Final Project

Banknote Authentication

Logan Yoonhyuk Lee

Taeheon Park

Zhen Zhang

Yonghyeon Kweon

STAT 5330

Professor Xiwei Tang

Section 001

May 8, 2019

**Introduction**

According to United States Department of Treasury, $70 million fake banknotes are in circulation. Detecting fake banknotes is a really important part that ensures the stability of the financial system. Our group is interested in using one dataset about banknote authentication to see how some of the models might help make the accurate detection of fake banknotes.

In the banknote classification, template mating technique, which is using image matching with specific bill type, was widely used. Even though this method is simple and fast, the template should be modified as bill type is changed. Wavelet transform method could make up this weakness by analyzing the characteristic of the bill data and extract the features from the image.

**Data summarization**

The dataset is collected from the UCI machine learning repository. The dataset contains 1372 observations, and it is composed of 4 continuous independent variables and one binary discrete response variable. The table1.1 below describes each independent and dependent variables.

|  |  |  |
| --- | --- | --- |
| **Independent**  **variable** | **Variance** of wavelet Transformed image | Continuous |
| **Skewness** of Wavelet Transformed image | Continuous |
| **Kurtosis** of Wavelet Transformed image | Continuous |
| **Entropy** of Wavelet Transformed image | Continuous |
| **Dependent Variable** | **Class** | Discrete (1 = fake, 0 = real ) |

Table 1.1 Independent and dependent variables

The response variable, which is Class, represents the authentication of the banknote. In other words, it is 1 when the banknote is fraud and 0 when the banknote is authentic.

Each predictor, Kurtosis, Skewness, Variance, and Entropy contributes to determining the authentication of the banknote:

* If Kurtosis is higher the image looks like a kind of clean and lighter, but lower kurtosis has some noise in the image.
* For the skewness, the image is darker and glossier if positively skewed, and lighter and matter if negatively skewed.
* The high variance gives you less noise image, but low variance gives you darker and noisy image.
* Entropy represents the degree of randomness. So, high entropy means more disordered, and low entropy means highly ordered.

If we apply this concept to the picture or the image, high entropy makes the picture and the image more exact with a variety of gray colors and low entropy makes a picture have just the same gray color.

**Data illustration & Exploratory analysis**

In the dataset, among the 1372 observations, around 55% are authentic cases, while the fake cases count the rest 45% of all the observations. This means that the dataset is pretty balanced.

To train the model, the dataset is randomly split into two parts: 75% of the observations are assigned into the training dataset, while the rest 25% of the observations are assigned into the testing dataset. Initially, we used all the variables described above (Kurtosis, Skewness, Variance, and Entropy). By applying LDA, QDA, Logistic Regression, SVM, Random Forest, and Adaboosting on the training dataset, we get the accuracy, sensitivity, and specificity.

In our case, sensitivity stands for the probability of accurately detecting a fake banknote; while specificity stands for accurately classify authentic banknotes to be real. The accuracy is the percentage of banknote cases that are classified correctly. Table 2.1 below represents the accuracy, the sensitivity, and the specificity values which have obtained from each different model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | LDA | QDA | Logistic Regression | SVM | Random  Forest | Adaboosting |
| Accuracy | 0.9650146 | 0.9766764 | 0.9854227 | 1 | 0.9941691 | 1 |
| Sensitivity | 1 | 1 | 0.9873418 | 1 | 1 | 1 |
| Specificity | 0.9351351 | 0.9567568 | 0.9837838 | 1 | 0.9891892 | 1 |

Table 2.1 Accuracy, Sensitivity, and Specificity by each model

According to the initial result generated using all four variables, we found that the accuracy, sensitivity, and specificity are nearly all in a really high level (above 97%), and the accuracy of SVM and Adaboosting even reach 100%. This phenomenon is pretty rare in a real-world situation and we began to explore the reason behind it by comparing different variables and checking the dataset.

**Comparing four variables**

After comparing the variables, we found one of the variables called ‘variance’ is much more important than the rest of the three variables according to the variable importance graph below (figure 1.1).

