Math 189: Final Project

Yilan Guo

3/12/2021

Introduction

In this project, we will use the Swiss bank notes dataset and determine whether we predict whether a note is false or counterfeit using supervised learning. Through the statistical analysis, we are going to using both LDA and logistic regression on each fold and also to repeat the process after using factor analysis to reduce the dimension. Finally, we will add the results across folds and determine what the best model is and if factor analysis is necessary or not.

Swiss Bank Notes

The dataset Swiss bank notes dataset comes from math189 class github repo[1] SNB.txt. Accessed on March 15, 2021, The dataset contains six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes:

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

```
swiss <- read.table("~/Downloads/MATH_189/ma189-main/Data/SBN.txt")
```

First, let us have a peak at the dataset Because the data does not contain a binary response, we have to manually add teh binary responses into the dataset. The first 100 observation are genuine and the last 100 observations are counterfeit.

```
swiss$Genuine = append(rep(1, 100),rep(0,100))
head(swiss)
```

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

[1] repo: Math189 *SBN.txt*, adopted from: "Multivariate Statistics: A practical approach", by Bernhard Flury and Hans Riedwyl, Chapman and Hall, 1988.

Analysis

Explore and visualize the data.

```
# means in genuine and counterfeit banknotes
means = NULL
means = rbind(means,colMeans(swiss[swiss$Genuine == 1,][,-7]))
means = rbind(means,colMeans(swiss[swiss$Genuine == 0,][,-7]))
means
```

Means in genuine and counterfeit banknotes

```
##       Length    Left   Right Bottom    Top Diagonal
## [1,] 214.969 129.943 129.720  8.305 10.168  141.517
## [2,] 214.823 130.300 130.193 10.530 11.133  139.450
```

```
# correlation
cor(swiss[,-7])
```

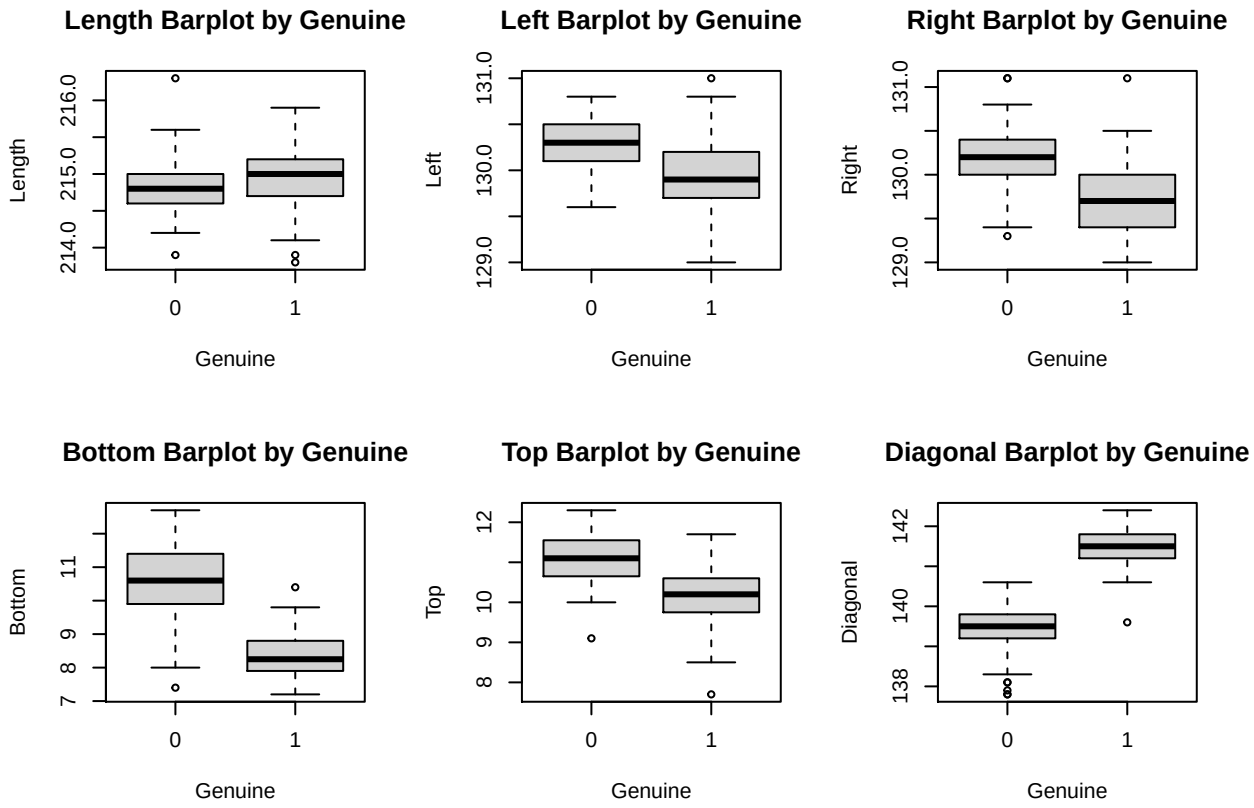
Correlation between variable

```
##               Length       Left      Right     Bottom         Top   Diagonal
## Length     1.00000000  0.2312926  0.1517628 -0.1898009 -0.06132141  0.1943015
## Left       0.23129257  1.0000000  0.7432628  0.4137810  0.36234960 -0.5032290
## Right      0.15176280  0.7432628  1.0000000  0.4867577  0.40067021 -0.5164755
## Bottom    -0.18980092  0.4137810  0.4867577  1.0000000  0.14185134 -0.6229827
## Top       -0.06132141  0.3623496  0.4006702  0.1418513  1.00000000 -0.5940446
## Diagonal   0.19430146 -0.5032290 -0.5164755 -0.6229827 -0.59404464  1.0000000
```

