STATS503 - HW2

Wenjing Li

2/16/2020

Question 1

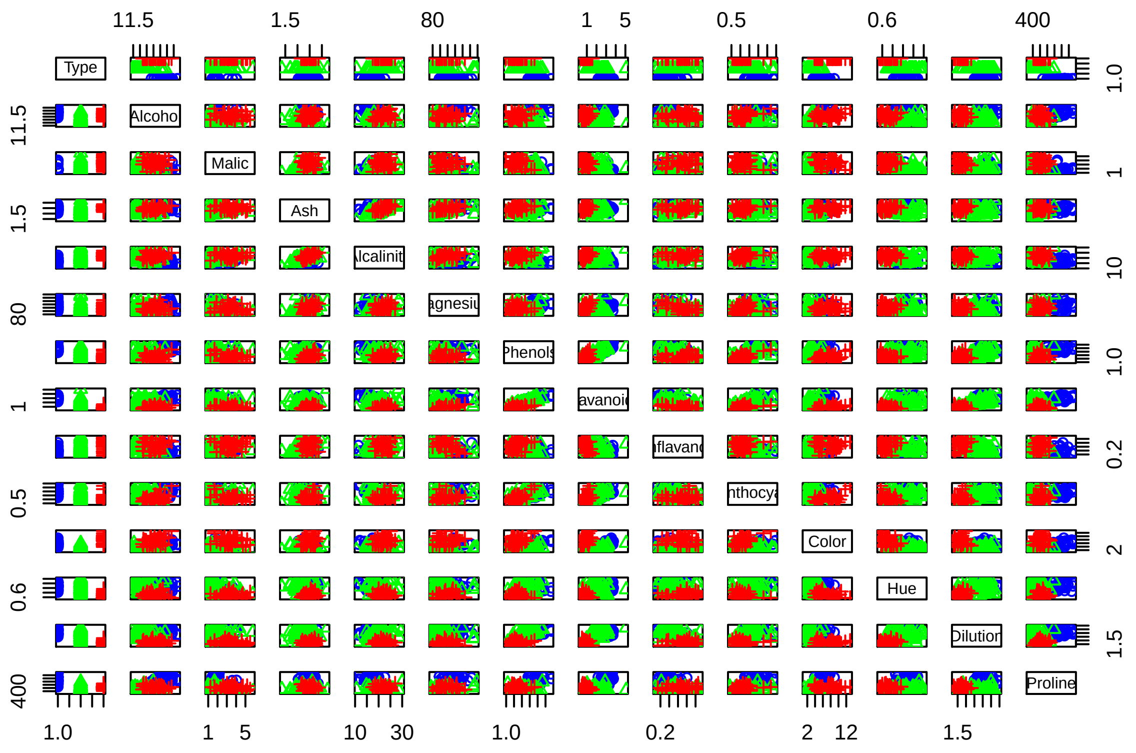
The wine dataset contains the results of a chemical analysis of wines grown in a specific area of Italy. This data set contains 178 observations of 14 variables. The data contains no missing values and consists of only numeric data, with a three-class target variable (Type) for classification.

We are interested in the prediction of the wine type.

1(a)