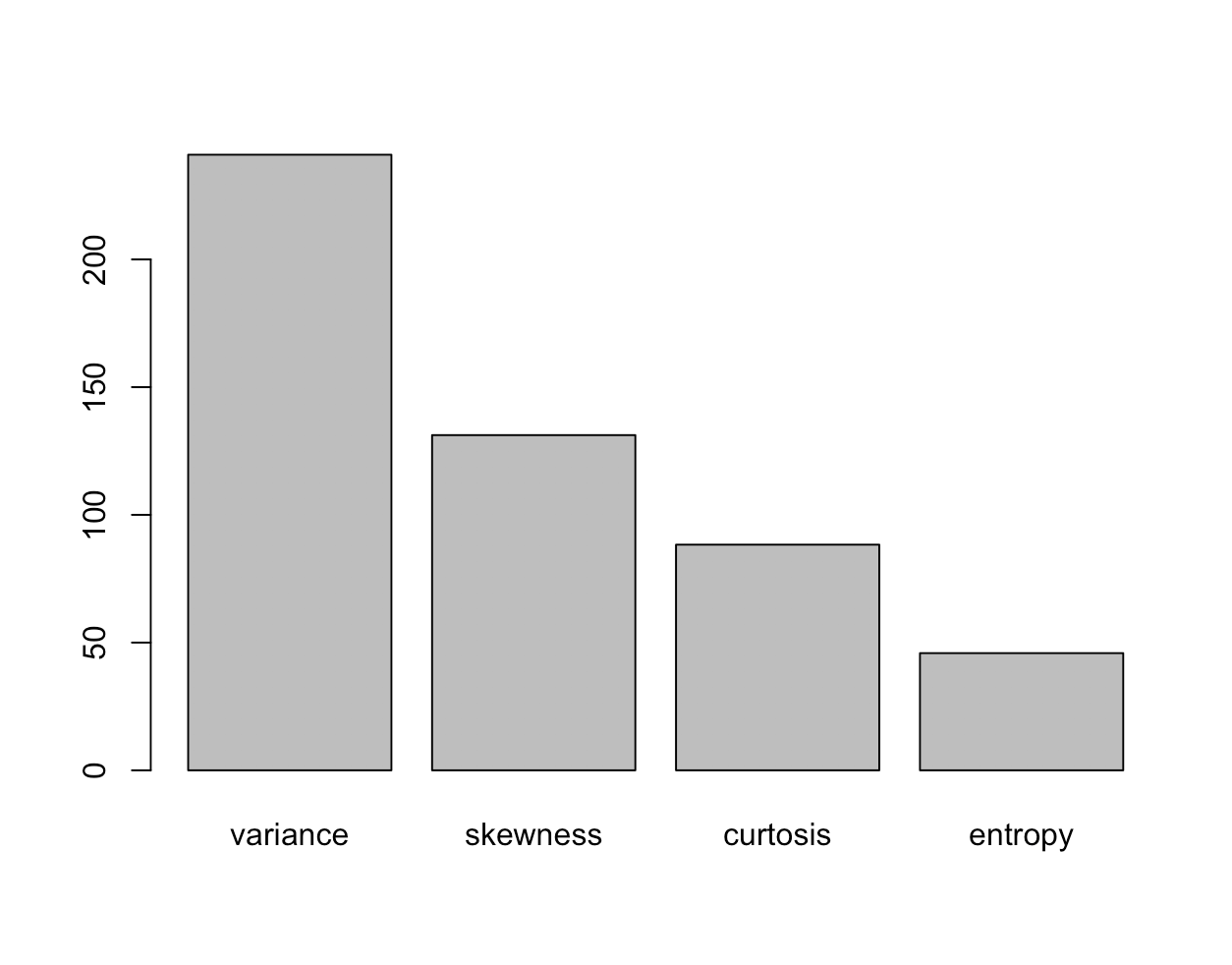


Figure 1.1 Variable Importance Barplot

Then, the value of four variables of fake and authentic banknote cases are compared in the boxplots. According to figure 2.1, the variance of fake and authentic banknotes differs way too much compared to the other three variables. The reason behind the high accuracy is because of the ‘variance’ variable, which is super helpful and significant when differentiating banknotes from authentic to fake. Compared to figure 2.1, none of the other three variables differs that significantly for authentic and fake banknote cases.