Boxplot

- Use boxplot to display the variable across genuine and counterfeit note.

```
# Boxplot
par(mfrow = c(2, 3))
boxplot(Length~Genuine,data=swiss, main="Length Barplot by Genuine",
   xlab="Genuine", ylab="Length")
boxplot(Left~Genuine,data=swiss, main="Left Barplot by Genuine",
   xlab="Genuine", ylab="Left")
boxplot(Right~Genuine,data=swiss, main="Right Barplot by Genuine",
   xlab="Genuine", ylab="Right")
boxplot(Bottom~Genuine,data=swiss, main="Bottom Barplot by Genuine",
   xlab="Genuine", ylab="Bottom")
boxplot(Top~Genuine,data=swiss, main="Top Barplot by Genuine",
   xlab="Genuine", ylab="Top")
boxplot(Diagonal~Genuine,data=swiss, main="Diagonal Barplot by Genuine",
   xlab="Genuine", ylab="Diagonal")
```



As we could see from the boxplots, Right, Bottom, and Diagonal of the genuine notes and counterfeit notes have the most difference in medians

```
library(lattice)
library(ellipse)
```

Correlation Plot

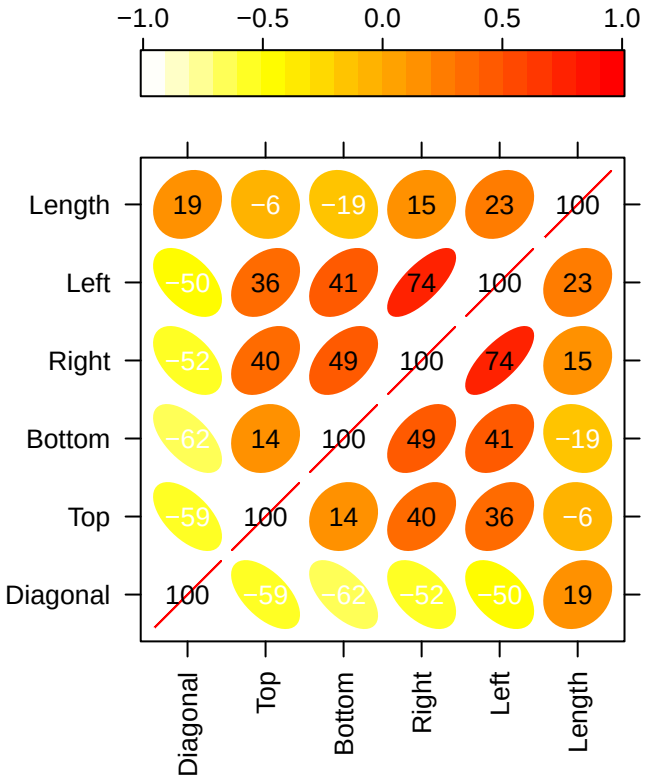
```
##
## Attaching package: 'ellipse'

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

```
cor_df <- cor(swiss[,-7])
# Function to generate correlation plot
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"))
}

# generate correlation plot
print(levelplot(cor_df[seq(6,0), seq(6,0)], at = do.breaks(c(-1.01, 1.01), 20),
          xlab = NULL, ylab = NULL, colorkey = list(space = "top"),
          col.regions=rev(heat.colors(100)),
          scales = list(x = list(rot = 90)),
          panel = panel.corrgram, label = TRUE))
```



Shown above, the variable left and right have the highest correlation, which is expectable since each note is in rectangle shape. This correlation would help us further analyze the factor analysis.

Task 2 Spling Training and Validation Sets

Divide into training and validation sets (which each must have some of the genuine and counterfeit notes), implementing K-fold cross-validation.

Compared to Leave-one-out, K-fold cross validation has the computational feasibility: K-fold CV only requires fitting the statistical learning method. And K-fold CV tends to give a more accurate result.

I will use 10 folds for this project (k = 10) and naturally each validation set has 20 observations.

Here is two way we could approach to this problem, one is directly using the **caret** package, which stands for Classification And REgression Training; the other way is to use caret to create fold, and we mannually separate the validation and training set.

```
library(caret)
```

Caret Package Method

```
## Loading required package: ggplot2
```

```
library(e1071)
set.seed(123)
fit_control = trainControl(method = 'cv', number = 10)
```

Manually separating the validation set and traning set. The method createFolds helps to split the validation sets and returns their index positions. By spliting the validation set and traning set manually, we are able to fix and visualize each validation set and training set.

