49	11.30	14.90
QUALITY	3.000	5.000	6.000	5.818	6.000	9.000

Table 3. 1: Statistical summary of attributes

3.2. Data Pre-Processing

Pre-processing data is the first step before designing a model. This step examines the distribution of data across various columns. Checking for null values is the first step of this preprocessing stage. After checking the dataset for missing values, we found some which were later filled using the mean of the columns in question. Figure 3.2 and Figure 3.3 shows us the visualization of the missing values in the dataset.

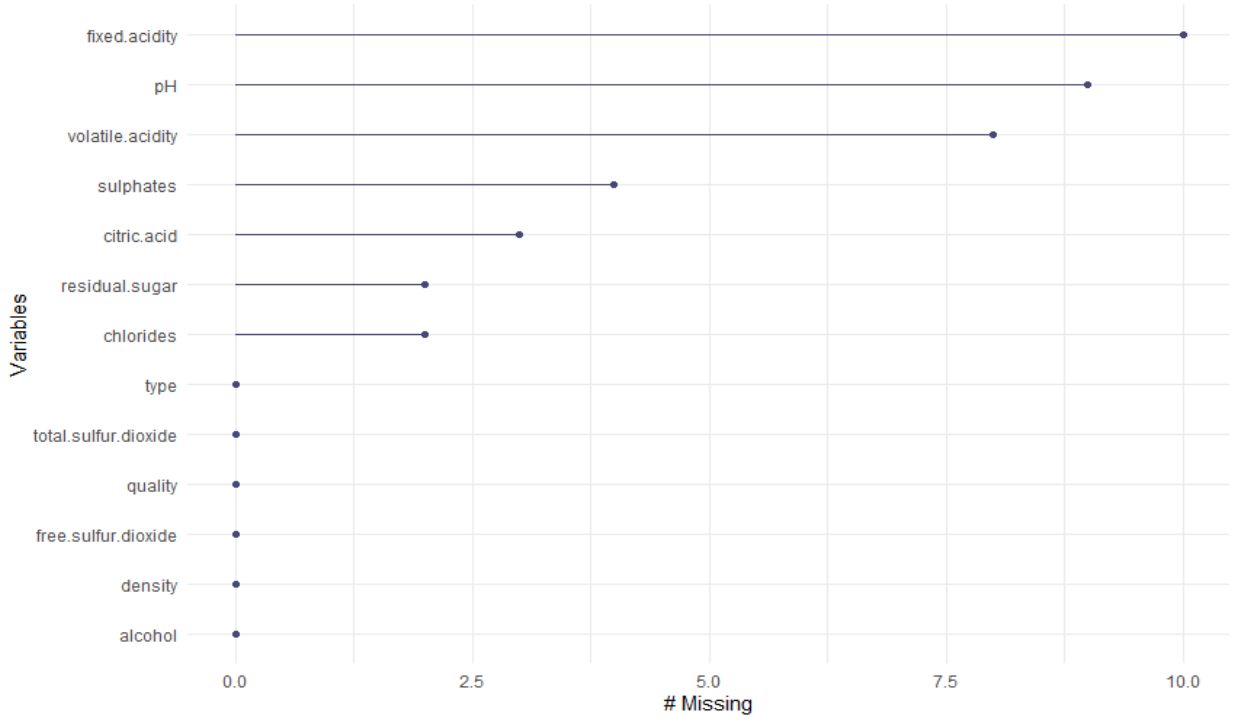

Figure 3. 2: Missing Values stacked visualization

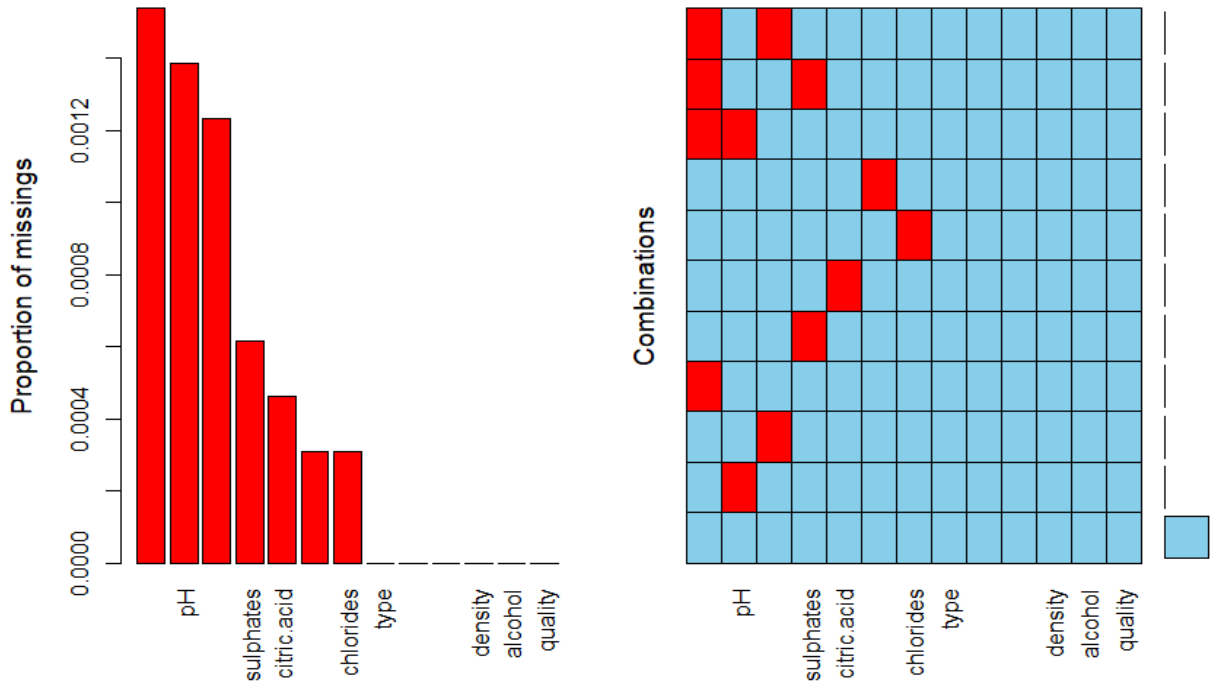

Figure 3. 3: Missing Values column visualization

3.3. Exploring that Dataset further

Data processing, cleaning, analysis, and visualization are all steps in the standard machine learning workflow. However, the cleaning has been completed, and this section will explore and visualize the data in order to gain a thorough understanding of the dataset and to assist us in properly analyzing it. Figures 3.4 and 3.5 provide an overview of the dataset's structure. The columns are skewed to the right in the majority of cases.

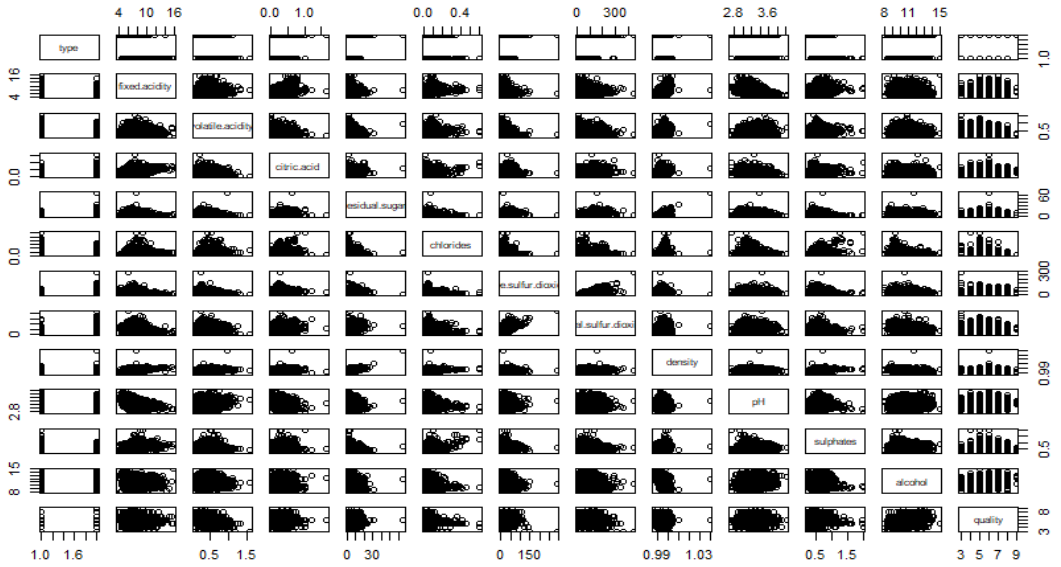

Figure 3. 4: Scatter plot

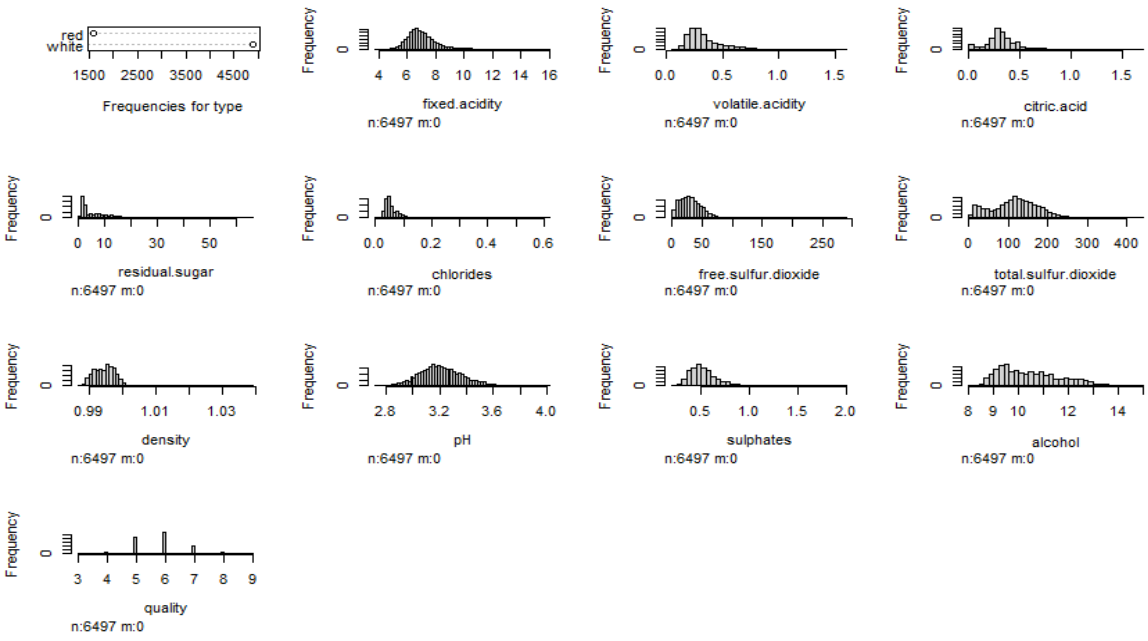

Figure 3. 5: Histogram plot

- Correlation analysis

The statistical technique of correlation analysis is used to determine the relationship strength between two or more numerically measured continuous variables. This method is most useful for identifying potential relationships between different variables. [14].

Whenever one parameter undergoes a methodical shift, the other is likely to follow suit; the variables shift together in lockstep over time. [14]. Figure 3.6 shows us that there are no highly correlated columns

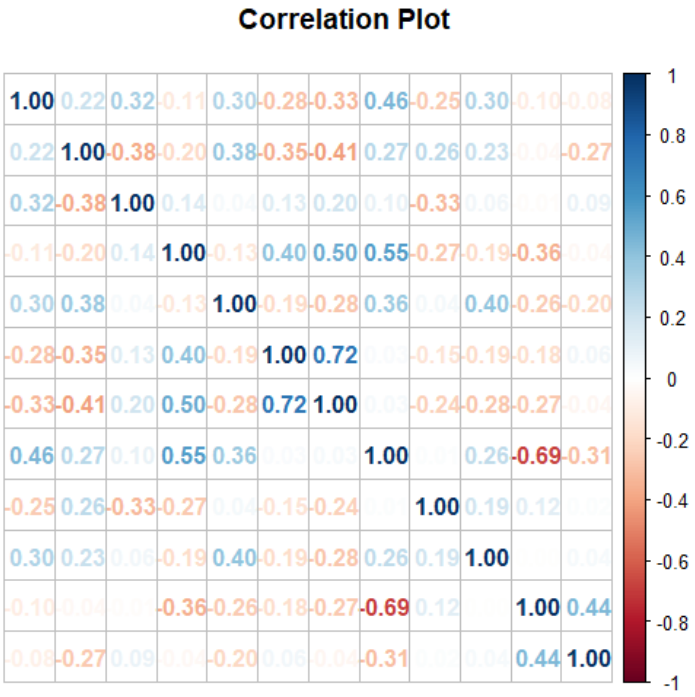

Figure 3. 6: Correlation plot

It was also necessary to comprehend the proportion of red wine to white wine in the dataset. Using Figure 3.6, we can see how white and red wine are distributed throughout the dataset.