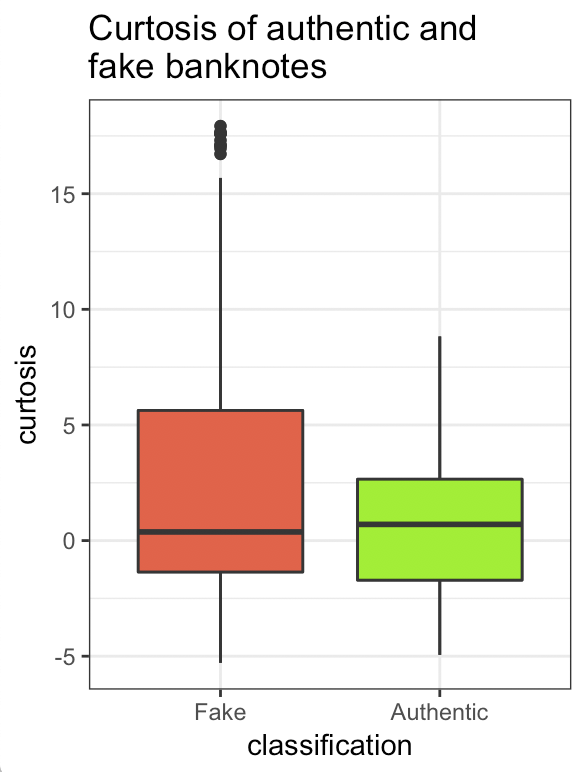
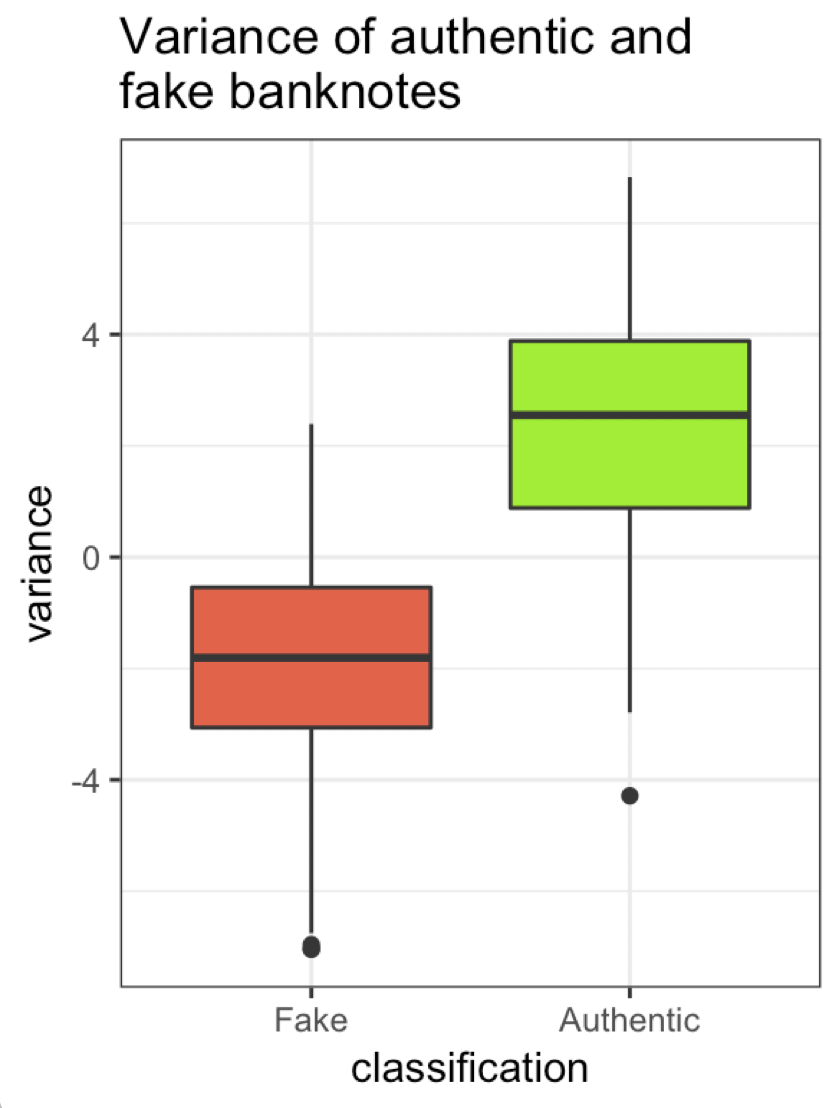


Figure 2.1 Figure 2.2

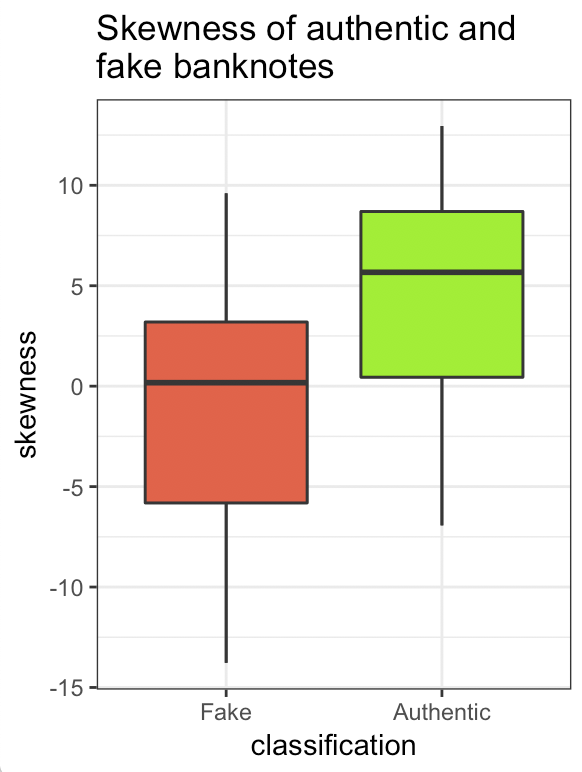
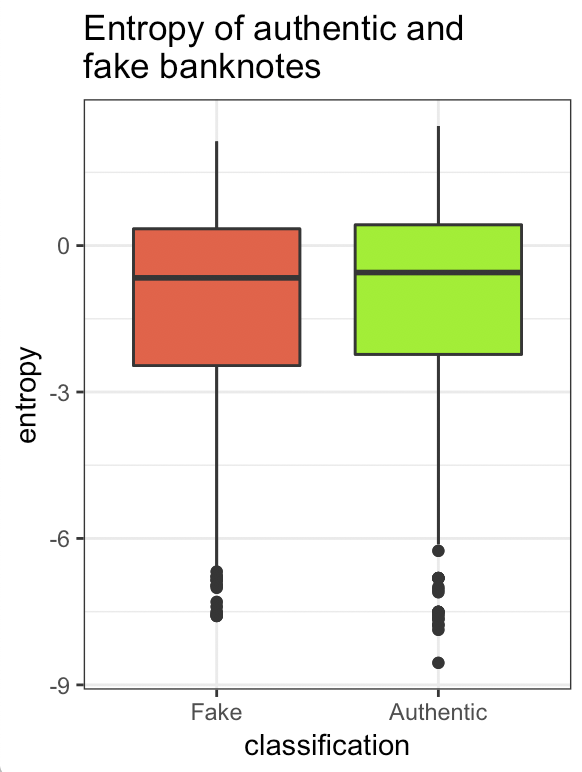


