In [ ]:

In [ ]:
```
red_raw <- read.csv("winequality-red.csv")
white_raw <- read.csv("winequality-white.csv")
# Add class labels and merge
red_class <- red_raw
white_class <- white_raw
red_class["class"] = 0
white_class["class"] = 1
combined_class <- rbind(red_class,white_class)
```

Goal 1. Distinguish white wine from red wine

(1a) Testing the means

Standardizating the attributes

In [ ]:
```
combined_standardized <- combined_class
combined_standardized[,1:11] <- scale(combined_class[1:11])
head(combined_standardized)
```

Comparing mean vectors

In [ ]:
```
library(Hotelling)
```

$H_0 : \mu_{red} = \mu_{white}$

$H_A : \mu_{red} \neq \mu_{white}$

In [ ]:
```
fit <- hotelling.test(combined_standardized[combined_standardized$class == 0,1
:11], combined_standardized[combined_standardized$class == 1,1:11])
cat("Hotelling's P-value:", fit$pval)
cat("\nReject Null:", fit$pval<0.05)
```

The mean vectors are different.

Attribute with biggest difference

In [ ]:
```
library(MASS)
```

In [ ]:
```
LDA <- lda(combined_standardized[,1:11], grouping = combined_standardized$clas
s)
t(LDA$scaling)
```

Density has the biggest LDA coefficient <=> biggest difference in mean.

(1b) Classification

Split data into train/CV/test and standardize using train's mean and std

In [ ]:
```
library(caret)
```

In [ ]:
```
set.seed(1995) #randomization`

#creating indices
trainIndex <- createDataPartition(combined_class$pH,p=0.8,list=FALSE)

#splitting data into training/testing data using the trainIndex object
trainCombined <- combined_class[trainIndex,1:11] #training data (80% of data)
classesTrainCombined <- combined_class[trainIndex,13]

test <- combined_class[-trainIndex,1:11] #testing data (20% of data)
classesTest <- combined_class[-trainIndex,13]

set.seed(5) #randomization`

#spliting trainCombined into train and CV for hyperparameter tuning
cvIndex <- createDataPartition(trainCombined$pH,p=0.25,list=FALSE)

cv <- trainCombined[cvIndex,]
classesCV <- classesTrainCombined[cvIndex]

train <- trainCombined[-cvIndex,]
classesTrain <- classesTrainCombined[-cvIndex]
```

In [ ]:
```
sd_train <- apply(train,2,sd)
mean_train <- apply(train,2,mean)

trainStd <- sweep(sweep(train, 2L, mean_train), 2, sd_train, "/")
cvStd <- sweep(sweep(cv, 2L, mean_train), 2, sd_train, "/")
testStd <- sweep(sweep(test, 2L, mean_train), 2, sd_train, "/")
```

K-nearest neighbor

In [ ]:
```
library(class)
```

In [ ]:
```
kBest <- 1
error <- 1.1
for (i in 1:20){
    predictedCV <- as.numeric(as.character(knn(trainStd, cvStd, cl=classesTrai
n, k = i)))
    if (sum(abs(predictedCV-classesCV))/length(predicted)<error){
        error <- sum(abs(predictedCV-classesCV))/length(predicted)
        kBest <- i
    }
}
cat("Optimal k =", kBest)
```

In [ ]:
```
predictedTest <- as.numeric(as.character(knn(trainStd, testStd, cl=classesTrai
n, k = 3)))
cat("Apparent Error Rate:", sum(abs(predictedTest-classesTest))/length(predict
edTest))
```

(1c) Clustering

K-means clustering

In [ ]:
```
library(stats)
```

In [ ]:
```
clusters <- kmeans(combined_standardized[,1:11],2)
```

In [ ]:
```
# Gini Impurity for cluster 1
classesC1 <- combined_standardized$class[clusters$cluster == 1]
gini1 <- 1 - (sum(classesC1==0)/length(classesC1))^2 - (1-(sum(classesC1==0)/l
ength(classesC1)))^2
classesC2 <- combined_standardized$class[clusters$cluster == 2]
gini2 <- 1 - (sum(classesC2==0)/length(classesC2))^2 - (1-(sum(classesC2==0)/l
ength(classesC2)))^2
```

In [ ]:
```
cat("Gini Impurity for Cluster 1:", gini1)
cat("\nGini Impurity for Cluster 2:", gini2)
```

Goal 2. Which variable is most important to wine quality

(2a) MANOVA

In [ ]:
```
redStd <- red_raw
redStd[,1:11] <- scale(red_raw[,1:11])
```

$H_0 : \mu_{low} = \mu_{med} = \mu_{high}$

$H_A : \mu_{low} \neq \mu_{med} \neq \mu_{high}$

In [ ]:
```
maov <- summary(manova(cbind(fixed.acidity,volatile.acidity,citric.acid,residu
al.sugar,chlorides,free.sulfur.dioxide,total.sulfur.dioxide,density,pH,sulphat
es,alcohol)~as.factor(quality), data=redStd))
maov
```

Reject the Null.

(2b) Regression

In [ ]:
```
set.seed(1995) #randomization`

#creating indices
trainIndex <- createDataPartition(red_raw$pH,p=0.8,list=FALSE)

#splitting data into training/testing data using the trainIndex object
redTrain <- red_raw[trainIndex,1:12] #training data (80% of data)

redTest <- red_raw[-trainIndex,1:12] #testing data (20% of data)
```

In [ ]:
```
redTrainQuality <- redTrain$quality
redTestQuality <- redTest$quality
sdRedTrain <- apply(redTrain[,1:11],2,sd)
meanRedTrain <- apply(redTrain[,1:11],2,mean)

redTrainStd <- sweep(sweep(redTrain[,1:11], 2L, meanRedTrain), 2, sdRedTrain,
"/")
redTestStd <- sweep(sweep(redTest[,1:11], 2L, meanRedTrain), 2, sdRedTrain,
"/")

redTrainStd['quality'] <- redTrainQuality
redTestStd['quality'] <- redTestQuality
```

Multiple Linear Regression

In [ ]:
```
fitLM <- lm(quality~fixed.acidity+volatile.acidity+citric.acid+residual.sugar+
chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+sulphates+alcoho
l,data=redTrainStd)
fitLM$coef
```

In [ ]:
```
pred <- predict(fitLM,redTestStd[,1:11])
sum((pred-redTestStd$quality)^2)/length(pred)
```

Random Forests

In [ ]:
```
library(randomForest)
```

In [ ]:
```
fitRF <- randomForest(quality~.,data=redTrainStd)
```

In [ ]:
```
pred <- predict(fitRF,redTestStd[,1:11])
sum((pred-redTestStd$quality)^2)/length(pred)
```

(2c) Regression on PCA

In [ ]:
```
loadings <- prcomp(redTrainStd[,1:11])$rotation[,1:2]
```

In [ ]:
```
PCA_train <- as.data.frame(as.matrix(redTrainStd[,1:11])%*%loadings)
PCA_train['quality'] <- redTrainStd$quality
PCA_test <- as.data.frame(as.matrix(redTestStd[,1:11])%*%loadings)
PCA_test['quality'] <- redTestStd$quality
```

In [ ]:
```
fitRFPCA <- randomForest(quality~.,data=PCA_train)
```

In [ ]:
```
predPCA <- predict(fitRFPCA,PCA_test[,1:2])
sum((predPCA-PCA_test$quality)^2)/length(predPCA)
```

In [ ]:
```

```