Final_Captsone_Project_Own.rmd

Kavish Seechurn

2024-05-14

Quality of Wine | Machine Learning Capstone Project

HarvardX: PH125.9x Data Science Capstone

1. Introduction

Background and Motivation

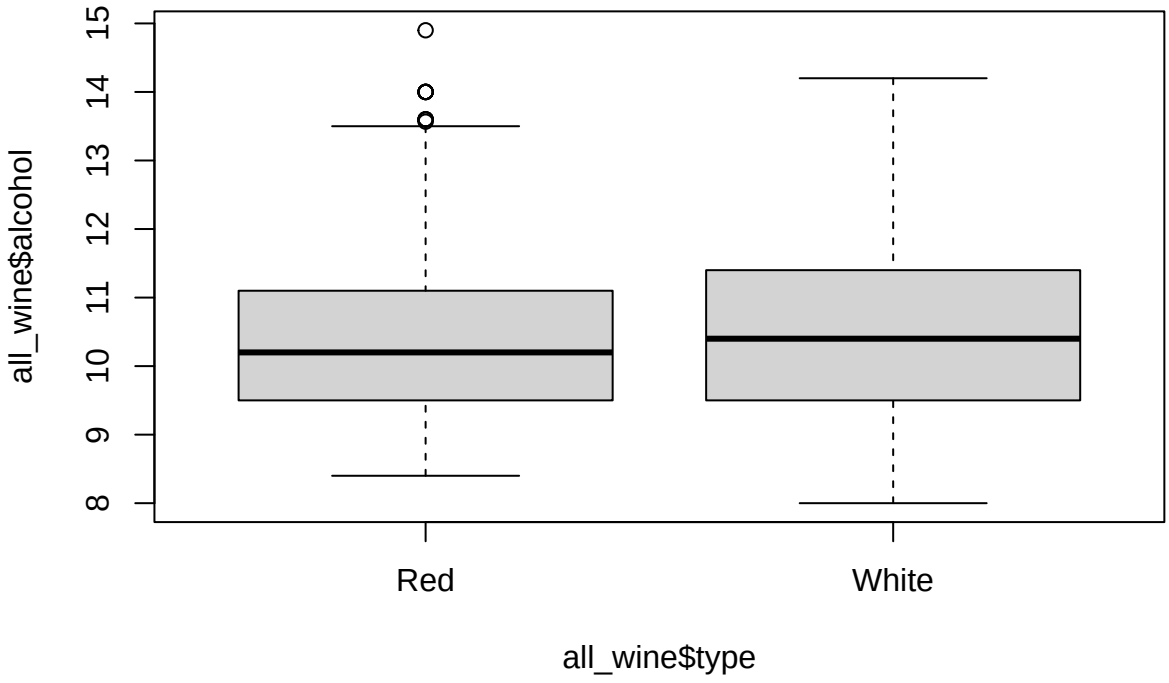
the wine quality dataset has info about red and white wines, like their acidity, alcohol, sugar, and pH levels. It also rates their quality. This data comes from the UCI Machine Learning Repository and is often used to make predictions and explore the data.

Data Loading

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(httr)) install.packages("curl", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("curl", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("curl", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("curl", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("curl", repos = "http://cran.us.r-project.org")
if (!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(recosystem)
library(dplyr)
library(curl)
library(caret)
library(randomForest)
library(gbm)
library(e1071)
library(recosystem)
```

```r
# Define the download URL (constructed from the Kaggle command)
dataset_url <- "https://archive.ics.uci.edu/static/public/186/wine+quality.zip"

# Get the current working directory
current_dir <- getwd()

# Construct the download file path
download_file <- paste0(current_dir, "/", basename(dataset_url))

# Download the file
curl_download(dataset_url, download_file)

# Print a message after download
cat("Downloaded Wine dataset to:", download_file, "\n")

#extract files to the current path
walk(download_file, unzip, exdir = current_dir)

# Function to read and clean the wine data
read_wine_data <- function(file_path) {
  # Read data as a data frame
  data <- read.csv(file_path, header = TRUE, sep = ";")

