Classification and Linear Discriminant Analysis

Jimmy G. Moore

September 7, 2018

Problem 1

Perform k-nearest neighbor classification using the training data with k = 1. Use this model to predict the class of each observation in the training data set. How many observations were incorrectly classified. Is this good?

The point of this problem is to showcase a scenario of **overfitting**. To start this problem, we will load our training and test sets.

```
#Load traing set
trainSet = read.csv("C:/Users/James Moore/Documents/LDA_model/dataFiles/PA_HW1_train.csv")
#Load test set
testSet = read.csv("C:/Users/James Moore/Documents/LDA_model/dataFiles/PA_HW1_test.csv")
```

Next we will conduct k-nearest neighbor on our training set with k = 1 and we are predicting each observation in our training set.

```
library(class)

#knn with k = 1 on training
modelTrain = knn(cbind(trainSet$x1,trainSet$x2),
                 cbind(trainSet$x1,trainSet$x2),
                 cl = trainSet$col,
                 k = 1,
                 prob = TRUE)
table(modelTrain,trainSet$col)
```

```
##
## modelTrain green red
##      green    75   0
##      red       0 100
```

This is a perfect classification. There are no observations that are incorrectly classified. However, this is not good, rather it is expected, because our model was trained on the same set that we are predicting which makes it an incredibly **overfit** model. I would not use this model.

Now we will build a knn model with our training set to predict on our test set

```
#knn with k = 1 on test
modelTest = knn(cbind(trainSet$x1,trainSet$x2),
                cbind(testSet$x1,testSet$x2),
                cl = trainSet$col,
                k = 1,
                prob = TRUE)

table(modelTest,testSet$col)
```

```
##
## modelTest green red
##     green   398 367
##     red     352 633
```

We can see from our table that 719 observations were incorrectly classified. That means about 41% of our observations we incorrectly classified. I do not think this is very good. Further analysis should be done, such as modifying K to find an optimal model.

Finally we will train a model for each k between 1 and 100. We will predict the observations in training and test set, and make a plot of the error rate as a funtion of our k value.

Our models were already created in the previous section so now we will simply modify them for each k

