Math405 – Lab 8

Nick Huo

2022-12-07

Question 1

1. n=25

a.

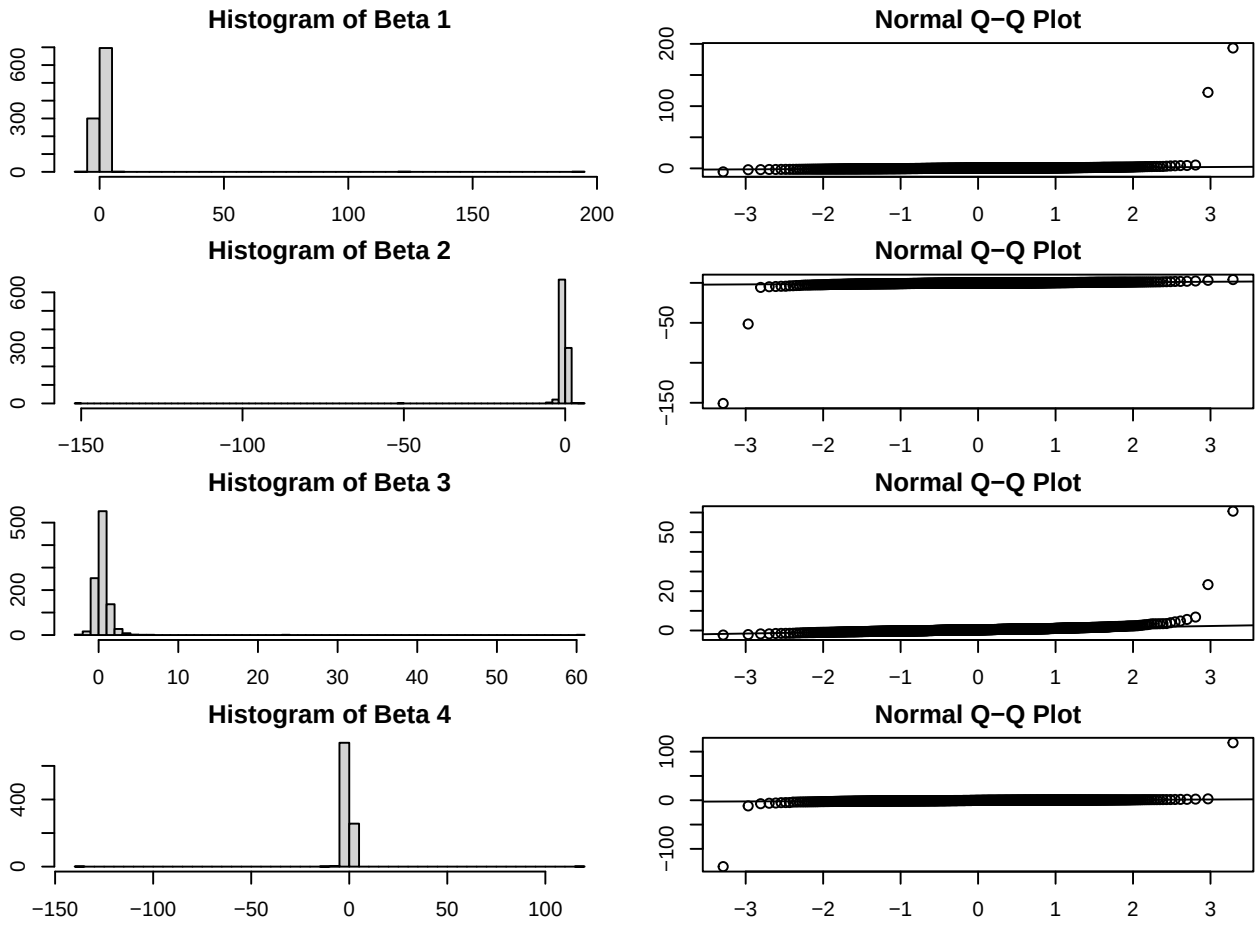
```
x25 <- read.csv("X25.csv")
X <- as.matrix(x25)
Beta <- matrix(c(0.2,-0.2,0.3,-0.3),4,1)
pi <- (exp(X%*%Beta)) / (1+exp(X%*%Beta))
```

b.

```
n <- nrow(x25)
Betas <- matrix(NA,1000,4)
for (i in 1:1000) {
    y_i <- rbinom(n,size=1,prob=pi)
    this_mod <- glm(y_i ~ ., data=as.data.frame(X), family=binomial(link="logit"))
    Betas[i,] <- this_mod$coefficients[-1]
}
```

c.

```
par(mfrow=c(4,2), mar=rep(2,4))
for (i in 1:4) {
    hist(Betas[,i], main=paste("Histogram of Beta", as.character(i)), breaks=60)
    qqnorm(Betas[,i])
    qqline(Betas[,i])
}
```



d.