  # Convert numeric columns to numeric (optional)
  data[, c(2:12)] <- lapply(data[, c(2:12)], as.numeric)

  # Return the data frame
  data
}
# Get current working directory
current_dir <- getwd()

# Construct file paths
red_wine_file <- paste0(current_dir, "/winequality-red.csv")
white_wine_file <- paste0(current_dir, "/winequality-white.csv")

# Try reading data with error handling
tryCatch({
  red_wine <- read_wine_data(red_wine_file)
  white_wine <- read_wine_data(white_wine_file)
  # ... rest of the code for combining, splitting, and saving data
}, error = function(e) {
  # Handle any errors during file reading
  cat("Error:", e$message, "\n")
})

# Combine data (optional for overall analysis)
all_wine <- rbind(red_wine, white_wine)

# Add a column indicating wine type (optional)
all_wine$type <- c(rep("Red", nrow(red_wine)), rep("White", nrow(white_wine)))

# Add a new column 'wine_rating' based on quality
```

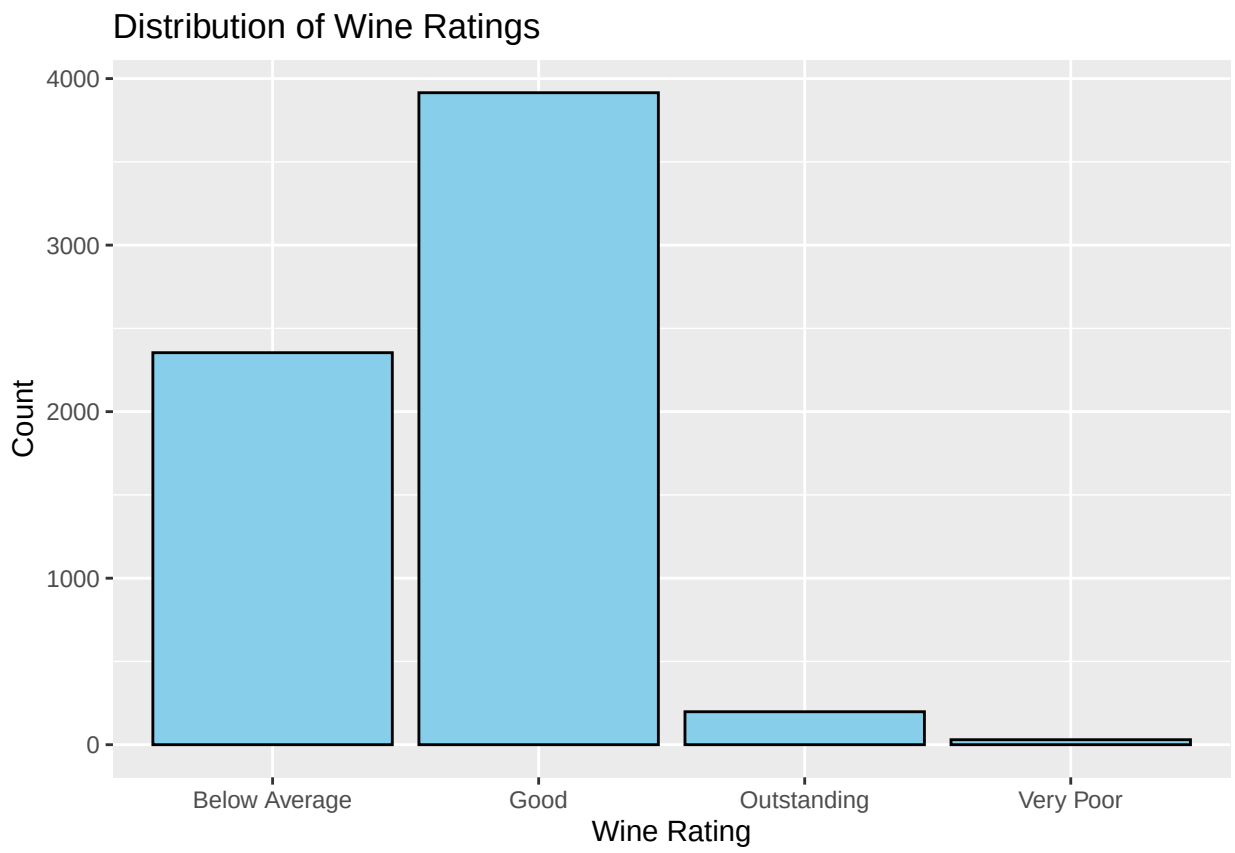
```r
all_wine$wine_rating <- ifelse(all_wine$quality <= 3, "Very Poor",
                               ifelse(all_wine$quality <= 5, "Below Average",
                                      ifelse(all_wine$quality <= 7, "Good", "Outstanding")))