```
library(caret)
validation = createFolds(swiss$Genuine, k = 10, list = TRUE, returnTrain = FALSE)
validation1 = swiss[validation$Fold01,]
training1 = swiss[-validation$Fold01,]
validation2 = swiss[validation$Fold02,]
training2 = swiss[-validation$Fold02,]
validation3 = swiss[validation$Fold03,]
training3 = swiss[-validation$Fold03,]
validation4 = swiss[validation$Fold04,]
training4 = swiss[-validation$Fold04,]
validation5 = swiss[validation$Fold05,]
training5 = swiss[-validation$Fold05,]
validation6 = swiss[validation$Fold06,]
training6 = swiss[-validation$Fold06,]
validation7 = swiss[validation$Fold07,]
training7 = swiss[-validation$Fold07,]
validation8 = swiss[validation$Fold08,]
training8 = swiss[-validation$Fold08,]
validation9 = swiss[validation$Fold09,]
training9 = swiss[-validation$Fold09,]
validation10 = swiss[validation$Fold10,]
training10 = swiss[-validation$Fold10,]
training_list = list(training1,training2,training3,training4,training5,
                     training6,training7,training8,training9,training10)
validation_list = list(validation1,validation2,validation3,validation4,
                       validation5,validation6,validation7,validation8,
                       validation9,validation10)
```

Task 3 LDA and Logistic Regression

On each fold, classify using both LDA and logistic regression, i.e., fit both models on the training set and evaluate performance on the validation set.

LDA Assumptions:

- Assumptions for the LDA are similar to those for MANOVA:

1. The data from group k has **common mean** vector $\underline{\mu}^{(k)}$, i.e.,

$$\mathbb{E}[x_{ij}^{(k)}] = \underline{\mu}_j^{(k)}.$$

   (The m components of the vector correspond to the m variables.) We assume this assumption hold since we believe each observation on each site is a random variable that is iid.
2. **Homoskedasticity**: The data from all groups have common covariance matrix $\mathbf{\Sigma}$, i.e.,

$$\mathbf{\Sigma} = \text{Cov}[\underline{x}_i^{(k)}, \underline{x}_i^{(k)}]$$

   for any record i, and the matrix does not depend on k (the group index). To test equal covariance, I conducted a Box's M-test, which tests for homogeneity of covariance matrices (the variances in each group are roughly equal) using the data obtained from multivariate normal chemical variable data according to one classification factor – site. The test is based on the chi-square approximation.

```
library(biotools)
```

```
## Loading required package: MASS
```

```
## ---
## biotools version 4.0
```

```
##
```

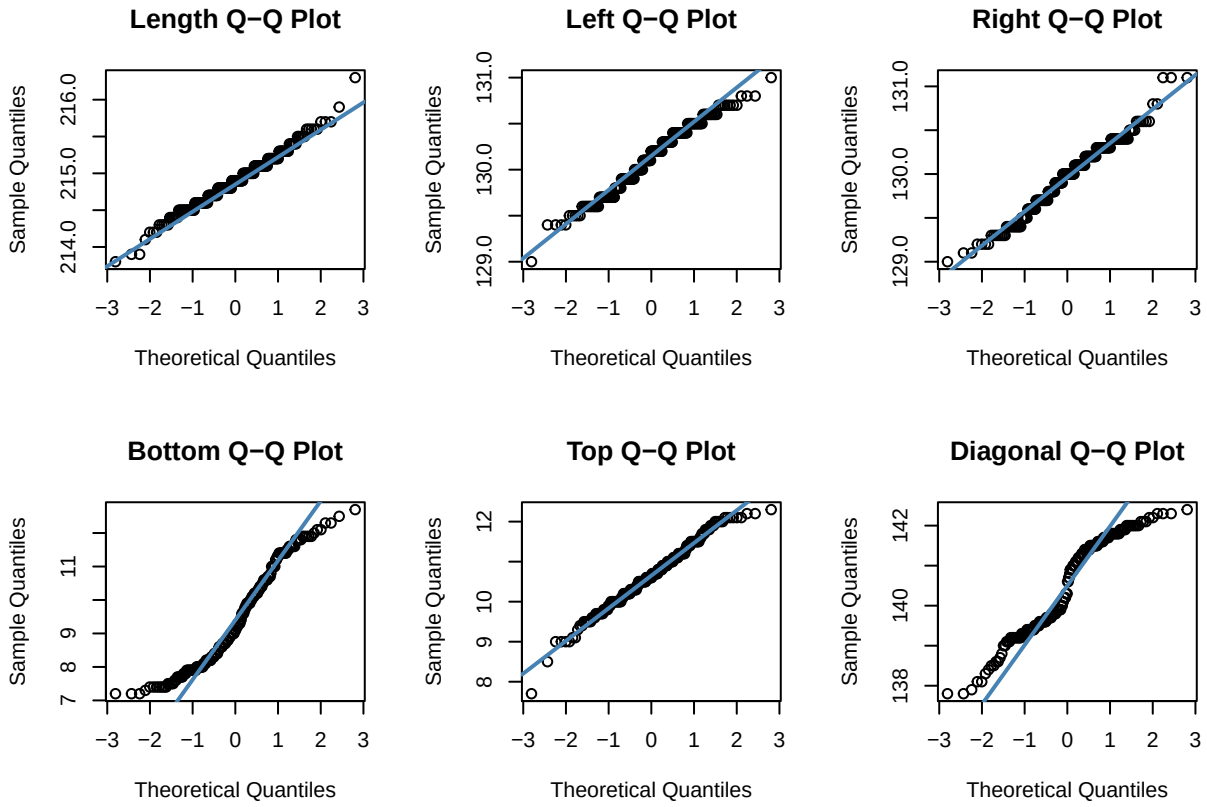
```
boxM(data = swiss[,-7],group = swiss[,7])
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  swiss[, -7]
## Chi-Sq (approx.) = 121.9, df = 21, p-value = 3.198e-16
```

Because the p-value is much greater than 0.05, we fail to reject the null which claims that the variances in each group are roughly equal.

3. **Independence**: The observations are independently sampled. For this project, we assume that that each observation is **i.i.d**.
4. **Normality**: The data are multivariate normally distributed. Univariate demonstrated through Q-Q plot. Almost every Q-Q plot follows a normal distribution, except the bottom and Diagonal are slightly tilted.