Figure 2.3 Figure 2.4

In a realistic world, this powerful predictor and highly-performed model would be welcome; however, for the project, we decided to remove the variance from the dataset and build our model based on the rest of three variables: kurtosis, entropy, and skewness. We are trying to mimic the situation when we don’t have variance as one of the predictors when making the prediction.

**Data analysis**

In order to analyze the given dataset, six models, LDA, QDA, Logistic Regression, SVM, Random Forest, and Adaboosting have been trained and performed on the dataset. Along with these classifiers, three measurements, accuracy, sensitivity, and specificity have been used to evaluate each model. For this project, sensitivity measures how the model correctly classifies the fake banknote as fraud and specificity measures how the model correctly classifies the real banknote as authentic. R has been used as a tool for analyzing the dataset and the parameters in some of the models were adjusted according to the given dataset. To be specific, the predicted value was set as 0.4 as the proportion of the class for logistic regression. Therefore, the predicted values from the glm function differentiate fraud banknote when the probability value is greater than 0.4. For SVM, kernel “radial” and cost =1 as a set of parameter. Random Forest used cross-validation with infold = 5 and parameters, ntree, mtry, and nodesize were set as 500, 2 and 1 respectively. Lastly, adaboosting is also tuned for the most desirable result.

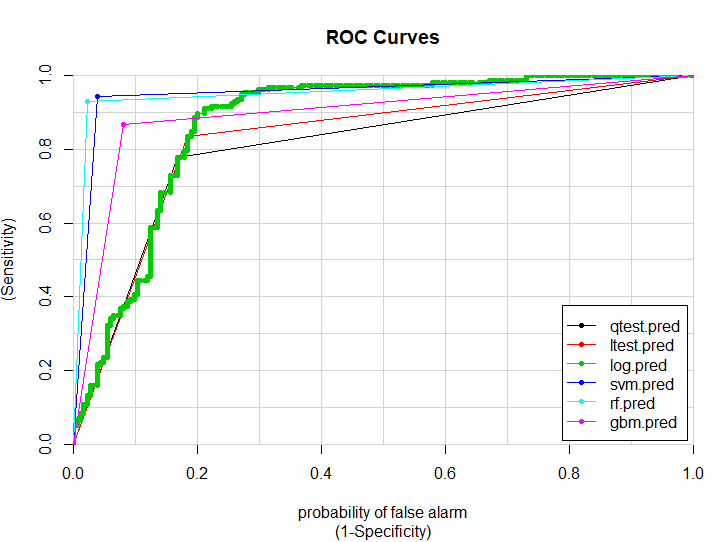
After the ‘variance’ predictor is removed, each model is fitted and predicted again. The table below is showing each value of measurement in six models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **LDA** | **QDA** | **Logistic Regression** | **SVM** | **Random**  **Forest** | **Adaboosting** |
| **Accuracy** | 0.8250729 | 0.8075802 | 0.8279883 | 0.9533528 | 0.9679300 | 0.8950437 |
| **Sensitivity** | 0.835443 | 0.778481 | 0.8544304 | 0.943038 | 0.9493671 | 0.8670886 |
| **Specificity** | 0.8162162 | 0.8324324 | 0.8054054 | 0.9621622 | 0.9837838 | 0.9189189 |

Table 3.1 Accuracy, Sensitivity, and Specificity by each model after removing ‘variance’

The results come out to be different from the previous. This time LDA, QDA, and logistic regression have the accuracy, sensitivity, and specificity around 0.8 which are much lower than the models with the ‘variance’ predictor. Adaboosting have results around 0.9 which are not bad, but SVM and Random Forest give the best results. Both SVM and Random Forest have accuracy, sensitivity, and specificity between 0.94 and 0.99. Overall, Random Forest has little bit higher results than SVM. Since the two results are very close to each other, it is difficult to decide the best one. Thus, additional investigations are conducted to determine which one of SVM and Random Forest is the most suitable classifier for the banknote dataset.

Accuracy of the model is important, but it considers only one threshold setting. In order to measure how the model performs at different threshold settings, the ROC curve was used to compare the ability of each model.



AUC represents the area under the ROC curve, and the table below is showing the AUC value in each model.

