Wine Quality Data Analysis Project Report

Group Members: Heyang Dong(hd385), Renee Lin(rxl3), Daniel Han(dhh78), James Lee(jyl84)

Abstract:

This report explores the application of machine learning techniques to predict the quality of Portuguese "Vinho Verde" wines using physicochemical properties. The dataset consists of 4,898 instances with 11 continuous features such as acidity, residual sugar, pH, and alcohol content, and one output variable representing wine quality on a scale from 0 to 10. The prediction task can be framed as both a regression and classification problem.

Multiple classification models are evaluated, including Logistic Regression, Random Forest, Boosting, LDA, QDA, and KNN. Cross-validation and performance metrics are employed to predict model accuracy and generalization. Random Forest proved to be the best classification model for predicting wine quality based on physicochemical properties. Linear Regression and Bagging are used as regression models, with test MSE serving as the evaluation metric. Bagging results in a lower test MSE than Linear Regression.

Introduction:

Evaluating wine quality is a subjective process often influenced by sensory perception and individual taste. In this project, we aim to build a predictive model that estimates the quality of Portuguese "Vinho Verde" wines based on measurable physicochemical features. Since wine quality in the dataset is given as an integer score from 3 to 9 and is typically interpreted in discrete terms (e.g. “this is a 7-point wine”), we first approach this as a classification problem rather than regression. Then, we also use regression methods for comparison.

We begin by preprocessing the data. This includes encoding the wine type as a binary feature (white = True, red = False) and standardizing the remaining predictors to have zero mean and unit variance. This step ensures that features on larger scales do not dominate the learning process. We then split the data into a training set (70%) and a test set (30%), and apply 5-fold cross-validation to estimate model performance reliably across different data splits.

We evaluate a wide set of machine learning algorithms including logistic regression, linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), k-nearest neighbors (KNN), random forest, and boosting. Each model is trained on the training set and validated through cross-validation to assess accuracy and generalizability. Based on the cross-validation results, we identify the most effective model. We then use linear regression and bagging methods.

Data Description:

The dataset used in this project consists of two files representing white and red variants of Portuguese "Vinho Verde" wine, collected from the northern region of Portugal. The data is sourced from the UCI Machine Learning Repository and originally collected by Cortez et al. (2009) in their study "Modeling wine preferences by data mining from physicochemical properties." The goal is to be able to predict wine quality based on various physicochemical properties obtained through lab testing. Both datasets contain several thousand samples, with 4,898 for white wine and 1,599 for red wine. There are no missing values in either dataset. After merging the white and red datasets, we introduce a binary variable to distinguish the two types of wine with white being TRUE and red being FALSE. This results in a combined dataset of 6,497 observations and 13 variables: 11 numeric predictors (UCI dataset), 1 target variable (quality from 0 to 10) and 1 added feature (type). However, the quality scores in this particular dataset fall between 3 and 9.

The 11 input features/variables reflect the physicochemical characteristics of the wine including fixed, volatile, and citric acidity, residual sugar, chlorides, free and total sulfur dioxide, density, pH, sulphates, and alcohol percentage. These variables were all measured in controlled lab settings. The target variable, quality, is a sensory score assigned by human tasters and is treated, in our personal analysis, as a categorical variable since wine ratings are given in whole numbers (ex: a wine is given a rating of 8 rather than 7.7).

There are some modeling challenges in this dataset as the classes are not balanced, meaning that there are a lot more "normal" or average rated wines than excellent or poor ones. This imbalance can affect how well models can perform, causing a bias in model predictions toward the majority classes. As a result, strong evaluation methods, such as cross validation, are necessary to ensure the model works well across all quality levels. Additionally, since the dataset

lacks information about the wine's brand, grape variety, price, etc., the predictions are based purely on wine's physicochemical properties, making it a good fit for classification approaches in machine learning.

Methodology:

Part 1 Classification

Wine quality takes integer values from 3 to 9. Also, due to the nature of wine quality, we are treating it as a categorical variable. Therefore, we use classification methods. There are many classification methods. In this project, we use 5-fold cv for all classification methods.
For each model, we first fit the model with the training data. Then, we use cross validation to test the effectiveness of each model. After checking the CV accuracy rate of each model, we select the best model, and use that model for final training and testing.

