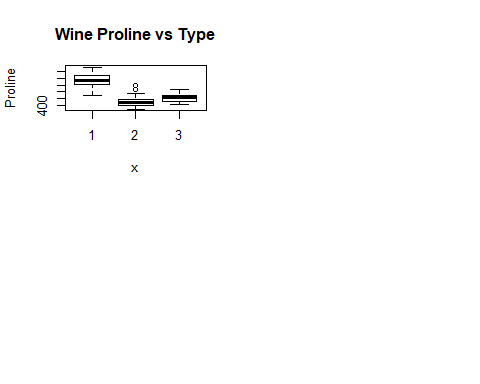
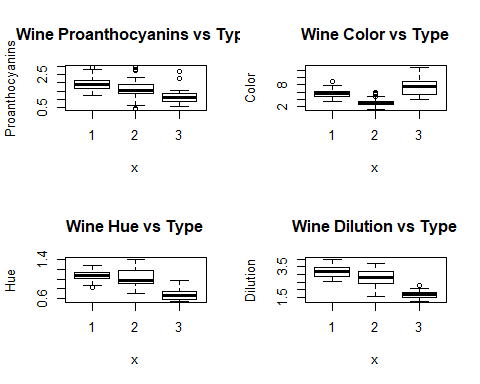
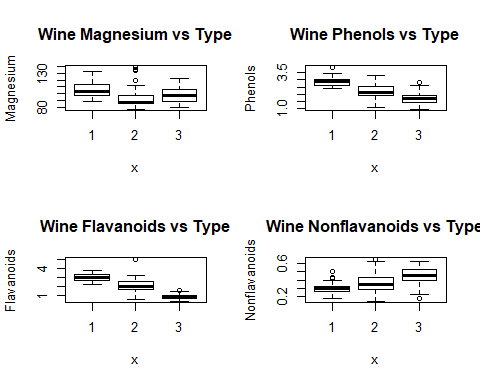
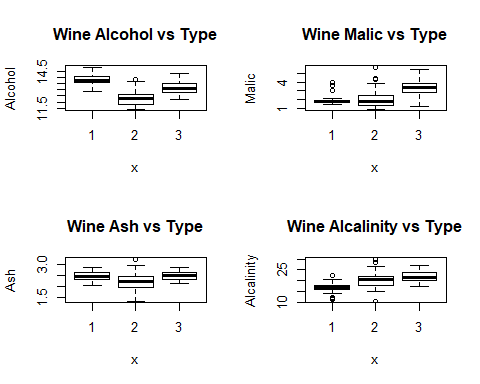
Stats 503: HW #2

Diana Liang

2/16/2020

# 1. Wine Dataset

## a) Explore the data



The most useful variables in predicting Type would have distinct values amongst the different Types. Alcohol, phenols, and flavanoids are almost entirely distinct between the 3 different wine Types and would be the most useful. Nonflavanoids, proanthocyanins, and proline have some overlap between the different Types but are still better than the remaining variables.

## b) LDA, QDA, Naive Bayes

# Load libraries  
library(MASS)  
library(e1071)

## Warning: package 'e1071' was built under R version 3.6.2

# LDA  
wine\_lda = lda(Type~., data=wine\_train)  
lda\_test\_pred = predict(wine\_lda, wine\_test)$class  
lda\_test\_err = mean(lda\_test\_pred != wine\_test$Type)  
  
# QDA  
wine\_qda = qda(Type~., data=wine\_train)  
qda\_test\_pred = predict(wine\_qda, wine\_test)$class  
qda\_test\_err = mean(qda\_test\_pred != wine\_test$Type)  
  
# Naive Bayes  
wine\_nb = naiveBayes(Type~., data=wine\_train)  
nb\_test\_pred = predict(wine\_nb, newdata = wine\_test)  
nb\_test\_err = mean(nb\_test\_pred != wine\_test$Type)

## [1] "LDA test error: 0.018"

## [1] "QDA test error: 0.036"

## [1] "Naive Bayes test error: 0.036"

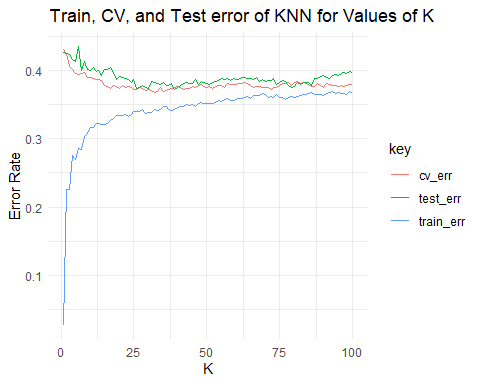
LDA provided the lowest test error of the 3 models while QDA and Naive Bayes provided similar test errors.

# 2. KNN of Theft Dataset

# K-fold CV for KNN  
knn\_cv\_err <- function(cv\_folds, knn\_k, train, train\_label){  
 obs = nrow(train)  
 fold\_size = floor(obs/cv\_folds)  
 cv\_error = rep(0, cv\_folds)  
 for(i in 1:cv\_folds){  
 # choose indices for this fold  
 if(i != cv\_folds){  
 cv\_test\_id = ((i-1)\*fold\_size + 1):(i\*fold\_size)  
 } else{  
 cv\_test\_id = ((i-1)\*fold\_size + 1):obs  
 }  
   
 # create cv train and test for this fold  
 cv\_train = train[-cv\_test\_id,]  
 cv\_test = train[cv\_test\_id,]  
   
 # standardize  
 avg = colMeans(cv\_train); sd = apply(cv\_train,2,sd)  
 cv\_train = scale(cv\_train, center = avg, scale = sd)  
 cv\_test = scale(cv\_test, center = avg, scale = sd)  
   
