Final Group One

Adetayo Oreoluwa Adetoro, Bahati Philbert Kayihura,Kevin Joseph Russell, and Mahbubul Hasan

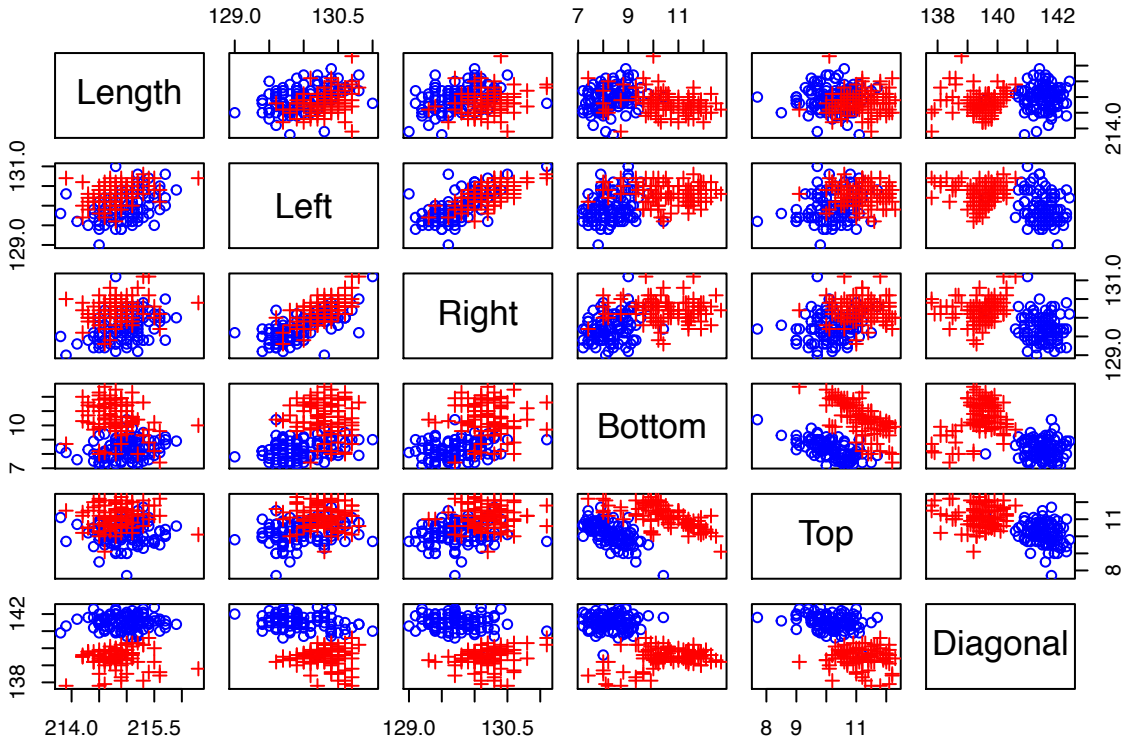
11/20/2020

Deacription of the data

The data set contain 6 measures of 100 genuine and 100 counterfeit Swiss franc banknotes. A data frame with the following variables: Status = the status of the banknote: genuine or counterfeit, Length= Length of bill (mm), Left= Width of left edge (mm), Right= Width of right edge (mm), Bottom=Bottom margin width (mm), Top=Top margin width (mm), Diagonal= Length of diagonal (mm)

Notice the the matrix plot below, red and blue represents counterfeit and genuine notes, and there are some overleaps among variables.

```
banknote<-read.table("banknote.txt", sep=" ", header=TRUE)
banknote<-as.data.frame(banknote)
banknote$Status<-as.factor(banknote$Status)#saving the leftmost column as a factor
colors<-c("red", "blue")[banknote$Status] #assigning a color and point type to
#each factor type (genuine, counterfeit)
pt<-c(3, 1)[banknote$Status]
pairs(banknote[,2:7], col=colors, pch=pt) #plotting the matrix scatterplot
```




```
# use caret to preprocess
processed <- preProcess(banknote[,-1], method = c('center', 'scale'))
banknote[,-1] <- predict(processed, banknote[,-1])
```

The support vector machine (SVM)

There is much more that can be said about SVM including the many options and different kernel smoothing functions available, but for this project, I would like to focus on these items, 1)brief definition of what is SVM , 2) SVM classification model, 3) SVM classification plot, 4) interpretation, 5)tuning or hyperparameter optimization, 6) best model selection, 7)confusion matrix, and 7)misclassification rate. We have tried to utilize materials from books, other lectures provided in the class. I did not find books R code useful for SVM since they used artificial data. Also, we have covered all the classification methods used in this course that includes logistic regression, linear discriminant analysis, support vector machine (SVM) , hierarchical/Regression Trees, KNN, QDA, and EDA etc.

The support vector machine (SVM) is a family of classification rules that contain both parametric (e.g., linear) and nonparametric (e.g., kernel based) methods. It is often considered as one of the ready to use classifiers.

From the model summary below, we can see number of support vector in the first model is 9 and used linear kernel. In the model two, number of support vector 38, 19 from each group, and in this case we used radial kernel. This kernel gave us the best accuracy score, and after tuning we have used redial kernel in the best model.

```
model <- svm(Status~., data = banknote, kernel="linear")
summary(model)
```

```
##
## Call:
## svm(formula = Status ~ ., data = banknote, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  9
##
##  ( 5 4 )
##
##
## Number of Classes:  2
##
## Levels:
##  counterfeit genuine
```

```
model <- svm(Status~., data = banknote)
summary(model)
```

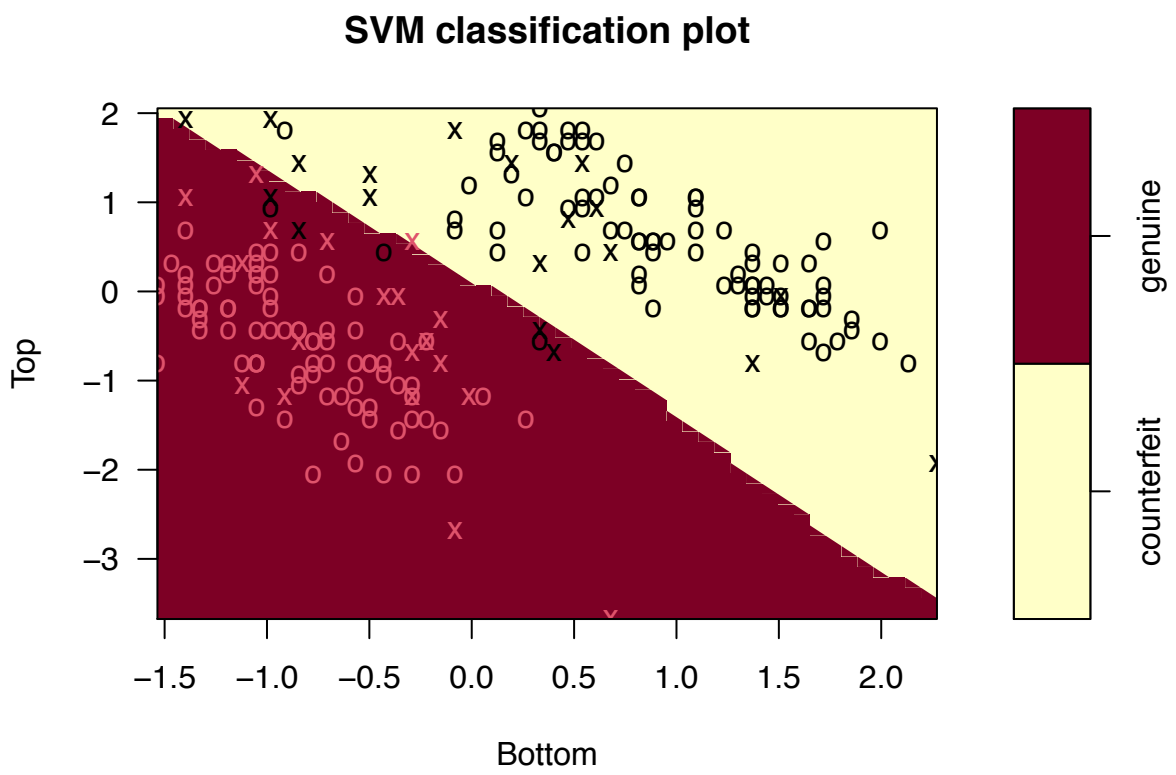
```
##
## Call:
## svm(formula = Status ~ ., data = banknote)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  38
##
##  ( 19 19 )
##
##
## Number of Classes:  2
##
## Levels:
##  counterfeit genuine
```

Next, From these plots, we can see that SVM with polynomial, linear and radial function do pretty well over SVM with sigmoid function. In fact, these remarks are confirmed by their corresponding accuracies.

