title: "Wine Quality Analysis"

author: "Siyuan Zhang"

date: "2025-12-19"

###DATA CHECK AND EDA

##Initialize environment and load data

#nitialize environment and organize data, tidy up data formats for subsequent

```
library("readxl")

library("dplyr")

library("data.table")

winequality<- fread("winequality-red.csv", sep = ";")

library("tidyverse")

glimpse(df)
```

Result:

```
Rows: 1,599
Columns: 12
$ fixed.acidity        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8, 7.5, …
$ volatile.acidity     <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, 0.600, 0.…
$ citric.acid          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00, 0.0…
$ residual.sugar       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.1, 1…
$ chlorides            <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.069, 0.…
$ free.sulfur.dioxide  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15, 17, 16, …
$ total.sulfur.dioxide <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 102, 5…
$ density              <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978, 0.9…
$ pH                   <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39, 3.3…
$ sulphates            <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47, 0.5…
$ alcohol              <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9.5, 10.5,…
$ quality              <fct> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 5, 7, …
```

Data preprocessing

#Obtain summaries of numeric variables and missing value situations

```
names(df) <- gsub("\\.", " ", names(df))

library(tidyverse)

library(skimr)

library(patchwork)
```

cat("========Data Type========")

summary(df)

Result:

```
> summary(df)
 fixed acidity   volatile acidity  citric acid    residual sugar
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
   chlorides       free sulfur dioxide total sulfur dioxide    density
 Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
 1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
 Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
 Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
 3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
 Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
      pH          sulphates         alcohol      quality
 Min.   :2.740   Min.   :0.3300   Min.   : 8.40   3: 10
 1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   4: 53
 Median :3.310   Median :0.6200   Median :10.20   5:681
 Mean   :3.311   Mean   :0.6581   Mean   :10.42   6:638
 3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   7:199
 Max.   :4.010   Max.   :2.0000   Max.   :14.90   8: 18
```

Missing value inspection

#Count the number of missing values in each column to ensure data integrity

```{r}

cat("======== Missing Value Inspection ========")

print(colSums(is.na(df)))

Result:

```
> print(colSums(is.na(df)))
       fixed acidity   volatile acidity   citric acid
                   0   0   0
      residual sugar   chlorides free sulfur dioxide
                   0   0   0
total sulfur dioxide   density   pH
                   0   0   0
           sulphates   alcohol   quality
                   0   0   0
```

*There are no missing values in the entire data frame df. This wine-quality dataset is complete and
```

intact in its current dimensions; no imputation or deletion of missing values is needed downstream.

skimr::skim(df)

```
> skimr::skim(df)
── Data Summary ────────────────────────
                           Values
Name                       df
Number of rows             1599
Number of columns          12
Key                        NULL
──────────────────────────
Column type frequency:
  factor                   1
  numeric                  11
──────────────────────────
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique
1 quality               0             1 FALSE          6
  top_counts
1 5: 681, 6: 638, 7: 199, 4: 53
```

```
── Variable type: numeric ──────────────────────────────────────────────────────
   skim_variable        n_missing complete_rate    mean       sd    p0    p25
 1 fixed acidity                0             1  8.32    1.74    4.6    7.1
 2 volatile acidity             0             1  0.528   0.179   0.12   0.39
 3 citric acid                  0             1  0.271   0.195   0      0.09
 4 residual sugar               0             1  2.54    1.41    0.9    1.9
 5 chlorides                    0             1  0.0875  0.0471  0.012  0.07
 6 free sulfur dioxide          0             1 15.9    10.5     1      7
 7 total sulfur dioxide         0             1 46.5    32.9     6     22
 8 density                      0             1  0.997   0.00189 0.990  0.996
 9 pH                           0             1  3.31    0.154   2.74   3.21
10 sulphates                    0             1  0.658   0.170   0.33   0.55
11 alcohol                      0             1 10.4     1.07    8.4    9.5
      p50    p75    p100 hist
 1  7.9    9.2    15.9   ▂▇▃▁▁
 2  0.52   0.64    1.58  ▃▇▃▁▁
 3  0.26   0.42    1     ▇▆▅▃▂
 4  2.2    2.6    15.5   ▇▁▁▁▁
 5  0.079  0.09    0.611 ▇▁▁▁▁
 6 14     21      72     ▇▆▂▁▁
 7 38     62     289     ▇▆▂▁▁
 8  0.997  0.998   1.00  ▁▂▅▇▂
 9  3.31   3.4     4.01  ▁▃▇▃▁
10  0.62   0.73    2     ▇▅▂▁▁
11 10.2   11.1    14.9   ▇▅▂▁▁
```

*The data are complete with no missing values and contain a moderate sample size (n = 1,599).

Quality ratings cluster around 5 and 6, indicating that the majority of the wines are of medium quality.

The distributions of several chemical components reveal potential outliers and skewness that should be addressed during preprocessing.

Alcohol content, volatile acidity, and residual sugar are likely to be key features influencing quality.

Response variable analysis

#Analyze the distribution of the target variable (wine quality)

```
cat("==================EDA==================")

cat("==========Response Variable==========")

df$quality <- as.numeric(as.character(df$quality))

p_hist_q <- ggplot(df, aes(quality)) +

geom_bar(fill = "orange", width = 0.7) +

scale_x_continuous(breaks = min(df$quality):max(df$quality)) +

labs(title = "Quality ", subtitle = " Output variable based on sensory data ") +

theme_minimal(base_size = 14)

p_box_q <- ggplot(df, aes(y = quality)) +

geom_boxplot(fill = "orange", alpha = 0.6, outlier.color = "red", width = 0.25) +

labs(title = "Quality ", subtitle = " Outliers ") +

theme_minimal(base_size = 14)


print(p_hist_q)

print(p_box_q)
```

#Most quality ratings fall between 5 and 7,#

#accounting for roughly 70–80 % of the scores, #

#which aligns with the typical pattern of wine sensory evaluation.#

Explanatory variable distribution analysis

#Identify outliers and the natural distribution of the data to provide a basis for subsequent processing

