The Patterns of Paying Debts of Credit Card Clients

Introduction

The purpose of this report is to predict a relationship between independent and dependent variables to see how the independent variables affect the dependent variable. The report is applied with different machine learning methods to analyze graphs and classification functions to see correlations and dependencies between these independent and dependent variables. Classification models also will be used in order to discuss how intercept and coefficients are affecting the estimated outcome and how important those are. This report is based on data called, "Default of Credit Card Clients Data Set" and further analyzes what and how the independent variables affect people not to pay off their debts. There are 25 variables in this data set. The dependent variable is default.payment.next.month. The independent variables are ID, MARRIAGE, AGE, etc. To predict the relationship between the independent and dependent variables, a box-plot method and logistic and probit classifications are used to explain these independent and dependent variables since the variables are discrete. In addition to that, ridge, lasso, tree, bagging, random forest, boosting, XG boost, support vector machine, and neural network models will be used to estimate a better prediction to explain this data. Innovative and effective density graphs are added to explain more of the relationship between the independent and the dependent variables to see how these are correlated. Another main purpose is to predict accuracy rates and error rates, and these will be compared with all of the models to find the best model that has the highest accuracy rate. This will improve the performance of the classification model.

X variables are:

- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years

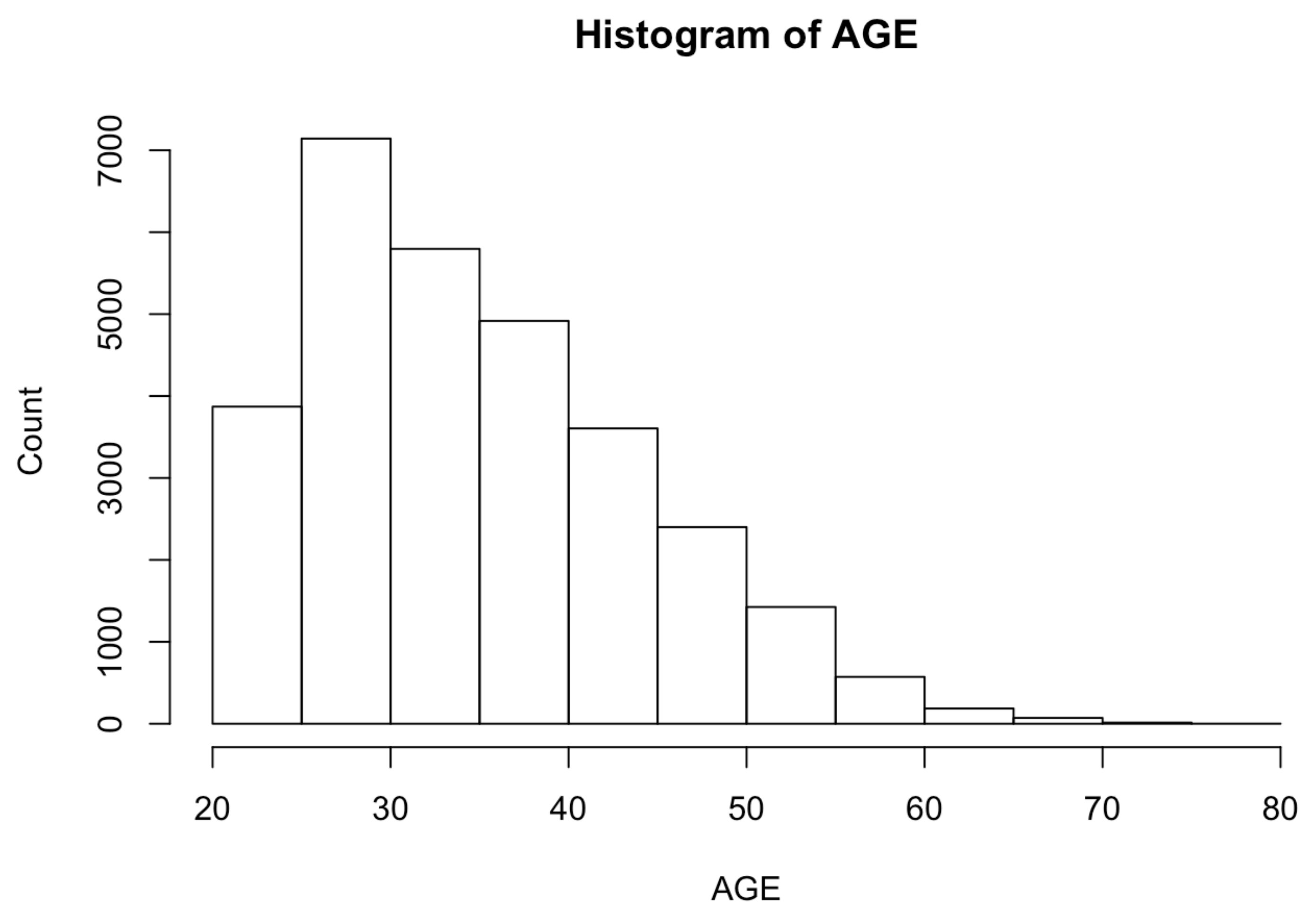
- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

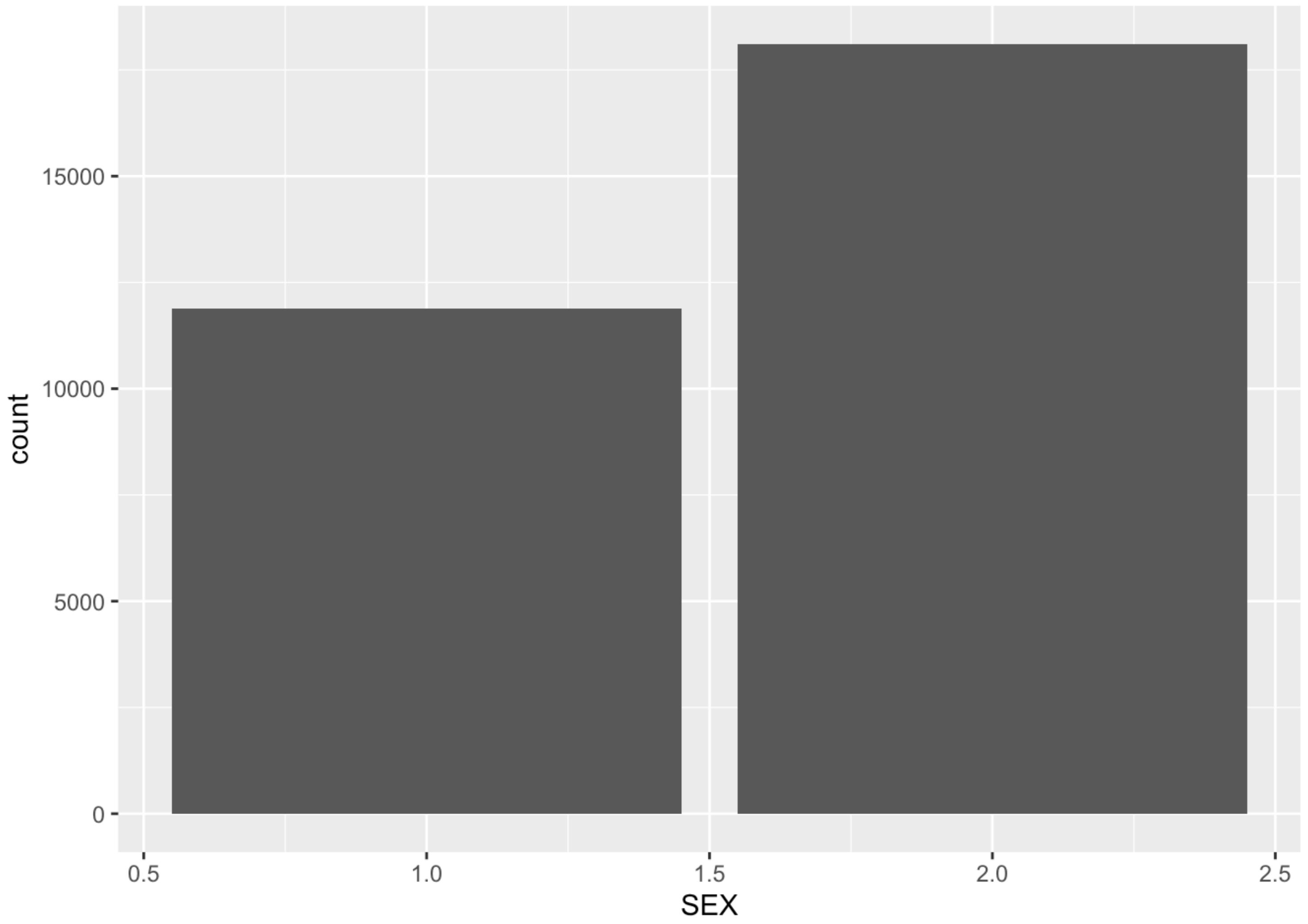
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Summary Statistics



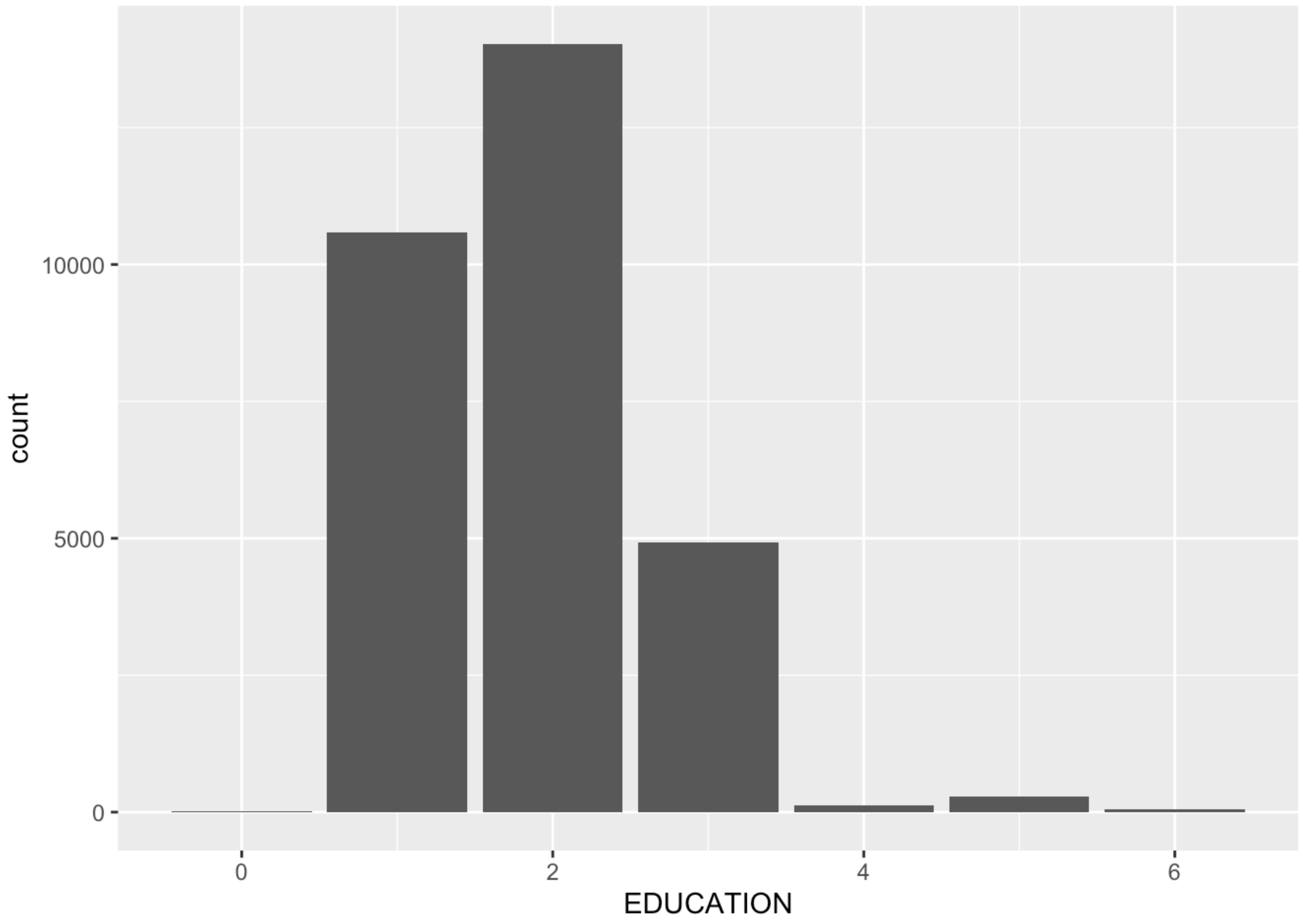
I can see this graph has a skewness to right. In other words, few people are older than other young people. Also, I can see that between 20 and 30 is the highest point in this graph. The range of the ages is from 1 to 56. The mean of the age is 15.4854. It is likely that young people have more debts than old people by its population.

```
x=as.factor(CC$SEX)
ggplot(data.frame(SEX), aes(x=SEX)) + geom_bar()
```



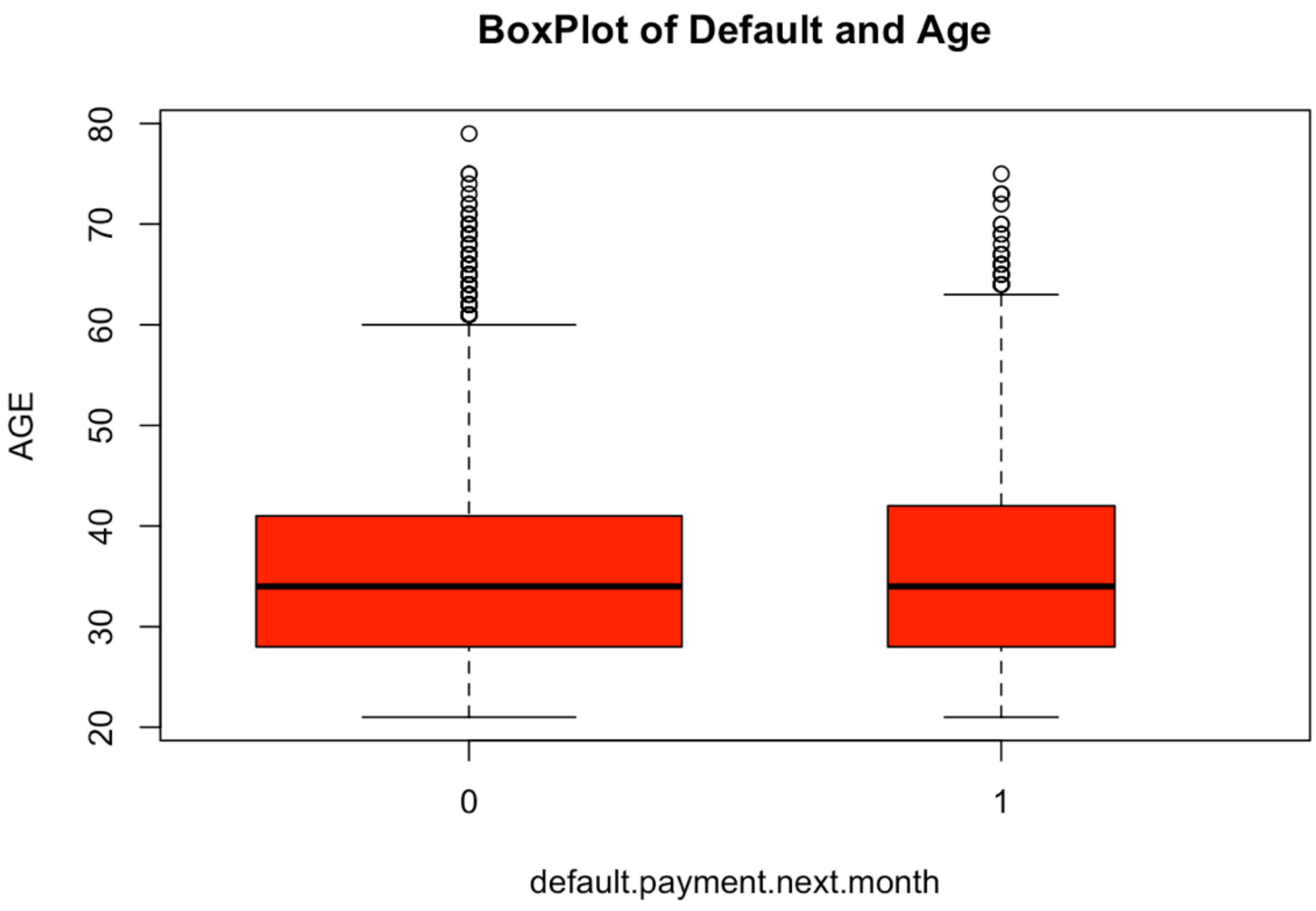
In this graph, 1.0 represents men, and 2.0 represents women. This graph shows that there are more women than men. This represents that women use more credit cards and did not pay off than men in this data.

```
ggplot(data.frame(EDUCATION), aes(x=EDUCATION)) + geom_bar()
```