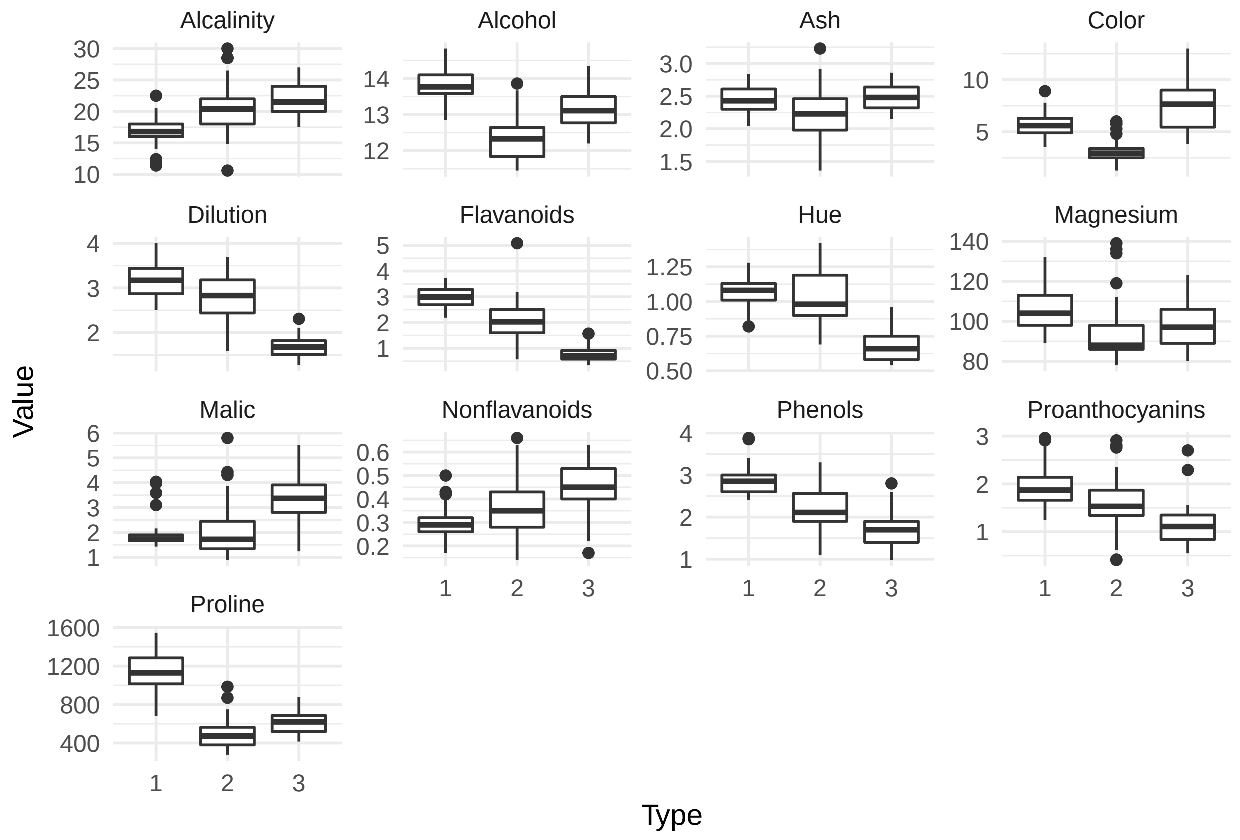
Take a look at the data set at first.

```
#generates scatter plots
par(mfrow=c(1,2))
pairs(dat[1:14], col=c("blue", "green","red")[dat$Type],
      pch=c(1,2,3)[dat$Type])
```



```r
par(xpd=TRUE)

#generates boxplots
dat_long = gather(dat, key = "Variable", value = "Value", -"Type")
ggplot(dat_long) + geom_boxplot(aes(x=Type, y = Value)) +
  facet_wrap(.~Variable, scales="free_y") + theme_minimal()
```



The scatter plots use three different colors to represent three types. From the scatter plots above, we can find that the predictors Alcalinity, Phenols, Flavanoids, Proanthocyanins, Color, Dilution, and Proline behave differently in each type. This also means they tend to be useful in predicting type.

Checking the first boxplot graph, we can find the predictor Proline has value way larger than all the other predictors, The predictor Phenols also has the same problem. So I deleted these two predictors and take a closer look at others.

From the second plot, we can confirm that there are no outliers nor leverage problem we need to concern before fitting models. The predictor Ash tends to have leverage, but since it behaves almost the same among different types in the scatter plot and will not influence the prediction too much. We can ignore the problem.

