Shulei Yang (sy8uu)
Machine Learning – Mini Project

1. **Exploratory Data Analysis**
   a. For the response variable, I calculated the percentage of spam mails. There are 39.74% of spam emails in the training dataset, and there are 38.74% of spam emails in the testing dataset.
   b. For the predictors, I tried to find out how many of them are continuous variables and how many of them are categorical variables. I think that if the values of a predictor are integers, it should be a categorical variable. And if the values of a predictor are 'doubles', it should be a continuous variable. Then I come to the conclusion that: There are 55 continuous variables and 2 categorical variables (V56, V57).

```
#=========================== Exploratory Data Analysis ================================#
print('Exploratory Data Analysis: ')
print(paste(' -The proportion of spam emails in the training dataset is ', percent(table(train$V58)['1']/nrow(train))))
print(paste(' -The proportion of spam emails in the testing dataset is ', percent(table(test$V58)['1']/nrow(test))))

cate_count <- 0
cate<-0
for(var in colnames(train[ ,-58])){
  if(typeof(train[ , var]) == 'integer'){
    cate_count = cate_count + 1
    cate<-c(cate, var)
  }
}
print(paste(' -There are ',ncol(train)-1-cate_count, ' continuous variables in the dataset'))
print(paste(' -There are ', cate_count, ' categorical variables in the dataset.'))
cate
```

2. **Modeling and Data Analysis**
   a. **LDA & QDA (V55-V57)**

	Accuracy	Sensitivity	Specificity
LDA	66.93%	19.33%	97.02%
QDA	68.16%	24.03%	96.07%

```
#================================ a) LDA & QDA V55-V57 ====================================#
a.lda <- lda(train[ ,55:57], train[ ,58])
a.lda.pred <- predict(a.lda, test[ ,55:57])$class
a.lda.result = table(a.lda.pred, test$V58)
#a.lda.pred   0   1
#         0 913 480
#         1  28 115
print(paste(' -Accuracy of the model is ', percent((a.lda.result[1,1] + a.lda.result[2,2])/sum(a.lda.result))))
print(paste(' -Sensitivity of the model is ', percent((a.lda.result[2,2])/sum(a.lda.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((a.lda.result[1,1])/sum(a.lda.result[ ,1]))))

a.qda <- qda(train[ ,55:57], train[ ,58])
a.qda.pred <- predict(a.qda, test[ ,55:57])$class
a.qda.result = table(a.qda.pred, test$V58)
#a.qda.pred   0   1
#         0 904 452
#         1  37 143
print(paste(' -Accuracy of the model is ', percent((a.qda.result[1,1] + a.qda.result[2,2])/sum(a.qda.result))))
print(paste(' -Sensitivity of the model is ', percent((a.qda.result[2,2])/sum(a.qda.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((a.qda.result[1,1])/sum(a.qda.result[ ,1]))))
```

b. LDA & QDA (V1-V57)

	Accuracy	Sensitivity	Specificity
LDA	87.76%	77.31%	94.37%
QDA	82.55%	94.45%	75.03%

```
#================================ b) LDA & QDA V1-V57 ================================#
b.lda <- lda(train[ ,1:57], train[ ,58])
b.lda.pred <- predict(b.lda, test[ ,1:57])$class
b.lda.result = table(b.lda.pred, test$V58)
#b.lda.pred   0   1
#          0 888 135
#          1  53 460
print(paste(' -Accuracy of the model is ', percent((b.lda.result[1,1] + b.lda.result[2,2])/sum(b.lda.result))))
print(paste(' -Sensitivity of the model is ', percent((b.lda.result[2,2])/sum(b.lda.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((b.lda.result[1,1])/sum(b.lda.result[ ,1]))))

b.qda <- qda(train[ ,1:57], train[ ,58])
b.qda.pred <- predict(b.qda, test[ ,1:57])$class
b.qda.result = table(b.qda.pred, test$V58)
#b.qda.pred   0   1
#          0 706  33
#          1 235 562
print(paste(' -Accuracy of the model is ', percent((b.qda.result[1,1] + b.qda.result[2,2])/sum(b.qda.result))))
print(paste(' -Sensitivity of the model is ', percent((b.qda.result[2,2])/sum(b.qda.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((b.qda.result[1,1])/sum(b.qda.result[ ,1]))))
```