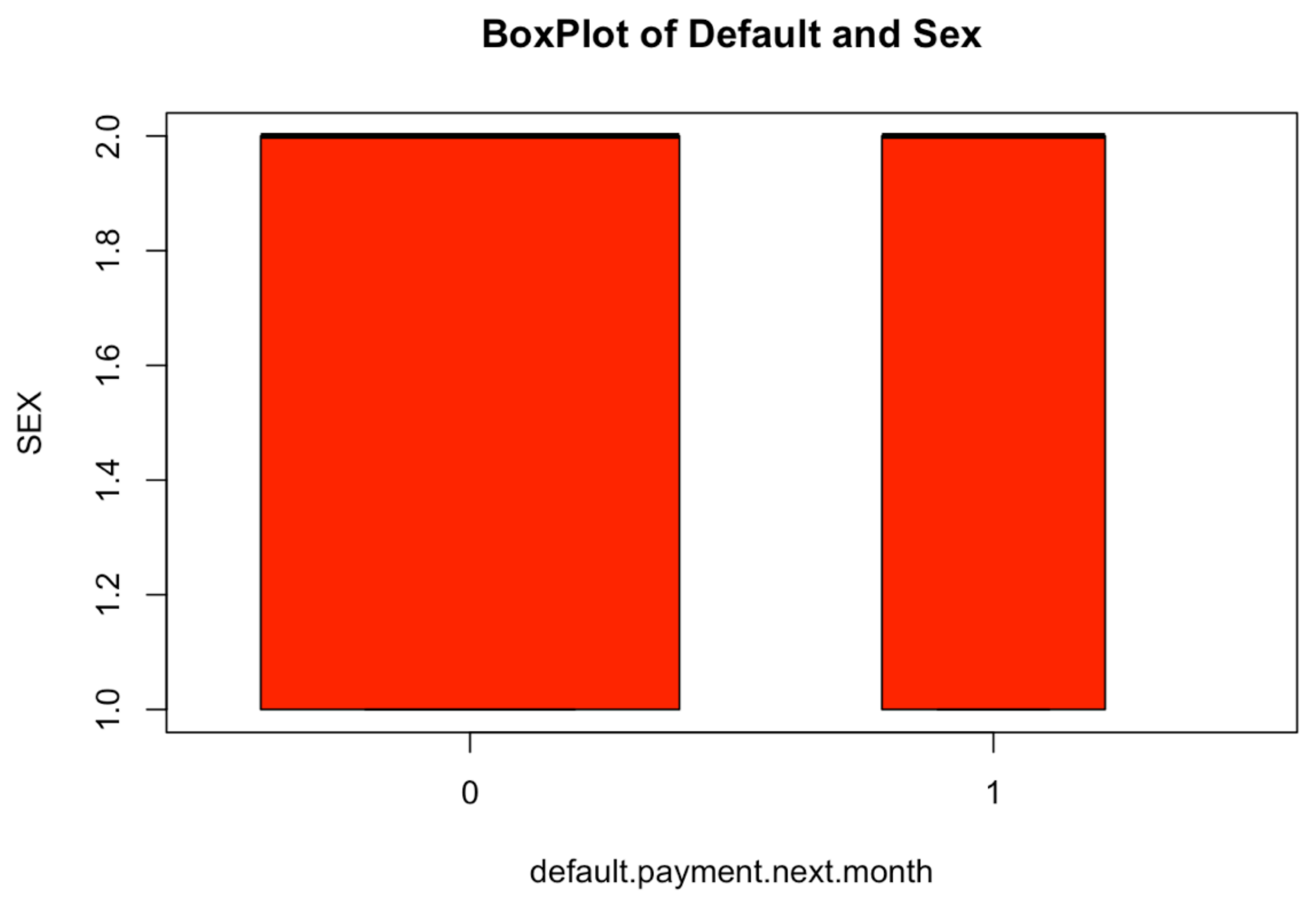
In this histogram, most of the people have college degrees than other higher education. In addition to that, people are having more graduate degrees than high school and unknowns degrees. Some people are having high school degrees only.

```
plot(as.factor(CC$default.payment.next.month), (CC$AGE), col="red",
varwidth=T, xlab="default.payment.next.month", ylab="AGE", main="Box
Plot of Default and Age")
```



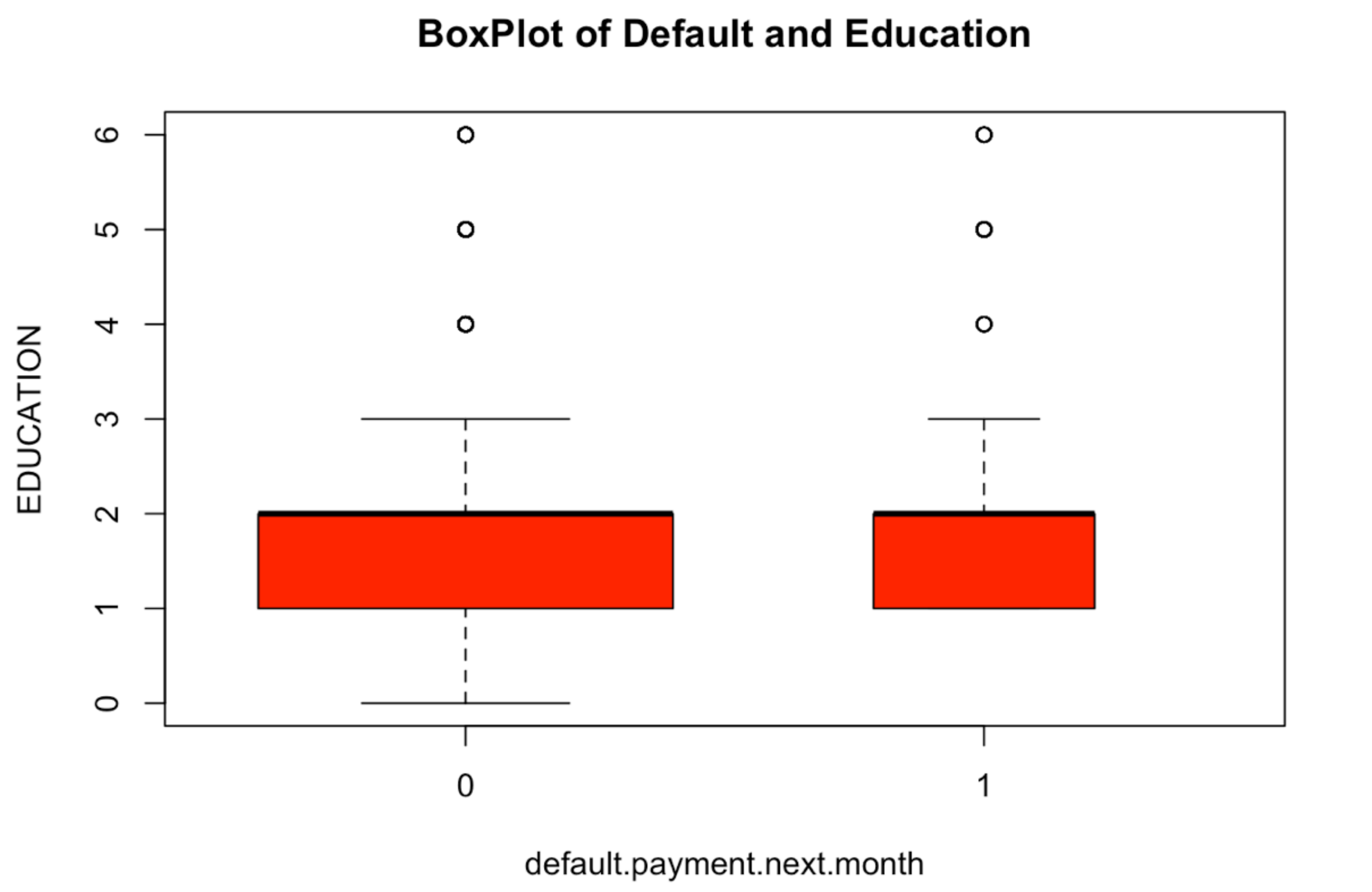
The default zero means it is not default, and the range is 40. The default one means it is default, and its range is almost similar to the default's zero. Moreover, the black lines of the red boxes are the median of the red boxes. The endpoints of the red boxes are the real medians for each of those sections. For example, the top endpoints of the red boxes are the median of greater than the black lines and vice versa.

```
plot(as.factor(CC$default.payment.next.month), (CC$SEX), col="red",
varwidth=T, xlab="default.payment.next.month", ylab="SEX", main="Box
Plot of Default and Sex")
```



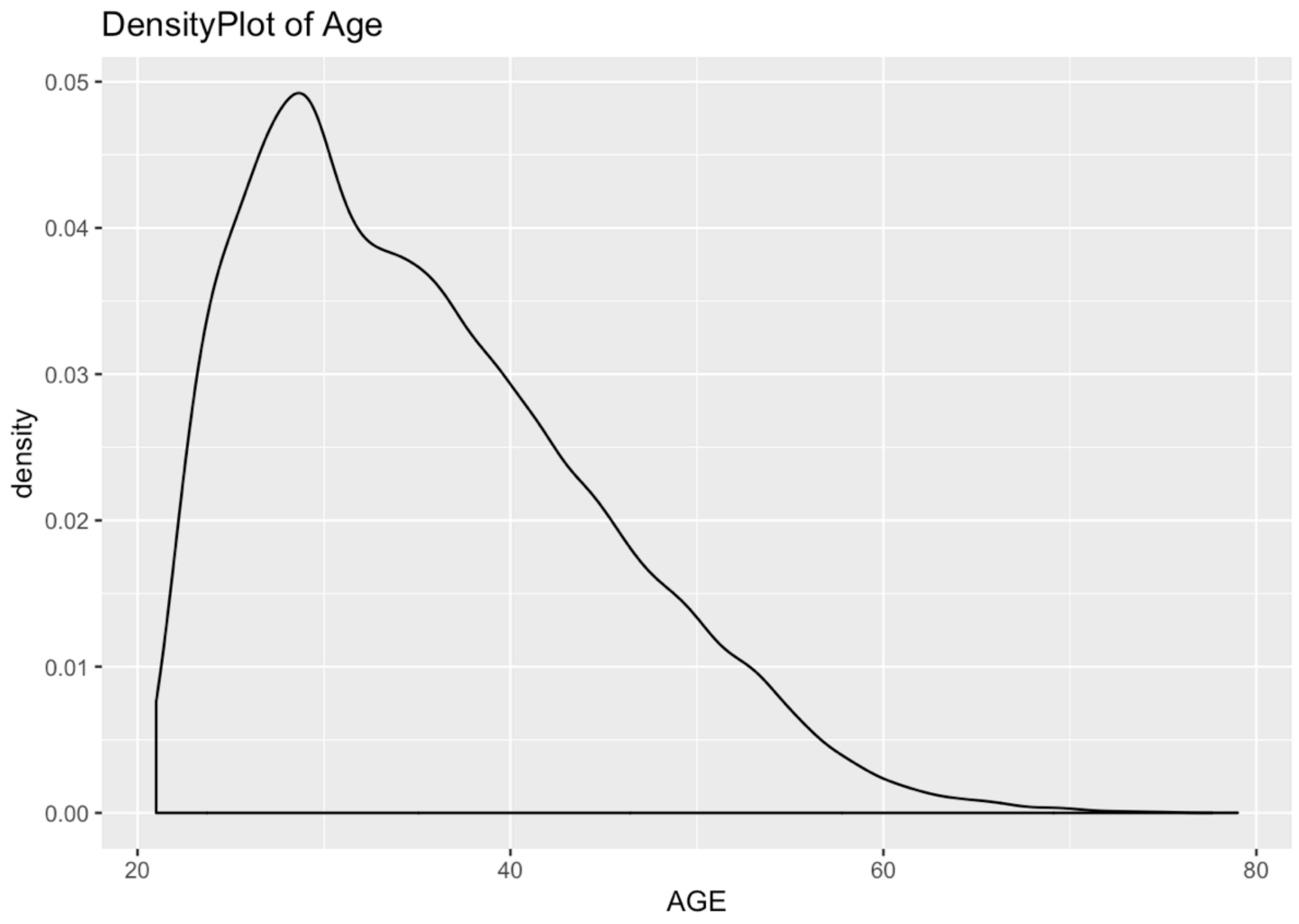
The default zero is not default, and the default one is default. In this box plot, there are no first quartile nor fourth quartile.

```
plot(as.factor(CC$default.payment.next.month), (CC$EDUCATION), col="
red", varwidth=T, xlab="default.payment.next.month", ylab="EDUCATION
", main="BoxPlot of Default and Education")
```



The default zero is not default, and the default one is default. In this box plot, the default 0's range is 3, and the default 1's range is 2. There are circles equally placed on the box plots from 6 to 4. This box plot also represents that the endpoints of the red boxes are the real median of each of these sections which would be the first quartiles and the fourth quartiles.

```
ggplot(CC, aes(AGE)) + geom_density() + ggtitle("DensityPlot of Age"
)
```



This density plot tells us that this graph has a skewness to the right because the line is going down to the right. It peaks between 20 to 40 which means there are a lot of people who are 20’s to ’40s. In other words, there are younger people than older people in this data.