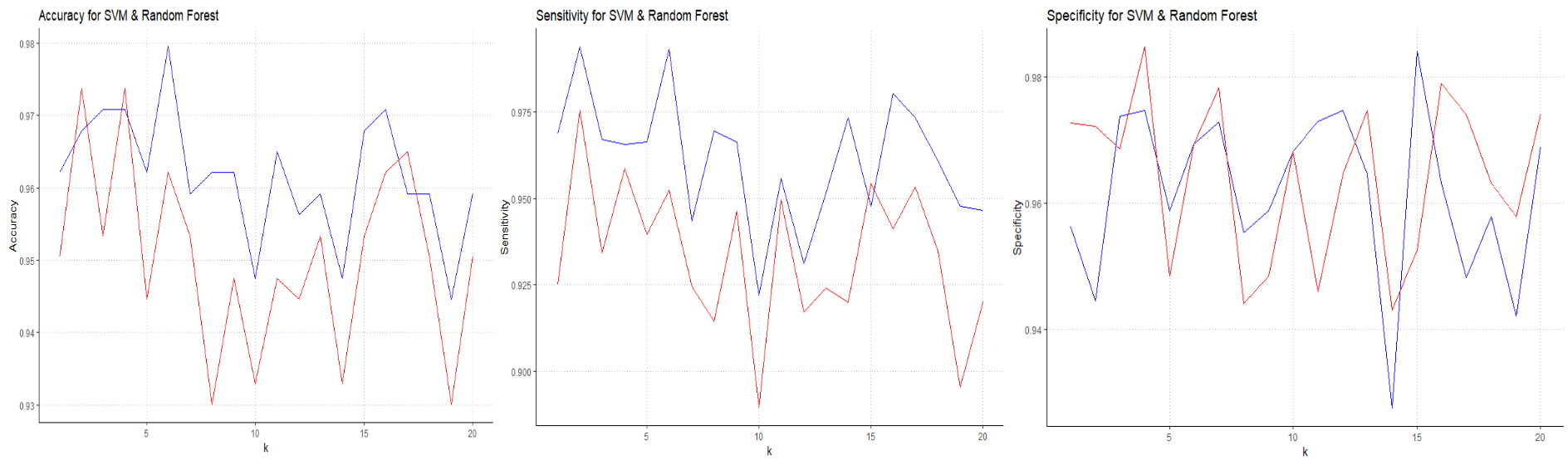
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **LDA** | **QDA** | **Logistic Regression** | **SVM** | **Random Forest** | **Adaboosting** |
| **AUC** | 0.8054567 | 0.8258296 | 0.8732125 | 0.9526001 | 0.9543791 | 0.8930038 |

Table 4.1 AUC of the classifiers

As shown in the plot, it is clear that the ROC curve of SVM and Random Forest (blue and light blue lines) have outstanding performance compared to the other models. Also, the AUC table is showing that SVM and Random Forest have the area around 0.95 indicating the best classification performance among all the models.

**Repetition**

In order to compare the model stability between SVM and Random Forest, the process of train and test dataset split has been repeated 20 times. The plot below is showing the result each accuracy, sensitivity, and specificity for each trial.



The blue line represents SVM and the red line represents Random Forest. The accuracy plot shows that Random Forest has higher values than SVM in the beginning, but it drastically decreases as the trial number goes up which makes SVM has higher accuracy overall. The sensitivity plot shows SVM has higher values than Random Forest for the most of the time. The specificity plot shows that SVM and Random Forest fluctuate in similar patterns, so it is hard to decide which of these two has better specificity.

**Conclusion and discussion**

In conclusion, we’ve trained six classifiers, LDA, QDA, Logistic regression, Random Forest, SVM, and Adaboosting to analyze the given dataset. In order to measure the performance ability of each model, accuracy, sensitivity, and specificity were calculated as a criterion. Even though all models performed well on the given data, SVM and Random Forest showed the highest ability.

In terms of banknote authentication analysis, sensitivity is considered a more important criterion over specificity. In other words, true positive, or false detection is more important for the classification of the given data. After the classification process was repeated to see the stability of each model, SVM showed slightly higher sensitivity. As a result, SVM has been selected as a final model for the given dataset.

In order to compare the performance between SVM and Random Forest, a few things have been considered. First of all, due to kernel tricks in SVM, an appropriate feature space could be created. In other words, SVM performs considering the distribution of the data points. Moreover, SVM is more suitable for two-class problems whereas Random Forest performs better for multi-class problems. Also, in most cases, SVM is used for a real-valued feature, which is no categorical ones. For these reasons, it can be assumed that SVM will be performed better on the given dataset.

However, there exist some drawbacks to be considered as well. Above all, choosing an appropriate kernel is tricky, so several trials to find the most “adequate” one might be needed. Furthermore, since it doesn’t provide probability estimates, if the data has an imbalanced proportion of the class, it would cause a problem. Also, SVM takes the longest training time among six models and obviously, SVM doesn’t suit for large dataset. Despite these drawbacks, SVM has been selected for the given dataset since there are only 1372 observations and the balanced proportion of the class.