From these two result tables we can see that when we increase our variables from V55-V57 to V1-V57, the Accuracy and Sensitivity all increase. Especially, the sensitivity increases a lot. But we also observe a lightly decrease in Specificity.

c. Logistic Regression Model and Linear SVM (V1-V57)

	Accuracy	Sensitivity	Specificity
Logistic	92.25%	85.38%	96.6%
SVM	91.15%	92.77%	90.12%

```
#============================ c) Logistic Regression & Linear SVM V1-V57 ===================================#
c.lg <- glm(V58 ~ ., data = train, family = binomial)
round(summary(c.lg)$coef, dig=3)
c.lg.pred <- predict(c.lg, test[ ,1:57]) > 0.5
c.lg.result <- table(c.lg.pred, test[ ,58])
#c.lg.pred   0   1
#    FALSE 909  87
#    TRUE   32 508
print('Logistic Regression results for V1-V57: ')
print(paste(' -Accuracy of the model is ', percent((c.lg.result[1,1] + c.lg.result[2,2])/sum(c.lg.result))))
print(paste(' -Sensitivity of the model is ', percent((c.lg.result[2,2])/sum(c.lg.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((c.lg.result[1,1])/sum(c.lg.result[ ,1]))))

c.svm <- svm(V58 ~ ., data = train, type='C-classification', kernel='linear',scale=FALSE, cost = 1)
c.svm.pred <- predict(c.svm, test[ ,1:57])
c.svm.result <- table(c.svm.pred, test[ ,58])
print('Linear SVM results for V1-V57: ')
print(paste(' -Accuracy of the model is ', percent((c.svm.result[1,1] + c.svm.result[2,2])/sum(c.svm.result))))
print(paste(' -Sensitivity of the model is ', percent((c.svm.result[2,2])/sum(c.svm.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((c.svm.result[1,1])/sum(c.svm.result[ ,1]))))
```

d. Non-Linear SVM (V55-V57)

	Accuracy	Sensitivity	Specificity
Non-linear SVM	80.14%	67.39%	88.20%

```r
#============= d) Non-Linear SVM V55-V57 =============#
d.svm <- ksvm(V58 ~ V55 + V56 + V57, data = train, kernel = 'rbfdot')
d.svm.pred <- predict(d.svm, test[ ,55:57]) > 0.5
d.svm.result <- table(d.svm.pred, test[, 58])
print('Non-Linear SVM results for V55-V57: ')
print(paste(' -Accuracy of the model is ', percent((d.svm.result[1,1] + d.svm.result[2,2])/sum(d.svm.result))))
print(paste(' -Sensitivity of the model is ', percent((d.svm.result[2,2])/sum(d.svm.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((d.svm.result[1,1])/sum(d.svm.result[ ,1]))))
```

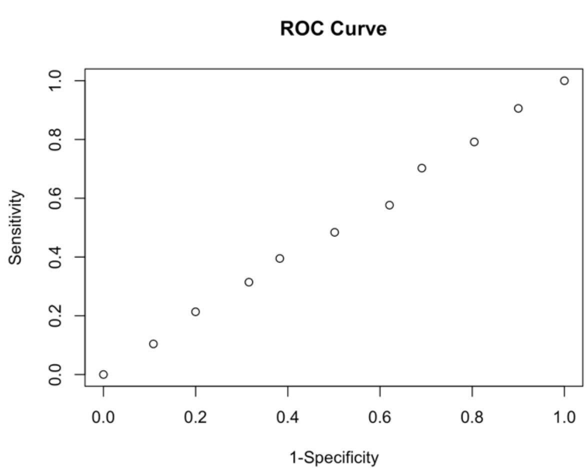
From these two results tables we can see that with a non-linear SVM, accuracy, sensitivity and specificity all decrease. However, the decrease of these three numbers might cause of the decrease of the numbers of variables. Since we have 57 variables when performing linear SVM, but only 3 variables when performing non-linear SVM.

