White Wine Quality Prediction using Statistical Learning

Presented by: Muhammad Salman Sabir

Supervisor: Prof. Salvatore Ingrassia

Data Set: White Wine Quality

Table of Contents

ABSTRACT

This project analyzes the "Wine Quality - White" dataset to predict wine quality (good/bad) using machine learning methods (Logistic Regression, Decision Tree, Random Forest). Data exploration, feature analysis, model training, and result interpretation are performed. The objective is to identify the chemical features that most strongly influence wine quality and select the best prediction model.

DATASET & VARIABLES

Source: UCI Wine Quality Dataset (White)

- **Total observations:** 3917 (training), 354 (test)
- **Target variable:** `quality` (converted to binary: Good = 1 if quality ≥ 6, Bad = 0 if < 6)
- **Variables:**
  - 11 physicochemical features (e.g. acidity, sugar, pH, alcohol, etc.)
  - 1 target variable (quality)
  - All columns are numeric
  - No Missing values

Attribute	Attribute Type	Description
fixed.acidity	Numeric	Tartaric acid concentration (g/dm^3)
volatile.acidity	Numeric	Acetic acid concentration (g/dm^3); can affect wine taste
citric.acid	Numeric	Citric acid concentration (g/dm^3); adds freshness and flavor
residual.sugar	Numeric	Amount of sugar remaining after fermentation (g/dm^3)
chlorides	Numeric	Salt concentration (g/dm^3)
free.sulfur.dioxide	Numeric	Free form SO_2 (mg/dm^3); prevents microbial growth
total.sulfur.dioxide	Numeric	Total SO_2 (mg/dm^3); free + bound forms
density	Numeric	Density of wine (g/cm^3)
pH	Numeric	Acidity level (lower = more acidic)
sulphates	Numeric	Potassium sulphate concentration (g/dm^3); antimicrobial, adds flavor
alcohol	Numeric	Alcohol content (% by volume)
quality	Integer	Wine quality score (0–10, as rated by experts)
quality_binary	Categorical	Target: Good quality wine (1 = quality ≥ 6, 0 = bad quality)

Exploratory Data Analysis:-

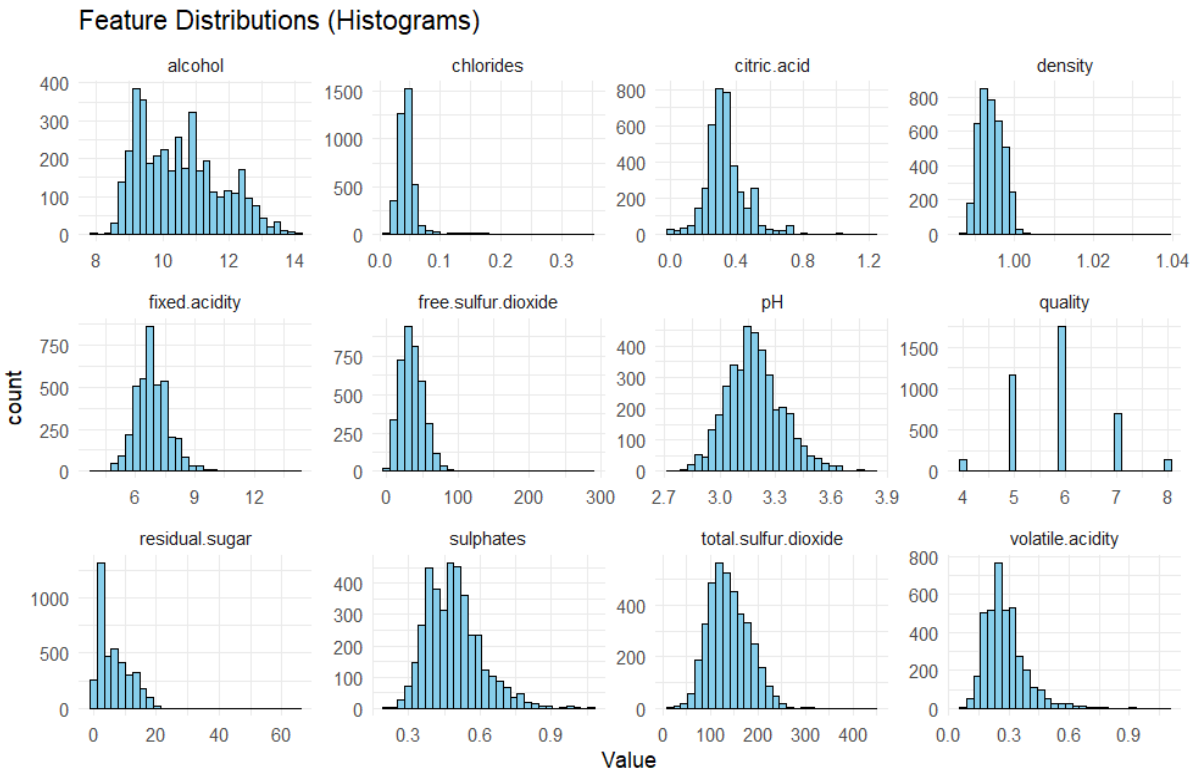
EDA is an approach to analyzing and understanding data through visual and statistical methods.