 # run KNN and find cv error  
 pred = knn(cv\_train, cv\_test, train\_label[-cv\_test\_id], k = knn\_k)  
 cv\_error[i] = mean(pred != train\_label[cv\_test\_id])  
 }  
 return(mean(cv\_error))  
}

The training error and test error will be calculated for each value of K, which spans 1 to 100. The CV error will also be calculated using the above function. These error values are plotted below.

set.seed(1)  
# make list of K's  
K = 1:100  
  
# set up training error  
train\_err = rep(0, 100)  
avg = colMeans(theft\_train[,-3])  
sd = apply(theft\_train[,-3],2,sd)  
std\_train = scale(theft\_train[,-3], center = avg, scale = sd)  
std\_test = scale(theft\_test[,-3], center = avg, scale = sd)  
  
# set up test error  
test\_err = rep(0, 100)  
  
# set up cv error  
cv\_err = rep(0, 100)  
  
# go through values of K  
for(i in 1:100){  
 pred\_train = knn(std\_train, std\_train, theft\_train$theft, K[i])  
 train\_err[i] = mean(pred\_train != theft\_train$theft)  
 pred\_test = knn(std\_train, std\_test, theft\_train$theft, K[i])  
 test\_err[i] = mean(pred\_test != theft\_test$theft)  
 cv\_err[i] = knn\_cv\_err(10, K[i], theft\_train[,-3], theft\_train$theft)  
}  
  
theft\_err <- data.frame(K, train\_err, cv\_err, test\_err)  
theft\_err <- gather(theft\_err, key, value, train\_err, cv\_err, test\_err)  
ggplot(theft\_err, aes(x = K, y = value, color = key)) +  
 geom\_line() + ylab("Error Rate") + xlab("K") +  
 ggtitle("Train, CV, and Test error of KNN for Values of K") +  
 theme\_minimal()



The best model would be the one with the lowest CV error, so the K that gives the KNN model with the lowest CV error is:

theft\_err[which.min(theft\_err$cv\_err),]

|  |
| --- |
|  |

|  | **K**  <int> | **train\_err**  <dbl> | **cv\_err**  <dbl> | **test\_err**  <dbl> |
| --- | --- | --- | --- | --- |
| 32 | 32 | 0.3425714 | 0.3677143 | 0.3833333 |

# 3. Weekly Dataset

## a) Logistic Regression Model to Predict Direction

part\_a = glm(Direction ~ Lag1 + Lag2, data = weekly\_data,   
 family = binomial)  
summary(part\_a)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = weekly\_data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.623 -1.261 1.001 1.083 1.506   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22122 0.06147 3.599 0.000319 \*\*\*  
## Lag1 -0.03872 0.02622 -1.477 0.139672   
## Lag2 0.06025 0.02655 2.270 0.023232 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1496.2 on 1088 degrees of freedom  
## Residual deviance: 1488.2 on 1086 degrees of freedom  
## AIC: 1494.2  
##   
## Number of Fisher Scoring iterations: 4

Both Lag1 and Lag2 seem to have little effect on Direction, but Lag1 does not seem to be significant at a = 0.05 while Lag2 is. Lag1 also influences Direction in the opposite manner as Lag2: greater Lag1 makes the “Up” direction more likely while greater Lag2 makes the “Down” direction more likely.

## b) Log Reg Model for Direction but not the first obs

b\_data = weekly\_data[-1, ]  
part\_b = glm(Direction ~ Lag1 + Lag2, data = b\_data,   
 family = binomial)  
summary(part\_b)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = b\_data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6258 -1.2617 0.9999 1.0819 1.5071   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22324 0.06150 3.630 0.000283 \*\*\*  
## Lag1 -0.03843 0.02622 -1.466 0.142683   
## Lag2 0.06085 0.02656 2.291 0.021971 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1494.6 on 1087 degrees of freedom  
## Residual deviance: 1486.5 on 1085 degrees of freedom  
## AIC: 1492.5  
##   
## Number of Fisher Scoring iterations: 4

This model without the first observation is almost identical to the original.

## c) Predict first obs

pred\_obs = predict(part\_b, weekly\_data[1, -1])  
pred\_p = binomial()$linkinv(pred\_obs)  
pred\_p

## 1   
## 0.5713923

if (pred\_p > 0.5){ pred\_label = "Up" } else{ pred\_label = "Down"}  
first\_obs <- data.frame(Predicted = pred\_label,  
 Actual = weekly\_data[1, 9])  
first\_obs

## Predicted Actual  
## 1 Up Down

The observation was not correctly classified.

## d) LOOCV

obs = nrow(weekly\_data)  
loocv\_test\_err = rep(0, obs)  
for(i in 1:obs){  
 temp = glm(Direction ~ Lag1 + Lag2, data = weekly\_data[-i,], family = binomial)  
 pred\_obs = predict(temp, weekly\_data[i, -1])  
 pred\_p = binomial()$linkinv(pred\_obs)  
 if(pred\_p > 0.5){   
 pred\_label = "Up"  
 } else{   
 pred\_label = "Down"  
 }  
 if(pred\_label != weekly\_data[i, 9]){  
 loocv\_test\_err[i] = 1  
 } else{  
 loocv\_test\_err[i] = 0  
 }  
}

## e) LOOCV test error

mean(loocv\_test\_err)

## [1] 0.4499541

The test error is incredibly high, in which almost half of the observations were predicted to have the wrong Direction label. This means Lag1 and Lag2 are not the best variables to predict Direction.