**Reference**

*UCI Machine Learning Repository: Banknote Authentication Data Set*, <https://archive.ics.uci.edu/ml/datasets/banknote+authentication>.

*1.3.5.11. Measures of Skewness and Kurtosis*, www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm.

*Board of Governors of the Federal Reserve System*, www.federalreserve.gov/faqs/currency\_12597.htm.

Admin. “A Guide for Using the Wavelet Transform in Machine Learning.” *Ahmet Taspinar*, 5 Apr. 2019, ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/.

**Code**

setwd("C:/Users/Taeheon Park/Desktop/STAT1601")

install.packages("ISLR")

library(ISLR)

install.packages("tidyverse")

library(tidyverse)

banknote<-read.table(file="data\_banknote\_authentication.txt",sep=",")

View(banknote)

names(banknote) <- c("variance of Wavelet Transformed image",

"skewness of Wavelet Transformed image",

"curtosis of Wavelet Transformed image",

"entropy of image",

"class")

banknote <- banknote %>%

select(-("V1"))

attach(banknote)

sampsize = floor(0.75\*nrow(banknote))

set.seed(123)

train\_ind = sample(seq\_len(nrow(banknote)),size = sampsize)

train =banknote[train\_ind,]

test=banknote[-train\_ind,]

##LDA & QDA

dig.lda=lda(V5~V2+V3+V4, data=train)

ltest.pred=predict(dig.lda, test)$class

ltable=table(ltest.pred, V5=test$V5)

dig.qda = qda(V5~V2+V3+V4, data=train)

qtest.pred=predict(dig.qda, test)$class

qtable=table(qtest.pred, V5=test$V5)

accuracy1<-sum(ltable[c(1,4)])/sum(ltable[1:4])

sensitivity1<-sum(ltable[c(4)])/sum(ltable[3:4])

specificity1<-sum(ltable[c(1)])/sum(ltable[1:2])

print(c(accuracy1,sensitivity1,specificity1))

accuracy2<-sum(qtable[c(1,4)])/sum(qtable[1:4])

sensitivity2<-sum(qtable[c(4)])/sum(qtable[3:4])

specificity2<-sum(qtable[c(1)])/sum(qtable[1:2])

print(c(accuracy2,sensitivity2,specificity2))

##Logistics

glm.fit<- glm(V5~V2+V3+V4, family='binomial',train)

log.pred <- predict(glm.fit, test)

glm.pred1 <- ifelse(log.pred > 0.4, "1","0") ## over 0.5 makes specificity increasing, so for the spam problem it's good to use over 0.5

lotable <- table(glm.pred1,test$V5)

accuracy4<-sum(lotable[c(1,4)])/sum(lotable[1:4])

sensitivity4<-sum(lotable[c(4)])/sum(lotable[3:4])

specificity4<-sum(lotable[c(1)])/sum(lotable[1:2])

print(c(accuracy4,sensitivity4,specificity4))

##SVM

svm.fit <- svm(V5 ~ ., data = train, type='C-classification', kernel='radial',scale=FALSE, cost = 1)

svm.pred=predict(svm.fit, test)

stable <- table(svm.pred, V5=test$V5)

## for kernel use one more kernel -> choose

accuracy100<-sum(stable[c(1,4)])/sum(stable[1:4])

sensitivity100<-sum(stable[c(4)])/sum(stable[3:4])

specificity100<-sum(stable[c(1)])/sum(stable[1:2])

print(c(accuracy100,sensitivity100,specificity100))

##Random Forest

control <- trainControl(method="cv", number=5)

rf.fit = randomForest(as.factor(V5)~.,data=train, ntree = 100, mtry = 2,importance=TRUE, trControl = control)

rf.pred <- predict(rf.fit, test)

tables17 <- table(rf.pred,test$V5)

accuracy17<-sum(tables17[c(1,4)])/sum(tables17[1:4])

sensitivity17<-sum(tables17[c(4)])/sum(tables17[3:4])

specificity17<-sum(tables17[c(1)])/sum(tables17[1:2])

print(c(accuracy17,sensitivity17,specificity17))

###gbm

library(gbm)

gbmgrid <- expand.grid(

n.trees = c(100,200,500,1000,2000),

shrinkage = seq(0.01,0.1,0.01),

interaction.depth = 1, n.minobsinnode = 10)

control <- trainControl(method="cv", number=5)

gbmfit <- train(as.factor(V5) ~ ., data = train, distribution="adaboost", method = "gbm", trControl = control, tuneGrid = gbmgrid, verbose=FALSE)

gbm.pred = predict(gbmfit, test)

tablee <- table(gbmpred,test$V5)

accuracy20<-sum(tablee[c(1,4)])/sum(tablee[1:4])

sensitivity20<-sum(tablee[c(4)])/sum(tablee[3:4])

specificity20<-sum(tablee[c(1)])/sum(tablee[1:2])

print(c(accuracy20,sensitivity20,specificity20))

