Model - Final Project

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Since the red and white tastes are quite different, the analysis will be performed separately.
red <- read.csv("winequality-red.csv", sep=";")
white <- read.csv("winequality-white.csv", sep=";")

red$color_variant <- "Red"
white$color_variant <- "White"
wines <- bind_rows(red, white)

summary(red)
```

```
##  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   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
```

```
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
##  color_variant
##  Length:1599
##  Class :character
##  Mode  :character
##
##
##
```

```
summary(white)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
##  color_variant
##  Length:4898
##  Class :character
##  Mode  :character
##
##
##
```

```
# Convert quality to a factor
wines$quality <- as.factor(wines$quality)
table(wines$quality)
```

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

```
features <- names(wines)[names(wines) != "color_variant"]
```

```r
#After calculating multicollinearity, we see that desntiy has a very high VIF which indicates a strong
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.3
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
trainWhite <- white %>% select(-color_variant)

lmFit <- lm(quality ~ ., data = trainWhite)
vif_values <- vif(lmFit)
print(vif_values)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             2.691435             1.141156             1.165215
##       residual.sugar            chlorides  free.sulfur.dioxide
##            12.644064             1.236822             1.787880
## total.sulfur.dioxide              density                   pH
##             2.239233            28.232546             2.196362
##            sulphates              alcohol
##             1.138540             7.706957
```

```r
# Remove density due to high multicollinearity. Updated VIF results show that the multicollinearity iss
trainWhite <- trainWhite %>% select(-density)
lmFit <- lm(quality ~ ., data = trainWhite)
vif_values <- vif(lmFit)
print(vif_values)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             1.356128             1.128298             1.159884
##       residual.sugar            chlorides  free.sulfur.dioxide
##             1.435215             1.203645             1.744627
## total.sulfur.dioxide                   pH            sulphates
##             2.153170             1.330912             1.056637
##              alcohol
##             1.647117
```

```r
#Explore higher-order polynomial relationships and interactions among features that could better predic

#For 10 remaining predictors, adding polynomial terms up to degree 2 and all interactions results in ma

# Load necessary library
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
expand_polynomial <- function(data, degree = 4) {
  # Extract the target variable
  target <- data$quality

  # Select all numeric predictors, excluding the target variable
  predictors <- data %>% select(-quality)

  # Initialize an empty list to store expanded features
  expanded_list <- list()

  # Loop through each predictor to add polynomial terms up to the specified degree
  for (col in names(predictors)) {
    # Generate polynomial terms for this column
    polynomial_terms <- poly(predictors[[col]], degree = degree, raw = TRUE)

    # Set proper column names for the polynomial terms
    colnames(polynomial_terms) <- paste0(col, "_poly_", 1:degree)

    # Add the polynomial terms to the list
    expanded_list[[col]] <- polynomial_terms
  }

  # Combine all polynomial features into a single data frame
  expanded_df <- do.call(cbind, expanded_list)

  # Add the target variable back into the expanded data frame
  expanded_df <- as.data.frame(expanded_df)
  expanded_df$quality <- target

  return(expanded_df)
}

# Apply the expansion function to the training data
train_poly <- expand_polynomial(trainWhite, degree = 2)

# Verify the expanded feature set
print(head(train_poly))
```

```
##   fixed.acidity_poly_1 fixed.acidity_poly_2 volatile.acidity_poly_1
## 1                  7.0                49.00                    0.27
## 2                  6.3                39.69                    0.30
## 3                  8.1                65.61                    0.28
## 4                  7.2                51.84                    0.23
## 5                  7.2                51.84                    0.23
## 6                  8.1                65.61                    0.28
##   volatile.acidity_poly_2 citric.acid_poly_1 citric.acid_poly_2
## 1                  0.0729               0.36             0.1296
## 2                  0.0900               0.34             0.1156
## 3                  0.0784               0.40             0.1600
```