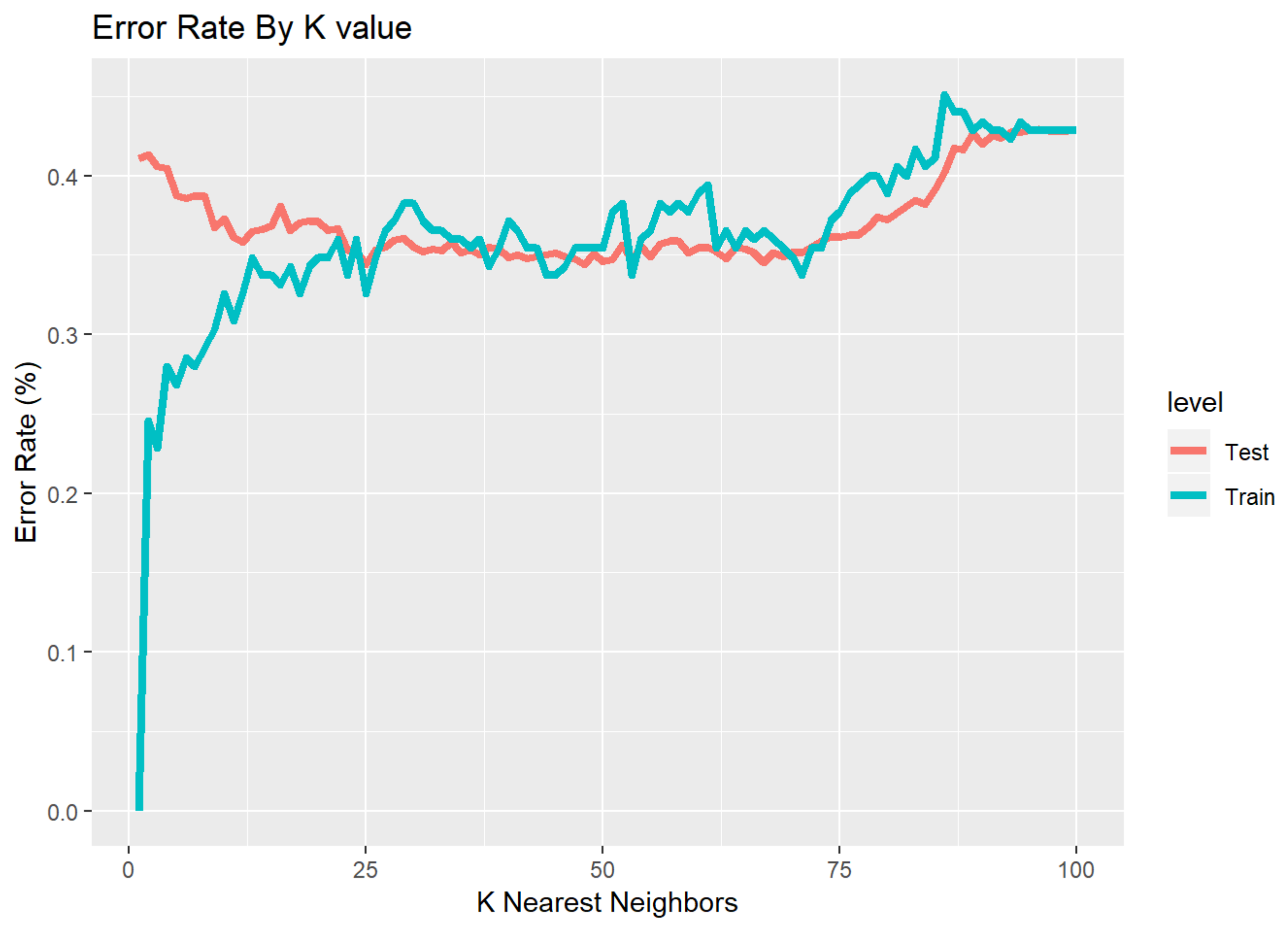
```r
#initialize error rate vectors
errorRateTrain = numeric(100)
errorRateTest = numeric(100)

#begin loop for k = 1 to 100
for (kVal in 1:100) {
  #knn with k = kVal on training
  modelTrain = knn(cbind(trainSet$x1,trainSet$x2),
                   cbind(trainSet$x1,trainSet$x2),
                   cl = trainSet$col,
                   k = kVal,
                   prob = TRUE)
  #knn with k = kVal on test
  modelTest = knn(cbind(trainSet$x1,trainSet$x2),
                  cbind(testSet$x1,testSet$x2),
                  cl = trainSet$col,
                  k = kVal,
                  prob = TRUE)

  #Calculate Error Rate
  tabTrain = table(modelTrain,trainSet$col)
  errorRateTrain[kVal] = 1-sum(diag(tabTrain))/sum(tabTrain)

  tabTest = table(modelTest,testSet$col)
  errorRateTest[kVal] = 1-sum(diag(tabTest))/sum(tabTest)
}

#plot
library(ggplot2)
dat  = data.frame(k = rep(seq(1:100),2),
                  errorRates = c(errorRateTrain,errorRateTest),
                  level = factor(c(rep("Train",100),rep("Test",100))))
ggplot(data = dat, aes(x = k, y = errorRates, color = level))+
  geom_line(size = 1.5)+
  ggtitle("Error Rate By K value")+
  xlab("K Nearest Neighbors")+
  ylab("Error Rate (%)")
```

