UNIVERSITY OF SCIENCE, VNU-HCM
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

BÁO CÁO ĐỒ ÁN

DỰ ĐOÁN
CHẤT LƯỢNG RƯỢU VANG

Course: **Xử lý số liệu thống kê**

Instructor: TS. Tô Đức Khánh

Students: Bùi Quang Thắng - 21280048
Huỳnh Thị Thu Thoảng - 21280074
Nguyễn Thị Bích Ngọc - 21280100
Đoàn Thị Mẫn Nhi - 21280102
Nguyễn Thúy Vy - 21280120

HO CHI MINH CITY, JULY 07, 2024

Contents

1 Giới thiệu bài toán

Trong bài toán này, chúng ta sẽ phân loại chất lượng rượu vang dựa trên các đặc trưng hóa học của chúng. Bộ dữ liệu bao gồm thông tin về hai loại rượu vang (đỏ và trắng) với các đặc trưng như độ pH, hàm lượng đường, và độ cồn. Mục tiêu là xây dựng một mô hình dự đoán chất lượng rượu vang, được đánh giá theo thang điểm từ 3 đến 9, từ đó giúp các nhà sản xuất rượu vang cải thiện quy trình sản xuất và đảm bảo chất lượng sản phẩm.

Phương pháp sử dụng:

- **Xử lý dữ liệu:** Kết hợp hai bộ dữ liệu rượu vang đỏ và trắng, kiểm tra và xử lý các giá trị thiếu (nếu có), và chuyển đổi các cột thành kiểu dữ liệu phù hợp.
- **Xử lý imbalance:** Sử dụng kỹ thuật SMOTE để cân bằng lại số lượng mẫu giữa các lớp chất lượng rượu vang.
- **Train mô hình:** Sử dụng và so sánh các mô hình để phân loại chất lượng rượu vang, bao gồm:
  - Random Forest: model phân loại sử dụng kỹ thuật bagging để kết hợp nhiều Decision Tree.
  - Naive Bayes: model phân loại sử dụng xác suất Bayes để tính các xác suất có điều kiện giữa các class.
- **Đánh giá mô hình:** Sử dụng confusion matrix để đánh giá hiệu suất của mô hình và phương pháp bootstrap để ước lượng độ chính xác và độ tin cậy của mô hình.
- **Phân tích feature importance:** Sử dụng độ giảm Gini để xác định tầm quan trọng của các đặc trưng đối với chất lượng rượu vang.
- **Sử dụng Bootstrap:** Áp dụng phương pháp bootstrap để đánh giá sự ổn định và độ tin cậy của mô hình bằng cách lặp lại quá trình huấn luyện trên các mẫu con của dữ liệu gốc và đo lường độ chính xác trên tập kiểm tra.

2 Khái quát về dữ liệu

- Dữ liệu có 2 file là `winequality-red.csv` và `winequality-white.csv` với các cột tương đồng nhau, nên ta cần merge 2 file lại cùng 1 dataframe chung, và tạo cột mới là `wine_type` để phân biệt giữa rượu vang đỏ và trắng, cuối cùng là chuyển cột `quality` thành kiểu dữ liệu `factor` (đây là sẽ là column target để handle target value).
- Đối với từng file dữ liệu:
  - File `winequality-red.csv` chứa 1599 dòng và 12 cột.
  - File `winequality-white.csv` chứa 4898 dòng và 12 cột.
- Việc ta gộp lại là để dễ so sánh, cũng như làm EDA cho cả 2 loại rượu vang.
- Khi load data, ta cũng sử dụng hàm `clean_names` để chuyển tên cột thành dạng snake_case, giúp cho việc xử lý dữ liệu dễ dàng hơn.

```
GLOBAL_PATH <- "~/Data"
winequality_red <- read.csv(glue("{GLOBAL_PATH}/winequality-red.csv"),
                            sep = ";") %>% clean_names()
winequality_white <- read.csv(glue("{GLOBAL_PATH}/winequality-white.csv"),
                              sep = ",") %>% clean_names()

# Thêm cột loại rượu
winequality_red$wine_type <- 0
winequality_white$wine_type <- 1

# merge lại
combined_wines <- rbind(winequality_red, winequality_white)
combined_wines$quality <- as.factor(combined_wines$quality)

# show only head and last 5 columns
kable(head(combined_wines[, 8:12], 8))
```

density	p_h	sulphates	alcohol	quality
0.9978	3.51	0.56	9.4	5
0.9968	3.20	0.68	9.8	5
0.9970	3.26	0.65	9.8	5
0.9980	3.16	0.58	9.8	6
0.9978	3.51	0.56	9.4	5
0.9978	3.51	0.56	9.4	5
0.9964	3.30	0.46	9.4	5
0.9946	3.39	0.47	10.0	7

- Vì các cột của cả hai dataset là giống nhau, nên ta sẽ chỉ tóm tắt về dữ liệu của một dataset. Các dữ liệu về mặt ý nghĩa và kiểu dữ liệu cũng sẽ tương tự đối với `winequality-white.csv` hay `winequality-red.csv`.
- Bảng thống kê tổng hợp các biến định lượng:

```
quantitative_vars <- c('fixed_acidity', 'volatile_acidity', 'citric_acid',
                       'residual_sugar', 'chlorides', 'free_sulfur_dioxide',
                       'total_sulfur_dioxide', 'density', 'p_h', 'sulphates',
                       'alcohol')

df_sum <- combined_wines %>%
  summarise(across(all_of(quantitative_vars),
    list(min = min, max = max, median = median, mean = mean, sd = sd),
    .names = "{.col}-{.fn}"))
```

```
df_stats_tidy <- df_sum  %>% gather(ten, gt)  %>%
separate(ten, into = c("bien", "tk"), sep = "-")  %>%
spread(tk, gt)
print.data.frame(df_stats_tidy, digits = 4)
```

```
##                    bien     max      mean   median    min        sd
## 1               alcohol  14.900  10.49180  10.3000 8.0000  1.192712
## 2             chlorides   0.611   0.05603   0.0470 0.0090  0.035034
## 3           citric_acid   1.660   0.31863   0.3100 0.0000  0.145318
## 4               density   1.039   0.99470   0.9949 0.9871  0.002999
## 5         fixed_acidity  15.900   7.21531   7.0000 3.8000  1.296434
## 6   free_sulfur_dioxide 289.000  30.52532  29.0000 1.0000 17.749400
## 7                   p_h   4.010   3.21850   3.2100 2.7200  0.160787
## 8        residual_sugar  65.800   5.44324   3.0000 0.6000  4.757804
## 9             sulphates   2.000   0.53127   0.5100 0.2200  0.148806
## 10 total_sulfur_dioxide 440.000 115.74457 118.0000 6.0000 56.521855
## 11     volatile_acidity   1.580   0.33967   0.2900 0.0800  0.164636
```

- Tóm tắt các giá trị định tính có trong dữ liệu:

Tên cột	Ý nghĩa	Giá trị
wine_type	Loại rượu (là cột tự tạo)	0 (red), 1 (white)
quality	Chất lượng rượu	3, 4, 5, 6, 7, 8, 9

- Với cột `quality`, giá trị càng cao thì chất lượng rượu càng tốt, và ngược lại.
- Số lượng mẫu của mỗi class trong cột `quality` của dữ liệu `winequality-red.csv`:

```
table(winequality_red$quality)
```

```
##
##   3   4   5   6   7   8
##  10  53 681 638 199  18
```

- Số lượng mẫu của mỗi class trong cột `quality` của dữ liệu `winequality-white.csv`:

```
table(winequality_white$quality)
```

```
##
##    3    4    5    6    7    8    9
##   20  163 1457 2198  880  175    5
```

- Dữ liệu cho thấy có rất nhiều giá trị định lượng (numerical), là các feature liên quan đến hóa học của rượu vang, và cột giá trị định tính (categorical) có tên `quality`, sẽ là cột target để phân loại chất lượng rượu vang.

3 Đề xuất và mục tiêu phân tích

- Đề xuất phân tích cho dữ liệu đã thu thập được:

Vấn đề	Đề xuất	Phương pháp sử dụng
Tìm hiểu dữ liệu	Kết hợp hai bộ dữ liệu rượu vang đỏ và trắng, kiểm tra và xử lý các giá trị thiếu (nếu có), và chuyển đổi các cột thành kiểu dữ liệu phù hợp.	Lập bảng thống kê tóm tắt dữ liệu định tính, định lượng Vẽ biểu đồ phân phối (cột, heatmap, pairplot, scatterplot, violin,…)
Dữ liệu bị imbalance	Relabel lại data. Sử dụng phương pháp cân bằng lại dữ liệu.	Relabel lại thành 5 classes (3-4, 5, 6, 7, 8-9). Chia dữ liệu thành train-test. Sử dụng phương pháp SMOTE.
Phân loại dữ liệu	Sử dụng các mô hình được liệt kê trong học phần để tiến hành phân loại trên cả dữ liệu gộp 2 loại rượu và trên từng loại rượu để so sánh.	Sử dụng các mô hình Naive Bayes và Random Forest
Cải thiện mô hình	Sử dụng phương pháp cải thiện	Sử dụng phương pháp bootstrap
Đặc trưng quan trọng của mô hình	Tìm đặc trưng quan trọng cho việc đánh giá ý nghĩa mô hình	Sử dụng phương pháp tìm độ quan trọng của các feature
Thực hiện thêm 1 thuật toán bên ngoài	Thử áp dụng một thuật toán bên ngoài chương trình học cho mô hình	Sử dụng thuật toán SVM
Đánh giá mô hình	Đánh giá giữa việc gộp 2 class/từng loại Đánh giá giữa các mô hình phân loại đã được chọn	Đánh giá dựa trên accuracy

- Mục tiêu chính của bài toán là xây dựng một mô hình phân loại chất lượng rượu vang dựa trên các đặc trưng hóa học của rượu, với mục tiêu đạt được độ chính xác cao nhất có thể. Trong đó cụ thể các mục tiêu khi thực hiện bài toán như sau:

  - Làm sạch: đảm bảo dữ liệu nhất quán, không bị trùng lặp và không chứa dữ liệu thiếu.
  - Khám phá: hiểu được cấu trúc của dữ liệu, tầm quan trọng của các biến giải thích và phân phối của biến mục tiêu
  - Xây dựng mô hình phân loại: tìm được mô hình phân loại phù hợp với hiệu suất có thể chấp nhận được (accuracy từ 0.6 trở lên), có khả năng phân loại rượu chất lượng thấp (quality = 3,4) và rượu chất lượng cao (quality = 8,9)
  - Đưa ra đề xuất: tìm ra được những yếu tố (biến) ảnh hưởng đến chất lượng rượu để từ đó cải thiện chất lượng rượu phù hợp với thị trường hơn.

