STATS503-HW2

Ningyuan Wang

2/14/2020

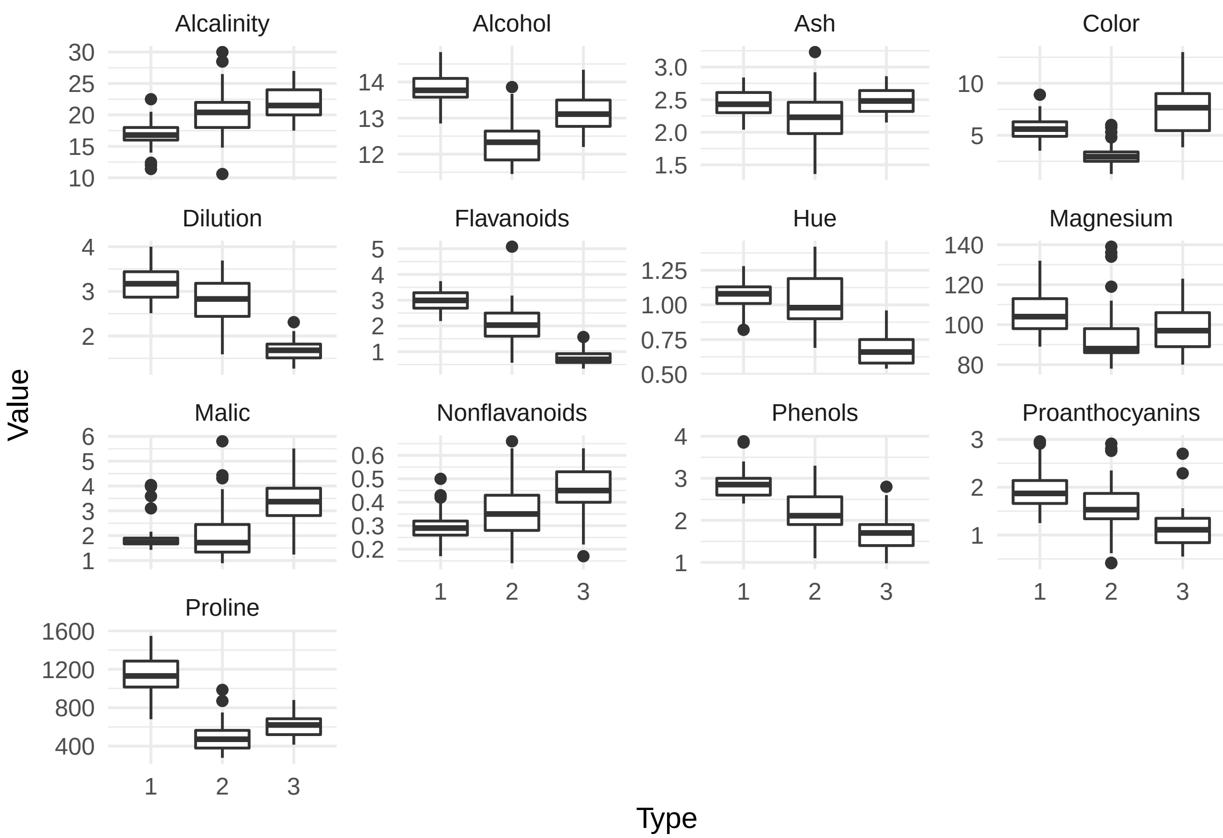
Problem 1

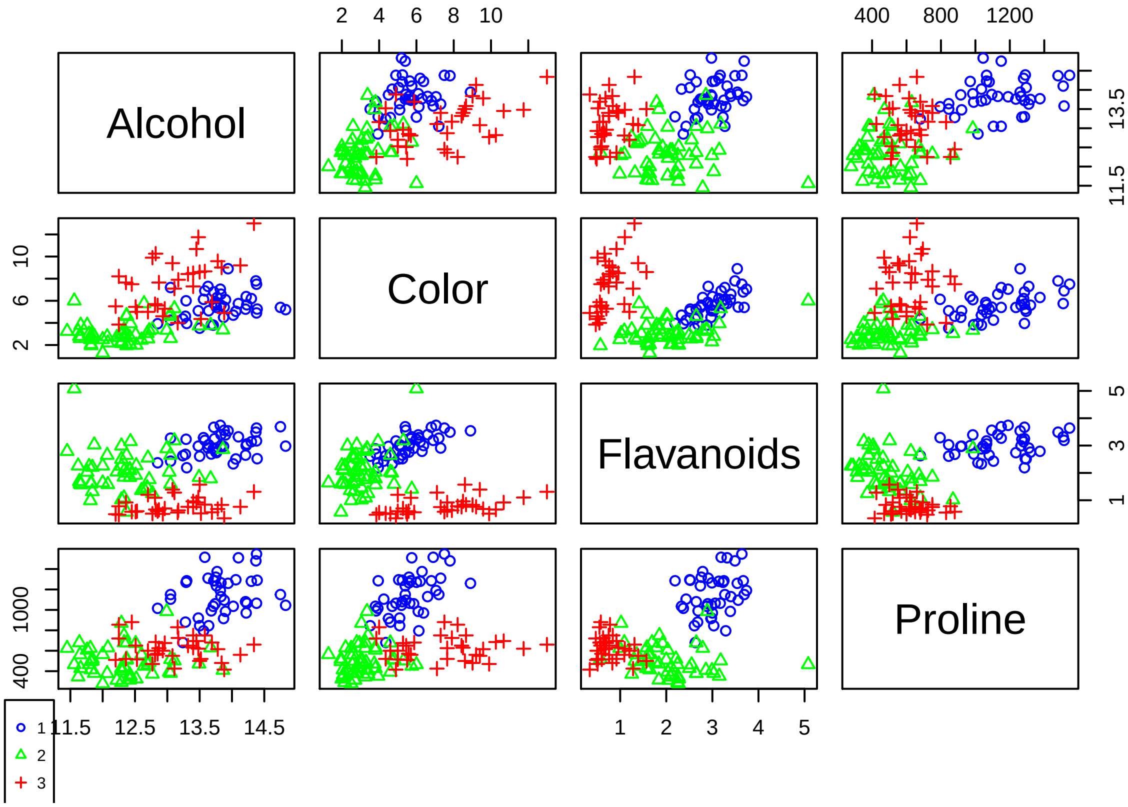
Data Exploration and Visualization

There were 123 observations with 14 variables in the training dataset, and type (3 levels) was the response variable. Based on the boxplots of variables, we notitced obvious difference between three types of wine in the variables below, and therefore, they may be recognized as useful predicotrs: Alcohol, Color, Flavanoids, and Proline. Then, we made scatterplots for those four variables, and the plots showed that the variables may help to recognize the type of wine.

```
# laod the datasets
wine_train = read.csv("wine_train.csv") %>% mutate(Type = as.factor(Type))
wine_test = read.csv("wine_test.csv") %>% mutate(Type = as.factor(Type))

# data exploration
summary(wine_train) # check missing values and data pattern
dim(wine_train)
```





Fit Models (LDA, QDA & Naive Bayes)

We used all variables (except response "Type") in training data as our predicotrs in all three models, because all variables were meaningful.

The test error of LDA model was 0.018.

```
# LDA model
wine_lda = lda(Type ~., data = wine_train)

# prediction
wine_lda_train_pred = predict(wine_lda, wine_train)$class