```r
par(mfrow = c(2, 3))
qqnorm(swiss$Length, pch = 1,main = "Length Q-Q Plot")
qqline(swiss$Length, col = "steelblue", lwd = 2)
qqnorm(swiss$Left, pch = 1,main = "Left Q-Q Plot")
qqline(swiss$Left, col = "steelblue", lwd = 2)
qqnorm(swiss$Right, pch = 1,main = "Right Q-Q Plot")
qqline(swiss$Right, col = "steelblue", lwd = 2)
qqnorm(swiss$Bottom, pch = 1,main = "Bottom Q-Q Plot")
qqline(swiss$Bottom, col = "steelblue", lwd = 2)
qqnorm(swiss$Top, pch = 1,main = "Top Q-Q Plot")
qqline(swiss$Top, col = "steelblue", lwd = 2)
qqnorm(swiss$Diagonal, pch = 1,main = "Diagonal Q-Q Plot")
qqline(swiss$Diagonal, col = "steelblue", lwd = 2)
```



Logistic Regression Assumptions:

1. The outcome is a **binary or dichotomous variable**. The output Genuine is a binary vairable; eithor 1 or 0.
2. There is a **linear relationship** between the logit of the outcome and each predictor variables. Recall that the logit function is logit(p) = log(p/(1-p)), where p is the probabilities of the outcome. Assume to be true.
3. There is **no influential values (extreme values or outliers)** in the continuous predictors Referring to the boxplots, there are only few outliers among the six variables.

4. There is **no high intercorrelations (i.e. multicollinearity)** among the predictors. There might contain some correlations among the variables; for example, the Right and Left size should be equal if the note is genuine, and also the diagonal might have a relationship with Length and Left/Right.

Caret Package Method Caret directly enables us to train and resample the data.

```
lda_fit <- train(as.factor(Genuine) ~ Length+Left+Right+Bottom+Top+Diagonal,
                 data = swiss, method = "lda",trControl = fit_control)
lda_fit$resample
```

```
##    Accuracy Kappa Resample
## 1      1.00   1.0   Fold01
## 2      0.95   0.9   Fold02
## 3      1.00   1.0   Fold03
## 4      1.00   1.0   Fold04
## 5      1.00   1.0   Fold05
## 6      1.00   1.0   Fold06
## 7      1.00   1.0   Fold07
## 8      1.00   1.0   Fold08
## 9      1.00   1.0   Fold09
## 10     1.00   1.0   Fold10
```

```
lr_fit <- train(as.factor(Genuine) ~ Length+Left+Right+Bottom+Top+Diagonal,
                data = swiss, method = "glm",family = 'binomial'
                ,trControl = fit_control)
lr_fit$resample
```

```
##    Accuracy Kappa Resample
## 1      1.00   1.0   Fold01
## 2      1.00   1.0   Fold02
## 3      0.95   0.9   Fold03
## 4      1.00   1.0   Fold04
## 5      0.95   0.9   Fold05
## 6      1.00   1.0   Fold06
## 7      0.95   0.9   Fold07
## 8      1.00   1.0   Fold08
## 9      0.95   0.9   Fold09
## 10     1.00   1.0   Fold10
```

Manually Applying Functions Also, we can mannually apply the LDA/LR to each traning sets and calculate the accuracy rate using validation sets.

```
lr_mean = NULL
lda_mean = NULL
library(MASS)
for (i in 1:10){
  fit.lr = glm(Genuine~Length+Left+Right+Bottom+Top+Diagonal,
               data=training_list[[i]],family=binomial,maxit = 100)
  prob = predict(fit.lr, validation_list[[i]], type="response")
  pred = rep(0, dim(validation_list[[i]])[1])
  pred[prob > .5] = 1
```

```
  lr_mean = rbind(lr_mean, mean(pred == (validation_list[[i]]$Genuine==1)))

  fit.lda = lda(Genuine ~ Length+Left+Right+Bottom+Top+Diagonal,
                data=training_list[[i]])
  pred.lda = predict(fit.lda, newdata = validation_list[[i]])
  lda_mean = rbind(lda_mean,mean(pred.lda$class== validation_list[[i]]$Genuine))
}
```

To better visualize the accuracy rate, we will create a table to summarize our findings.

```
names = c("Fold 1","Fold 2","Fold 3","Fold 4","Fold 5",
          "Fold 6","Fold 7","Fold 8","Fold 9","Fold 10")

result = data.frame(names,lda_mean,lr_mean)
colnames(result) <- c("Name","LDA","LR")
result
```

```
##       Name  LDA   LR
## 1   Fold 1 1.00 0.90
## 2   Fold 2 1.00 0.95
## 3   Fold 3 1.00 1.00
## 4   Fold 4 1.00 1.00
## 5   Fold 5 1.00 1.00
## 6   Fold 6 0.95 0.95
## 7   Fold 7 1.00 1.00
## 8   Fold 8 1.00 1.00
## 9   Fold 9 1.00 1.00
## 10 Fold 10 1.00 1.00
```

The two methods of using package and manually computing yield the (almost) same result; slight difference can due to probability.

Task 4: Reduce the dimension

Refine the six covariates using a factor model to reduce the dimension and remove any redundancy, re-running analysis on the factor scores.

Factor Analysis is a method for modeling observed variables, and their covariance structure, in terms of a smaller number of underlying unobservable (latent) factors.

Assumptions for factor model:

- Assumptions on mean vector:

1. The common factors all have mean zero:
$$\mathbb{E}[\underline{f}] = 0.$$
2. The specific factors (or random errors) all have mean zero:
$$\mathbb{E}[\underline{\epsilon}] = 0.$$

- Assumptions on variance-covariance matrix:

1. Common factors satisfy:
$$\text{Cov}[\underline{f}] = I_m,$$
where I_m is an $m \times m$-dimensional identity matrix.
2. Random errors satisfy:
$$\text{Cov}[\underline{\epsilon}] = \mathbf{\Psi} = \text{diag}(\psi_1, \ldots, \psi_p).$$
3. Common factors and random errors are uncorrelated:
$$\text{Cov}[\underline{f}, \underline{\epsilon}] = 0.$$

- With the assumptions imposed on the mean vector, we can avoid the flexibility issue of the mean vector. Now, the mean vector is fully explained by the intercept:
$$\mathbb{E}[\underline{x}] = \underline{\mu} + \mathbf{L}\,\mathbb{E}[\underline{f}] + \mathbb{E}[\underline{\epsilon}] = \underline{\mu}.$$
- With the assumptions imposed on variance-covariance matrix, we can avoid the flexibility issue of $\mathbf{\Sigma}$:
$$\text{Cov}[\underline{x}] = \mathbf{L}\,\text{Cov}[\underline{f}]\,\mathbf{L}' + \text{Cov}[\underline{\epsilon}] = \mathbf{L}\,\mathbf{L}' + \mathbf{\Psi}.$$
This is the matrix of factor loadings times its transpose, plus a diagonal matrix containing the specific variances.

And for this project, we assume those assumptions hold.

Factor Analysis

As the scree plot shown below, I will choose m = 2 as the "elbow" occurs around m=2.