- Với các đề xuất và mục tiêu đã đề ra, ta sẽ tiến hành thực hiện các bước phân tích dữ liệu, xử lý dữ liệu, xây dựng mô hình và đánh giá mô hình để mang lại kết quả tốt nhất cho bài toán phân loại chất lượng rượu vang.

4 EDA - Exploratory Data Analysis

- Trước khi xây dựng mô hình, ta cần phải hiểu rõ về dữ liệu, qua đó có thể đưa ra những nhận xét và giả định về dữ liệu, từ đó có thể xử lý dữ liệu một cách hiệu quả.
- Ta kiểm tra các giá trị bị thiếu bằng hàm `is.na`:

```
sum(is.na(combined_wines))
```

```
## [1] 0
```

- Từ đây cho thấy dữ liệu cho thấy không có giá trị missing nào, điều này chứng tỏ ta không cần phải dùng các kỹ thuật để handle missing data, hay điền khuyết giá trị.
- Dưới đây sẽ là phân phối của các class theo cột quality, bài toán yêu cầu là phân loại chất lượng rượu vang, thì sẽ liên quan đến cột `quality` này. Ta có các class với giá trị trải dài từ 3 đến 9.

```
combined_wines$quality <- as.factor(combined_wines$quality)
table(combined_wines$quality)
```

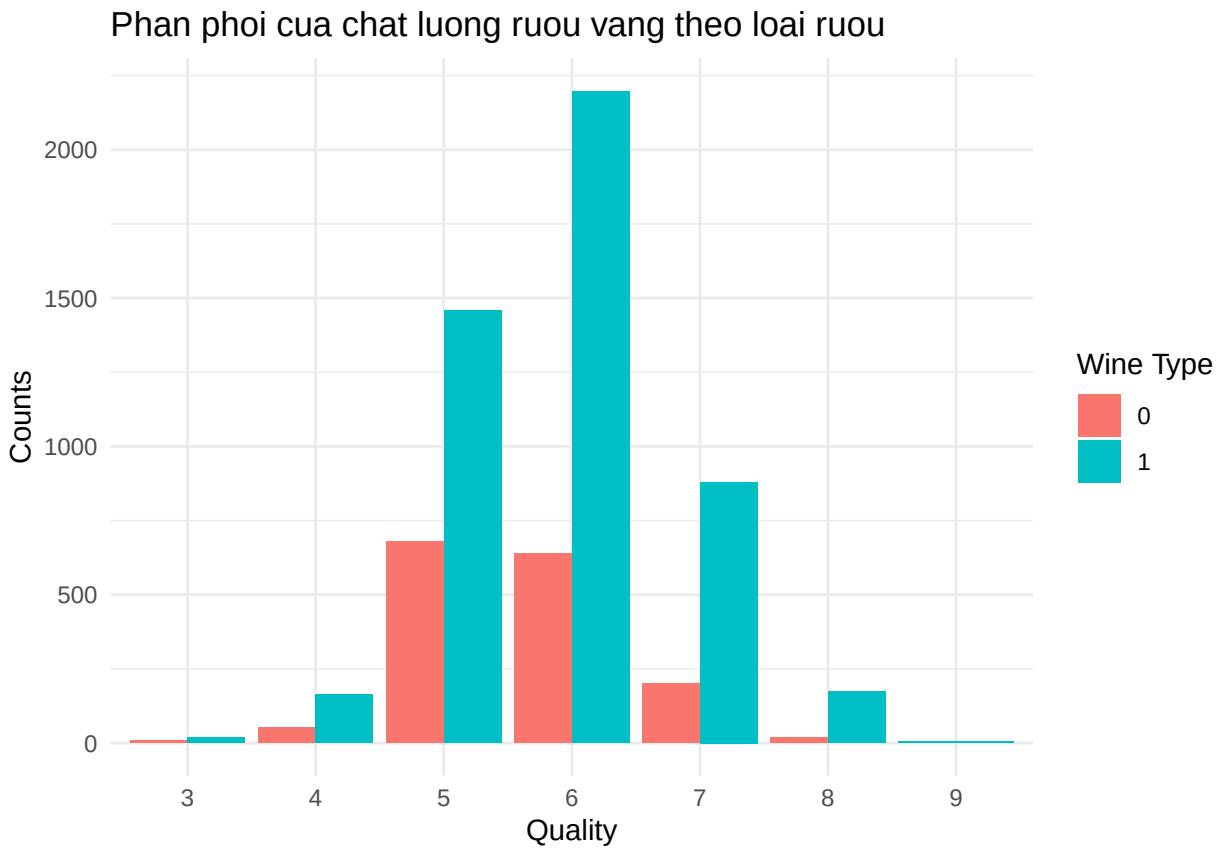
```
##
##    3    4    5    6    7    8    9
##   30  216 2138 2836 1079  193    5
```

- **Nhận xét:** data bị imblance ở chỗ: thiếu các chai rượu có chất lượng rất thấp (3,4,...) và rất cao (9,8,...). Điều này sẽ ảnh hưởng đến việc train mô hình, vì mô hình sẽ dễ bị bias về các class có số lượng mẫu nhiều hơn. Do đó, ta cần relabel và resample

lại dữ liệu để cân bằng số lượng mẫu giữa các class, mà ý tưởng chính ở đây là gộp class 3 và 4 thành một class, và class 8 và 9 thành một class, sau đó sử dụng kỹ thuật SMOTE để oversample số lượng mẫu tương đương giữa các class. Do 3 và 9 có quá ít để oversample, nên việc relabel lại như trên là việc cần thiết.

- Phân phối của chất lượng rượu vang theo loại rượu

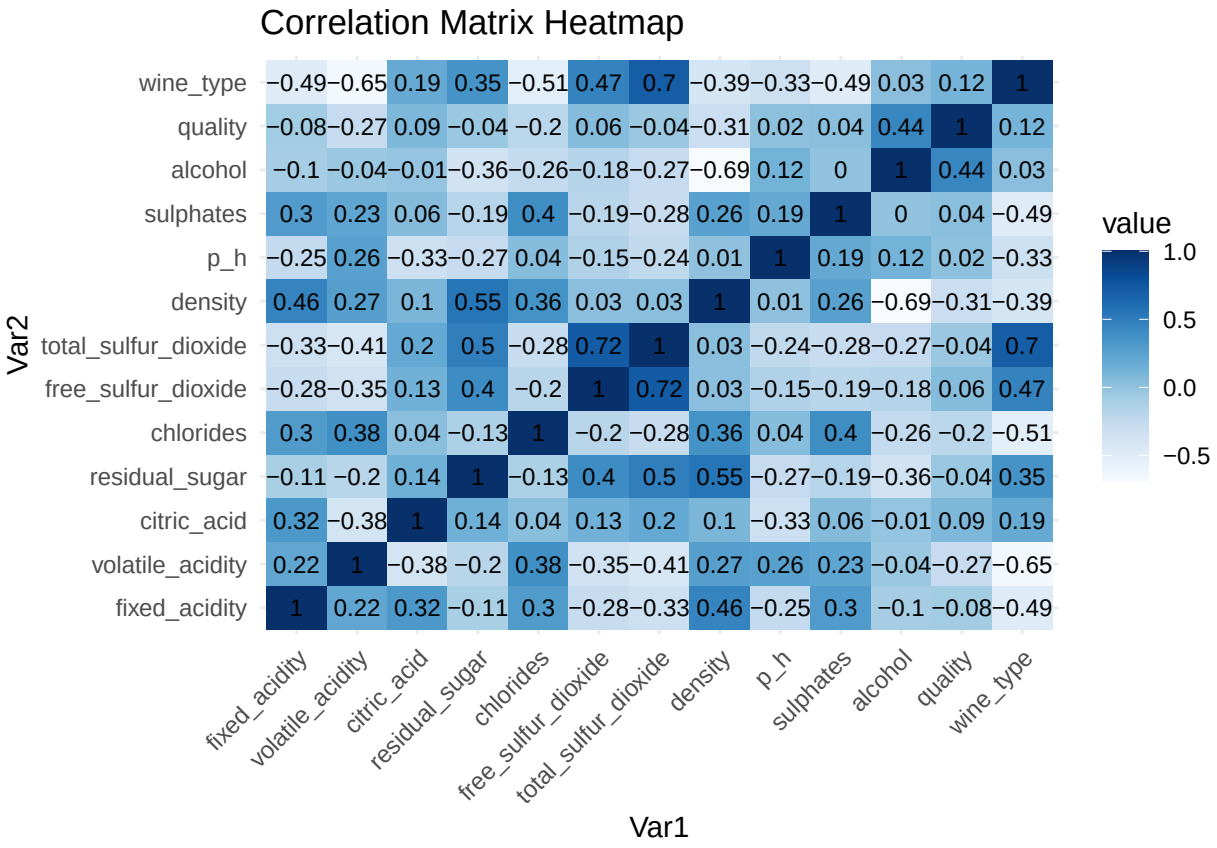
```
ggplot(combined_wines, aes(x = quality, fill = as.factor(wine_type))) +
  geom_bar(position = "dodge") +
  labs(title = "Phan phoi cua chat luong ruou vang theo loai ruou",
       x = "Quality",
       y = "Counts",
       fill = "Wine Type") +
  theme_minimal()
```



- Qua biểu đồ trên, ta dễ dàng nhận xét được, rượu vang trắng có số lượng nhiều hơn vượt trội so với rượu vang đỏ, và đồng thời chất lượng 5 và 6 cho rượu vang chiếm tỉ lệ lớn trong tập dữ liệu thu thập được.