wine_lda_test_pred = predict(wine_lda, wine_test)$class

# error
lda_train_err = mean(wine_lda_train_pred != wine_train$Type)
lda_test_err = mean(wine_lda_test_pred != wine_test$Type)
lda_test_err
```

```
## [1] 0.01818182
```

The test error of QDA model was 0.593.

```
# QDA model
wine_qda = qda(Type ~ ., data = wine_train)

# prediction
```

```r
wine_qda_train_pred = predict(wine_qda, wine_train)$class
wine_qda_test_pred = predict(wine_qda, wine_test)$class

# error
qda_train_err = mean(wine_qda_train_pred != wine_train$Type)
qda_test_err = mean(wine_qda_train_pred != wine_test$Type)
qda_test_err
```

```
## [1] 0.5934959
```

The test error of Naive Bayes model was 0.036.

```r
# NB model
NBclassfier=naiveBayes(Type~ ., data=wine_train)

# prediction
wine_nb_train_pred = predict(NBclassfier,newdata = wine_train)
wine_nb_test_pred = predict(NBclassfier,newdata = wine_test)

# error
nb_train_err = mean(wine_nb_train_pred!=wine_train$Type)
nb_test_err = mean(wine_nb_test_pred!=wine_test$Type)
nb_test_err
```

```
## [1] 0.03636364
```

According to test errors with three models, we chose LDA as our final model for predicting type of wine, since it had best test error, and it was also the simplest model considering to both degrees of predictors and model assumptions.