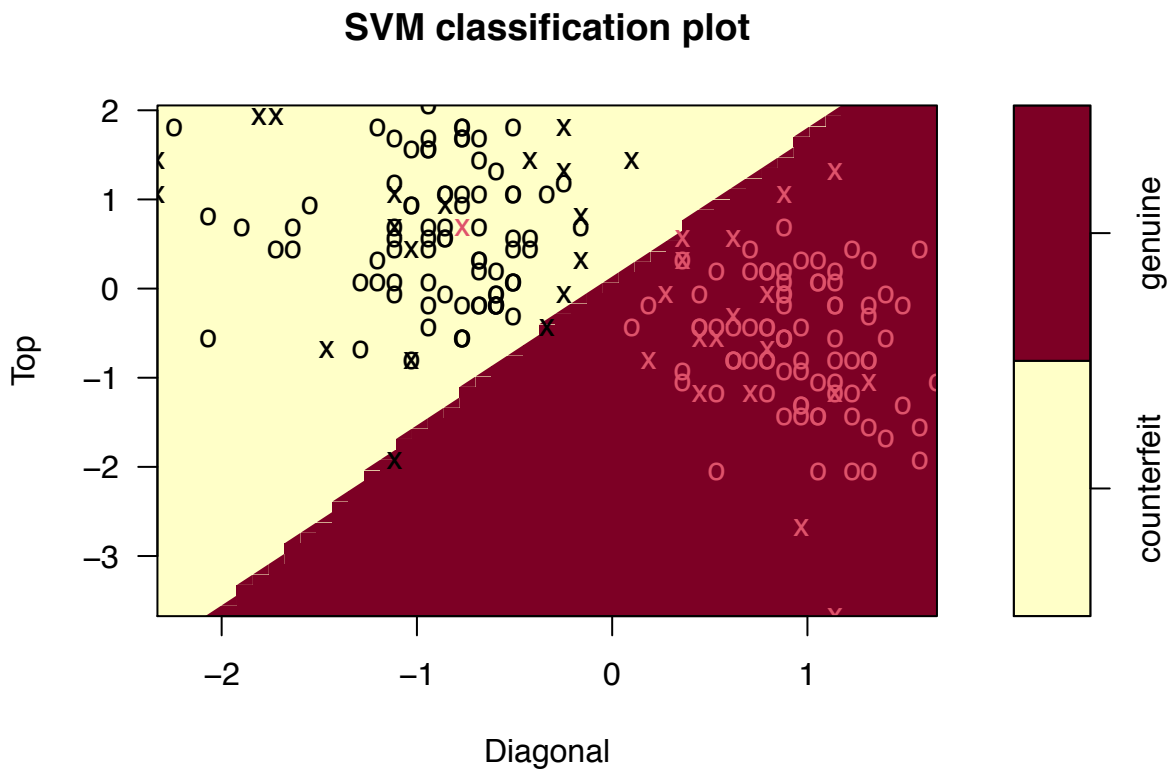
In these two plots below, this visualization shows us the data, support vector, and the decision boundary. In this case support vectors are presented by cross symbol and decision boundaries are color coded by two different color which represents genuine and counterfeit note.

Plus, The groups above and below this sine wave are plotted as cross sign and circles , respectively. Those observations are identified as support vectors. These are critical to estimating the boundaries plotted in solid. The search for support vectors is similar to observations located on the convex hull, except that the classification regions are not necessarily convex. The SVM algorithm, then, includes a search for classification regions as well as those observations that bound and determine the regions as shown two dimensions of two types note in this problem.

```
# All variables
plot(model, data= banknote,Top~Bottom, main = )
```




```
plot(model, data=banknote,Top~Diagonal)
```

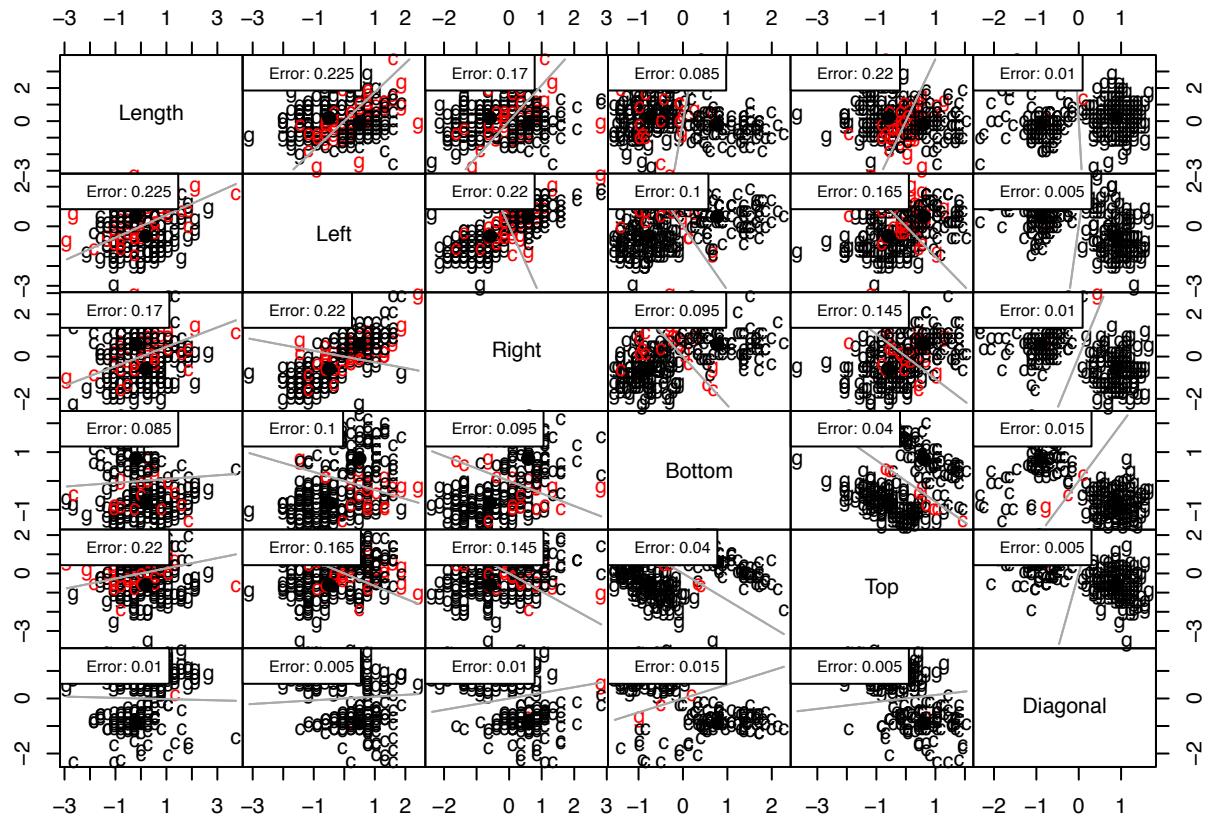


In the next matrix plot, we have used a package name *Klar* and used partimat function. This function creates the optimal separating line is shown as the solid gray line, the closest points to the line are circled. Essentially, the separating line corresponding to the maximal margin classifier represents the middle line of the widest space that we can fit between the two classes. The default SVM packge e1071 can display two sides where by using partimat function we can display more then two variables and support vectors and decision boundary everything in one plot.

Notice that there are a few points (circled) that are closest to the gray separating line. These points are the support vectors for this problem. It can be shown that only the support vectors are enough to define the optimal classification rule fully. If we move the support vectors, the optimal separating line also changes.

When the two classes are not linearly separable (e.g., in the matrix plot below), often we will not be able to find a line that entirely separates the groups, that is, the maximal margin classifier can not be computed. Thus, the two classes cannot be classified exactly. We can, however, generalize the ideas to develop a classification rule that almost separates the classes. To do so, we allow a few points to fall on the wrong side of the margin or separating line. Such a classifier is called a support vector classifier or a soft-margin classifier In Figure 8, the optimal separating line is shown as the solid red line, the closest points to the line are circled, and the margin is shown using the dashed black lines. Essentially, the separating line corresponding to the maximal margin classifier represents the middle line of the widest space that we can fit between the two classes.

```
partimat(as.factor(Status)~., data = banknote, method = "lda",
         plot.matrix = TRUE,image.colors = c("#FFBBBB","white", "steelblue"),imageplot = FALSE)
```



Confusion matrix shows one misclassifications in the 2nd category but model predicted to be in the first category. And the misclassification rate in this case is 0.005. Other kernels gave much higher misclassifications score. Next, we will fine tune our model to get better classification.

```
# confusion matrix & Misclassification rate
pred<- predict(model, banknote)
tab<- table(Predicted= pred, Actual= banknote$Status)
tab
```

```
##               Actual
## Predicted     counterfeit genuine
##   counterfeit         100       1
##   genuine               0      99
```

```
1-sum(diag(tab))/sum(tab)
```

```
## [1] 0.005
```

Confusion matrix shows one misclassfications in the 2nd catagory.

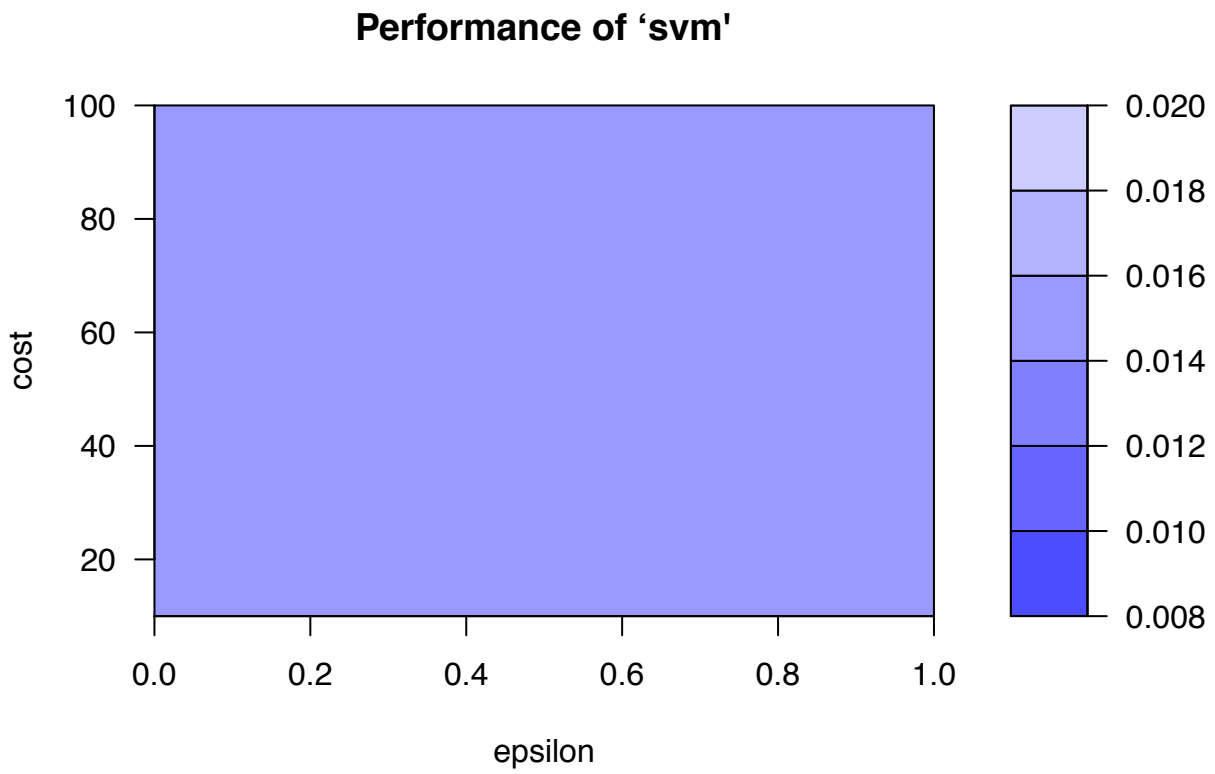
```
#Tuning

set.seed(123)
tmodel<- tune(svm, Status~.,data=banknote,
              ranges = list(epsilon= seq(0,1, 0.1), cost= 10^(1:2)))

summary(tmodel)
```

```
##
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##        0   10
##
## - best performance: 0.015
##
## - Detailed performance results:
##    epsilon cost error dispersion
## 1      0.0   10 0.015 0.03374743
## 2      0.1   10 0.015 0.03374743
## 3      0.2   10 0.015 0.03374743
## 4      0.3   10 0.015 0.03374743
## 5      0.4   10 0.015 0.03374743
## 6      0.5   10 0.015 0.03374743
## 7      0.6   10 0.015 0.03374743
## 8      0.7   10 0.015 0.03374743
## 9      0.8   10 0.015 0.03374743
## 10     0.9   10 0.015 0.03374743
## 11     1.0   10 0.015 0.03374743
## 12     0.0  100 0.015 0.03374743
## 13     0.1  100 0.015 0.03374743
## 14     0.2  100 0.015 0.03374743
## 15     0.3  100 0.015 0.03374743
## 16     0.4  100 0.015 0.03374743
## 17     0.5  100 0.015 0.03374743
## 18     0.6  100 0.015 0.03374743
## 19     0.7  100 0.015 0.03374743
## 20     0.8  100 0.015 0.03374743
## 21     0.9  100 0.015 0.03374743
## 22     1.0  100 0.015 0.03374743
```