- Mối quan hệ các biến đến chất lượng rượu thông qua correlation matrix:

```
corr_wines <- combined_wines
corr_wines$quality <- as.numeric(corr_wines$quality)
corr_matrix <- cor(corr_wines[, sapply(corr_wines, is.numeric)])
melted_corr_matrix <- melt(corr_matrix)

# Tạo biểu đồ heatmap với ma trận corr
ggplot(data = melted_corr_matrix, aes(x = Var1, y = Var2, fill = value)) +
geom_tile() + scale_fill_gradientn(colors = brewer.pal(9, "Blues")) +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3) +
  theme_minimal() + labs(title = "Correlation Matrix Heatmap") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



- Có thể dễ dàng thấy, biến quality có tương quan (correlation) mạnh đến feature **"alcohol"**, nên ta phỏng đoán đây sẽ là feature quan trọng nhất, quyết định chính nên chất lượng của rượu vang.
- Bên cạnh đó, có thể nhận xét rằng, có các biến tương quan dương và tương quan âm với target value (quality), các feature này sẽ ảnh hưởng trực tiếp và rõ rệt đến chất lượng rượu vang:
  - Tương quan dương:
    - alcohol (corr = 0.44), là nồng độ cồn, thể hiện độ mạnh của rượu

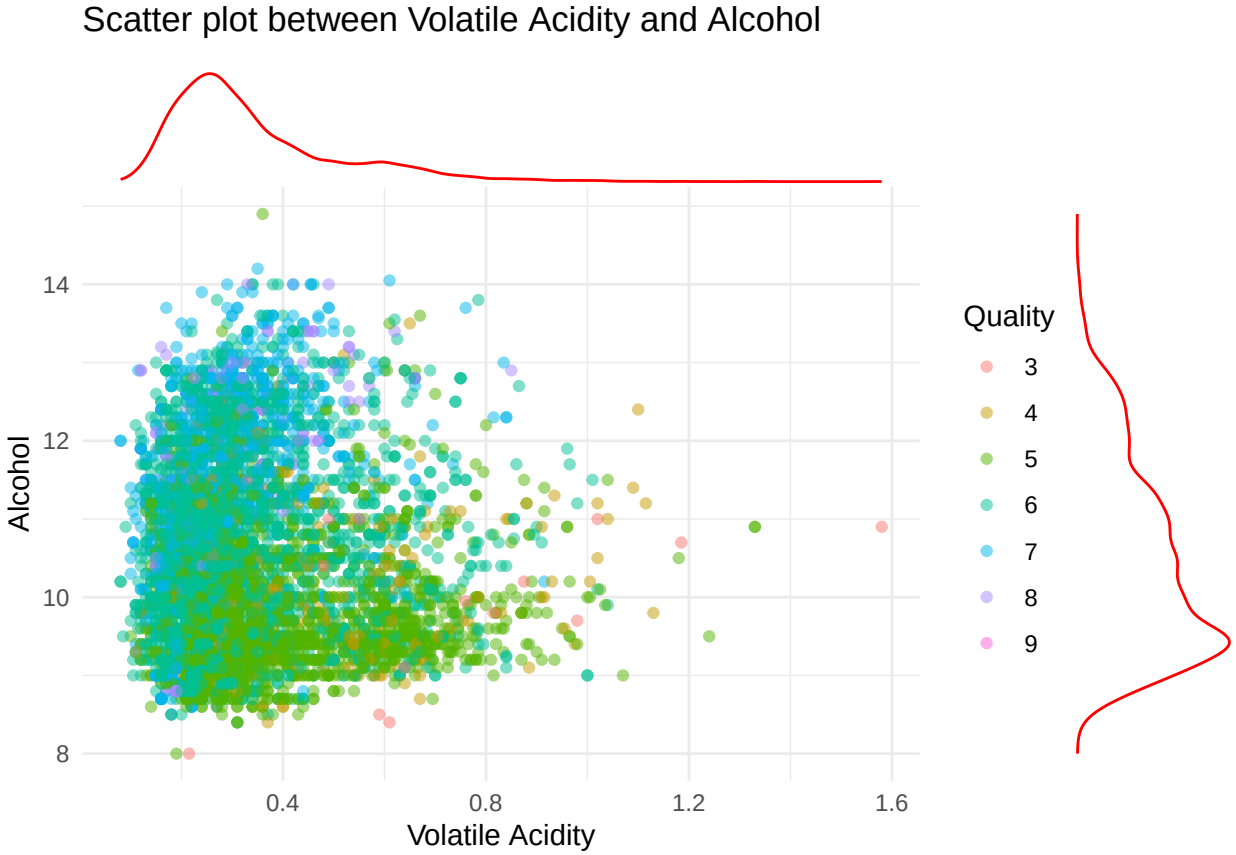
  - Tương quan âm:
    * chlorides (corr = -0.13), thể hiện vị mặn của rượu
    * volatile acidity (corr = -0.27)m là độ axit bay hơi của rượu, thể hiện mức độ chua của rượu
    * density (corr = -0.31), thể hiện độ đặc của rượu
  - Các feature còn lại có tương quan với quality không quá mạnh nên ta tạm thời chỉ xem xét các feature trên khi EDA.

- Đầu tiên, ta sẽ dùng pairplot để xem xét mối quan hệ giữa 4 biến đã đề cập trên với chất lượng rượu vang:

```
ggpairs(combined_wines, columns = c("alcohol", "chlorides",
                                    "volatile_acidity", "density"),
  aes(color = quality), title = "Pairplot for Quality", progress = FALSE,
  upper = list(continuous = wrap(ggally_cor, size = 2.1)),
)
```

- Chọn hai biến có |corr| cao nhất và vẽ biểu đồ marginal density plot để xem xét mối quan hệ giữa chúng với chất lượng rượu vang:

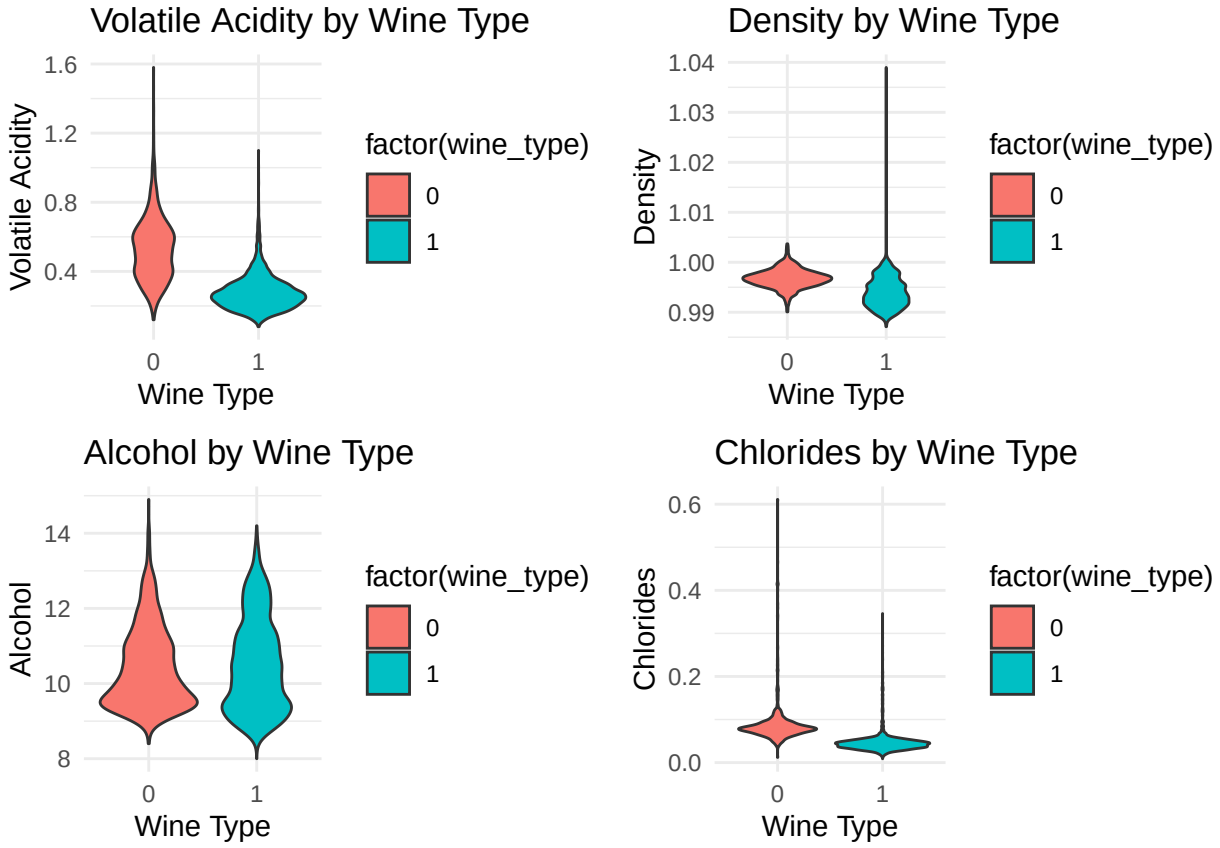
```
p <- ggplot(combined_wines, aes(x = `volatile_acidity`,
                                y = alcohol, color = quality)) +
  geom_point(alpha = 0.5) +
  labs(title = "Scatter plot between Volatile Acidity and Alcohol",
       x = "Volatile Acidity",y = "Alcohol",color = "Quality") +
  theme_minimal()
# Add marginal density plots
ggMarginal(p, type = "density", color = "red")
```



- Chất lượng rượu vang (quality) như được phân sẵn thành các cụm riêng trên biểu đồ (đặc biệt là các class 5 và 6), điều đó chứng tỏ hai biến này đóng vai trò không kém quan trọng trong việc đánh giá chất lượng cho rượu vang.
- Tiếp theo, ta sẽ xem xét các biến trên qua violin plot, để xem xét feature range của từng biến theo từng loại rượu vang:

```
p1 <- ggplot(combined_wines, aes(x = factor(wine_type), y = volatile_acidity,
                                 fill = factor(wine_type))) +
  geom_violin() + labs(x = "Wine Type", y = "Volatile Acidity") +
  ggtitle("Volatile Acidity by Wine Type") + theme_minimal()
```

```
p2 <- ggplot(combined_wines, aes(x = factor(wine_type), y = density,
                                 fill = factor(wine_type))) +
  geom_violin() + labs(x = "Wine Type", y = "Density") +
  ggtitle("Density by Wine Type") + theme_minimal()

p3 <- ggplot(combined_wines, aes(x = factor(wine_type), y = alcohol,
                                 fill = factor(wine_type))) +
  geom_violin() + labs(x = "Wine Type", y = "Alcohol") +
  ggtitle("Alcohol by Wine Type") + theme_minimal()

p4 <- ggplot(combined_wines, aes(x = factor(wine_type), y = chlorides,
                                 fill = factor(wine_type))) +
  geom_violin() + labs(x = "Wine Type", y = "Chlorides") +
  ggtitle("Chlorides by Wine Type") + theme_minimal()

grid.arrange(p1, p2, p3, p4, ncol = 2)
```



- Do hai dữ liệu rượu vang đỏ và trắng có các đặc trưng hóa học khác nhau với những khoảng giá trị khác nhau, nên ta cần phải xem xét và train mô hình cho từng loại rượu vang riêng biệt, lẫn cả khi merge lại.

- Từ đó, ta có thể rút ra kết luận rằng, mô hình và cách xử lý dữ liệu nào là phù hợp nhất cho bài toán phân loại chất lượng rượu vang, nhằm đánh giá bài toán một cách công tâm và hiệu quả nhất.

5 Feature Engineering

5.1 Relabel lại trên merge data và red, white wine data

- Ta sẽ relabel lại các class 3 và 4 thành 4, class 8 và 9 thành 8, để giảm số lượng class bị imbalance quá mức, sau đó chuyển cột quality thành kiểu dữ liệu factor.

```
combined_wines$quality <- as.character(combined_wines$quality)
combined_wines$quality[combined_wines$quality %in% c("3", "4")] <- "4"
combined_wines$quality[combined_wines$quality %in% c("8", "9")] <- "8"
combined_wines$quality <- as.factor(combined_wines$quality)

winequality_red$quality <- as.character(winequality_red$quality)
winequality_red$quality[winequality_red$quality %in% c("3", "4")] <- "4"
winequality_red$quality[winequality_red$quality %in% c("8", "9")] <- "8"
winequality_red$quality <- as.factor(winequality_red$quality)

winequality_white$quality <- as.character(winequality_white$quality)
winequality_white$quality[winequality_white$quality %in% c("3", "4")] <- "4"
winequality_white$quality[winequality_white$quality %in% c("8", "9")] <- "8"
winequality_white$quality <- as.factor(winequality_white$quality)
```

5.2 Chia tập dữ liệu thành train và test

- Dữ liệu được chia thành hai tập: train (80%) và test (20%) bằng cách sử dụng hàm `createDataPartition`.
- Ta sử dụng test size như trên là vì có quá ít data ở hai class 4 và 8, nên ta cần capture được nhiều dữ liệu nhất thuộc về các class này cho tập train.

```
set.seed(123)
split <- createDataPartition(combined_wines$quality, p = 0.80, list = FALSE)
train_data <- combined_wines[split, ]
test_data <- combined_wines[-split, ]
```

5.3 Xử lý imbalanced data bằng SMOTE

- Đầu tiên, ta cần kiểm tra lại xem các class bị imbalance ở những label nào:

```
table(train_data$quality)
```

```
##
##    4    5    6    7    8
##  197 1711 2269  864  159
```

- Chúng ta sử dụng **SMOTE** của thư viện `scutr` để cân lại dữ liệu huấn luyện, nhằm đảm bảo mỗi lớp chất lượng có số lượng mẫu tương đương nhau hoặc gần xấp xỉ với nhau.
- Như vậy, class 5 và 6 là hai class có số lượng quá vượt trội so với các class còn lại, idea ở đây là chọn 1000 làm con số sample đại diện cho mỗi class, sau đó dùng SMOTE để oversampling những class còn lại với nhau. Cuối cùng là merge những subset đã được handle lại. (chỉ cần handle ở train set)

```
# viết hàm smote_all để handle imbalanced data, với tên cột,
# và tên target cần handle, sử dụng oversample SMOTE

custom_smote_all <- function(data, target, target_name, n,
                             exclude = c()) {
  target <- as.factor(target)
  target_name <- as.character(target_name)
  target_levels <- levels(target)
  target_levels <- target_levels[!target_levels %in% exclude]

  smoted_data <- data.frame()
  for (i in target_levels) {
    subset <- dplyr::filter(data, data[[target_name]] == i)
    smoted_subset <- oversample_smote(subset, i, target_name, n)
    smoted_data <- rbind(smoted_data, smoted_subset)
  }

  # rbind with original data of excluded classes
  excluded_data <- dplyr::filter(data, data[[target_name]] %in% exclude)
  smoted_data <- rbind(smoted_data, excluded_data)

  # convert to factor
  smoted_data[[target_name]] <- as.factor(smoted_data[[target_name]])
```

```
  return(smoted_data)
}

train_data_balanced <- custom_smote_all(train_data, train_data$quality,
                                        "quality", 1000, c("5", "6"))

table(train_data_balanced$quality)
```

```
##
##    4    5    6    7    8
## 1000 1711 2269 1000 1000
```

```
test_data$quality <- as.factor(test_data$quality)
```

- Có thể thấy, data sẽ balance hơn rất nhiều. Bằng việc relabel và resample lại dữ liệu, ta đã giảm được hiện tượng imbalanced data, giúp mô hình học tốt hơn, tránh các hiện tượng bias do số lượng mẫu quá ít.

6 Phân loại sử dụng dataset merge giữa red và white wine

6.1 Logistic Regression

- Mô hình đầu tiên mà ta sử dụng là **Logistic Regression**, mô hình này thường được sử dụng trong các bài toán phân loại, đặc biệt là trong bài toán binary classification.
- Tuy nhiên, do bài toán của ta là multi-class classification, nên ta sẽ sử dụng phương pháp **Hồi Quy Multinominal Logistic** để phân loại chất lượng rượu vang.

```
# Train Logistic Regression model
logit_model <- multinom(quality ~ ., data = train_data_balanced)
```

```
## # weights:  70 (52 variable)
## initial  value 11233.876629
## iter  10 value 10591.912738
## iter  20 value 10067.063630
## iter  30 value 9019.241391
## iter  40 value 8656.287377
## iter  50 value 8644.166143
## iter  60 value 8638.529185
```

```
## iter  70 value 8619.614208
## iter  80 value 8611.976591
## iter  90 value 8609.161380
## iter 100 value 8598.304470
## final  value 8598.304470
## stopped after 100 iterations
```

```
logit_predictions <- predict(logit_model, test_data)
```

- Tiến hành đánh giá mô hình bằng cách sử dụng Confusion Matrix.

