MATH 189: Final Report

Kelly Kong

2/8/2021

```
# import libraries
library(lattice)
library(ellipse)
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
library(ggplot2)
library(caret)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.0     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(ROCR)
library(topicmodels)
library(rcompanion)
```

Introduction

The dataset we will be analyzing contains six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes.

I will be trying to predict whether a note is genuine or counterfeit by creating a model using both LDA and Logistic Regression. I will create a base model for both model types and continue to improve it through factor modeling to finalize a model that reduces dimensionality and is accurate.

Data

1. Length of the note
2. Width of the Left-Hand side of the note
3. Width of the Right-Hand side of the note

4. Width of the Bottom Margin
5. Width of the Top Margin
6. Diagonal Length of Printed Area

```
sbn <- read.csv("SBN.txt", sep = '')
```

```
genuine <- replicate(100, 'genuine')
counterfeit <- replicate(100, 'counterfeit')
sbn$Classifier <- c(genuine, counterfeit)
sbn <- sbn[, c(7, 1, 2, 3, 4, 5, 6)]
head(sbn)
```

```
##     Classifier Length  Left Right Bottom  Top Diagonal
## BN1    genuine  214.8 131.0 131.1    9.0  9.7    141.0
## BN2    genuine  214.6 129.7 129.7    8.1  9.5    141.7
## BN3    genuine  214.8 129.7 129.7    8.7  9.6    142.2
## BN4    genuine  214.8 129.7 129.6    7.5 10.4    142.0
## BN5    genuine  215.0 129.6 129.7   10.4  7.7    141.8
## BN6    genuine  215.7 130.8 130.5    9.0 10.1    141.4
```

Data Citation

Citation: https://github.com/tuckermcelroy/ma189/blob/main/Data/SBN.txt, SBN Dataset, March 15th 2021, 12:00pm

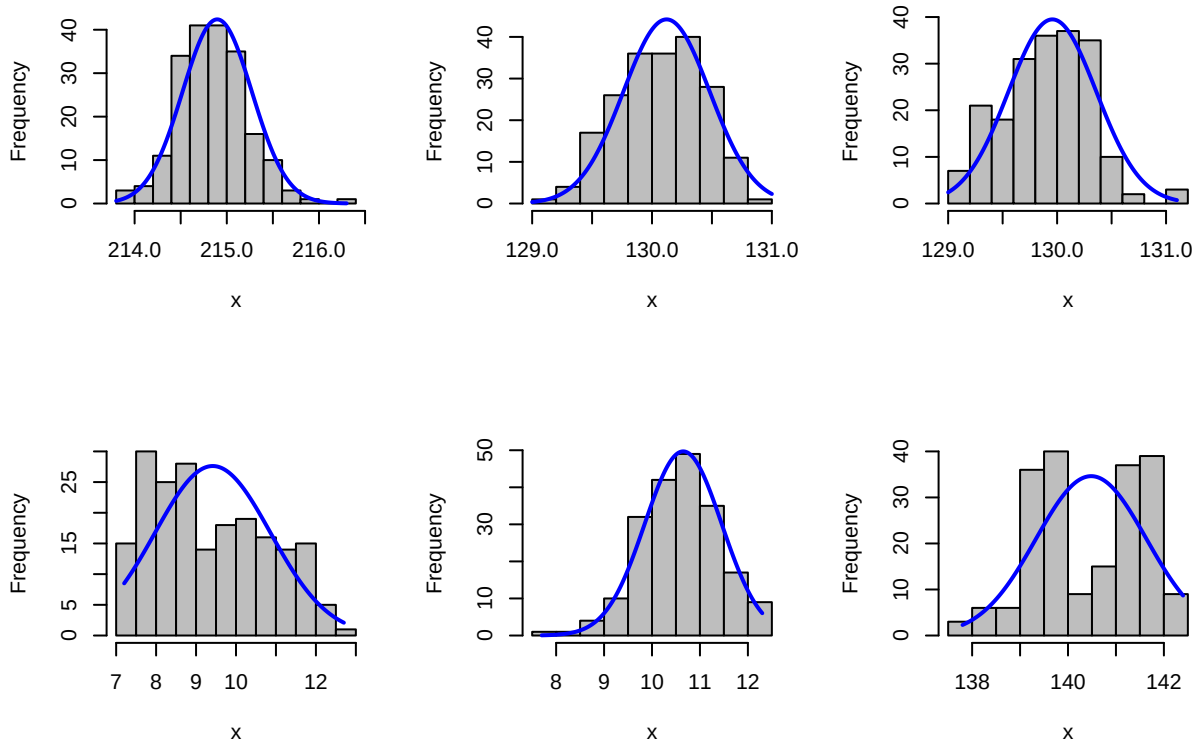
Body

```
mean_var = rbind(apply(sbn[2:7], 2, mean), apply(sbn[2:7], 2, var))
rownames(mean_var) = c("Sample Mean", "Sample Variance")
mean_var
```