Next, tuning is also called hyper parameter optimization, it helps to select best model. Notice in the SVM model performance plot displayed mostly high cost value and slightly lower values, which is better. Tmodel summary shows 10 fold cross validation and epsilon 0, and cost 8.

```
# Plot tmodel
plot(tmodel)
```



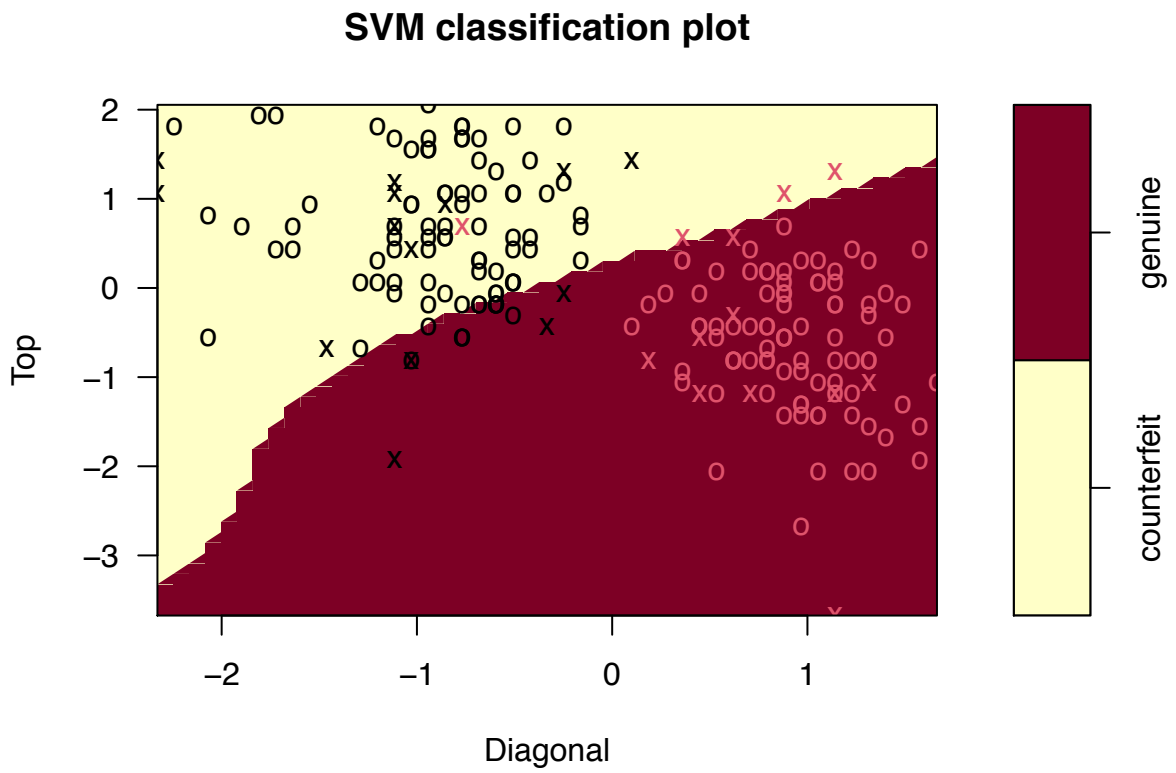
Finally, the best model improves the reduces the misclassifications score. Notice the decision boundary has changed now.

```
#best model

model.1<- tmodel$best.model
summary(model.1)
```

```
##
## Call:
## best.tune(method = svm, train.x = Status ~ ., data = banknote, ranges = list(epsilon = seq(0,
##     1, 0.1), cost = 10^(1:2)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  27
##
##  ( 13 14 )
##
##
## Number of Classes:  2
##
## Levels:
```

```
##  counterfeit genuine
```

```
plot(model.1, data= banknote,Top~Diagonal)
```



Best model improves the score.

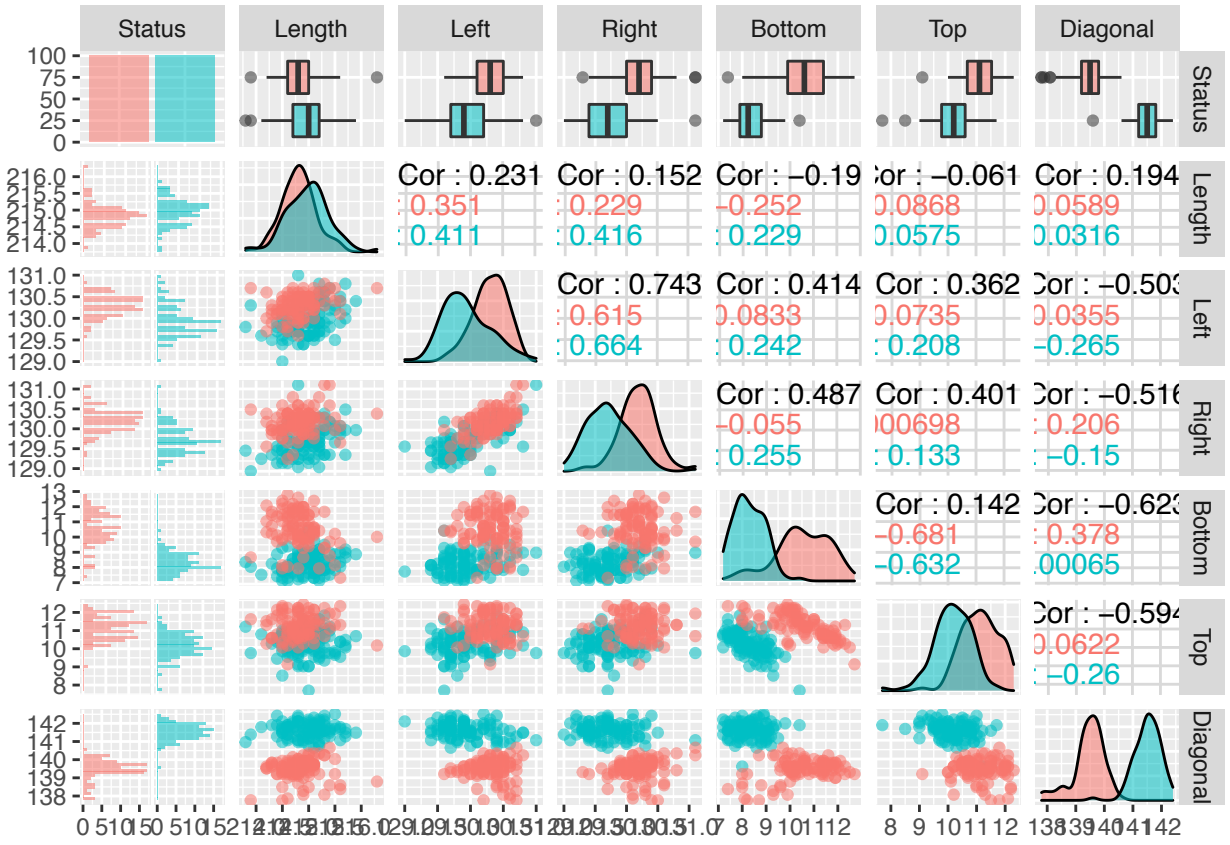
```
# confusion matrix & Misclassification rate
pred<- predict(model.1, banknote)
tab<- table(Predicted= pred, Actual= banknote$Status)
tab
```

```
##              Actual
## Predicted     counterfeit genuine
##   counterfeit         100       0
##   genuine               0     100
```

```
1-sum(diag(tab))/sum(tab)
```

```
## [1] 0
```

```
set.seed(2020)

banknote <- read_csv("banknote.csv")

banknote$Status <- as.factor(banknote$Status)

# head(banknote)
ggpairs(banknote[,-1], aes(color = Status, alpha = .7))
```



Here we split out data 70/30 into train and test sets. Also, data is centered and scaled for use in the various classification and clustering methods.

```
train <- banknote %>%
  dplyr::sample_frac(.7)
test <- dplyr::anti_join(banknote, train, by = 'X1')

train <- train[-1]
test <- test[-1]

processed <- preProcess(train[,-1], method = c('center', 'scale'))
train[,-1] <- predict(processed, train[,-1])
test[,-1] <- predict(processed, test[,-1])
```

Classification

K-nearest neighbors

10-fold cross-validation is used to choose our value of k (the number of nearest neighbors)