```
# Confusion Matrix
logit_cm <- confusionMatrix(logit_predictions, test_data$quality,mode = "everything")
print(logit_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7   8
##          4  19  56  23   2   0
##          5  12 215 113   9   1
##          6  14 149 358 118  14
##          7   2   1  18  16   2
##          8   2   6  55  70  22
##
## Overall Statistics
##
##                Accuracy : 0.4857
##                  95% CI : (0.4582, 0.5133)
##     No Information Rate : 0.4372
##     P-Value [Acc > NIR] : 0.0002433
##
##                   Kappa : 0.2433
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.38776   0.5035   0.6314  0.07442  0.56410
## Specificity           0.93510   0.8448   0.5959  0.97874  0.89428
## Pos Pred Value        0.19000   0.6143   0.5482  0.41026  0.14194
## Neg Pred Value        0.97494   0.7761   0.6755  0.84181  0.98511
```

```
## Precision             0.19000   0.6143   0.5482   0.41026   0.14194
## Recall                0.38776   0.5035   0.6314   0.07442   0.56410
## F1                    0.25503   0.5534   0.5869   0.12598   0.22680
## Prevalence            0.03778   0.3292   0.4372   0.16577   0.03007
## Detection Rate        0.01465   0.1658   0.2760   0.01234   0.01696
## Detection Prevalence  0.07710   0.2699   0.5035   0.03007   0.11951
## Balanced Accuracy     0.66143   0.6742   0.6136   0.52658   0.72919
```

- Độ chính xác của mô hình là khá tốt, tuy nhiên vẫn còn một số class bị dự đoán sai, đặc biệt là class 4 và 8 (kể cả khi ta đã resample lại). Ngoài ra, ta cũng cần xem xét các giá trị F1 và Recall của mô hình.
- Có thể thấy Precision rất tệ ở class 4 và 8, ngược lại với class 5 và 6, điều này có thể là do số lượng mẫu ban đầu của hai class này quá ít, nên khi oversample, model vẫn bị thiếu thông tin về hai class này.
- Với Recall, ta cũng thấy rằng, mô hình có thể dự đoán tốt class 5 và 6, nhưng lại tệ ở class 4 và 8.

6.2 Discriminant Analysis

- Mô hình tiếp theo mà ta sử dụng là **Discriminant Analysis**, mô hình này cũng thường được sử dụng trong bài toán phân loại, đặc biệt là trong bài toán multi-class classification.
- Mà cụ thể, ở đây ta sẽ sử dụng **Quadratic Discriminant Analysis** (QDA) để phân loại chất lượng rượu vang.

```r
# Train QDA model
qda_model <- qda(quality ~ ., data = train_data_balanced)
qda_predictions <- predict(qda_model, test_data)
```

- Tiến hành đánh giá mô hình bằng cách sử dụng Confusion Matrix.

```r
# Confusion Matrix
qda_cm <- confusionMatrix(qda_predictions$class, test_data$quality,mode = "everything")
print(qda_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7   8
##          4  21  55  13   1   0
```

```
##          5  12 204 121   7   0
##          6   7 112 184  50   7
##          7   2  19  93  63  12
##          8   7  37 156  94  20
##
## Overall Statistics
##
##                Accuracy : 0.3793
##                  95% CI : (0.3528, 0.4064)
##     No Information Rate : 0.4372
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1804
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.42857   0.4778   0.3245  0.29302  0.51282
## Specificity           0.94471   0.8391   0.7589  0.88355  0.76630
## Pos Pred Value        0.23333   0.5930   0.5111  0.33333  0.06369
## Neg Pred Value        0.97680   0.7660   0.5912  0.86282  0.98067
## Precision             0.23333   0.5930   0.5111  0.33333  0.06369
## Recall                0.42857   0.4778   0.3245  0.29302  0.51282
## F1                    0.30216   0.5292   0.3970  0.31188  0.11331
## Prevalence            0.03778   0.3292   0.4372  0.16577  0.03007
## Detection Rate        0.01619   0.1573   0.1419  0.04857  0.01542
## Detection Prevalence  0.06939   0.2652   0.2776  0.14572  0.24210
## Balanced Accuracy     0.68664   0.6584   0.5417  0.58829  0.63956
```

- Cũng giống như mô hình Logistic Regression, mô hình QDA cũng cho kết quả khá tốt, tuy nhiên vẫn còn một số class bị dự đoán sai, đặc biệt là class 4 và 8.
- Các chỉ số như Precision và F1, các kết quả dự đoán tệ đều rơi vào hai class 4 và 8, nguyên nhân vẫn là do số lượng mẫu ban đầu của hai class này quá ít, nên khi oversample, model vẫn bị thiếu thông tin về hai class này.
- Tuy nhiên, có thể nhận xét được rằng, độ chính xác tổng thể của QDA là thấp hơn so với Logistic Regression, điều này có thể là do QDA không phù hợp với dữ liệu của ta.

6.3 Random Forest

6.3.1 Không sử dụng Bootstrap

- Mô hình tiếp theo mà ta sử dụng đó là Random Forest, do mô hình có khả năng generalize tốt trên những tập data bị imbalanced, mà cụ thể là dữ liệu về rượu vang của ta ở đây. Không những vậy, Random Forest cũng là một mô hình có tính giải thích cao, giúp ta hiểu rõ hơn về quyết định của mô hình.
- Nhóm đã lưu ý rằng, Random Forest chỉ là kỹ thuật bagging của Decision Tree, nên sẽ chỉ tập trung vào việc train mô hình Random Forest thay vì bắt đầu từ Decision Tree.

```
rf_model <- randomForest(quality ~ ., data = train_data_balanced)
rf_predictions <- predict(rf_model, test_data)
```

- Sau đó ta có thể đánh giá mô hình bằng cách sử dụng Confusion Matrix.

```
rf_cm <- confusionMatrix(rf_predictions, test_data$quality,mode = "everything")
print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7   8
##          4  17  22   4   0   0
##          5  19 297  87   5   0
##          6  11 105 427  84  14
##          7   2   3  33 107   8
##          8   0   0  16  19  17
##
## Overall Statistics
##
##                Accuracy : 0.6669
##                  95% CI : (0.6405, 0.6926)
##     No Information Rate : 0.4372
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4941
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.34694   0.6956   0.7531   0.4977  0.43590
## Specificity           0.97917   0.8724   0.7068   0.9575  0.97218
## Pos Pred Value        0.39535   0.7279   0.6661   0.6993  0.32692
## Neg Pred Value        0.97448   0.8538   0.7866   0.9056  0.98233
## Precision             0.39535   0.7279   0.6661   0.6993  0.32692
## Recall                0.34694   0.6956   0.7531   0.4977  0.43590
## F1                    0.36957   0.7114   0.7070   0.5815  0.37363
## Prevalence            0.03778   0.3292   0.4372   0.1658  0.03007
## Detection Rate        0.01311   0.2290   0.3292   0.0825  0.01311
## Detection Prevalence  0.03315   0.3146   0.4942   0.1180  0.04009
## Balanced Accuracy     0.66305   0.7840   0.7300   0.7276  0.70404
```

- Tiếp tục xem xét các giá trị F1 và Recall của mô hình

```
print(rf_cm$byClass)
```

```
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: 4   0.3469388   0.9791667      0.3953488      0.9744817 0.3953488
## Class: 5   0.6955504   0.8724138      0.7279412      0.8537683 0.7279412
## Class: 6   0.7530864   0.7068493      0.6661466      0.7865854 0.6661466
## Class: 7   0.4976744   0.9574861      0.6993464      0.9055944 0.6993464
## Class: 8   0.4358974   0.9721781      0.3269231      0.9823293 0.3269231
##             Recall        F1 Prevalence Detection Rate Detection Prevalence
## Class: 4 0.3469388 0.3695652 0.03777949     0.01310717           0.03315343
## Class: 5 0.6955504 0.7113772 0.32922128     0.22898998           0.31457209
## Class: 6 0.7530864 0.7069536 0.43716268     0.32922128           0.49421742
## Class: 7 0.4976744 0.5815217 0.16576715     0.08249807           0.11796453
## Class: 8 0.4358974 0.3736264 0.03006939     0.01310717           0.04009252
##          Balanced Accuracy
## Class: 4         0.6630527
## Class: 5         0.7839821
## Class: 6         0.7299679
## Class: 7         0.7275803
## Class: 8         0.7040377
```

- Nhìn chung, độ chính xác ở class 4 và 8 đã có cải thiện rõ rệt so với hai mô hình trước, điều này là do Random Forest có khả năng generalize tốt hơn trên dữ liệu imbalanced, và tránh overfitting hơn so với hai mô hình trước.

6.3.2 Phương pháp Bootstrap

Trong phần này, ta sử dụng phương pháp bootstrap để cải thiện chất lượng mô hình. Phương pháp này giúp đánh giá sự ổn định và độ tin cậy của mô hình bằng cách lặp lại quá trình

huấn luyện nhiều lần trên các mẫu con của dữ liệu gốc và đo lường độ chính xác trên tập kiểm tra.

Trong bài toán này, ta sử dụng bootstrap với R = 100.