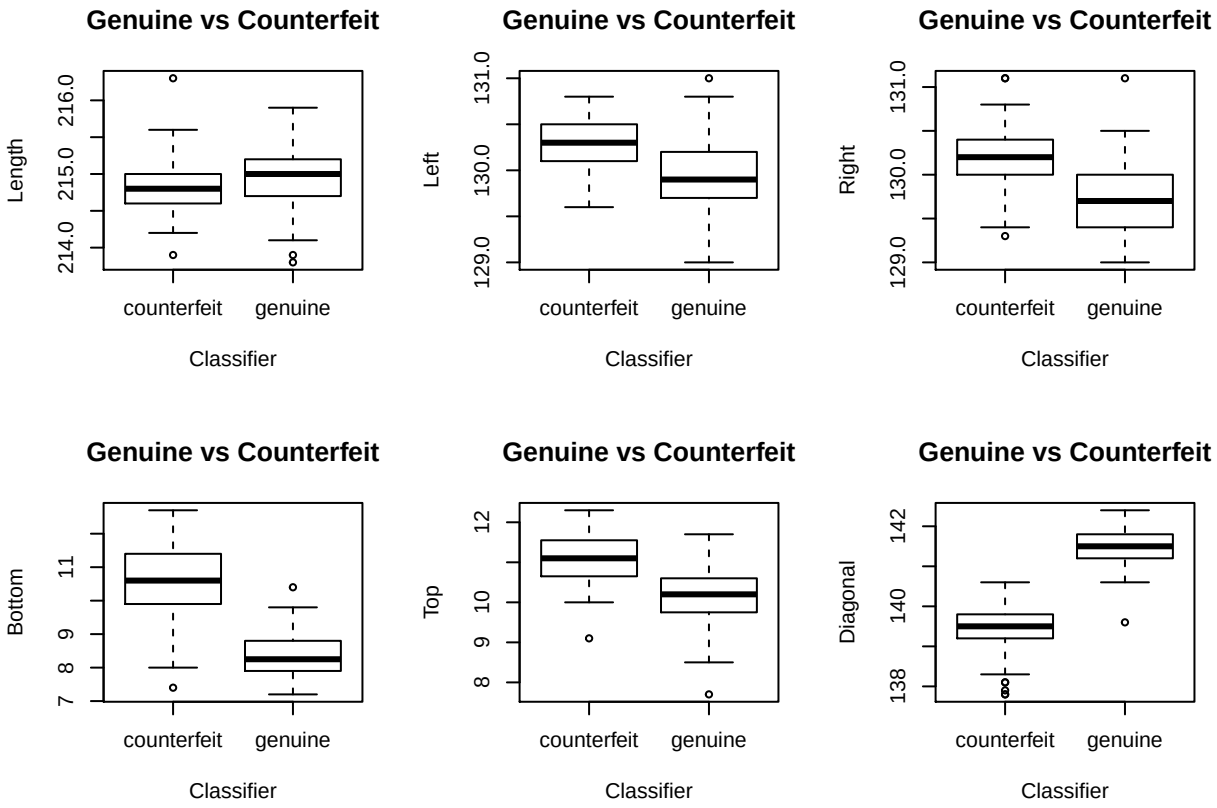
```
##                     Length        Left       Right   Bottom        Top
## Sample Mean     214.896000 130.1215000 129.9565000 9.417500 10.6505000
## Sample Variance   0.141793   0.1303394   0.1632741 2.086878  0.6447234
##                   Diagonal
## Sample Mean     140.483500
## Sample Variance   1.327716
```

```
par(mfrow = c(2, 3))

length = sbn$Length
plotNormalHistogram(length)

left = sbn$Left
plotNormalHistogram(left)

right = sbn$Right
plotNormalHistogram(right)

bottom = sbn$Bottom
plotNormalHistogram(bottom)

top = sbn$Top
plotNormalHistogram(top)
```

```
diagonal = sbn$Diagonal
plotNormalHistogram(diagonal)
```

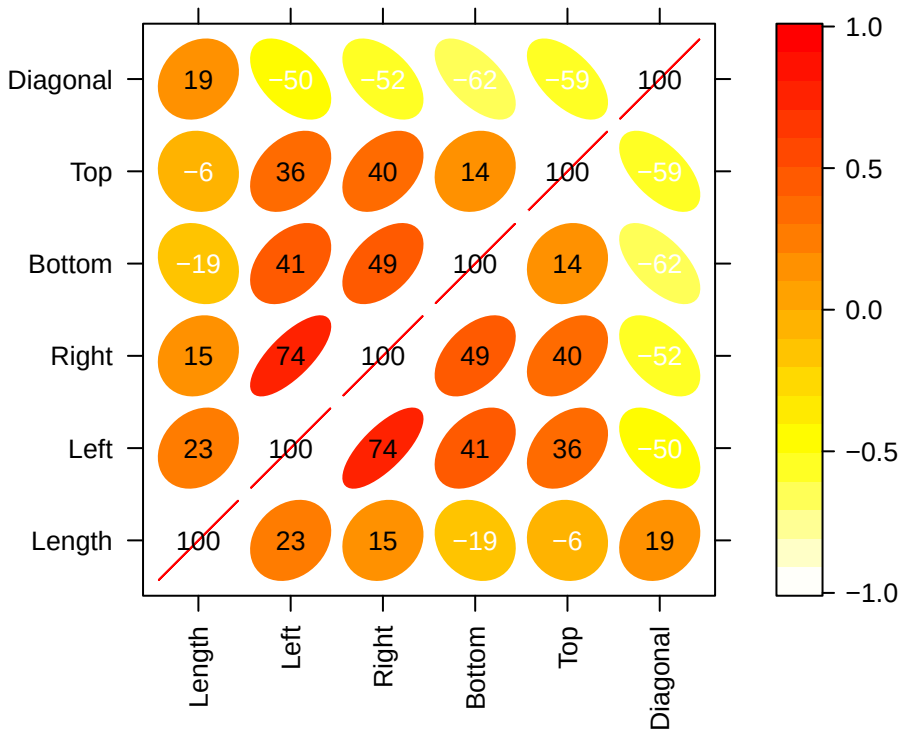



```
par(mfrow = c(2, 3))

boxplot(Length ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")

boxplot(Left ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")

boxplot(Right ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")

boxplot(Bottom ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")

boxplot(Top ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")

boxplot(Diagonal ~ Classifier, data = sbn, main = "Genuine vs Counterfeit")
```