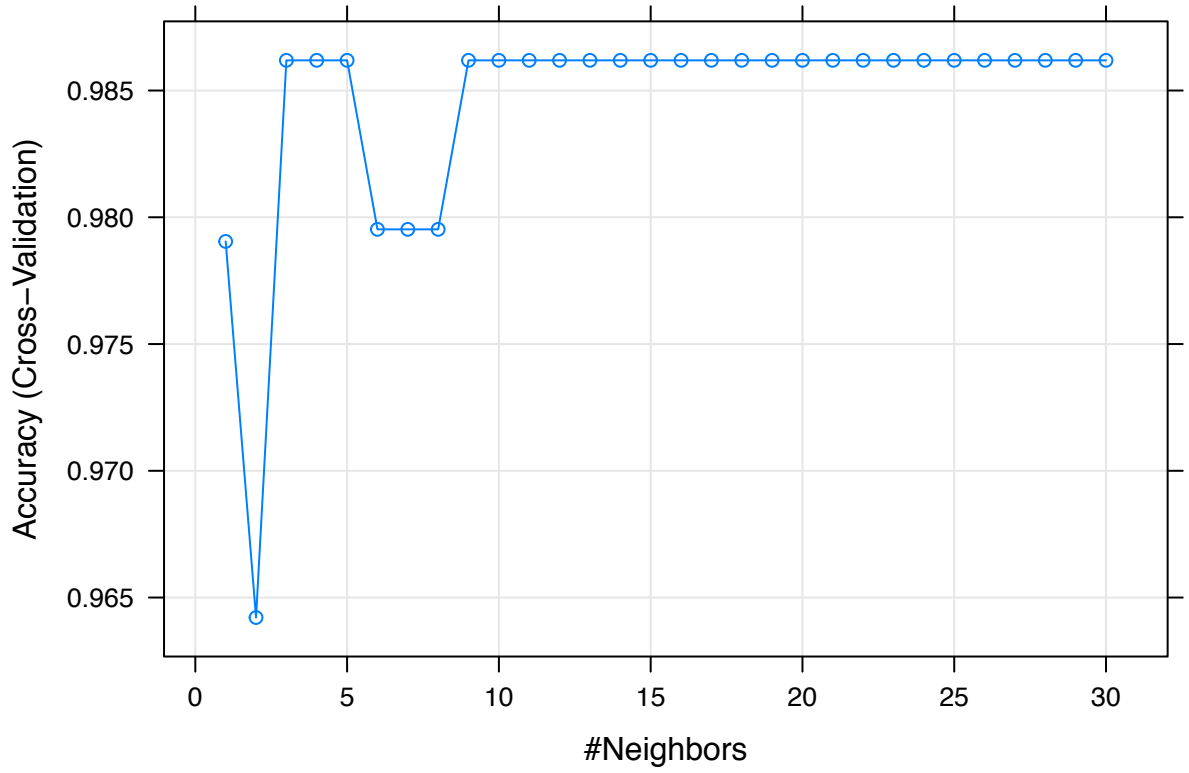
```
set.seed(2020)
trControl_knn <- trainControl(method = 'cv', number = 10)

modelknn <- train(Status ~ ., data = train,
                  method = 'knn', tuneGrid = expand.grid(k = 1:30),
                  trControl = trControl_knn,
                  metric = 'Accuracy')

modelknn
```

```
## k-Nearest Neighbors
##
## 140 samples
##   6 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 127, 126, 125, 126, 126, 125, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9790476  0.9581542
##    2  0.9642125  0.9282059
##    3  0.9861905  0.9724399
##    4  0.9861905  0.9724399
##    5  0.9861905  0.9724399
##    6  0.9795238  0.9589264
##    7  0.9795238  0.9589264
##    8  0.9795238  0.9589264
##    9  0.9861905  0.9724399
##   10  0.9861905  0.9724399
```

```
##   11  0.9861905  0.9724399
##   12  0.9861905  0.9724399
##   13  0.9861905  0.9724399
##   14  0.9861905  0.9724399
##   15  0.9861905  0.9724399
##   16  0.9861905  0.9724399
##   17  0.9861905  0.9724399
##   18  0.9861905  0.9724399
##   19  0.9861905  0.9724399
##   20  0.9861905  0.9724399
##   21  0.9861905  0.9724399
##   22  0.9861905  0.9724399
##   23  0.9861905  0.9724399
##   24  0.9861905  0.9724399
##   25  0.9861905  0.9724399
##   26  0.9861905  0.9724399
##   27  0.9861905  0.9724399
##   28  0.9861905  0.9724399
##   29  0.9861905  0.9724399
##   30  0.9861905  0.9724399
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 30.
```

The accuracy is 0.986 in sample.

```
plot(modelknn)
```




```
modelknn$bestTune
```

```
##     k
## 30 30
```

Our cross-validation choose k = 30 nearest neighbors for the calculation. This is somewhat expected because there appears to be a largely linear separation between the groups. However, it's worth noting that there is not much difference between various choices of k.

```
predknn <- predict(modelknn, test)
confusionMatrix(as.factor(predknn), as.factor(test$Status))
```

OOS

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    counterfeit genuine
##   counterfeit          32       0
##   genuine               0      28
##
##                Accuracy : 1
```

```
##                  95% CI : (0.9404, 1)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5333
##          Detection Rate : 0.5333
##    Detection Prevalence : 0.5333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : counterfeit
##
```

The oos sample accuracy is 1. There is very good separation between the real and counterfeit notes making this possible.

Quadratic Discriminant Analysis

If we look at the covariance matrices for the two respective classes in the training set, they do not appear wildly different.

```
set.seed(2020)
knitr::kable(cov(
train %>%
  dplyr::filter(Status == 'counterfeit') %>%
  dplyr::select(-Status)
))
```

	Length	Left	Right	Bottom	Top	Diagonal
Length	1.0270571	0.2334700	0.1598311	-0.2048204	0.0593705	0.0220671
Left	0.2334700	0.5132927	0.3550870	-0.0636289	-0.0141862	-0.0061194
Right	0.1598311	0.3550870	0.5495369	-0.0349397	-0.0248172	0.0471293
Bottom	-0.2048204	-0.0636289	-0.0349397	0.6291825	-0.3837030	0.1322686
Top	0.0593705	-0.0141862	-0.0248172	-0.3837030	0.5542867	-0.0180724
Diagonal	0.0220671	-0.0061194	0.0471293	0.1322686	-0.0180724	0.2099766

```
knitr::kable(cov(
  train %>%
    dplyr::filter(Status == 'genuine') %>%
    dplyr::select(-Status)
))
```

	Length	Left	Right	Bottom	Top	Diagonal
Length	0.9519805	0.4193954	0.4218466	0.1132866	0.0647636	0.0083855
Left	0.4193954	1.0274252	0.6292740	0.1037057	0.2075192	-0.0993286
Right	0.4218466	0.6292740	0.8411500	0.1156552	0.0967401	-0.0467314
Bottom	0.1132866	0.1037057	0.1156552	0.2116668	-0.2560549	0.0131359
Top	0.0647636	0.2075192	0.0967401	-0.2560549	0.6740633	-0.0949870
Diagonal	0.0083855	-0.0993286	-0.0467314	0.0131359	-0.0949870	0.1695044

Box's M Test is a way of more rigorously examining this:

```
biotools::boxM(train[,2:7], train$Status)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  train[, 2:7]
## Chi-Sq (approx.) = 90.665, df = 21, p-value = 1.239e-10
```

We do reject the null hypothesis that the variances are equal. If one looks back to the scatter plots, we can see that along certain dimensions the data does have a somewhat different variance among the classes.

There's no parameter that needs to be optimized here so we will proceed with a simple in sample/oos test.

```
qda.fit <- qda(Status ~ ., data = train)
qda.fit
```

```
## Call:
## qda(Status ~ ., data = train)
##
## Prior probabilities of groups:
## counterfeit     genuine
##   0.4857143   0.5142857
##
## Group means:
##                 Length       Left      Right     Bottom        Top   Diagonal
## counterfeit -0.1401352  0.4893486  0.5666757  0.7866255  0.6390899 -0.9240352
## genuine      0.1323499 -0.4621626 -0.5351937 -0.7429241 -0.6035849  0.8726999
```

```
qda.class <- predict(qda.fit, newdata = test)$class
confusionMatrix(qda.class, test$Status)
```

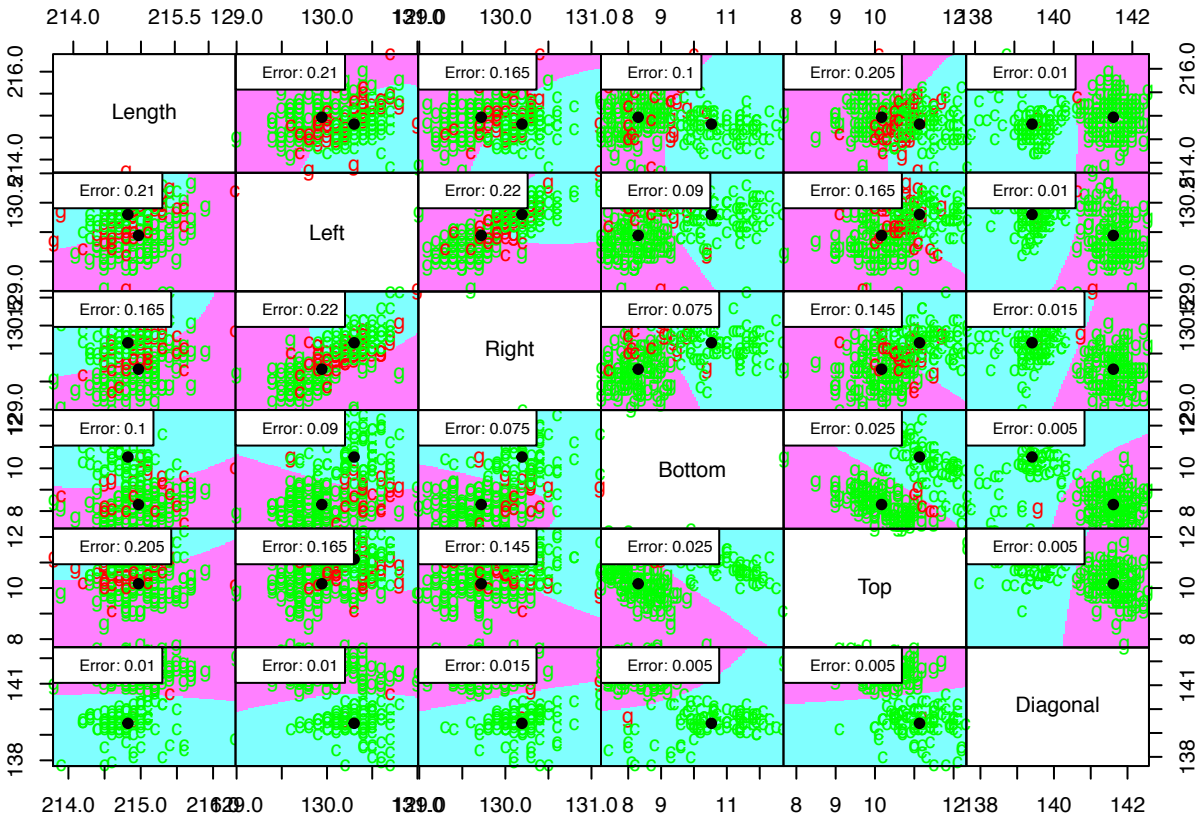
```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    counterfeit genuine
##   counterfeit          32       0
##   genuine               0      28
##
##                Accuracy : 1
##                  95% CI : (0.9404, 1)
##     No Information Rate : 0.5333
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5333
##          Detection Rate : 0.5333
##    Detection Prevalence : 0.5333
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : counterfeit
##
```

Essentially, we have the same situation here. Our qda model perfectly classifies the Swiss notes.

Just for interest's sake, below is the decision boundary plotted for the entire data using qda.