Variable	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum
fixed.acidity	3.90	6.30	6.80	6.86	7.30	14.20
volatile.acidity	0.08	0.21	0.26	0.28	0.32	1.10
citric.acid	0.00	0.27	0.32	0.33	0.39	1.23
residual.sugar	0.60	1.70	5.20	6.39	9.85	65.80
chlorides	0.009	0.036	0.043	0.046	0.050	0.346
free.sulfur.dioxide	2.00	23.00	34.00	35.44	46.00	289.00
total.sulfur.dioxide	9.00	109.00	134.00	138.79	168.00	440.00
density	0.987	0.9917	0.9937	0.9940	0.9961	1.039
pH	2.72	3.09	3.18	3.19	3.28	3.82
sulphates	0.22	0.41	0.47	0.49	0.55	1.08
alcohol	8.00	9.50	10.40	10.52	11.40	14.20

This table shows the **summary statistics** for each feature in the wine quality dataset. Here's a breakdown in simple terms:

- **Each row (Min, 1st Qu., Median, Mean, 3rd Qu., Max)** gives insights into the distribution of each variable.
- Most variables like **residual sugar**, **free sulfur dioxide**, and **total sulfur dioxide** have a **big gap between mean and max**, which suggests **outliers or skewed distributions**.
- **Alcohol** ranges from 8.0 to 14.2%, with a median of 10.4%, and is positively correlated with better wine quality.
- **Volatile acidity** and **density** are more tightly clustered, but still vary enough to influence quality.
- The **quality** score itself ranges from 4 to 8, with a median and mean around 6, showing that most wines are rated in the average-to-good range.