Logistic/Probit Classification

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX,data=CC, f
amily=binomial)
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##               default.payment.next.month
## glm.pred          0     1
##   notDefault 23364  6636
```

```
(6636)/(6636+23364)
```

```
## [1] 0.2212
```

```
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION +
##     SEX, family = binomial, data = CC)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8767  -0.7204  -0.6847  -0.6569   1.8140
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154597   0.077354 -14.926  < 2e-16 ***
## AGE          0.001425   0.001530   0.931    0.352
## EDUCATION    0.082852   0.017604   4.706 2.52e-06 ***
## SEX         -0.194583   0.028325  -6.870 6.44e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 31633  on 29996  degrees of freedom
## AIC: 31641
##
## Number of Fisher Scoring iterations: 4
```

In this logistic regression, its accuracy rate is very low which is 0.2212. This is not good because it is not explaining my data clearly. In addition to that, EDUCATION and SEX variables are significantly important because their p-values are very low. In other words, these variables are affecting the dependent variable. AGE variable is not that important in this case since its p-value is high. This variable can be removed to run a better regression.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0,data
=CC, family=binomial)
glm.probs=predict(glm.fits,CC, type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##             default.payment.next.month
## glm.pred         0     1
##   Default      729  1692
##   notDefault 22635  4944
```

```
(1692+22635)/(729+1692+22635+4944)
```

```
## [1] 0.8109
```

```
logitBestOutcome = 0.8109
mean(glm.pred==CC$default.payment.next.month)
```

```
## [1] 0
```

```
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION +
##     SEX + PAY_0, family = binomial, data = CC)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0381  -0.6736  -0.6312  -0.3241   2.4781
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.397981   0.082167 -17.014  < 2e-16 ***
## AGE          0.006222   0.001607   3.872 0.000108 ***
## EDUCATION   -0.022145   0.019375  -1.143 0.253056
## SEX         -0.115736   0.030110  -3.844 0.000121 ***
## PAY_0        0.738180   0.014352  51.434  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28498  on 29995  degrees of freedom
## AIC: 28508
##
## Number of Fisher Scoring iterations: 4
```

In this logistic regression, its accuracy rate is 0.8109 which means this regression predicts much better than the first regression. Moreover, AGE, SEX, PAY_0 variables are significantly important that their p-values are low which means these variables are affecting the dependent variable. EDUCATION variable is now insignificant in this regression. Furthermore, if one unit of AGE goes up, the dependent variable goes up by 0.006222. If one unit of PAY_0 goes up, the log odds ratio goes up by 0.738180.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0+PAY_
2,data=CC, family=binomial)
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##              default.payment.next.month
## glm.pred         0     1
##   Default      642  1579
##   notDefault 22722  5057
```

```
(1579+22722)/(642+5057+1579+22722)
```

```
## [1] 0.8100333
```

```
summary(glm.fits)
```

```
## 
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION + 
##     SEX + PAY_0 + PAY_2, family = binomial, data = CC)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.1538  -0.6767  -0.6311  -0.3107   2.5322  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.408054   0.082423 -17.083  < 2e-16 ***
## AGE          0.007197   0.001613   4.462 8.13e-06 ***
## EDUCATION   -0.042958   0.019630  -2.188   0.0286 *  
## SEX         -0.097242   0.030246  -3.215   0.0013 ** 
## PAY_0        0.626553   0.017452  35.902  < 2e-16 ***
## PAY_2        0.172919   0.015014  11.517  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28364  on 29994  degrees of freedom
## AIC: 28376
## 
## Number of Fisher Scoring iterations: 4
```

In this logistic regression, its accuracy rate is 0.81. It is less than the second regression. In other words, the second regression predicts better than this regression. For its coefficients, AGE, PAY_0, PAY_2 are significantly important coefficients because their p-values are very low. In this case, SEX is less important than those variables since its p-value is over zero. Education variable is still not that important. Hence, the important variables affect the dependent variable. If one unit of AGE goes up, the dependent variable, y, goes up by 0.007197. In contrast, If EDUCATION goes up by one unit, y goes up by -0.042958.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0+PAY_
2+PAY_3,data=CC, family=binomial)
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##             default.payment.next.month
## glm.pred         0     1
##   Default      578  1471
##   notDefault 22786  5165
```

```
(1471+22786)/(578+5165+1471+22786)
```

```
## [1] 0.8085667
```

```
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION +
##     SEX + PAY_0 + PAY_2 + PAY_3, family = binomial, data = CC)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1818  -0.6799  -0.6289  -0.3036   2.5552
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.411348   0.082461 -17.115  < 2e-16 ***
## AGE          0.007477   0.001614   4.632 3.63e-06 ***
## EDUCATION   -0.047123   0.019684  -2.394  0.01667 *
## SEX         -0.093267   0.030275  -3.081  0.00207 **
## PAY_0        0.618474   0.017602  35.136  < 2e-16 ***
## PAY_2        0.093954   0.019594   4.795 1.63e-06 ***
## PAY_3        0.115596   0.018142   6.372 1.87e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28324  on 29993  degrees of freedom
## AIC: 28338
##
## Number of Fisher Scoring iterations: 4
```

In this logistic regression, its accuracy rate is 0.8086. This is lower than the second regression. This model will not be used since the second regression predicts better than this regression. For coefficients, AGE and PAY_0, PAY_2, and PAY_3 are significantly important since their p-values are very low. In this regression, SEX is less important than those significant variables. In addition to that, EDUCATION is important because its p-value is higher than the other coefficients. If one unit of EDUCATION's coefficient goes up, the dependent variable goes up by -0.047123. In contrast, for PAY_3, if one unit is added, the dependent variable goes up by 0.115596.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0+PAY_
2+PAY_3+PAY_4,data=CC, family=binomial)
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##              default.payment.next.month
## glm.pred          0     1
##   Default       573  1470
##   notDefault 22791  5166
```

```
(1470+22791)/(573+5166+1470+22791)
```

```
## [1] 0.8087
```

```
summary(glm.fits)
```

```
## 
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION + 
##     SEX + PAY_0 + PAY_2 + PAY_3 + PAY_4, family = binomial, data 
= CC)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -3.1816  -0.6815  -0.6300  -0.3020   2.5609  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.407927   0.082483 -17.069  < 2e-16 ***
## AGE          0.007520   0.001615   4.658 3.20e-06 ***
## EDUCATION   -0.048164   0.019698  -2.445  0.01448 *  
## SEX         -0.092525   0.030281  -3.056  0.00225 ** 
## PAY_0        0.614433   0.017715  34.685  < 2e-16 ***
## PAY_2        0.090270   0.019711   4.580 4.66e-06 ***
## PAY_3        0.086222   0.022133   3.896 9.79e-05 ***
## PAY_4        0.045674   0.019462   2.347  0.01893 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28318  on 29992  degrees of freedom
## AIC: 28334
## 
## Number of Fisher Scoring iterations: 4
```

In this logistic regression, its accuracy is 0.8087. It is higher than the fourth regression. However, this is lower than the second regression, so the second regression will be used. For coefficients, AGE, PAY_0, PAY_2, PAY_3 are significantly important since their p-values are very low. SEX is less important than those variables. In this regression, PAY_4 and EDUCATION are not important. In this case, if SEX goes up by one unit, the dependent variable is likely to go up by -0.092525. However, PAY_0's coefficient is 0.614433 which means the dependent variable goes up by that amount if one unit of PAY_0 goes up.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0,data
=CC, family=binomial)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION +
##     SEX + PAY_0, family = binomial, data = CC)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0381  -0.6736  -0.6312  -0.3241   2.4781
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.397981   0.082167 -17.014  < 2e-16 ***
## AGE          0.006222   0.001607   3.872 0.000108 ***
## EDUCATION   -0.022145   0.019375  -1.143 0.253056
## SEX         -0.115736   0.030110  -3.844 0.000121 ***
## PAY_0        0.738180   0.014352  51.434  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28498  on 29995  degrees of freedom
## AIC: 28508
##
## Number of Fisher Scoring iterations: 4
```

The second regression is the best regression because of its accuracy rate: 0.81. In this best regression, most independent variables are affecting the dependent variable. For instance, AGE, SEX, and PAY_0 coefficients are very important due to their p-values. Also, there are three stars next to the p-values. However, EDUCATION is not important because it's p-value is 0.253056. This is high that there is no star next to its p-value. Moreover, the Akaike information criterion is 28,508. A lower number of AIC explains a model better than a higher number of AIC.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0,data
=CC, family=binomial)
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
table(glm.pred,default.payment.next.month)
```

```
##             default.payment.next.month
## glm.pred         0     1
##   Default      729  1692
##   notDefault 22635  4944
```

```
(1692)/(1692+4944)
```

```
## [1] 0.2549729
```

```
4944/(4944+1692)
```

```
## [1] 0.7450271
```

```
summary(glm.fits)
```

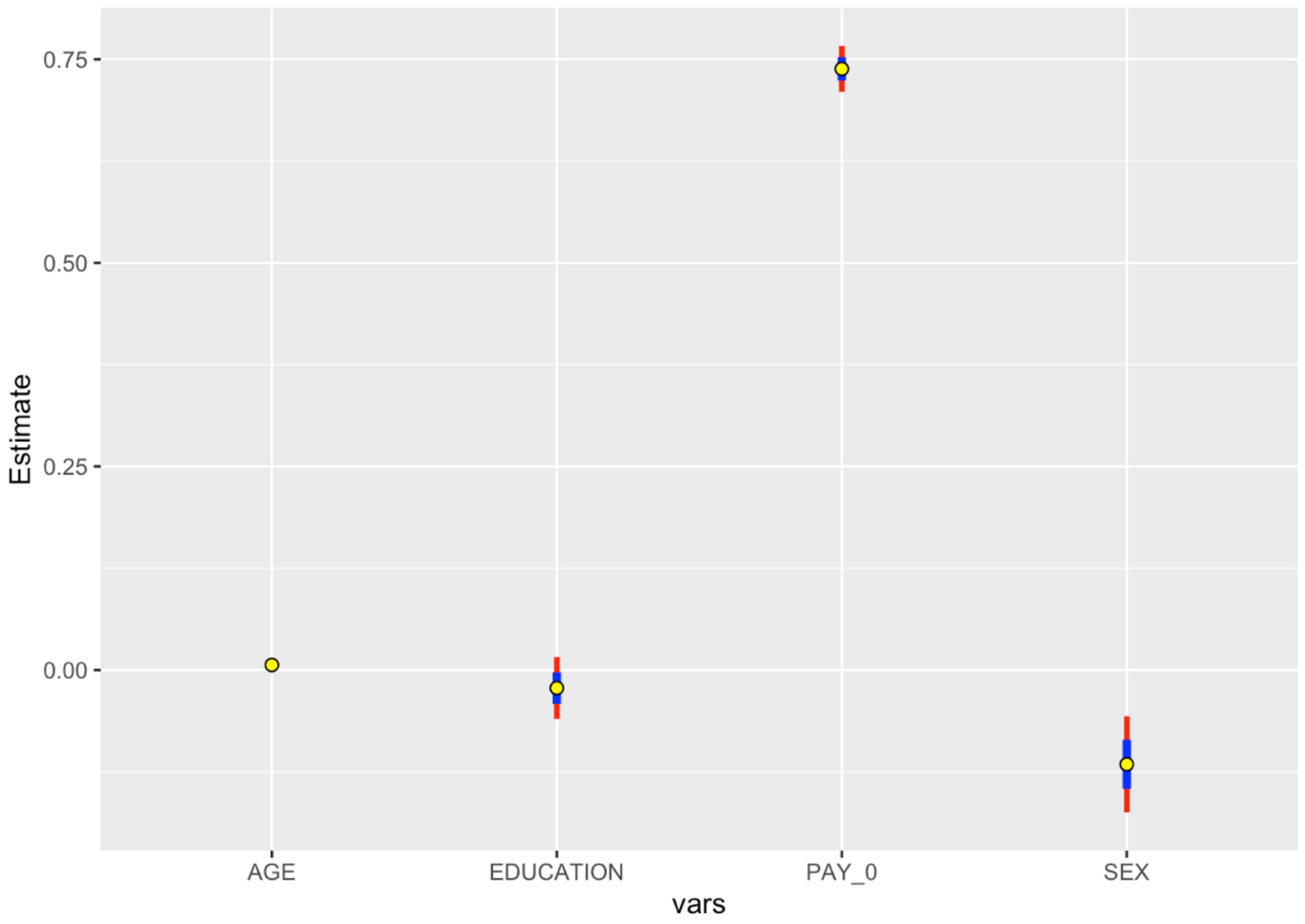
```
##
## Call:
## glm(formula = default.payment.next.month ~ AGE + EDUCATION +
##     SEX + PAY_0, family = binomial, data = CC)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0381  -0.6736  -0.6312  -0.3241   2.4781
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.397981   0.082167 -17.014  < 2e-16 ***
## AGE          0.006222   0.001607   3.872 0.000108 ***
## EDUCATION   -0.022145   0.019375  -1.143 0.253056
## SEX         -0.115736   0.030110  -3.844 0.000121 ***
## PAY_0        0.738180   0.014352  51.434  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31705  on 29999  degrees of freedom
## Residual deviance: 28498  on 29995  degrees of freedom
## AIC: 28508
##
## Number of Fisher Scoring iterations: 4
```

For this logistic regression, the true positive rate is higher than the false positive rate. This means my logistic regression is accurate. In other words, this regression is reliable.

```
glm.summary = summary(glm.fits)
glm.summary$coefficients
```

```
##                 Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept) -1.397980798 0.082166785 -17.013941 6.473363e-65
## AGE          0.006221779 0.001606836   3.872069 1.079152e-04
## EDUCATION   -0.022144605 0.019374805  -1.142959 2.530557e-01
## SEX         -0.115735560 0.030109866  -3.843775 1.211559e-04
## PAY_0        0.738179821 0.014351951  51.434110 0.000000e+00
```

```
coefs = as.data.frame(glm.summary$coefficients[-1,1:2])
names(coefs)[2] = "se"
coefs$vars = rownames(coefs)
ggplot(coefs, aes(vars, Estimate)) + geom_errorbar(aes(ymin=Estimate
- 1.96*se, ymax=Estimate + 1.96*se), lwd=1, colour="red", width=0) +
geom_errorbar(aes(ymin=Estimate - se, ymax=Estimate + se), lwd=1.5,
colour="blue", width=0) +
geom_point(size=2, pch=21, fill="yellow")
```



In this graph, yellow dots represent coefficients. Education's coefficient is negative as well as SEX because their yellow dots are less than zero. This means that these coefficients have a negative relationship with the dependent variable. Red dots represent the confidence intervals error using 1.96 estimation point times standard error. For instance, there are red lines above and below EDUCATION, PAY_0, and SEX's yellow dots. Blue dots represent this graph's estimations and standard errors. As the red lines, there are blue lines inside the red lines.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0,data
=CC, family=binomial(link = "probit"))
glm.probs=predict(glm.fits,CC, type="response")
glm.pred=rep("notDefault", nrow(CC))
glm.pred[glm.probs>.5]="Default"
glm.pred = ifelse(glm.pred == "notDefault", 0, 1)
table(glm.pred,CC$default.payment.next.month)
```

```
##
## glm.pred     0     1
##        0 23014  5793
##        1   350   843
```

```
mean(glm.pred==CC$default.payment.next.month)
```

```
## [1] 0.7952333
```

In this probit regression, its mean is 0.7952 as well as its accuracy rate. The best regression's accuracy rate is 0.81 which is higher than this probit regression. In other words, the best regression is a better model.

Extra Credit: KNN

```
library(class)

set.seed(1)

trainData = sample_frac(CC, size = .8)
testData = setdiff(CC, trainData)


trainDataX = trainData[, c("AGE", "EDUCATION", "SEX", "PAY_0")]


testDataX = testData[, c("AGE", "EDUCATION", "SEX", "PAY_0")]


trainDataY = trainData[, "default.payment.next.month"]
```

```
knn.pred=knn(trainDataX, testDataX, trainDataY, k = 30)



table(knn.pred,testData$default.payment.next.month)
```

```
##
## knn.pred    0    1
##        0 4464  944
##        1  172  420
```

```
(4464+420)/(4464+944+172+420)
```

```
## [1] 0.814
```

```
knnBestOutcome = 0.814
```

In this KNN, like the probit regression, the best logistic regression's accuracy rate is 0.8109, and it is higher than the first test, 0.8058333 Hence, the best logistic regression explains this data better than this KNN. However, when k = 5, the accuracy rate is

0.8088333 When k increases, the accuracy rate increases. When k = 10, the accuracy rate is 0.8145, but when k = 30, the accuracy rate is 0.814. It is higher than the best logistic regression's accuracy rate.