```
partimat(Status ~ . -X1, data = banknote, method = 'qda', plot.matrix = T, imageplot = T,
         col.correct='green', col.wrong='red', cex=1)
```



Clustering

K-means

While we essentially know the number of clusters here (we're assuming it's two). What if we didn't? It's easy to imagine that we have some unknown groups of Swiss notes. There could be real, counterfeit, notes with printing errors, some Euro that got stuck in the pile, or perhaps we're able to notice measurement errors by tracking down significant outliers.

It would obviously be useful to be able to discover groups from the data that we don't know ahead of time.

```
set.seed(2020)

# select only the measurement variables from our data
dat <- banknote %>%
  dplyr::select(-X1, -Status)
head(dat)
```

```
## # A tibble: 6 x 6
##   Length  Left Right Bottom   Top Diagonal
##    <dbl> <dbl> <dbl>  <dbl> <dbl>    <dbl>
## 1   215.  131   131.    9     9.7     141
## 2   215.  130.  130.    8.1   9.5     142.
## 3   215.  130.  130.    8.7   9.6     142.
## 4   215.  130.  130.    7.5  10.4     142
## 5   215   130.  130.   10.4   7.7     142.
## 6   216.  131.  130.    9    10.1     141.
```

```
# normalize the data
processed <- preProcess(dat, method = c('center', 'scale'))
datn <- predict(processed, dat)
```

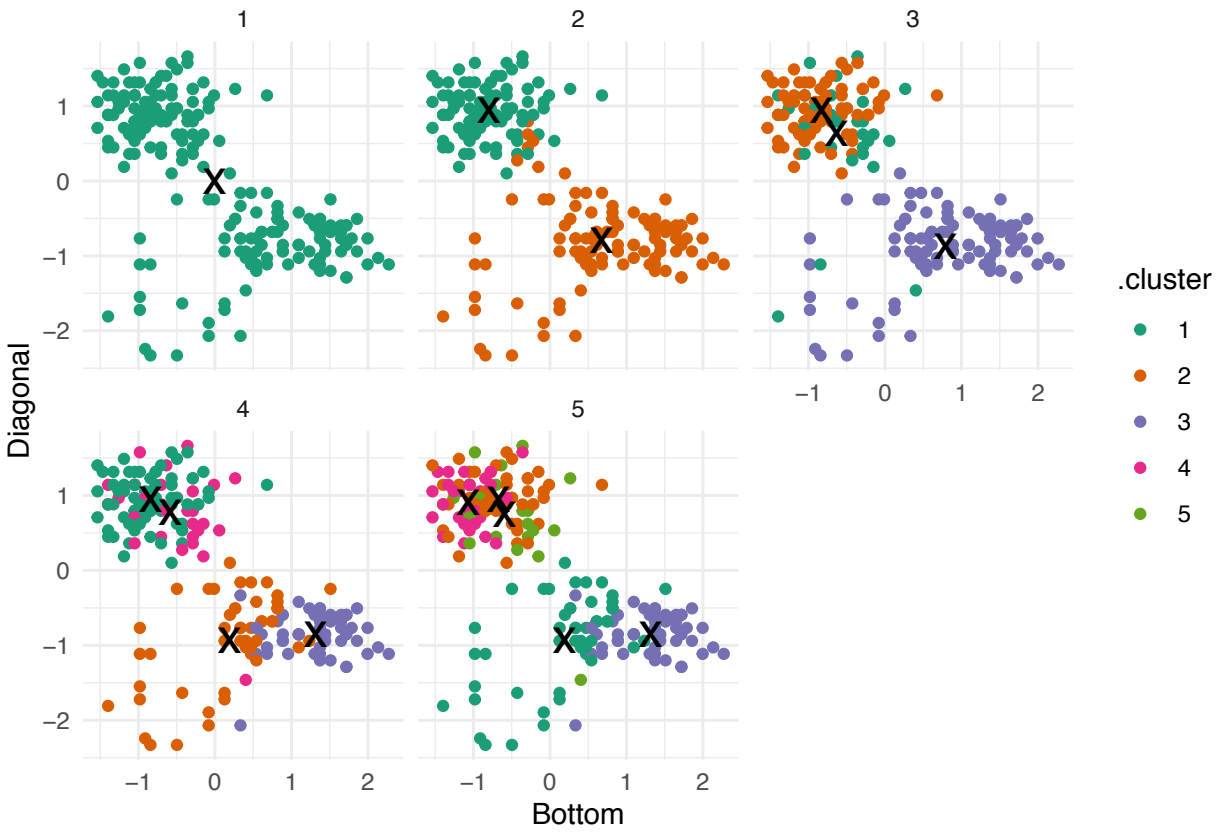
We'll look for up to 5 different clusters.

```
kclusts <- tibble(k = 1:5) %>%
  mutate(
    kclust = map(k, ~kmeans(datn, .x)),
    tidied = map(kclust, tidy),
    glanced = map(kclust, glance),
    augmented = map(kclust, augment, datn)
  )

clusters <- kclusts %>%
  unnest(cols = c(tidied))
assignments <- kclusts %>%
  unnest(cols = c(augmented))
clusterings <- kclusts %>%
  unnest(cols = c(glanced))
```

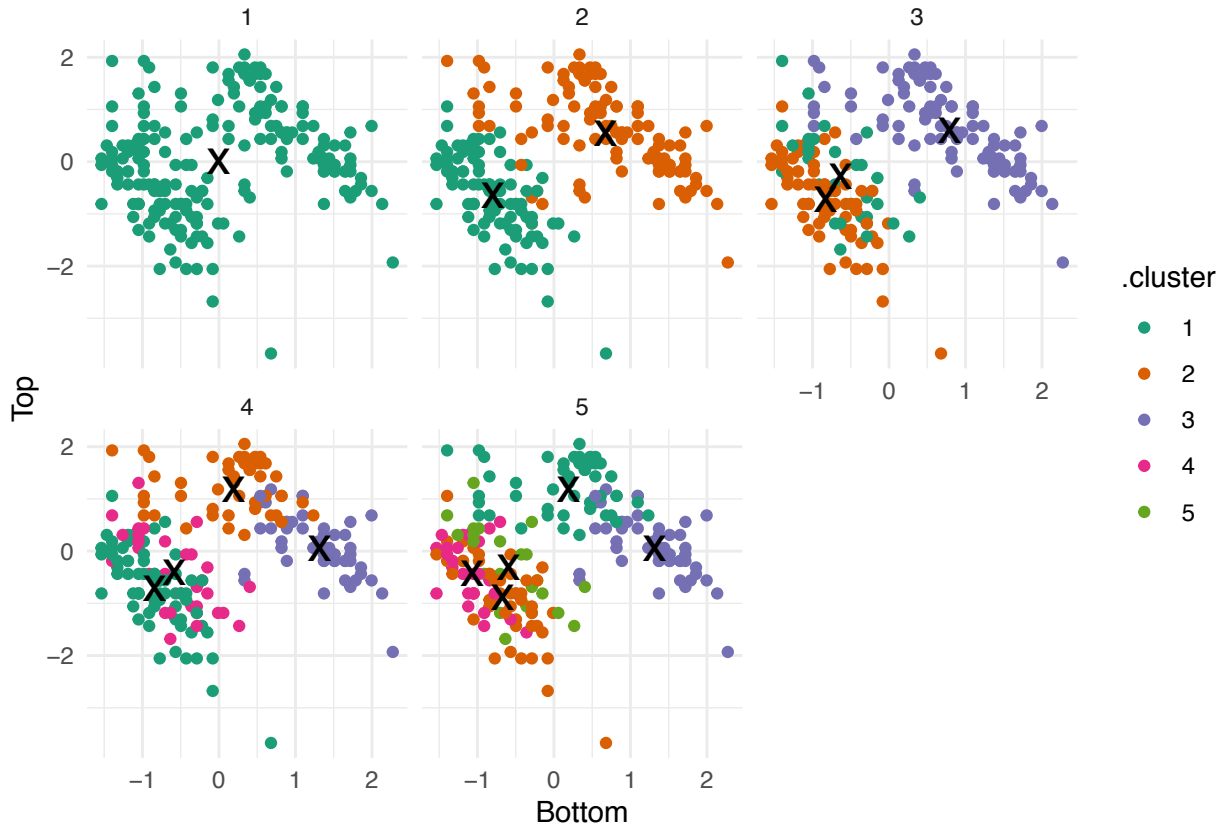
Possible Clusters Since we have 6 variables, there are up to 15 different combinations that are possible to plot on a 2d graph. Here we'll only choose a few based off prior PCA work. (This choice could be made many other ways as well.)

Plotted with center of cluster marked:

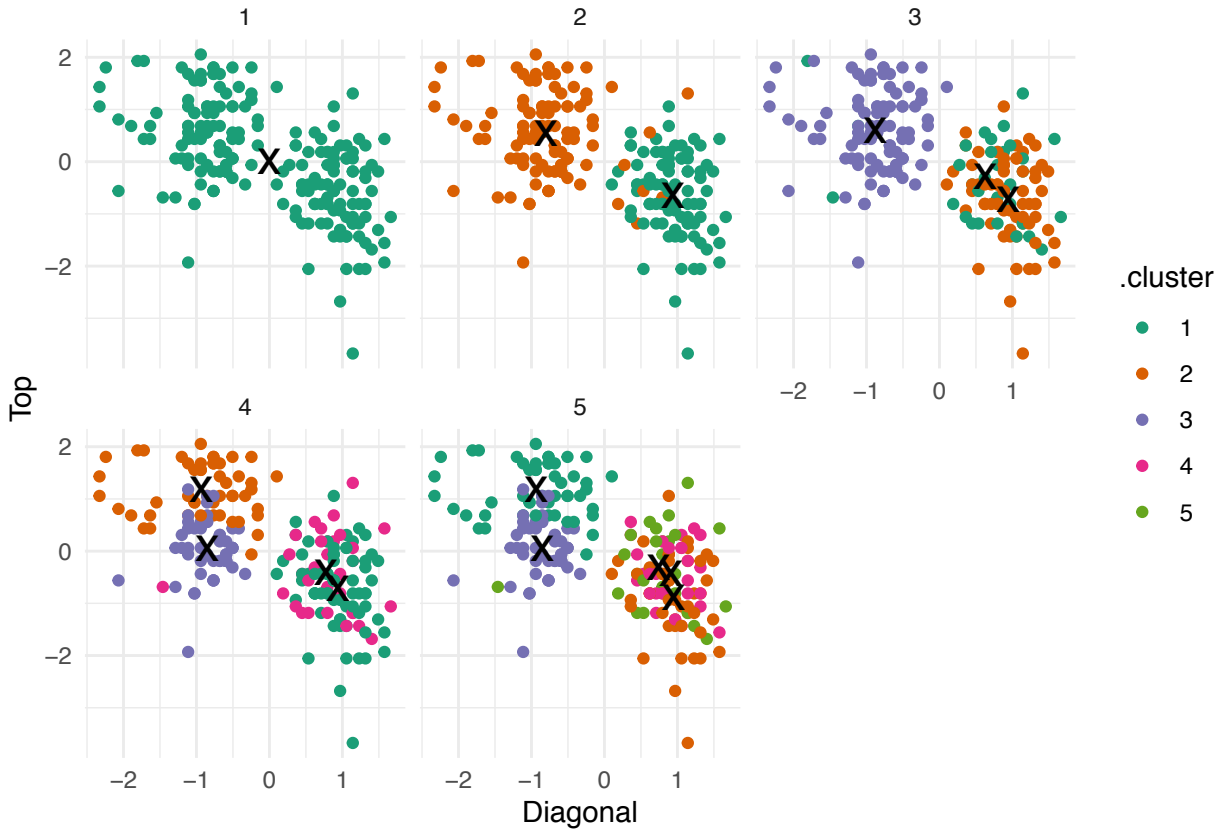
```
p1 <- ggplot(assignments, aes(x = Bottom, y = Diagonal)) +
  geom_point(aes(color = .cluster)) +
  theme_minimal() +
  scale_color_brewer(palette = "Dark2") +
  facet_wrap(~k)