#(Since most regions have strict regulations defining wine mainly in terms of acidity and alcohol content, while other variables serve primarily as references for wine taste and quality, the following analysis separately examines the distributions of acidity and alcohol.)

```
cat("==========Explanatory Variable==========")

p_alcohol <- ggplot(df, aes(x = alcohol)) +

geom_histogram(aes(y = ..density..), bins = 20, fill = "orange", alpha = 0.5) +

geom_density(color = "darkorange", linewidth = 1) +

geom_vline(xintercept = c(8, 22), linetype = "dashed", color = "red") +

labs(title = "Alcohol content distribution

(extreme values marked: <8 % and >22 %)", x = "Alcohol content %", y = "Density") +

theme_minimal(base_size = 14)

print(p_alcohol)

p_volatile_acidity <-ggplot(df, aes(x = `volatile acidity`))+

geom_histogram(aes(y = after_stat(density)),bins = 20,fill = "orange",alpha = 0.5)+

geom_density(color = "darkorange", linewidth = 1)+

geom_vline(xintercept = c(0, 2), linetype = "dashed", color = "red")+

labs(title = "Distribution of volatile acidity",x = "Volatile acidity",y = "Density")+

theme_minimal(base_size = 14)

print(p_volatile_acidity )

p_fixed_acidity <-ggplot(df, aes(x = `fixed acidity`))+

geom_histogram(aes(y = after_stat(density)),bins = 20,fill = "orange",alpha = 0.5)+

geom_density(color = "darkorange", linewidth = 1)+

geom_vline(xintercept = c(0, 20), linetype = "dashed", color = "red")+

labs(title = "Distribution of fixed acidity",x = "Fixed acidity",y = "Density")+
```

```
theme_minimal(base_size = 14)

print(p_fixed_acidity)

p_citric_acid <-ggplot(df, aes(x = `citric acid`))+

geom_histogram(aes(y = after_stat(density)),bins = 20,fill = "orange",alpha = 0.5)+

geom_density(color = "darkorange", linewidth = 1)+geom_vline(xintercept = c(0, 1), linetype = "dashed", color = "red")+

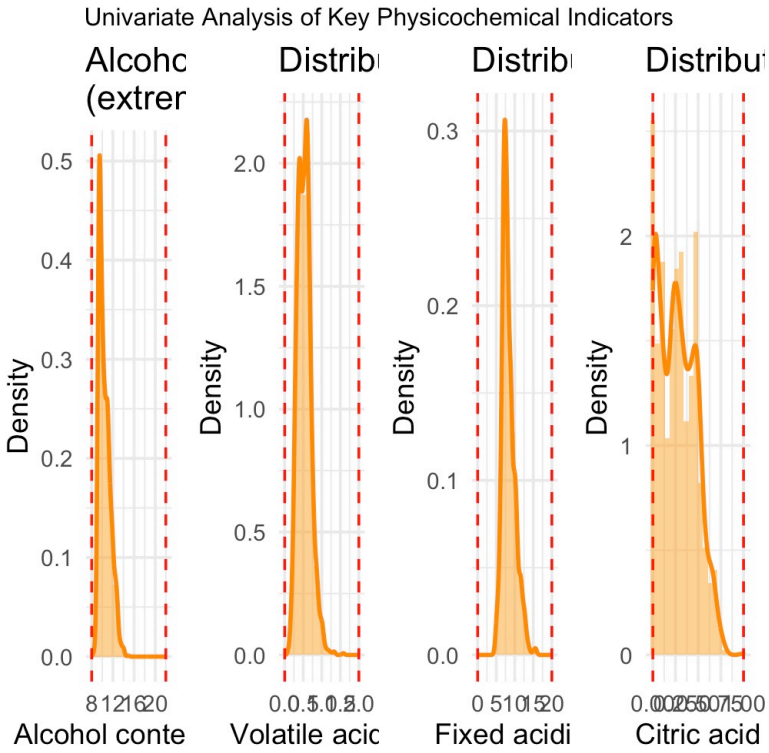
labs(title = "Distribution of citric acid",x = "Citric acid",y = "Density")+

theme_minimal(base_size = 14)

print(p_citric_acid)

library("gridExtra")

grid.arrange(p_alcohol, p_volatile_acidity,p_fixed_acidity,p_citric_acid ,ncol = 4,top = "Univariate Analysis of Key Physicochemical Indicators")
```



*This univariate analysis plot is intended to illustrate the distribution characteristics of four key physicochemical indicators in the wine dataset. It helps us understand, for each indicator:

1.Central tendency: the location of the peak, reflecting where the data are concentrated (e.g., alcohol content centers around 12%).

2.Dispersion: the range and shape of the distribution, showing how spread out the values are (e.g., citric acid is more dispersed, whereas volatile acidity is more concentrated).

3.Distribution shape: whether the distribution is unimodal or multimodal and whether it is symmetric, revealing the underlying structure of the data (e.g., the multimodality of citric acid may suggest the presence of distinct sub-groups).

Multivariate correlation analysis

#Visualize correlations among variables

```
library("RColorBrewer")

library("corrplot")

library("ggcorrplot")

corrplot(cor_matrix,method = "color", type = "full",

col = rev(brewer.pal(11, "YlOrRd")),

tl.cex = 0.8,

tl.srt = 90,

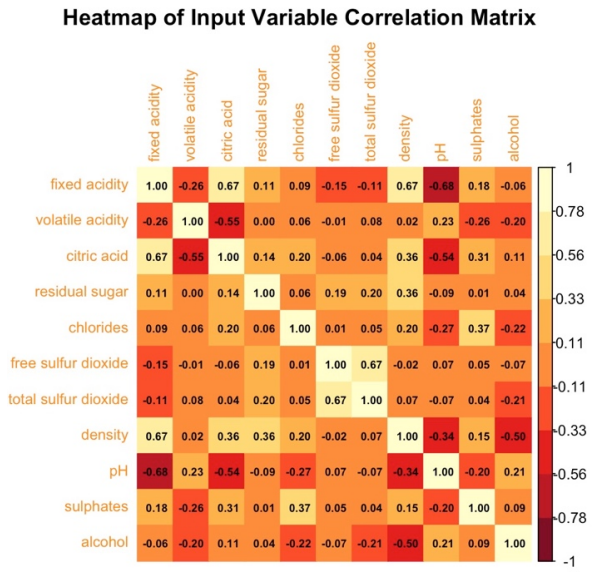
tl.col="darkorange",

addCoef.col = "black",

number.cex = 0.6,

title = "Heatmap of Input Variable Correlation Matrix",

mar = c(0, 0, 2, 0))
```