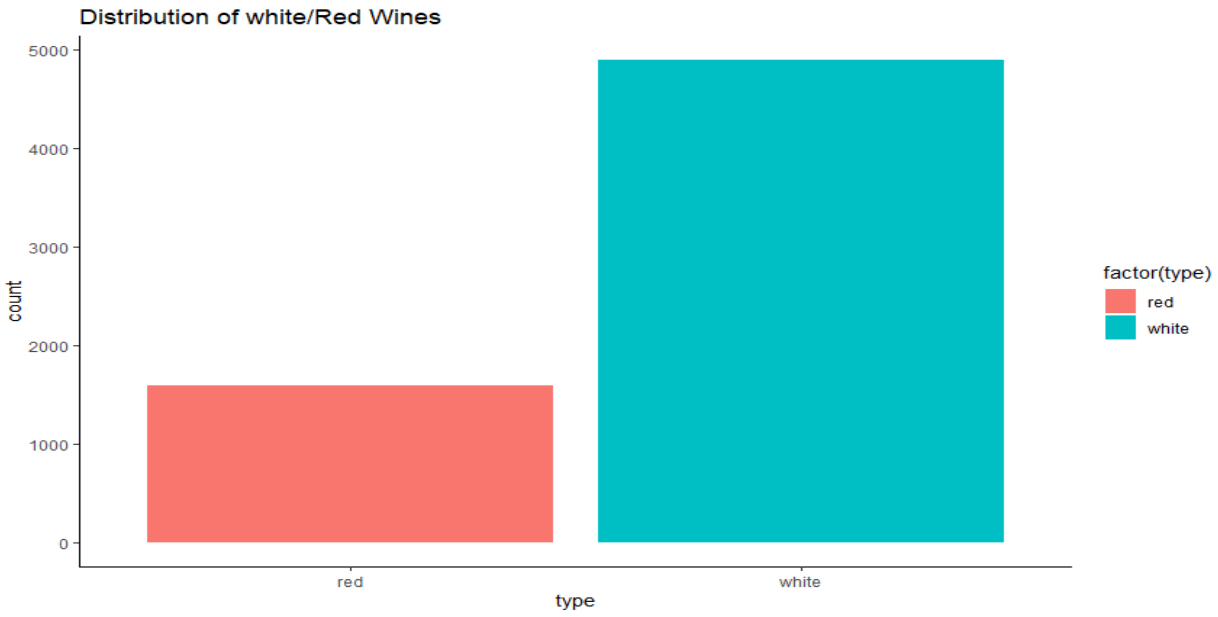

Figure 3. 7: Distribution of White/Red Wines

The Quality column in the wine dataset is the target column with 7 distinct values. The values in the column are the ratings of the quality of the wine from 1 to 10 (1 being bad and 10 being excellent.

Please, see Figure3.7 and Figure 3.8 below.

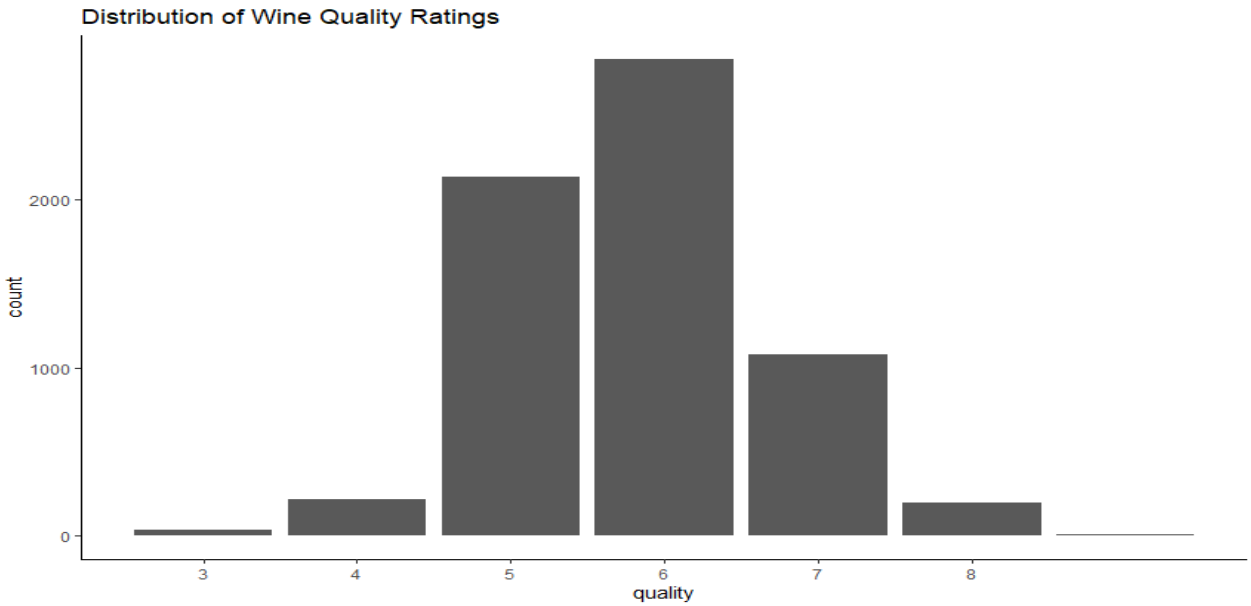

Figure 3. 8:Distribution of Wine quality ratings

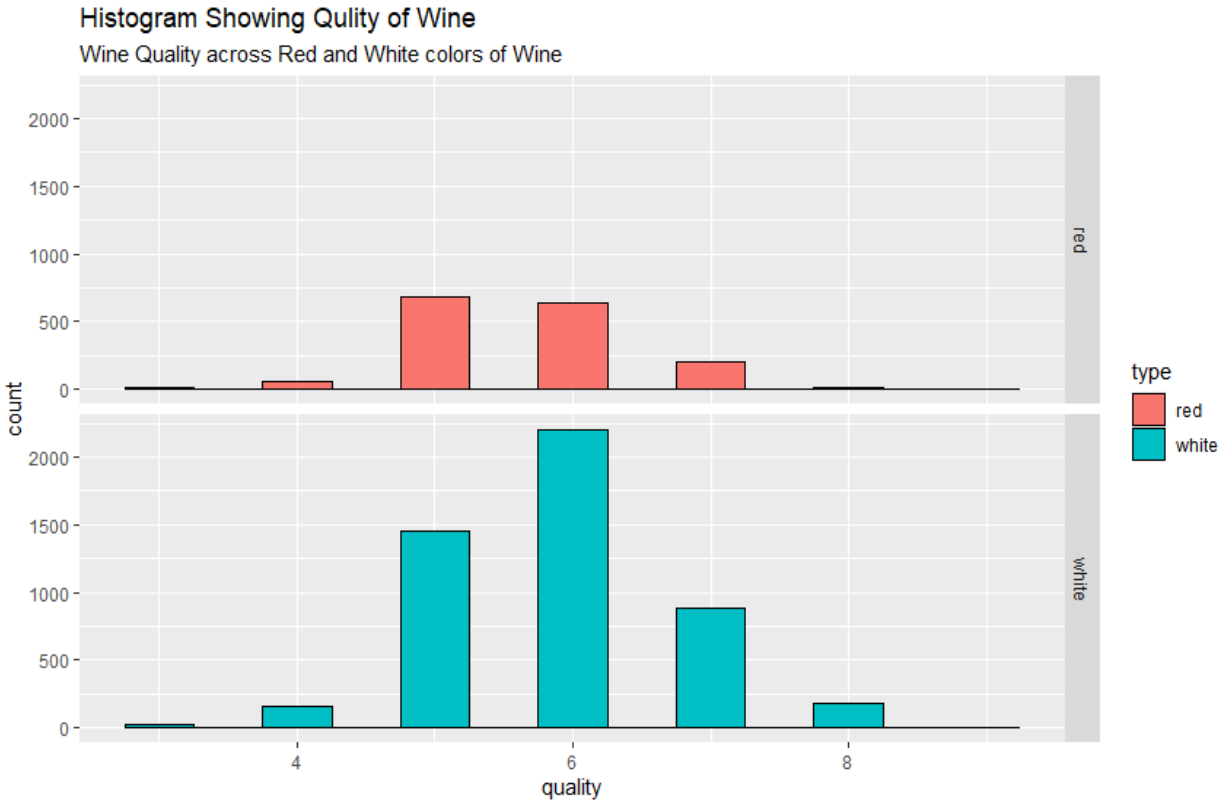

Figure 3. 9: Distribution of wine quality across red wine and White wine

- Downsampling

There are several quality ratings classes in the wine quality dataset, ranging from 3 to 9. The focus is on the quality label variable as it is the target column, however the values were categorized into “good” ratings (represented as 1 if rating is greater than 6) and “bad” ratings (represented as 0 if rating is less than 6). This is because obvious imbalance and also several rating scores contain a small number of wine samples, and thus similar quality ratings were combined into a single quality class rating (*figure 3.9*).

Imbalanced classification occurs when one class considerably vastly outnumber the other. This occurs more frequently in binary classification than multi-level classification. [14].

The term " Imbalanced " refers to the difference between the independent and dependent variables. As a result, an Imbalanced proportion of classes exists in the dependent variable. As a result of the unequal distribution of classes, a data set is Imbalanced. [14].

Even after categorizing the target column, Figure 3.9 demonstrates that there is still an imbalance in the column. One strategy for resolving such a class imbalance is to downsample the dataset in such a way that the issues are mitigated. Downsampling all classes in the training set at random to ensure that their class frequencies match the least prevalent class. Figure 3.10 shows us a visualization of the dataset after the downsampling process was carried out.