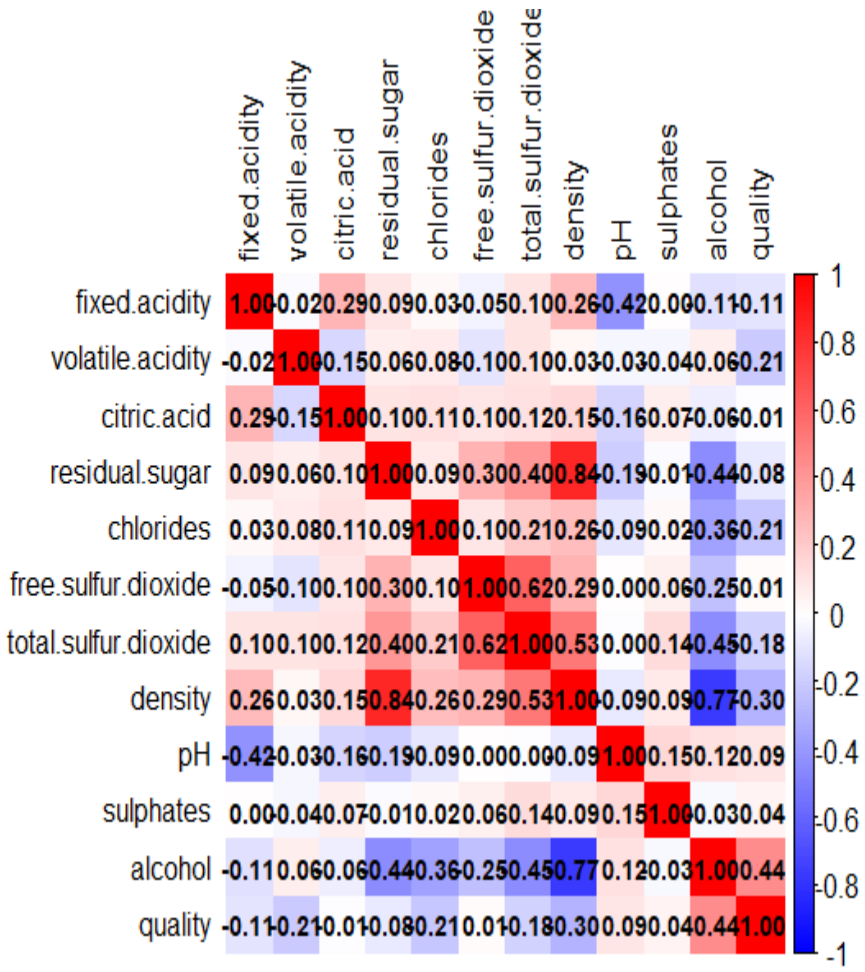
Histograms for Each Numeric Feature



Insights:-

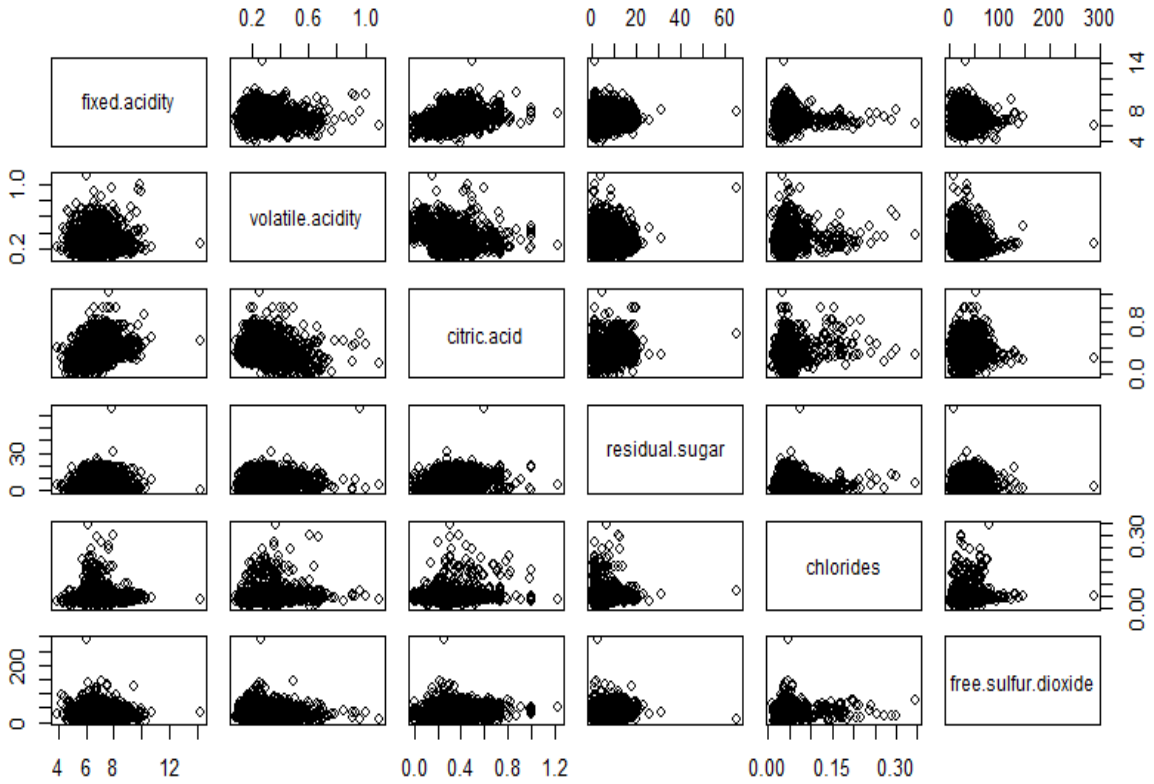
- **Most features are right-skewed**, especially residual.sugar, chlorides, and volatile.acidity, meaning most values are concentrated on the lower end with a few large outliers.
- **pH**, density, and total.sulfur.dioxide show more balanced, bell-shaped distributions.
- **quality is discrete** and clearly unbalanced, peaking at quality scores 5 and 6, which justifies the binary transformation (quality_binary) used in classification.

Correlation Heat map:-



This heat map shows how each feature in the dataset is related to the others. The closer a value is to 1 or -1, the stronger the relationship. You can see that alcohol stands out with a positive link to wine quality, meaning higher alcohol content usually gets a better rating. On the other hand, volatile acidity is negatively related so more acidity usually means lower quality. Most other features don't show a strong direct connection to quality, but you can spot some strong relationships between features themselves, like between density and residual sugar and total and free sulfur dioxide. This gives you a quick sense of which variables might matter most for predicting wine quality.

Scatter Plot:-



This is a **scatterplot matrix (pair plot)** showing pairwise relationships between several numerical features in the wine dataset:

- Each small plot shows the relationship between two variables. For example, the top-left corner shows fixed.acidity vs. volatile.acidity.
- Diagonal cells are labeled with the variable name.
- Most plots show **dense clustering** with some outliers, especially in variables like residual.sugar and free.sulfur.dioxide.
- You can visually spot **possible linear or non-linear trends**:
    - residual.sugar, chlorides, and free.sulfur.dioxide show **right-skewed distributions** and a few extreme values.
    - Some variables (like citric.acid vs volatile.acidity) show **mild negative correlation**.
- This matrix is helpful for detecting **multicollinearity, spread, and potential interactions** between features before modeling.

EDA Conclusion:-

- Most features are numeric and show slight skewness; **residual sugar** has significant outliers.
- **Alcohol** is positively correlated with wine quality higher alcohol often means better wine.
- **Volatile acidity** and **density** are negatively correlated with quality higher values tend to lower quality.
- The dataset has no major missing values and is ready for modeling.
- Correlation heat map and scatter plots helped identify key predictors of quality.
- These insights guided feature selection and model development in later stages.

Modeling:-

Logistic Regression:-

```
> # ----------------- Logistic Regression -----------------
> log_model <- glm(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], family = "binomial")
> log_probs <- predict(log_model, newdata = valSet[, features], type = "response")
> log_preds <- ifelse(log_probs > 0.5, 1, 0)
> log_acc <- mean(log_preds == valSet$quality_binary)
> cat("Logistic Regression Accuracy:", log_acc, "\n")
Logistic Regression Accuracy: 0.7701149
>
> # Confusion Matrix
> confusionMatrix(as.factor(log_preds), valSet$quality_binary)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 148  66
         1 114 455

               Accuracy : 0.7701
                 95% CI : (0.739, 0.7992)
    No Information Rate : 0.6654
    P-Value [Acc > NIR] : 9.562e-11

                  Kappa : 0.4591

 Mcnemar's Test P-Value : 0.0004598

            Sensitivity : 0.5649
            Specificity : 0.8733
         Pos Pred Value : 0.6916
         Neg Pred Value : 0.7996
             Prevalence : 0.3346
         Detection Rate : 0.1890
   Detection Prevalence : 0.2733
      Balanced Accuracy : 0.7191

       'Positive' Class : 0
```

Here's a simple explanation of your **logistic regression confusion matrix and statistics**:

- **Accuracy:** The model correctly predicts wine quality about 77% of the time.
- **Sensitivity (Recall for class 0):** 56% of bad wines are correctly identified.
- **Specificity:** 87% of good wines are correctly identified as good.
- **Precision for class 0:** When the model predicts a wine is bad, it's right about 69% of the time.
- The model is a bit better at identifying good wines than bad wines, as shown by the higher specificity and precision for the positive class (good).
- The confusion matrix (top) shows the counts of correct and incorrect predictions for each class.
- **Kappa** and **p-values** indicate the model performs significantly better than random guessing.