Strong positive correlations

citric acid ↔ fixed acidity: 0.67

density ↔ fixed acidity: 0.67

citric acid ↔ density: 0.36

free sulfur dioxide ↔ total sulfur dioxide: 0.67

Strong negative correlations

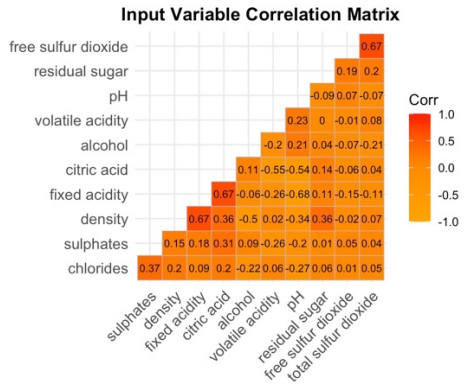
fixed acidity ↔ pH: –0.68 (higher acidity → lower pH)

pH ↔ citric acid: –0.54

pH ↔ density: –0.34

alcohol ↔ density: –0.50

```
p_corr <- ggcorrplot(cor_matrix, hc.order = TRUE,
  type = "lower",
    lab = TRUE,
    lab_size = 3,
    colors = c("YlOrRd"),
    title = " Input Variable Correlation Matrix ",
    ggtheme = theme_minimal(base_size = 12))
theme(plot.title = element_text(hjust = 0.5, face = "bold"))
print(p_corr)
```



Pairwise scatterplot matrix

#In-depth analysis of relationships among variables and quality grouping patterns

```
library(GGally)

library(ggplot2)

cont_var <- c(1:11)

df11 <- df[ , ..cont_var]

var_lab <- c("fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxide","total sulfur dioxide","density","pH","sulphates","alcohol")

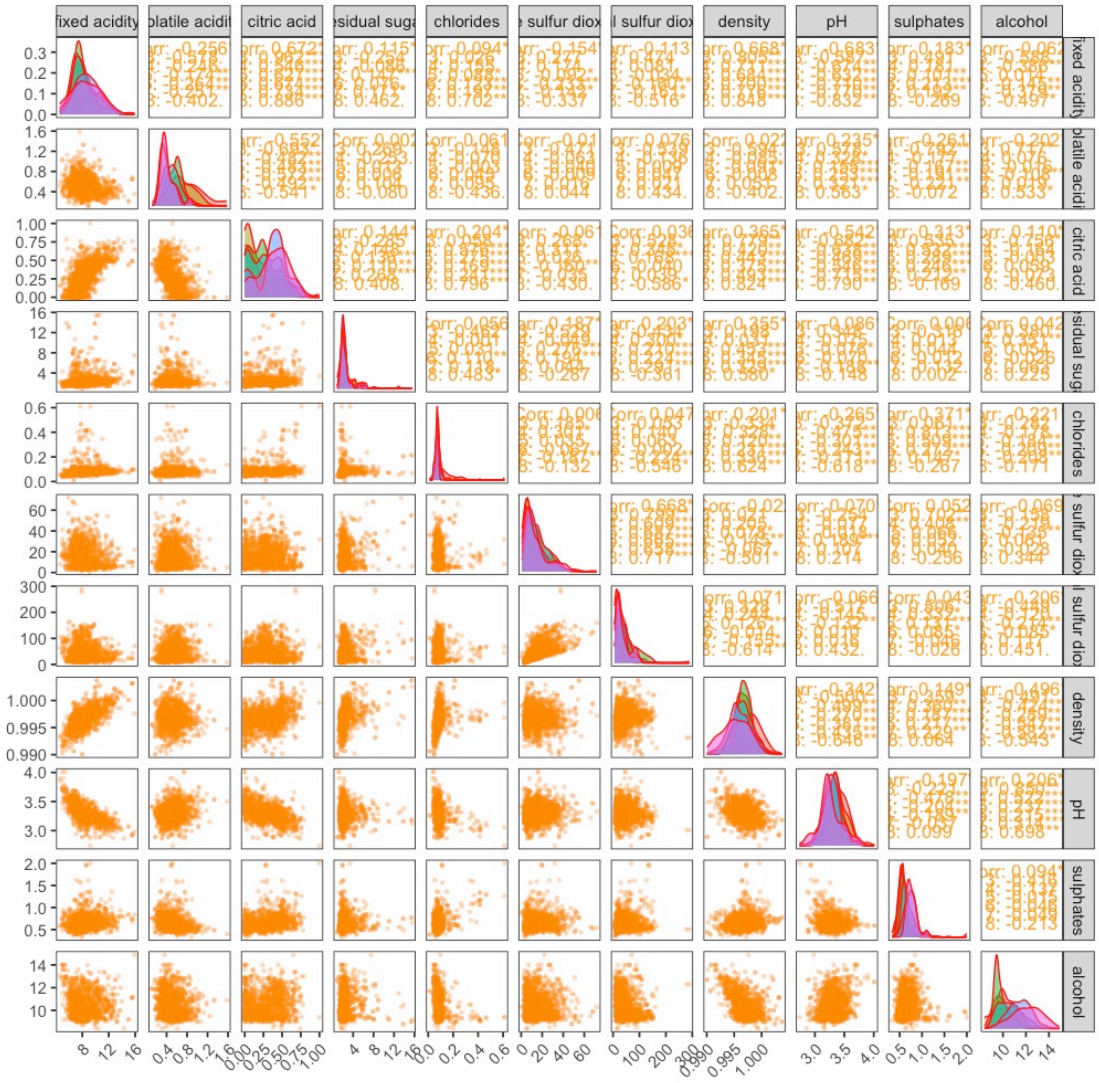
my_upper <- wrap("cor",
                 size = 2.2,
                 colour = "orange")

my_lower <- wrap("points",
                 size= 0.35,
                 alpha = 0.25,
                 colour = "darkorange")

my_diag  <- wrap("densityDiag",
                 colour = "red",
                 size = 0.25)

pairs <- ggpairs(
  df11,
  columnLabels = var_lab,
  aes(color = factor(df$quality), alpha = 0.55),
  upper = list(continuous = my_upper),
  lower = list(continuous = my_lower),
  diag  = list(continuous = my_diag)
) +
  theme_bw(base_size = 7) +
```

```
  theme(
    axis.text.x = element_text(size = 5.5, angle = 45, hjust = 1),
    axis.text.y = element_text(size = 5.5),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.position = "bottom")
print(pairs)
```




```
library(ggplot2)
library(gridExtra)
```

```
library(grid)

plot_list <- lapply(names(input_vars), function(var) {

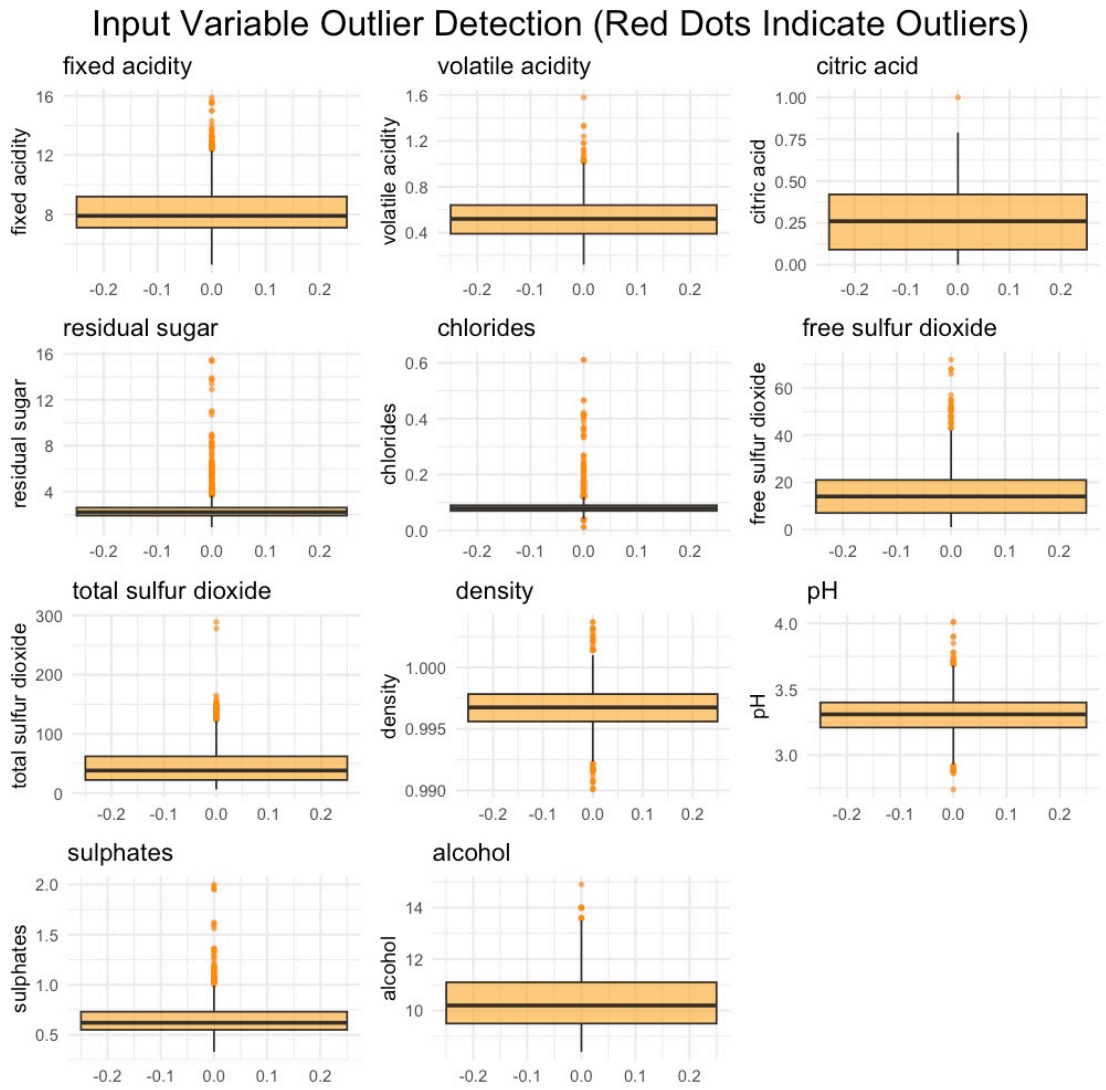
  ggplot(df, aes(y = .data[[var]])) +

    geom_boxplot(fill = "orange", alpha = 0.6, outlier.color = "darkorange", width =
0.5,outlier.size = 0.3) +

    labs(title = paste(var), y = var) +

    theme_minimal(base_size = 7)})

grid.arrange(grobs = plot_list, ncol = 3, top = " Input Variable Outlier Detection (Red Dots
Indicate Outliers)")
```