Ridge

```
library(tree)
library(ggplot2)

set.seed(1)
train = CC %>% sample_frac(0.7)
test = CC %>% setdiff(train)


x_train = model.matrix(default.payment.next.month~., train)[,-1]
x_test = model.matrix(default.payment.next.month~., test)[,-1]
y_train = train$default.payment.next.month
y_test = test$default.payment.next.month
```

```
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

```
grid = 10^seq(10, -2, length = 100)
ridge_mod = glmnet(x_train, y_train, alpha = 0, lambda = grid)
cv.out = cv.glmnet(x_train, y_train, alpha = 0)


bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 0.01343141
```

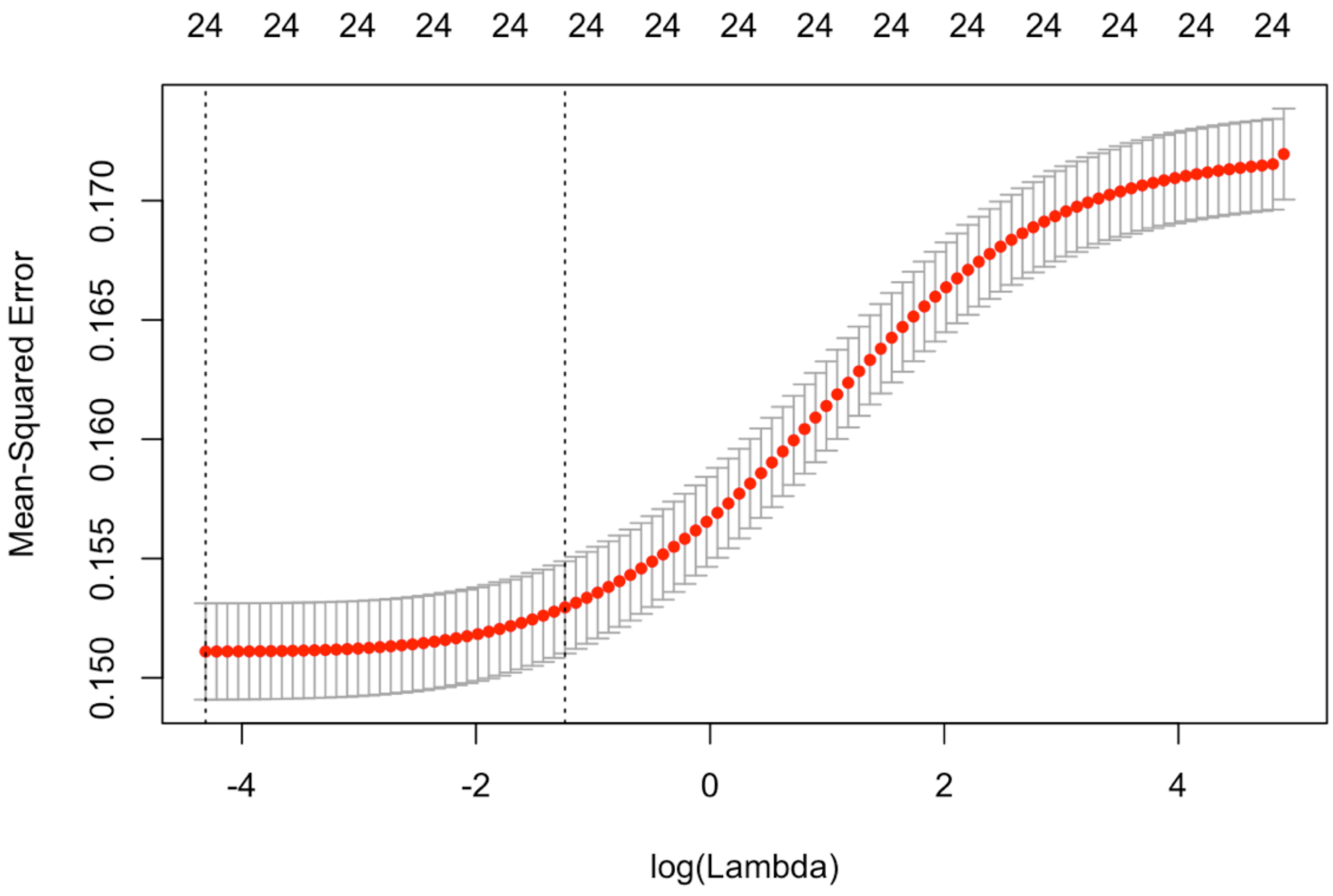
The lambda with the lowest error value is 0.01343141. Now, this is the best lambda.

```
cv.out$lambda.1se
```

```
## [1] 0.289371
```

0.289371 is the lambda with the 1 standard error from the best lambda.

```
plot(cv.out)
```

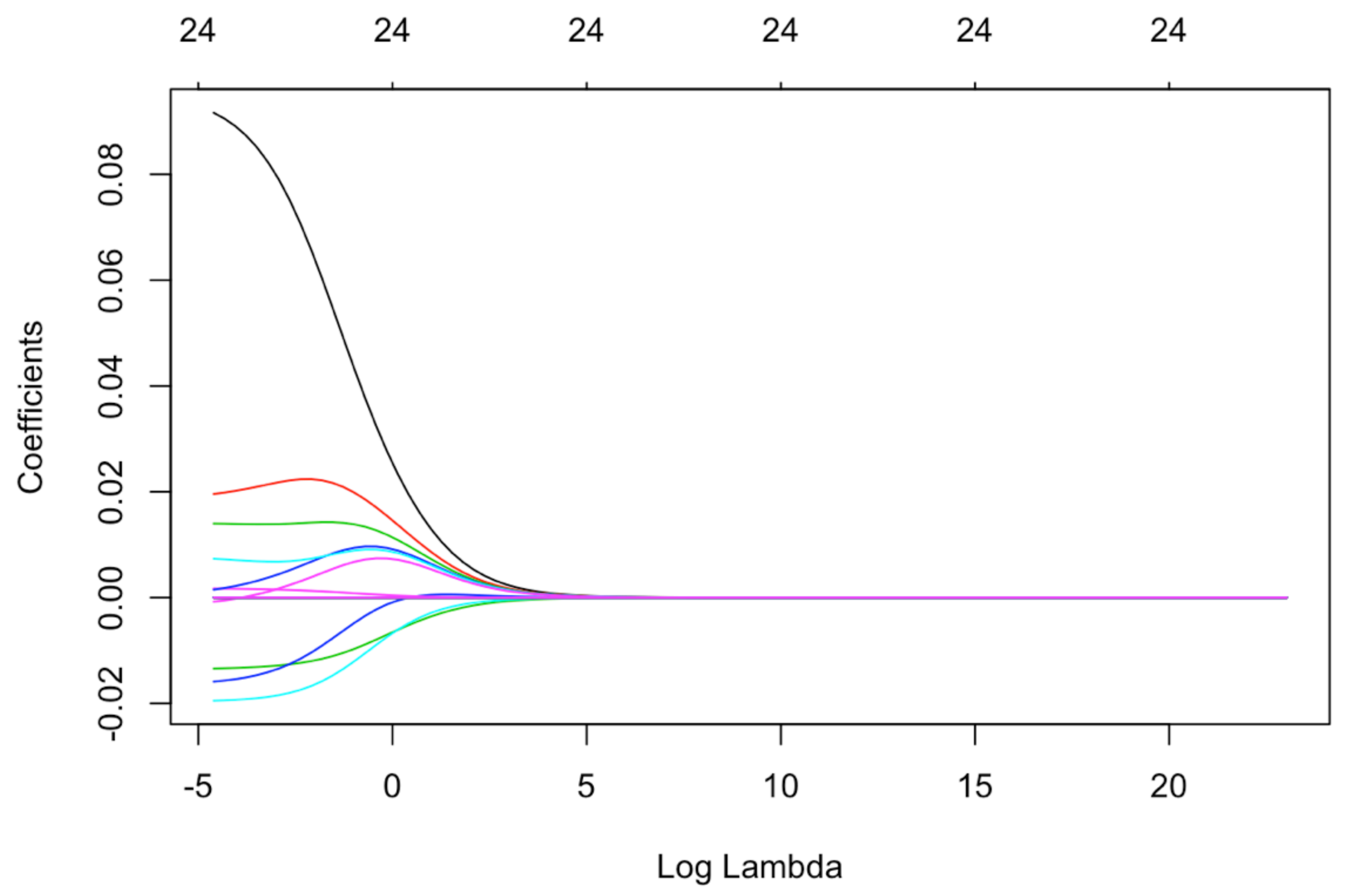


This is based on 1 standard error of the best lambda. The plot is log lambda against MSE. As the log lambda goes up, MSE goes up.

```
predict(ridge_mod, type = "coefficients", s = bestlam)
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  2.921553e-01
## ID           2.959644e-07
## LIMIT_BAL   -9.595318e-08
## SEX         -1.335495e-02
## EDUCATION   -1.566980e-02
## MARRIAGE    -1.941737e-02
## AGE          1.696724e-03
## PAY_0        9.045529e-02
## PAY_2        1.987068e-02
## PAY_3        1.395748e-02
## PAY_4        1.828479e-03
## PAY_5        7.225524e-03
## PAY_6       -5.156990e-04
## BILL_AMT1   -3.515845e-07
## BILL_AMT2   -6.187269e-08
## BILL_AMT3   -6.020680e-08
## BILL_AMT4   -7.390860e-08
## BILL_AMT5    7.375946e-08
## BILL_AMT6    3.251499e-08
## PAY_AMT1    -5.046970e-07
## PAY_AMT2    -1.591877e-07
## PAY_AMT3    -1.035154e-07
## PAY_AMT4    -2.623182e-07
## PAY_AMT5    -2.756453e-07
## PAY_AMT6    -2.124381e-07
```

```
plot(ridge_mod, xvar = "lambda")
```



Each of the lines represents the coefficients and everything goes to 0 as we increase the log lambda.

```
ridge_prob = predict(ridge_mod, s = bestlam, newx = x_test, type = "
response")
ridge_pred = ifelse(ridge_prob > 0.5, 1, 0)
table(ridge_pred, y_test)
```

```
##           y_test
## ridge_pred    0    1
##          0 6917 1744
##          1   80  259
```

```
(6917+259)/(6917+1744+80+259)
```

```
## [1] 0.7973333
```

```
ridgePrediction = 0.7973333
```

This is the ridge's prediction which is 0.7973333. As the ridge's prediction is compared to the best model's prediction, this is lower than the best model. Hence, this prediction will not preferable to analyze the report.

Lasso

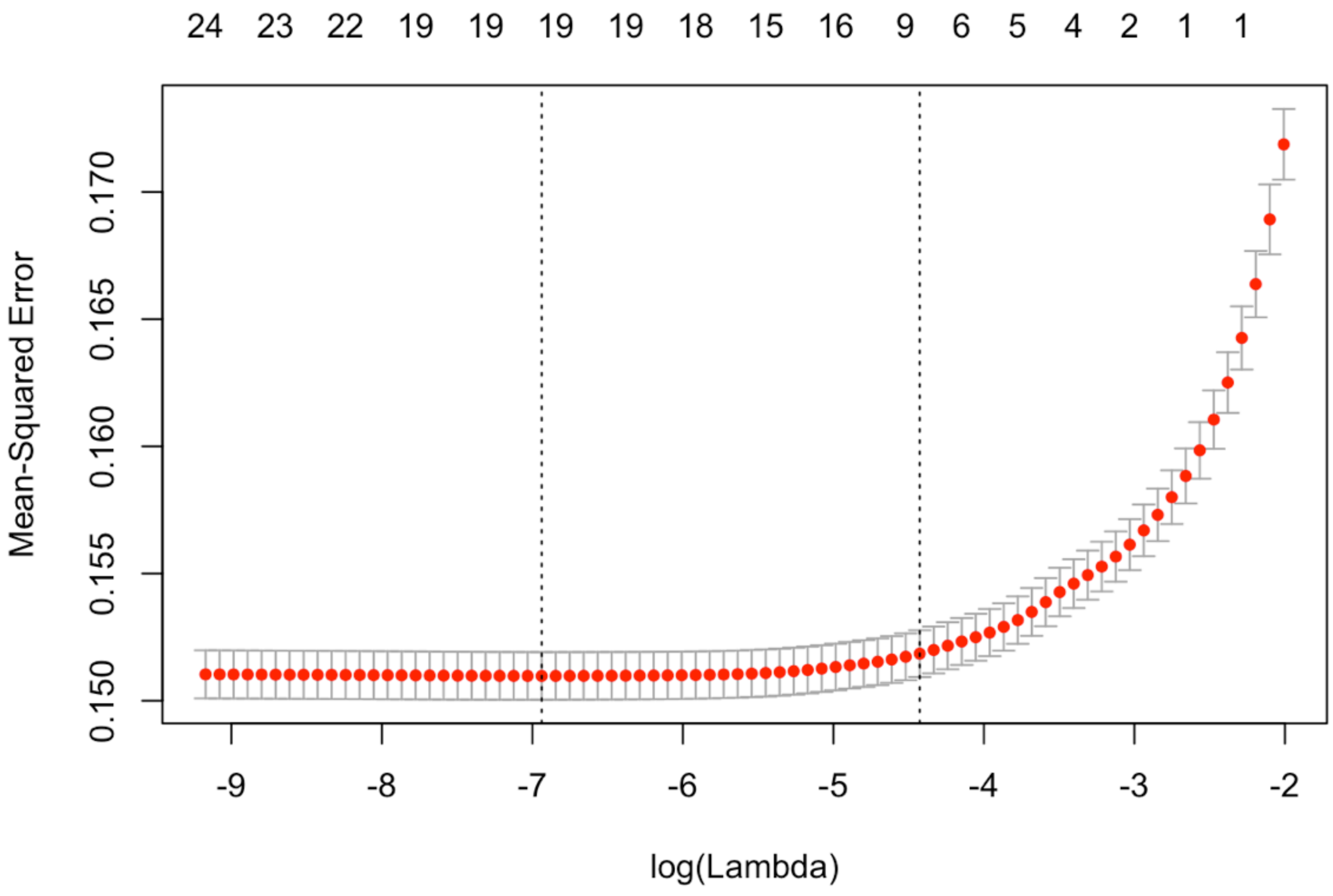
```
set.seed(1)
cv.out = cv.glmnet(x_train, y_train, alpha = 1)
```

```
cv.out$lambda.1se
```

```
## [1] 0.01195684
```

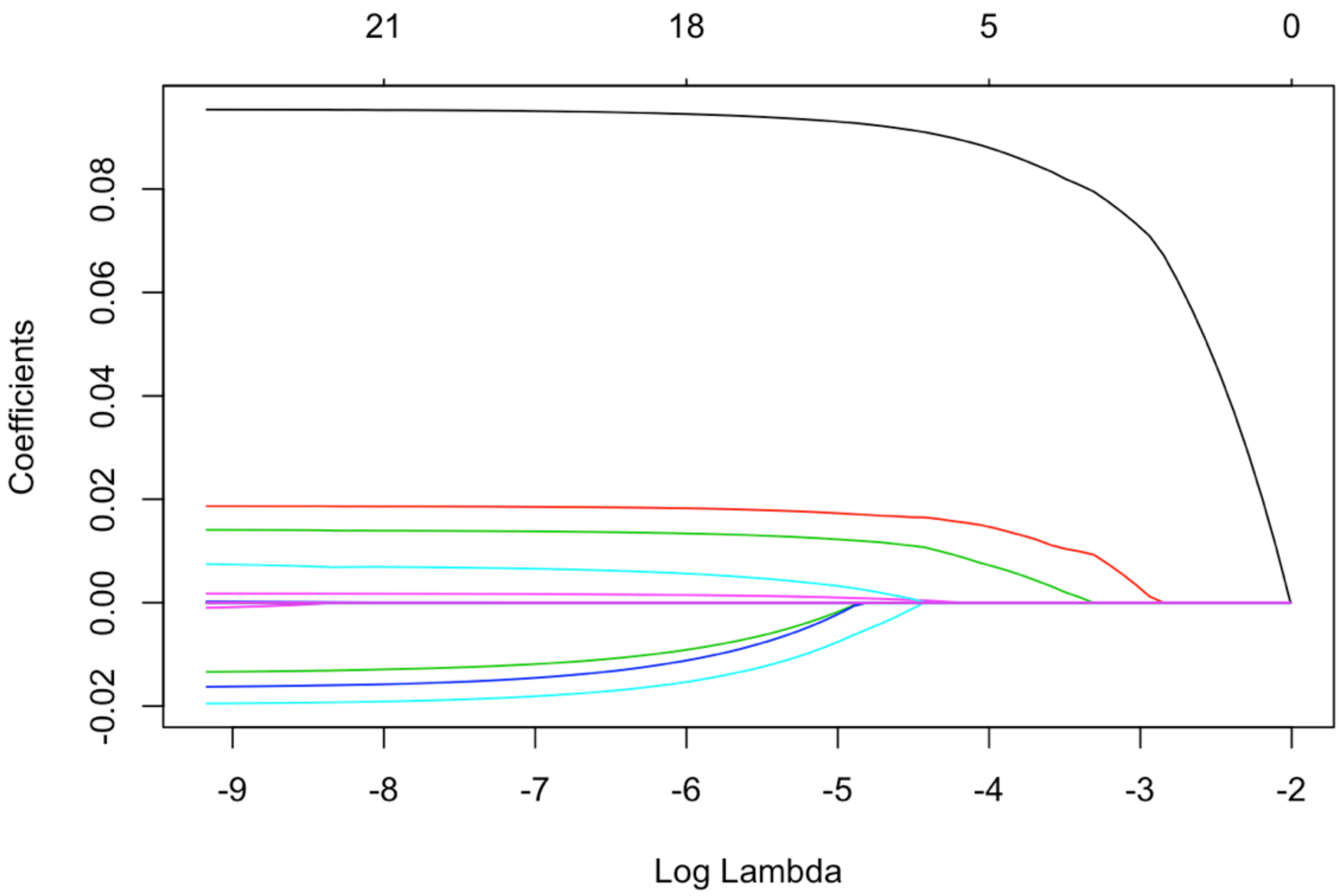
This is the lambda with the 1 standard error from the best lambda.