When $n = 25$, each of the β coefficients has a sampling distribution is unimodal but also extremely skewed to one direction or the other. It is hard to say that this is normal. The QQ plots also indicate the skewness. And the variance is huge.

2. n=50

a.

```
x50 <- read.csv("X50.csv")
X <- as.matrix(x50)
Beta <- matrix(c(0.2,-0.2,0.3,-0.3),4,1)
pi <- (exp(X%*%Beta)) / (1+exp(X%*%Beta))
```

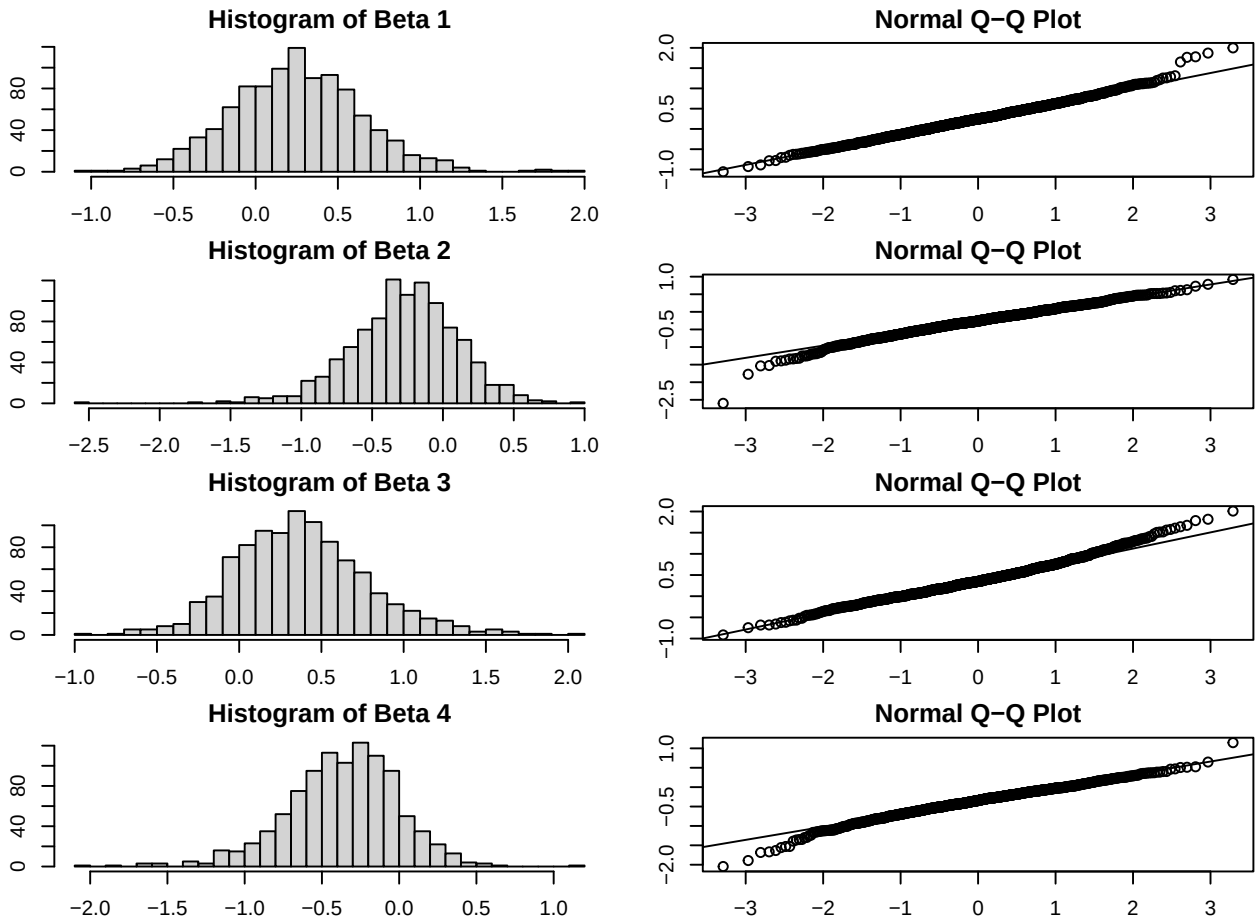
b.

```
n <- nrow(x50)
Betas <- matrix(NA,1000,4)
```

```
for (i in 1:1000) {
    y_i <- rbinom(n,size=1,prob=pi)
    this_mod <- glm(y_i ~ ., data=as.data.frame(X), family=binomial(link="logit"))
    Betas[i,] <- this_mod$coefficients[-1]
}
```