1(2)

Next, I trained three models, LDA, QDA, and Naive Bayes respectively, on the training data to predict type. And get the error table as below.

```r
#LDA
##trains LDA model using training data
lda_train = lda(dat$Type ~ ., data = dat)

##tests LDA model using testing data and training data
lda_train_pred = predict(lda_train, dat)$class
lda_test_pred = predict(lda_train, dat_test)$class
lda_train_err = mean(lda_train_pred != dat$Type)
lda_test_err = mean(lda_test_pred != dat_test$Type)

#QDA
##trains QDA model using training data
qda_train = qda(dat$Type ~ ., data = dat)

##tests QDA model using testing data and training data
qda_train_pred = predict(qda_train, dat)$class
qda_test_pred = predict(qda_train, dat_test)$class
qda_train_err = mean(qda_train_pred != dat$Type)
qda_test_err = mean(qda_test_pred != dat_test$Type)

#Naive Bayes
##trains NB model using training data
NBclassfier = naiveBayes(Type ~ ., data = dat)

##tests NB model using testing data and training data
nb_train_pred = predict(NBclassfier, newdata = dat)
nb_test_pred = predict(NBclassfier, newdata = dat_test)
nb_train_err = mean(nb_train_pred != dat$Type)
nb_test_err = mean(nb_test_pred != dat_test$Type)

#create table
table_error = c(lda_train_err, lda_test_err, qda_train_err, qda_test_err,
                nb_train_err, nb_test_err)
dim(table_error) = c(2,3)
colnames(table_error) = c("LDA", "QDA", "NB")
rownames(table_error) = c("Training Error", "Testing Error")
cap = paste("*Testing and Training Error for 3 Methods*")
knitr::kable(table_error, caption = cap)
```

Table 1: *Testing and Training Error for 3 Methods*

	LDA	QDA	NB
Training Error	0.0000000	0.0000000	0.0162602
Testing Error	0.0181818	0.0363636	0.0363636

Training errors are smaller than testing errors among these three models. And the LDA model behaves best with testing error 0.018. QDA and NB model have the same performance on the testing set while QDA behaves better at the training data set.

Question 2

The data set provides nearly 12 years of crime reports from across all of San Francisco's neighborhoods. Given time and location, we can predict the category of crime that occurred.

The data frame contains 5,000 observations of 4 variables, with no missing values and only numeric data, and a class target variable (Theft) for classification.

I used 10-fold cross-validation and evaluate the performance of the best KNN model.

```r
#defines function to compute K-fold CV on training set
Kfold_CV_knn <- function(K,K_knn,train,train_label){
  fold_size = floor(nrow(train)/K)
  cv_error = rep(0,K)
  for(i in 1:K) {
    if(i != K) {
      CV_test_id = ((i-1)*fold_size+1):(i*fold_size)
    }else{
      CV_test_id = ((i-1)*fold_size+1):nrow(train)
    }
    CV_train = train[-CV_test_id,]
    CV_test = train[CV_test_id,]
    # calculate the mean and standard deviation using CV_train
    mean_CV_train = colMeans(CV_train)
    sd_CV_train = apply(CV_train,2,sd)
    # normalize the CV_train and CV_test using above mean and sd
    CV_train = scale(CV_train,center = mean_CV_train,scale = sd_CV_train)
    CV_test = scale(CV_test,center = mean_CV_train,scale = sd_CV_train)
    # Fit knn
    pred_CV_test = knn(CV_train,CV_test,train_label[-CV_test_id],k = K_knn)
    # Calculate CV error by taking averages
    cv_error[i] = mean(pred_CV_test!=train_label[CV_test_id])
  }
  return(mean(cv_error))
}

#trains the model for k=1:150
set.seed(123)
K_fold = 10
K_knn = 1:150
cv_error = rep(0,length(K_knn))
for(i in 1:length(K_knn)){
  cv_error[i] = Kfold_CV_knn(K = K_fold, K_knn = K_knn[i],
                             train = dat[,-3],train_label = dat$theft)
}

#minimum CV error & k value
min_error = min(cv_error)
best_k = which(cv_error == min(cv_error))

#data normalization
mean_train = colMeans(dat[,-3])
sd_train = apply(dat[,-3],2,sd)
std_train = scale(dat[,-3],center = mean_train,scale = sd_train)
std_test = scale(dat_test[,-3],center = mean_train,scale = sd_train)
```

```r
# prediction on test data
pred_test = knn(train = std_train,test = std_test,cl = dat$theft,k=best_k)
test_err = mean(pred_test!=dat_test$theft)
test_err
```

```
## [1] 0.384
```