The models we try in this project are logistic regression, LDA, QDA, KNN, Random Forest and Boosting. One note for the classification part is that we use accuracy instead of error, but the idea is the same.

Data Preprocessing

First, we convert the categorical variable "types" into binary categorical variables, white being True and red being False.

Second, we scale all the predictors except for type into values with mean of 0 and standard deviation of 1. The scales of the original dataset were very different. The smallest ones were 0 to 1 and some of them were hundreds. We do this so that the large ones don't overly dominate the model.

Split dataset into training and test set

We randomly split 70% data to the training and 30% to the test set. Since there are only 12 predictors in the dataset, it's very unlikely that there is overfitting or too much computation. Therefore, in this project, we are not using any subset selection. We use all predictors.

Cross validation

In this step, for Logistic Regression, we first use multinom() to fit the logistic regression model. Then, we use set.seed(1) to make sure the results are reproducible. The "folds" are

randomly generated from the training set. Then, in the for loop, cross validation is done to estimate the effectiveness of the model. The cross validation accuracy is calculated. The similar process is done for all the other models.

In the cross validation step, cross validation is not used for tuning the parameters. For convenience, we pick some default parameters for this step, since this step is just about getting a general idea of which model works better. After knowing how each model roughly works, we will pick the best one or two models for the final training and testing.

One note for KNN: KNN uses numeric variables as predictors. Therefore, "type" is removed when using KNN, since it's binary. When "type" was included, the code didn't work.

Part 2 Regression

For comparison, we also use regression models to predict the quality.

We first do linear regression on three datasets: the dataset for white wine, red wine, and the combined dataset (containing both white and red wine). In each dataset we create a linear model with quality as the response variable and all other variables as predictors. Then, we calculate the training and test MSE for each model. After that, we create a bagging model and compute the test MSE for this model. Next, we calculate variable importance and create a variable importance plot. Lastly, we make a correlation matrix for both white wine and red wine datasets.

Results:

The cross validation accuracy rate of all models are as follows:

- Logistic Regression: 0.5432214
- Random Forest: 0.6604858
- Boosting: 0.5679767
- LDA: 0.5298893
- QDA: NA
- KNN: 0.5407512

QDA is not working because there are probably some folds with too few of at least one level of quality. For example, only one or zero level 5 in a fold. QDA needs calculation of covariance matrices of the levels and is too sensitive, so we decide to forfeit it entirely.

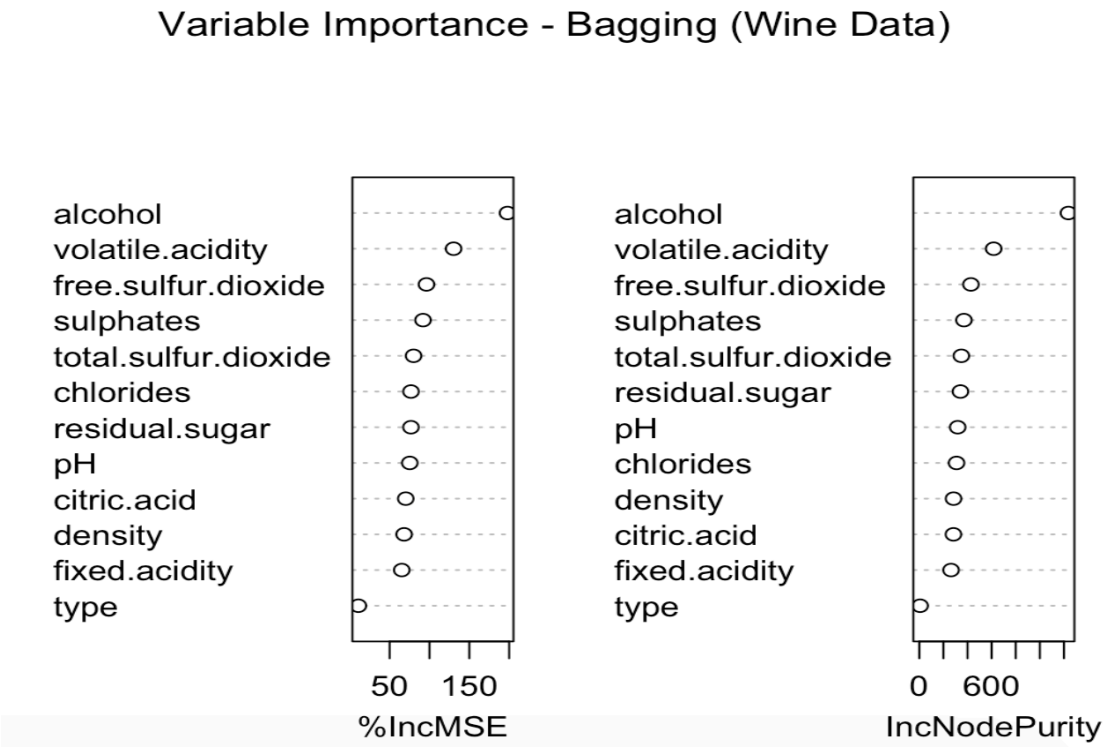
Random Forest is apparently the best model, significantly better than the rest.