From the graphs above, we can see the mean and variance for both genuine and counterfeit, including some outliers and distributions that aren't normal compared to the rest. This gives me a good overview as to how the data I'm dealing with looks like and allows me to take it into consideration when concluding my analysis.

```
cor_sbn <- cor(sbn[2:7],)
panel.corrgram <- function(x, y, z, subscripts, at, level = 0.9, label = FALSE, ...) {
     require("ellipse", quietly = TRUE)
     x <- as.numeric(x)[subscripts]
     y <- as.numeric(y)[subscripts]
     z <- as.numeric(z)[subscripts]
     zcol <- level.colors(z, at = at,  ...)
     for (i in seq(along = z)) {
         ell=ellipse(z[i], level = level, npoints = 50,
                     scale = c(.2, .2), centre = c(x[i], y[i]))
         panel.polygon(ell, col = zcol[i], border = zcol[i], ...)
     }
     if (label)
         panel.text(x = x, y = y, lab = 100 * round(z, 2), cex = 0.8,
                    col = ifelse(z < 0, "white", "black"))
}

print(levelplot(cor_sbn[seq(1,6), seq(1,6)], at = do.breaks(c(-1.01, 1.01), 20),
           xlab = NULL, ylab = NULL, colorkey = list(space = "right"), col.regions=rev(heat.colors(100)),
           scales = list(x = list(rot = 90)),
           panel = panel.corrgram, label = TRUE,))
```



In this levelplot we can see that the attributes Right and Left are highly correlated in comparison to the rest. Just from anaylzing the dataset itself and this plot, it makes sense as geometrically these correlated attributes would align.

Using Caret Package:

I decided to use the Caret package in R studio as it satisfies the K-fold Cross Validation as I can implement different number of folds / split (80/20) as well as choose which method to use, in which I will be using LDA and Logistic Regression. In the following sections I have written functions for each method. In each function, I am shuffling the dataset then using trainControl to specify that I will be using cross-validation with 5 folds. I then train that model using the specified method and output the results of my 5-fold cross validation.

Choosing K:

A lower value of K is more biased and hence undesirable. On the other hand, higher value of K is less biased, but can suffer from large variability. Because a k value of 5 is a pretty typical and common number I decided to split the data into 5 subsets since our dataset of 200 is quite small and a 40 size validation would give us a 80/20 split.

Assumptions of LDA:

1) Multivariate normality

```
# Performs the Shapiro-Wilk Test for multivariate normality
#install.packages("mvnormtest")
library(mvnormtest)
input <- t(sbn[2:7])
mshapiro.test(input)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
```

```
## W = 0.95953, p-value = 1.758e-05
```

The above function performs the Shapiro-Wilk Test for multivariate normality. Because our p-value is less than 0.05 this indicates that our dataset may not show multivariate normality.

Reference: https://cran.r-project.org/web/packages/mvnormtest/mvnormtest.pdf

2) Homogeneity of variance/covariance

```
bartlett.test(Length ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Length by Classifier
## Bartlett's K-squared = 0.9052, df = 1, p-value = 0.3414
```

```
bartlett.test(Left ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Left by Classifier
## Bartlett's K-squared = 12.229, df = 1, p-value = 0.0004706
```

```
bartlett.test(Right ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Right by Classifier
## Bartlett's K-squared = 3.0082, df = 1, p-value = 0.08285
```

```
bartlett.test(Top ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Top by Classifier
## Bartlett's K-squared = 0.040464, df = 1, p-value = 0.8406
```

```
bartlett.test(Bottom ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Bottom by Classifier
## Bartlett's K-squared = 29.987, df = 1, p-value = 4.35e-08
```

```
bartlett.test(Diagonal ~ Classifier, data = sbn)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Diagonal by Classifier
## Bartlett's K-squared = 4.7962, df = 1, p-value = 0.02852
```

The above function tests the null hypothesis that the variances in each of the groups are the same. From the results, we can reject Left, Bottom and Diagonal since their p-values are less than 0.05. This means that the above mentioned, do not have the same variances. However, we failed to reject Right and Top since their

p-values were higher than 0.05. This indicates that their variances may be the same. To visually see the differences in variances for all 6 predictors, we can look at the boxplot plotted earlier!

3) Multicollinearity

From the levelplot (serving as a correlation matrix) we can see that there's correlation between certain variables, which may suggest multicollinearity, so we shall continue with our assumptions.

4) Independence

Independence is a bit harder to prove, so we are going to assume that there is independence in our dataset (such as in the sample collecting process).

From these assumptions, some of the variables didn't show things like homogeneity of variance/covariance and the dataset didn't show multivariate normality and on top of that, there were things that couldn't be prove and had to be assumed. Because of all of this, there may be some sort of biased in our results in terms of LDA. Regardless, we will continue with creating our LDA model.

K-Fold Cross Validation: LDA

```
# Dividing into training and validation sets: 80% / 20%
set.seed(42)
rows <- sample(nrow(sbn))
sbn <- sbn[rows, ]


lda_model <- function(df) {
        train.control <- trainControl(method = "cv", number = 5, p = 0.8)

        # Train the model
        lda_model <- train(Classifier~., data = df, trControl = train.control, method="lda")

        # Summarize the results
        return(lda_model)
}

lda_base_model = lda_model(sbn)
lda_base_model
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   6 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.995     0.99
```

Assumptions of Logistic Regression:

1) Independence of errors

2) Linearity in the logit for continuous variables

3) Absence of multicollinearity
4) Lack of strongly influential outliers

From our boxplot and histogram, we can see that there are some outliers that may or may not be strongly influential.

Similar to LDA, we are going to assume that all these are true, and proceed with our logistic regression model knowing that there may or may not be biased in the results.