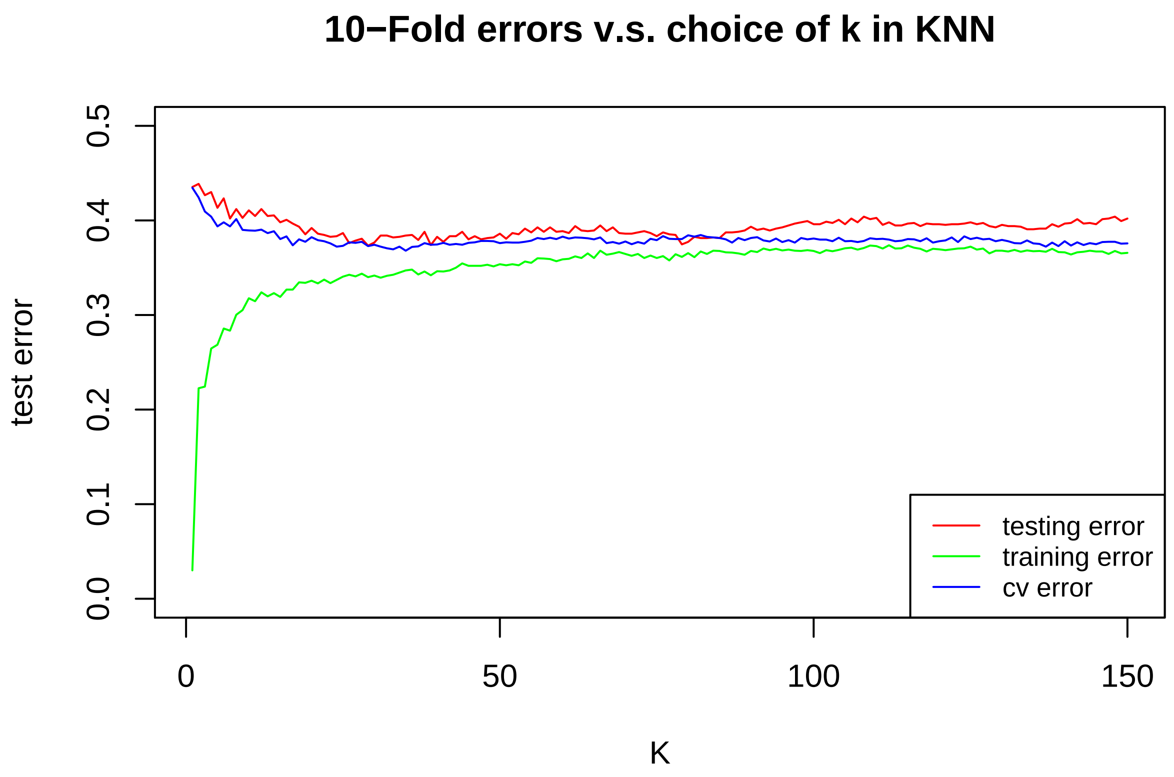
I chose the best model based on the error value and the lower value means the model is better. The best k is 35, and the minimum cv error is 0.368. The testing error when k is equal to 35 is 0.384 and the training error at this time is 0.347. Since these three errors are similar, the model is relatively stable and the model is useable. K = 35 is a good choice.