Problem 2

Data Exploration and Visualization

There were 3500 observations with 4 variables in the theft training set. "Theft" was treated as the response and the other three were predictors. Boxplots for predictors show below.

```
##        X                 Y            theft            hour      
##  Min.   :-122.5   Min.   :37.71   FALSE:1808   Min.   : 0.00  
##  1st Qu.:-122.4   1st Qu.:37.76   TRUE :1692   1st Qu.:10.00  
##  Median :-122.4   Median :37.78                Median :15.00  
##  Mean   :-122.4   Mean   :37.77                Mean   :13.86  
##  3rd Qu.:-122.4   3rd Qu.:37.79                3rd Qu.:19.00  
##  Max.   :-122.4   Max.   :37.81                Max.   :23.00
```

```
## [1] 3500    4
```

K-fold Cross Validation of KNN model

We applied K-fold cross validation to help select best k as our prediction model. Finally, we found best k was 33 with cross validation error at 0.369. In terms of model performance, based on the plot of training error, CV error and testing error, we think k = 33 was convenienced to be a good model considering to model complexity and error rate. Also, since this was a psudo data (i.e. we can compute the teting error), after computing testing error, we found the testing error with k = 33 was among one of lowest testing error rate as well. The table of training error, testing error and CV error shows below as well (due to page limitation, we only show 5 rows).

```
# scale both training and test data in the same way
mean_train = colMeans(train_x)
std_train = sqrt(diag(var(train_x)))
# training data
train_x = scale(train_x, center = mean_train, scale = std_train)
# test data
test_x = scale(test_x, center = mean_train, scale = std_train)

#  training  and testing error for earch k
K_knn = 1:150
train_error = c()
test_error = c()

for(i in 1:length(K_knn)){
  pred_train <- knn(train_x,
                    train_x,
                    train_label,
                    k = K_knn[i])
  train_error[i] = mean(pred_train != train_label)
  pred_test = knn(train_x,
                  test_x,
                  train_label,
                  k = K_knn[i])
  test_error[i] = mean(pred_test != test_label)
}

errors = data.frame(train_error, test_error, K_knn)
```

```
# CV error
Kfold_CV_knn <- function(K,K_knn,train,train_label){
  # Arguments: K- the number of fold, K_knn- the number of k in KNN
  fold_size = floor(nrow(train)/K)
  cv_error = rep(0,K)
```

```r
  for(i in 1:K){
    # iteratively select K-1 folds as training data in CV procedure, remaining as test data.
    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)

    cv_error[i] = mean(pred_CV_test!=train_label[CV_test_id])
    # Calculate CV error by taking averages
    cv_error[i] = mean(pred_CV_test!=train_label[CV_test_id])
  }
  return(mean(cv_error))
}


# compute errors
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 = theft_train[,-3],train_label = theft_
}


min(cv_error) #0.369
```

```
## [1] 0.3694286
```

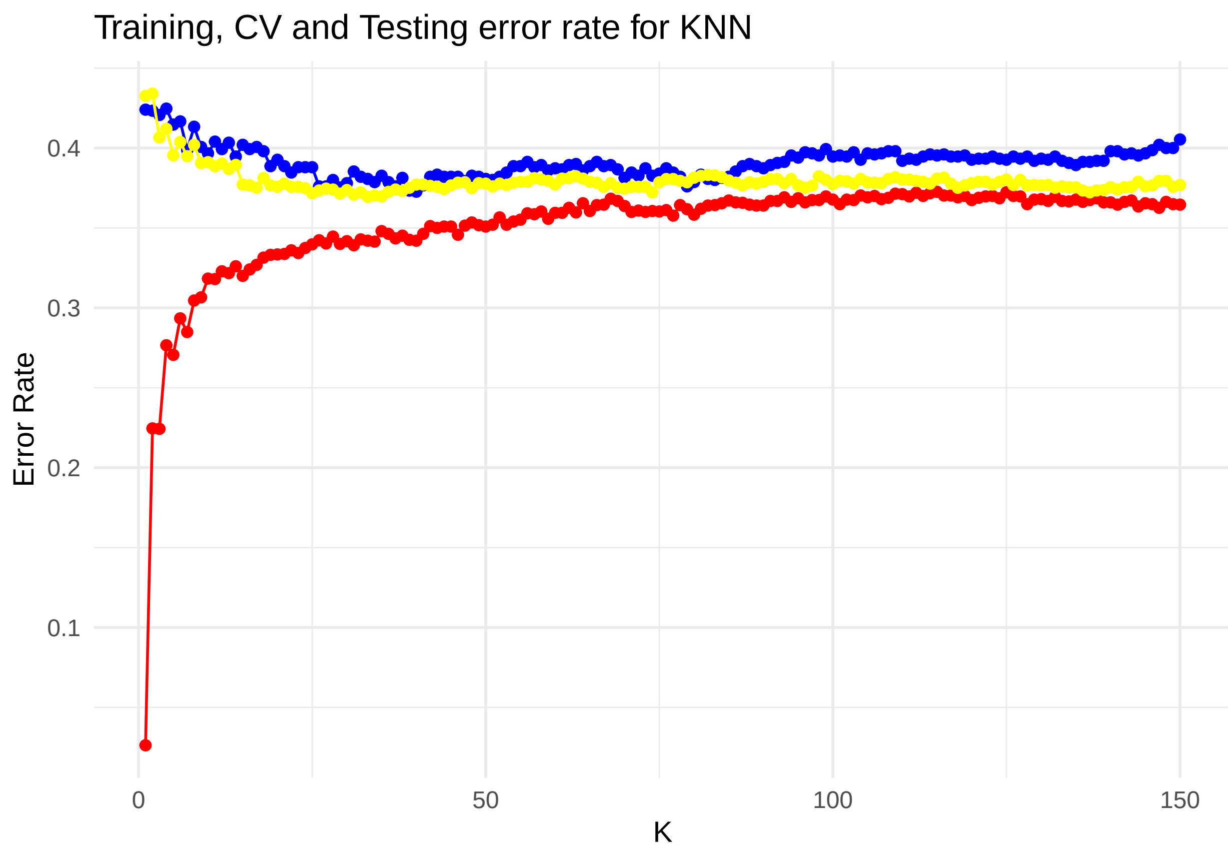
```r
best_k = which(cv_error == min(cv_error))
best_k #33
```

```
## [1] 33
```

```r
df_errors = cbind(errors, cv_error)
head(df_errors, n = 5)
```

```
##   train_error test_error K_knn  cv_error
## 1  0.02628571  0.4240000     1 0.4325714
## 2  0.22457143  0.4233333     2 0.4340000
## 3  0.22428571  0.4206667     3 0.4065714
```

```
## 4  0.27657143  0.4246667     4 0.4120000
## 5  0.27057143  0.4146667     5 0.3954286
```