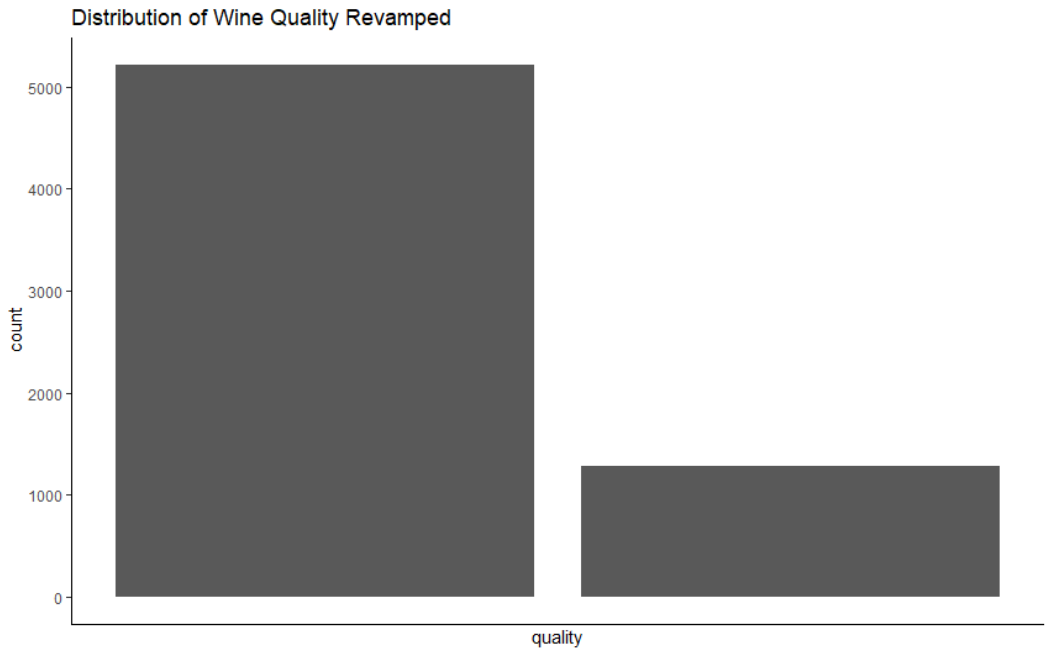

Figure 3. 10: categorized target column

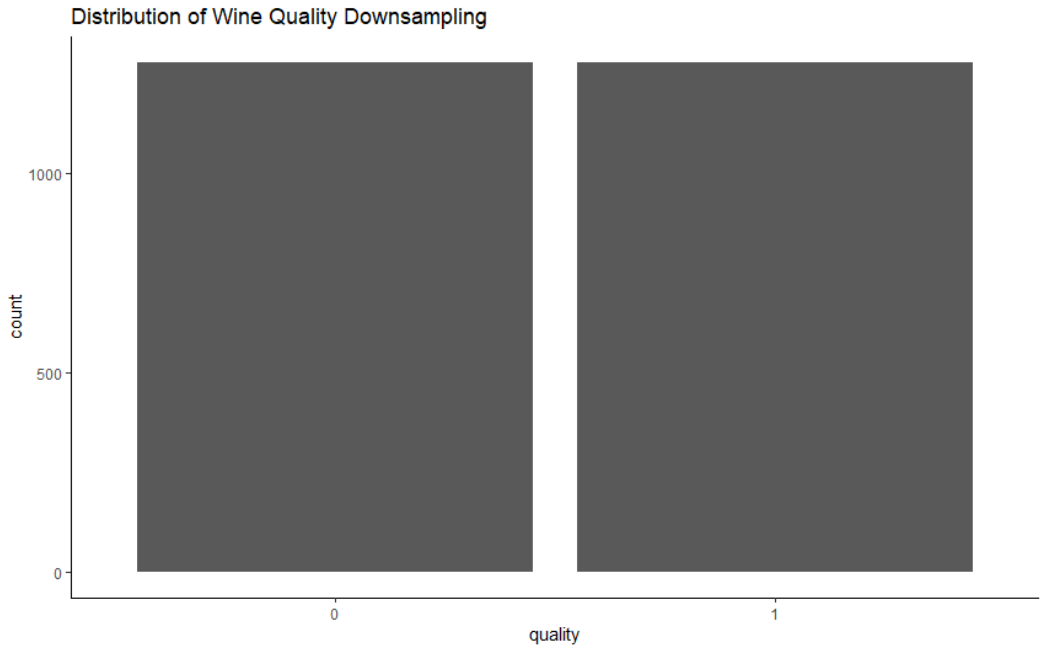

Figure 3. 11: Categorized target column after downsampling

- Feature Scaling (Normalization)

Feature scaling removes measurement units from variables in a dataset. It is commonly used to enhance machine learning accuracy. Standardization and normalization are two of the most frequently used methods for scaling features. While the terms are frequently used interchangeably, they actually mean quite different things. Standardization is the process of converting data to have a mean of 0 and a standard deviation of 1, whereas normalization converts data to a range of 0 to 1 [16]. For the purpose of this project, the Normalization process was carried out.

- Split/Train/Test

Train and test sets must be separated in Marching Learning. Linear regression, Random Forest, Naive Bayes classification, Logistic regression and decision trees are examples of supervised learning algorithms that require data to be partitioned into training and testing sets.

The idea is to train the model with observations from the training dataset and then use it to predict from the testing dataset. Splitting the training dataset helps to avoid overfitting and improves the accuracy of the training dataset [17]. The dataset was split in ratio 70% to 30% percent for Training and Testing respectively.

4. Performance Evaluation

At this stage, the dataset would have been cleaned, explored, encoded, downsampled, normalized and would have also split the data into training and test. In this project, three different Machine Learning techniques; Logistic, random forests (RF) and Regression Support vector machines (SVM)were all used to classify the data.

4.1. Support vector machines (SVMs) Model Evaluation

The implementation of an SVM is slightly different than that of other machine learning techniques. It has the capability of classifying, regressing, and detecting outliers. Support Vector Machine is a discriminative classifier with a separative hyperplane as its formal design. It is a representation of examples as points that are mapped in such a way that points belonging to different categories are separated by the largest possible gap. Additionally, an SVM is capable of performing non-linear classification [18].

Considering how the Support Vector Machine operates. The primary goal of a support vector machine is to optimize the segregation of the input data [18]. The steps carried out in this stage was to apply the Model on the training data and predict using the test data after which the Confusion Matrix (and other statistics), Precision, Recall and F1 Score were calculated

4.1.1. Performance Evaluation Metrices (SVM)

Metrics	SVM
Accuracy	73.89%
Kappa	47.78%
Sensitivity	67.89%
Specificity	79.90%
Pos Pred Value	77.15%
Neg Pred Value	71.33%
Prevalence	50.00%
Detection Rate	33.94%

Class	PRECISION	RECALL	F1 SCORE
0	6.79%	77.15%	72.22%
1	79.90%	71.33%	75.37%

Detection Prevalence	43.99%
Balanced Accuracy	73.89%

Table 4. 1: Performance Evaluation Metrices (SVM)

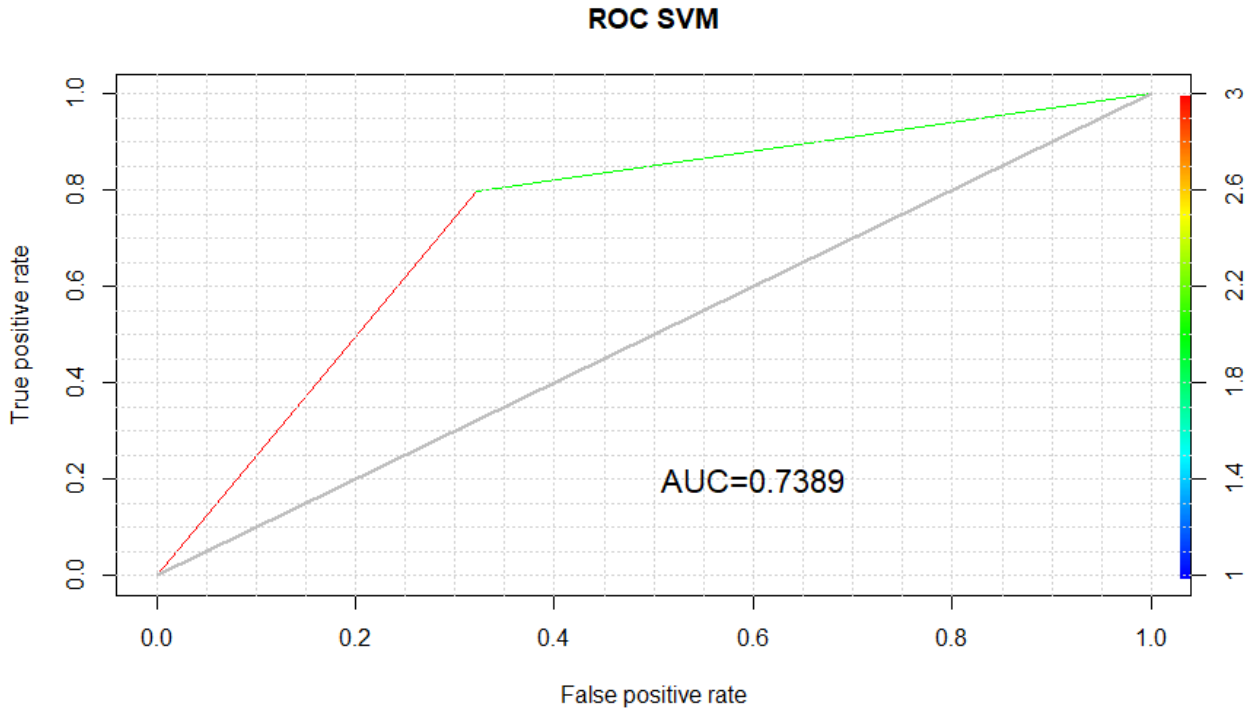

Figure 4. 1: Support Vector Machine (SVM) ROC Curve

4.2. Random Forest Model Evaluation

The Random Forest algorithm is a method for supervised classification. As implied by the name, the objective is to create a forest in some manner and to make it random. There is a direct correlation between the how many trees in a forest and how accurate the results are: the higher the number of trees, the more precise result it gives. However, it is worth noting that creating the forest is not synonymous with developing the decision using the information gain or gain index approach [4]. The steps carried out in this stage was to apply the Model on the training data and predict using

the test data after which the Confusion Matrix (and other statistics), Precision, Recall and F1 Score were calculated.

4.2.1. Performance Evaluation Metrices (Random Forest)

Metrics	RF
Accuracy	73.63%
Kappa	47.26%
Sensitivity	65.27%
Specificity	81.98%
Pos Pred Value	78.37%
Neg Pred Value	70.25%
Prevalence	50.00%
Detection Rate	32.64%
Detection Prevalence	41.64%
Balanced Accuracy	73.63%

Class	PRECISION	RECALL	F1 SCORE
0	65.27%	78.37%	71.23%
1	81.98%	70.25%	75.66%

Table 4. 2: Performance Evaluation Metrices (RF)

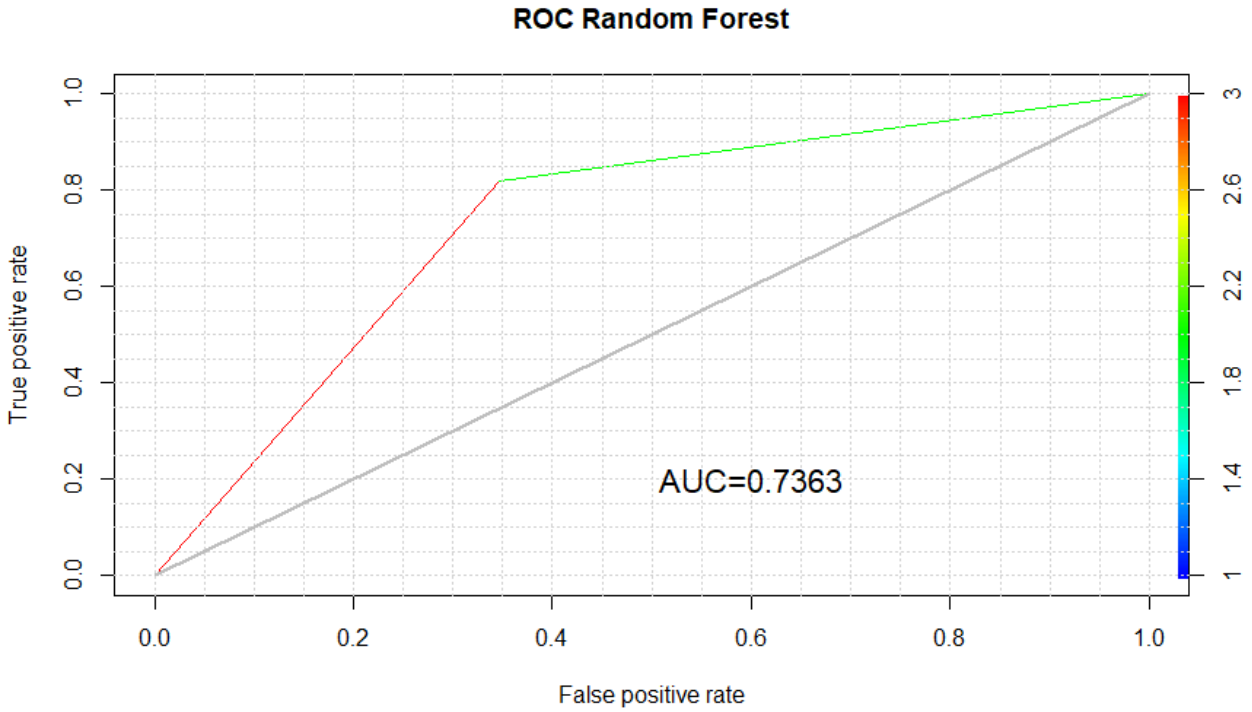

Figure 4. 2: Random Forest ROC Curve

4.3. Logistic Regression Model Evaluation

Using a binary dependent variable in a regression model is called logistic regression. The data as well as the correlation between the two binary variable and one or more nominal, proportion independent variables are accounted for using this regression technique. An individual's behaviour pattern and ability to predict an outcome can be determined using the logistic regression analysis, which is widely used. The answer is either "yes" or "no." This regression analysis is commonly used in every organization, and makes decisions based on the values predicted. As a matter of fact, the day-to-day operations of an organization are affected. An additional use for this software is that it can aid in risk assessment. [18].

The steps carried out in this stage was to apply the Model on the training data and predict using the test data after which the Confusion Matrix (and other statistics), Precision, Recall and F1 Score were calculated.