# Explore data (optional)
summary(all_wine)  # View summary statistics
boxplot(all_wine$alcohol ~ all_wine$type)
```



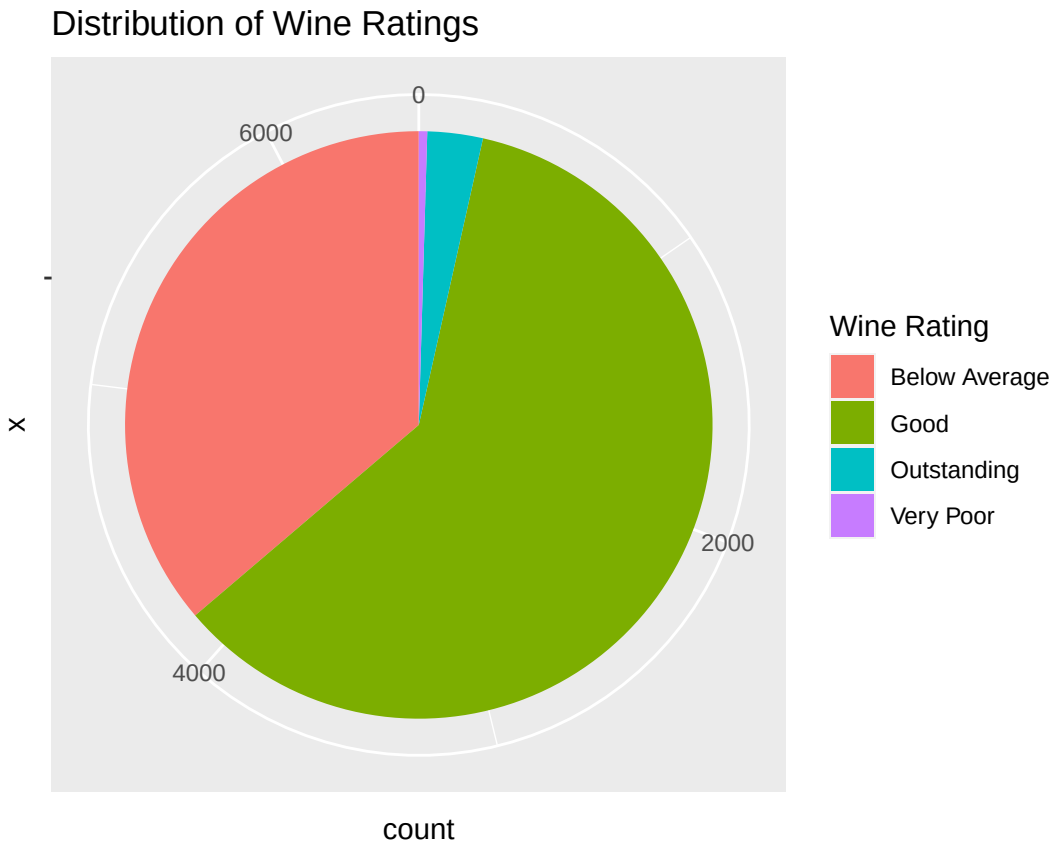

```r
# Split data into train and test sets (80% train, 20% test)
set.seed(123)  # Set random seed for reproducibility
split_index <- createDataPartition(all_wine$quality, p = 0.8, list = FALSE)
train_data <- all_wine[split_index,]
test_data <- all_wine[-split_index,]

# Write train and test data to CSV files
write.csv(train_data, "winequality-train.csv", row.names = FALSE)
write.csv(test_data, "winequality-test.csv", row.names = FALSE)