Problem 3

The data set had 1089 observations with 9 variables.

```
data("Weekly")
summary(Weekly) # checkig missing values
dim(Weekly)
```

a. Logistic regression model with Lag1 and Lag2

First, we fitted a logistic model to predict direction using variables "lag1" and "lag2" with all data. In the model, the variable lag2 was statistically significant, which may indicate lag2 is a statistcally useful variable. However, in the confusion matrix of variable "direction", we noticed the error was 0.44 even if using all data for model training. We don't think the model performed well at this point.

```
# logistic model with lag1 and lag 2
fit1 = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)

pred1 = predict(fit1, Weekly)
# prediction
predProbs1 = binomial()$linkinv(pred1)
pred_log1 = rep("Down", nrow(Weekly))
```

```r
# Assign the label to be "Yes" if the probability is greater than 0.5
pred_log1[predProbs1 > .5] = "Up"
table(pred_log1, Weekly$Direction)
```

```
##
## pred_log1 Down  Up
##      Down   38  38
##      Up    446 567
```

```r
err_log1 = sum(pred_log1 != Weekly$Direction) / nrow(Weekly)
err_log1
```

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

b & c.

We trained a logistic model using lag1 and lag2 with all but the first observation. Based on the confusion matrix, the testing error of the model was 0.443. it was a little better than the previous model but still not performed well overall. With this model, the first observation was predicted as Up, however, the actual response was Down. The first observation was not correctly classified.

Based on the results from part a - c, we would like to use other variables as predictors and try to improve model performance.

```r
# logistic model with lag1 and lag 2
fit2 = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1,], family = binomial)

pred2 = predict(fit2, Weekly)
# prediction
predProbs2 = binomial()$linkinv(pred2)
pred_log2 = rep("Down", nrow(Weekly))

# Assign the label to be "Yes" if the probability is greater than 0.5
pred_log2[predProbs2 > .5] = "Up"
table(pred_log2, Weekly$Direction)
```

```
##
## pred_log2 Down  Up
##      Down   38  37
##      Up    446 568
```

```r
err_log2 = sum(pred_log2 != Weekly$Direction) / nrow(Weekly)
err_log2
```

```
## [1] 0.4435262
```

```r
# predict for the first observation
identical(pred_log2[1], Weekly$Direction[1])
```

```
## [1] FALSE
```

d.

With LOOCV approach, we fit 1089 logistic regression model, and the overall testing error was 0.45. The error rate was pretty high, which indicated that the logistic model did not perform well. In terms of the situation, we may consider the following analyses to improve model performance and accuracy. Firstly, analyze the relationship of predictiors "lag1" and "lag2" and consider use other variables as predictors to improve model performances. Secondly, we may consider use other models such as LDA or Naive Bayes after doing EDA analysis.

```r
# set up
n = nrow(Weekly)
pred = c()
predProbs = c()
pred_log = rep("Down", nrow(Weekly))
err = c()
# the for loop helps compute the LOOCV error for a logistic regression model
for (i in 1:n){
  new_data = Weekly[-i,]
  fit = glm(Direction ~ Lag1 + Lag2, data = new_data, family = binomial)
  pred[i] = predict(fit, Weekly[i,])
  predProbs[i] = binomial()$linkinv(pred[i])
  pred_log[i][predProbs[i] > .5] = "Up"

  if (pred_log[i] != Weekly$Direction[i]){
    err[i] = 1
    } else{
      err[i] = 0
    }
}

mean(err)
```

```
## [1] 0.4499541
```