```
# hàm tính bootstrap cho randomForrest
bootstrap_accuracy <- function(data, indices) {
  d <- data[indices, ]
  rf_model <- randomForest(quality ~ ., data = d)
  predictions <- predict(rf_model, test_data)
  cm <- confusionMatrix(predictions, test_data$quality)
  return(cm$overall['Accuracy'])
}
```

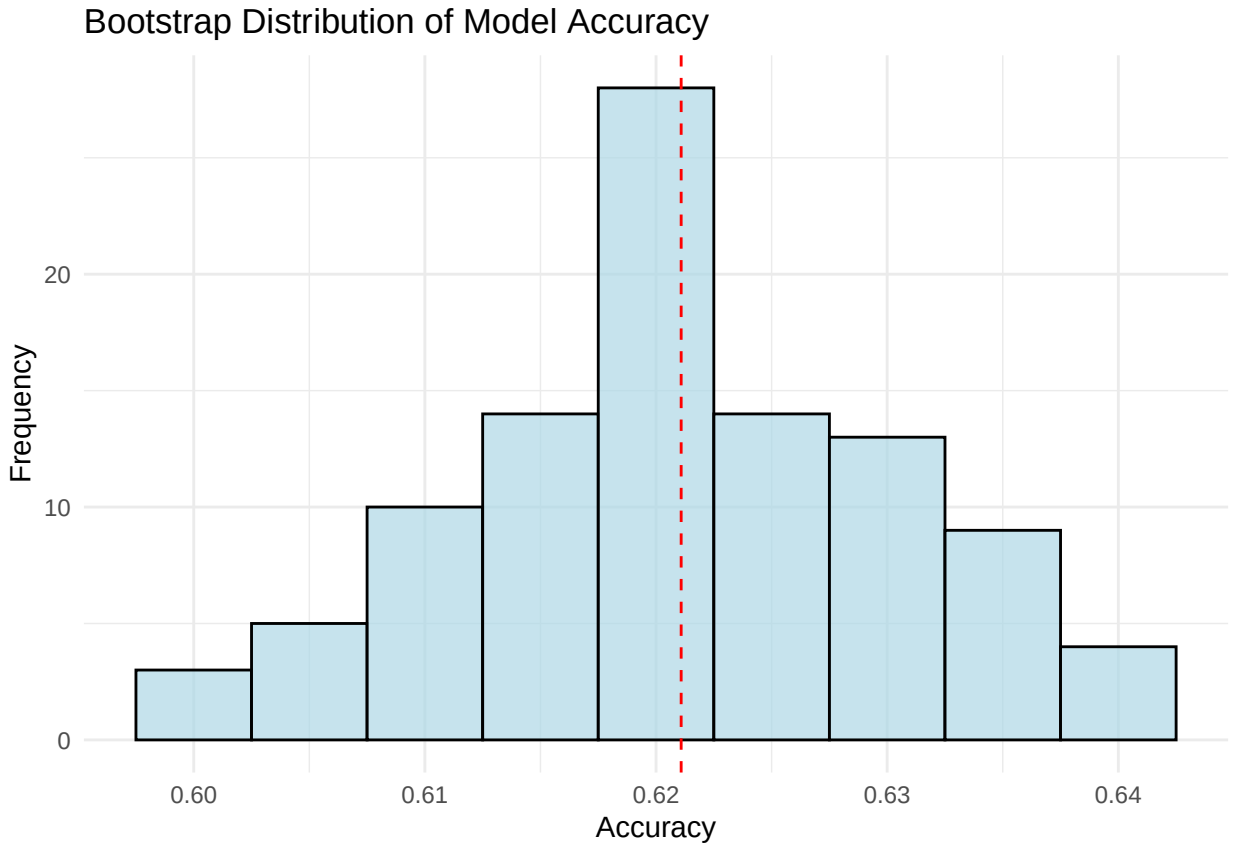
- Tiến hành apply bootstrap với R = 100.

```
# Apply bootstrap method
set.seed(123)
results <- boot(
  data = train_data_balanced,
  statistic = bootstrap_accuracy,
  R = 100
)

# Display bootstrap results
print(results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = train_data_balanced, statistic = bootstrap_accuracy,
##     R = 100)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.6669237 -0.04583655 0.009138086
```

- Ta vẽ ra đồ thị về phân phối của độ chính xác mô hình với bootstrap trên như sau:

```r
accuracy_distribution <- data.frame(Accuracy = results$t)
ggplot(accuracy_distribution, aes(x = Accuracy)) +
  geom_histogram(binwidth = 0.005, fill = 'lightblue', color = 'black',
                 alpha = 0.7) +
  geom_vline(aes(xintercept = mean(Accuracy)), color = 'red',
             linetype = 'dashed') +
  labs(title = "Bootstrap Distribution of Model Accuracy",
       x = "Accuracy",
       y = "Frequency") +
  theme_minimal()
```

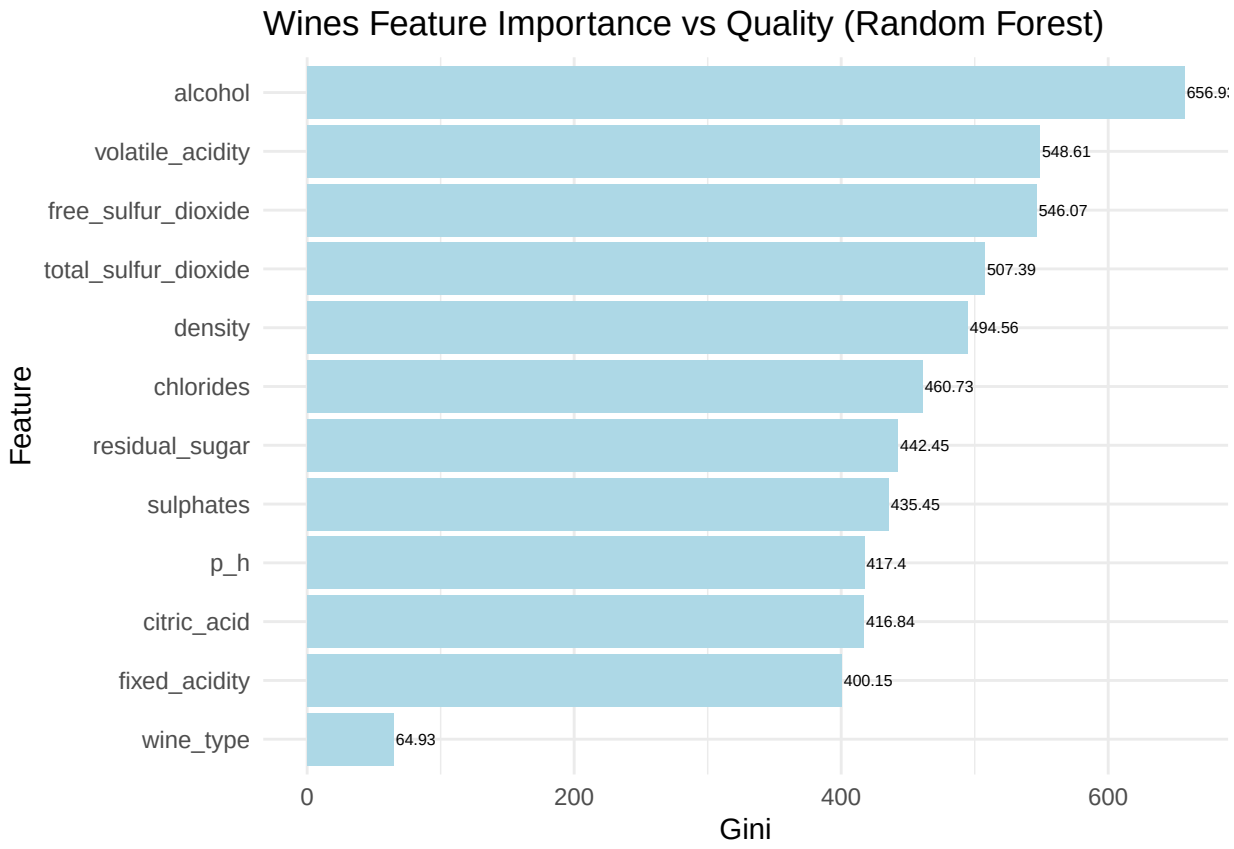


- Kết quả của phương pháp bootstrap cho ta thấy phân phối của độ chính xác, giúp ta hiểu rõ hơn về sự biến thiên và độ tin cậy của mô hình. Từ đó ta có thể nhận xét rằng, độ chính xác của mô hình là rất ổn định từ khoảng 0.60 tới 0.66.

6.3.3 Đánh giá tầm quan trọng của các feature với cột quality

Ta tiếp tục đánh giá feature importance bằng cách so sánh Gini của từng feature với target value là quality của mô hình RF vừa được train phía trên:

```
rf_importance_df <- as.data.frame(importance(rf_model))
rf_importance_df$Feature <- rownames(rf_importance_df)
rf_importance_plot <- ggplot(data = rf_importance_df,
  aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill="lightblue") +
  geom_text(aes(label = round(MeanDecreaseGini, 2)), hjust = -0.05, size = 2) +
  coord_flip() +
  labs(title = "Wines Feature Importance vs Quality (Random Forest)",
       x = "Feature", y = "Gini") +
  theme_minimal()
rf_importance_plot
```



- Như vậy, rượu có độ cao, độ axit bay hơi, độ axit cố định, nồng độ cồn, độ pH là những feature quan trọng nhất ảnh hưởng đến chất lượng rượu vang.
- Có thể nhận thấy, hai feature rất tương quan nhau như đã đề cập ở phần EDA, đó là **alcohol** và **volatile.acidity**, cũng nắm Feature Importance cao nhất trong mô hình Random Forest. Điều này chứng tỏ rằng, nồng độ cồn và độ axit bay hơi là hai yếu tố quan trọng nhất ảnh hưởng đến chất lượng rượu vang, và phải đi kèm với nhau.
- Vì mô hình Random Forest có khả năng giải thích tốt, nên việc sử dụng mô hình này để phân loại chất lượng rượu vang là một lựa chọn tốt.

⇒ **Kết luận:** Ta có thể đề xuất cho nhà sản xuất rượu vang rằng, để cải thiện chất lượng sản phẩm, họ cần chú ý nhất đến nồng độ cồn và độ axit bay hơi trong quá trình sản xuất, để đảm bảo chất lượng sản phẩm cuối cùng, vì đây sẽ là hai yếu tố quyết định chất lượng rượu vang. Bên cạnh đó, việc kiểm soát các yếu tố khác như độ pH, hàm lượng đường, độ đặc cũng rất quan trọng.

6.4 Naive Bayes

- Naive Bayes là một trong những thuật toán phân loại dựa trên xác suất Bayes, rất hiệu quả và đơn giản, đặc biệt đối với các bài toán phân loại có số lượng mẫu nhỏ hoặc dữ liệu bị thiếu.
- Train model Naive Bayes với phương pháp Laplace Smoothing, để model tự thêm 1 vào các dữ liệu bị thiếu tương ứng với từng class. Tránh việc gây ra các xác suất Bayes bằng 0, điều này sẽ giảm được hiện tượng overfitting.