```
pca_result = prcomp(swiss[,-7], scale = TRUE)
eigen_val = pca_result$sdev^2
pve = eigen_val/sum(eigen_val)

n.factors = 2
fa_fit = factanal(swiss[,-7], n.factors,
                  romatation = "varimax", scores = "regression")
loading = fa_fit$loadings[,1:2]
t(loading)
```

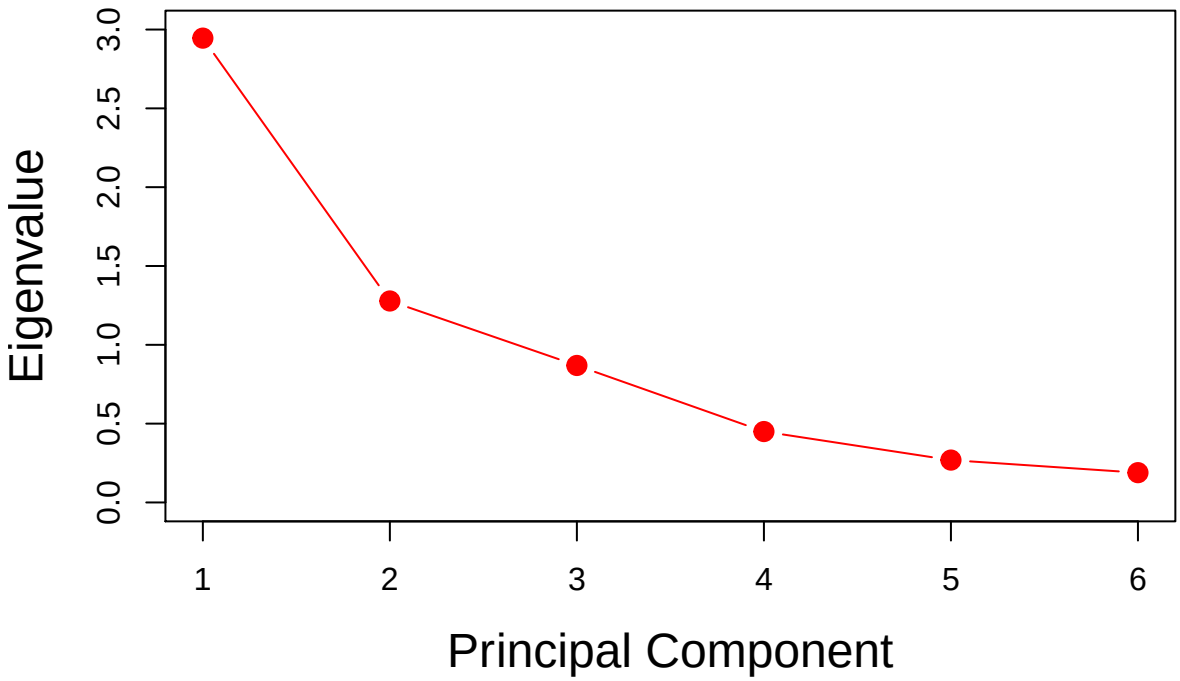
```
##             Length      Left     Right    Bottom        Top    Diagonal
## Factor1 -0.1670846 0.5527990 0.5603285 0.6323679 0.59914467 -0.99529478
## Factor2  0.4305069 0.7105278 0.6142669 0.1047625 0.05087066  0.06627958
```

The loading factors are scaled from 0 to 1 and represent the coefficients which tell us how strong the relationship is between the variable adn the factor. Loading close to -1 or 1 indicate that the factor storngly influences teh variable; while loading cloase to 0 indicate that the influence is weak.

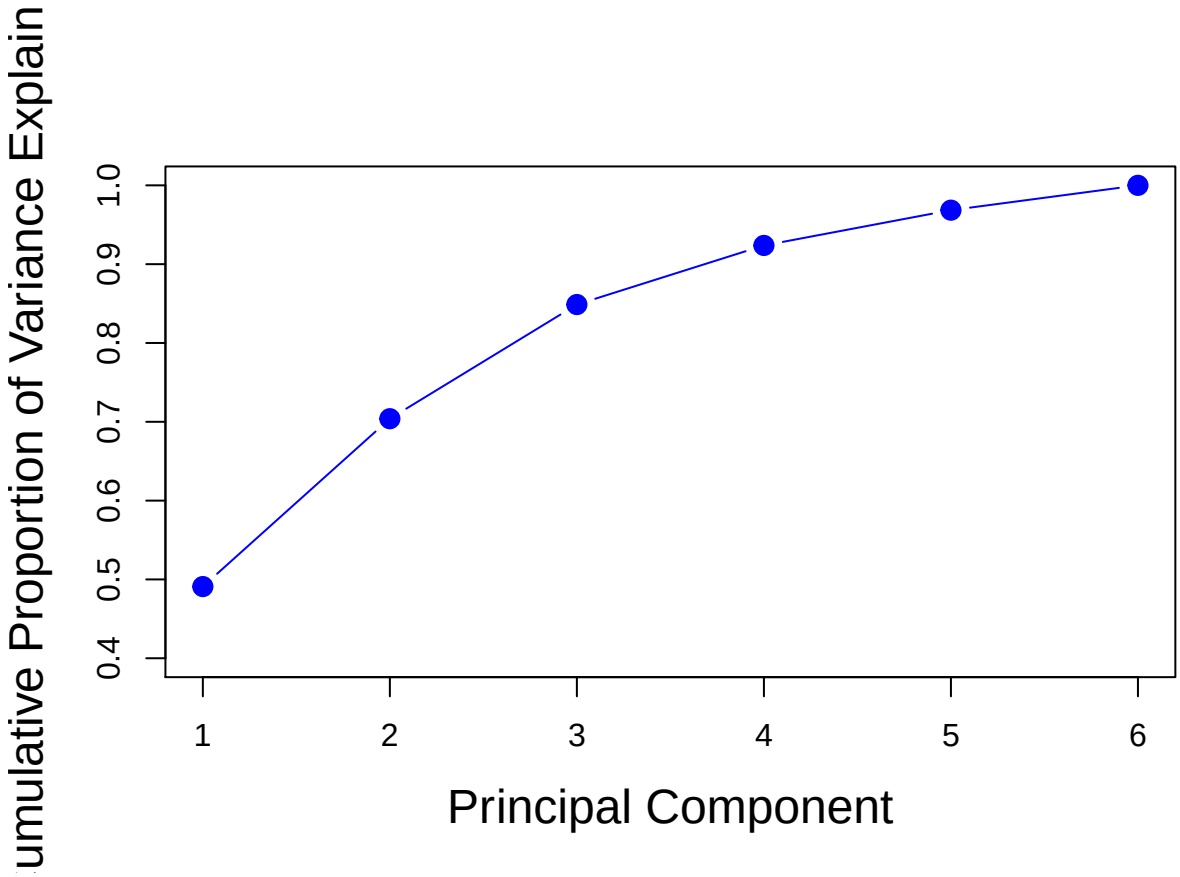
For **factor 1**, diagonal has a extremely high loading (-0.995), indicating factor 1 and diagonal are highly correlated. And diagonal alone may explain most of the variance of genuine/fake notes.

Variable left and variable right that are highly correlated in the correlation graph dominant the **factor 2**. Left and right may explain most of the variance of genuine/fake notes.