4.3.1. Performance Evaluation Metrices (Logistic Regression)

Metrics	RF
Accuracy	72.32%
Kappa	44.65%
Sensitivity	68.41%
Specificity	76.24%
Pos Pred Value	74.22%
Neg Pred Value	70.70%
Prevalence	50.00%
Detection Rate	34.20%
Detection Prevalence	46.08%
Balanced Accuracy	72.32%

Class	PRECISION	RECALL	F1 SCORE
0	68.41%	74.22%	71.20%
1	76.24%	70.70%	73.37%

Table 4. 3: Performance Evaluation Metrices (Logistic Regression)

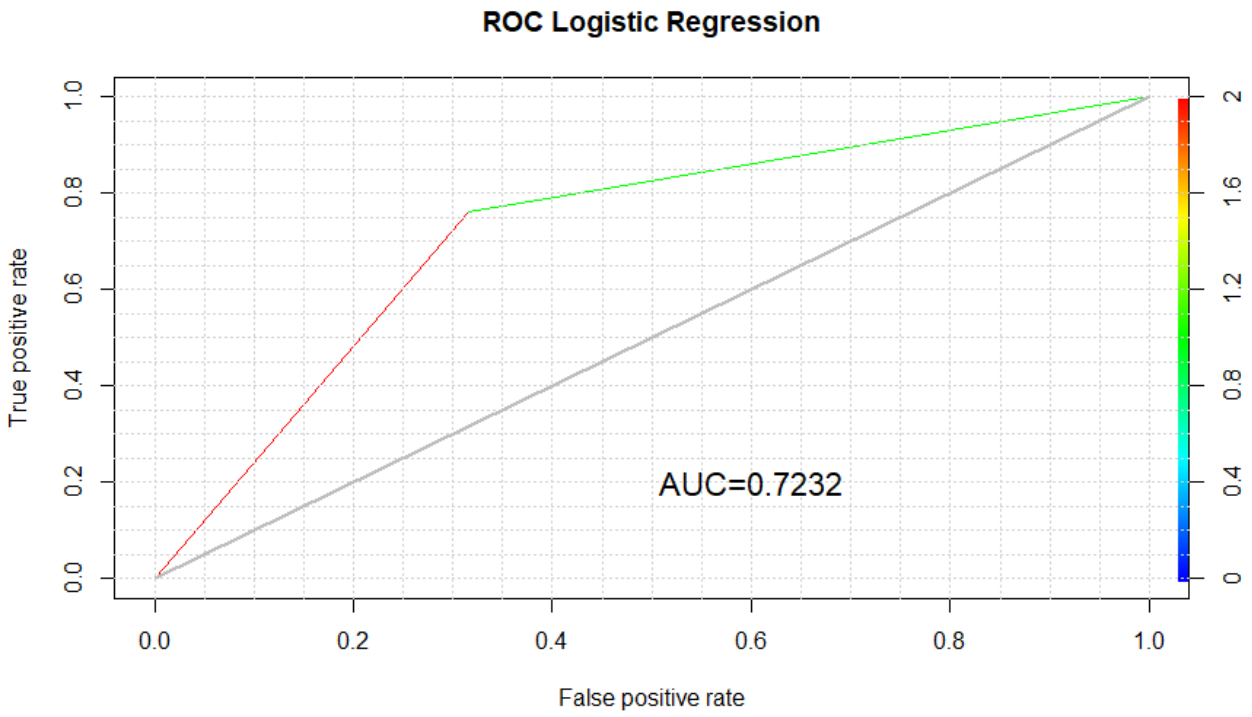

Figure 4. 3: Logistic Regression ROC Curve

4.4. Parameter Tuning

The process of hyperparameter tuning is to determine the optimal combination of hyperparameters that will allow the model to perform optimally. Setting the optimal combination of hyperparameters is the only way to maximize model performance.

Choosing the optimal hyperparameter combination is not an easy task. There are two methods for configuring them.

Manual hyperparameter tuning: This method involves manually setting (and experimenting with) various combinations of hyperparameters. This is a time-consuming process that is impractical when there are numerous hyperparameters to try.

Automated hyperparameter tuning: This method finds the optimal hyperparameters by automating and optimizing the process.

The manual parameter tuning was applied on all the models. The steps carried out in this stage was to apply the Model with the tuned parameters on the training data and predict using the test data for all the models after which the Confusion Matrix (and other statistics), Precision, Recall and F1 Score were calculated.

4.4.1. Performance Evaluation Metrices (SVM vs SVM-Tuned Parameters)

Metrics	SVM	SVM PT
Accuracy	73.89%	74.15%
Kappa	47.78%	48.30%
Sensitivity	67.89%	77.29%
Specificity	79.90%	71.66%
Pos Pred Value	77.15%	68.41%
Neg Pred Value	71.33%	79.90%
Prevalence	50.00%	44.26%
Detection Rate	33.94%	34.20%
Detection Prevalence	43.99%	50.00%
Balanced Accuracy	73.89%	74.47%

	Class	PRECISION	RECALL	F1 SCORE
SVM	0	67.90%	77.15%	72.22%
	1	79.90%	71.33%	75.37%
SVM PT	0	77.29%	68.41%	72.58%
	1	71.66%	79.90%	75.56%

Table 4. 4: SVM vs SVM PT Metrics Comparison

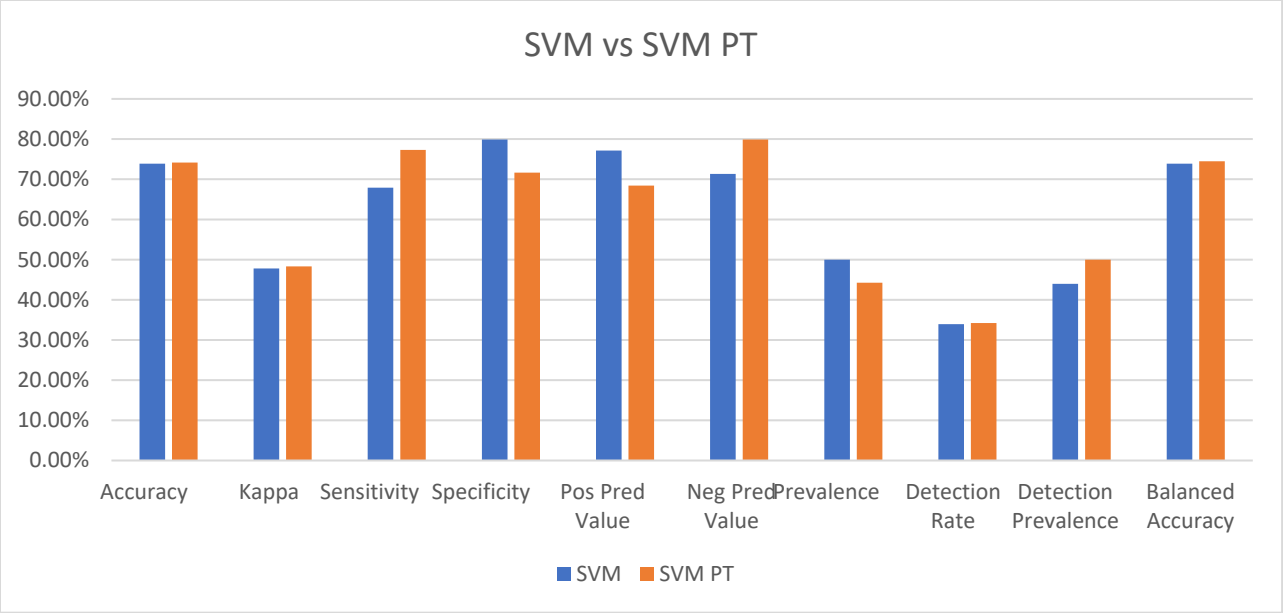

Figure 4. 4: SVM vs SVM PT Metrics Comparison

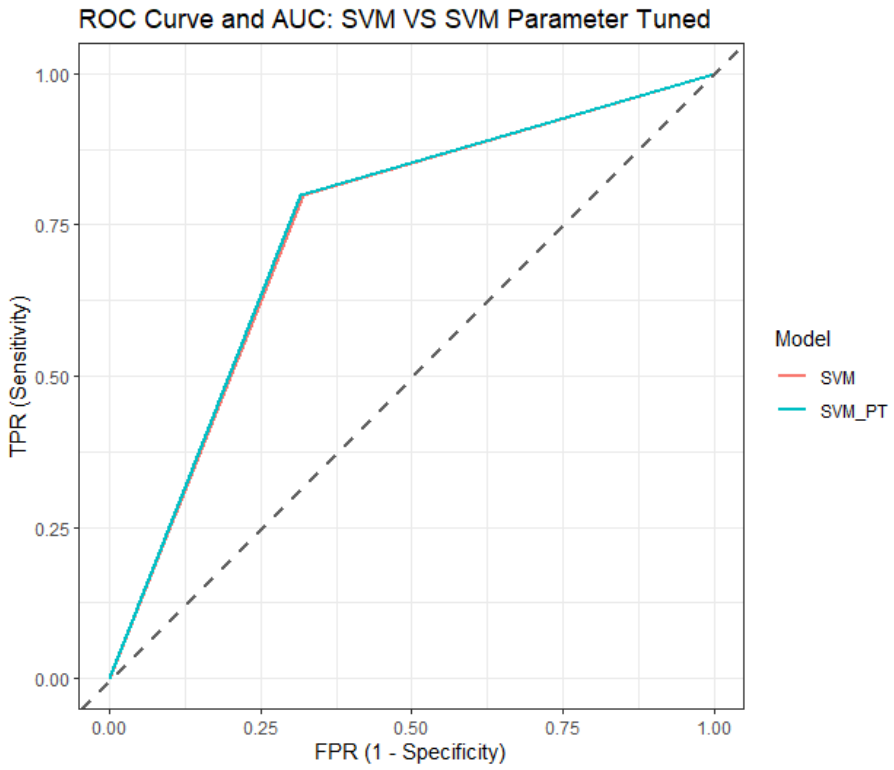

Figure 4. 5: SVM vs SVM PT ROC Curve Comparison

4.4.2. Performance Evaluation Metrices (RF vs RF-Tuned Parameters)

Metrics	RF	RF PT
Accuracy	73.63%	79.37%
Kappa	47.26%	58.75%
Sensitivity	65.27%	75.20%
Specificity	81.98%	83.55%
Pos Pred Value	78.37%	82.05%
Neg Pred Value	70.25%	77.11%
Prevalence	50.00%	50.00%
Detection Rate	32.64%	37.60%
Detection Prevalence	41.64%	45.82%
Balanced Accuracy	73.63%	79.37%

	Class	PRECISION	RECALL	F1 SCORE
RF	0	65.27%	78.37%	71.23%
	1	81.98%	70.25%	75.66%
RF PT	0	75.20%	82.05%	78.47%
	1	83.55%	77.11%	80.20%