In short, this logistic regression model is more reliable for picking out good quality wines, but it sometimes misses bad ones.

Decision Tree:-

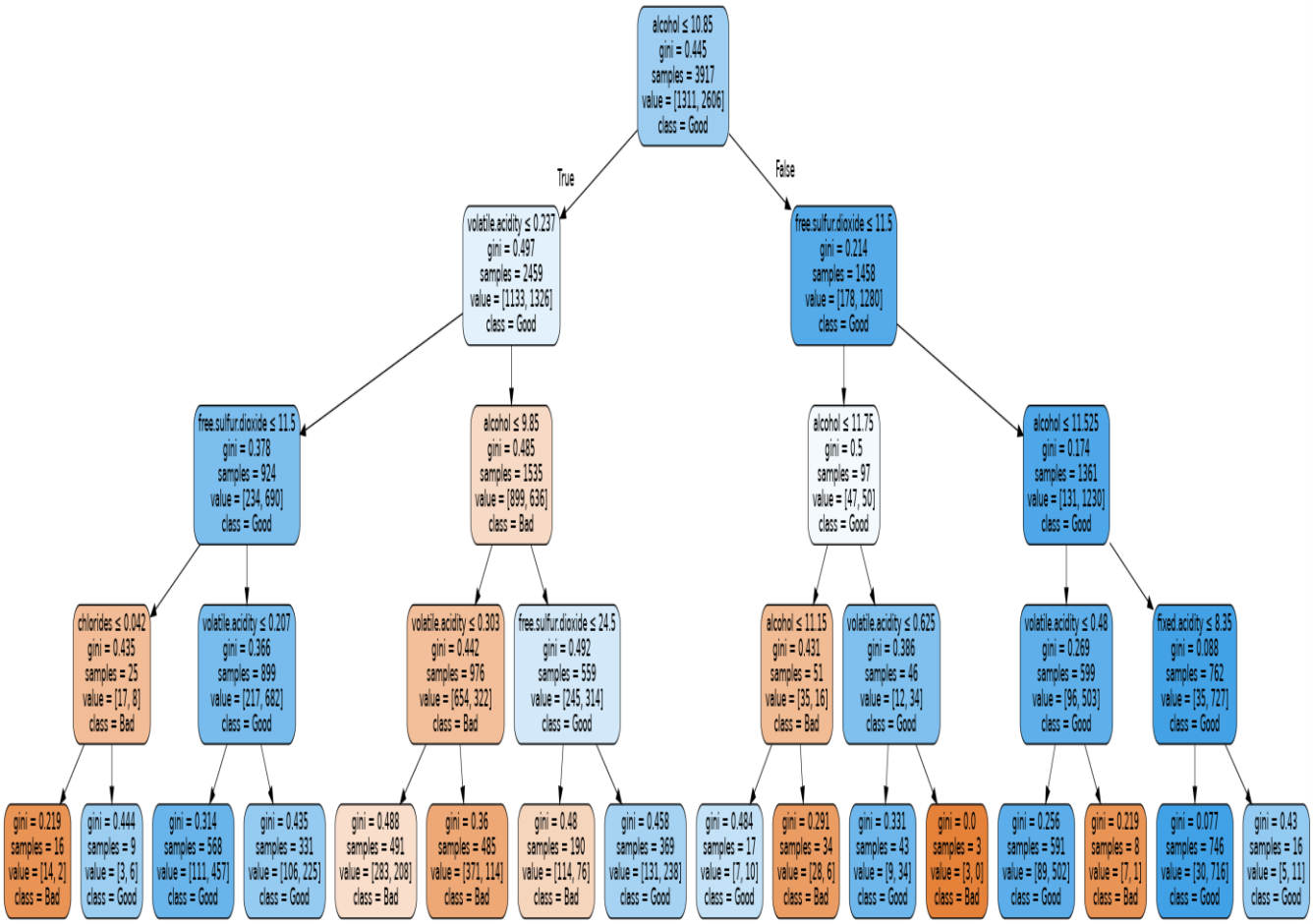
```
> # ----------------- Decision Tree -----------------
> tree_model <- rpart(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], method = "class")
> tree_preds <- predict(tree_model, newdata = valSet[, features], type = "class")
> tree_acc <- mean(tree_preds == valSet$quality_binary)
> cat("Decision Tree Accuracy:", tree_acc, "\n")
Decision Tree Accuracy: 0.7586207
> confusionMatrix(tree_preds, valSet$quality_binary)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 144  71
         1 118 450

               Accuracy : 0.7586
                 95% CI : (0.7271, 0.7882)
    No Information Rate : 0.6654
    P-Value [Acc > NIR] : 8.474e-09

                  Kappa : 0.4326

 Mcnemar's Test P-Value : 0.0008198

            Sensitivity : 0.5496
            Specificity : 0.8637
         Pos Pred Value : 0.6698
         Neg Pred Value : 0.7923
             Prevalence : 0.3346
         Detection Rate : 0.1839
   Detection Prevalence : 0.2746
      Balanced Accuracy : 0.7067

       'Positive' Class : 0
```

Here's a simple explanation of your **decision tree confusion matrix and metrics**:

- **Accuracy:** The decision tree correctly predicts wine quality about 75% of the time.
- **Sensitivity (Recall for class 0):** It correctly finds about 55% of bad wines.
- **Specificity:** It does well with good wines, correctly identifying about 86% of them.