```
## 4                  0.0529               0.32            0.1024
## 5                  0.0529               0.32            0.1024
## 6                  0.0784               0.40            0.1600
##   residual.sugar_poly_1 residual.sugar_poly_2 chlorides_poly_1 chlorides_poly_2
## 1                  20.7                428.49           0.045         0.002025
## 2                   1.6                  2.56           0.049         0.002401
## 3                   6.9                 47.61           0.050         0.002500
## 4                   8.5                 72.25           0.058         0.003364
## 5                   8.5                 72.25           0.058         0.003364
## 6                   6.9                 47.61           0.050         0.002500
##   free.sulfur.dioxide_poly_1 free.sulfur.dioxide_poly_2
## 1                         45                       2025
## 2                         14                        196
## 3                         30                        900
## 4                         47                       2209
## 5                         47                       2209
## 6                         30                        900
##   total.sulfur.dioxide_poly_1 total.sulfur.dioxide_poly_2 pH_poly_1 pH_poly_2
## 1                         170                       28900      3.00    9.0000
## 2                         132                       17424      3.30   10.8900
## 3                          97                        9409      3.26   10.6276
## 4                         186                       34596      3.19   10.1761
## 5                         186                       34596      3.19   10.1761
## 6                          97                        9409      3.26   10.6276
##   sulphates_poly_1 sulphates_poly_2 alcohol_poly_1 alcohol_poly_2 quality
## 1             0.45           0.2025            8.8          77.44       6
## 2             0.49           0.2401            9.5          90.25       6
## 3             0.44           0.1936           10.1         102.01       6
## 4             0.40           0.1600            9.9          98.01       6
## 5             0.40           0.1600            9.9          98.01       6
## 6             0.44           0.1936           10.1         102.01       6
```

```
#K-Fold Cross-Validation (K-Folds = 10): We'll split the expanded dataset into 10 folds. Then, for each
#We'll train the Lasso regression model on the remaining 9 folds.
#We'll evaluate the model's performance on the held-out fold.
#This process will be repeated for each fold, ensuring that each data point is used for both training a

library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(caret)

# Extract predictors (X) and response variable (y)
X <- as.matrix(train_poly[, -ncol(train_poly)])
y <- train_poly$quality

set.seed(1)
```

```r
num_folds <- 10
folds <- createFolds(y, k = num_folds)

# Initialize vectors to store results
test_r_squared <- numeric(num_folds)
test_mse <- numeric(num_folds)
aic_values <- numeric(num_folds)


# Perform cross-validated Lasso regression
for (i in 1:num_folds) {
  X_train <- X[-folds[[i]], ]
  y_train <- y[-folds[[i]]]
  X_test <- X[folds[[i]], ]
  y_test <- y[folds[[i]]]

  # Train Lasso regression model
  lasso_model <- cv.glmnet(X_train, y_train, alpha = 1, nfolds = num_folds)

  # Plot the cross-validated mean squared error (MSE) against log(lambda)
  #plot(lasso_model)

  # Get the optimal lambda value with the lowest MSE
  optimal_lambda <- lasso_model$lambda.min
  #cat("Optimal lambda value:", optimal_lambda, "\n")
  #plot(lasso_model$glmnet.fit,
  #   "lambda", label=FALSE)

  # Fit the final Lasso model with the optimal lambda
  final_model <- glmnet(X_train, y_train, alpha = 1, lambda = optimal_lambda)

  #coefficients <- coef(final_model)
    # Predict on test set
  y_pred <- predict(final_model, s = optimal_lambda, newx = X_test)

    # Calculate R^2
  test_r_squared[i] <- cor(y_pred, y_test)^2

  # Calculate MSE
  test_mse[i] <- mean((y_pred - y_test)^2)

  # Calculate AIC
  #deviance <- deviance(final_model)
  #num_coeffs <- sum(coef(final_model) != 0)
  #aic <- 2 * num_coeffs + deviance
  #aic_values[i] = aic
}