Table 4. 5: RF vs RF PT Metrics Comparison

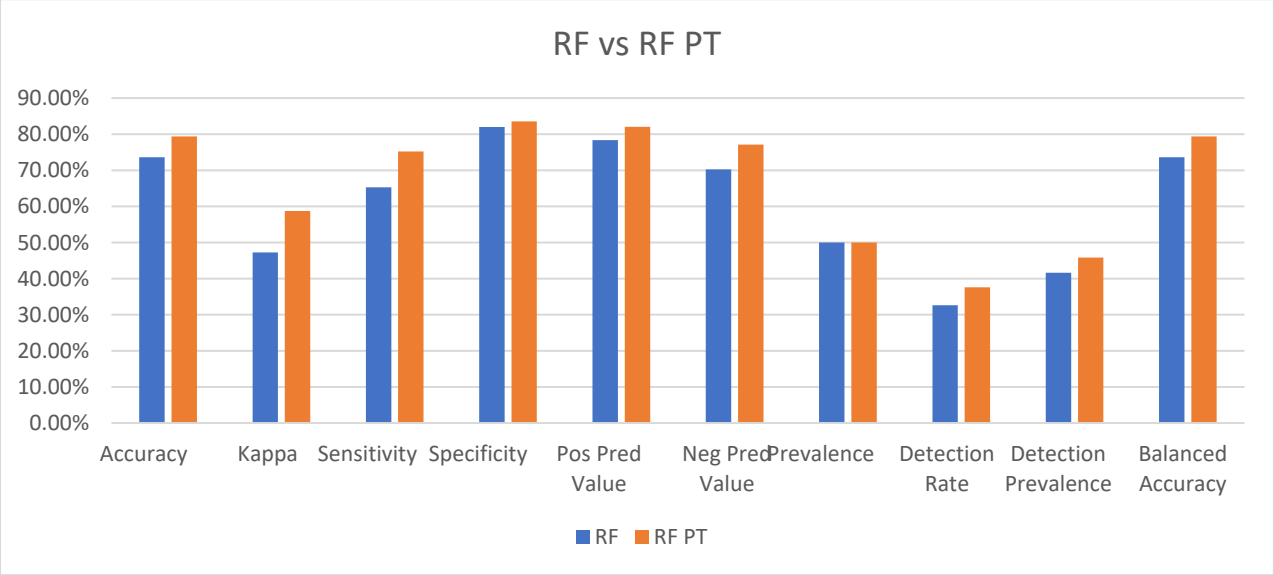

Figure 4. 6: RF vs RF PT Metrics Comparison

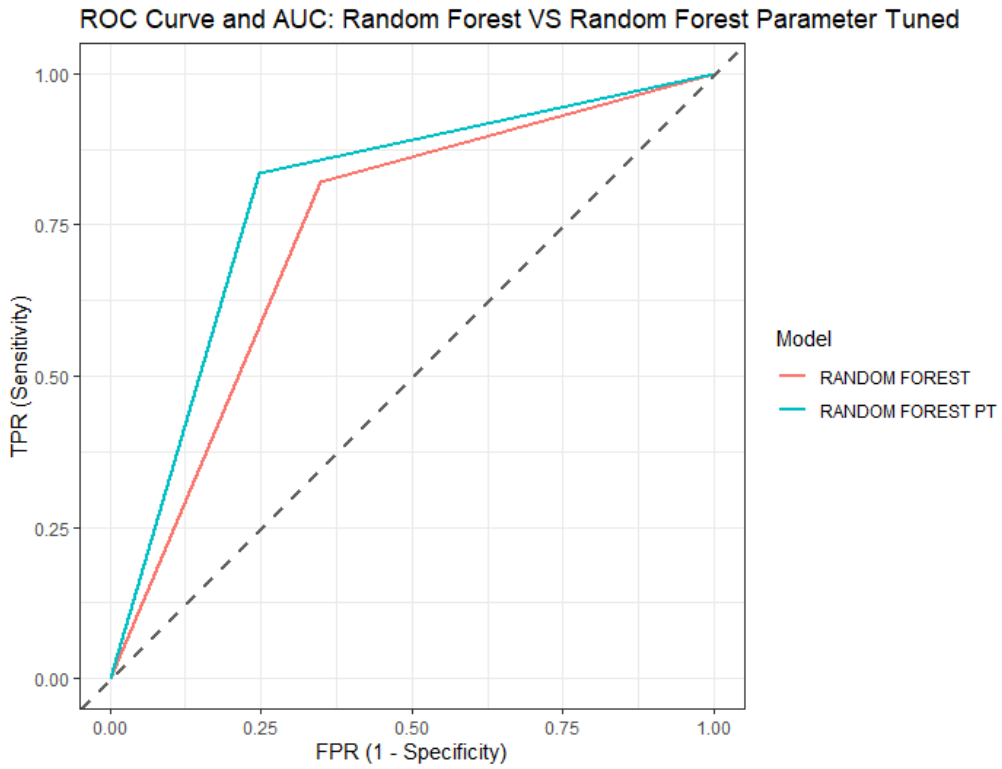

Figure 4. 7: RF vs RF PT ROC Curve Comparison

4.4.3. Performance Evaluation Metrices (LR vs LR-Tuned Parameters)

Metrics	LR	LR PT
Accuracy	72.32%	72.98%
Kappa	44.65%	45.95%
Sensitivity	68.41%	64.49%
Specificity	76.24%	81.46%
Pos Pred Value	74.22%	77.67%
Neg Pred Value	70.70%	69.64%
Prevalence	50.00%	50.00%
Detection Rate	34.20%	32.25%
Detection Prevalence	46.08%	41.51%
Balanced Accuracy	72.32%	72.98%

	Class	PRECISION	RECALL	F1 SCORE
LR	0	68.41%	74.22%	71.20%
	1	76.24%	70.70%	73.37%
LR PT	0	63.71%	77.96%	70.11%
	1	81.98%	69.32%	75.12%

Table 4. 6: LR vs LR PT Metrics Comparison

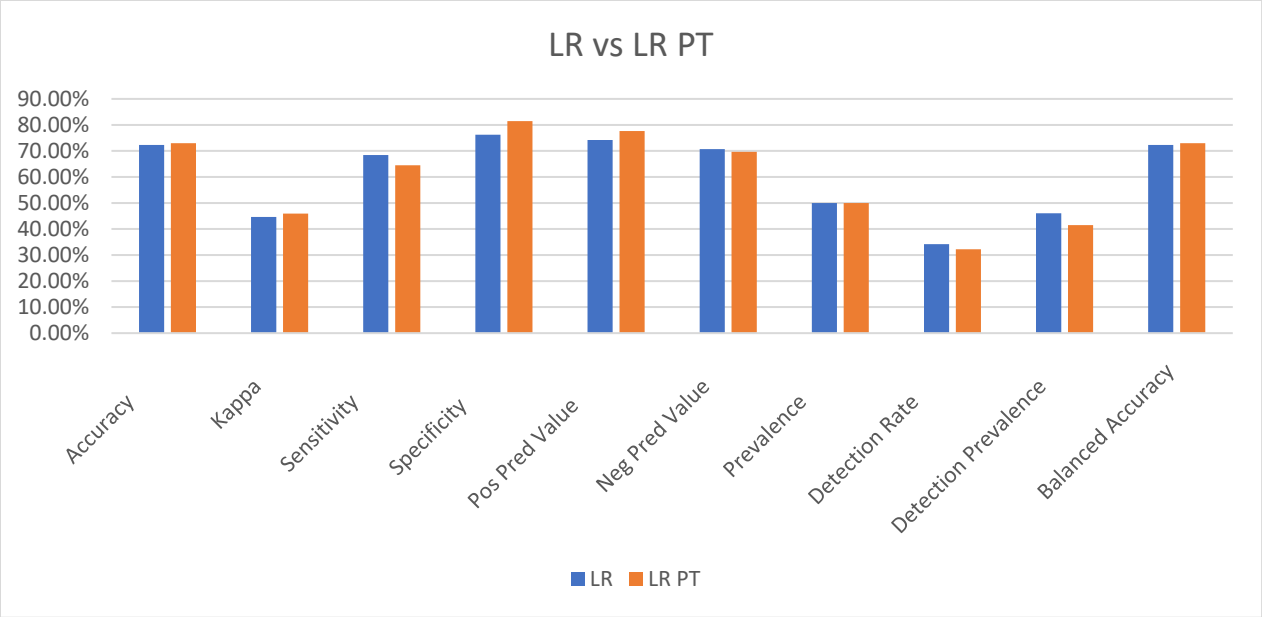

Figure 4. 8: LR vs LR PT Metrics Comparison

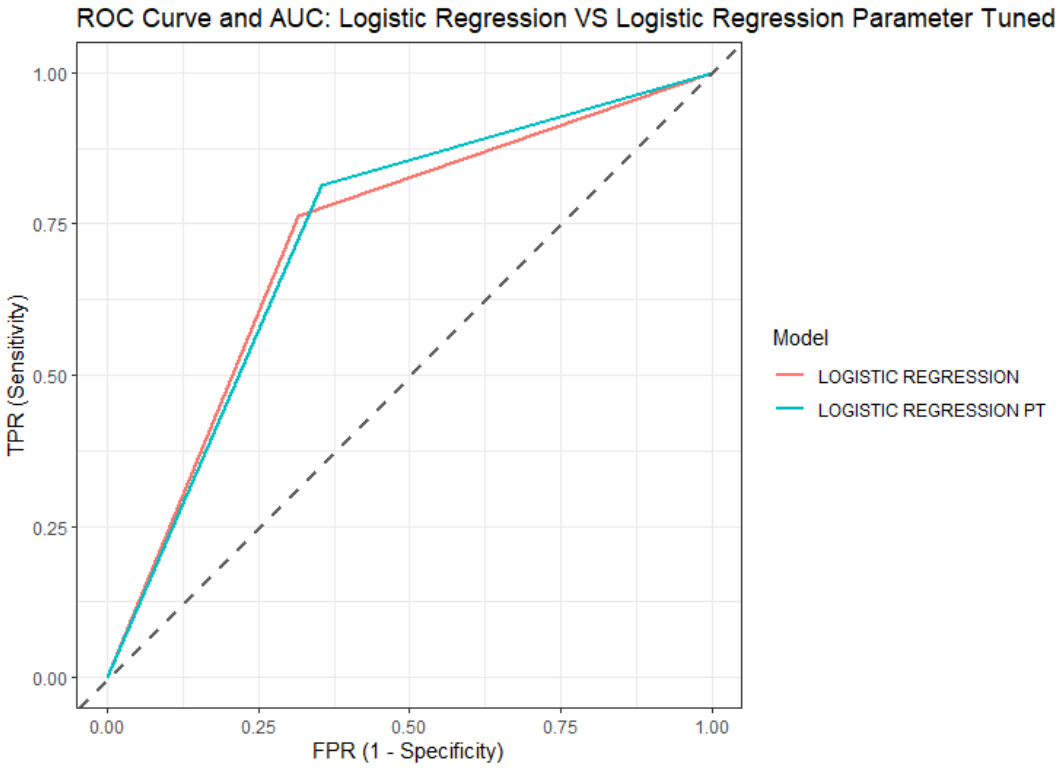

Figure 4. 9: LR vs LR PT ROC Curve Comparison

4.5. Model Metrics Comparison before and after Parameter Tuning

4.5.1. Model Metrics Comparison before Parameter Tuning

	SVM	Random Forest	Logistic Regression
Accuracy	73.89%	73.63%	72.32%
Kappa	47.78%	47.26%	44.65%
Sensitivity	67.89%	65.27%	68.41%
Specificity	79.90%	81.98%	76.24%
Pos Pred Value	77.15%	78.37%	74.22%
Neg Pred Value	71.33%	70.25%	70.70%
Prevalence	50.00%	50.00%	50.00%
Detection Rate	33.94%	32.64%	34.20%
Detection Prevalence	43.99%	41.64%	46.08%
Balanced Accuracy	73.89%	73.63%	72.32%

MODELs	Class	PRECISION	RECALL	F1 SCORE
SVM	0	6.79%	77.15%	72.22%
	1	79.90%	71.33%	75.37%
RF	0	65.27%	78.37%	71.23%
	1	81.98%	70.25%	75.66%
LOG	0	68.41%	74.22%	71.20%
	1	76.24%	70.70%	73.37%