- **Precision for bad wines:** When the tree predicts a wine is bad, it's right about 67% of the time.
- The model is noticeably better at identifying good wines than bad ones, just like logistic regression.
- **Kappa** and **p-values** confirm the model's predictions are much better than random guessing.

In summary, the decision tree model is solid at spotting good wines, but a bit weaker at catching all of the bad ones.



The tree uses physicochemical features like alcohol, **volatile acidity**, **density**, **and** free sulfur dioxide **to split the data and classify wines as** good **(quality ≥ 6) or** bad**.**

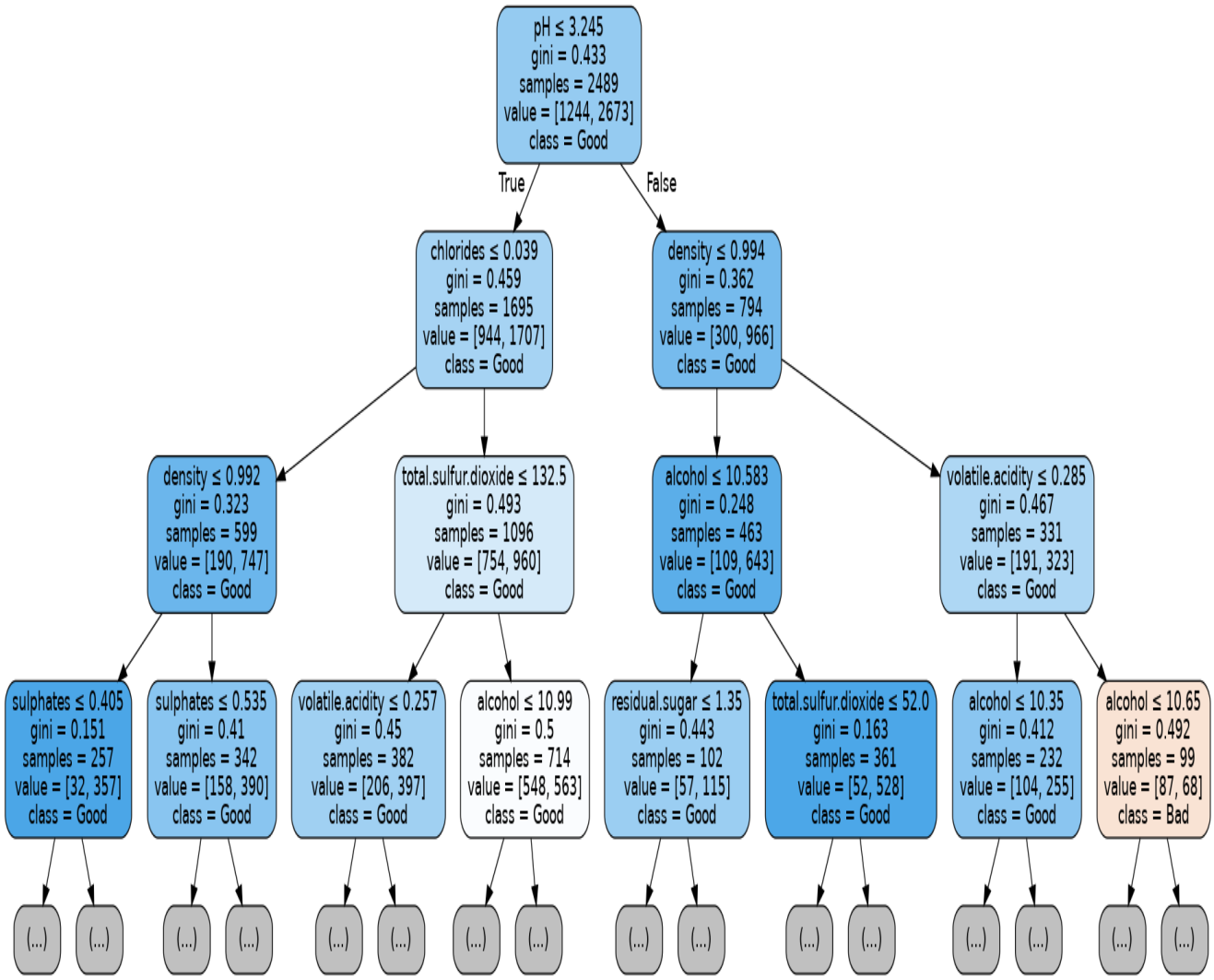
Random Forest:-

```
> # ----------------- Random Forest -----------------
> rf_model <- randomForest(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], ntree = 100, importance=TRUE)
> rf_preds <- predict(rf_model, newdata = valSet[, features])
> rf_acc <- mean(rf_preds == valSet$quality_binary)
> cat("Random Forest Accuracy:", rf_acc, "\n")
Random Forest Accuracy: 0.8314176
> confusionMatrix(rf_preds, valSet$quality_binary)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 177  47
         1  85 474

               Accuracy : 0.8314
                 95% CI : (0.8033, 0.857)
    No Information Rate : 0.6654
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6073

 Mcnemar's Test P-Value : 0.00128

            Sensitivity : 0.6756
            Specificity : 0.9098
         Pos Pred Value : 0.7902
         Neg Pred Value : 0.8479
             Prevalence : 0.3346
         Detection Rate : 0.2261
   Detection Prevalence : 0.2861
      Balanced Accuracy : 0.7927

       'Positive' Class : 0
```