A Random Forest is trained using the training set, and tested with the test set. One change is that we raise ntree to 200 in the final training, since in RF, larger ntree doesn't lead to overfitting or other problems. It is not too much computation, and it can potentially give a better model.
Final RF accuracy rate: 0.6738462

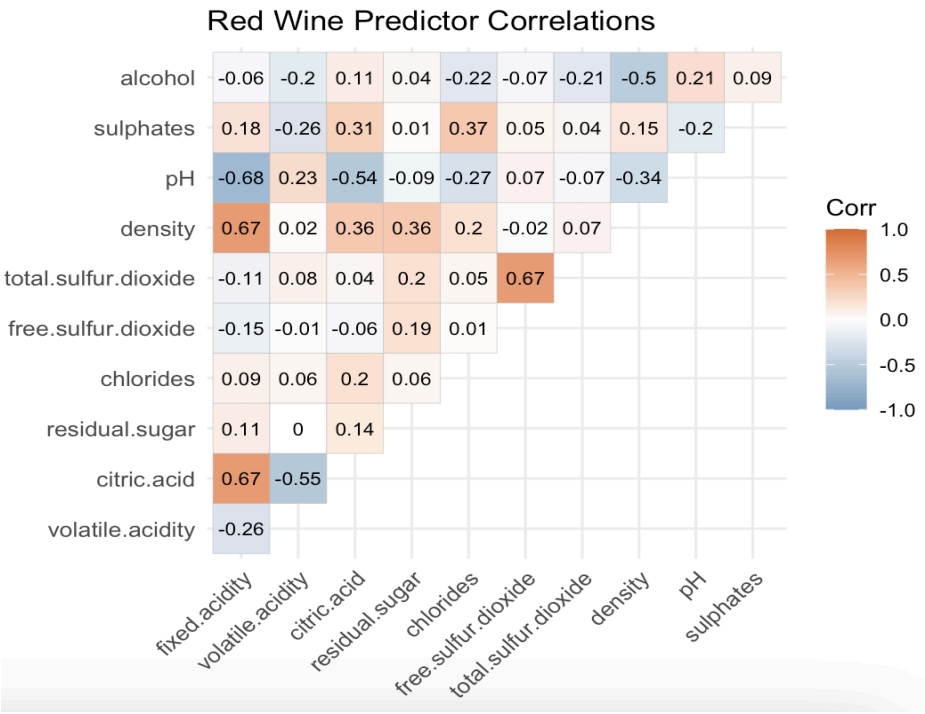
The test MSE for regression methods are as follows:

- Linear regression for the white wine dataset: 0.5588385
- Linear regression for the red wine dataset: 0.4344044
- Linear regression for the combined dataset: 0.5450086
- Bagging for the combined dataset: 0.3845191