```
plot(cv.out)
```



This is based on 1 standard error of the best lambda. The plot is the log lambda against MSE. As the log lambda goes up, MSE goes up. In this graph, it clearly shows that the red circle line is increasing as the log lambda is increasing.

```
lasso_mod = glmnet(x_train, y_train, alpha = 1)
plot(lasso_mod, xvar = "lambda")
```



In this graph, it tells about the coefficients against the log lambda. It shows that every coefficients meets at between 0 when the log lambda increases.

```
bestlam = cv.out$lambda.1se

(coef_pred = predict(lasso_mod, s = bestlam, type = "coefficients"))
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
##                           1
## (Intercept)  2.370629e-01
## ID           .
## LIMIT_BAL   -5.962229e-08
## SEX          .
## EDUCATION    .
## MARRIAGE     .
## AGE          4.329352e-04
## PAY_0        9.099017e-02
## PAY_2        1.648254e-02
## PAY_3        1.070042e-02
## PAY_4        .
## PAY_5        1.062687e-04
## PAY_6        .
## BILL_AMT1   -2.967308e-07
## BILL_AMT2    .
## BILL_AMT3    .
## BILL_AMT4    .
## BILL_AMT5    .
## BILL_AMT6    .
## PAY_AMT1    -2.190115e-07
## PAY_AMT2     .
## PAY_AMT3     .
## PAY_AMT4     .
## PAY_AMT5     .
## PAY_AMT6     .
```

These are the coefficient correspondent with the best lambda.

```
lasso_prob = predict(lasso_mod, s = bestlam, newx = x_test, type = "
response")
lasso_pred = ifelse(lasso_prob > 0.5, 1, 0)

table(lasso_pred, y_test)
```

```
##             y_test
## lasso_pred    0    1
##          0 6962 1902
##          1   35  101
```

```
(6962+101)/(6962+1902+35+101)
```

```
## [1] 0.7847778
```

```
lassoPrediction = 0.7847778
```

This is my lasso prediction, 0.7847778, that now, I will compare this to the logistic regression.

```
glm.fits=glm(default.payment.next.month~AGE+EDUCATION+SEX+PAY_0,data
=train, family=binomial)

logOddsProb = predict(glm.fits, test, type = "response")
logOddsPred = ifelse(logOddsProb > 0.5, 1, 0)
table(logOddsPred, y_test)
```

```
##            y_test
## logOddsPred    0    1
##           0 6809 1550
##           1  188  453
```

```
(453 + 6809)/(6809 + 1550 + 188 + 453)
```

```
## [1] 0.8068889
```

```
logOddPrediction = 0.8068889
```

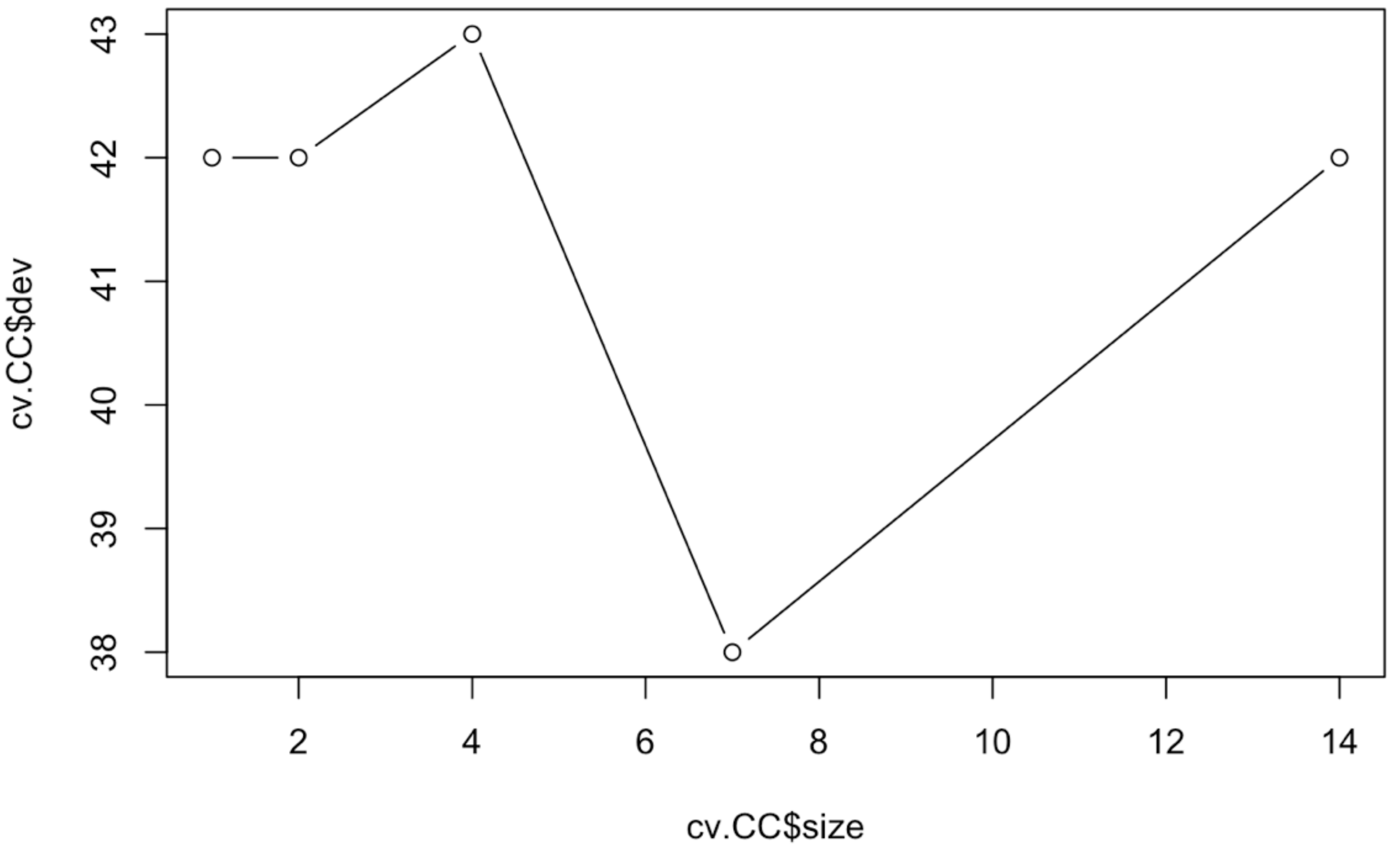
When I compare the lasso to the logistic regressions, the lasso is less than the logistic regressions. In addition to that, the lasso is greater than the ridge, so the lasso is preferable to further analyze the data.

Decision Tree

```
set.seed(2)

train = CC %>% sample_n(200)
test = CC %>% setdiff(train)


tree_CC = tree(factor(default.payment.next.month) ~ ., train)


set.seed(3)
cv.CC = cv.tree(tree_CC, FUN = prune.misclass)
```

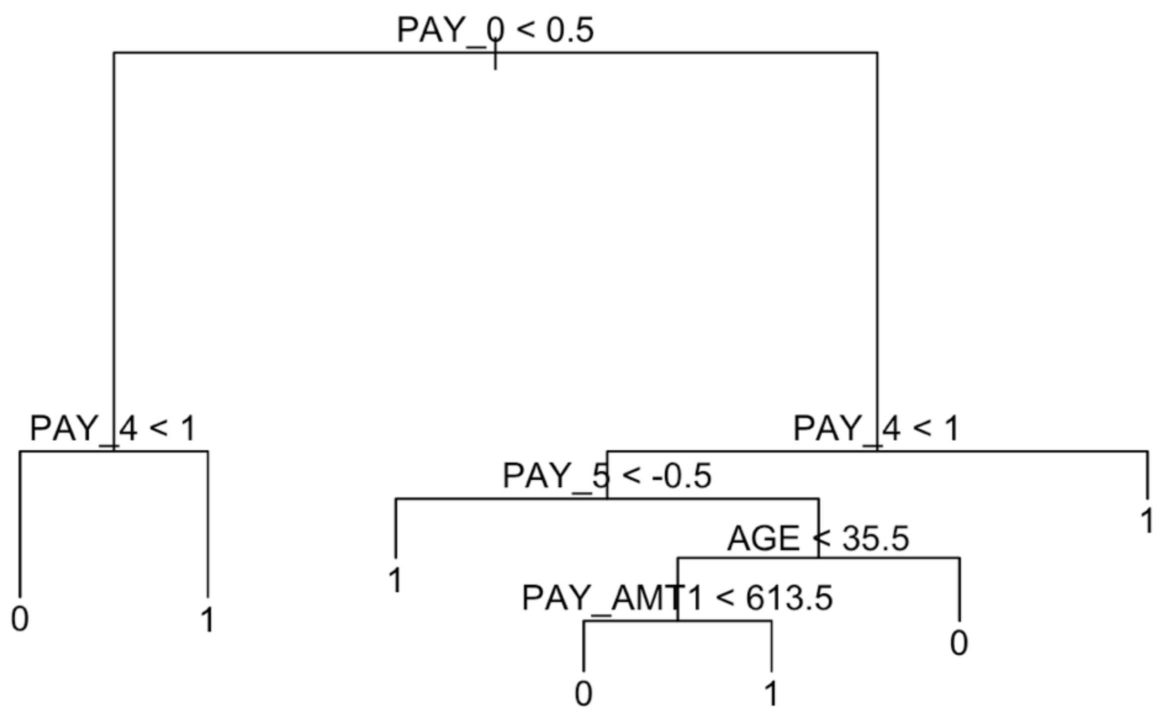
This is a tree that will be used to predict the data.

```
plot(cv.CC$size, cv.CC$dev, type = "b")
```



My best sign is 7.0 which means this has the lowest error.

```
prune_CC = prune.misclass(tree_CC, best = 7)
plot(prune_CC)
text(prune_CC, pretty = 0)
```



The tree is very simple. PAY_0 is repayment status in September 2005. If PAY_0 is less than 0.5, it is likely that a person is not paying off his or her debts when PAY_4 is less than 1. If PAY_0 is greater than 0.5, it is likely that a person is paying off his or her debts when other tree matches up to be default on credit card debts.

```
tree_pred = predict(prune_CC, test, type = "class")
table(tree_pred, test$default.payment.next.month)
```

```
##
## tree_pred     0     1
##         0 19926  3702
##         1  3277  2895
```

```
(19926 + 2895)/ (19926 + 3702 + 3277 + 2895)
```

```
## [1] 0.7658054
```

```
treePrediction = 0.7658054
```

The tree prediction is 0.7658054. It is higher than the logistic regression, but it is not higher than KNN when k = 30.

Bagging

```
library(dplyr)
library(ggplot2)

set.seed(1)
CC_train = CC %>%
  sample_frac(.7)

CC_test = CC %>%
  setdiff(CC_train)
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)
bag.CC = randomForest(factor(default.payment.next.month) ~., data =
CC_train, mtry = ncol(CC_train)-1, importance = TRUE, ntree=500)
```

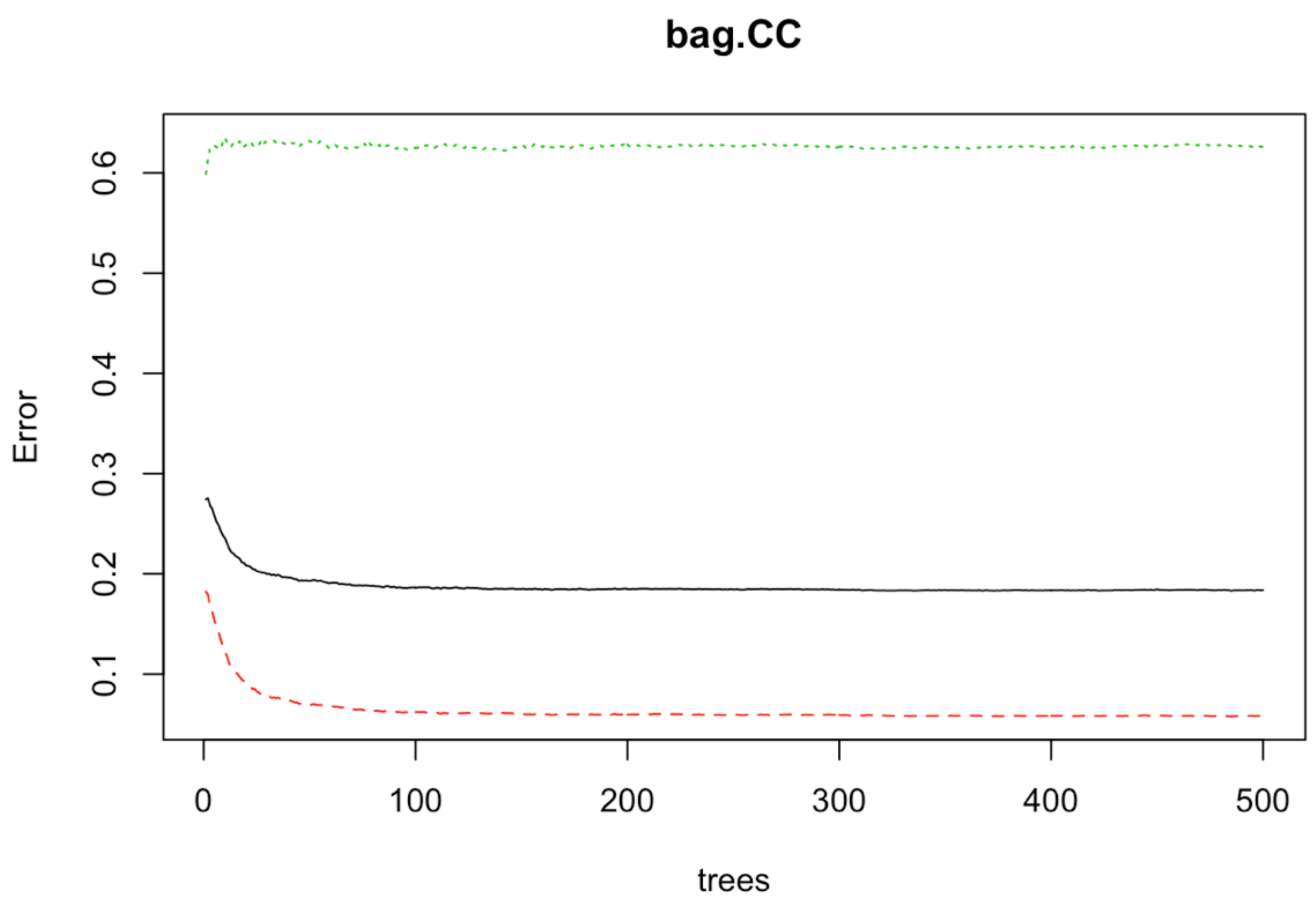
```
bag.CC
```