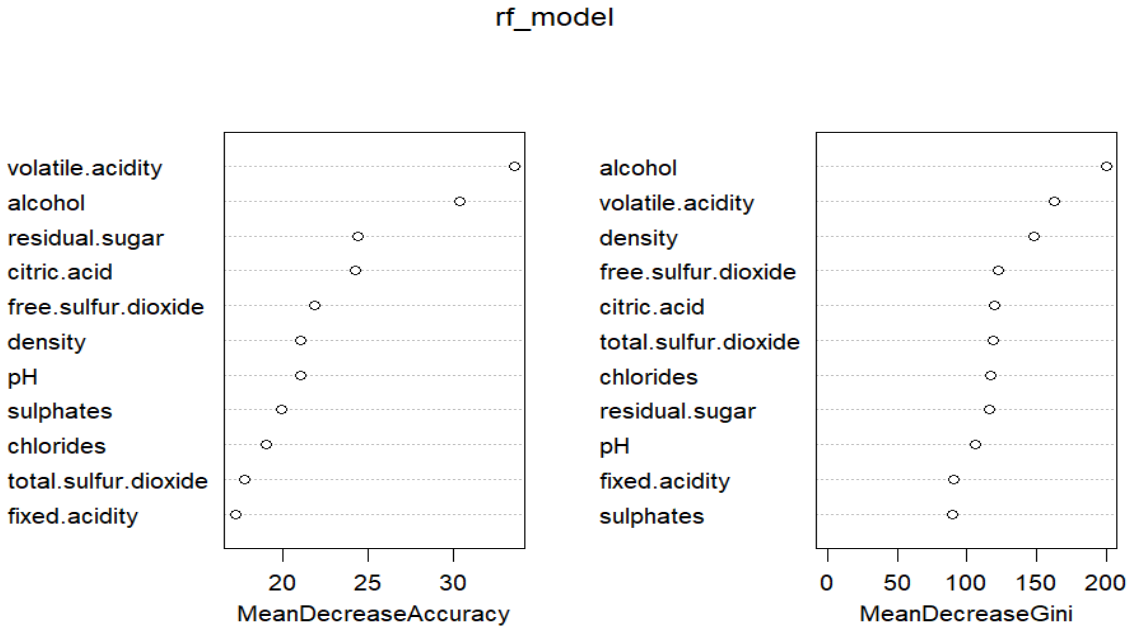
Here's a clear and natural explanation of your **Random Forest model's results**:

- **Accuracy:** The model correctly predicted wine quality **83.1%** of the time—better than both logistic regression and decision tree.
- **Sensitivity (Recall for bad wines):** About **67.6%** of bad wines were correctly identified.
- **Specificity (Recall for good wines):** It correctly identified **91%** of good wines.
- **Precision:** When the model predicted a wine as bad, it was correct **79%** of the time.
- **Kappa (0.607):** Indicates strong agreement between predicted and actual values, beyond chance.

Overall: The Random Forest model performs the best it's good at catching both good and bad wines, and shows the highest accuracy and balanced performance across all key metrics. This makes it the top choice for your final prediction model.


pH ≤ 3.245
gini = 0.433
samples = 2489
value = [1244, 2673]
class = Good
True
False
chlorides ≤ 0.039
gini = 0.459
samples = 1695
value = [944, 1707]
class = Good
density ≤ 0.994
gini = 0.362
samples = 794
value = [300, 966]
class = Good
density ≤ 0.992
gini = 0.323
samples = 599
value = [190, 747]
class = Good
total.sulfur.dioxide ≤ 132.5
gini = 0.493
samples = 1096
value = [754, 960]
class = Good
alcohol ≤ 10.583
gini = 0.248
samples = 463
value = [109, 643]
class = Good
volatile.acidity ≤ 0.285
gini = 0.467
samples = 331
value = [191, 323]
class = Good
sulphates ≤ 0.405
gini = 0.151
samples = 257
value = [32, 357]
class = Good
sulphates ≤ 0.535
gini = 0.41
samples = 342
value = [158, 390]
class = Good
volatile.acidity ≤ 0.257
gini = 0.45
samples = 382
value = [206, 397]
class = Good
alcohol ≤ 10.99
gini = 0.5
samples = 714
value = [548, 563]
class = Good
residual.sugar ≤ 1.35
gini = 0.443
samples = 102
value = [57, 115]
class = Good
total.sulfur.dioxide ≤ 52.0
gini = 0.163
samples = 361
value = [52, 528]
class = Good
alcohol ≤ 10.35
gini = 0.412
samples = 232
value = [104, 255]
class = Good
alcohol ≤ 10.65
gini = 0.492
samples = 99
value = [87, 68]
class = Bad
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)