c.

```
par(mfrow=c(4,2), mar=rep(2,4))
for (i in 1:4) {
    hist(Betas[,i], main=paste("Histogram of Beta", as.character(i)), breaks=30)
    qqnorm(Betas[,i])
    qqline(Betas[,i])
}
```



d.

When $n = 50$, the histograms and QQ plots still indicate some slight skewness, but mostly we see that the sampling distributions are approximately normal. Variance now is a lot smaller.

3. n=100

a.

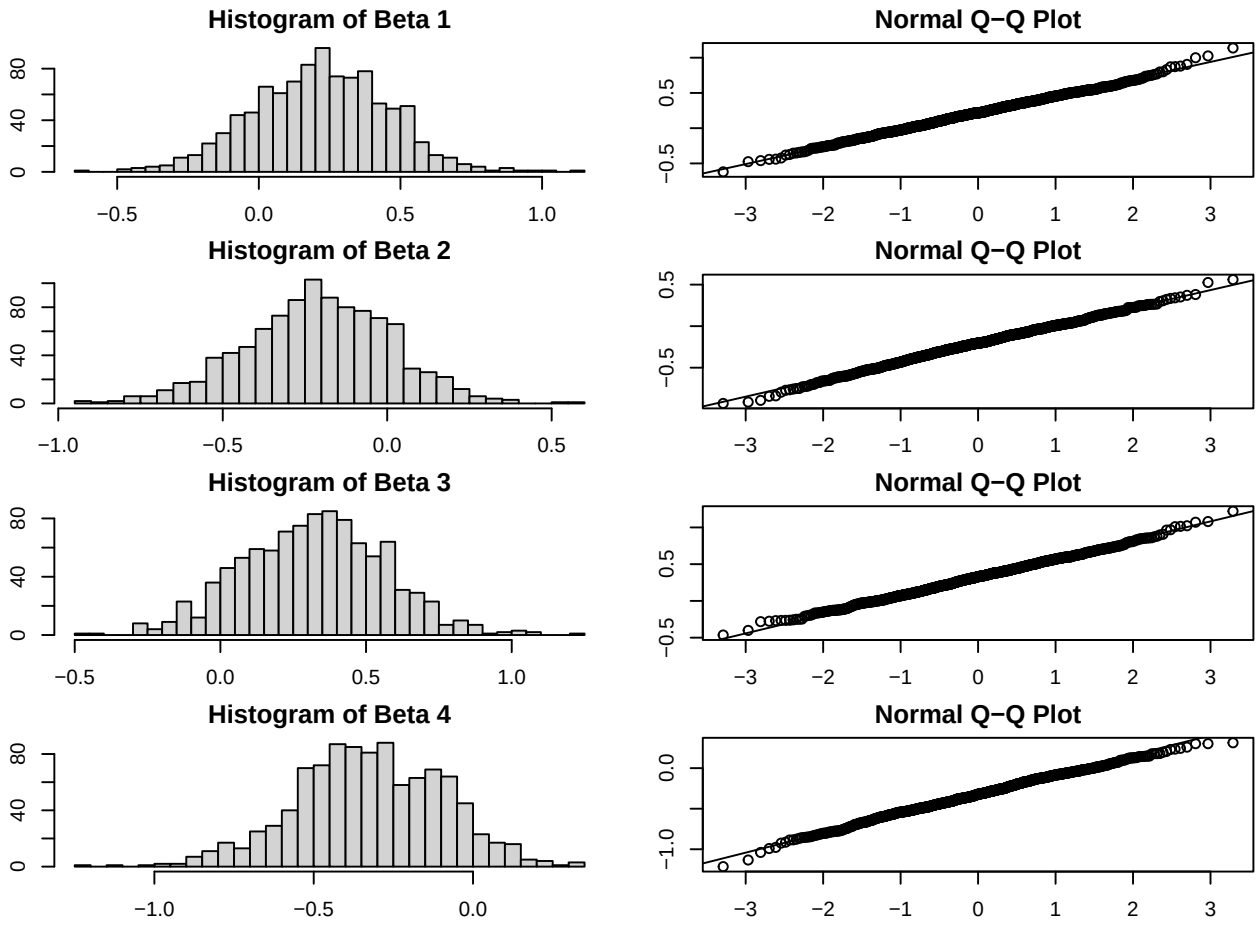
```
x100 <- read.csv("X100.csv")
X <- as.matrix(x100)
Beta <- matrix(c(0.2,-0.2,0.3,-0.3),4,1)
pi <- (exp(X%*%Beta)) / (1+exp(X%*%Beta))
```