Several variables contain outliers:

1.Almost every variable shows red-flagged outliers, indicating that the dataset contains extreme or

anomalous observations along multiple dimensions.

Outlier patterns differ across variables:

2.For some variables (e.g., citric acid, total sulfur dioxide, residual sugar, alcohol) the outlying values are extremely high, far beyond the normal range.

3.For others (e.g., fixed acidity, volatile acidity) the outliers are less extreme but still noticeably higher than the bulk of the data.

4. A few variables exhibit outliers whose visual “prominence” varies—some barely stand out, others are clear extremes.

Data dispersion varies by variable:

5.Different box lengths and whisker spans reflect differing within-variable variability.

6.Citric acid’s tiny box shows highly concentrated values.

7.Fixed acidity’s longer box indicates greater dispersion.

Outlier detection

#Use the Z-score method (|Z| > 3) to identify outliers

```
z_scores <- as.data.frame(lapply(input_vars, function(x) (x - mean(x))/sd(x)))

outliers_z <- sapply(z_scores, function(x) sum(abs(x) > 3))

cat("======= Number of outliers by the Z-score method=======")

print(outliers_z)
```

```
> print(outliers_z)
       fixed.acidity     volatile.acidity          citric.acid
                  12                   10                    1
     residual.sugar            chlorides  free.sulfur.dioxide
                  30                   31                   22
total.sulfur.dioxide              density                   pH
                  15                   18                    8
           sulphates              alcohol
                  27                    8
```

```
library(dplyr)

quality_summary <- df %>%count(quality) %>%mutate(percentage = n / sum(n) * 100)

cat("=======Distribution of quality scores=======")

print(quality_summary)
```

```
> print(quality_summary)
   quality      n percentage
     <num> <int>       <num>
1:       3     10  0.6253909
2:       4     53  3.3145716
3:       5    681 42.5891182
4:       6    638 39.8999375
5:       7    199 12.4452783
6:       8     18  1.1257036
```