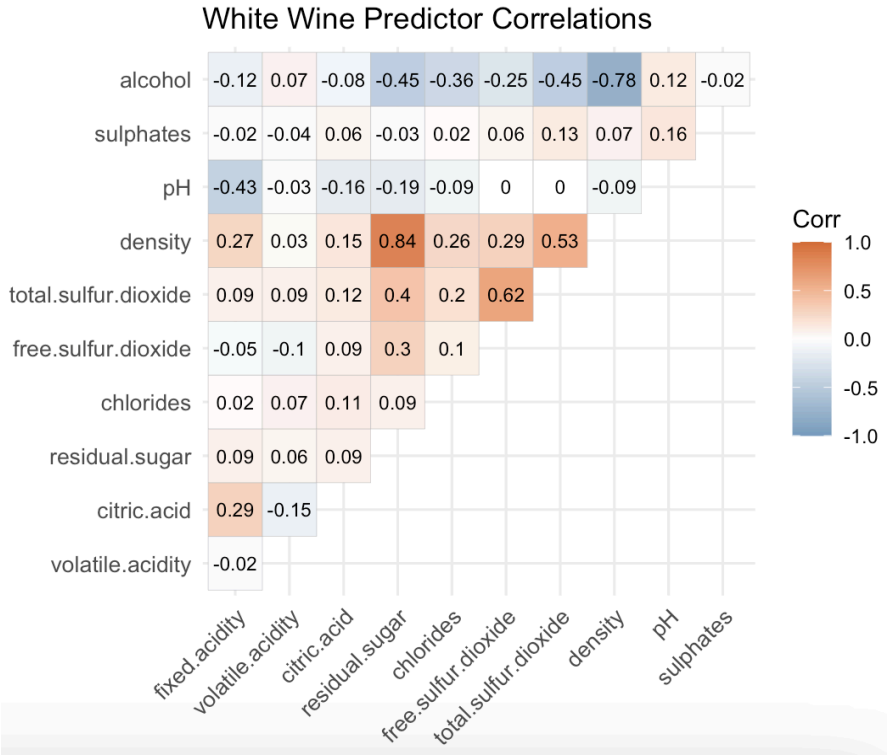
The most important variables for higher wine quality in decreasing order are Alcohol, volatile.acidity, free.sulfur.dioxide, sulphates, residual.sugar, chlorides, total.sulfur.dioxide, citric.acid, pH, fixed.acidity, density, type:



Correlation matrix between predictors for red wine dataset shows that some variables with strong positive correlation are: density and fixed.acidity, citric.acid and fixed.acidity, free.sulfur.dioxide and total.sulfur.dioxide. Some variables with strong negative correlation are pH and fixed.acidity, volatile.acidity and citric.acid, pH and citric.acid:



Correlation matrix between predictors for white wine dataset shows that some variables with strong positive correlation are: density and residual.sugar, free.sulfur.dioxide and total.sulfur.dioxide, density and total.sulfur.dioxide. Some variables with strong negative correlation are alcohol and density, alcohol and total.sulfur.dioxide, alcohol and residual.sugar:



Conclusion:

In this project, we apply multiple machine learning classification models to predict the quality of Portuguese "Vinho Verde" wines using physicochemical properties. After preprocessing and scaling the data, we decide to test 6 different models using 5-fold cross validation: logistic regression, random forest, boosting, LDA, QDA, and KNN.

Among these models, we find that random forest performs the best with a cross-validation accuracy of 66.05%, significantly outperforming the others. When evaluated on the test set, the final random forest model achieves an accuracy of 67.79%. While apparently this is not an ideal accuracy, it is actually very acceptable given that this is the rate of getting a precise prediction of this very subjective integer quality, which means that a prediction is not included in the accuracy even if it is only one level higher or lower than the actual quality. Our results show that models like random forest are especially effective when dealing with imbalanced datasets that have numerous classes and a wide range of feature values. In contrast, simpler models like logistic regression and LDA were able to provide baseline results, but struggled to capture more complex patterns in the data due to their lack of flexibility.

We also find that bagging significantly outperformed linear regression, as the test MSE of bagging for the combined dataset is 0.38 while the test MSE of linear regression for the combined dataset is 0.55. According to the bagging model, the most significant variables in predicting wine quality in decreasing order are Alcohol, volatile.acidity, free.sulfur.dioxide. The most strongly correlated variables are density and residual.sugar in white wine and there was a tie in correlation for white wine between density and fixed.acidity, citric.acid and fixed.acidity and free.sulfur.dioxide and total.sulfur.dioxide. The results show that regression methods are also applicable in this case. The choice of classification or regression in application would depend on the actual need.