b.

```
n <- nrow(x100)
Betas <- matrix(NA,1000,4)
for (i in 1:1000) {
    y_i <- rbinom(n,size=1,prob=pi)
    this_mod <- glm(y_i ~ ., data=as.data.frame(X), family=binomial(link="logit"))
    Betas[i,] <- this_mod$coefficients[-1]
}
```

c.

```
par(mfrow=c(4,2), mar=rep(2,4))
for (i in 1:4) {
    hist(Betas[,i], main=paste("Histogram of Beta", as.character(i)), breaks=30)
    qqnorm(Betas[,i])
    qqline(Betas[,i])
}
```



d.

When $n = 100$, each of the β coefficients has a sampling distribution that is becoming less skewed and more normal. The tails are getting smaller, and the QQ plots are following on the straight line more. The variance is even smaller.

4. n=500

a.

```
x500 <- read.csv("X500.csv")
X <- as.matrix(x500)
Beta <- matrix(c(0.2,-0.2,0.3,-0.3),4,1)
pi <- (exp(X%*%Beta)) / (1+exp(X%*%Beta))
```

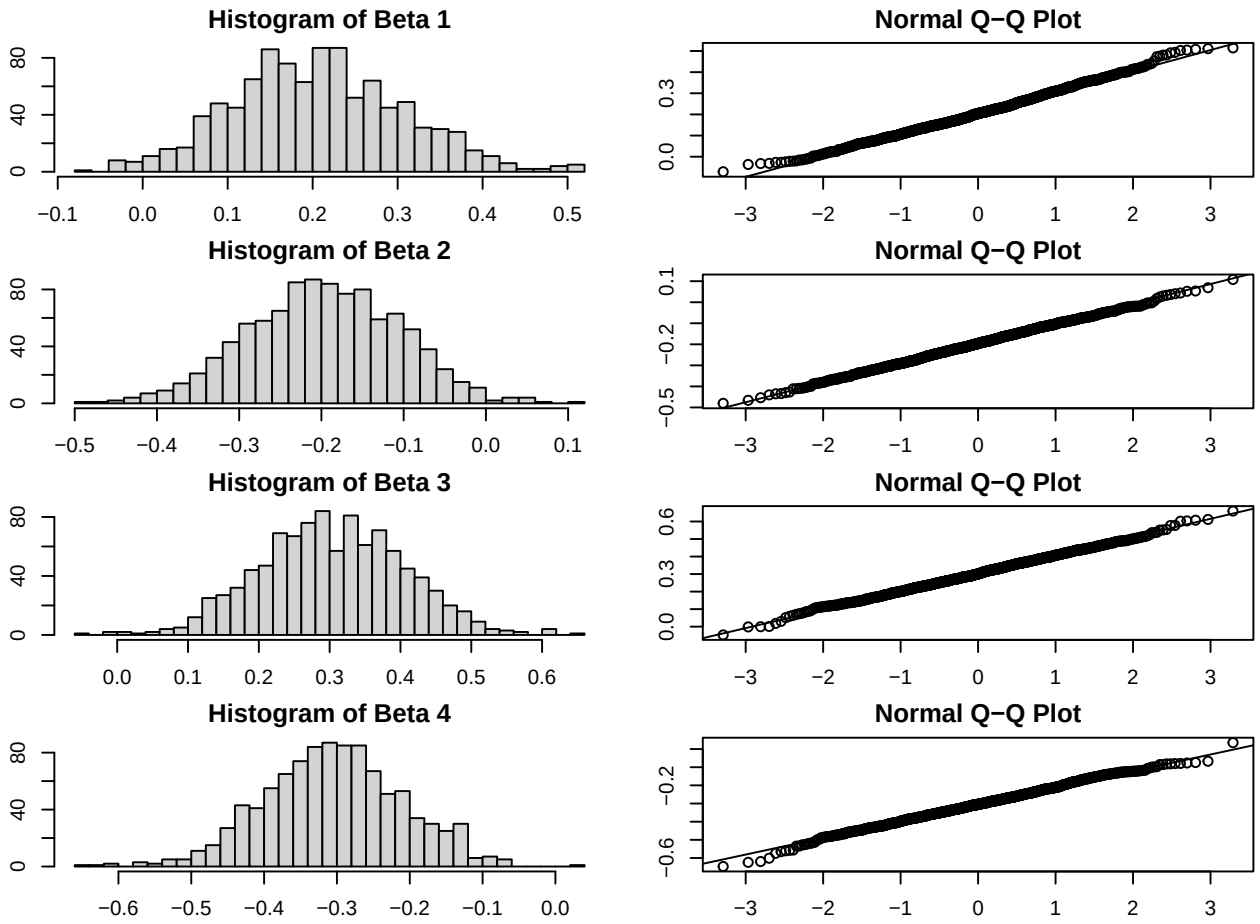
b.

```
n <- nrow(x500)
Betas <- matrix(NA,1000,4)
```

```
for (i in 1:1000) {
    y_i <- rbinom(n,size=1,prob=pi)
    this_mod <- glm(y_i ~ ., data=as.data.frame(X), family=binomial(link="logit"))
    Betas[i,] <- this_mod$coefficients[-1]
}
```