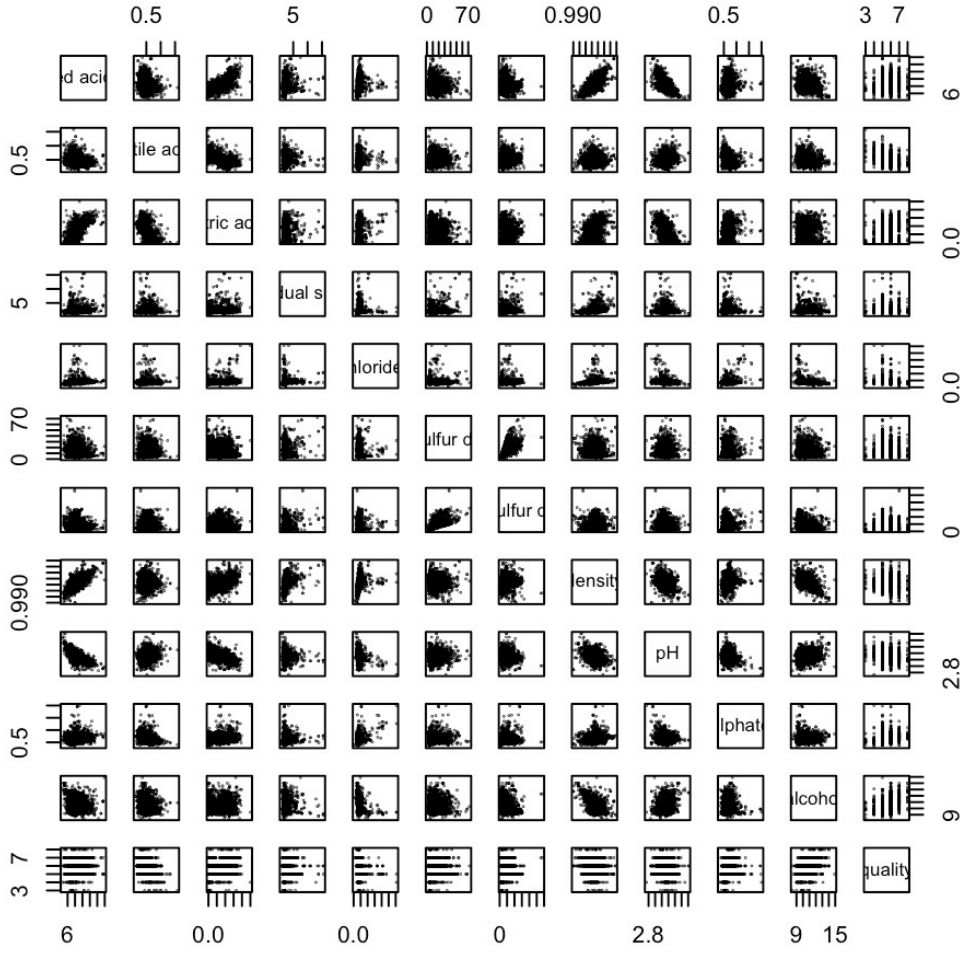
```
cor_with_quality <- cor(df[, 1:11], df$quality) %>%
  as.data.frame() %>%tibble::rownames_to_column("variable") %>%
  arrange(desc(abs(V1))) %>%head(3)
```

Variables strongly related to quality

```
cat("======= Variables strongly related to quality =======")
print(cor_with_quality)
```

```
> print(cor_with_quality)
          variable         V1
1          alcohol  0.4761663
2 volatile acidity -0.3905578
3        sulphates  0.2513971
```

```
strong_cor_pairs <- which(abs(cor_matrix) > 0.6 & abs(cor_matrix) < 1, arr.ind = TRUE)
strong_cor_df <- data.frame(
  var1 = rownames(cor_matrix)[strong_cor_pairs[, 1]],
  var2 = colnames(cor_matrix)[strong_cor_pairs[, 2]],
  cor = cor_matrix[strong_cor_pairs])
cat("======= Strongly correlated input variable pairs=======")
print(strong_cor_df)
```



###GLM

```
library(tidyverse)

library(car)

library(MASS)
```

##Data Exploration and Visualization

#Create a scatterplot matrix of variables to initially explore relationships between them

#Display all variable names in the dataset

```
pairs(df[, c(1:11, 12)], pch = 20,cex=0.1)

cat("Variable"); print(names(df))
```

```
df_clean <- df %>%filter(`alcohol` >= 8 & `alcohol` <= 22,`volatile acidity` <= 1.5,
                         `residual sugar` <= 65)
```

##Filtering Outliers: Limit reasonable ranges for alcohol content, volatile acidity, and residual sugar level

```
df_clean <- df_clean %>%mutate(
  log_residual_sugar = log(`residual sugar` + 1),
  log_chlorides = log(`chlorides` + 0.001))
```

##Building a GLM Model

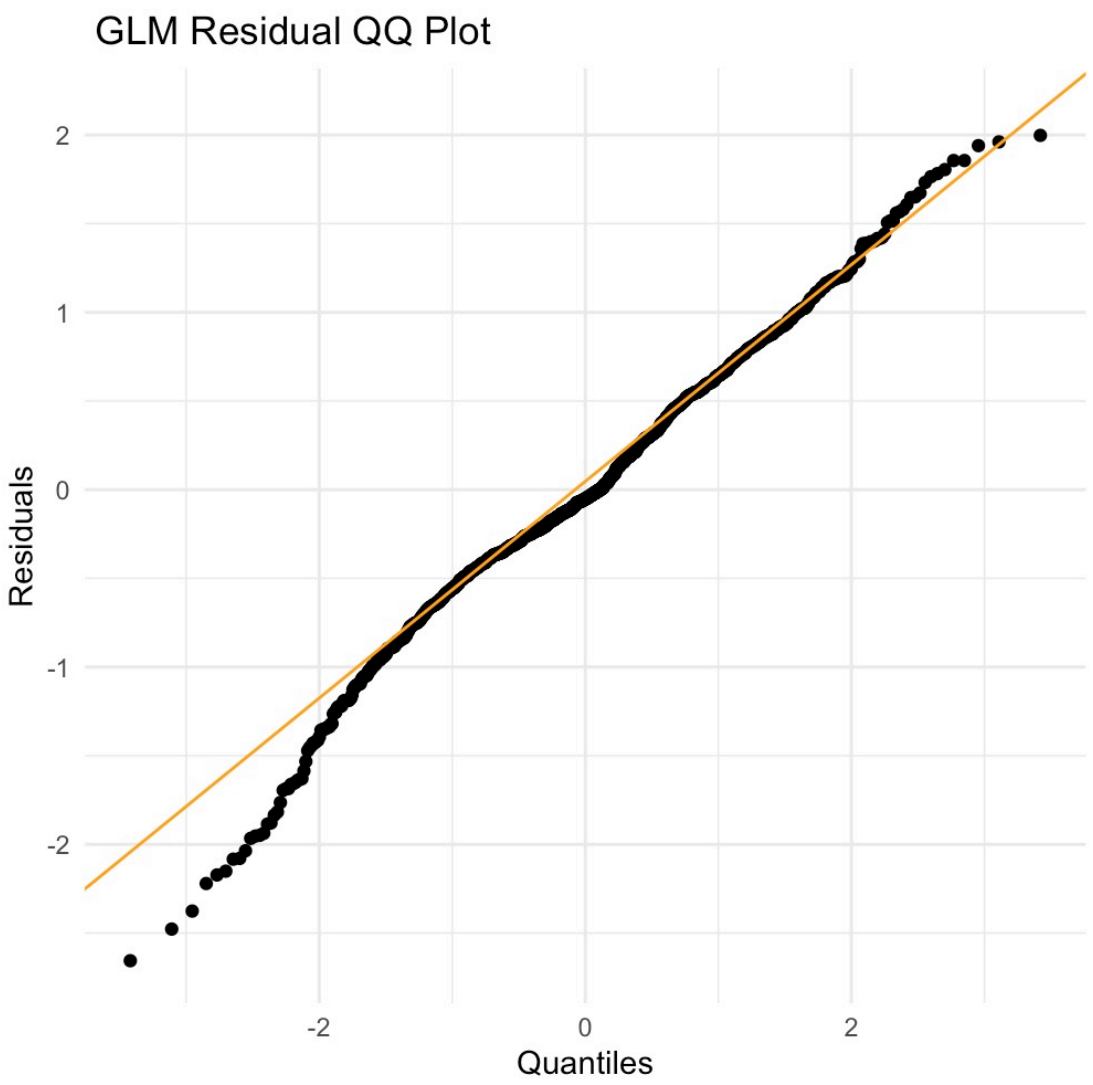
#Define the generalized linear model formula, using wine quality as the dependent variable and multiple chemical indicators as independent variables.

```
glm_formula <- quality ~
  `fixed acidity` + `volatile acidity` + `citric acid` + log_residual_sugar +
  log_chlorides + `free sulfur dioxide` + `total sulfur dioxide` +
  density + pH + sulphates + alcohol
glm_model <- glm(formula = glm_formula,data = df_clean,
                 family = gaussian(link = "identity"), na.action = na.exclude )


cat("======= GLM model summary=======")
summary(glm_model)
```