K-Fold Cross Validation: Logistic Regression

```
# Dividing into training and validation sets: 80% / 20%
set.seed(42)
rows <- sample(nrow(sbn))
sbn <- sbn[rows, ]

lg_model <- function(df) {
        train.control <- trainControl(method = "cv", number = 5)

        # Train the model
        lg_model <- train(Classifier~., data = df, trControl = train.control, method="glm")

        # Summarize the results
        return(lg_model)
}

lg_base_model = lg_model(sbn)
```

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

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

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

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

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

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

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

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

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

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

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

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

```
lg_base_model
```

```
## Generalized Linear Model
##
## 200 samples
##   6 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.98      0.96
```

For logistic regression there are errors as there may be too many predictors for the GLM model. This issue, will be addressed in the following sections as I will perform dimension reduction through a factor model.

OVERALL:

The LDA model resulted in a 99 - 99.5% accuracy while the logistic regression model had a 98 - 99% accuracy. Although these models are doing well, we have too many predictors which isn't efficient as it takes up storage and resources. In order to fix this we will reduce the amount of predictors using a factor model and see if we can get an even better (unlikely since predictions are already so high), equally, or close accuracy as the original model with all 6 predictors.

Assumptions for the Factor Model:

The assumptions for Factor Model:

(1) factors and random errors have zero means;

(2) factors have identity covariance matrix;

(3) random errors have diagonal covariance matrix;

(4) factors and random errors are uncorrelated.

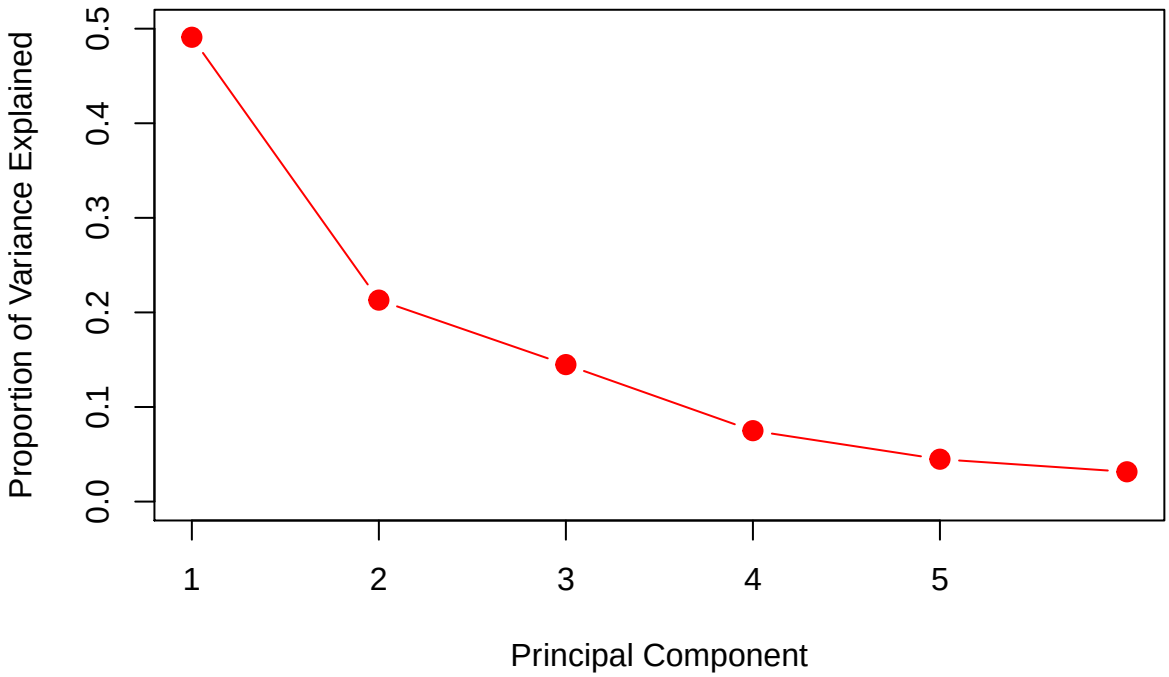
Because we don't know whether all of these assumptions are true, there may or may not be some sort of bias. However, we will continue with our final model using the results from the Factor Model with these assumptions. I believe using the factor model helps us cut down on unncessary factors that may not heavily impact the determination of whether it's genuine or counterfeit. We can see a better relationship between values that the levelplot was suggesting.

```
pca_result = prcomp(sbn[2:7], scale = TRUE)
pca_var = pca_result$sdev^2
pve = pca_var / sum(pca_var)
out2 = cbind(pca_var, pve, cumsum(pve))
colnames(out2) = c("Eigenvalues", "Proportion", "Cumulative")
rownames(out2) = c("Length", "Left", "Right", "Bottom", "Top", "Diagonal")
summary(pca_result)
```