```
plot(eigen_val  , xlab=" Principal Component ", ylab=" Eigenvalue ",
     ylim=c(0,3), xaxt="n" ,type='b', col="red", cex=2,
pch=20, cex.lab=1.5)
axis(1, at=c(1,2,3,4,5,6),labels=c(1,2,3,4,5,6))
```

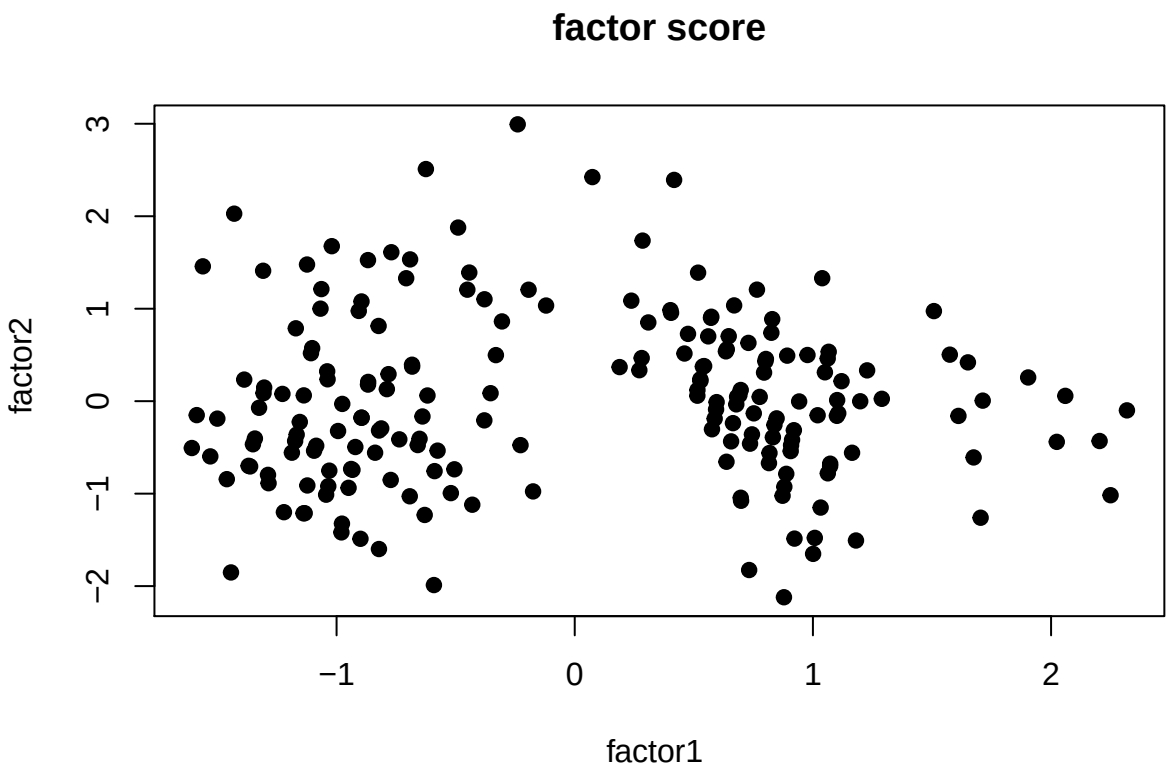