##Model Diagnosis and Testing

```
qq_plot <- ggplot(data.frame(residuals = residuals(glm_model)), aes(sample = residuals)) +
  stat_qq() +
  stat_qq(size = 1)+
  stat_qq_line(color = "orange") +
  labs(title = " GLM Residual QQ Plot", x = " Quantiles ", y = " Residuals ") +
  theme_minimal()
print(qq_plot)
```




```
shapiro_test <- shapiro.test(residuals(glm_model))

cat("======= Normality Test for Residuals=======")

print(shapiro_test)
```

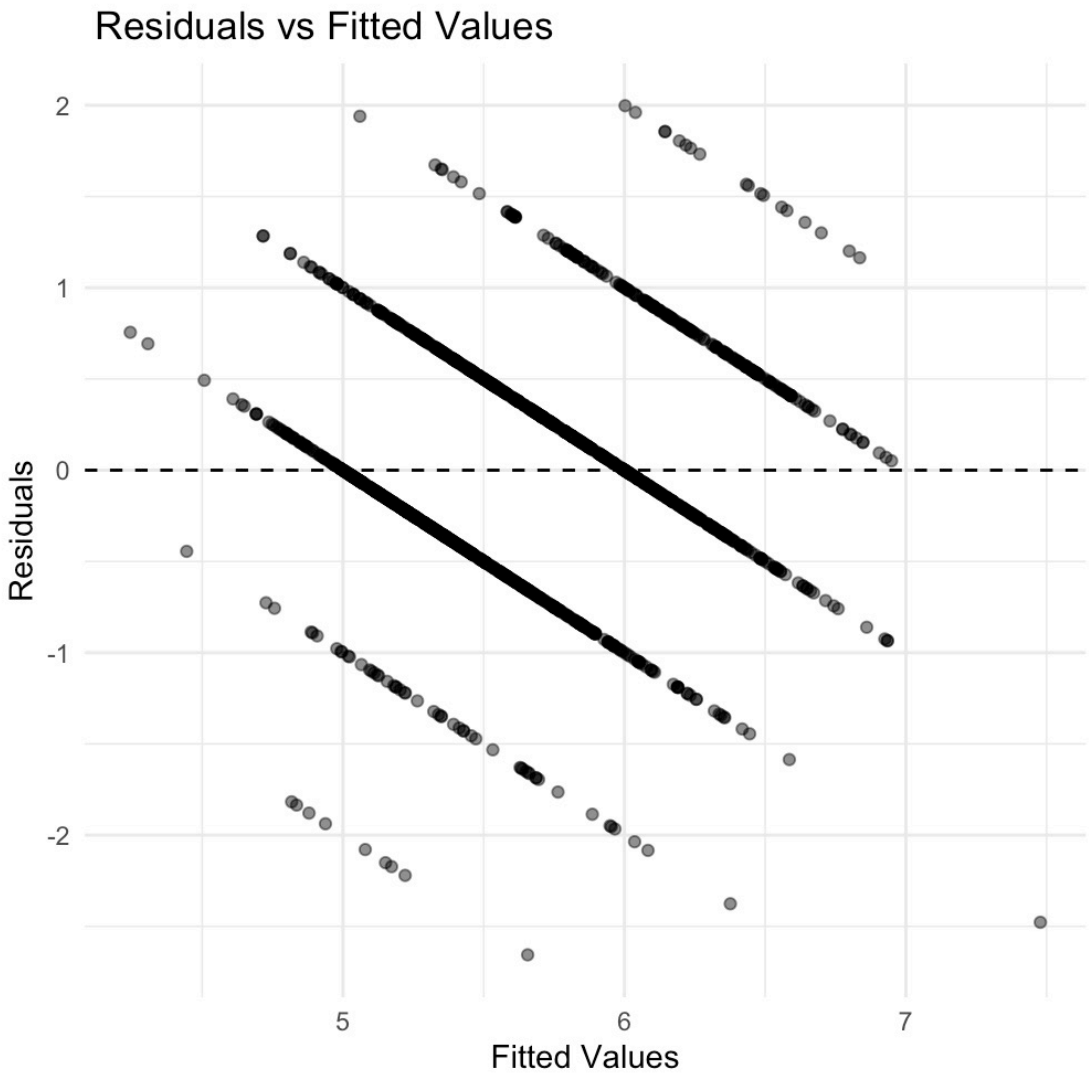
```
> print(shapiro_test)

        Shapiro-Wilk normality test

data:  residuals(glm_model)
W = 0.99001, p-value = 5.307e-09
```

```
resid_fit_plot <- ggplot(data.frame(fitted = fitted(glm_model),
                                    resid = residuals(glm_model)), aes(x = fitted, y =
resid)) +
  geom_point(alpha = 0.5) + geom_hline(yintercept = 0, linetype = "dashed") +
```

```
  labs(title = " Residuals vs Fitted Values", x = " Fitted Values ", y = " Residuals ") +
  theme_minimal()
print(resid_fit_plot)
```



*Wine quality scores are not continuous but integer-valued from 1 to 10.Consequently, the model's fitted values are also effectively integers, creating the distinct banded regions visible in the plot.

```
library(lmtest)
bp_test <- bptest(glm_model)
cat("======= Breusch-Pagan =======")
print(bp_test)
vif_values <- car::vif(glm_model)
```

```
cat("======= VIF values for each variable =======")

print(vif_values)
```

```
> print(vif_values)
       `fixed acidity`    `volatile acidity`         `citric acid`
              7.990067             1.765019              3.023851
   log_residual_sugar       log_chlorides `free sulfur dioxide`
              1.979572             1.459264              1.943589
`total sulfur dioxide`              density                    pH
              2.182094             7.402471              3.406602
             sulphates              alcohol
              1.346568             3.399818
```

```

**##Use AIC criterion for stepwise regression to automatically select the optimal combination of variables and avoid overfitting.**

#combination of variables and avoid overfitting.