# Print message
cat("Data split and saved to winequality-train.csv and winequality-test.csv\n")

# Initialize an empty data frame for storing RMSE values
rmse_table <- data.frame(Method = character(), Model_Name = character(), RMSE = numeric())
```

```
### Exploratory Data Analysis (EDA)
```

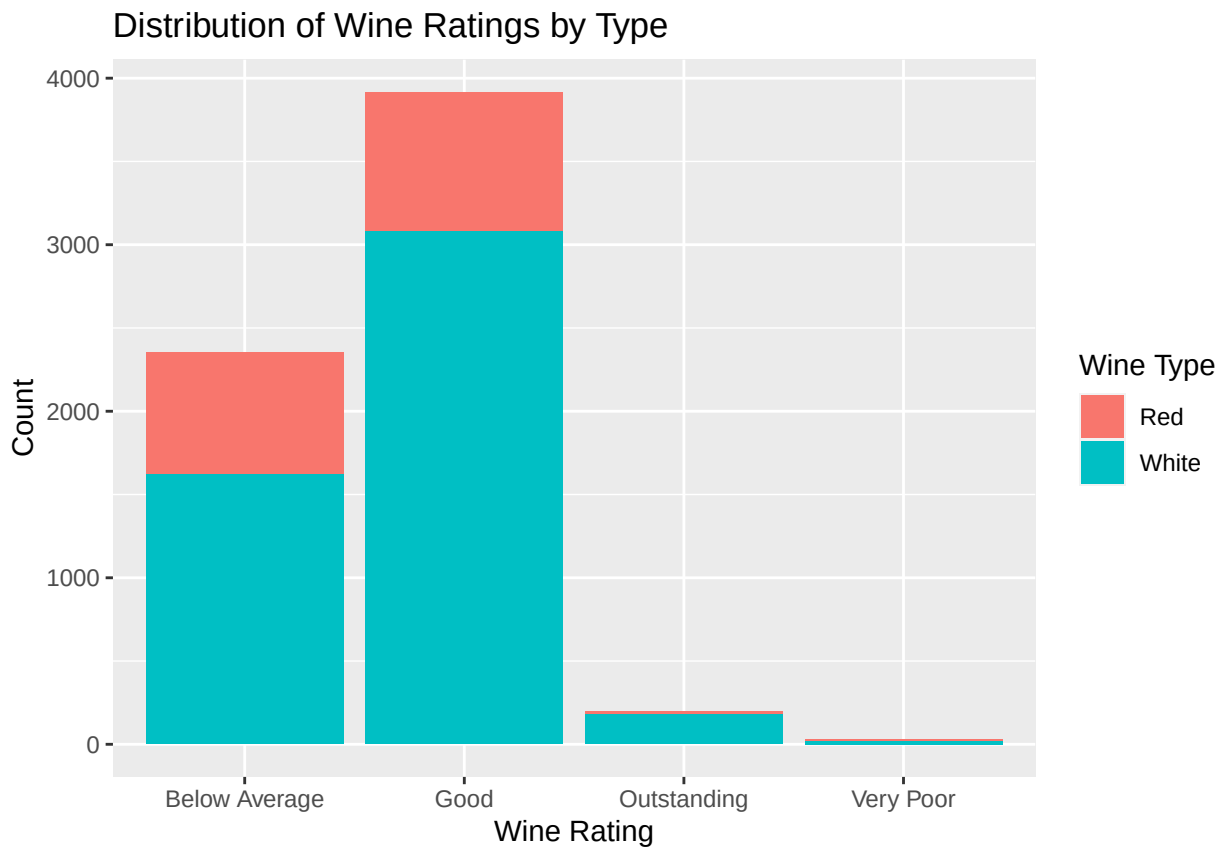
```
# Bar plot of Wine Ratings
ggplot(all_wine, aes(x = wine_rating)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Wine Ratings",
       x = "Wine Rating",
       y = "Count")
```