```
plot(cumsum(pve), xlab=" Principal Component ",
     ylab ="Cumulative Proportion of Variance Explained ",
     ylim=c(0.4,1) , xaxt="n",type='b', col="blue", cex=2, pch=20, cex.lab=1.5)
axis(1, at=c(1,2,3,4,5,6),labels=c(1,2,3,4,5,6))
```




```
fact1=fa_fit$scores[,1]
fact2=fa_fit$scores[,2]
```

```
# factor score scatter plot
plot(fact1, fact2, main="factor score",
   xlab="factor1 ", ylab="factor2 ", pch=19)
```



Using the scatter plot enables us to investigate the data structure and to detect outliers and sub-clusters of factor scores. As the scatter plot shown above, it seems that there are **two clusters**: one cluster is around (-1,0), and the other cluster is around(1,0). Those two clear cluster meet with out expectation since we are trying to classify a binary output: Genuine or counterfeit.

Notice: Because the scatter graph depends on the proportion of variance explained due to dimension reduction; there might still be missing information.

```
factor_table = data.frame(fa_fit$scores)
factor_table$Genuine = append(rep(1, 100),rep(0,100))
head(factor_table)
```

```
##        Factor1    Factor2 Genuine
## BN1 -0.2394320  2.9936843       1
## BN2 -1.0944004 -0.5357972       1
## BN3 -1.5006479 -0.1888711       1
## BN4 -1.3429929 -0.4038795       1
## BN5 -1.1875631 -0.5570299       1
## BN6 -0.6248881  2.5100863       1
```

```
set.seed(123)
fit_control = trainControl(method = 'cv', number = 10)
```

Use the caret package to re-run the lda/logistic regression on factor scores

- LDA

```
lda_fa_fit <- train(as.factor(Genuine) ~ Factor1+Factor2, data = factor_table,
                    method ="lda",trControl = fit_control)
lda_fa_fit$resample
```

```
##    Accuracy Kappa Resample
## 1      1.00   1.0   Fold01
## 2      1.00   1.0   Fold02
## 3      1.00   1.0   Fold03
## 4      1.00   1.0   Fold04
## 5      1.00   1.0   Fold05
## 6      1.00   1.0   Fold06
## 7      1.00   1.0   Fold07
## 8      0.95   0.9   Fold08
## 9      1.00   1.0   Fold09
## 10     1.00   1.0   Fold10
```

- Logistic Regression

```
lr_fa_fit <- train(as.factor(Genuine) ~ Factor1+Factor2, data = factor_table,
                   method ="glm",family = 'binomial',trControl = fit_control)
lr_fa_fit$resample
```

```
##    Accuracy Kappa Resample
## 1      1.00   1.0   Fold01
## 2      0.95   0.9   Fold02
## 3      0.95   0.9   Fold03
## 4      1.00   1.0   Fold04
## 5      1.00   1.0   Fold05
## 6      1.00   1.0   Fold06
## 7      1.00   1.0   Fold07
## 8      1.00   1.0   Fold08
## 9      1.00   1.0   Fold09
## 10     1.00   1.0   Fold10
```

Mannually run LDA/logisitic regression Using the save validation set index as calculated above, we again manually separate the validation sets and training sets for each folds. One advantage of manually splitting is that it keeps each fold have the same data and better see the effect of using the factor analysis.

```
library(caret)
validation1_fa = factor_table[validation$Fold01,]
training1_fa = factor_table[-validation$Fold01,]
validation2_fa = factor_table[validation$Fold02,]
training2_fa = factor_table[-validation$Fold02,]
validation3_fa = factor_table[validation$Fold03,]
training3_fa = factor_table[-validation$Fold03,]
validation4_fa = factor_table[validation$Fold04,]
training4_fa = factor_table[-validation$Fold04,]
validation5_fa = factor_table[validation$Fold05,]
training5_fa = factor_table[-validation$Fold05,]
```

```
validation6_fa = factor_table[validation$Fold06,]
training6_fa = factor_table[-validation$Fold06,]
validation7_fa = factor_table[validation$Fold07,]
training7_fa = factor_table[-validation$Fold07,]
validation8_fa = factor_table[validation$Fold08,]
training8_fa = factor_table[-validation$Fold08,]
validation9_fa = factor_table[validation$Fold09,]
training9_fa = factor_table[-validation$Fold09,]
validation10_fa = factor_table[validation$Fold10,]
training10_fa = factor_table[-validation$Fold10,]
training_fa_list = list(training1_fa,training2_fa,training3_fa,training4_fa,
                        training5_fa,training6_fa,training7_fa,training8_fa,
                        training9_fa,training10_fa)
validation_fa_list = list(validation1_fa,validation2_fa,validation3_fa,
                          validation4_fa,validation5_fa,validation6_fa,
                          validation7_fa,validation8_fa,validation9_fa,validation10_fa)

lr_fa_mean = NULL
lda_fa_mean = NULL

# fit the LDA/LR function
for (i in 1:10){
  fit.lr = glm(Genuine~Factor1+Factor2,data=training_fa_list[[i]],
               family=binomial,maxit = 100)
  prob = predict(fit.lr, validation_fa_list[[i]], type="response")
  pred = rep(0, dim(validation_fa_list[[i]])[1])
  pred[prob > .5] = 1
  lr_fa_mean = rbind(lr_fa_mean,
                     mean(pred == (validation_fa_list[[i]]$Genuine==1)))

  fit.lda = lda(Genuine ~Factor1+Factor2, data=training_fa_list[[i]])
  pred.lda = predict(fit.lda, newdata = validation_fa_list[[i]])
  lda_fa_mean = rbind(lda_fa_mean,
                      mean(pred.lda$class== validation_fa_list[[i]]$Genuine))
}
```

To better visualize this, I combine the results into this table.