Table 4. 7: SVM vs RF vs LR Metrics Comparison

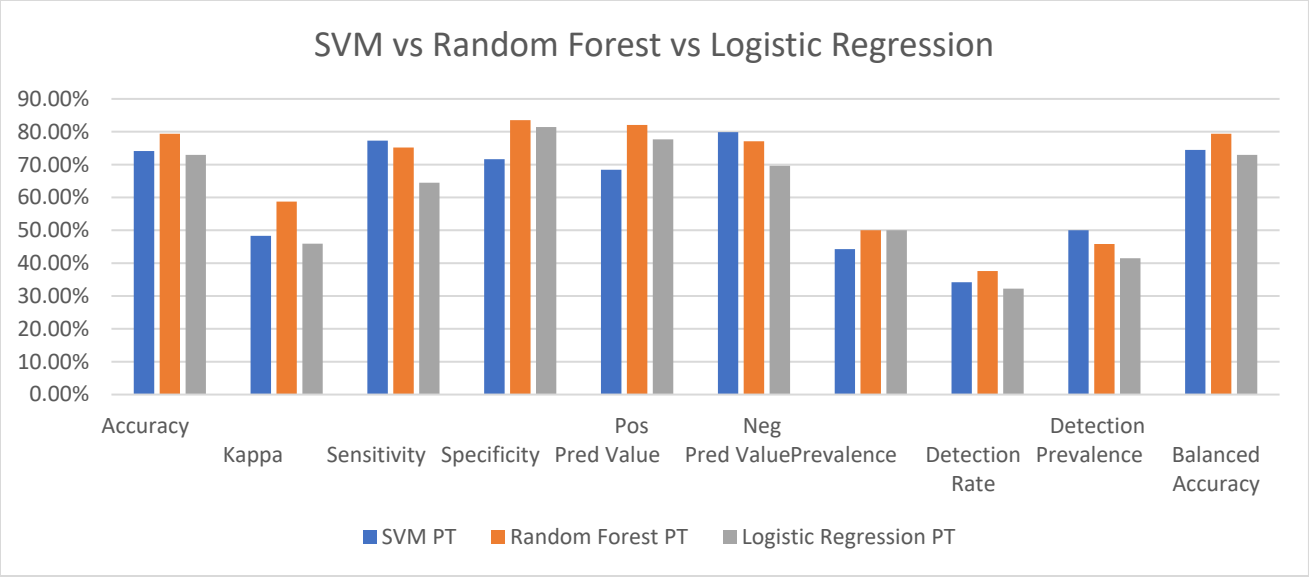

Figure 4. 10: SVM vs RF vs LR Metrics Comparison

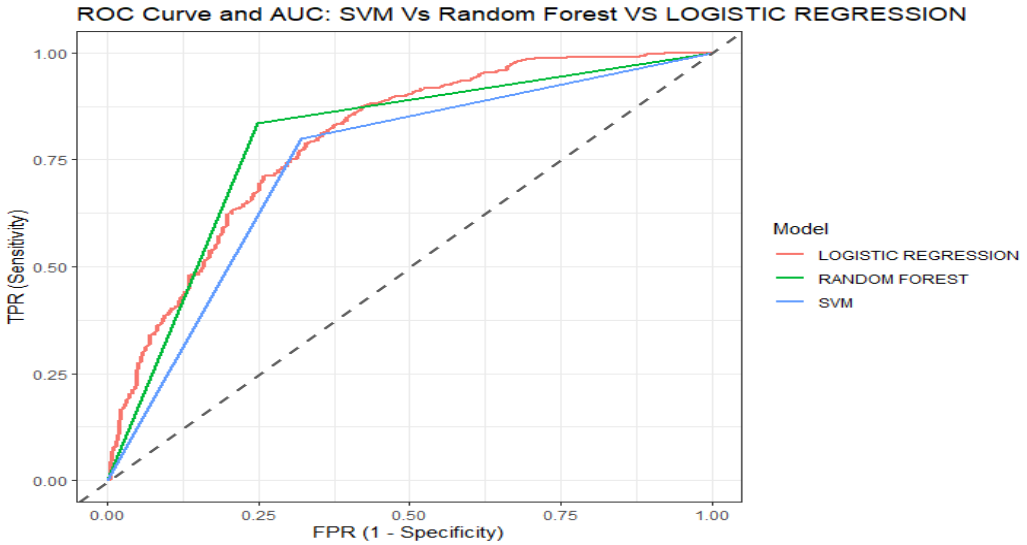

Figure 4. 11: SVM vs RF vs LR ROC Curve Comparison

4.5.2. Model Metrics Comparison after Parameter Tuning

	SVM PT	Random Forest PT	Logistic Regression PT
Accuracy	74.15%	79.37%	72.98%
Kappa	48.30%	58.75%	45.95%
Sensitivity	77.29%	75.20%	64.49%
Specificity	71.66%	83.55%	81.46%
Pos Pred Value	68.41%	82.05%	77.67%

Neg Pred Value	79.90%	77.11%	69.64%
Prevalence	44.26%	50.00%	50.00%
Detection Rate	34.20%	37.60%	32.25%
Detection Prevalence	50.00%	45.82%	41.51%
Balanced Accuracy	74.47%	79.37%	72.98%

MODELs	Class	PRECISION	RECALL	F1 SCORE
SVM PT	0	77.29%	68.41%	72.58%
	1	71.66%	79.90%	75.56%
RF PT	0	75.20%	82.05%	78.47%
	1	83.55%	77.11%	80.20%
LOG PT	0	63.71%	77.96%	70.11%
	1	81.98%	69.32%	75.12%

Table 4. 8: SVM PT vs RF PT vs LR PT Metrics Comparison

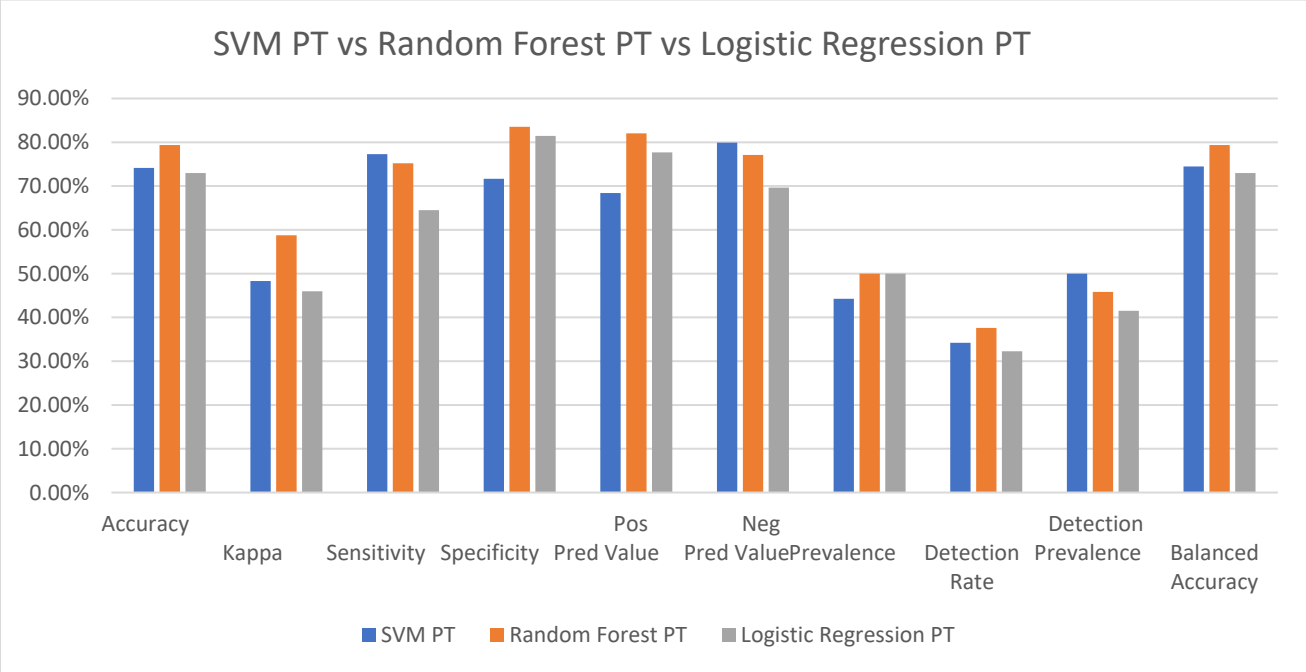

Figure 4. 12: SVM PT vs RF PT vs LR PT Metrics Comparison

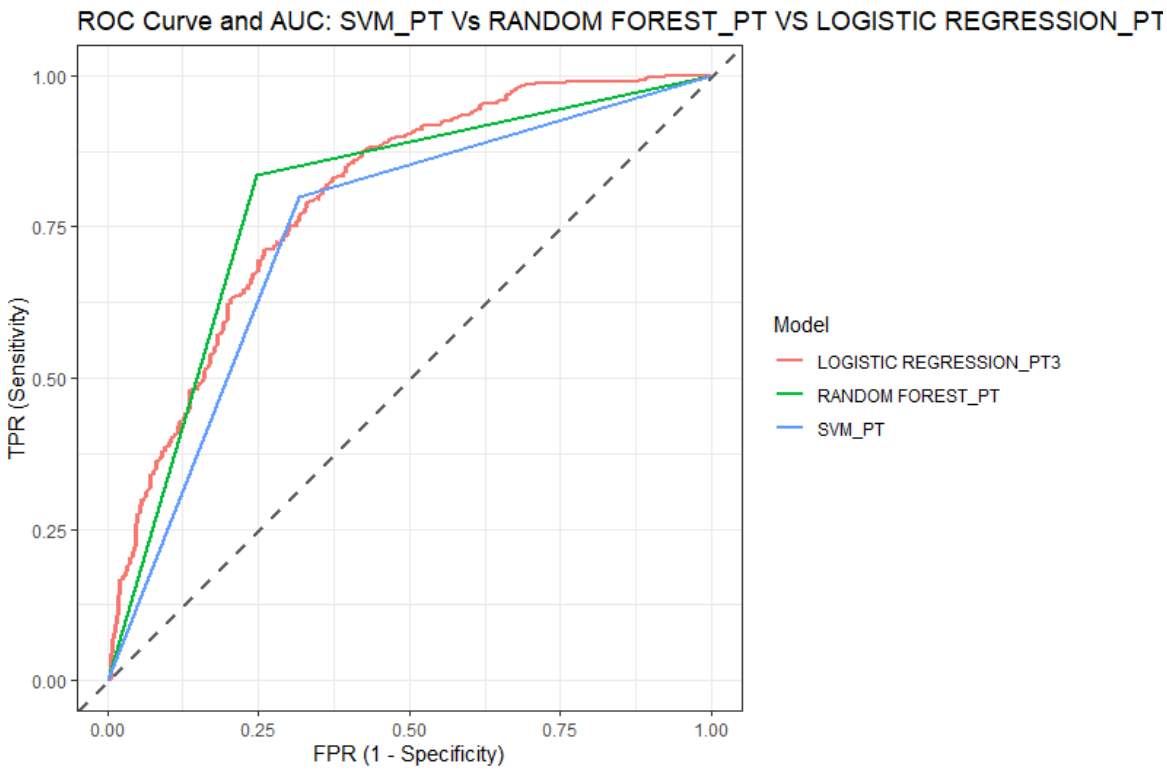

Figure 4. 13: SVM PT vs RF PT vs LR PT ROC Curve Comparison

5. Conclusion

The study's specific objective is to determine the quality of wine using a variety of physicochemical variables. After applying percentage split mode, the results are expressed as a percentage of correctly classified instances, precision, recall, F measure, and ROC. On these datasets, various classifiers such as Support Vector Machine (SVM), Random Forest, and logistic regression are evaluated. The results of the experiments indicate that the Support Vector Machine (SVM) Algorithm slightly outperforms Random Forests the and logistic regression algorithms in classification tasks. The support vector machine has an accuracy of 73.89% while Random Forest has 73.63% and Logistic regression 72.32%.

After Parameter tuning, the Model accuracy of Random Forest increased to 79.37%, while Support Vector Machine (SVM) also increased to 74.15%. There was also a slight increment for the Logistic Regression Technique as it increased to 72.98%.

References

[1] I. Janszky, Ahnve, J. Magnusson, H. Alinagizadeh, M. Blom and M. Ericson, "Wine drinking is associated with increased heart rate variability in women with coronary heart disease," *Heart, 91(3),* pp. 314-318, 2005.

[2] S. Aich, A. A. Al-Absi, K. Lee Hui and M. Sain, "Prediction of Quality for Different Type of Wine based on Different Feature Sets Using Supervised Machine Learning Techniques," *International Conference on Advanced Communication Technology, ICACT,* Vols. 2019-February, pp. 1122-1127, 4 2019.

[3] K. R. Dahal, J. N. Dahal, H. Banjade and S. Gaire, "Prediction of Wine Quality Using Machine Learning Algorithms," *Open Journal of Statistics,* vol. 11, pp. 278-289, 2021.

[4] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems,* vol. 47, no. 4, pp. 547-553, 11 2009.

[5] S. Ebeler, in *Flavor Chemistry — Thirty Years of Progres*, Kluwer Academic Publishers, 1999, p. 409–422.

[6] A. Rudnitskaya, L. Luvova, A. Legin, Y. Vlasov, C. Natale, and A. D'Amico, "Evaluation of Italian wine by the electronic tongue: recognition, quantitative analysis and correlation with human sensory perception," *Analytica Chimica Acta 484,* pp. 33-34, 2003.

[7] E. Turban, R. Sharda, J. Aronson and D. King, Business Intelligence, A Managerial Approach, Prentice-Hall, 2007.

[8] B. Ag, Wine Quality Prediction Using Different Machine Learning Techniques, vol. 8, 2020, p. 4.

[9] Y. Subba Reddy and P. P. Govindarajulu, "An Efficient User Centric Clustering Approach for Product Recommendation Based on Majority Voting: A Case Study on Wine Data Set," *IJCSNS International Journal of Computer Science and Network Security,* vol. 17, no. 10, 2017.

[10] L. R. Caroline , W. Philippa, S. Jiyeon, R. ,Jeroen, B. Jordana and S. Tim, "Red Wine Consumption Associated With Increased Gut Microbiota α-Diversity in 3 Independent Cohorts," *Gastroenterology,* p. 158, 2019.