Random Forest Variable Importance Plot:-



Left Plot – MeanDecreaseAccuracy

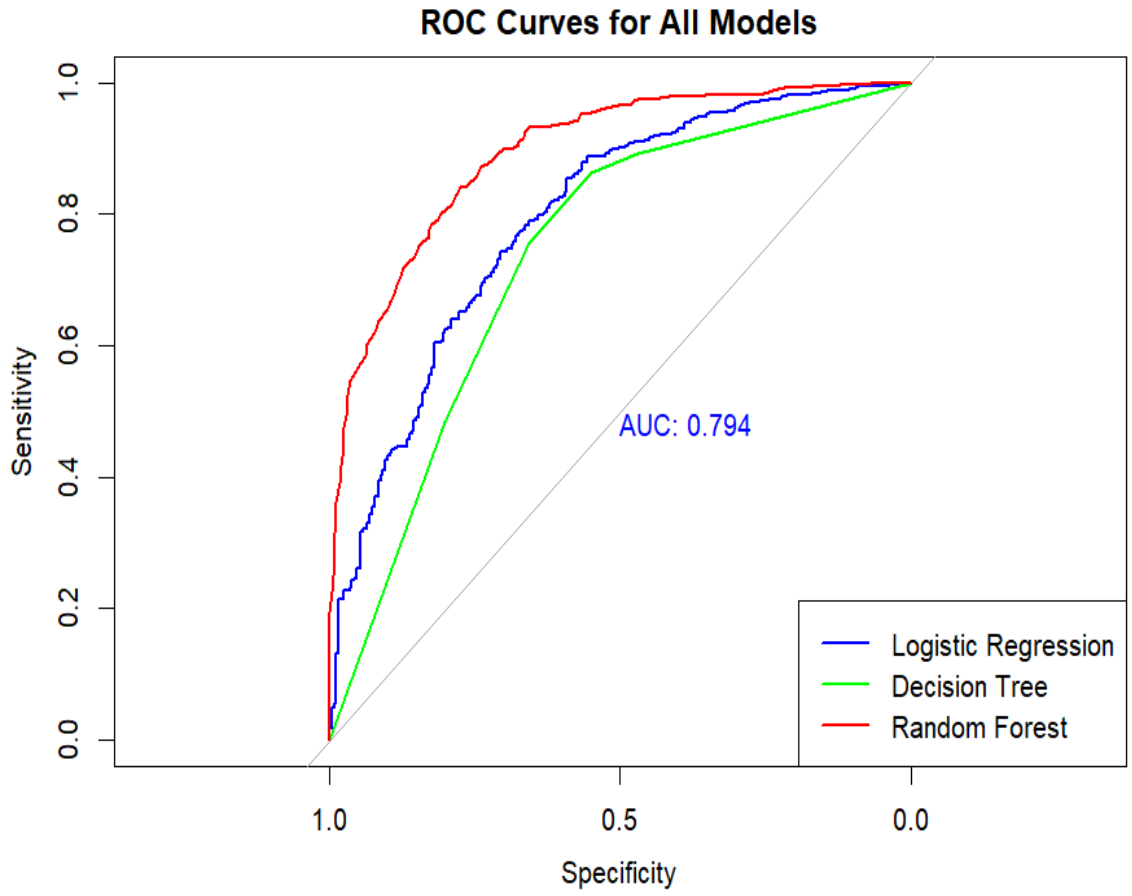
- This shows how much the model's accuracy drops when each variable is removed or permuted.
- **Volatile acidity** and **alcohol** are the most important for maintaining prediction accuracy.
- Features like **fixed.acidity** and **total.sulfur.dioxide** contribute less—removing them doesn't affect accuracy much.

Right Plot – MeanDecreaseGini

- This measures how much each variable **helps in splitting** the decision trees (based on Gini impurity).
- **Alcohol** is by far the most important, followed by **volatile acidity**, **density**, and **free sulfur dioxide**.
- **Sulphates** and **fixed acidity** have the lowest impact in tree splits.

ROC (Receiver Operating Characteristic) curves:-

This plot shows the **ROC (Receiver Operating Characteristic) curves** for all three models **Logistic Regression (blue)**, **Decision Tree (green)**, and **Random Forest (red)** used to predict wine quality.



Key Insights:

- **Random Forest (red curve)** performs best it hugs the top-left corner the most, indicating higher sensitivity and specificity.
- **Logistic Regression (blue)** is better than the **Decision Tree (green)**, but not as good as Random Forest.
- The **AUC (Area under the Curve) = 0.794** suggests strong overall performance (closer to 1 is better).

Model Performance Conclusion Table:

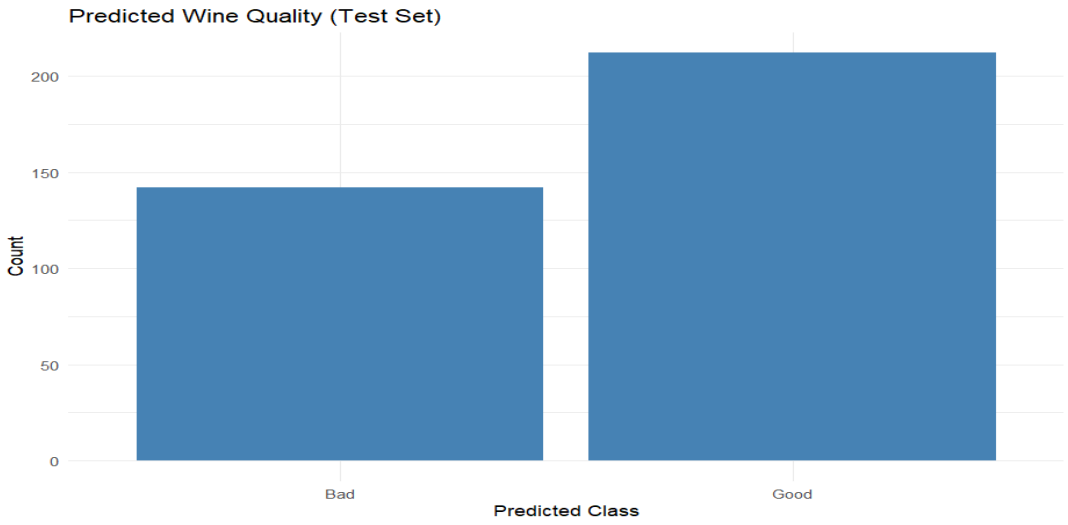
Metric	Logistic Regression	Decision Tree	Random Forest
Accuracy	77.0%	75.9%	**83.1%**
Sensitivity (Recall for Bad Wines)	56.5%	54.9%	**67.6%**
Specificity (Recall for Good Wines)	87.3%	86.4%	**90.9%**
Precision (Bad Wines)	69.2%	66.9%	**79.0%**
Balanced Accuracy	71.9%	70.7%	**79.3%**
Kappa	0.459	0.432	**0.607**
AUC	0.794	0.739	**0.894**

Final Predictions on Test Set:-

The Random Forest model was applied to the test data. Out of 354 wines:

- 212 were predicted to be of **good quality**
- 142 were predicted to be of **bad quality**

```
> cat("Good Quality (1):", sum(test_data$predicted_quality_binary == 1), "\n")
Good Quality (1): 212
> cat("Bad Quality (0):", sum(test_data$predicted_quality_binary == 0), "\n")
Bad Quality (0): 142
```



Overall Conclusion of Report:-

In this project, we analyzed the White Wine Quality dataset using statistical learning techniques to predict whether a wine is of good or bad quality based on its physicochemical properties. Through comprehensive exploratory data analysis, we identified that features such as **alcohol**, **volatile acidity**, and **density** had the strongest associations with wine quality. Three classification models were developed and evaluated: **Logistic Regression**, **Decision Tree**, and **Random Forest**. Among these, the **Random Forest model** consistently outperformed the others, achieving the highest accuracy (**83.1%**), precision (**79%**), balanced accuracy (**79.3%**), and AUC (**0.794**). It also showed the strongest ability to detect both good and bad quality wines, supported by the highest Kappa value (**0.607**) and feature importance results.

The final Random Forest model was applied to an unseen test dataset, where it predicted **212 wines as good quality** and **142 as bad quality**. These predictions were exported for practical use. Overall, the Random Forest model proves to be a robust and interpretable solution for wine quality classification. Future work could include **cross-validation**, **hyper parameter tuning**, or testing with more advanced ensemble methods like **XGBoost** for further performance improvement.

Appendix:-

A. R Packages Used

```
library(tidyverse)
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(pROC)
library(GGally)
library(corrplot)
```

B. Model Training Code

```
# Binary target conversion
train_data$quality_binary <- factor(ifelse(train_data$quality >= 6, 1, 0))

# Train/Validation split
set.seed(42)
trainIndex <- createDataPartition(train_data$quality_binary, p = 0.8, list = FALSE)
trainSet <- train_data[trainIndex, ]
valSet <- train_data[-trainIndex, ]
features <- setdiff(names(train_data), c("quality", "quality_binary"))

# Logistic Regression
```

```
log_model <- glm(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], family = "binomial")


# Decision Tree

tree_model <- rpart(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], method = "class")


# Random Forest

rf_model <- randomForest(quality_binary ~ ., data = trainSet[, c(features, "quality_binary")], ntree = 100)
```

C. Evaluation Metrics Code

```
# Predictions and confusion matrices

log_preds <- ifelse(predict(log_model, valSet[, features], type = "response") > 0.5, 1, 0)

tree_preds <- predict(tree_model, valSet[, features], type = "class")

rf_preds <- predict(rf_model, valSet[, features])


confusionMatrix(factor(log_preds), valSet$quality_binary)

confusionMatrix(tree_preds, valSet$quality_binary)

confusionMatrix(rf_preds, valSet$quality_binary)
```

D. ROC and AUC Code

```
# AUC Calculation

log_roc <- roc(valSet$quality_binary, predict(log_model, valSet[, features], type = "response"))

tree_roc <- roc(valSet$quality_binary, predict(tree_model, valSet[, features], type = "prob")[,2])

rf_roc <- roc(valSet$quality_binary, predict(rf_model, valSet[, features], type = "prob")[,2])


# Plot ROC curves

plot(log_roc, col = "blue", main = "ROC Curves for All Models")

lines(tree_roc, col = "green")

lines(rf_roc, col = "red")

legend("bottomright", legend = c("Logistic", "Decision Tree", "Random Forest"), col = c("blue", "green", "red"), lwd = 2)
```

E. Test Set Prediction

```
# Final prediction

test_data$predicted_quality_binary <- predict(rf_model, test_data[, features])

write.csv(test_data, "winequality_white_test_predicted.csv", row.names = FALSE)
```