p1 + geom_point(data = clusters, size = 6,
                shape = 'x')
```




```
p2 <- ggplot(assignments, aes(x = Bottom, y = Top)) +
  geom_point(aes(color = .cluster)) +
  theme_minimal() +
  scale_color_brewer(palette = "Dark2") +
  facet_wrap(~k)

p2 + geom_point(data = clusters, size = 6,
                shape = 'x')
```

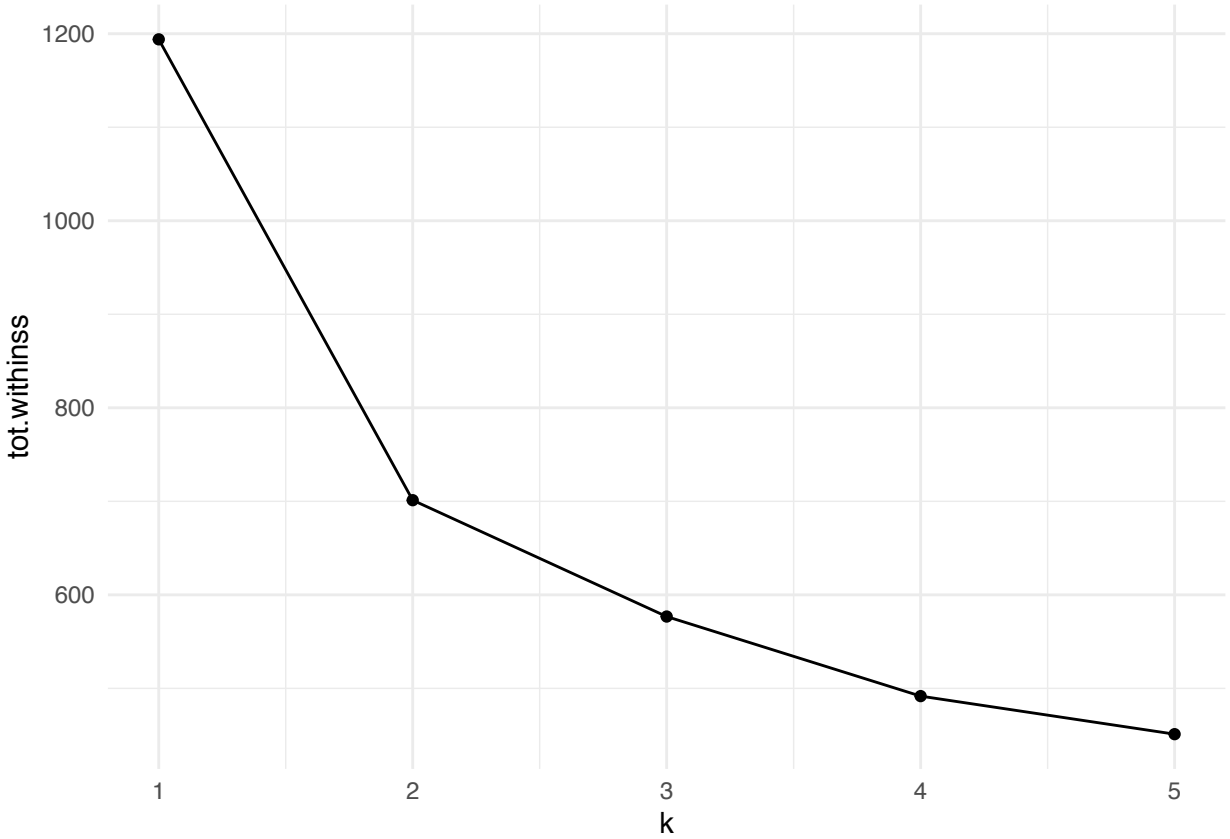



```
p3 <- ggplot(assignments, aes(x = Diagonal, y = Top)) +
  geom_point(aes(color = .cluster)) +
  theme_minimal() +
  scale_color_brewer(palette = "Dark2") +
  facet_wrap(~k)

p3 + geom_point(data = clusters, size = 6,
                shape = 'x')
```



Total within sum of squares variance within clusters This is useful when trying to decide on the proper value of k.

```
clusterings %>%
  ggplot(aes(k, tot.withinss)) +
  geom_line() +
  geom_point() +
  theme_minimal()
```



Assuming no prior knowledge, k-means does a pretty good job of separating the clusters. While this is quite simple analysis, I think the possible third cluster near the overlap of the two main ones warrants further investigation. It appears that some counterfeit operation at least had the sizing down close.

LDA and Logistic Regression

```
library(MASS)
banknote<-read.table("banknote.txt", sep=" ", header=TRUE)
banknote<-as.data.frame(banknote)
```

For the LDA classification, let???s divide the data set into two subsets, one for training and one for test:

```
train=sample(1:nrow(banknote), nrow(banknote)/2)
banknote_train=banknote[train,]
banknote_test=banknote[-train,]
```

Below is the model for LDA:

Interpretation of the results from the LDA model: For the Prior probability of groups, there is a probability of 0.48 to get a counterfeit banknote and a probability of 0.52 to get a genuine banknote from the training data set.

The model also gives the mean for independent variables used in this model. The model also gives the coefficients of linear discriminants. (Status= -0.2953171 *Length + 0.7235078* Left - 0.5297249 *Right - 1.2081389*Bottom - 1.2127242*Top + 1.6776095*Diagonal.

```
fit=lda(as.factor(Status)~.,data=banknote_train)

fit
```

```
## Call:
## lda(as.factor(Status) ~ ., data = banknote_train)
##
## Prior probabilities of groups:
## counterfeit     genuine
##        0.47        0.53
##
## Group means:
##                Length     Left    Right    Bottom      Top Diagonal
## counterfeit 214.8383 130.3234 130.2298 10.640426 11.10426 139.4787
## genuine     214.9566 129.9811 129.7340  8.298113 10.18302 141.4906
##
## Coefficients of linear discriminants:
##                   LD1
## Length    0.01208876
## Left      0.79371187
## Right    -0.72364778
## Bottom   -1.06063044
## Top      -1.12848857
## Diagonal  1.73466610
```

Below is the result of prediction using the test subset in our model:

$posterior: Every row of these below fitted values sums to one (or within rounding error of one). Each row is the estimated probability of class membership for this banknote.

```
pred=predict(fit,banknote_test)
pred
```

Confusion Matrix: From the confusion Matrix above, there is 1 missclassification for counterfeit and zero misclassification for genuine.

Lets calculate the probability of getting a right classification using our model:

```
pred_class=pred$class
table(pred_class,banknote_test$Status)
```

```
##
## pred_class    counterfeit genuine
##   counterfeit          53       1
##   genuine               0      46
```

```
mean(pred_class==banknote_test$Status)
```

```
## [1] 0.99
```

Logistic Regression

```
banknote[banknote$Status=="genuine",]$Status<-"0"
banknote[banknote$Status=="counterfeit",]$Status<-"1"
banknote$Status<- as.factor(banknote$Status)
```

Let???s divide the banknote data set into two subsets. One for the training and onether one for the test.