[11] Y. Er and A. Ayten , "The Classification of White Wine and Red Wine According to Their Physicochemical Qualities," *International Journal of Intelligent Systems and Applications in Engineering,* vol. 4, no. Special Issue-1, pp. 23-26, 12 2016.

[12] "Kaggle.com," [Online]. Available: https://www.kaggle.com/ishanoze15/winequality-eda/data .

[13] "Correlation Analysis - Market Research," [Online]. Available: https://www.djsresearch.co.uk/glossary/item/Correlation-Analysis-Market-Research.

[14] "Imbalanced Classification Problems in R," [Online]. Available: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/.

[15] "Feature scaling in R: five simple methods - Data Tricks," [Online]. Available: https://datatricks.co.uk/feature-scaling-in-r-five-simple-methods.

[16] "How to Split data into train and test in R | R-bloggers," [Online]. Available: https://www.r-bloggers.com/2021/12/how-to-split-data-into-train-and-test-in-r/.

[17] I. Isharma, "Quality Prediction of Red Wine based on Different Feature Sets Using Machine Learning Techniques," 2018. [Online]. Available: www.ijsr.net.

[18] P. Appalasamy, A. Mustapha, N. Rizal, F. Johari and A. Mansor, "Classification-based Data Mining Approach for Quality Control in Wine Production," *Journal of Applied Sciences,* vol. 12, no. 6, pp. 598-601, 3 2012.

[19] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," 11 2009. [Online].