```
##
## Call:
##  randomForest(formula = factor(default.payment.next.month) ~ .,
data = CC_train, mtry = ncol(CC_train) - 1, importance = TRUE,
ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 24
##
##         OOB estimate of  error rate: 18.36%
## Confusion matrix:
##       0    1 class.error
## 0 15413  954  0.05828802
## 1  2902 1731  0.62637600
```

```
(921+2886)/(15425+921+2886+1768)
```

```
## [1] 0.1812857
```

```
bagOutcome = 0.1813


plot(bag.CC)
```



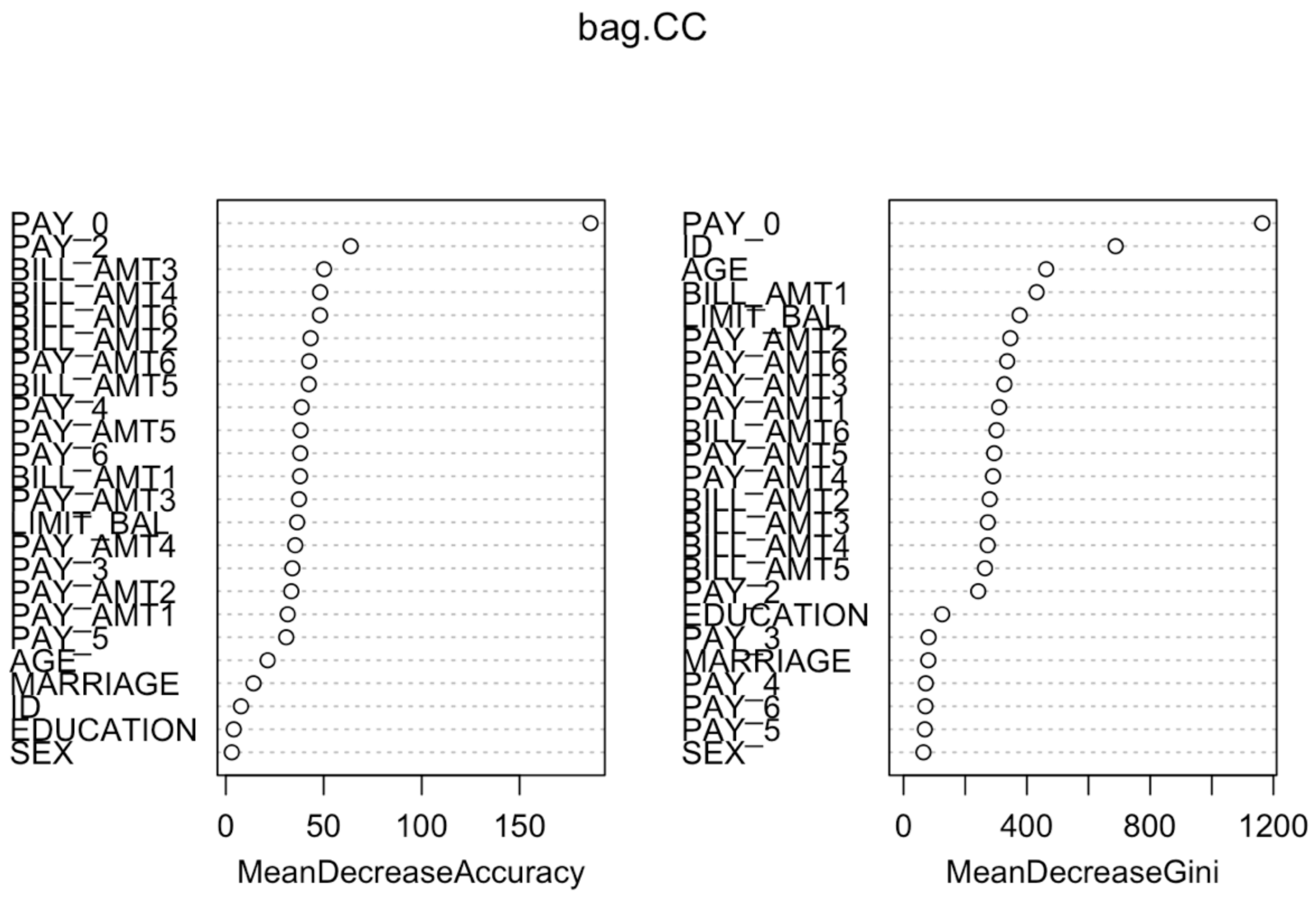
The error rate is 0.1813. Lasso’s error rate: 0.2152222. Ridge’s error rate: 0.2026667. This error rate is less than the Lasso and Ridge’s error rates.

```
importance(bag.CC)
```

```
##                     0          1 MeanDecreaseAccuracy MeanDecreas
eGini
## ID           6.751036  3.7420442             7.763453        687.
93697
## LIMIT_BAL   29.242794 22.0450113            36.435435        376.
69789
## SEX          3.439867  0.1622211             3.050621         64.
52262
## EDUCATION    4.948864 -0.5250499             4.049119        125.
24336
## MARRIAGE    18.128713 -4.6286185            14.168490         80.
24282
## AGE         24.123514  0.6913807            21.363182        462.
```

```
65101
## PAY_0     123.998794 114.8997401            186.166409       1163.
76264
## PAY_2      60.461269  -2.6601221             63.722161        242.
22901
## PAY_3      28.589579   8.2491930             33.983485         81.
13927
## PAY_4      36.978192  -5.8220827             38.694844         72.
08571
## PAY_5      26.387597   6.6459226             30.850995         68.
95827
## PAY_6      31.646482   6.0370411             37.994199         70.
98139
## BILL_AMT1  24.392491  22.9434815             37.908255        431.
59532
## BILL_AMT2  39.514740  -8.7252409             43.289142        279.
45227
## BILL_AMT3  46.895709 -13.6180273             50.155816        273.
38128
## BILL_AMT4  44.022908  -8.9846954             48.172726        273.
09882
## BILL_AMT5  36.348822  -3.9714953             42.371787        264.
13259
## BILL_AMT6  39.247154   0.6537762             48.111377        301.
40102
## PAY_AMT1   30.532856 -11.8350781             31.587296        310.
18070
## PAY_AMT2   29.824111   3.7878942             33.428105        346.
26033
## PAY_AMT3   31.379723   2.2613568             37.380317        327.
20417
## PAY_AMT4   28.621597   4.6013539             35.427767        290.
23865
## PAY_AMT5   32.517699   2.4675908             38.195627        293.
96354
## PAY_AMT6   38.726644   7.2503139             42.568295        336.
11887
```

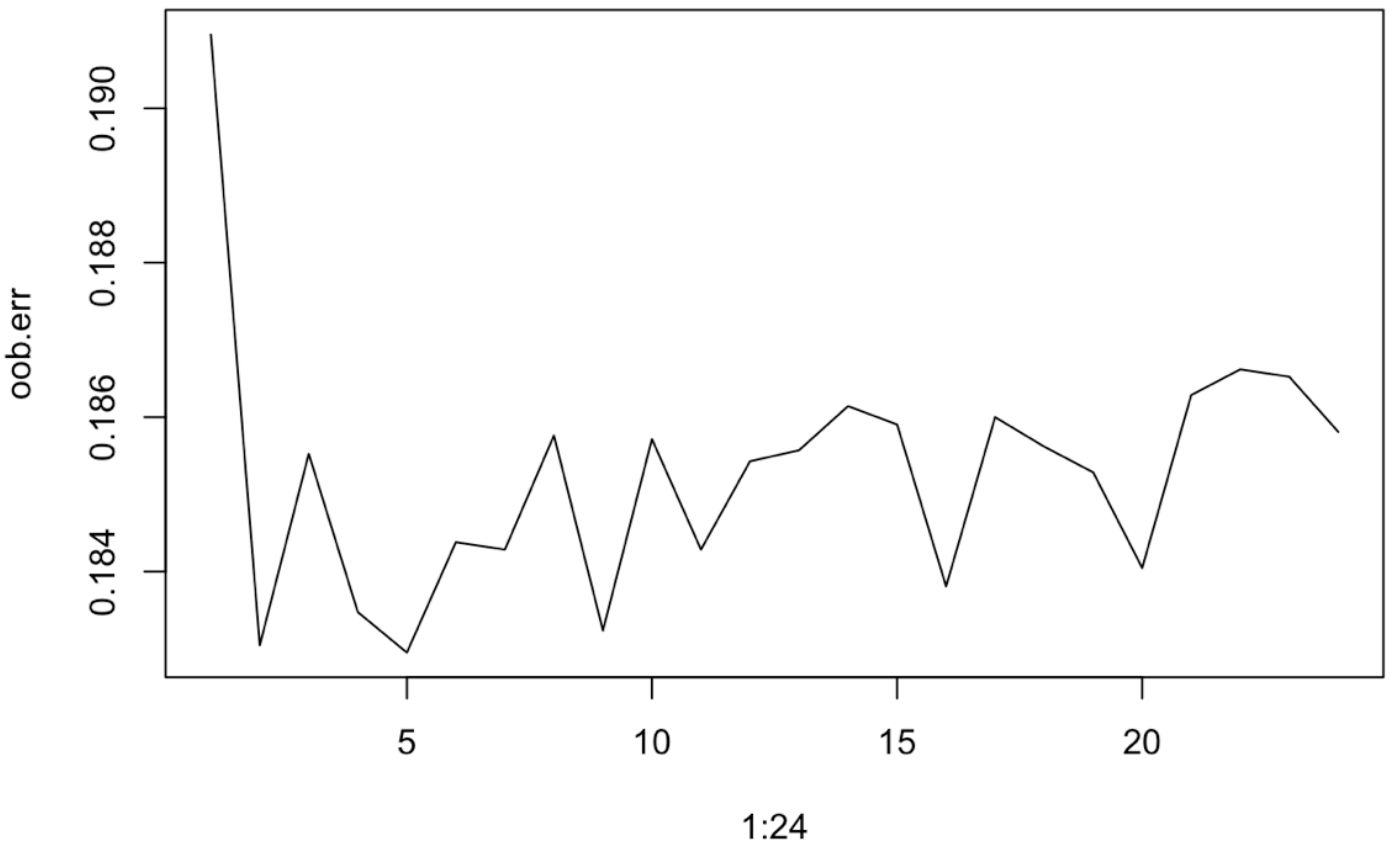
```
varImpPlot(bag.CC)
```



In this importance matrix, PAY_0 is the most important variable. PAY_0 is repayment status in Septembe in 2005.

Random Forest

```
set.seed(1)

oob.err<-double(24)
test.err<-double(24)


for(mtry in 1:24) {
  rf=randomForest(factor(default.payment.next.month) ~ . , data = CC
_train, mtry=mtry, ntree=150)
  oob.err[mtry] = rf$err.rate[150] #Error of all Trees fitted on tra
ining

  pred<-predict(rf,CC_test) #Predictions on Test Set for each Tree
  test.err[mtry]= with(CC_test, sum(pred != default.payment.next.mon
th)) # Error rate for classification

 # print(mtry)
}


plot(1:24, oob.err,type = "l" )
```



This is a plot of the out of bag error rate versus the number of variables considered that are each split of each tree. The lowest error rate is at 9.

```
set.seed(2)
rf.CC = randomForest(factor(default.payment.next.month) ~., data = C
C_train, mtry = 9, importance = TRUE, do.trace = 100)
```

```
## ntree      OOB      1      2
##   100:  18.58%  6.04% 62.90%
##   200:  18.43%  5.83% 62.94%
##   300:  18.38%  5.77% 62.94%
##   400:  18.34%  5.75% 62.83%
##   500:  18.32%  5.72% 62.83%
```

```
yhat.rf = predict(rf.CC, newdata = CC_test)

table(CC_test$default.payment.next.month, yhat.rf)
```

```
##     yhat.rf
##         0    1
##    0 6625  372
##    1 1266  737
```

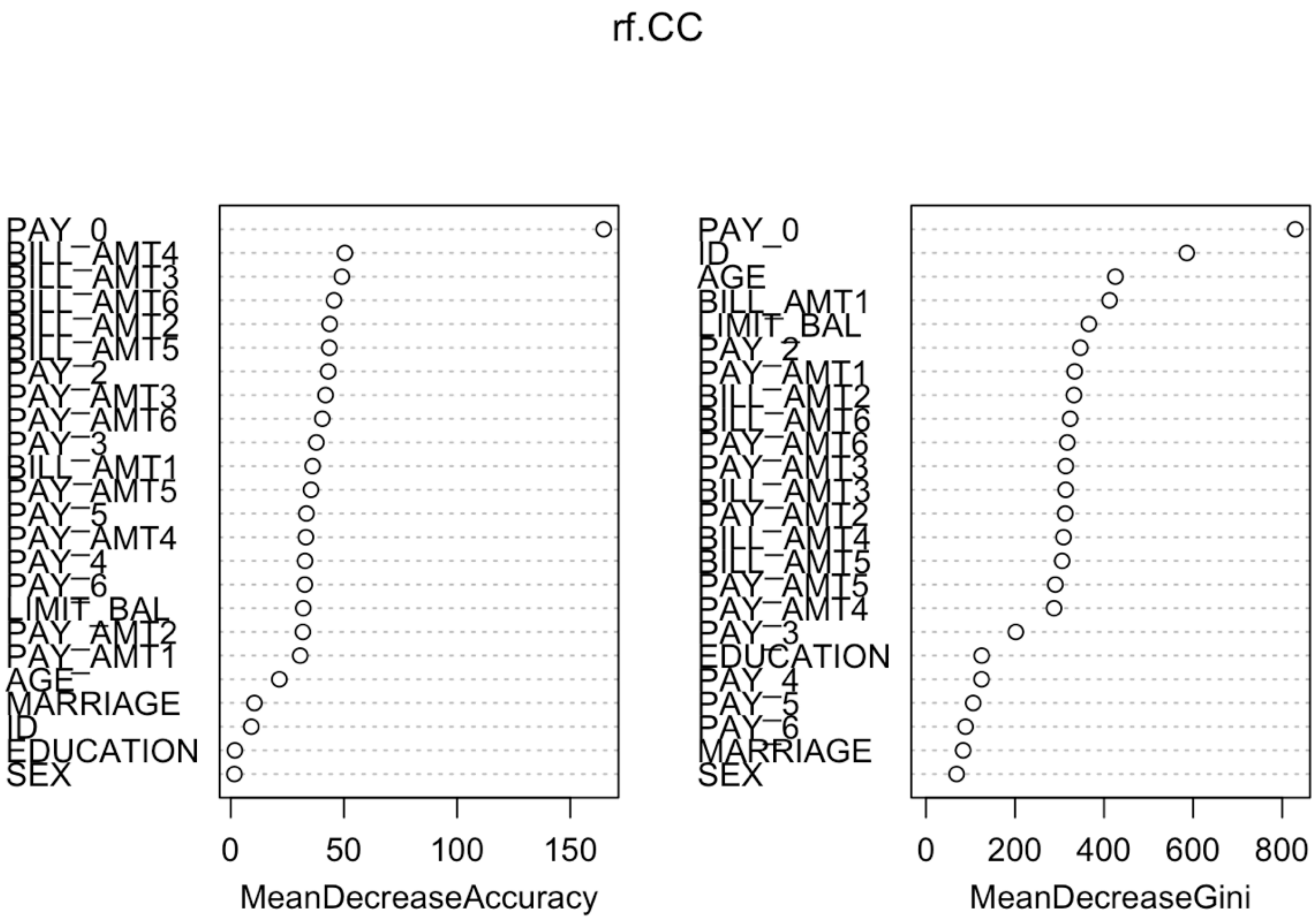
```
(375+1286)/(6643+375+1286+696)
```