c.

```
par(mfrow=c(4,2), mar=rep(2,4))
for (i in 1:4) {
    hist(Betas[,i], main=paste("Histogram of Beta", as.character(i)), breaks=30)
    qqnorm(Betas[,i])
    qqline(Betas[,i])
}
```



d.

When $n = 500$, we see each of the β coefficients has a sampling distribution that is really normal, indicated by the plots. And the variance is getting even smaller.

5.

I notice that, as the sample size increases, the distributions are becoming more less skewed and more normal. And the variance is also decreasing.

Question 2

1. Import and Split Dataset

```
spam <- read.csv("spam.csv")
spam.train <- spam[spam$test.id==FALSE,-59]
spam.test <- spam[spam$test.id==TRUE,-59]
```

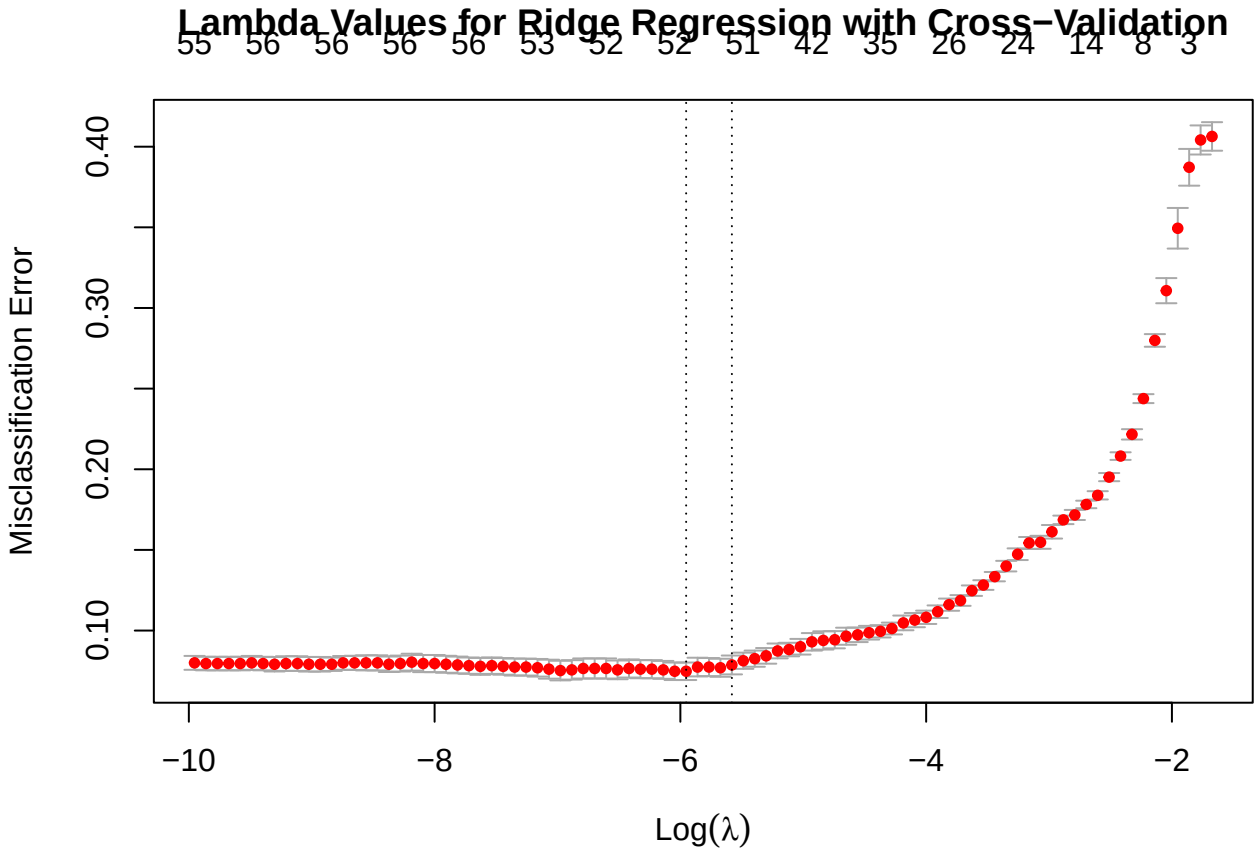
2. Run LASSO on training data w/ 10-fold CV to select variables