Our analysis confirms that machine learning models are able to effectively predict wine quality based solely on physicochemical factors, even in the absence of other variables such as grape type, brand, price, etc. In the future, we could explore more sophisticated methods such as hyperparameter tuning, deep learning methods or SVM in order to improve accuracy and overall performance.

References:

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). *Modeling wine preferences by data mining from physicochemical properties*. Decision Support Systems, 47(4), 547–553. https://doi.org/10.1016/j.dss.2009.05.016

Appendix/Codes:

Links for complete, knitted documents of the code, to show how effective it runs and reproducible:

Link for Classification Methods ML Final Project Code.pdf

Link for Regression Methods Final-Project-Regression-3740.pdf

Below are all the codes:

```

Data Preprocessing

```{r}
library(readr)
library(xgboost)
library(nnet)
library(MASS)
library(randomForest)
library(caret)
library(class)
library(gbm)
library(dplyr)
wine <- read_csv("wine-quality-white-and-red.csv")
p <- ncol(wine) -1
n <- nrow(wine)
unique(wine$quality) # This shows that quality takes value from 3 to 9
```
```

wine$type <- trimws(wine$type) # Remove all the spaces in the column, so I could let R read the string and do stuff, like converting it into bernoulli.

wine$type <- ifelse(wine$type == "white", TRUE, FALSE) # Convert the wine type into binary categorical variables true or false, white being true and red being false.

num_cols <- setdiff(names(wine), c("quality", "type"))
wine[num_cols] <- scale(wine[num_cols]) # These two lines scale the predictors except "type" into values with mean of 0 and sd of 1.

names(wine) <- make.names(names(wine)) # This is prepared for random forest
```

Splitting dataset into training and test set.
```{r}
set.seed(1)
train_index <- sample(1:n, size = 0.7 * n) # randomly pick indexes for training set
wine_train <- wine[train_index, ]
wine_test <- wine[-train_index, ]

sum(wine$type)/n # 0.7538864, proportion of white in original dataset
sum(wine_train$type)/(0.7*n) #  0.7500165, proportion of white in train set
sum(wine_test$type)/(0.3*n) #  0.7629162, proportion of white in test set
# These three lines are to make sure that the split is roughly equal for white and red, so that there would be a set with white way more than red. It turns out the proportion is very close to the original proportion, which is good!
```
Fit a Logistic Regression Model
```{r}

```
logit_model <- multinom(quality ~ . - quality, data = wine_train)
# Cross Validation
set.seed(1)
k <- 5
folds <- sample(1:k, nrow(wine_train), replace = TRUE)
cv_accuracy <- rep(NA, k)
for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
  valid_fold <- wine_train[folds == j, ]

  model_fold <- multinom(quality ~ . - quality, data = train_fold)

  preds <- predict(model_fold, newdata = valid_fold)

  cv_accuracy[j] <- mean(preds == valid_fold$quality)
}
mean(cv_accuracy) # 0.5432214
```

Fit a Random Forest Model

```{r}
set.seed(1)
k <- 5
folds <- sample(1:k, nrow(wine_train), replace = TRUE)

cv_accuracy_rf <- rep(NA, k)

for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
```

```
  valid_fold <- wine_train[folds == j, ]

  rf_model <- randomForest(as.factor(quality) ~ . - quality, data = train_fold, ntree = 100)

  preds <- predict(rf_model, newdata = valid_fold)

  cv_accuracy_rf[j] <- mean(preds == valid_fold$quality)
}

mean(cv_accuracy_rf) # 0.6604858
```