```
## [1] 0.1845556
```

```
randomForestOutcome = 0.1845556
```

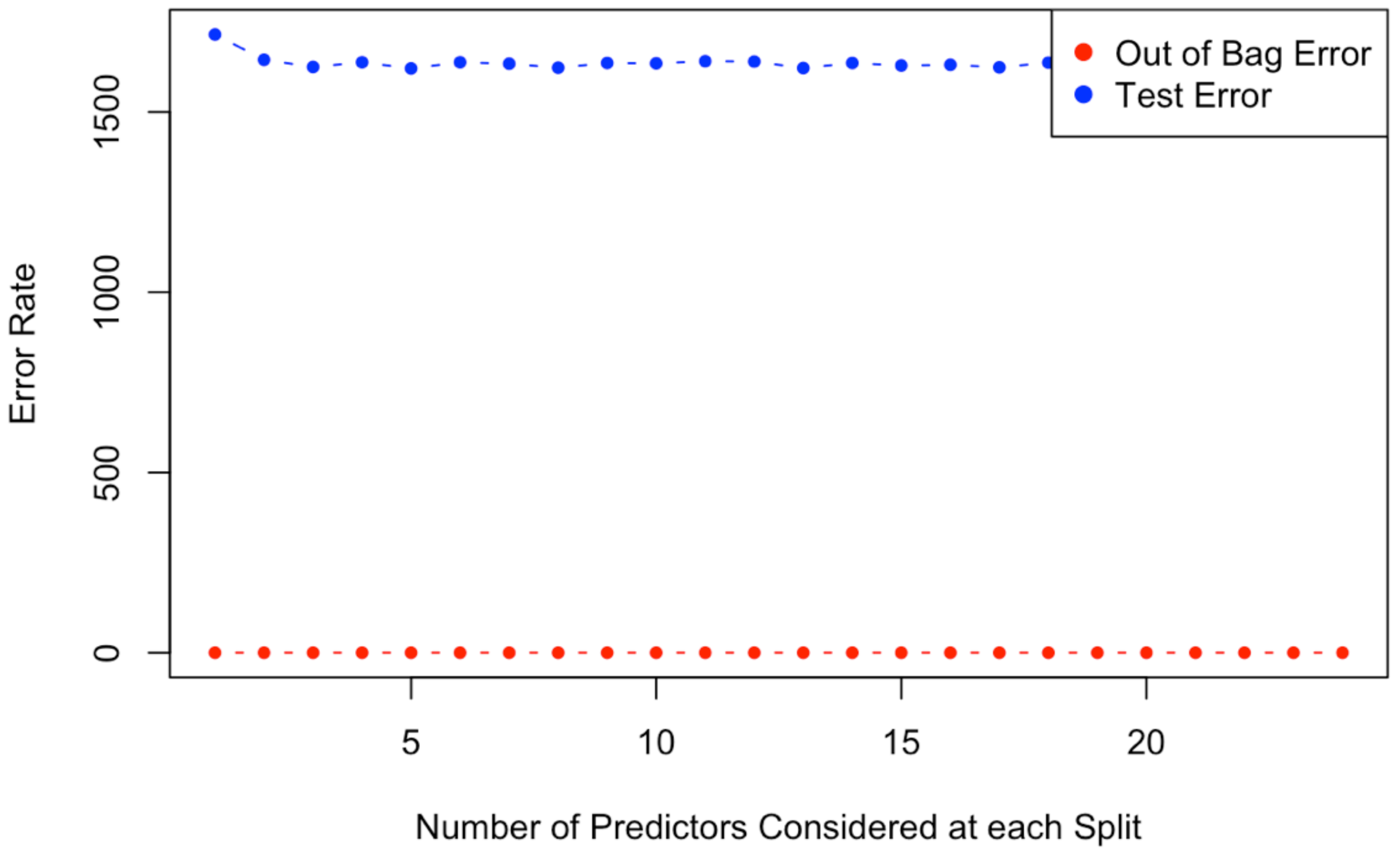
The error rate is 0.1845556. The bagging error rate is 0.1907619. The lasso's error rate: 0.2152222. The ridge's error rate: 0.2026667. Random Forest's error rate is the lowest error rate which is our best model.

```
varImpPlot(rf.CC)
```



In this importance matrix, PAY_0 is the most important variable. PAY_0 is repayment status in September 2005.

```
matplot(1:mtry , cbind(oob.err,test.err), pch=20 , col=c("red","blue
"),type="b", ylab="Error Rate",xlab="Number of Predictors Considered
at each Split")
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19,
col=c("red","blue"))
```



This graph is nicely plotted by Out of Bag Error and Test Error vs mtry. In addition to that, this tells us that two lines have a similar pattern.

Boosting

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(1)
CC_train = CC %>%
  sample_frac(.7)

CC_test = CC %>%
  setdiff(CC_train)
```

```
set.seed(1)

boost.CC = gbm(default.payment.next.month~.,
                   data = CC_train,
                   n.trees = 500)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
yhat.boost = predict(boost.CC,
                         newdata = CC_test,
                         n.trees = 500,  type= "response")




yhat.boost_class = ifelse(yhat.boost > 0.5, 1, 0)




table(yhat.boost_class, CC_test$default.payment.next.month)
```

```
##
## yhat.boost_class    0    1
##                0 6703 1337
##                1  294  666
```

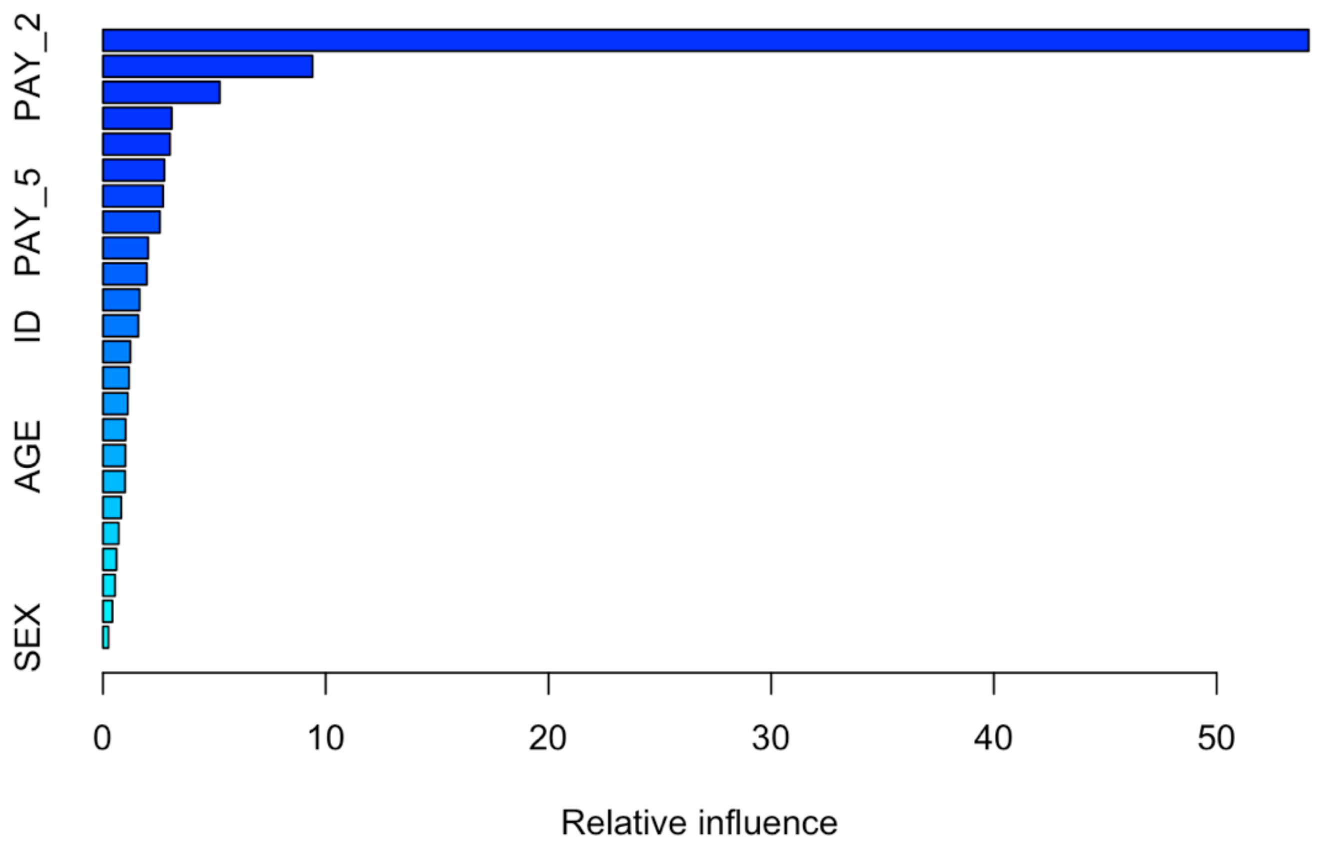
```
(1337 + 294)/(6703+1337+294+666)
```

```
## [1] 0.1812222
```

```
boostOutcome = 0.1812222
```

The error rate is 0.1812. The random forest's error rate is 0.1801111. The bagging error rate is 0.1907619. The lasso's error rate: 0.2152222. The ridge's error rate: 0.2026667. Random Forest's error rate is the lowest error rate. Therefore, the random forest's error rate is the lowest among the error rates.

```
summary(boost.CC)
```



	var	rel.inf
	<fctr>	<dbl>
PAY_0	PAY_0	54.1284131
PAY_2	PAY_2	9.4064922

PAY_3	PAY_3	5.2443437
LIMIT_BAL	LIMIT_BAL	3.0875884
PAY_AMT2	PAY_AMT2	3.0142926
PAY_4	PAY_4	2.7572086
BILL_AMT1	BILL_AMT1	2.6996745
PAY_5	PAY_5	2.5608754
PAY_AMT3	PAY_AMT3	2.0276179
PAY_AMT1	PAY_AMT1	1.9699074

1-10 of 24 rows Previous **1** 2 3 Next

In this importance matrix, PAY_0’s relative influence is 54.1284131. Therefore, this variable is the most important variable.

XGboost

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
Y_train <- as.matrix(CC_train[,"default.payment.next.month"])
X_train <- as.matrix(CC_train[!names(CC_train) %in% c("default.payme
nt.next.month")])
dtrain <- xgb.DMatrix(data = X_train, label = Y_train)

X_test <- as.matrix(CC_test[!names(CC_train) %in% c("default.payment
.next.month")])
```

```
set.seed(1)
CC.xgb = xgboost(data=dtrain,
                     max_depth=5,
                     eta = 0.1,
                     nrounds=40, # max number of boosting iterations
(trees)
                     lambda=0,
                     print_every_n = 10,
                     objective= "binary:logistic", verbose = FALSE)

yhat.xgb <- predict(CC.xgb,X_test)

yhat.xgb_class = ifelse(yhat.xgb > 0.5, 1, 0)

table(yhat.xgb_class, CC_test$default.payment.next.month)
```

```
##
## yhat.xgb_class    0    1
##              0 6707 1303
##              1  290  700
```

```
(1303+290)/(6707+1303+290+700)
```

```
## [1] 0.177
```

```
XGBoostOutcome = 0.177
```

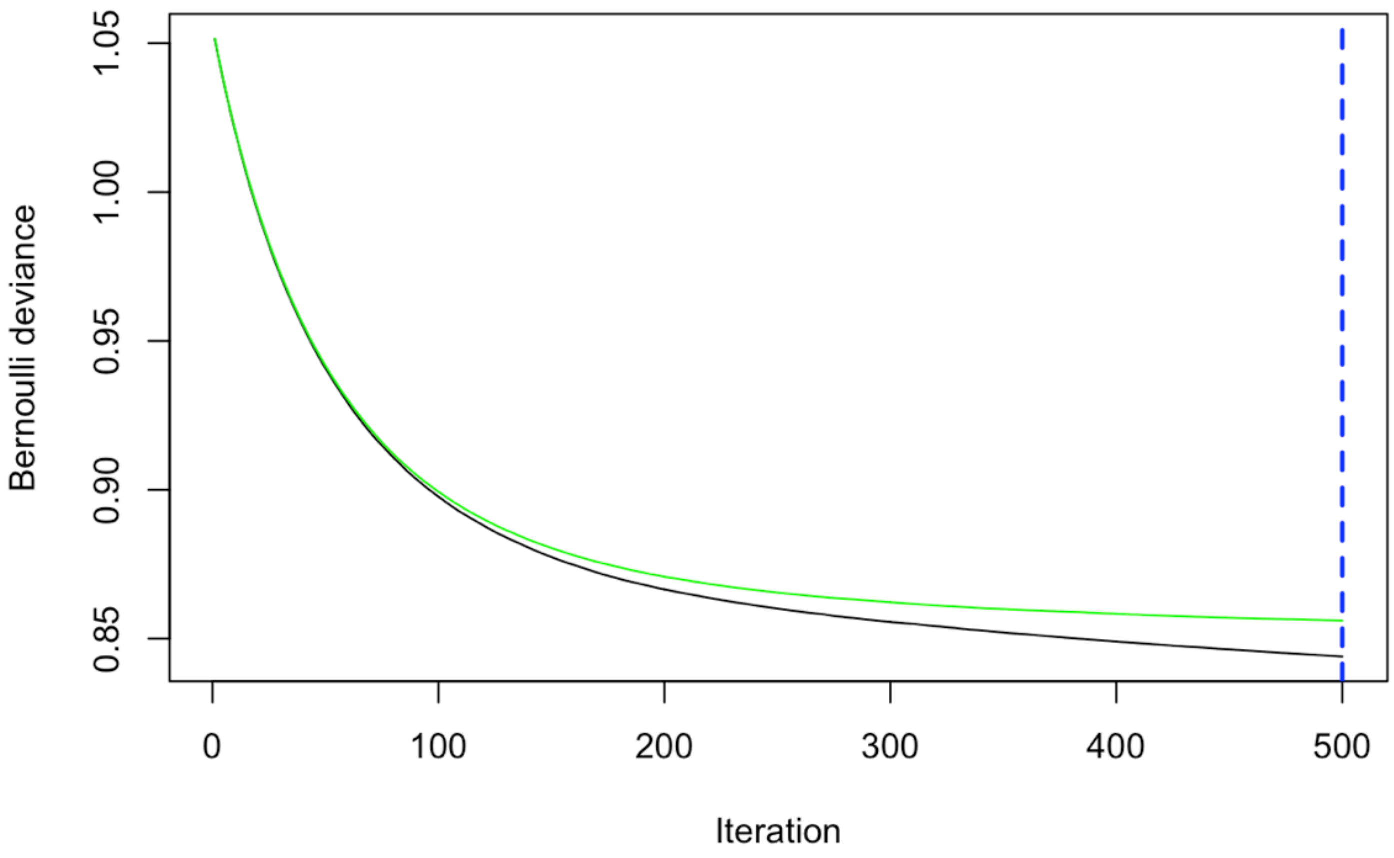
5(b): XG Boost's error rate is 0.177. The boost's error rate is 0.1812222. The bagging error rate is 0.1907619. The lasso's error rate: 0.2152222. The ridge's error rate: 0.2026667. The XG Boost's error rate is the lowest. This shows the XG Boost is the best model.

Extra Credit: Using grid search for the Boosting model

```
set.seed(1)