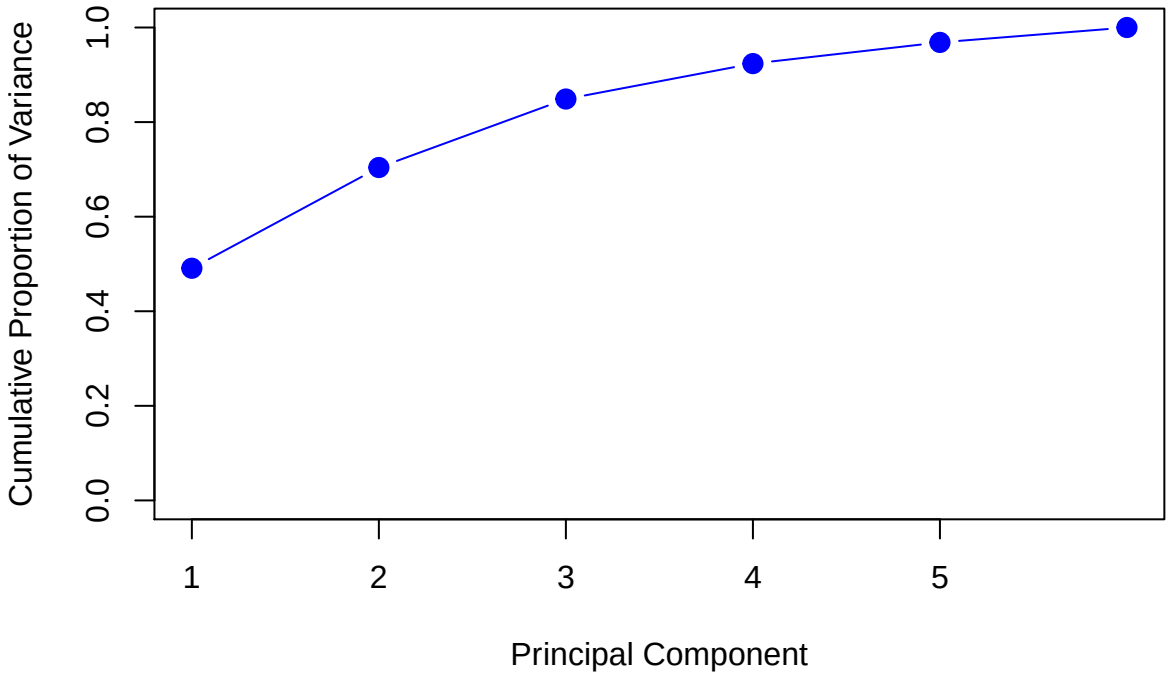
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.7163 1.1305 0.9322 0.67065 0.51834 0.43460
## Proportion of Variance 0.4909 0.2130 0.1448 0.07496 0.04478 0.03148
## Cumulative Proportion  0.4909 0.7039 0.8488 0.92374 0.96852 1.00000
```

Here we can see that as long as we have two or more factors, we would have enough variables to explain the original variance. We are aiming for 60% and higher and more than two would satisfy this requirement.

```
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0, 0.5), xa
axis(1, at = c(1, 2, 3, 4, 5), labels = c(1, 2, 3, 4, 5))
```




```
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance", ylim = c(0,
axis(1, at = c(1, 2, 3, 4, 5), labels = c(1, 2, 3, 4, 5))
```



To visually see the 60% or more of the original variance, I plotted the above graphs to show that as long as we have two or more factors we satisfy what we're aiming for. Also because the elbow of the scree plot is from 2-3, I think using either number of predictors would be the best.

Applying Factor Scores to Our Models

From our scree plot we saw that around 2-3 would be the best as it is the "elbow" of the graph, and after some experimenting (code not shown) 3 performed better than 2:

```r
# Getting factor score matrix (200 x 2)
n.factors <- 3
fa_fit <- factanal(sbn[2:7], n.factors, scores = "regression", rotation="varimax")
head(fa_fit$scores)
```

```
##           Factor1     Factor2    Factor3
## BN6   -0.23115784 -0.64297756  2.4866194
## BN9   -0.76232036 -0.57409582 -1.0516492
## BN178  0.17602343  1.12359629 -0.0885213
## BN113  0.09443955  0.50400193  2.2223524
## BN28   0.45890242 -1.46680750  1.2000017
## BN152  1.75998007 -0.05393159 -1.4733127
```

```r
# Reducing the dimensionality of data to three predictors
reduced_data <- cbind(sbn[1], fa_fit$scores)

#Labeling the data
names(reduced_data) <- c("Classifier", "Factor 1", "Factor 2", "Factor 3")
head(reduced_data)
```

```
##        Classifier    Factor 1    Factor 2   Factor 3
## BN6       genuine -0.23115784 -0.64297756  2.4866194
## BN9       genuine -0.76232036 -0.57409582 -1.0516492
## BN178 counterfeit  0.17602343  1.12359629 -0.0885213
## BN113 counterfeit  0.09443955  0.50400193  2.2223524
## BN28      genuine  0.45890242 -1.46680750  1.2000017
## BN152 counterfeit  1.75998007 -0.05393159 -1.4733127
```

```r
lda_model(reduced_data)
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.995     0.99
```

```r
lg_model(reduced_data)
```

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

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

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

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

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

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

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

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

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

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

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

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

## Generalized Linear Model
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.985     0.97
```

Reducing our predictors from 6 to 3 seem to give us results that about the same or even slightly better than our base model. Again, although it's not shown, I tried two factors and although it did perform well and averaged around the same as the base model, I think three factors was the slightly better option. LDA -> 6 predictors: 99 - 99.5%, 3 predictors: 99.5% Logistic Regression -> 6 predictors: 98 - 99%, 3 predictors: 99 - 99.5%

Through this finding that with three predictors did equally as well and sometimes 0.5% as well as our base model, it shows that some of the predictors didn't contribute much for it to change. This reduces redundancy in our model and improves it through efficiency. However, there are other things to consider that may have affected our results such as a small dataset of only 200 rows (which may explain why our base model did so well) and also the assumptions mentioned before doing our LDA and Logistic Regression model. Those assumptions for the models may create a bias and inaccuracy, however, we will continue with our analysis.