```
set.seed(1234)
ind<-sample(2, nrow(banknote), replace=TRUE, prob=c(0.7, 0.3))
train<-banknote[ind==1,]
test<-banknote[ind==2,]
```

For the Logistic Regression, we use the option ???family = binomial???

```
mymodel<- glm(Status~ Length + Left + Right+ Bottom + Top + Diagonal, data=train, family='binomial')
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(mymodel)
```

```
##
## Call:
## glm(formula = Status ~ Length + Left + Right + Bottom + Top +
##     Diagonal, family = "binomial", data = train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.749e-05  -2.100e-08  -2.100e-08   2.100e-08   4.928e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5860.18 9635289.98   0.001    1.000
## Length          -19.56   39937.00   0.000    1.000
## Left             27.83  104119.44   0.000    1.000
## Right           -15.98   90449.25   0.000    1.000
## Bottom           46.11   23972.78   0.002    0.998
## Top              43.20   31889.85   0.001    0.999
## Diagonal        -29.17   20571.68  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.0018e+02  on 144  degrees of freedom
```

```
## Residual deviance: 1.0376e-08  on 138  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

Using our model to predict the outcome of the response ???Status??? using the training data set.

```
 p1<- predict(mymodel, train, type='response')
head(p1)
```

```
##            1            2            3            4            6            7
## 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16
```

```
plot(p1)
```

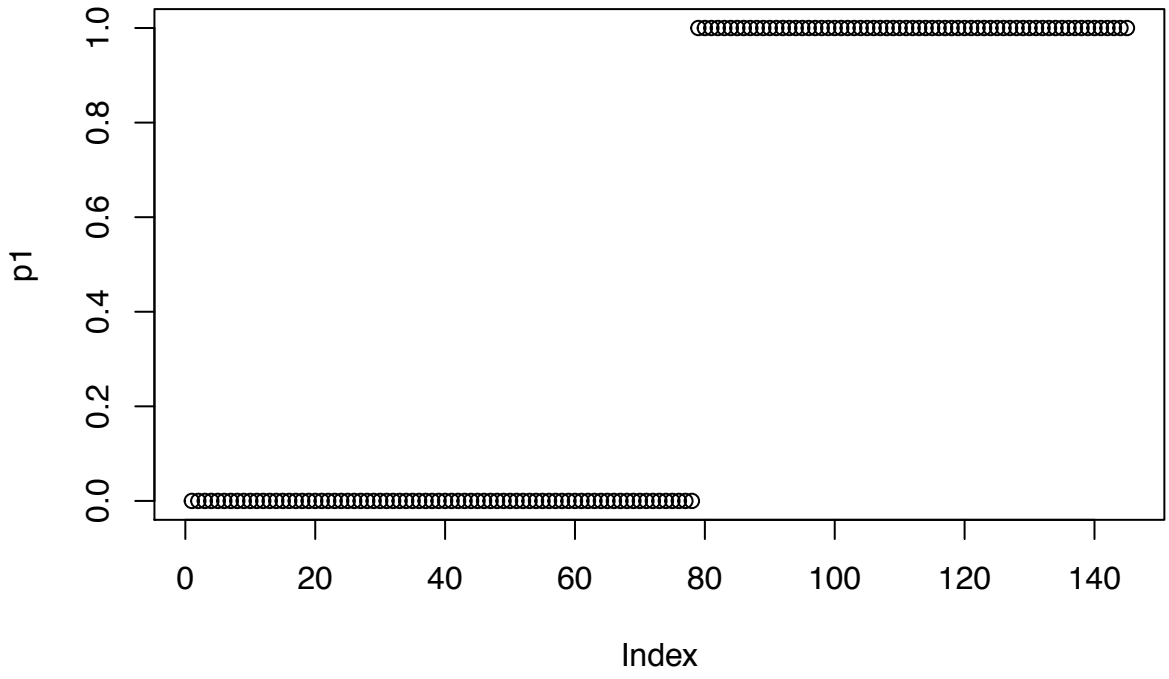


Table to compare Predicted Values vs Actual Values for the training data set (Confusion Matrix)

```
pred1<-ifelse(p1>0.5, 1,0)
table(predicted=pred1, Actual=train$Status)
```

```
##          Actual
## predicted  0  1
##         0 78  0
##         1  0 67
```

From the confusion matrix above, there is no missclassifiation for the training data set

Using our model to predict the outcome of the response ???Status??? using the test data set.

```
p2<- predict(mymodel, test, type='response')
plot(p2)
```

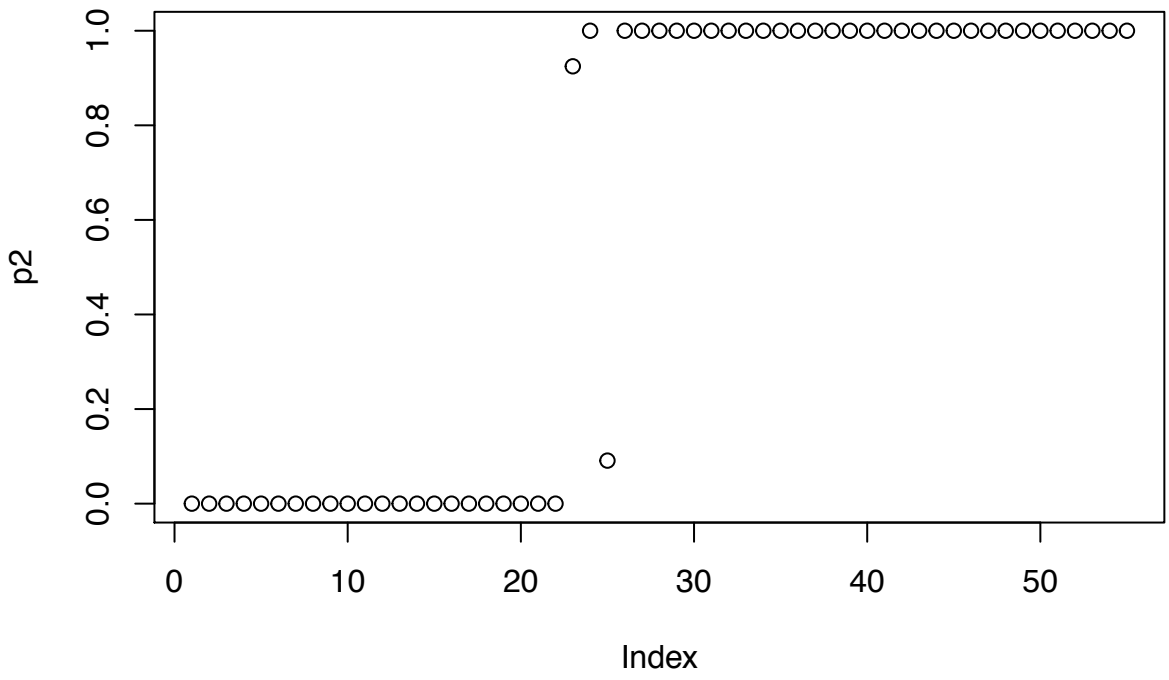


Table to compare Predicted Values vs Actual Values for the test data set (Confusion Matrix)

```
pred2<-ifelse(p2>0.5, 1,0)
table(predicted=pred2, Actual=test$Status)
```

```
##          Actual
## predicted  0  1
##         0 22  1
##         1  0 32
```

From the above Confusion Matrix, there is one misclassification for ???0??? Genuine and zero misclassification for ???1??? (???Counterfeit???)

```
with(mymodel, pchisq(null.deviance-deviance, df.null-df.residual, lower.tail = F))
```

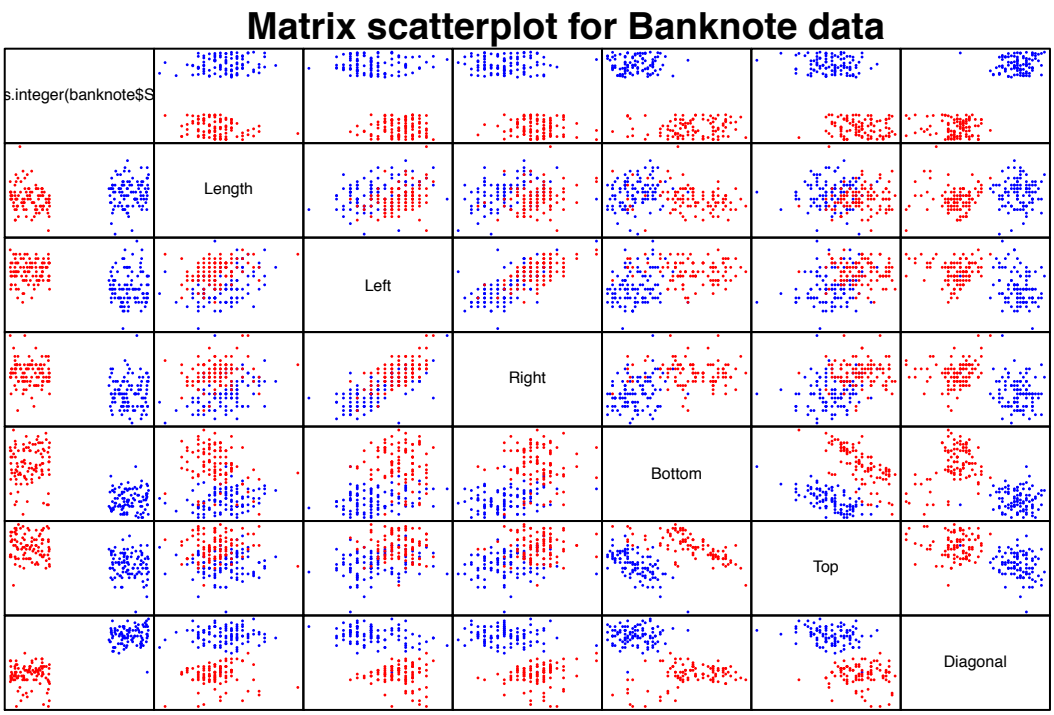
```
## [1] 1.739595e-40
```

MVA Final Project

Adetayo Adetoro

11/22/2020

Matrix Plot



Two stepwse, univariate logistic models:

```
##
## Call:
## glm(formula = Status ~ Length + Left + Right + Bottom + Top +
##     Diagonal, family = binomial, data = BS)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -7.237e-05  -2.100e-08   0.000e+00   2.100e-08   8.216e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.013e+03  3.063e+07   0.000    1.000
## Length      -3.084e+01  1.666e+05   0.000    1.000
## Left         1.075e+01  3.254e+05   0.000    1.000
```

```
## Right       -1.056e+01  2.541e+05   0.000    1.000
## Bottom      -5.876e+01  4.354e+04  -0.001    0.999
## Top         -4.983e+01  3.730e+04  -0.001    0.999
## Diagonal     4.040e+01  3.655e+04   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.7726e+02  on 199  degrees of freedom
## Residual deviance: 2.0593e-08  on 193  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

(Intercept)	Length	Left	Right	Bottom	Top	Diagonal
2013	-30.84	10.75	-10.56	-58.76	-49.83	40.4