Appendix

```
install.packages("tidyverse")
install.packages("caret")
install.packages("lubridate")
install.packages("readr")
install.packages("icesTAF")
install.packages("knitr")
install.packages("VIM")
install.packages("naniar")
install.packages("Hmisc")
install.packages("corrplot")
install.packages('caTools')
install.packages("ROCR")
install.packages("PROC")


library("tidyverse")
library("caret")
library("lubridate")
library("readr")
library("icesTAF")
library("knitr")
library("ggplot2")
library("VIM")
library("naniar")
library("Hmisc")
library("corrplot")
library("caTools")
library(ROCR)
library(pROC)

.
# Import the datasets.
# 'wine' is the red and white wine dataset

wine <- read.csv("C:/Users/computer/Desktop/Teesside University/CIS4047-N-
BF1-2021 Data Science Foundations/Accessment/20%/winequalityN.csv", sep =
',', header = TRUE)
head(wine, n = 10)
tail(wine, n = 3)

colnames(wine)         # View the headers
summary(wine)          # View Summary

str(wine)              #Check Datatypes
sum(is.na(wine))       #check for NULL values
is.na(wine)

# Missing Values Visualization
gg_miss_var(wine)
res<-summary(aggr(wine, sortVar=TRUE))$combinations

# Replace missing values with column mean
# put in a new dataframe
wineq <- wine
```

```r
wineq$fixed.acidity[is.na(wineq$fixed.acidity)]<-
mean(wineq$fixed.acidity,na.rm=TRUE)
wineq$pH[is.na(wineq$pH)]<-mean(wineq$pH,na.rm=TRUE)
wineq$volatile.acidity[is.na(wineq$volatile.acidity)]<-
mean(wineq$volatile.acidity,na.rm=TRUE)
wineq$sulphates[is.na(wineq$sulphates)]<-mean(wineq$sulphates,na.rm=TRUE)
wineq$citric.acid[is.na(wineq$citric.acid)]<-
mean(wineq$citric.acid,na.rm=TRUE)
wineq$residual.sugar[is.na(wineq$residual.sugar)]<-
mean(wineq$residual.sugar,na.rm=TRUE)
wineq$chlorides[is.na(wineq$chlorides)]<-mean(wineq$chlorides,na.rm=TRUE)

#check for missing values again
sum(is.na(wineq))
summary(wineq)


hist.data.frame(wineq)

#Scatterplot Matrix of Variables
plot(wineq)

# Use only numerical values
winequ <- wineq[, -1]

head(wineq)

#Correlation Heatmap of Variables
corrplot(cor(winequ))

# Correlations
corrplot(cor(winequ), method = "number",
         title = "Correlation Plot",
         tl.pos = "n", mar = c(2, 1, 3, 1))

#Distribution of wine quality ratings
ggplot(wineq,aes(x=quality))+geom_bar(stat = "count",position = "dodge")+
  scale_x_continuous(breaks = seq(3,8,1))+
  ggtitle("Distribution of Wine Quality Ratings")+
  theme_classic()

#Distribution of white/red wines
ggplot(wineq,aes(x=type,fill=factor(type)))+geom_bar(stat = "count",position
= "dodge")+ #scale_y_continuous(labels=scales::percent, limits=(10, 100))+
  #scale_x_continuous(breaks = seq(0,1,1),limits=(10, 1000))+
  ggtitle("Distribution of White/Red Wines")+
  theme_classic()

#Distribution of quality for white/red wines
ggplot(wineq, aes(quality,fill=type, type=c("white"))) +
  geom_histogram(binwidth = .5,col="black") +
  facet_grid(type ~ .)+
  labs(title="Histogram Showing Qulity of Wine",
       subtitle="Wine Quality across Red and White colors of Wine")

str(wineq)
```

```r
#Will give us column number which might insignificant variance
nzv <- nearZeroVar(wineq)
print(paste("---Column number with----", nzv))

# Encoding the type column
wineq$type = factor(wineq$type,
                          levels = c('white','red'),
                          labels = c(1.0, 2.0))
str(wineq)

# Splitting the dataset 70/30
set.seed(123)
split = sample.split(wineq$quality, SplitRatio = 0.7)

#Creating the training set and test set separately
#training_set = subset(wineq, split == TRUE)
#test_set = subset(wineq, split == FALSE)
#training_set
#test_set

# Create Normalize function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Normalize
wineq.norm<- as.data.frame(lapply(wineq[,2:12], normalize))




#Rename
winee<-wineq.norm
head(winee)

plotViewport(winee$quality)

#If quality is greater than 6 then wine is good else wine is Bad
winee$quality <- ifelse (as.integer(wineq$quality) > 6, 1, 0)

#Distribution of quality revamped

ggplot(winee,aes(x=quality))+geom_bar(stat = "count",position = "dodge")+
  scale_x_continuous(breaks = seq(3,8,1))+
  ggtitle("Distribution of Wine Quality Revamped")+
  theme_classic()

#Downsampling
set.seed(47)
winef <- winee
winef$quality<-as.factor(winef$quality)
winef<- downSample(x = winef[,-12], y = winef$quality, list = F, yname =
"quality")

#Distribution of quality revamped after downsampling
```

```r
ggplot(winef,aes(x=quality))+geom_bar(stat = "count",position = "dodge")+
  #scale_x_continuous(breaks = seq(3,8,1))+
  ggtitle("Distribution of Wine Quality Downsampling")+
  theme_classic()

#winee$type<-wineq$type
#winee$quality<-wineq$quality



summary(winef)
head(winee)

#Applying the SVM Machine Learning Technique

intrain <- createDataPartition(y = winef$quality, p= 0.7, list = FALSE)
training <- winef[intrain,]
testing <- winef[-intrain,]
 dim(testing)

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
install.packages("e1071")
library("e1071")

svm_linear = svm(formula = quality ~ .,
                 data = training,
                 type = 'C-classification',
                 kernel = 'linear')
#Predicting SVM

pred_svm_linear=predict(svm_linear,testing)

#Confusion Matric for SVM
confusionMatrix(pred_svm_linear,testing$quality)
cm_SVM <- table(pred_svm_linear,testing$quality)

#Calculating other Metrices
n_svm = sum(cm_SVM) # number of instances
nc_svm = nrow(cm_SVM) # number of classes
diag_svm = diag(cm_SVM) # number of correctly classified instances per class
rowsums_svm = apply(cm_SVM, 1, sum) # number of instances per class
colsums_svm = apply(cm_SVM, 2, sum) # number of predictions per class
p_svm = rowsums_svm / n_svm # distribution of instances over the actual
classes
q_svm = colsums_svm / n_svm # distribution of instances over the predicted
classes


diag_svm
rowsums_svm
colsums_svm


accuracy_svm = sum(diag_svm) / n_svm

accuracy_svm
```

```r
precision_svm = diag_svm / colsums_svm
recall_svm = diag_svm / rowsums_svm
f1_svm = 2 * precision_svm * recall_svm / (precision_svm + recall_svm)
data.frame(precision_svm, recall_svm, f1_svm)


# Plotting ROC Curve

ROCRpredsvm = prediction(as.numeric(pred_svm_linear), testing$quality)
ROCRperfsvm = performance(ROCRpredsvm, "tpr", "fpr")
aucsvm <- slot(performance(ROCRpredsvm, "auc"), "y.values")[[1]] # Area Under
Curve
plot(ROCRperfsvm, colorize=TRUE)
abline(h=seq(0,1,0.05), v=seq(0,1,0.05), col = "lightgray", lty = "dotted")
lines(c(0,1),c(0,1), col = "gray", lwd =2)
text(0.6,0.2,paste("AUC=", round(aucsvm,4), sep=""), cex=1.4)
title("ROC SVM")

#Parameter Tuning
svm_pt <- svm(quality ~., data = training, kernel  = "radial",
              cost = 1, gamma = 0.04545455, epsilon = 0.1)
pred_svm_pt=predict(svm_pt, testing)

confusionMatrix(testing$quality, pred_svm_pt)
cm_SVM_pt <- table(testing$quality, pred_svm_pt)

#Calculating other Metrices for SVM Parameter Tuning
n_SVM_pt = sum(cm_SVM_pt) # number of instances
nc_SVM_pt = nrow(cm_SVM_pt) # number of classes
diag_SVM_pt = diag(cm_SVM_pt) # number of correctly classified instances per
class
rowsums_SVM_pt = apply(cm_SVM_pt, 1, sum) # number of instances per class
colsums_SVM_pt = apply(cm_SVM_pt, 2, sum) # number of predictions per class
p_SVM_pt = rowsums_SVM_pt / n_SVM_pt # distribution of instances over the
actual classes
q_SVM_pt = colsums_SVM_pt / n_SVM_pt # distribution of instances over the
predicted classes


diag_SVM_pt
rowsums_SVM_pt
colsums_SVM_pt


accuracy_SVM_pt = sum(diag_SVM_pt) / n_SVM_pt

accuracy_SVM_pt

precision_SVM_pt = diag_SVM_pt / colsums_SVM_pt
recall_SVM_pt = diag_SVM_pt / rowsums_SVM_pt
f1_SVM_pt = 2 * precision_SVM_pt * recall_SVM_pt / (precision_SVM_pt +
recall_SVM_pt)
data.frame(precision_SVM_pt, recall_SVM_pt, f1_SVM_pt)

#Comparing ROC Curve Before and adter Parameter tuning

test_auc <- function(prob) {
```

```
  roc(testing$quality, prob)
}

auc_svm <- test_auc(as.numeric(pred_svm_linear))

auc_svm_pt <- test_auc(as.numeric(pred_svm_pt))


df_auc_svm <- bind_rows(data_frame(TPR = auc_svm$sensitivities,
                                   FPR = 1 - auc_svm$specificities,
                                   Model = "SVM"),
                        data_frame(TPR = auc_svm_pt$sensitivities,
                                   FPR = 1 - auc_svm_pt$specificities,
                                   Model = "SVM_PT"))

df_auc_svm %>%
  ggplot(aes(FPR, TPR, color = Model)) +
  geom_line(size = 1) +
  theme_bw() +
  coord_equal() +
  geom_abline(intercept = 0, slope = 1, color = "gray37", size = 1, linetype
= "dashed") +
  labs(x = "FPR (1 - Specificity)",
       y = "TPR (Sensitivity)",
       title = "ROC Curve and AUC: SVM VS SVM Parameter Tuned")




#Random Forest

#Building the decision tree
install.packages("rpart")
install.packages("rpart.plot")
install.packages("randomForest")
install.packages("rattle")
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)
library(rattle)

training$quality<-as.factor(training$quality)
testing$quality<-as.factor(testing$quality)

summary(training)

#Applying Random forest Model
rf <-rpart(training$quality~., data=training)

#Making predictions
pred_rf <- predict(rf, testing, type="class")

#Random Forest Confusion Matrix
confusionMatrix(as.factor(pred_rf),testing$quality)
cm_rf <- table(as.factor(pred_rf),testing$quality)
```

```r
#ROC for Random Forest
par(mfrow=c(1,1))
ROCRpredrf = prediction(as.numeric(pred_rf), testing$quality)
ROCRperfrf = performance(ROCRpredrf, "tpr", "fpr")
aucrf <- slot(performance(ROCRpredrf, "auc"), "y.values")[[1]] # Area Under
Curve
plot(ROCRperfrf, colorize=TRUE)
abline(h=seq(0,1,0.05), v=seq(0,1,0.05), col = "lightgray", lty = "dotted")
lines(c(0,1),c(0,1), col = "gray", lwd =2)
text(0.6,0.2,paste("AUC=", round(aucrf,4), sep=""), cex=1.4)
title("ROC Random Forest")

#Calculating other Metrices

n_rf = sum(cm_rf) # number of instances
nc_rf = nrow(cm_rf) # number of classes
diag_rf = diag(cm_rf) # number of correctly classified instances per class
rowsums_rf = apply(cm_rf, 1, sum) # number of instances per class
colsums_rf = apply(cm_rf, 2, sum) # number of predictions per class
p_rf = rowsums_rf / n_rf # distribution of instances over the actual classes
q_rf = colsums_rf / n_rf # distribution of instances over the predicted
classes


diag_rf
rowsums_rf
colsums_rf


accuracy_rf = sum(diag_rf) / n_rf

accuracy_rf

precision_rf= diag_rf / colsums_rf
recall_rf = diag_rf / rowsums_rf
f1_rf = 2 * precision_rf * recall_rf / (precision_rf + recall_rf)
data.frame(precision_rf, recall_rf, f1_rf)



# Random Forest parameter tuning
rf_pt <- randomForest(quality ~., data = training, ntree  = 10000,
                 mtry= 10, importance = TRUE, PROXIMITY=TRUE)


# Predict RF PT
predrf_pt=predict(rf_pt,testing)

#Confusion Matrix for RF PT
confusionMatrix(predrf_pt,testing$quality)

cm_rf_pt <- table(predrf_pt,testing$quality)

#Calculating other Metrices for Random Forest Parameter Tuning
n_rf_pt = sum(cm_rf_pt) # number of instances
nc_rf_pt = nrow(cm_rf_pt) # number of classes
```

```
diag_rf_pt = diag(cm_rf_pt) # number of correctly classified instances per
class
rowsums_rf_pt = apply(cm_rf_pt, 1, sum) # number of instances per class
colsums_rf_pt = apply(cm_rf_pt, 2, sum) # number of predictions per class
p_rf_pt = rowsums_rf_pt / n_rf_pt # distribution of instances over the actual
classes
q_rf_pt = colsums_rf_pt / n_rf_pt # distribution of instances over the
predicted classes


diag_rf_pt
rowsums_rf_pt
colsums_rf_pt


accuracy_rf_pt = sum(diag_rf_pt) / n_rf_pt

accuracy_rf_pt

precision_rf_pt = diag_rf_pt / colsums_rf_pt
recall_rf_pt = diag_rf_pt / rowsums_rf_pt
f1_rf_pt = 2 * precision_rf_pt * recall_rf_pt / (precision_rf_pt +
recall_rf_pt)
data.frame(precision_rf_pt, recall_rf_pt, f1_rf_pt)

#Comparing ROC Curve Before and after Parameter tuning for Random Forest

test_auc <- function(prob) {
  roc(testing$quality, prob)
}

auc_rf <- test_auc(as.numeric(pred_rf))

auc_rf_pt <- test_auc(as.numeric(predrf_pt))


df_auc_rf <- bind_rows(data_frame(TPR = auc_rf$sensitivities,
                                  FPR = 1 - auc_rf$specificities,
                                  Model = "RANDOM FOREST"),
                       data_frame(TPR = auc_rf_pt$sensitivities,
                                  FPR = 1 - auc_rf_pt$specificities,
                                  Model = "RANDOM FOREST PT"))

df_auc_rf%>%
  ggplot(aes(FPR, TPR, color = Model)) +
  geom_line(size = 1) +
  theme_bw() +
  coord_equal() +
  geom_abline(intercept = 0, slope = 1, color = "gray37", size = 1, linetype
= "dashed") +
  labs(x = "FPR (1 - Specificity)",
       y = "TPR (Sensitivity)",
       title = "ROC Curve and AUC: Random Forest VS Random Forest Parameter
Tuned")
```

```r
#Logistic Regression model

log <- glm(formula = quality ~., data=training, family = "binomial")

summary(log)

#Predict Model for Logistic Regression

pred_log<- predict(log, testing, type = "response")

predicted_log<-ifelse(pred_log> 0.5,1,0)

#COnfusion Matrix for Logistic Regression

confusionMatrix(as.factor(predicted_log), testing$quality)

cm_log <- table(data = as.factor(predicted_log), reference = testing$quality)

cm_log

#Calculating other Metrices for Logistic Regression
n_log = sum(cm_log) # number of instances
nc_log = nrow(cm_log) # number of classes
diag_log = diag(cm_log) # number of correctly classified instances per class
rowsums_log = apply(cm_log, 1, sum) # number of instances per class
colsums_log = apply(cm_log, 2, sum) # number of predictions per class
p_log = rowsums_log / n_log # distribution of instances over the actual
classes
q_log = colsums_log / n_log # distribution of instances over the predicted
classes


diag_log
rowsums_log
colsums_log


accuracy_log = sum(diag_log) / n_log

accuracy_log

precision_log = diag_log / colsums_log
recall_log = diag_log / rowsums_log
f1_log = 2 * precision_log * recall_log / (precision_log + recall_log)
data.frame(precision_log, recall_log, f1_log)



#ROC Curve for Logistic Regression

par(mfrow=c(1,1))
ROCRpred = prediction(predicted_log, testing$quality)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
auc <- slot(performance(ROCRpred, "auc"), "y.values")[[1]] # Area Under Curve
```

```
plot(ROCRperf, colorize=TRUE)
abline(h=seq(0,1,0.05), v=seq(0,1,0.05), col = "lightgray", lty = "dotted")
lines(c(0,1),c(0,1), col = "gray", lwd =2)
text(0.6,0.2,paste("AUC=", round(auc,4), sep=""), cex=1.4)
title("ROC Logistic Regression")

# Parameter Tuning for Logistic Regression
#1 Try out the train function to see if 'parameter' gets tuned
set.seed(1)
log_pt <- train(quality ~., data=training, method='glm')
log_pt
predlog_pt<- predict(log_pt, testing, type = "raw")
confusionMatrix(data = as.factor(predlog_pt), reference = testing$quality)


#2 try another parameter tuning logistic regression

set.seed(123)
log_pt2 <- train(
  quality ~.,
  data = training,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
predlog_pt2<- predict(log_pt2, testing)
confusionMatrix(data = as.factor(predlog_pt2), reference = testing$quality)

#3 try another parameter tuning logistic regression

log_pt3 <- glm(formula = quality ~., data=training, family = "binomial")

summary(log_pt3)

#Predict Model for Logistic Regression Parameter Tuning 3

pred_log_pt3<- predict(log_pt3, testing, type = "response")

predicted_log_pt3<-ifelse(pred_log_pt3> 0.455,1,0)

#COnfusion Matrix for Logistic Regression parameter Tuning 3

confusionMatrix(as.factor(predicted_log_pt3), testing$quality)

cm_log_pt3 <- table(data = as.factor(predicted_log_pt3), reference =
testing$quality)

cm_log_pt3

#Calculating other Metrices for Logistic Regression Parameter Tuning 3
n_log_pt3 = sum(cm_log_pt3) # number of instances
nc_log_pt3 = nrow(cm_log_pt3) # number of classes
diag_log_pt3 = diag(cm_log_pt3) # number of correctly classified instances
per class
rowsums_log_pt3 = apply(cm_log_pt3, 1, sum) # number of instances per class
colsums_log_pt3 = apply(cm_log_pt3, 2, sum) # number of predictions per class
```

```r
p_log_pt3 = rowsums_log_pt3 / n_log_pt3 # distribution of instances over the
actual classes
q_log_pt3 = colsums_log_pt3 / n_log_pt3 # distribution of instances over the
predicted classes


diag_log_pt3
rowsums_log_pt3
colsums_log_pt3


accuracy_log_pt3 = sum(diag_log_pt3) / n_log_pt3

accuracy_log_pt3

precision_log_pt3 = diag_log_pt3 / colsums_log_pt3
recall_log_pt3 = diag_log_pt3 / rowsums_log_pt3
f1_log_pt3 = 2 * precision_log_pt3 * recall_log_pt3 / (precision_log_pt3 +
recall_log_pt3)
data.frame(precision_log_pt3, recall_log_pt3, f1_log_pt3)

#Comparing ROC Curve Before and after Parameter tuning for Logistic
Regression

test_auc <- function(prob) {
  roc(testing$quality, prob)
}

auc_log <- test_auc(as.numeric(predicted_log))

auc_log_pt <- test_auc(as.numeric(predicted_log_pt3))


df_auc_log <- bind_rows(data_frame(TPR = auc_log$sensitivities,
                                   FPR = 1 - auc_log$specificities,
                                   Model = "LOGISTIC REGRESSION"),
                        data_frame(TPR = auc_log_pt$sensitivities,
                                   FPR = 1 - auc_log_pt$specificities,
                                   Model = "LOGISTIC REGRESSION PT"))

df_auc_log%>%
  ggplot(aes(FPR, TPR, color = Model)) +
  geom_line(size = 1) +
  theme_bw() +
  coord_equal() +
  geom_abline(intercept = 0, slope = 1, color = "gray37", size = 1, linetype
= "dashed") +
  labs(x = "FPR (1 - Specificity)",
       y = "TPR (Sensitivity)",
       title = "ROC Curve and AUC: Logistic Regression VS Logistic Regression
Parameter Tuned")


#Comparing SVM, Random Forest and Logistic regression after Parameter Tuning.

test_auc <- function(prob) {
  roc(testing$quality, prob)
```

```
}

auc_svm <- test_auc(as.numeric(pred_svm_linear))

auc_rf_pt <- test_auc(as.numeric(pred_rf))

auc_log <- test_auc(as.numeric(pred_log))


df_auc <- bind_rows(data_frame(TPR = auc_svm$sensitivities,
                               FPR = 1 - auc_svm$specificities,
                               Model = "SVM"),
                    data_frame(TPR = auc_log_pt$sensitivities,
                               FPR = 1 - auc_log_pt$specificities,
                               Model = "RANDOM FOREST"),
                    data_frame(TPR = auc_log$sensitivities,
                               FPR = 1 - auc_log$specificities,
                               Model = "LOGISTIC REGRESSION"))

df_auc%>%
  ggplot(aes(FPR, TPR, color = Model)) +
  geom_line(size = 1) +
  theme_bw() +
  coord_equal() +
  geom_abline(intercept = 0, slope = 1, color = "gray37", size = 1, linetype
= "dashed") +
  labs(x = "FPR (1 - Specificity)",
       y = "TPR (Sensitivity)",
       title = "ROC Curve and AUC: SVM Vs Random Forest VS LOGISTIC
REGRESSION")



#Comparing SVM, Random Forest and Logistic regression before Parameter
Tuning.

test_auc <- function(prob) {
  roc(testing$quality, prob)
}

auc_svm_pt <- test_auc(as.numeric(pred_svm_pt))

auc_rf_pt<- test_auc(as.numeric(predrf_pt))

auc_log_pt <- test_auc(as.numeric(pred_log_pt3))


df_auc_pt <- bind_rows(data_frame(TPR = auc_svm_pt$sensitivities,
                                  FPR = 1 - auc_svm_pt$specificities,
                                  Model = "SVM_PT"),
                       data_frame(TPR = auc_rf_pt$sensitivities,
                                  FPR = 1 - auc_rf_pt$specificities,
                                  Model = "RANDOM FOREST_PT"),
                       data_frame(TPR = auc_log_pt$sensitivities,
                                  FPR = 1 - auc_log_pt$specificities,
                                  Model = "LOGISTIC REGRESSION_PT3"))
```

```r
df_auc_pt%>%
  ggplot(aes(FPR, TPR, color = Model)) +
  geom_line(size = 1) +
  theme_bw() +
  coord_equal() +
  geom_abline(intercept = 0, slope = 1, color = "gray37", size = 1, linetype
= "dashed") +
  labs(x = "FPR (1 - Specificity)",
       y = "TPR (Sensitivity)",
       title = "ROC Curve and AUC: SVM_PT Vs RANDOM FOREST_PT VS LOGISTIC
REGRESSION_PT")
```