Fit a Boosting Model

```{r}
set.seed(1)
cv_accuracy_boost <- rep(NA, k)

for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
  valid_fold <- wine_train[folds == j, ]

  train_fold$type_factor <- as.factor(train_fold$type)
  valid_fold$type_factor <- as.factor(valid_fold$type)

  train_fold$quality_gbm <- as.numeric(as.factor(train_fold$quality)) - 1
  valid_fold$quality_gbm <- as.numeric(as.factor(valid_fold$quality)) - 1

  gbm_model <- gbm(quality_gbm ~ . - quality - quality_gbm - type + type_factor,
```

```
                data = train_fold,
                distribution = "multinomial",
                n.trees = 100,
                interaction.depth = 3,
                shrinkage = 0.1,
                n.minobsinnode = 10,
                verbose = FALSE)

  preds_prob <- predict(gbm_model, newdata = valid_fold, n.trees = 100, type = "response")
  preds <- apply(preds_prob, 1, which.max) - 1

  cv_accuracy_boost[j] <- mean(preds == valid_fold$quality_gbm)
}

mean(cv_accuracy_boost) # 0.5679767


```

Fit a LDA Model

```{r}


set.seed(1)
k <- 5
folds <- sample(1:k, nrow(wine_train), replace = TRUE)

cv_accuracy_lda <- rep(NA, k)

for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
```

```
  valid_fold <- wine_train[folds == j, ]

  lda_model <- lda(as.factor(quality) ~ . - quality, data = train_fold)

  preds <- predict(lda_model, newdata = valid_fold)$class

  cv_accuracy_lda[j] <- mean(preds == valid_fold$quality)
}
mean(cv_accuracy_lda) # 0.5298893
```

Fit a QDA model

```{r}


set.seed(1)
k <- 5
folds <- sample(1:k, nrow(wine_train), replace = TRUE)

cv_accuracy_qda <- rep(NA, k)

for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
  valid_fold <- wine_train[folds == j, ]

  qda_model <- qda(as.factor(quality) ~ . - quality, data = train_fold)

  preds <- predict(qda_model, newdata = valid_fold)$class

  cv_accuracy_qda[j] <- mean(preds == valid_fold$quality)
}
```

mean(cv_accuracy_qda) # There is this error indicating some groups is too small for qda.
```
QDA calculates covariance matrix for every level. There are a lot of levels of qualities, so probably one level appeared very few in a fold. QDA is too sensitive for this project. QDA is forfeit.

Do KNN
```{r}

set.seed(1)
k <- 5
folds <- sample(1:k, nrow(wine_train), replace = TRUE)

cv_accuracy_knn <- rep(NA, k)

for (j in 1:k) {
  train_fold <- wine_train[folds != j, ]
  valid_fold <- wine_train[folds == j, ]


 train_X <- train_fold[, !(names(train_fold) %in% c("quality", "type"))]
valid_X <- valid_fold[, !(names(valid_fold) %in% c("quality", "type"))]
# Type is removed because KNN is based on numeric predictors
  train_y <- as.factor(train_fold$quality)

  preds <- knn(train_X, valid_X, train_y, k = 5)

  cv_accuracy_knn[j] <- mean(preds == valid_fold$quality)
}
```
```

```
mean(cv_accuracy_knn) # 0.5407512
```

For all of the models, Random Forest is apparently the best. Its cv accuracy is significantly higher than the rest. Therefore, RF is the final model.

Fit the final random forest model with the entire training set

```{r}
set.seed(1)
final_rf_model <- randomForest(as.factor(quality) ~ . - quality,
                               data = wine_train,
                               ntree = 200)

rf_preds_test <- predict(final_rf_model, newdata = wine_test)
RF_test_accuracy <- mean(rf_preds_test == wine_test$quality)
RF_test_accuracy # The final RF test accuracy rate is 0.6779487

```

Regression Code:

```
wine_white = read.csv("winequality-white.csv")
wine_red = read.csv("winequality-red.csv")
```

```{r}
names(wine_white)
names(wine_red)
```

White Wine Model and training MSE

```{r}
model_white = lm(quality ~ fixed.acidity+volatile.acidity+citric.acid+
                 residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
                 density+pH+sulphates+alcohol, data = wine_white)
summary(model_white)
pred_white = predict(model_white, newdata = wine_white)
mse_white = mean((wine_white$quality - pred_white)^2)
print(mse_white)
```