#######ROC

########Plots

install.packages("caTools")

library(caTools)

colAUC(cbind(qtest.pred, ltest.pred, log.pred, svm.pred, rf.pred, gbm.pred), test$V5, plotROC = T)

######## Decision Tree

install.packages("rpart.plot")

library(rpart.plot)

cforest(V5 ~ ., data=train, controls=cforest\_control(mtry=2, mincriterion=0))

tree1 <- rpart(formula = V5~.,

data = train,

method = 'class')

invisible(rpart.plot(tree1))

##Random Prediction

y.test.prediction=rbinom(343,1,0.5)

tablesd <- table(y.test.prediction, test$V5)

accuracy5<-sum(tablesd[c(1,4)])/sum(tablesd[1:4])

sensitivity5<-sum(tablesd[c(4)])/sum(tablesd[3:4])

specificity5<-sum(tablesd[c(1)])/sum(tablesd[1:2])

print(c(accuracy5,sensitivity5,specificity5))

##ROC

y.test.prediction0=rbinom(343,1,0.0)

tables0 <- table(y.test.prediction0, test$V5)

accuracy6<-sum(tables0[1])/sum(tables0[1:2])

sensitivity6<-0

specificity6<-1

print(c(accuracy6,sensitivity6,specificity6))

y.test.prediction1=rbinom(343,1,0.1)

tables1 <- table(y.test.prediction1, test$V5)

accuracy7<-sum(tables1[c(1,4)])/sum(tables1[1:4])

sensitivity7<-sum(tables1[c(4)])/sum(tables1[3:4])

specificity7<-sum(tables1[c(1)])/sum(tables1[1:2])

print(c(accuracy7,sensitivity7,specificity7))

y.test.prediction2=rbinom(343,1,0.2)

tables2 <- table(y.test.prediction2, test$V5)

accuracy8<-sum(tables2[c(1,4)])/sum(tables2[1:4])

sensitivity8<-sum(tables2[c(4)])/sum(tables2[3:4])

specificity8<-sum(tables2[c(1)])/sum(tables2[1:2])

print(c(accuracy8,sensitivity8,specificity8))

y.test.prediction3=rbinom(343,1,0.3)

tables3 <- table(y.test.prediction3, test$V5)

accuracy9<-sum(tables3[c(1,4)])/sum(tables3[1:4])

sensitivity9<-sum(tables3[c(4)])/sum(tables3[3:4])

specificity9<-sum(tables3[c(1)])/sum(tables3[1:2])

print(c(accuracy9,sensitivity9,specificity9))

##sensi = spam, speci = non spam

y.test.prediction4=rbinom(343,1,0.4)

tables4 <- table(y.test.prediction4, test$V5)

accuracy10<-sum(tables4[c(1,4)])/sum(tables4[1:4])

sensitivity10<-sum(tables4[c(4)])/sum(tables4[3:4])

specificity10<-sum(tables4[c(1)])/sum(tables4[1:2])

print(c(accuracy10,sensitivity10,specificity10))

y.test.prediction5=rbinom(343,1,0.5)

tables5 <- table(y.test.prediction5, test$V5)

accuracy11<-sum(tables5[c(1,4)])/sum(tables5[1:4])

sensitivity11<-sum(tables5[c(4)])/sum(tables5[3:4])

specificity11<-sum(tables5[c(1)])/sum(tables5[1:2])

print(c(accuracy11,sensitivity11,specificity11))

y.test.prediction6=rbinom(343,1,0.6)

tables6 <- table(y.test.prediction6, test$V5)

accuracy12<-sum(tables6[c(1,4)])/sum(tables6[1:4])

sensitivity12<-sum(tables6[c(4)])/sum(tables6[3:4])

specificity12<-sum(tables6[c(1)])/sum(tables6[1:2])

print(c(accuracy12,sensitivity12,specificity12))

y.test.prediction7=rbinom(343,1,0.7)

tables7 <- table(y.test.prediction7, test$V5)

accuracy13<-sum(tables7[c(1,4)])/sum(tables7[1:4])

sensitivity13<-sum(tables7[c(4)])/sum(tables7[3:4])

specificity13<-sum(tables7[c(1)])/sum(tables7[1:2])

print(c(accuracy13,sensitivity13,specificity13))

y.test.prediction8=rbinom(343,1,0.8)

tables8 <- table(y.test.prediction8, test$V5)

accuracy14<-sum(tables8[c(1,4)])/sum(tables8[1:4])

sensitivity14<-sum(tables8[c(4)])/sum(tables8[3:4])

specificity14<-sum(tables8[c(1)])/sum(tables8[1:2])

print(c(accuracy14,sensitivity14,specificity14))

y.test.prediction9=rbinom(343,1,0.9)

tables9 <- table(y.test.prediction9, test$V5)

accuracy15<-sum(tables9[c(1,4)])/sum(tables9[1:4])

sensitivity15<-sum(tables9[c(4)])/sum(tables9[3:4])

specificity15<-sum(tables9[c(1)])/sum(tables9[1:2])