Next, I generate a plot about the error trend for each choice of k(k range is 1:150, which is large enough).

```r
#generates plots
##generates testing error plot
test_error = rep(0,length(K_knn))
for (i in 1:length(K_knn)) {
   test_error[i] = mean(knn(train = std_train, test = std_test,
                            cl = dat$theft,k=i) != dat_test$theft)
}
colors <- rainbow(3)
plot(test_error~K_knn,type='l',main = '10-Fold errors v.s. choice of k in KNN',
     xlab = 'K',ylab = 'test error',ylim=c(0, 0.5), col=colors[1])

##generates training error plot
train_error = rep(0,length(K_knn))
for (i in 1:length(K_knn)) {
   train_error[i] = mean(knn(train = std_train, test = std_train,
                             cl = dat$theft,k=i) != dat$theft)
}
lines(train_error ~ K_knn, type = 'l', col = colors[2])

##generates cv error plot
lines(cv_error~K_knn, type = 'l', col = colors[3])
legend("bottomright", legend = c('testing error', 'training error', 'cv error'),
       col = colors, lty = 1, cex=0.8)
```



The green line is the training error, which glows rapidly at first and stays stable at about 0.35. CV error and testing error have similar trend and testing error is slightly larger than the CV error which makes sense. These three lines are all stable at about 0.35 - 0.38, so our previous model is a good choice.

Question 3

Weekly is a dataset of S&P 500 stock index returns. Given other predictors, we can predict the return direction on a given week.

3(a)

I used two predictors Lag1 and Lag2 to make the prediction. Logistic Regression may seem a good try.

```
data(Weekly)

#fit logistic model
logistic_model = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
summary(logistic_model)$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)  0.22122405 0.06146572  3.599145 0.0003192652
## Lag1        -0.03872222 0.02621658 -1.477013 0.1396722362
## Lag2         0.06024830 0.02654589  2.269590 0.0232324586
```

From the Logistic Regression above, the predictor Lag2 is significant while the predictor Lag1 is not. AIC is 1494.2. The whole model works not quite well.

3(b)

Next, I used the data except the first observation to make a similar regression. Take the first observation out at first, and then refit the model.

```
#leave-the-first-one-out logistic model
logistic_model_leave1 = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1,],
                            family = binomial)
summary(logistic_model_leave1)$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)  0.22324305 0.06149894  3.630031 0.0002833875
## Lag1        -0.03843317 0.02621860 -1.465874 0.1426825151
## Lag2         0.06084763 0.02656088  2.290874 0.0219707105
```

From the model summary above, the predictor Lag2 is still significant while the predictor Lag1 is still not. AIC is 1492.5, which is smaller than before. The whole model doesn't change too much and still works not quite well.

3(c)

Since the new model doesn't use the information on the first observation. I used the first observation as a cv data set and made a prediction on it.

```
#predicts the direction of the first observation
sum(logistic_model_leave1$coefficients*cbind(1,Weekly[1,2:3]))
```

```
## [1] 0.287534
```

Since the posterior probability for the first observation is 0.287, that means $P(Direction = "Up"|Lag1, Lag2) = 0.287 < 0.5$. The prediction of the first observation is "Down", which is correctly classified.

3(d)

Use a similar method to create a LOOCV by myself. The code is below.

```
error = 0
for (i in 1:dim(Weekly)[1]) {
  logistic_model_leave_one = glm(Direction ~ Lag1 + Lag2,
                                 data = Weekly[-i,], family = binomial)
  test = ifelse(sum(logistic_model_leave_one$coefficients*cbind(1,Weekly[i,2:3]))>0.5,
                1, 0)
  test = factor(test)
  levels(test) = c('Up','Down')
  error = error + ifelse(test == Weekly$Direction[i], 0, 1)
}
```

3(e)

Loop over the dataset and get the average error.

```r
(loocv_error = error/dim(Weekly)[1])
```

```
## [1] 0.4444444
```

The average error is 0.44, which is not good. This means there are about 2 observations being classified wrongly among 5 observations. The bad performance of the prediction may because there is another model that can fit the data set better.