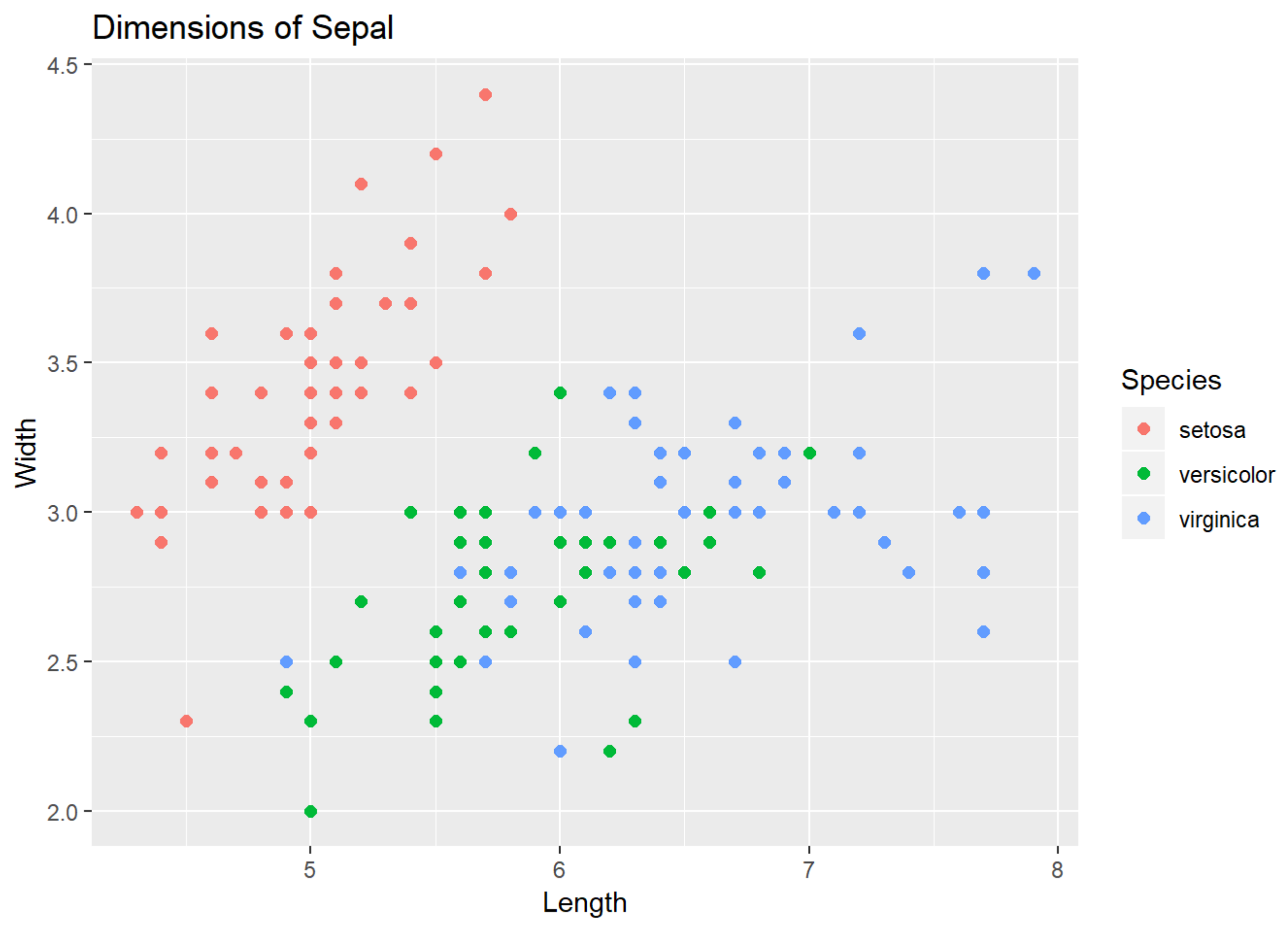


We can see from our graph that the error rate is always increasing when we make predictions on our training set. This means there is no optimal value of k when you train a model to predict on itself. However, we can see when we predict on our test set there is a our error rate is lowest when k is between 25 - 75 nearest neighbors. For computational purposes the most efficient model would be the lowest k value with the lowest erorr rate. For that reason I think for this specific set of data, k-nearest neighbor models are optimized when k = 25.

Problem 2

Plot all irises with Sepal.Length on the x-axis and Sepal.Width on the y-axis using different colors for each species.

```
ggplot(data = iris, aes(x = Sepal.Length,y = Sepal.Width,color = Species))+
  geom_point(size = 2)+
  ggtitle("Dimensions of Sepal")+
  xlab("Length")+
  ylab("Width")
```



Problem 3

Perform linear discriminant analysis using the iris data with only Sepal.Length and Sepal.Width as predictors. Make predictions about the species of each iris and create a confusion matrix for these predictions.

```
library(MASS)
library(VGAM)
library(caret)
library(e1071)

#Build Model
ldaModel = lda(Species~Sepal.Width+Sepal.Length,data = iris)
#Make Predictions
out = predict(ldaModel,iris)
#Output Confusion Matrix
confusionMatrix(out$class,iris$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         49          0         0
##   versicolor      1         36        15
##   virginica       0         14        35
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.727, 0.8608)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 0.9800            0.7200           0.7000
## Specificity                 1.0000            0.8400           0.8600
## Pos Pred Value              1.0000            0.6923           0.7143
## Neg Pred Value              0.9901            0.8571           0.8515
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3267            0.2400           0.2333
## Detection Prevalence        0.3267            0.3467           0.3267
## Balanced Accuracy           0.9900            0.7800           0.7800
```

Above we can see that our lda model did a good job of discriminating between setosa and the other species, and also did a pretty good job with with the versicolor and virgininca, which were clearly the most closely related in terms of Sepal Width and Length.