White Wine Train-Test Split and test MSE

```{r}
set.seed(1)
n = nrow(wine_white)
train_index = sample(1:n, size = 0.7*n)
white_train <- wine_white[train_index, ]
white_test <- wine_white[-train_index, ]
white_model = lm(quality ~ fixed.acidity+volatile.acidity+citric.acid+
                 residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
                 density+pH+sulphates+alcohol, data = white_train)
pred_white = predict(white_model, newdata=white_test)
testmse_white = mean((white_test$quality - pred_white)^2)
print(testmse_white)
```

Red Wine Model and training MSE

```{r}
model_red = lm(quality ~ fixed.acidity+volatile.acidity+citric.acid+
```

```
        residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
        density+pH+sulphates+alcohol, data = wine_red)
summary(model_red)
pred_red = predict(model_red, newdata = wine_red)
mse_red = mean((wine_red$quality - pred_red)^2)
print(mse_red)
```

Red Wine Train-Test Split and test MSE

```{r}
set.seed(1)
n = nrow(wine_red)
train_index = sample(1:n, size = 0.7*n)
red_train <- wine_red[train_index, ]
red_test <- wine_red[-train_index, ]
red_model = lm(quality ~ fixed.acidity+volatile.acidity+citric.acid+
        residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
        density+pH+sulphates+alcohol, data = red_train)
pred_red = predict(red_model, newdata=red_test)
testmse_red = mean((red_test$quality - pred_red)^2)
print(testmse_red)
```

Combined Wine Data and training MSE

```{r}
wine_white_red = read.csv("wine-quality-white-and-red.csv")
model_combined <- lm(quality ~ ., data = wine_white_red)
summary(model_combined)
pred_combined = predict(model_combined, newdata = wine_white_red)
```

```
mse_combined = mean((wine_white_red$quality - pred_combined)^2)
print(mse_combined)
```

Combined Train-Test and test MSE

```{r}
set.seed(1)
n = nrow(wine_white_red)
train_index = sample(1:n, size = 0.7*n)
combined_train <- wine_white_red[train_index, ]
combined_test <- wine_white_red[-train_index, ]
combined_model = lm(quality ~ ., data = combined_train)
pred_combined = predict(combined_model, newdata=combined_test)
testmse_combined = mean((combined_test$quality - pred_combined)^2)
print(testmse_combined)
```

Bagging and variable importance

```{r}
library(tree)
library(randomForest)
set.seed(1)
bag_wine = randomForest(quality ~ ., data = wine_white_red, mtry = 12, importance = TRUE)
importance(bag_wine)
varImpPlot(bag_wine, main = "Variable Importance - Bagging (Wine Data)")
```

Bagging Train-Test and training MSE and test MSE

```{r}
set.seed(1)
n <- nrow(wine_white_red)
train_index <- sample(1:n, size = 0.7 * n)
train_data <- wine_white_red[train_index, ]
test_data =  wine_white_red[-train_index, ]
bag_model <- randomForest(quality ~ ., data = train_data, mtry = 12, importance = TRUE)
train_preds = predict(bag_model, newdata = train_data)
test_preds = predict(bag_model, newdata = test_data)
train_mse = mean((train_data$quality - train_preds)^2)
test_mse = mean((test_data$quality - test_preds)^2)
print(train_mse)
print(test_mse)
```

Correlation Plots

```{r}
red_predictors <- wine_red[, !(names(wine_red) %in% c("quality"))]
white_predictors <- wine_white[, !(names(wine_white) %in% c("quality"))]
cor_red <- cor(red_predictors)
cor_white <- cor(white_predictors)
```

```{r}
install.packages("ggcorrplot")
library(ggcorrplot)
```

```{r}
# Red wine correlation matrix
```

```
ggcorrplot(cor_red, method = "square", type = "upper", lab = TRUE, lab_size = 3,
        title = "Red Wine Predictor Correlations",
        colors = c("#6D9EC1", "white", "#E46726"), tl.cex = 10,
        ggtheme = ggplot2::theme_minimal())

# White wine correlation matrix
ggcorrplot(cor_white, method = "square", type = "upper", lab = TRUE, lab_size = 3,
        title = "White Wine Predictor Correlations",
        colors = c("#6D9EC1", "white", "#E46726"), tl.cex = 10,
        ggtheme = ggplot2::theme_minimal())
```