# Calculate average test R^2 and MSE
average_test_r_squared <- mean(test_r_squared)
average_test_mse <- mean(test_mse)
average_aic <- mean(aic_values)
```

```r
# Print results
cat("Average Test R-squared (White):", average_test_r_squared, "\n")
```

```
## Average Test R-squared (White): 0.278164
```

```r
cat("Average Test MSE (White):", average_test_mse, "\n")
```

```
## Average Test MSE (White): 0.5871471
```

```r
#cat("Average AIC:", average_aic, "\n")
```

Now we are starting the model building for the RED Wine.

```r
#RED

trainRed <- red %>% select(-color_variant)

lmFit <- lm(quality ~ ., data = trainRed)
vif_values <- vif(lmFit)
print(vif_values)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             7.767512             1.789390             3.128022
##       residual.sugar            chlorides  free.sulfur.dioxide
##             1.702588             1.481932             1.963019
## total.sulfur.dioxide              density                   pH
##             2.186813             6.343760             3.329732
##            sulphates              alcohol
##             1.429434             3.031160
```

```r
# Remove density due to high multicollinearity. Updated VIF results show that the multicollinearity iss
trainRed <- trainRed %>% select(-density)
lmFit <- lm(quality ~ ., data = trainRed)
vif_values <- vif(lmFit)
print(vif_values)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             2.975491             1.759879             3.127791
##       residual.sugar            chlorides  free.sulfur.dioxide
##             1.099433             1.468893             1.948691
## total.sulfur.dioxide                   pH            sulphates
##             2.173240             2.239412             1.341524
##              alcohol
##             1.299603
```

```r
library(caret)

expand_polynomial <- function(data, degree = 4) {
  # Extract the target variable
```

```r
  target <- data$quality

  # Select all numeric predictors, excluding the target variable
  predictors <- data %>% select(-quality)

  # Initialize an empty list to store expanded features
  expanded_list <- list()

  # Loop through each predictor to add polynomial terms up to the specified degree
  for (col in names(predictors)) {
    # Generate polynomial terms for this column
    polynomial_terms <- poly(predictors[[col]], degree = degree, raw = TRUE)

    # Set proper column names for the polynomial terms
    colnames(polynomial_terms) <- paste0(col, "_poly_", 1:degree)

    # Add the polynomial terms to the list
    expanded_list[[col]] <- polynomial_terms
  }

  # Combine all polynomial features into a single data frame
  expanded_df <- do.call(cbind, expanded_list)

  # Add the target variable back into the expanded data frame
  expanded_df <- as.data.frame(expanded_df)
  expanded_df$quality <- target

  return(expanded_df)
}

# Apply the expansion function to the training data
train_poly2 <- expand_polynomial(trainRed, degree = 2)

# Verify the expanded feature set
print(head(train_poly2))
```

```
##   fixed.acidity_poly_1 fixed.acidity_poly_2 volatile.acidity_poly_1
## 1                  7.4                54.76                    0.70
## 2                  7.8                60.84                    0.88
## 3                  7.8                60.84                    0.76
## 4                 11.2               125.44                    0.28
## 5                  7.4                54.76                    0.70
## 6                  7.4                54.76                    0.66
##   volatile.acidity_poly_2 citric.acid_poly_1 citric.acid_poly_2
## 1                  0.4900               0.00             0.0000
## 2                  0.7744               0.00             0.0000
## 3                  0.5776               0.04             0.0016
## 4                  0.0784               0.56             0.3136
## 5                  0.4900               0.00             0.0000
## 6                  0.4356               0.00             0.0000
##   residual.sugar_poly_1 residual.sugar_poly_2 chlorides_poly_1 chlorides_poly_2
## 1                   1.9                  3.61            0.076         0.005776
## 2                   2.6                  6.76            0.098         0.009604
```