```
glm_step <- MASS::stepAIC(glm_model,direction = "both",   trace = FALSE)

cat("======= Optimized model results after stepwise regression=======")

summary(glm_step)
```

**##Results Interpretation**

#Extract significant variables ($p < 0.05$), rank them by the magnitude of their effects, indicate the direction of positive or negative impact, and identify the key factors that have a significant influence on wine quality.

```
sig_vars <- summary(glm_step)$coefficients %>%as.data.frame() %>%

   rownames_to_column("variable") %>%

   filter(`Pr(>|t|)` < 0.05) %>%

   mutate(effect = ifelse(Estimate > 0, " positive effect ", " negative effect "),

             magnitude = abs(Estimate)) %>%

   arrange(desc(magnitude))

cat("======= Variables that have a significant impact on quality =======")

print(sig_vars[, c("variable", "effect", "Estimate", "Pr(>|t|)")])
```
```

```
> print(sig_vars[, c("variable", "effect", "Estimate", "Pr(>|t|)")])
                variable            effect     Estimate     Pr(>|t|)
1            (Intercept)  positive effect  3.679918988 1.440709e-20
2     `volatile acidity`  negative effect -0.987538403 1.735190e-21
3              sulphates  positive effect  0.808312914 5.599571e-14
4                     pH  negative effect -0.482580225 4.743204e-05
5                alcohol  positive effect  0.288214974 8.127956e-59
6          log_chlorides  negative effect -0.249853598 1.602430e-05
7  `free sulfur dioxide`  positive effect  0.005002161 1.882114e-02
8 `total sulfur dioxide`  negative effect -0.003490192 4.298085e-07
```

###DEEP LEARNING

cat("==========Deep Learning==========")

library(keras)

library(tidyverse)

library(caret)

library(readr)

library(tensorflow)

##Data Preparation and Preprocessing

#Set a random seed to ensure reproducibility, and split the data into training and testing sets with an 80/20 ratio.

set.seed(123)

X <- df[, 1:11]

y <- df$quality

preprocess_params <- preProcess(X, method = c("center", "scale"))

X_scaled <- predict(preprocess_params, X)

X_matrix <- as.matrix(X_scaled)

y_vector <- as.numeric(y)

train_index <- createDataPartition(y_vector, p = 0.8, list = FALSE)

```
```

##Neural Network Architecture Design

```
cat("======= Build a neural network model =======")

model <- keras_model_sequential() %>%

  layer_dense(units = 128, activation = "relu", input_shape = ncol(x_train)) %>%

  layer_batch_normalization() %>%

  layer_dropout(rate = 0.3) %>%


  layer_dense(units = 64, activation = "relu") %>%

  layer_batch_normalization() %>%

  layer_dropout(rate = 0.3) %>%


  layer_dense(units = 32, activation = "relu") %>%

  layer_dropout(rate = 0.2) %>%


  layer_dense(units = 16, activation = "relu") %>%

  layer_dense(units = 1, activation = "linear")
```

##Model Compilation and Configuration

#Track both MAE and MSE simultaneously

```
model %>% compile(optimizer = optimizer_adam(learning_rate = 0.001),loss = "mse",

                  metrics = c("mae", "mse"))

cat("=======Model architecture=======")

summary(model)
```

##Callback Settings

```
callbacks_list <- list(callback_early_stopping(monitor = "val_loss",

                                               patience = 15,restore_best_weights =
TRUE),

                       callback_reduce_lr_on_plateau(monitor = "val_loss",factor =
```

0.5,patience = 8,

min_lr = 0.0001),

callback_model_checkpoint(filepath =
"best_wine_model.h5",save_best_only = TRUE,

monitor = "val_loss"))

```

##Model Training and Monitoring

#Train for 200 epochs: maximum number of iterations

#Batch size 64: balance memory usage and training efficiency

```{r}

cat("======= Start training the model =======")

history <- model %>% fit(x_train, y_train,epochs = 200,batch_size = 64,

validation_split = 0.2,verbose = 1,callbacks = callbacks_list)

cat("======= Plot training history =======")

plot(history)

##Model Evaluation and Prediction

#Quantitative evaluation: Compute metrics such as MSE and MAE on the test set

```{r}

cat("======= Model Evaluation =======")

test_evaluation <- model %>% evaluate(x_test, y_test, verbose = 0)

cat("Test loss (MSE)", test_evaluation[[1]], "\n")

cat("est Mean Absolute Error (MAE", test_evaluation[[2]], "\n")

cat("Test Mean Squared Error ", test_evaluation[[3]], "\n")

at("=======Prediction=======")

predictions <- model %>% predict(x_test)

predictions <- as.numeric(predictions)
```

```
cat("Root Mean Squared Error (RMSE)", rmse, "\n")

cat("Mean Absolute Error (MAE)", mae, "\n")

cat("Coefficient of Determination (R²)", r_squared, "\n")
```

##Performance Analysis and Result Interpretation

```{r}
results_df <- data.frame(ActualQuality = y_test,
                         PredictedQuality = round(predictions, 2),
                         AbsoluteError = abs(y_test - predictions),
                         RelativeError = abs(y_test - predictions) / y_test * 100)


cat("=======Predicted results=======")

print(head(results_df, 10))


quality_performance <- results_df %>%
  mutate(QualityGrade = as.factor(ActualQuality)) %>%
  group_by(QualityGrade) %>%
  summarise(SampleCount = n(),
            MeanAbsoluteError = mean(AbsoluteError),
            Accuracy = sum(round(PredictedQuality) == ActualQuality) / n() *
100) %>%arrange(QualityGrade)


cat("\nPerformance Analysis by Quality Grade:\n")

print(quality_performance)
```

###RANDOM FOREST

```
cat("==================Random Forest==================")

library(randomForest)

library(caret)

library(ggplot2)

library(dplyr)
```