Finding the Best Two Predictors

Linear Model: Reduced Factor Model #1

```r
# prep data for lm() -> change to binary: 1, 0
sbn_glm <- read.csv("SBN.txt", sep = '')
genuine <- replicate(100, 1)
counterfeit <- replicate(100, 0)
sbn_glm$Classifier <- c(genuine, counterfeit)
sbn_glm <- sbn_glm[, c(7, 1, 2, 3, 4, 5, 6)]
head(sbn_glm)
```

```
##     Classifier Length  Left Right Bottom  Top Diagonal
## BN1          1  214.8 131.0 131.1    9.0  9.7    141.0
## BN2          1  214.6 129.7 129.7    8.1  9.5    141.7
## BN3          1  214.8 129.7 129.7    8.7  9.6    142.2
## BN4          1  214.8 129.7 129.6    7.5 10.4    142.0
## BN5          1  215.0 129.6 129.7   10.4  7.7    141.8
## BN6          1  215.7 130.8 130.5    9.0 10.1    141.4
```

```
lm <- lm(Classifier~., data = sbn_glm)
summary(lm)
```

```
##
## Call:
## lm(formula = Classifier ~ ., data = sbn_glm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38691 -0.08306 -0.00763  0.07578  0.57724
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.540e+01  6.617e+00  -3.838 0.000168 ***
## Length       6.667e-04  2.977e-02   0.022 0.982157
## Left         1.108e-01  4.366e-02   2.537 0.011986 *
## Right       -1.130e-01  3.972e-02  -2.844 0.004937 **
## Bottom      -1.487e-01  1.013e-02 -14.669  < 2e-16 ***
## Top         -1.568e-01  1.707e-02  -9.188  < 2e-16 ***
## Diagonal     2.071e-01  1.504e-02  13.768  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1402 on 193 degrees of freedom
## Multiple R-squared:  0.9242, Adjusted R-squared:  0.9218
## F-statistic: 391.9 on 6 and 193 DF,  p-value: < 2.2e-16
```

In this summary table, we can also see that all but length seems to be a significant value, with diagonal, bottom, and top being the most significant out of all 6 predictors. Because their p-value are very close to 0 this indicates high significance.

From this summary, we will use Diagonal, Bottom, and Top as our predictors for our first reduced dimensional model as it suggests high correlation as a predictor.

Fa: Reduced Factor Model #2

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:rcompanion':
##
##     phi
```

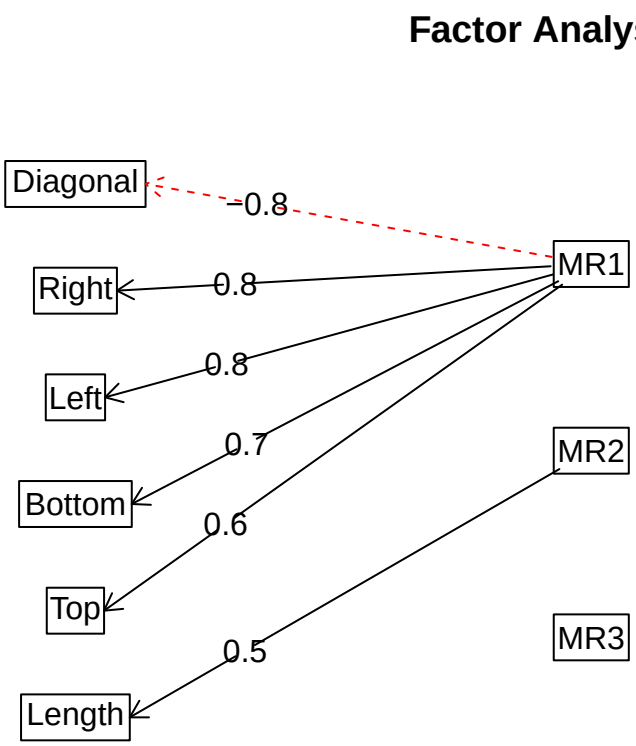
```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
fit <- fa(sbn[2:7], 3, scores = "regression", rotation="varimax")
```

```
## Loading required namespace: GPArotation
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : I
## am sorry, to do these rotations requires the GPArotation package to be installed
```

```
fa.diagram(fit)
```



From this factor analysis graph, it seems that the factors diagonal, right, left, bottom, and top seem to group together for MR1. However we can see that Right and Left have a higher impact in comparison to Diagonal, Bottom and Top. As for the MR2, there is only length that does not seem to have as such a high impact on predicting the genuinety of a bill but still plays an important role toward it. As for MR3, I'm not completely sure whether these types of graphs can graph three or more factors. However, just from looking at the graph, I believe that because right and left give the same computation 0.8, they'd have the same affect. For the third factor, I will choose Bottom as it's the next highest.

From this graph, we will use Right, Bottom, and Length as our predictors for our second reduced dimensional model as it suggests high correlation as a predictor.

MLE: Reduced Factor Model #2