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
lasso_cv <- cv.glmnet(x=as.matrix(spam.train[,-58]),y=as.matrix(spam.train[,58]),
          family="binomial",type.measure="class")
plot(lasso_cv, main="Lambda Values for Ridge Regression with Cross-Validation")
```




```
lasso_cv$lambda.min
```

```
## [1] 0.002597817
```

```
lasso_cv$lambda.1se
```

```
## [1] 0.003768988
```

3.

```
lasso_mod <- glmnet(x=as.matrix(spam.train[,-58]),y=as.matrix(spam.train[,58]),
          lambda=lasso_cv$lambda.1se, family="binomial")
lasso_mod$beta
```

```
## 57 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## make              -0.2627389828
## address           -0.0788662454
## all                0.2444006414
## num3d              0.1304006066
## our                0.4848413250
## over               0.4335633941
## remove             2.1641515752
```

```
## internet           0.5511110853
## order              0.1668437017
## mail               0.0770542679
## receive           -0.1389491633
## will              -0.1459607642
## people             .
## report             .
## addresses          0.6341095094
## free               0.6629999751
## business           0.7292409024
## email              0.2207963102
## you                0.0720284125
## credit             0.3554444012
## your               0.2456317898
## font               0.2213294803
## num000             2.1710463963
## money              0.6615124133
## hp                -1.2796606501
## hpl               -0.2584985755
## george            -0.5674350797
## num650             0.2358254409
## lab               -0.1009599481
## labs              -0.1043833540
## telnet             .
## num857             .
## data              -0.8994323336
## num415             .
## num85             -0.4930021155
## technology         0.3205313481
## num1999           -0.0791705502
## parts             -0.2008168708
## pm                -0.0873274548
## direct            -0.1143455833
## cs                -0.9088696986
## meeting           -0.8434055211
## original          -0.3254808342
## project           -0.3622822478
## re                -0.5687044122
## edu               -0.5395078275
## table             -0.2385871495
## conference        -0.6080517161
## charSemicolon     -0.6904457651
## charRoundbracket  -0.1305700854
## charSquarebracket -0.3488851020
## charExclamation    0.4511301328
## charDollar         4.5098312052
## charHash           0.3920096754
## capitalAve         .
## capitalLong        0.0036990314
## capitalTotal       0.0006803063
```

4.

```
pi_pred <- predict(lasso_mod, newx=as.matrix(spam.test[,-58]), type="response")
summary(pi_pred)
```

```
##        s0
##  Min.   :0.00000
##  1st Qu.:0.03298
##  Median :0.18346
##  Mean   :0.37632
##  3rd Qu.:0.78666
##  Max.   :1.00000
```

5.