```
# Pie Chart of Wine Ratings
ggplot(all_wine, aes(x = "", fill = wine_rating)) +
  geom_bar(width = 1) +
  coord_polar("y", start = 0) +
  labs(title = "Distribution of Wine Ratings",
       fill = "Wine Rating")
```

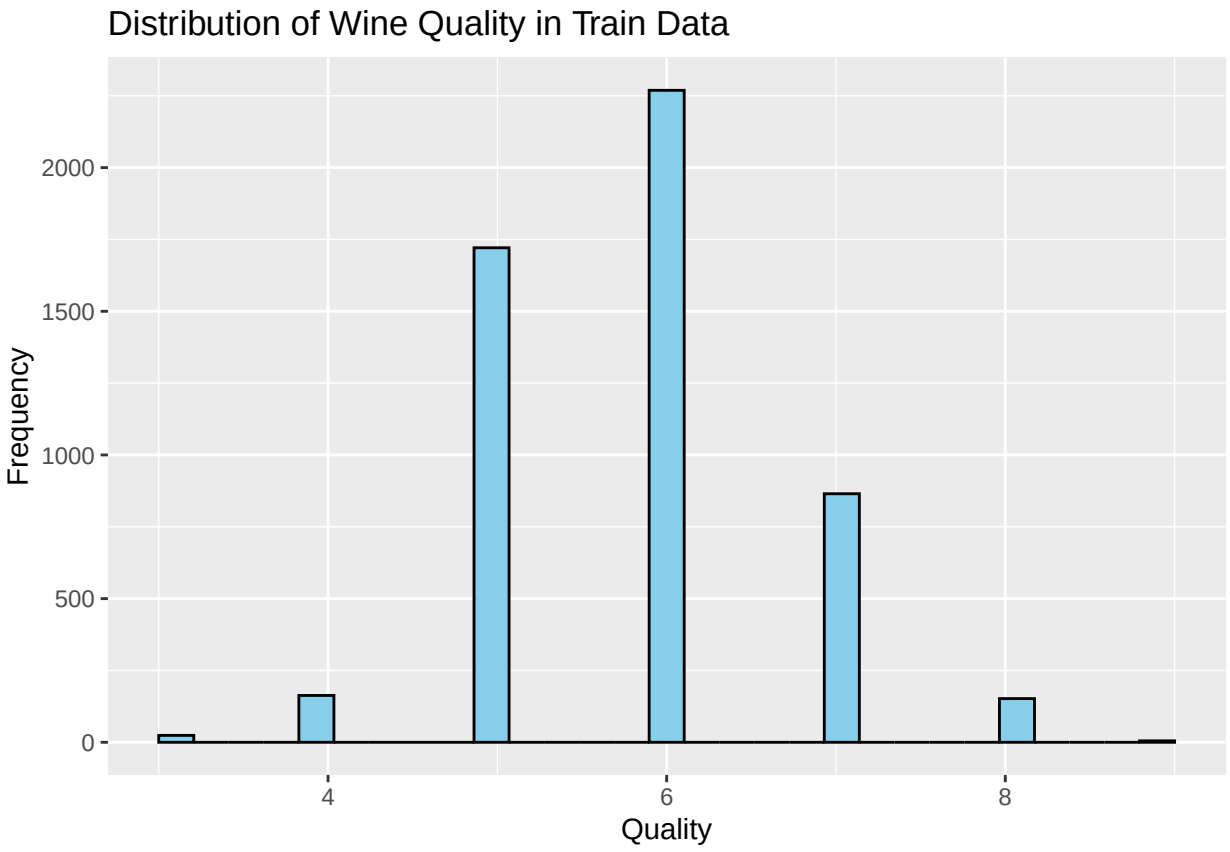