CC.boost.cv <- gbm(default.payment.next.month~., data = CC_train,
                      n.trees=500,
                      interaction.depth=4,
                      shrinkage = 0.01,
                      verbose=F,
                      cv.folds=5)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
bestTreeForPrediction <- gbm.perf(CC.boost.cv)
```



```
bestTreeForPrediction
```

```
## [1] 500
```

```
min(CC.boost.cv$cv.error)
```

```
## [1] 0.8560555
```

```
which.min(CC.boost.cv$cv.error)
```

```
## [1] 500
```

```
set.seed(1)
CC_train = CC %>%
  sample_frac(.7)

CC_test = CC %>%
  setdiff(CC_train)

set.seed(1)
yhat.boost <- gbm(default.payment.next.month~., data = CC_train,
                    n.trees=500,
                    interaction.depth=4,
                    shrinkage = 0.01,
                    verbose=F)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
yhat.boost <- predict(yhat.boost, CC_test, n.trees = 500)

yhat.boost_class = ifelse(yhat.boost > 0.5, 1, 0)

table(yhat.boost_class, CC_test$default.payment.next.month)
```

```
##
## yhat.boost_class    0    1
##                 0 6809 1483
##                 1  188  520
```

```
(1483+188)/(6809+1483+188+520)
```

```
## [1] 0.1856667
```

After tuning the number of trees, the error rate is 0.1856667, but this is higher than the XG Boost error rate. Therefore, the best model is XG Boost.

Neural Network

Since TAs, Cran and Jonathan, and I couldn’t solve this error message:

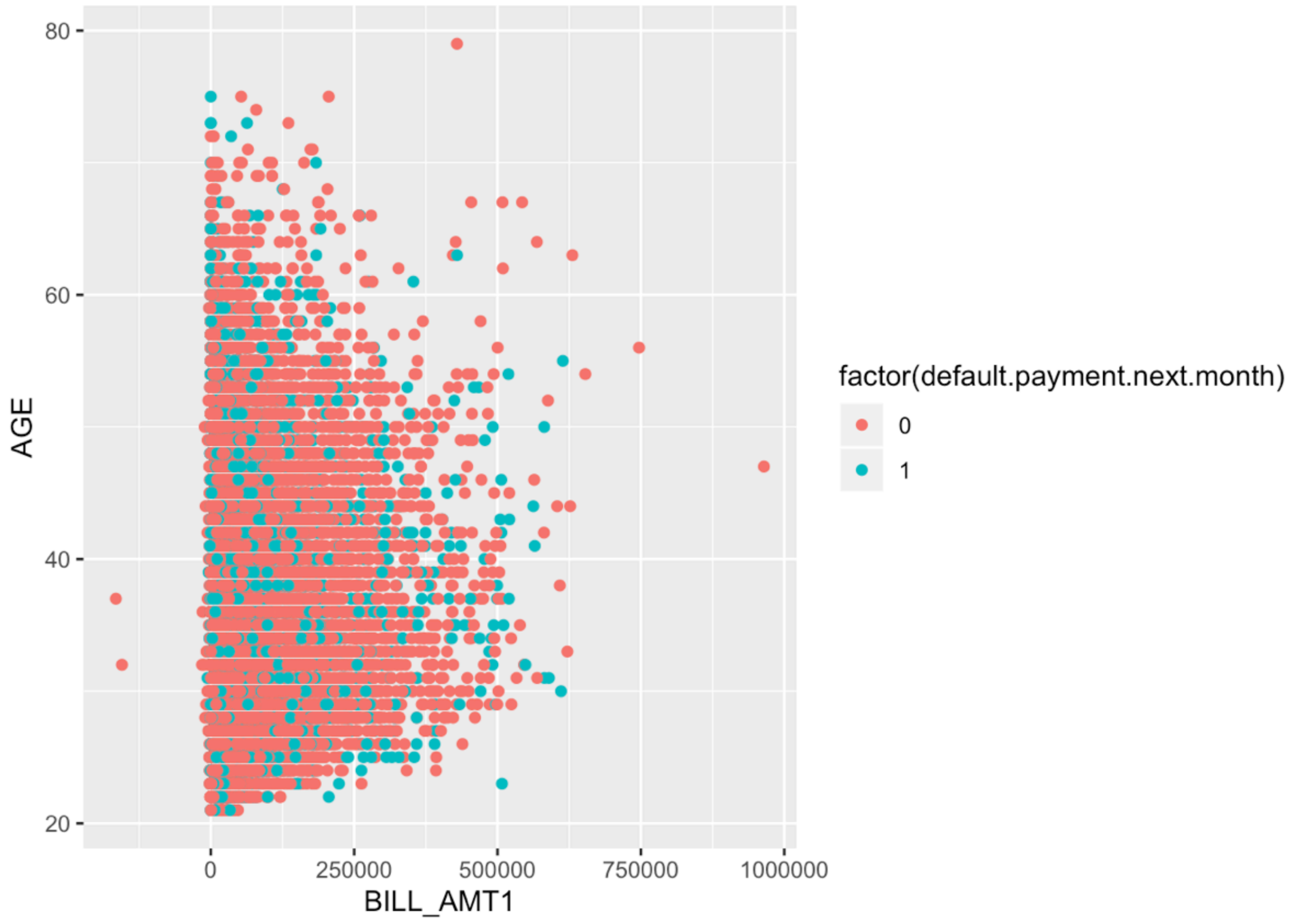
ERROR: Could not find a version that satisfies the requirement tensorflow==2.0.0 (from versions: none) ERROR: No matching distribution found for tensorflow==2.0.0 Error: Error installing package(s): ‘tensorflow==2.0.0’, ‘keras’, ‘tensorflow-hub’, ‘h5py’, ‘pyyaml’, ‘requests’, ‘Pillow’, ‘scipy’,

I did nerual network on RStudio Cloud in order to perfrom nerual network.

Extra Credit: SVM

```
library(dplyr)
library(ggplot2)
library(e1071)
```

```
set.seed(1)

ggplot(CC, aes( BILL_AMT1, AGE, colour = factor(default.payment.next
.month))) +
  geom_point()
```



By looking at this graph, this data set does not perform well using support vector machine because the red and blue dots are randomly placed in this graph. Also, svmfits do not plot against train sets. Hence, support vector machine is not useful in this case.

```
##```{r}

train = CC %>%
  sample_frac(0.7)

test = CC %>%
  setdiff(train)

svmfit = svm(default.payment.next.month~., data = train, kernel = "r
adial",  gamma = 1, cost = 1)

plot(svmfit, train)



##```{r}
svmfit = svm(default.payment.next.month~., data = train, kernel = "r
adial", gamma = 1, cost = 1e5)
plot(svmfit, train)



##```{r}
set.seed(1)
tune.out = tune(svm, default.payment.next.month~., data = train, ker
nel = "radial",
                ranges = list(cost = c(0.1,1,10,100,1000), gamma = c
(0.5,1,2,3,4)))


bestmod = tune.out$best.model


plot(bestmod, train)

table(true = test$default.payment.next.month, pred = predict(tune.ou
t$best.model, newdata = test))
```

Comparing All Models

As predicting all of the models, for the ridge, its error rate is 0.2026667, and for the lasso, its prediction is not that great because its error rate is higher than the ridge which is 0.2152. The tree prediction is 0.7658, and its error rate is 0.2341946. The bagging's error rate is 0.1876, and the random forest's error rate is 0.1845556. For the boosting model, the error rate was 0.1812222, but after tuning the parameter, the error rate becomes 0.1856667. The XG boost's error rate is 0.177. Therefore, the XG boost model has the lowest error rate that predicts our data accurately. The XG boost model is used to analyze this data set.

Conclusion

The report is based on using complex machine learning methods to show the relationship between the dependent and the independent variables. The dependent variable is "default.payment.next.month," and there are 24 dependent variables. In this report, it represents how the independent variables affect the dependent variable by looking at the histograms, box plot, and density graphs. It clearly tells us that there are more women than men and very important coefficients that affect the dependent variable by looking at the summary statistics. There are three stars next to coefficients if the coefficients are very important. These coefficients are mostly affecting the dependent variable, and there are coefficients having positive and negative relationships with the dependent variable. In addition to that, women have more debts that they did not pay off than men. Most people are college graduates, and some people have graduate degrees and high school degrees only. Most of the people's age range is between 20 to 40. In other words, the majority of the people are younger than 50 years old.

After viewing the summary statistics, the logistic and probit classifications, K-Nearest Neighborhood, ridge, lasso, tree, bagging, random forest, boost, XG boost, support vector machine, and neural network models are used to predict the testing sets and calculate the accuracy rates and the error rates in order to interpret the data set to compare these models. For the logistic regressions, the training set is used to fit the model and examine how it predicts the data set using the testing set. For the probit classifications, this model is similar to the logistic regression, but the difference is in the nonlinear function that fits in the data set. The best logistic regression model's accuracy rate is higher than the probit classification model. These two models are using a non-linear parametric model as well as K-Nearest Neighborhood. For the K-Nearest Neighborhood, this specifies K which represents the number of nearest neighbors that considers each observation. For the ridge and the lasso models, these are shrinkage methods because these shrink the estimated coefficients toward zero using a hyperparameter: lambda. For the tree model, the best size is seven that this is the lowest error, and this will be used to prune the tree in order to find its accuracy rate. For bagging, a bootstrap is used to generate B that separates the training sets and trains a tree on each of them. This method and the random forest have three parameters: mtry, ntree, and nodesize. Mtry is the number of variables at each split. Ntree is the number of trees, and nodesize is the number of observations in the terminal nodes. Also, for the random forest, out of bag errors and test errors are used to analyze the data set. For boosting, this is a tree-based model and fits the tree on residuals. Similar to the ridge and the lasso, the XG boost model is putting the data set into a matrix format. Since the support vector machine is using a non-linear kernel, and the radial kernel is used to analyze the data set. this model is tunned to predict the data set better. However, the data set does not perform well with the support vector machine model, so this model is not used to analyze this data. After running neural network, this model's error rate is very high that this model will not be used to analyze this data set. As a result, the XG boost model has the lowest error rate that this predicts the data set accurately. This tells how the independent and the dependent variables correlate to each other.

The Patterns of Paying Debts of Credit Card Clients

Neural Network

```
CC=read.csv("UCI_Credit_Card.csv",header=T)

attach(CC)

library(tensorflow)
library(keras)
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

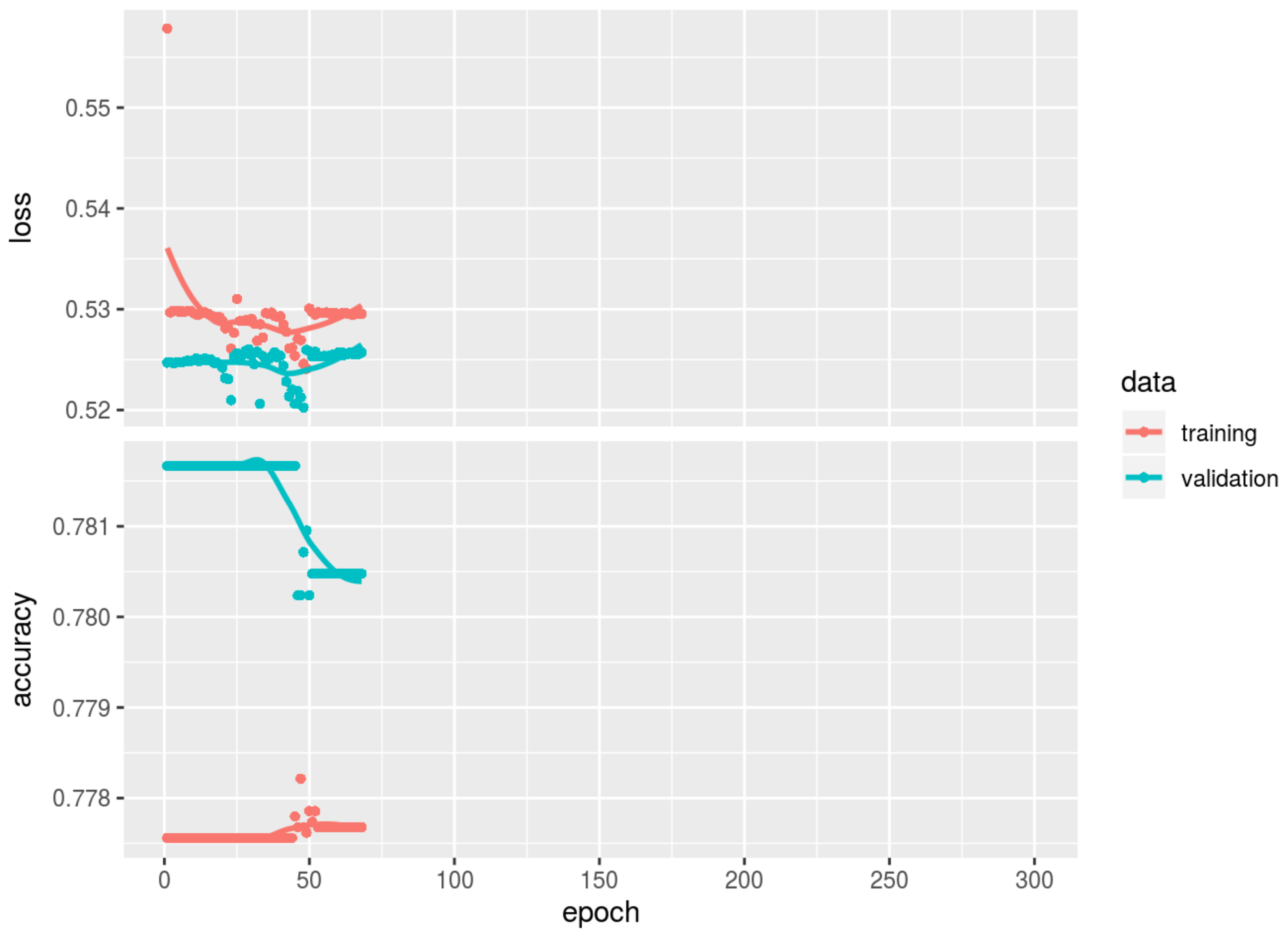
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
CC$default.payment.next.month_high <- ifelse(CC$default.payment.next
.month > mean(CC$default.payment.next.month), 1, 0)



set.seed(2)


CC_train = CC %>%
  sample_frac(.7)

CC_test = CC %>%
  setdiff(CC_train)
```

```
set.seed(1)
train_labels <- to_categorical(CC_train[,"default.payment.next.month
_high"], 2)


train_data <- as.matrix(CC_train[!names(CC_train) %in% c("default.pa
yment.next.month", "default.payment.next.month_high")])


test_data <- as.matrix(CC_test[!names(CC_train) %in% c("default.paym
ent.next.month", "default.payment.next.month_high")])

test_labels <- to_categorical(CC_test[,"default.payment.next.month_h
igh"])
```

```
model <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "softmax",
              input_shape = dim(train_data)[2]) %>%
  layer_dense(units = 64, activation = "softmax") %>%
  layer_dense(units = 2, activation= "softmax")
```

```
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

early_stop <- callback_early_stopping(monitor = "val_loss", patience
= 20)

epochs=300



history_class <- model %>% fit(
  train_data,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  callbacks = list(early_stop)
)

plot(history_class)
```



Epoch means the number of changing times. These graph stops at 45, and the validation’s accuracy rate is much higher than the training.

```
test_predictions <- model %>% predict(test_data)
```

```
set.seed(1)
test_class <- model %>% predict_classes(test_data)


table(test_labels[,2], test_class)
```

```
##    test_class
##        0    1
##   0 7001   17
##   1 1971   11
```

```
(16+1967)/(7002+16+1967+15)
```

```
## [1] 0.2203333
```

This model's prediction of an error rate is 0.2203333. Since this error rate is very high compared to the other models, this model will not be used in order to analyze this data set.