```
pred_results <- c()
for (i in pi_pred) {
    if (i>0.5) {
        pred_results <- c(pred_results, 1)
    }
    else {
        pred_results <- c(pred_results, 0)
    }
}
summary(pred_results)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.3448  1.0000  1.0000
```

```
length(pred_results)
```

```
## [1] 2300
```

```
sum(pred_results)
```

```
## [1] 793
```

6.

```
pred_actual <- data.frame(predicted=pred_results, actual=spam.test[,58])
p1a1 <- nrow(pred_actual[pred_actual$predicted==1 & pred_actual$actual==1,])
p1a0 <- nrow(pred_actual[pred_actual$predicted==1 & pred_actual$actual==0,])
p0a1 <- nrow(pred_actual[pred_actual$predicted==0 & pred_actual$actual==1,])
p0a0 <- nrow(pred_actual[pred_actual$predicted==0 & pred_actual$actual==0,])

tab_df <- data.frame("Actual Spam"=c(p1a1,p0a1),
                     "Actual No-Spam"=c(p1a0,p0a0),
                     row.names=c("Predicted Spam","Predicted No-Spam"))
knitr::kable(tab_df)
```

	Actual.Spam	Actual.No.Spam
Predicted Spam	739	54
Predicted No-Spam	140	1367

In the 800 emails predicted as spam, 747 were actually spam and 53 were not spam.

In the 1500 emails predicted as not spam, 132 were actually spam and 1368 were not spam.

7.

```
misclf <- (p0a1+p1a0)/nrow(pred_actual)
misclf
```

```
## [1] 0.08434783
```

The misclassification rate is about 8.5%

8.

I think Type I Error (False Positive) is a bigger deal here, because we might miss important emails. False Negative will result in us seeing more spam emails. But for most people, they wouldn't be easily tricked.

9.

If we raise the cut-off probability (such as to 0.7), it is "harder" for an email to be classified as spam, thus Type I Error would decrease, resulting in less blocked real emails. But Type II Error (False Negative) will increase, resulting in more spam emails in the inbox. I think the misclassification rate my increase because there were more Spam emails and the increase in False Negative Error will be more than the decrease in False Positive Error.

And if we decrease the cut-off probability (such as to 0.3), we will see the opposite. So I think the misclassification rate my decrease because there were more Spam emails and the decrease in False Negative Error will be more than the increase in False Positive Error.

Question 3

1.

```
sensitivity <- p1a1/(p1a1+p0a1)
sensitivity
```

```
## [1] 0.8407281
```

Given that an email is spam, the probability that we will correctly identify it as spam is 84.98%.

2.

```
specificity <- p0a0/(p1a0+p0a0)
specificity
```

```
## [1] 0.9619986
```

Given that an email is not spam, the probability that we will correctly identify it as not spam is 96.27%.

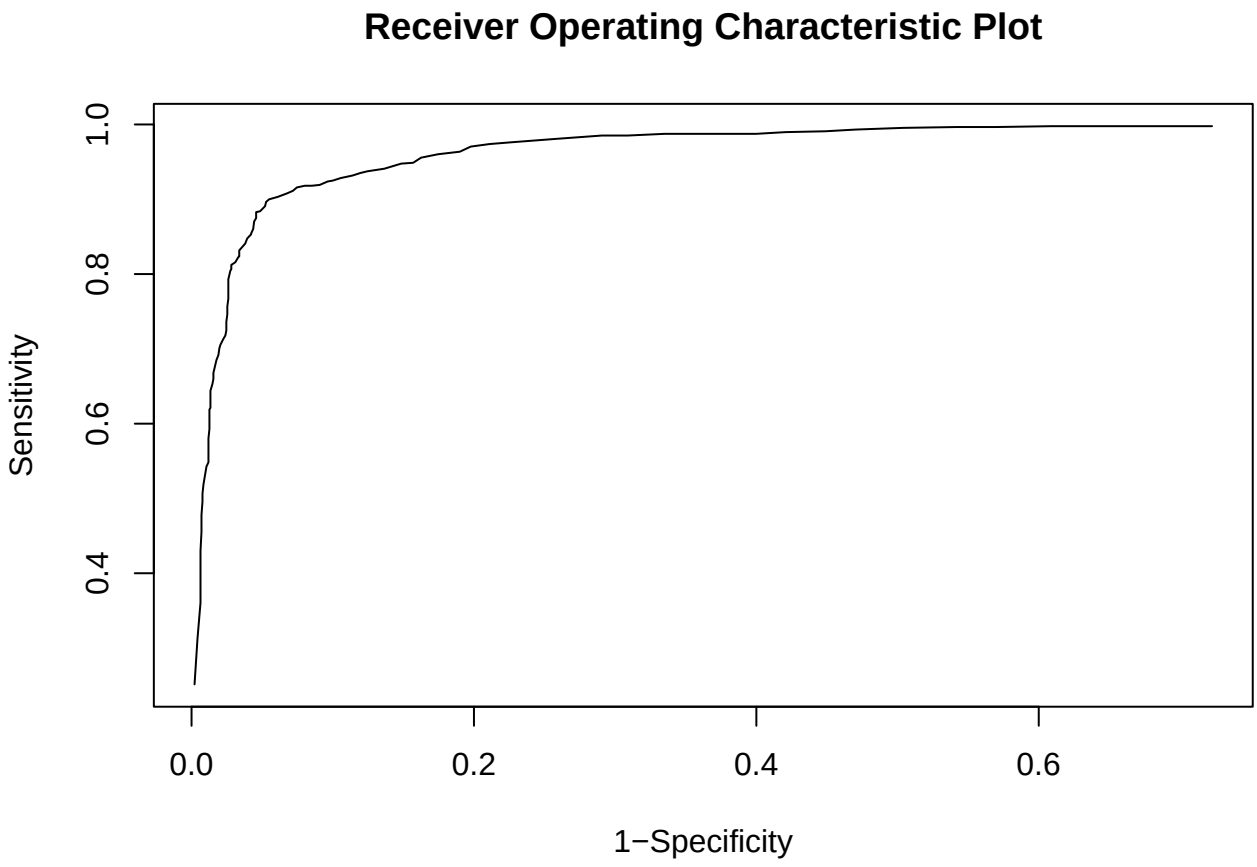
3.