```
## 3                   2.3                  5.29           0.092         0.008464
## 4                   1.9                  3.61           0.075         0.005625
## 5                   1.9                  3.61           0.076         0.005776
## 6                   1.8                  3.24           0.075         0.005625
##   free.sulfur.dioxide_poly_1 free.sulfur.dioxide_poly_2
## 1                         11                        121
## 2                         25                        625
## 3                         15                        225
## 4                         17                        289
## 5                         11                        121
## 6                         13                        169
##   total.sulfur.dioxide_poly_1 total.sulfur.dioxide_poly_2 pH_poly_1 pH_poly_2
## 1                          34                        1156      3.51   12.3201
## 2                          67                        4489      3.20   10.2400
## 3                          54                        2916      3.26   10.6276
## 4                          60                        3600      3.16    9.9856
## 5                          34                        1156      3.51   12.3201
## 6                          40                        1600      3.51   12.3201
##   sulphates_poly_1 sulphates_poly_2 alcohol_poly_1 alcohol_poly_2 quality
## 1             0.56           0.3136            9.4          88.36       5
## 2             0.68           0.4624            9.8          96.04       5
## 3             0.65           0.4225            9.8          96.04       5
## 4             0.58           0.3364            9.8          96.04       6
## 5             0.56           0.3136            9.4          88.36       5
## 6             0.56           0.3136            9.4          88.36       5
```

```r
library(glmnet)
library(caret)

# Extract predictors (X) and response variable (y)
X2 <- as.matrix(train_poly2[, -ncol(train_poly2)])
y2 <- train_poly2$quality

set.seed(1)
num_folds2 <- 10
folds2 <- createFolds(y2, k = num_folds2)

# Initialize vectors to store results
test_r_squared2 <- numeric(num_folds2)
test_mse2 <- numeric(num_folds2)
aic_values2 <- numeric(num_folds2)


# Perform cross-validated Lasso regression
for (i in 1:num_folds2) {
  X_train2 <- X[-folds2[[i]], ]
  y_train2 <- y[-folds2[[i]]]
  X_test2 <- X[folds2[[i]], ]
  y_test2 <- y[folds2[[i]]]

  # Train Lasso regression model
  lasso_model2 <- cv.glmnet(X_train2, y_train2, alpha = 1, nfolds = num_folds2)

  # Plot the cross-validated mean squared error (MSE) against log(lambda)
```

```r
  #plot(lasso_model)

  # Get the optimal lambda value with the lowest MSE
  optimal_lambda2 <- lasso_model2$lambda.min
  #cat("Optimal lambda value:", optimal_lambda, "\n")
  #plot(lasso_model$glmnet.fit,
  #   "lambda", label=FALSE)

  # Fit the final Lasso model with the optimal lambda
  final_model2 <- glmnet(X_train2, y_train2, alpha = 1, lambda = optimal_lambda)

  #coefficients <- coef(final_model)
    # Predict on test set
  y_pred2 <- predict(final_model2, s = optimal_lambda2, newx = X_test2)

    # Calculate R^2
  test_r_squared2[i] <- cor(y_pred2, y_test2)^2

  # Calculate MSE
  test_mse2[i] <- mean((y_pred2 - y_test2)^2)

  # Calculate AIC
  #deviance <- deviance(final_model)
  #num_coeffs <- sum(coef(final_model) != 0)
  #aic <- 2 * num_coeffs + deviance
  #aic_values[i] = aic
}


# Calculate average test R^2 and MSE
average_test_r_squared2 <- mean(test_r_squared2)
average_test_mse2 <- mean(test_mse2)
average_aic2 <- mean(aic_values2)

# Print results
cat("Average Test R-squared (Red):", average_test_r_squared2, "\n")
```

```
## Average Test R-squared (Red): 0.2804342
```

```r
cat("Average Test MSE (Red):", average_test_mse2, "\n")
```

```
## Average Test MSE (Red): 0.6482802
```

```r
#cat("Average AIC:", average_aic, "\n")
```