```
# MLE for factor model
n.factors <- 3

# Fit factor model
fa_fit <-factanal(sbn[2:7], n.factors, rotation = "varimax")

# Factor loadings
loading <- fa_fit$loadings[,1:3]
loading
```

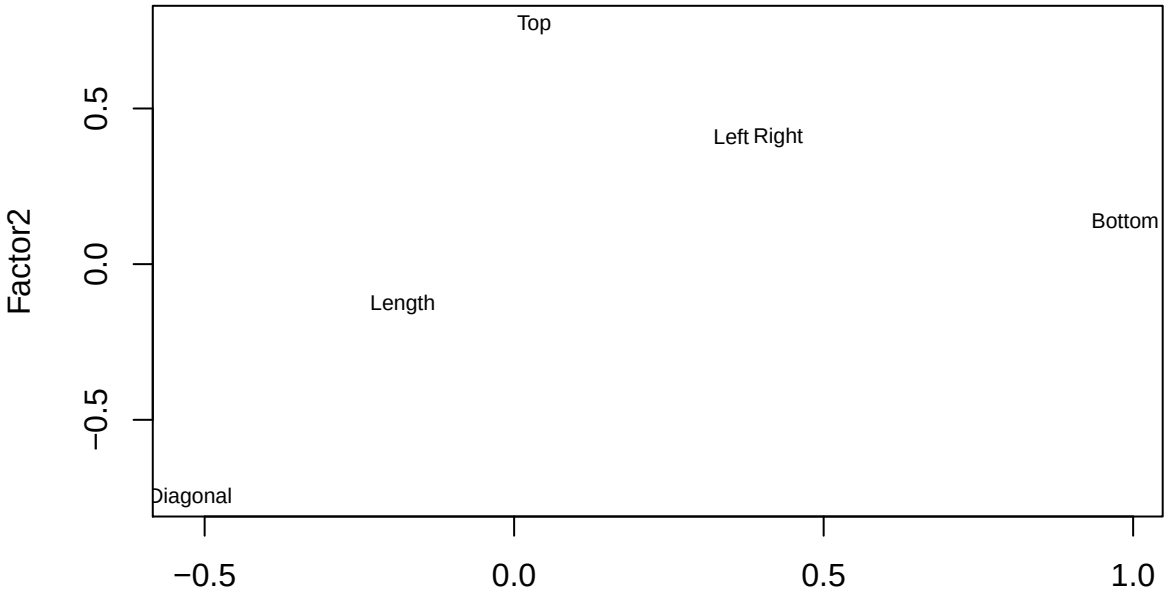
```
##              Factor1    Factor2      Factor3
## Length   -0.17985416 -0.1316173  0.484011079
## Left      0.35096981  0.4103754  0.709647304
## Right     0.42663237  0.4058786  0.601424210
## Bottom    0.98730708  0.1416129  0.013079131
## Top       0.03239197  0.7690831  0.076352362
```

```
## Diagonal -0.52336391 -0.7498327 -0.004236968
```

Here we can see that from using the function factanal, the highest for Factor1 is Bottom with .987, Factor2 is Top with .769, and lastly Factor3 is Left with .709.

From this factanal chart, we will use Bottom, Top, and Left as our predictors for our third reduced dimensional model as it suggests high correlation as a predictor.

```
# plot factor 1 by factor 2
plot(loading,type="n") # set up plot
text(loading,labels=names(sbn[2:7]),cex=.7) # add variable names
```



To visualize the above chart with the loadings, the results above give us a good interpretation of common factors.

With three factors (only two shows), for Factor1 our best option would be Bottom, Factor2 would be Top and lastly, even though there's no Factor3 Axis, we will use Right as it's slightly higher in Factor1 and the same in Factor2 as Left.

From this visualization, we will use Bottom, Top, and Right as our predictors for our last reduced dimensional model as it suggests high correlation as a predictor.

Applying Reduced Dimensions to Models

```
# Reduced Dimension LDA Model 1: Bottom, Top, Diagonal
reduced_sbn1 = sbn[, c(1, 5, 6, 7)]
reduced_lda_model1 = lda_model(reduced_sbn1)
reduced_lda_model1
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
```

```
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.995     0.99
```

```
# Reduced Dimension LDA Model 2: Right, Bottom, Length
reduced_sbn2 = sbn[, c(1, 2, 4, 5)]
reduced_lda_model2 = lda_model(reduced_sbn2)
reduced_lda_model2
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.89      0.78
```

```
# Reduced Dimension LDA Model 3: Bottom, Top, Left
reduced_sbn3 = sbn[, c(1, 3, 5, 6)]
reduced_lda_model3 = lda_model(reduced_sbn3)
reduced_lda_model3
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.96      0.92
```

```
# Reduced Dimension LDA Model 4: Bottom, Top, Right
reduced_sbn4 = sbn[, c(1, 4, 5, 6)]
reduced_lda_model4 = lda_model(reduced_sbn4)
reduced_lda_model4
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.965     0.93
```

```
# Reduced Dimension LG Model 1: Bottom, Top, Diagonal
reduced_lg_model1 = lda_model(reduced_sbn1)
reduced_lg_model1
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.995     0.99
```

```
# Reduced Dimension LG Model 2: Right, Bottom, Length
reduced_lg_model2 = lda_model(reduced_sbn2)
reduced_lg_model2
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.895     0.79
```

```
# Reduced Dimension LG Model 3: Bottom, Top, Left
reduced_lg_model3 = lda_model(reduced_sbn3)
reduced_lg_model3
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
```

```
## Resampling results:
##
##   Accuracy  Kappa
##   0.96      0.92
```

```
# Reduced Dimension LG Model 4: Bottom, Top, Right
reduced_lg_model4 = lda_model(reduced_sbn4)
reduced_lg_model4
```

```
## Linear Discriminant Analysis
##
## 200 samples
##   3 predictor
##   2 classes: 'counterfeit', 'genuine'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 160, 160, 160, 160, 160
## Resampling results:
##
##   Accuracy  Kappa
##   0.965     0.93
```

Conclusion