```
# Stacked Bar Plot by Wine Type
ggplot(all_wine, aes(x = wine_rating, fill = type)) +
  geom_bar(position = "stack") +
  labs(title = "Distribution of Wine Ratings by Type",
       x = "Wine Rating",
       y = "Count",
       fill = "Wine Type")
```

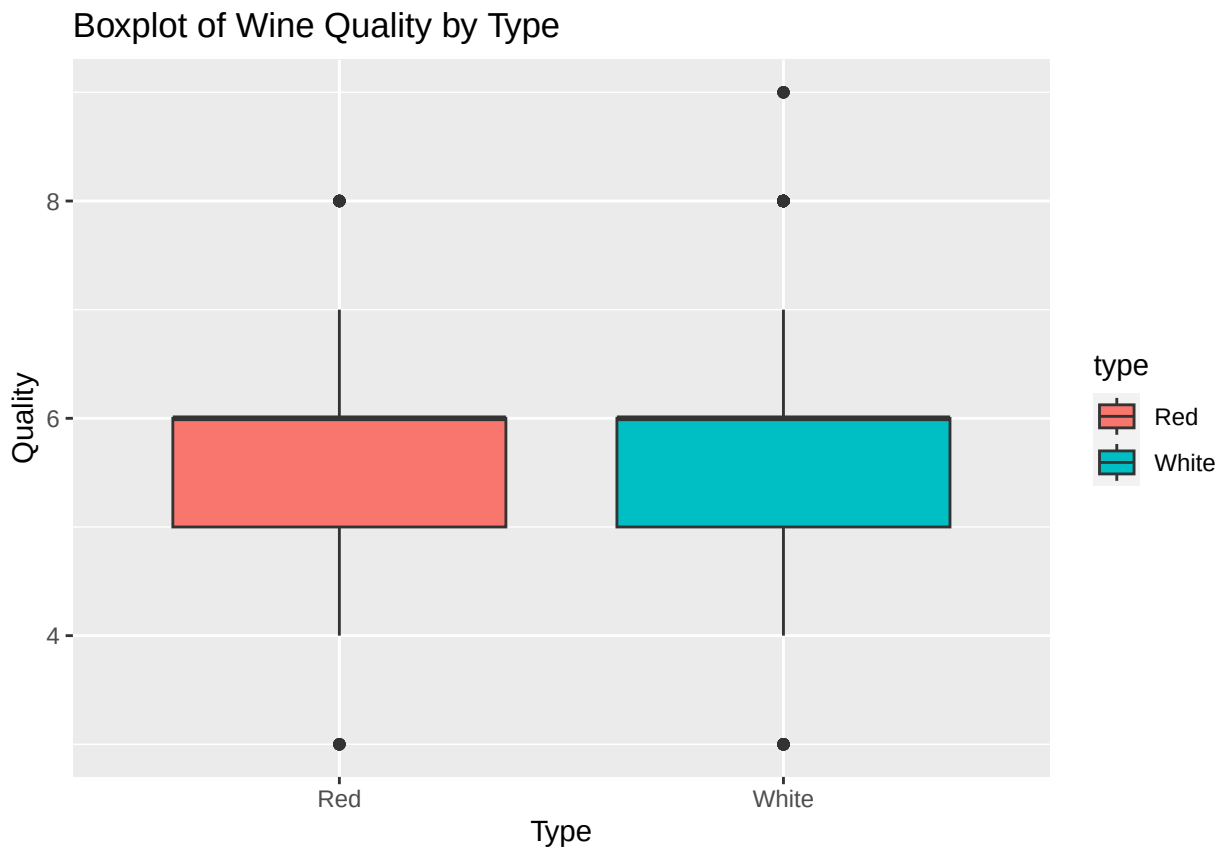



```
#Histogram of Quality:
ggplot(train_data, aes(x = quality)) +
  geom_histogram(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Wine Quality in Train Data",
       x = "Quality",
       y = "Frequency")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