e. Random Prediction

	Accuracy	Sensitivity	Specificity
Random	49.67%	50.59%	49.10%

```r
#==================================== e) random prediction ==========================================#
test$e.rand = rbinom(nrow(test), 1, 0.5)
e.svm.result <- table(test$e.rand, test[, 58])
print('[Random Prediction] Non-Linear SVM results for V55-V57: ')
print(paste(' -Accuracy of the model is ', percent((e.svm.result[1,1] + e.svm.result[2,2])/sum(e.svm.result))))
print(paste(' -Sensitivity of the model is ', percent((e.svm.result[2,2])/sum(e.svm.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((e.svm.result[1,1])/sum(e.svm.result[ ,1]))))
```

f. ROC curve




```
> f.result
        Sensitivity Specificity       sum
prob=0    0.0000000   1.0000000 1.0000000
prob=0.1  0.1025210   0.9117960 1.0143170
prob=0.2  0.1462185   0.7991498 0.9453683
prob=0.3  0.2857143   0.6843783 0.9700926
prob=0.4  0.4000000   0.6046759 1.0046759
prob=0.5  0.4924370   0.5058448 0.9982818
prob=0.6  0.6218487   0.4112646 1.0331134
prob=0.7  0.7092437   0.2826780 0.9919217
prob=0.8  0.7747899   0.2040383 0.9788282
prob=0.9  0.8991597   0.1083953 1.0075550
prob=1    1.0000000   0.0000000 1.0000000
```

The best probability is 0.4. Because when probability equals 0.4, the sum of sensitivity and specificity is maximized.

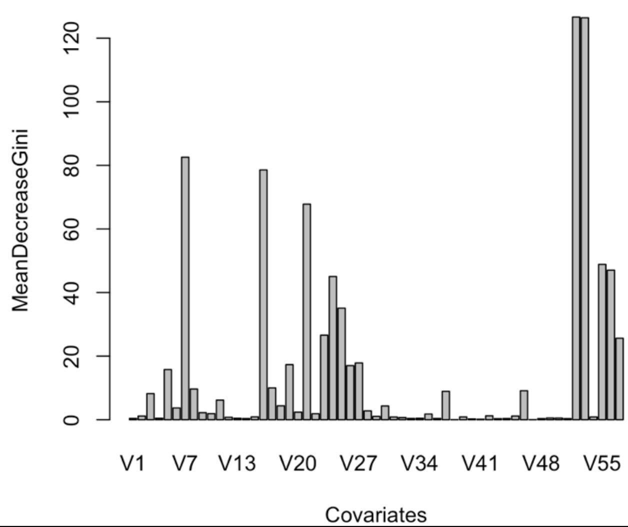
```r
#==================================== f) random prediction sequence of prob ====================================
f.probs = seq(0.1, 0.9, 0.1)
f.result = data.frame(matrix(nrow = length(f.probs)+2, ncol = 2))
colnames(f.result) <- c('Sensitivity', 'Specificity')
row.names(f.result) <- sapply(c(0, f.probs, 1), function(x) paste('prob=', x, sep = ''))
f.result[1, ] = c(0, 1)
f.result[11, ] = c(1, 0)
for(prob in f.probs){
  rand.prob <- rbinom(nrow(test), 1, prob)
  table <- table(rand.prob, test[ , 58])
  f.result[prob*10+1, 'Sensitivity'] = percent((table[2,2])/sum(table[ ,2]))
  f.result[prob*10+1, 'Specificity'] = percent((table[1,1])/sum(table[ ,1]))
}
x_axes = 1- f.result$Specificity
plot(x = x_axes, y = f.result$Sensitivity, xlab= '1-Specificity' , ylab = 'Sensitivity' , main = 'ROC Curve');
f.result$sum = apply(f.result, 1, sum)
```

g. Random Forest

	Accuracy	Sensitivity	Specificity
Random Forest	92.77%	88.24%	95.64%

```r
#============================================== g) random forest ==============================================#
g.rf.fit <- randomForest(as.factor(V58) ~ ., data = train, ntree = 500, mtry = 7, nodesize = 300, importance = TRUE)
g.rf.pred <- predict(g.rf.fit, test[ ,1:57])
g.rf.result <- table(g.rf.pred, test$V58)
print('Random Forest results for V1-V57: ')
print(paste(' -Accuracy of the model is ', percent((g.rf.result[1,1] + g.rf.result[2,2])/sum(g.rf.result))))
print(paste(' -Sensitivity of the model is ', percent((g.rf.result[2,2])/sum(g.rf.result[ ,2]))))
print(paste(' -specificity of the model is ', percent((g.rf.result[1,1])/sum(g.rf.result[ ,1]))))
```

h. MeanDecreasseGini




```r
#==== h) random forest MeanDecreaseGini barplot ====#
barplot(importance(g.rf.fit)[,4], xlab = 'Covariates', ylab = 'MeanDecreaseGini')
```