```
## Start:  AIC=14
## Status ~ Length + Left + Right + Bottom + Top + Diagonal
##
##            Df Deviance    AIC
## - Right     1    0.000 12.000
## - Left      1    0.000 12.000
## - Length    1    0.000 12.000
## - Top       1    0.000 12.000
## <none>           0.000 14.000
## - Bottom    1   15.056 27.056
## - Diagonal  1   17.778 29.778
##
## Step:  AIC=12
## Status ~ Length + Left + Bottom + Top + Diagonal
##
##            Df Deviance    AIC
## - Left      1    0.000 10.000
## - Length    1    0.000 10.000
## - Top       1    0.000 10.000
## <none>           0.000 12.000
## - Bottom    1   15.703 25.703
## - Diagonal  1   19.186 29.186
##
## Step:  AIC=10
## Status ~ Length + Bottom + Top + Diagonal
##
##            Df Deviance    AIC
## - Length    1    0.000  8.000
## - Top       1    0.000  8.000
## <none>           0.000 10.000
## - Bottom    1   18.011 26.011
## - Diagonal  1   19.947 27.947
##
## Step:  AIC=8
## Status ~ Bottom + Top + Diagonal
##
```

```
##            Df Deviance    AIC
## - Top       1    0.000  6.000
## <none>           0.000  8.000
## - Bottom    1   18.186 24.186
## - Diagonal  1   20.018 26.018
##
## Step:  AIC=6
## Status ~ Bottom + Diagonal
##
##            Df Deviance     AIC
## <none>           0.000   6.000
## - Bottom    1   21.109  25.109
## - Diagonal  1  114.279 118.279
```

```
##
## Call:
## glm(formula = Status ~ Bottom + Diagonal, family = binomial,
##     data = BS)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -6.58e-04  -2.00e-08   0.00e+00   2.00e-08   5.51e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -99422.9  5433597.5  -0.018    0.985
## Bottom         -688.7    37796.3  -0.018    0.985
## Diagonal        751.8    41093.2   0.018    0.985
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.7726e+02  on 199  degrees of freedom
## Residual deviance: 1.0424e-06  on 197  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

(Intercept)	Bottom	Diagonal
-99423	-688.7	751.8

The Logistic Regression

```
##
## Call:
## glm(formula = Status ~ Length + Left + Right + Bottom + Top +
##     Diagonal, family = binomial, data = banknote)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -7.237e-05  -2.100e-08   0.000e+00   2.100e-08   8.216e-05
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.013e+03  3.063e+07   0.000    1.000
## Length      -3.084e+01  1.666e+05   0.000    1.000
## Left         1.075e+01  3.254e+05   0.000    1.000
## Right       -1.056e+01  2.541e+05   0.000    1.000
## Bottom      -5.876e+01  4.354e+04  -0.001    0.999
## Top         -4.983e+01  3.730e+04  -0.001    0.999
## Diagonal     4.040e+01  3.655e+04   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.7726e+02  on 199  degrees of freedom
## Residual deviance: 2.0593e-08  on 193  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

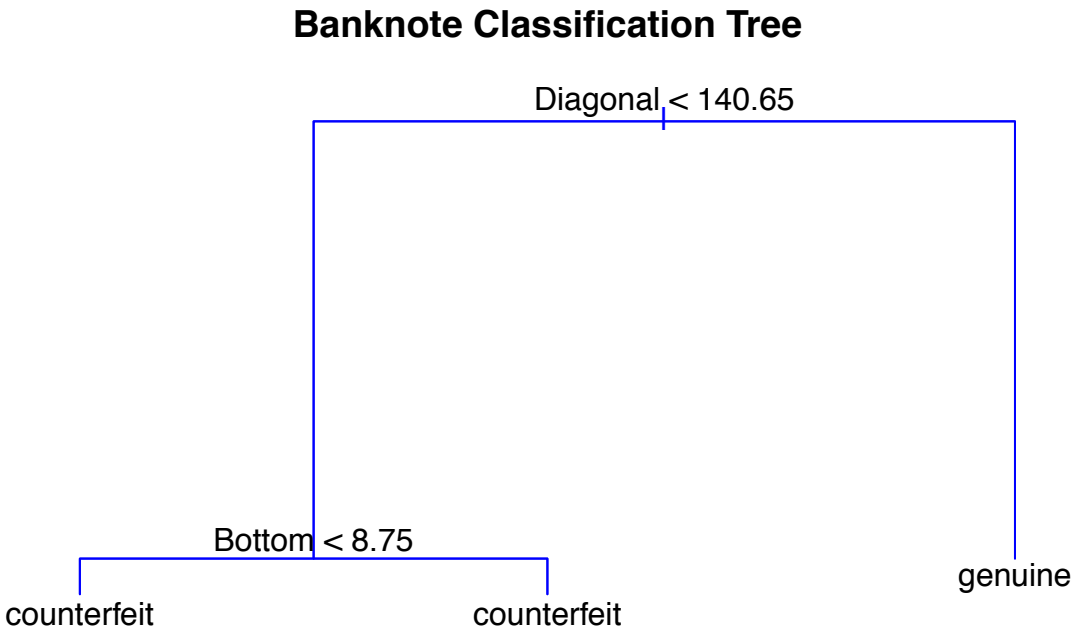
(Intercept)	Length	Left	Right	Bottom	Top	Diagonal
2013	-30.84	10.75	-10.56	-58.76	-49.83	40.4

```
##          189          190          191          192          193          194
## 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16
##          195          196          197          198          199          200
## 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16 2.220446e-16

##             genuine
## counterfeit       0
## genuine           1

##
## glm.pred      counterfeit genuine
##   counterfeit         100       0
##   genuine               0     100

## [1] 1

## [1] 1
```

The diagonal matrix of the confusion matrix whic is equal to the mean indicates correct prediction.

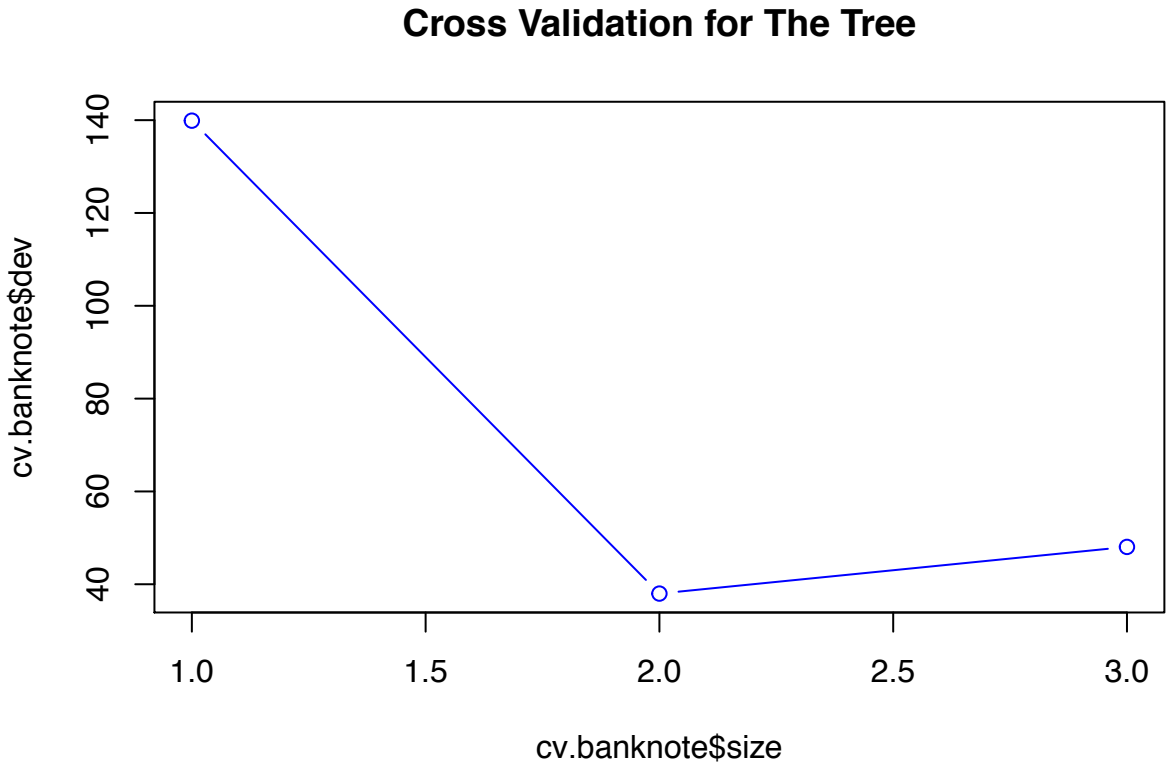
Regression and Classification Trees

```
##
## Classification tree:
## tree(formula = Status ~ ., data = banknote, subset = train)
## Variables actually used in tree construction:
## [1] "Diagonal" "Bottom"
## Number of terminal nodes:  3
## Residual mean deviance:  0.06938 = 6.73 / 97
## Misclassification error rate: 0.02 = 2 / 100
```



The Cross Validation

```
## cross validation of banknote

cv.banknote=cv.tree(tree.banknote)
plot(cv.banknote$size, cv.banknote$dev, type="b", col = "blue")
title(main = "Cross Validation for The Tree")
```



The Prune Tree

```
## Pruning the tree to terminal node

prune.banknote=prune.tree(tree.banknote, best = 3)
plot(prune.banknote, col = "blue")
text(prune.banknote, pretty=0)
title(main = "Prune Tree")
```