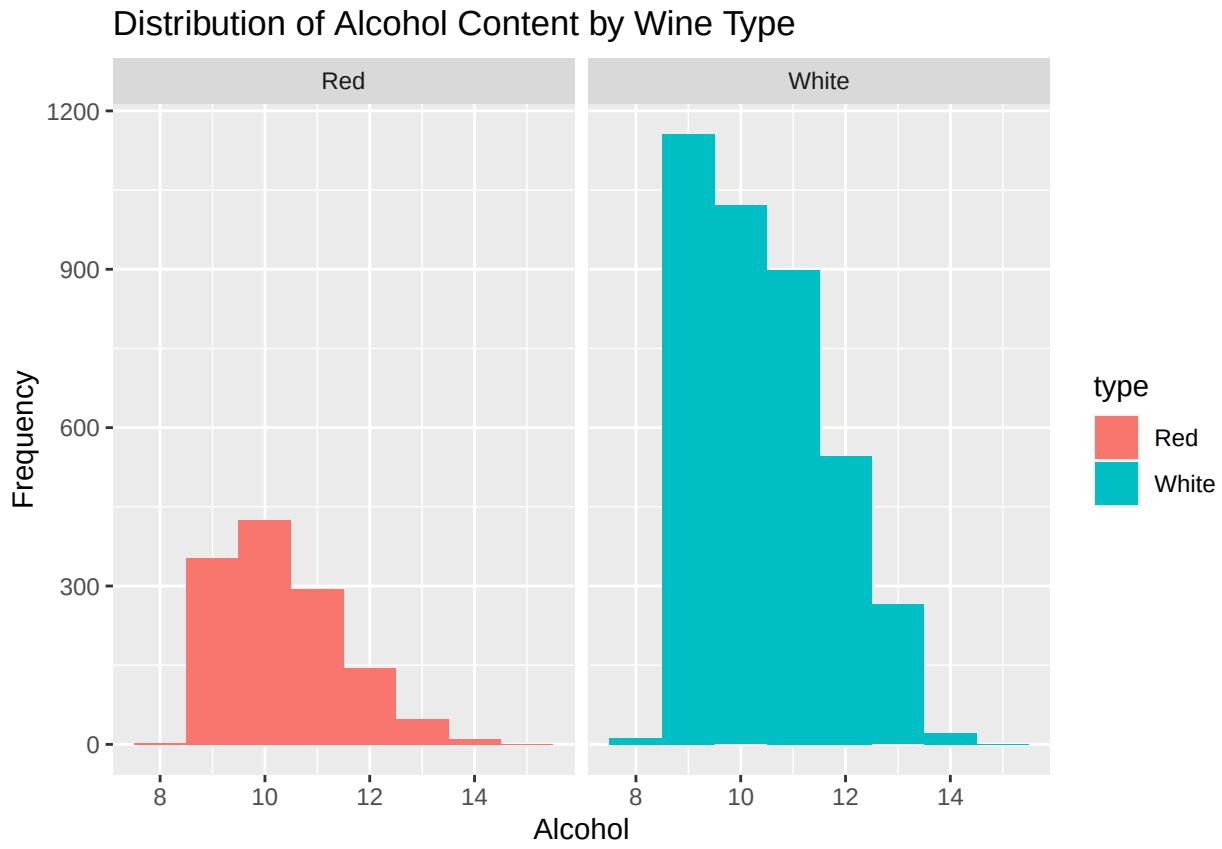



```r
# Boxplot of Wine Quality by Type:
ggplot(train_data, aes(x = type, y = quality, fill = type)) +
  geom_boxplot() +
  labs(title = "Boxplot of Wine Quality by Type",
       x = "Type",
       y = "Quality")
```




```
# Scatterplot of Alcohol Content vs. Quality:
ggplot(train_data, aes(x = alcohol, y = quality)) +
  geom_point(color = "darkgreen") +
  labs(title = "Scatterplot of Alcohol Content vs. Quality",
       x = "Alcohol",
       y = "Quality")
```


```
# Faceted Histograms of Various Attributes:
ggplot(train_data, aes(x = alcohol, fill = type)) +
  geom_histogram(binwidth = 1, position = "dodge") +
  facet_wrap(~type) +
  labs(title = "Distribution of Alcohol Content by Wine Type",
       x = "Alcohol",
       y = "Frequency")
```




```
### Training and Testing of the Algorithm
### Model 1 - Simple Linear Regression
# Train a linear regression model
lm_model <- train(quality ~ ., data = train_data, method = "lm")

# Summarize the model
print(lm_model)
```

```
## Linear Regression
##
## 5199 samples
##   13 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 5199, 5199, 5199, 5199, 5199, 5199, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3742107  0.8149187  0.2895875
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# Make predictions on the test dataset
predictions <- predict(lm_model, newdata = test_data)
```

```r
# Evaluate model performance
RMSE <- sqrt(mean((test_data$quality - predictions)^2))
print(paste("Root Mean Squared Error (RMSE) - Linear Regression:", RMSE))
```

```
## [1] "Root Mean Squared Error (RMSE) - Linear Regression: 0.381816202813976"
```

```r
# Add RMSE value to the table
rmse_table <- rbind(rmse_table, data.frame(Method = "Linear Regression", Model_Name = "lm_model", RMSE =

### Model 2 - Random Forest
# Train a random forest regression model
rf_model <- randomForest(quality ~ ., data = train_data)

# Make predictions on the test dataset
predictions_rf <- predict(rf_model, newdata = test_data)

# Evaluate model performance
RMSE_rf <- sqrt(mean((test_data$quality - predictions_rf)^2))
print(paste("Root Mean Squared Error (RMSE) - Random Forest:", RMSE_rf))
```

```
## [1] "Root Mean Squared Error (RMSE) - Random Forest: 0.352499703521206"
```

```r
# Add RMSE value to the table
rmse_table <- rbind(rmse_table, data.frame(Method = "Random Forest", Model_Name = "rf_model", RMSE = RM

### Model 3 - Support Vector Regression
# Train a support vector regression model
svm_model <- svm(quality ~ ., data = train_data)

# Make predictions on the test dataset
predictions_svm <- predict(svm_model, newdata = test_data)

# Evaluate model performance
RMSE_svm <- sqrt(mean((test_data$quality - predictions_svm)^2))