```
cutoff <- seq(0.01,0.99, 0.01)
cutoff_df <- data.frame("cutoff"=cutoff, "sensitivity"=NA, "specificity"=NA,
                        row.names=cutoff)

# for each cut-off prob, recompute predicted outcome based on cut-off
for (p in cutoff) {
    pred_results_i <- c()
    for (i in pi_pred) {
        if (i > p) {
            pred_results_i <- c(pred_results_i, 1)
        }
        else {
            pred_results_i <- c(pred_results_i, 0)
        }
    }
    pred_actual_p <- data.frame(predicted=pred_results_i, actual=spam.test[,58])
    p1a1_p <- nrow(pred_actual_p[pred_actual_p$predicted==1 & pred_actual_p$actual==1,])
    p1a0_p <- nrow(pred_actual_p[pred_actual_p$predicted==1 & pred_actual_p$actual==0,])
    p0a1_p <- nrow(pred_actual_p[pred_actual_p$predicted==0 & pred_actual_p$actual==1,])
    p0a0_p <- nrow(pred_actual_p[pred_actual_p$predicted==0 & pred_actual_p$actual==0,])
    sens_p <- p1a1_p/(p1a1_p+p0a1_p)
    spec_p <- p0a0_p/(p1a0_p+p0a0_p)
    cutoff_df[cutoff_df$cutoff==p,]$sensitivity <- sens_p
    cutoff_df[cutoff_df$cutoff==p,]$specificity <- spec_p
}
```

4.

```
plot(x=(1-cutoff_df$specificity), y=cutoff_df$sensitivity, type='l',
     xlab="1-Specificity", ylab="Sensitivity", main="Receiver Operating Characteristic Plot")
```



5.

```
dists <- c()
for (i in 1:nrow(cutoff_df)) {
    dist <- sqrt((1-cutoff_df[i,]$sensitivity)^2 + (1-cutoff_df[i,]$specificity)^2)
    dists <- c(dists,dist)
}
cutoff_df[which(dists==min(dists)),]
```

```
##      cutoff sensitivity specificity
## 0.35   0.35   0.9158134   0.9254046
```

Looking at the ROC plot, I think the cut-off probability that corresponds to the elbow-point for my model is the one that yields around 0.9 sensitivity and 0.05 (1-specificity).

By finding the cut-off probability that yields the minimum distance in the ROC plot to the point (0,1), I found 0.35 as the best cut-off probability, where we achieved around 91.6% sensitivity and 92.5% specificity.

The performance is quite well I think, with is being sensitive enough to spam emails, while making sure that only a few non-spam emails got misclassified as spam, causing us to potentially miss important emails, like an actually winning.