```
folds <- c('fold1 accuracy', 'fold2 accuracy', 'fold3 accuracy', 'fold4 accuracy', 'fold5 accuracy', 'av
lda_base <- c(lda_base_model$resample$Accuracy, lda_base_model$results$Accuracy)
lg_base <- c(lg_base_model$resample$Accuracy, lg_base_model$results$Accuracy)

reduced_lda_model1 <- c(reduced_lda_model1$resample$Accuracy, reduced_lda_model1$results$Accuracy)
reduced_lg_model1 <- c(reduced_lg_model1$resample$Accuracy, reduced_lg_model1$results$Accuracy )
reduced_lda_model2 <- c(reduced_lda_model2$resample$Accuracy, reduced_lda_model2$results$Accuracy)
reduced_lg_model2 <- c(reduced_lg_model2$resample$Accuracy, reduced_lg_model2$results$Accuracy)
reduced_lda_model3 <- c(reduced_lda_model3$resample$Accuracy, reduced_lda_model3$results$Accuracy)
reduced_lg_model3 <- c(reduced_lg_model3$resample$Accuracy, reduced_lg_model3$results$Accuracy)
reduced_lda_model4 <- c(reduced_lda_model4$resample$Accuracy, reduced_lda_model4$results$Accuracy)
reduced_lg_model4 <- c(reduced_lg_model4$resample$Accuracy, reduced_lg_model4$results$Accuracy )

across_models <- data.frame(folds, lda_base, lg_base,
                            reduced_lda_model1, reduced_lg_model1,
                            reduced_lda_model2, reduced_lg_model2,
                            reduced_lda_model3, reduced_lg_model3,
                            reduced_lda_model4, reduced_lg_model4)
across_models
```

```
##            folds lda_base lg_base reduced_lda_model1 reduced_lg_model1
## 1 fold1 accuracy    1.000   0.975              1.000             1.000
## 2 fold2 accuracy    1.000   0.975              1.000             0.975
## 3 fold3 accuracy    1.000   1.000              0.975             1.000
## 4 fold4 accuracy    1.000   0.975              1.000             1.000
## 5 fold5 accuracy    0.975   0.975              1.000             1.000
## 6   avg accuracy    0.995   0.980              0.995             0.995
##   reduced_lda_model2 reduced_lg_model2 reduced_lda_model3 reduced_lg_model3
## 1              0.875             0.875              0.975             0.950
```

```
## 2              0.875             0.825              0.975             0.975
## 3              0.925             0.900              0.950             0.900
## 4              0.875             1.000              0.950             0.975
## 5              0.900             0.875              0.950             1.000
## 6              0.890             0.895              0.960             0.960
##   reduced_lda_model4 reduced_lg_model4
## 1              0.975             0.950
## 2              1.000             0.975
## 3              0.950             0.975
## 4              0.950             0.950
## 5              0.950             0.975
## 6              0.965             0.965
```

This table shows the accuracy per fold for each model used. In total we have one base model per method (LDA and Logistic Regression) and four models with varying predictors and accuracies.

LDA Analysis: Fold1: lda_base (all 6 predictors) and reduced_lda_model1 (Bottom, Top, Diagonal) did better - Because they both give us a 100% accuracy and reduced_lda_model1 has less predictors we will choose that one since it's more efficient

Fold2: lda_base (all 6 predictors), reduced_lda_model2 (Bottom, Top, Diagonal), and reduced_lda_model4 (Bottom, Top, Right) - Because both reduced_lda_model2 and reduced_lda_model have the same number of predictors and efficiency, either would be fine for fold2

Fold3: lda_base (all 6 predictors) - It seems like lda_base did the best with 100% accuracy, yet reduced_lda_model2 (more efficient but less accurate) is a close second

Fold4: lda_base (all 6 predictors), reduced_lg_model1 (Bottom, Top, Diagonal),
- Although they both have 100% accuracy, reduced_lg_model1 is more efficient for fold4

Fold5: reduced_lda_model1 (Bottom, Top, Diagonal) - Because it's the only model with 100% accuracy for fold5

Overall: lda_base (all 6 predictors) and reduced_lda_model1 (Bottom, Top, Diagonal) tied, but we will choose reduced_lda_model1 since it requires less predictors yet outputs the same accuracy making it more efficient by cutting down from 6 factors to 3 factors (1/2).

Logistic Regression Analysis: Fold1: reduced_lg_model1 (Bottom, Top, Diagonal) - Because reduced_lg_model1 was the only model that gave us 100% accuracy for fold1

Fold2: lg_base (all 6 predictors), reduced_lg_model1 (Bottom, Top, Diagonal), reduced_lg_model3 (Bottom, Top, Left), and reduced_lg_model4 (Bottom, Top, Right) - Because these all had accuracies of 97.5% we can choose between any of the three reduced models since they would be more efficient

Fold3: lg_base (all 6 predictors), reduced_lg_model1 (Bottom, Top, Diagonal) - Both lg_base and reduced_lg_model1 got 100%, and again we will choose reduced_lda_model1 for fold3 due to efficiency

Fold4: reduced_lg_model1 (Bottom, Top, Diagonal), - Because they both give us a 100% accuracy and reduced_lda_model1 has less predictors we will choose that one since it's more efficient

Fold5: reduced_lda_model1 (Bottom, Top, Diagonal), and reduced_lda_model3 (Bottom, Top, Left) - Both would be suitable as they both have 3 factors with 100% accuracy for fold5

Overall: Reduced_lda_model1 (Bottom, Top, Diagonal) has the highest accuracy for Logistic Regression in which it peformed than the bast model with all 6 attributes. This may show that for logistic regression in specific, there are factors that are redundant causing the accuracy to be ever so slightly lower.

Although this dataset is quite small and 6 factors isn't as comparable to other tables with over 10+ factors, I think using a factor model to reduce dimension helped improve this model in it's efficiency. 'reduced_lda_model1' and 'reduced_lg_model1' with the factors/predictors (Bottom, Top, Diagonal) got 99.5% across both methods of LDA and LG which is better than the base_model with all 6 features in

terms of the LG model. If we were to choose a model that would work the best, either LDA and LG for my first reduced model would work well. However, again there are bias, certain assumptions that are not met, in addition to our small data sample, that could affect our results. Because our base model was already performing very well, in a future report, we could explore these models even further on a bigger dataset with more predictors to see the full effects of dimensional reduction.

NOTE: ty to the TA (or even professor?) who is grading this because this report is very long and tedious! also thank you for an amazing quarter. :'))