print(paste("Root Mean Squared Error (RMSE) - Support Vector Regression:", RMSE_svm))
```

```
## [1] "Root Mean Squared Error (RMSE) - Support Vector Regression: 0.414867921048729"
```

```r
# Add RMSE value to the table
rmse_table <- rbind(rmse_table, data.frame(Method = "Support Vector Regression", Model_Name = "svm_model



### Model 4 - Matrix Factorization using SVD

##Prepare data
numeric_columns <- sapply(all_wine, is.numeric)  # Identify numeric columns
wine_matrix <- as.matrix(all_wine[, numeric_columns])  # Select numeric columns
missing_values <- any(is.na(wine_matrix))  # Check for missing values

if (missing_values) {
```

```r
  # Handle missing values appropriately
  wine_matrix <- na.omit(wine_matrix)  # Remove rows with missing values
}

# Perform Singular Value Decomposition (SVD)
svd_result <- svd(wine_matrix)

# Number of dimensions to keep (k)
k <- 10  # Adjust as needed

# Retrieve U, Sigma, and V matrices from SVD
U <- svd_result$u[, 1:k]  # Keep only k dimensions
Sigma <- diag(svd_result$d[1:k])  # Keep only k dimensions
V <- svd_result$v[, 1:k]  # Keep only k dimensions

# Perform matrix multiplication to reconstruct the original matrix
reconstructed_matrix <- U %*% Sigma %*% t(V)

# Extract predicted ratings (quality)
predicted_ratings <- reconstructed_matrix[, which(names(all_wine) == "quality")]

# Calculate RMSE
RMSE <- sqrt(mean((predicted_ratings - wine_matrix[, "quality"])^2))


rmse_table <- rbind(rmse_table, data.frame(Method = "Matrix Factorization using SVD", Model_Name = "SVD

# Add RMSE value to the table
print(rmse_table)
```

```
##                           Method Model_Name         RMSE
## 1              Linear Regression   lm_model 0.3818162028
## 2                  Random Forest   rf_model 0.3524997035
## 3      Support Vector Regression  svm_model 0.4148679210
## 4 Matrix Factorization using SVD        SVD 0.0001052307
```

```r
###Selection
```

Based on the table, We can conclude the following:

Random Forest has the lowest RMSE among all the models, indicating that it performs the best in predicting wine quality based on the given features. Linear Regression and Support Vector Regression have higher RMSE compared to Random Forest, suggesting that they are less effective in capturing the underlying patterns in the data. The Matrix Factorization using SVD model has an extremely low RMSE, almost close to zero. However, this could indicate overfitting or some other anomaly in the model. Further investigation is needed to validate its performance. In summary, based on RMSE values alone, Random Forest appears to be the most promising model for predicting wine quality in this dataset.