```
result$LDA_Factor = lda_fa_mean
result$LR_Factor = lr_fa_mean
result
```

```
##       Name  LDA   LR LDA_Factor LR_Factor
## 1   Fold 1 1.00 0.90       1.00      0.95
## 2   Fold 2 1.00 0.95       1.00      1.00
## 3   Fold 3 1.00 1.00       1.00      1.00
## 4   Fold 4 1.00 1.00       1.00      1.00
## 5   Fold 5 1.00 1.00       1.00      1.00
## 6   Fold 6 0.95 0.95       0.95      0.95
## 7   Fold 7 1.00 1.00       1.00      1.00
## 8   Fold 8 1.00 1.00       1.00      1.00
## 9   Fold 9 1.00 1.00       1.00      1.00
## 10 Fold 10 1.00 1.00       1.00      1.00
```

Cross-validation

```
final_row = NULL
final_row = cbind(final_row,lda_fit$results$Accuracy)
final_row = cbind(final_row,lr_fit$results$Accuracy)
final_row = cbind(final_row,lda_fa_fit$results$Accuracy)
final_row = cbind(final_row,lr_fa_fit$results$Accuracy)
final_row
```

```
##       [,1] [,2]  [,3] [,4]
## [1,] 0.995 0.98 0.995 0.99
```

```
# result_table
# LR represents logistic regression
rbind(result,list("Cross Fold",0.995,0.98,.994,0.99))
```

```
##          Name   LDA   LR LDA_Factor LR_Factor
## 1      Fold 1 1.000 0.90      1.000      0.95
## 2      Fold 2 1.000 0.95      1.000      1.00
## 3      Fold 3 1.000 1.00      1.000      1.00
## 4      Fold 4 1.000 1.00      1.000      1.00
## 5      Fold 5 1.000 1.00      1.000      1.00
## 6      Fold 6 0.950 0.95      0.950      0.95
## 7      Fold 7 1.000 1.00      1.000      1.00
## 8      Fold 8 1.000 1.00      1.000      1.00
## 9      Fold 9 1.000 1.00      1.000      1.00
## 10    Fold 10 1.000 1.00      1.000      1.00
## 11 Cross Fold 0.995 0.98      0.994      0.99
```

As the final result table shown above,

1. Overall the **LDA has a better performance than the logistic regression**. No matter with or without factor analysis, the LDA accuracy rate among 10 folds is higher than LR accuracy rate. About 3/10 folds will have a mis-prediction in LR; while using LDA, only 1/10 will have a mis-prediction.
2. **The accuracy rate in Cross Folds is extremly high**. As the last row shows, every model (with or without factor analysis) has an accuracy rate of 1, indicating that there is a decent amount of differences in genuine notes and counterfeit notes.
3. **Factor analysis does not help improving our prediction rate**. Although the using factor analysis, the logistic regression accuracy is slight improved. For other accuracy, the factor analysis has no impact on the accuracy as the original predictions already has a almost perfect result.

Conclusion

In this project, I used the caret package to predict whether the swiss notes are geninue or counterfeit by LDA and Logistic regression. After implementing K-fold cross-validation and running the LDA and logistic regression on each fold, we shall see that there is no fold's accuracy rate is below 95% and most of the fold has an accuracy rate of 1. Moreover, **LDA has a much better performance than logistic regression** among those 10 folds. About 3/10 folds will have a mis-prediction in LR; while using LDA, only 1/10 will have a mis-prediction and the rest labels in the validation set are perfectly classified.

The over all **accuracy rates are extremly high** among each folds/cross-folds and each models. On the last row, using corss-folds, every models are able to achieve an accuracy rate higher than 98%, indicating

that using supervised learning, we can **almost perfectly predict whether a swiss note is genuine or counterfeit**.

According to the result_table, there is no (significant) difference between the accuracy rate from last two columns and result from the first two columns. This indifference implies that **using factor analysis does not help us to improve the model or explain the data variation**. One possible reasons is that the variables in original dataset are good predictors which could give a perfect prediction on the output, and factor analysis cannot improve the result anymore. Although the factor analysis does not help improving accuracy, it does tell us that the two factors from factor model are heavily based on diagonal, left/right. This result can inform bankers to pay more attention to the diagonal, left/right size of the counterfeit notes.

However, I do recognize there are a few inefficiencies in my project. For instance, I only utilize Q-Q plot to visualize normality assumption and **did not test the assumption formally**. As a result,even though the Q-Q plots of bottom and diagonal are tilted, it is not statistically sufficient to conclude anything. Also, one of the logistic regression assumptions is **no high intercorrelations (i.e. multicollinearity)** among the predictors (shown on the correlation plot, the left and right are highly correlated). But again since no formal test is conducted, we cannot formally state that the assumption is violated. These assumptions of the models, **especially for normality assumption, need to be further validate**. Also I did not have a **training set** in this project, and thus we could not really test how well our models can predict.