##Data Preprocessing

```
names(df) <- make.names(names(df))

df$quality <- as.factor(df$quality)

table(df$quality)
```

##Data Splitting

#70/30 split: 70% for training, 30% for testing, while maintaining

```
randomnessset.seed(123)

train_index <- createDataPartition(df$quality, p = 0.7, list = FALSE)

train_data <- df[train_index, ]

test_data <- df[-train_index, ]

train_data$quality <- factor(train_data$quality)

test_data$quality <- factor(test_data$quality, levels = levels(train_data$quality))
```

##Basic Random Forest Model Construction

#500 trees: A sufficient number of trees to ensure stability

```
set.seed(123)

rf_model <- randomForest(quality ~ .,
                         data = train_data,
                         ntree = 500,
                         mtry = sqrt(ncol(train_data)-1),
                         importance = TRUE,
```

```
                                    na.action = na.omit)

print(rf_model)
```

##Basic Model Prediction and Evaluation

#Generate predictions: Perform classification predictions on the test set

#Confusion matrix: Comprehensively evaluate classification performance (accuracy, recall, F1 score, etc.)

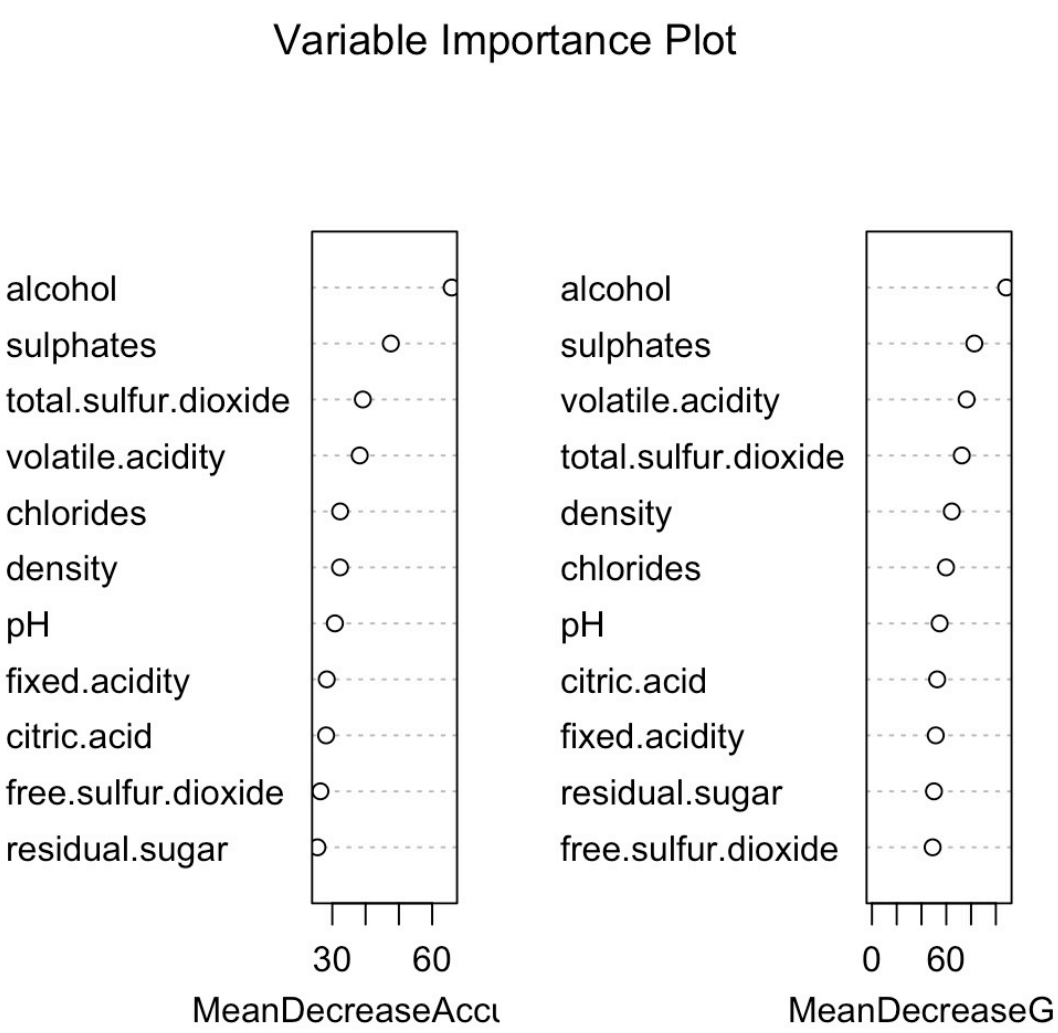
```
predictions <- predict(rf_model, test_data)

conf_matrix <- confusionMatrix(predictions, test_data$quality)

print(conf_matrix)
```

```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction   3   4   5   6   7   8
         3   0   0   0   0   0   0
         4   1   0   0   1   0   0
         5   2  10 160  43   2   0
         6   0   5  44 138  29   1
         7   0   0   0   9  28   3
         8   0   0   0   0   0   1

Overall Statistics

               Accuracy : 0.6855
                 95% CI : (0.6418, 0.727)
    No Information Rate : 0.4277
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4868

 Mcnemar's Test P-Value : NA
```

##Variable Importance Analysis

#Visualize variable importance to identify the features that contribute most to wine quality classification, providing a basis for feature selection and model interpretation.

```
importance_plot <- varImpPlot(rf_model, main = " Variable Importance Plot")

print(importance_plot)
```



##Hyperparameter Tuning

#More robust performance evaluation

#Multiple parameter combinations: adjust mtry (number of features considered at each split) and minimum node sample size

```
control <- trainControl(method = "cv", number = 5, verboseIter = TRUE)

tune_grid <- expand.grid(mtry = c(2, 4, 6, 8, 10),

                         splitrule = "gini",min.node.size = c(1, 5, 10))

rf_tune <- train(quality ~ .,data = train_data,method = "ranger",trControl = control,

                 tuneGrid = tune_grid,importance = "impurity")

print(rf_tune)
```

##Final Model Evaluation

```
#Re-predict: Ensure consistency of factor levels
final_predictions <- predict(rf_model, test_data)
final_predictions <- factor(final_predictions, levels = levels(test_data$quality))
final_conf_matrix <- confusionMatrix(final_predictions, test_data$quality)
print(final_conf_matrix)
accuracy <- final_conf_matrix$overall['Accuracy']
cat("model accuracy:", round(accuracy, 3), "\n")
print(final_conf_matrix$byClass[, c("Precision", "Recall", "F1")])
```