print(c(accuracy15,sensitivity15,specificity15))

y.test.prediction10=rbinom(343,1,1.0)

tables10 <- table(y.test.prediction10, test$V5)

accuracy16<-sum(tables10[2])/sum(tables10[1:2])

sensitivity16<-sum(tables10[2])/sum(tables10[2])

specificity16<-0

print(c(accuracy16,sensitivity16,specificity16))

sense = c(sensitivity6, sensitivity7, sensitivity8, sensitivity9, sensitivity10, sensitivity11, sensitivity12, sensitivity13, sensitivity14, sensitivity15, sensitivity16)

specs = c(specificity6, specificity7, specificity8, specificity9, specificity10, specificity11, specificity12, specificity13, specificity14, specificity15, specificity16)

accur = c(accuracy6, accuracy7, accuracy8, accuracy9, accuracy10, accuracy11, accuracy12, accuracy13, accuracy14, accuracy15, accuracy16)

plot(x=(1-specs),y=sense, type='l')

## repeating 20 times

svmacc = vector()

svmsensi = vector()

svmspec = vector()

rfacc = vector()

rfsensi = vector()

rfspec = vector()

n=nrow(test)

for (i in seq(1:20)){

sampsize = floor(0.75\*nrow(banknote))

set.seed(i)

train\_ind = sample(seq\_len(nrow(banknote)),size = sampsize)

train =banknote[train\_ind,]

test=banknote[-train\_ind,]

svmfit.linear = svm(class ~ ., data = train, type='C-classification', kernel = "radial", cost = 1, scale = FALSE)

svm.pred.linear <- predict(svmfit.linear, test)

svm.table.linear <- table(svm.pred.linear, test$class)

svm.accuracy.linear <- (svm.table.linear["0","0"] + svm.table.linear["1","1"])/n

svm.sensitivity.linear <- svm.table.linear["1","1"]/(svm.table.linear["0","1"] + svm.table.linear["1","1"])

svm.specificity.linear <- svm.table.linear["0","0"]/(svm.table.linear["1","0"] + svm.table.linear["0","0"])

svmacc <- c(svmacc, svm.accuracy.linear)

svmsensi <- c(svmsensi, svm.sensitivity.linear)

svmspec <- c(svmspec, svm.specificity.linear)

control <- trainControl(method="cv", number=5)

rf.fit = randomForest(as.factor(class)~.,data=train, ntree = 500, mtry = 2,nodedize = 1,importance=TRUE, trControl = control)

rf.predictions <- predict(rf.fit, test)

rf.table <- table(rf.predictions,test$class)

rf.accuracy <- (rf.table["0","0"] + rf.table["1","1"])/n

rf.sensitivity <- rf.table["1","1"]/(rf.table["0","1"] + rf.table["1","1"])

rf.specificity <- rf.table["0","0"]/(rf.table["1","0"] + rf.table["0","0"])

rfacc <- c(rfacc, rf.accuracy)

rfsensi <- c(rfsensi, rf.sensitivity)

rfspec <- c(rfspec, rf.specificity)

}

svmacc

svmsensi

svmspec

rfacc

rfsensi

rfspec

accdf <- data.frame(x=seq(1:20),svmacc,rfacc)

sensidf <- data.frame(x=seq(1:20),svmsensi,rfsensi)

specdf <- data.frame(x=seq(1:20),svmspec,rfspec)

ggplot(accdf) + geom\_line(aes(x=x,y=svmacc),col='blue') + geom\_line(aes(x=x,y=rfacc),col='cadetblue1') +

theme\_bw() +

theme(panel.grid.major=element\_line(linetype="dotted",color='#b3b3b3')) +

theme(panel.grid.minor=element\_blank()) +

theme(axis.line=element\_line(colour = "black"),

panel.border=element\_blank(),

panel.background= element\_blank()) +

ggtitle('Accuracy for SVM & Random Forest')+

xlab('k') + ylab('Accuracy')

ggplot(sensidf) + geom\_line(aes(x=x,y=svmsensi),col='Blue') +

geom\_line(aes(x=x,y=rfsensi),col='red') +

theme\_bw() +

theme(panel.grid.major=element\_line(linetype="dotted",color='#b3b3b3')) +

theme(panel.grid.minor=element\_blank()) +

theme(axis.line=element\_line(colour = "black"),

panel.border=element\_blank(),

panel.background= element\_blank()) +

ggtitle('Sensitivity for SVM & Random Forest')+

xlab('k') + ylab('Sensitivity')

ggplot(specdf) + geom\_line(aes(x=x,y=svmspec),col='Blue') +

geom\_line(aes(x=x,y=rfspec),col='red') +

theme\_bw() +

theme(panel.grid.major=element\_line(linetype="dotted",color='#b3b3b3')) +

theme(panel.grid.minor=element\_blank()) +

theme(axis.line=element\_line(colour = "black"),

panel.border=element\_blank(),

panel.background= element\_blank()) +

ggtitle('Specificity for SVM & Random Forest')+

xlab('k') + ylab('Specificity')