Quesion 7

ON the plot produced in question 5, plot the predicted value of species of a grid of points using the same color scheme used in part 1

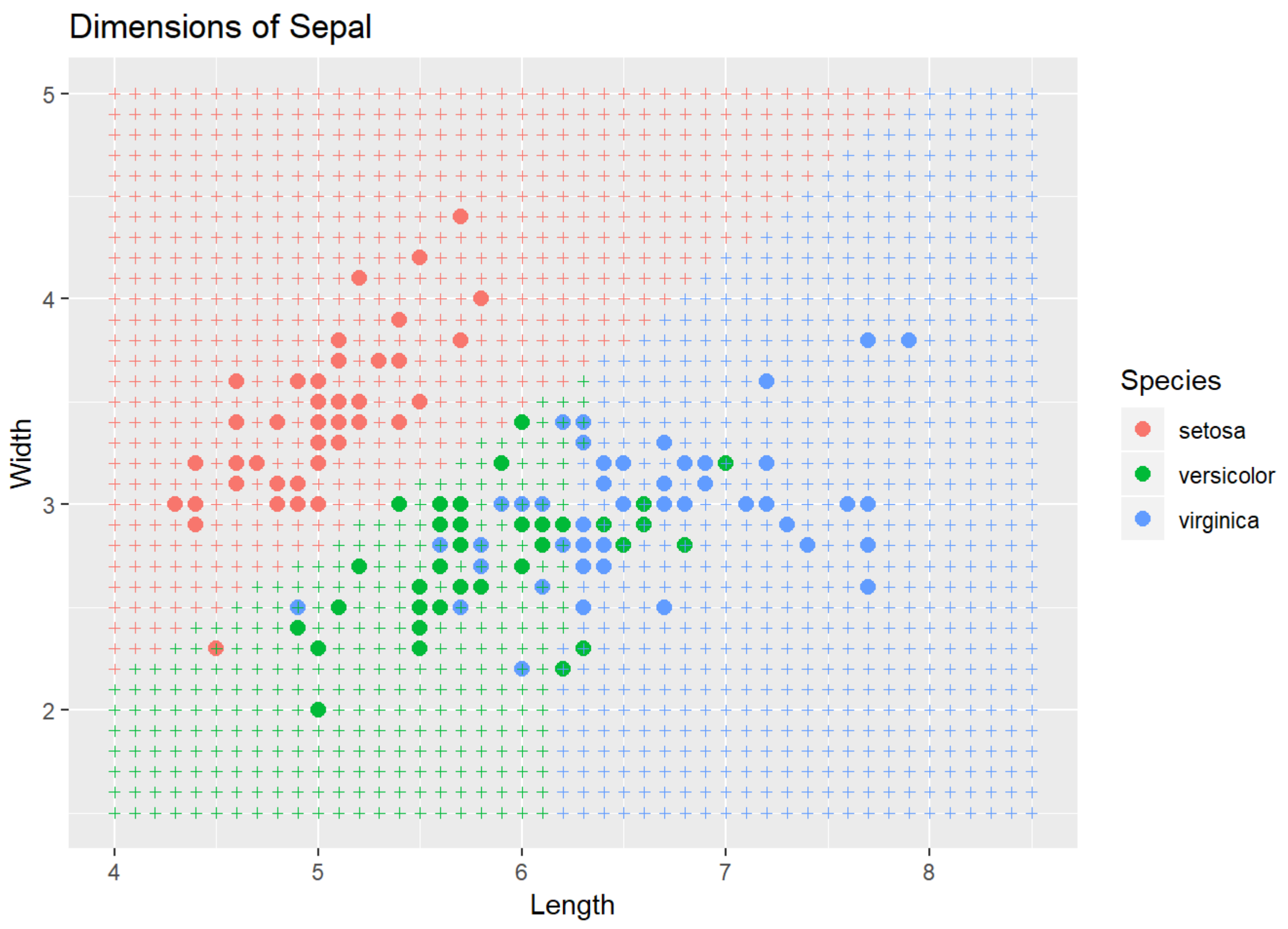
To begin we will create a grid of points and classify them appropriately

```
pred = expand.grid(seq(4,8.5,0.1),seq(1.5,5,0.1))
names(pred) = c("Sepal.Length","Sepal.Width")
out = predict(ldaModel,pred)
```

Next we will format our predictions into a data frame and plot the grid under the original plot of Length v. Width

```
dat = data.frame(x = pred$Sepal.Length,
                 y = pred$Sepal.Width,
                 level = factor(out$class))

ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length,y = Sepal.Width,color = Species),size = 2.5)+
  geom_point(data = dat, aes(x = x,y = y, color = level),size = 1, shape = 3)+
  ggtitle("Dimensions of Sepal")+
  xlab("Length")+
  ylab("Width")
```



The above graph shows a grid layout of the classifcations from our model under the truth. As we can see, our model did a pretty good job, especially in classifying setosa.