```
nb_model <- naiveBayes(
  quality ~ .,
  data = train_data_balanced,
  laplace = 1
)

nb_predictions <- predict(nb_model, test_data)
```

- Đánh giá mô hình bằng Confusion matrix:

```
nb_cm <- confusionMatrix(nb_predictions, test_data$quality)
nb_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7   8
##          4   9  62  40   5   1
##          5  11 184 129  20   1
##          6  18 103 149  37  11
##          7   1   3   8  16   0
##          8  10  75 241 137  26
##
## Overall Statistics
##
##                Accuracy : 0.2961
##                  95% CI : (0.2713, 0.3217)
```

```
##     No Information Rate : 0.4372
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1055
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.183673   0.4309   0.2628  0.07442  0.66667
## Specificity          0.913462   0.8149   0.7685  0.98891  0.63196
## Pos Pred Value       0.076923   0.5333   0.4686  0.57143  0.05317
## Neg Pred Value       0.966102   0.7447   0.5730  0.84318  0.98391
## Prevalence           0.037779   0.3292   0.4372  0.16577  0.03007
## Detection Rate       0.006939   0.1419   0.1149  0.01234  0.02005
## Detection Prevalence 0.090208   0.2660   0.2452  0.02159  0.37702
## Balanced Accuracy    0.548568   0.6229   0.5156  0.53166  0.64931
```

- Nhìn chung, độ chính xác của Naive Bayes là khá tệ khi so với Random Forest, vẫn bị confuse khá nhiều giữa các class.
- Mô hình có độ chính xác của từng class khá ổn định nhau (0.5-0.6), nhưng độ chính xác tổng thể của mô hình là khá thấp, do đó ta vẫn giữ quyết định chọn Random Forest là mô hình tốt nhất cho bài toán phân loại chất lượng rượu vang.

6.5 SVM (Further work)

- Tiếp theo, ta sẽ sử dụng mô hình SVM để phân loại chất lượng rượu vang.

```
svm_model <- svm(quality ~ ., data = train_data_balanced)
svm_predictions <- predict(svm_model, test_data)
svm_cm <- confusionMatrix(svm_predictions, test_data$quality)
print(svm_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   4   5   6   7   8
##          4  20  52  14   2   0
##          5  14 240 116   6   0
##          6  14 128 371 109  16
##          7   1   4  18  43   7
```

```
##          8   0   3  48  55  16
##
## Overall Statistics
##
##                Accuracy : 0.532
##                  95% CI : (0.5044, 0.5594)
##     No Information Rate : 0.4372
##     P-Value [Acc > NIR] : 4.428e-12
##
##                   Kappa : 0.3065
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.40816   0.5621   0.6543  0.20000  0.41026
## Specificity           0.94551   0.8437   0.6342  0.97227  0.91574
## Pos Pred Value        0.22727   0.6383   0.5815  0.58904  0.13115
## Neg Pred Value        0.97601   0.7970   0.7026  0.85948  0.98043
## Prevalence            0.03778   0.3292   0.4372  0.16577  0.03007
## Detection Rate        0.01542   0.1850   0.2860  0.03315  0.01234
## Detection Prevalence  0.06785   0.2899   0.4919  0.05628  0.09406
## Balanced Accuracy     0.67684   0.7029   0.6443  0.58614  0.66300
```

- Cũng như Naive Bayes, mô hình SVM cho kết quả với độ chính xác khoảng 0.5-0.6.
- Tuy nhiên, mô hình SVM có thể cải thiện được nhiều so với Naive Bayes, nhưng vẫn không bằng Random Forest.

7 Phân loại sử dụng model riêng cho red và white

- Trong phần này, ta sẽ xem xét hiệu suất của các mô hình Random Forest khi áp dụng cho từng loại rượu vang riêng biệt, đó là red wine và white wine.

```r
# Chia dữ liệu thành tập huấn luyện
# và kiểm tra riêng biệt cho từng loại rượu
set.seed(123)
trainIndexRed <- createDataPartition(winequality_red$quality,
                                     p = .8, list = FALSE)
trainIndexWhite <- createDataPartition(winequality_white$quality,
                                     p = .8, list = FALSE)
```

```
trainDataRed <- winequality_red[trainIndexRed, ]
testDataRed <- winequality_red[-trainIndexRed, ]

trainDataWhite <- winequality_white[trainIndexWhite, ]
testDataWhite <- winequality_white[-trainIndexWhite, ]

# set quality as factor
trainDataRed$quality <- as.factor(trainDataRed$quality)
testDataRed$quality <- as.factor(testDataRed$quality)

trainDataWhite$quality <- as.factor(trainDataWhite$quality)
testDataWhite$quality <- as.factor(testDataWhite$quality)
```

- Sau đó sử dụng hàm SMOTE custom để cân bằng lại dữ liệu cho từng loại rượu vang.

```
trainDataRed <- custom_smote_all(trainDataRed, trainDataRed$quality,
                                          "quality", 1000, c("5", "6"))

trainDataWhite <- custom_smote_all(trainDataWhite, trainDataWhite$quality,
                                            "quality", 1000, c("5", "6"))
```

7.0.1 Áp dụng mô hình Random Forest cho red và white wine

- Việc train mô hình Random Forest cho red và white wine tương tự như trên, sau đó so sánh kết quả giữa hai loại rượu vang.

```
rf_model_red <- randomForest(quality ~ ., data = trainDataRed)
rf_predictions_red <- predict(rf_model_red, testDataRed)
rf_cm_red <- confusionMatrix(rf_predictions_red, testDataRed$quality)

rf_model_white <- randomForest(quality ~ ., data = trainDataWhite)
rf_predictions_white <- predict(rf_model_white, testDataWhite)
rf_cm_white <- confusionMatrix(rf_predictions_white, testDataWhite$quality)
```

7.0.2 Comparison plot giữa red và white wine

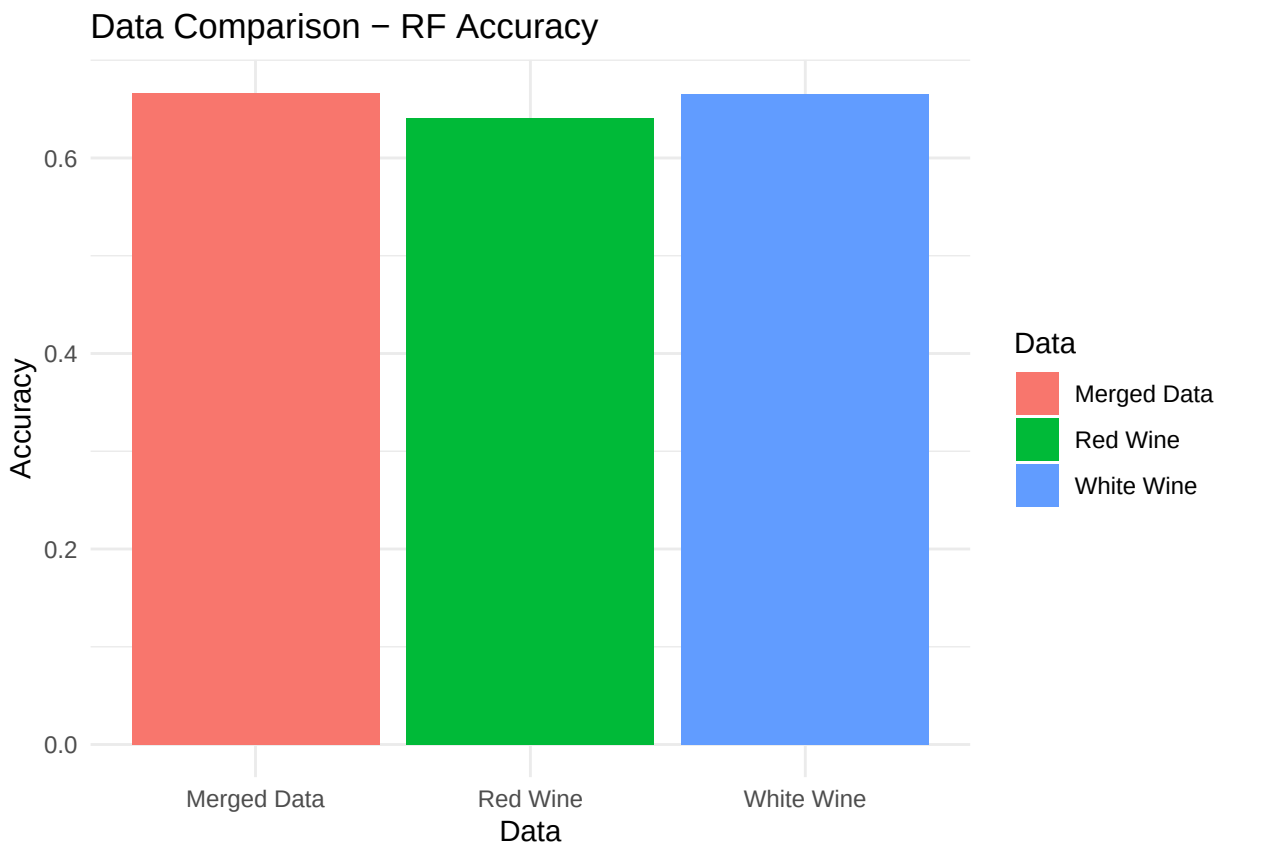
- Vẽ biểu đồ so sánh độ chính xác giữa red và white wine:

```
data_comparisons <- data.frame(
  Data = c("Red Wine", "White Wine", "Merged Data"),
  Accuracy = c(rf_cm_red$overall['Accuracy'],
```

```
                rf_cm_white$overall['Accuracy'],
                rf_cm$overall['Accuracy'])
)

kable(data_comparisons)
```

Data	Accuracy
Red Wine	0.6403785
White Wine	0.6656442
Merged Data	0.6669237

```
ggplot(data_comparisons, aes(x = Data, y = Accuracy, fill = Data)) +
  geom_bar(stat = "identity") +
  labs(title = "Data Comparison - RF Accuracy", x = "Data", y = "Accuracy") +
  theme_minimal()
```



- Như vậy, ta có thể thấy rằng mô hình Random Forest dù áp dụng cho cách xử lý dữ liệu khác nhau (merge, red wine, white wine) đều cho kết quả tương đương nhau, với độ chính xác khoảng 0.6-0.7.

- Tuy nhiên, mô hình Random Forest cho white wine có độ chính xác cao hơn so với red wine, điều này có thể do dữ liệu white wine có số lượng mẫu nhiều hơn, nên mô hình dễ generalize hơn.

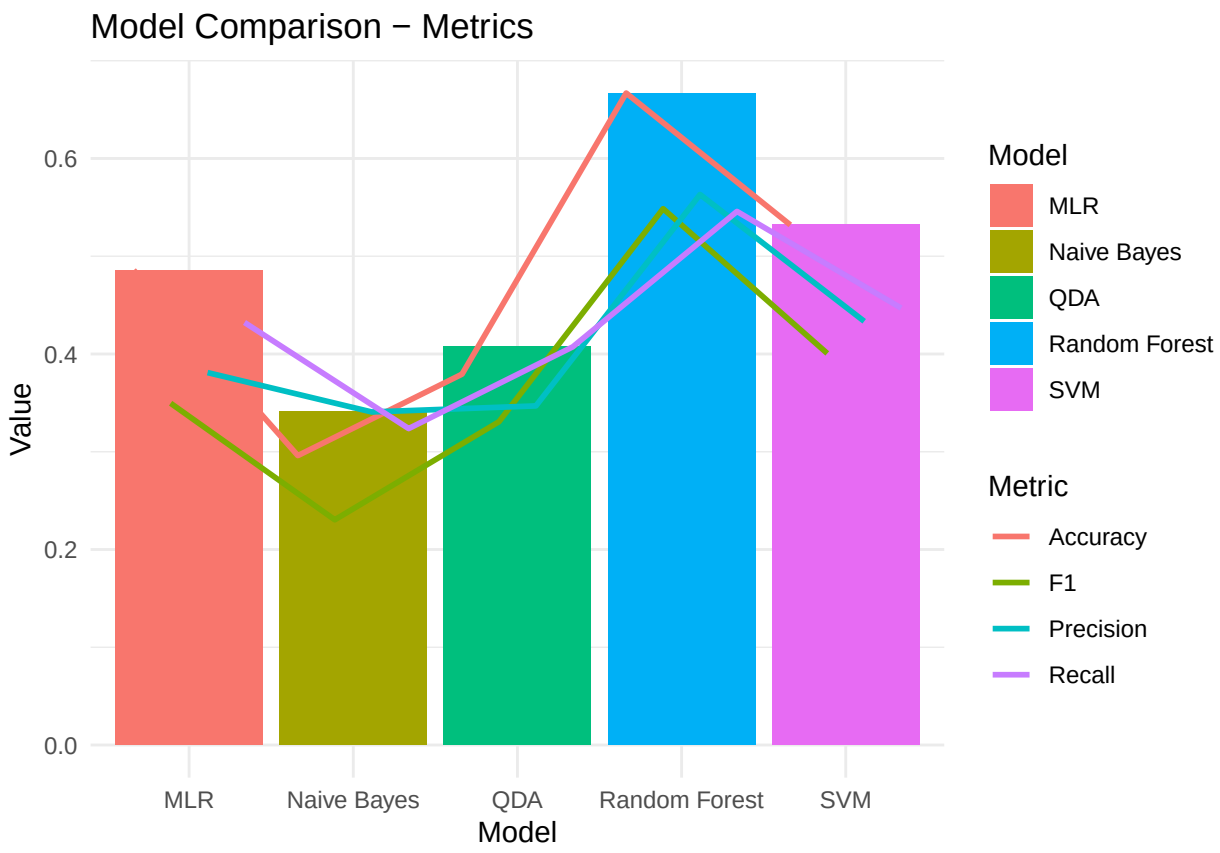
8 So sánh và lựa chọn mô hình tốt nhất

- Để hiểu rõ hơn về hiệu suất của các mô hình học máy khác nhau, chúng ta cần so sánh các chỉ số như độ chính xác (Accuracy), F1, độ chính xác (Precision) và độ nhạy (Recall). Bước này giúp xác định mô hình nào có kết quả dự đoán chất lượng rượu vang tốt nhất dựa trên các tiêu chí đánh giá khác nhau.
- Biểu đồ so sánh dưới đây hiển thị các chỉ số của mô hình đã train phía trên. Chúng ta sẽ dựa vào biểu đồ này và xem xét độ chính xác, F1, Precision và Recall của từng mô hình để đưa ra kết luận cuối cùng.

```
# Assuming 'cm_list' is a list containing all confusion matrices
cm_list <- list(logit_cm, qda_cm, rf_cm, nb_cm, svm_cm)
models <- c("MLR", "QDA",
            "Random Forest", "Naive Bayes", "SVM")

accuracy <- numeric(length(models))
f1_scores <- numeric(length(models))
precisions <- numeric(length(models))
recalls <- numeric(length(models))

# Calculate the metrics for each model
for (i in 1:length(cm_list)) {
  cm <- cm_list[[i]]$byClass
  accuracy[i] <- cm_list[[i]]$overall['Accuracy']
  f1_scores[i] <- ifelse(is.na(mean(cm[, "F1"], na.rm = TRUE)),
                         0, mean(cm[, "F1"], na.rm = TRUE))
  precisions[i] <- ifelse(is.na(mean(cm[, "Precision"], na.rm = TRUE)),
                          0, mean(cm[, "Precision"], na.rm = TRUE))
  recalls[i] <- ifelse(is.na(mean(cm[, "Recall"], na.rm = TRUE)),
                       0, mean(cm[, "Recall"], na.rm = TRUE))
}

# Create the data frame
model_comparisons2 <- data.frame(
  Model = models, Accuracy = accuracy,
  F1 = f1_scores, Precision = precisions, Recall = recalls
)

kable(model_comparisons2)
```

Model	Accuracy	F1	Precision	Recall
MLR	0.4857363	0.3496235	0.3809433	0.4322365
QDA	0.3793369	0.3307033	0.3468991	0.4073364
Random Forest	0.6669237	0.5486088	0.5631412	0.5458295
Naive Bayes	0.2960678	0.2304026	0.3406816	0.3236917
SVM	0.5319969	0.4005731	0.4334528	0.4469603

```
model_comparisons2_long <- model_comparisons2 %>%
  pivot_longer(cols = c("Accuracy", "F1", "Precision", "Recall"),
               names_to = "Metric", values_to = "Value")

ggplot(model_comparisons2_long, aes(x = Model, y = Value, fill = Model)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_line(aes(group = Metric, color = Metric),
            position = position_dodge(width = 0.9), linewidth=1) +
  labs(title = "Model Comparison - Metrics", x = "Model", y = "Value") +
  theme_minimal()
```



- Từ đó, ta có thể dễ dàng chọn được **Random Forest** làm mô hình thích hợp nhất cho bài toán phân loại chất lượng rượu vang, với độ chính xác cao nhất.

- Ngoài ra, mô hình **Random Forest** cũng rất phù hợp với dữ liệu imbalanced, và có khả năng generalize tốt trên cả dữ liệu merge và dữ liệu riêng lẻ.

9 Kết luận

- Các mô hình khác nhau được sử dụng để phân loại chất lượng rượu vang dựa trên các đặc trưng hóa học của chúng, bao gồm thông tin về hai loại rượu vang (đỏ và trắng) với các đặc trưng như độ pH, hàm lượng đường, và độ cồn. Mà trong đó, mô hình **"Random Forest"** đạt được độ chính xác tổng thể khá tốt sau khi được train trên tập dữ liệu đã được xử lý bằng kỹ thuật SMOTE để cân bằng số lượng mẫu giữa các lớp chất lượng rượu vang.
- Đề xuất cho nhà sản xuất rượu vang: nếu muốn cải thiện chất lượng rượu vang, họ nên tập trung vào việc kiểm soát độ cồn và độ axit bay hơi trong quá trình sản xuất, vì đây là hai yếu tố quan trọng nhất ảnh hưởng đến chất lượng rượu vang. Ngoài ra, việc kiểm soát các yếu tố khác như độ pH, hàm lượng đường, độ đặc cũng rất quan trọng.
- Tuy nhiên, model vẫn gặp khó khăn trong việc phân loại các lớp chất lượng rất thấp dù đã được relabel và resample lại. Phương pháp bootstrap với 100 lần lặp lại cho thấy độ chính xác của mô hình dao động trong khoảng từ 0.60 đến 0.66, chứng tỏ mô hình có sự ổn định và độ tin cậy khá cao.