CreditCardReport3

Introduction

In the third report, I will be using data from the Credit Card Acceptance Report. In this project, I will be incorporating results from my first and second reports while rulling LASSO and Ridge regressions. LASSO, or Least Absolute Selection and Shrinkage Operator, and Ridge are shrinkage methods. I will be conducting Ridge first in my report. Then, I will be running a LASSO regression. This would allow me to then compare both methods to how my logit method in report 2 performed. I will then run a regression or classification tree. Also, for extra credit, I will use boot-strap and fit 100 different trees to my boot-strap subsamples.

```
library(ISLR)
library(AER)
library(ggplot2)
library(dplyr)
library(glmnet)
data("CreditCard")
CreditCard = data.frame(CreditCard)
names(CreditCard)
```

```
##  [1] "card"        "reports"     "age"         "income"      "share"
##  [6] "expenditure" "owner"       "selfemp"     "dependents"  "months"
## [11] "majorcards"  "active"
```

```
dim(CreditCard)
```

```
## [1] 1319   12
```

1)Divide credit card data into training and testing subsets and setting up for ridge and lasso regressions

```
set.seed(1)
train = CreditCard %>% sample_n(654)
test = CreditCard %>% setdiff(train)
x_train = model.matrix(card~., train)[,-1]
x_test = model.matrix(card~., test)[,-1]
y_train = train$card
y_test = test$card

x = model.matrix(card~., CreditCard)[,-1]
y = CreditCard$card
```

RIDGE REGRESSION

2a.Tuning the model: Cross-Validation

```
grid = 10^seq(10, -2, length = 100)
ridge_mod = glmnet(x, y, alpha = 0, lambda = grid, family = "binomial")
cv.out = cv.glmnet(x_train, y_train, alpha = 0, family = "binomial") # Fit ridge regression model
on training data
```

2b. Choosing lambda within one standard error of minimum lambda

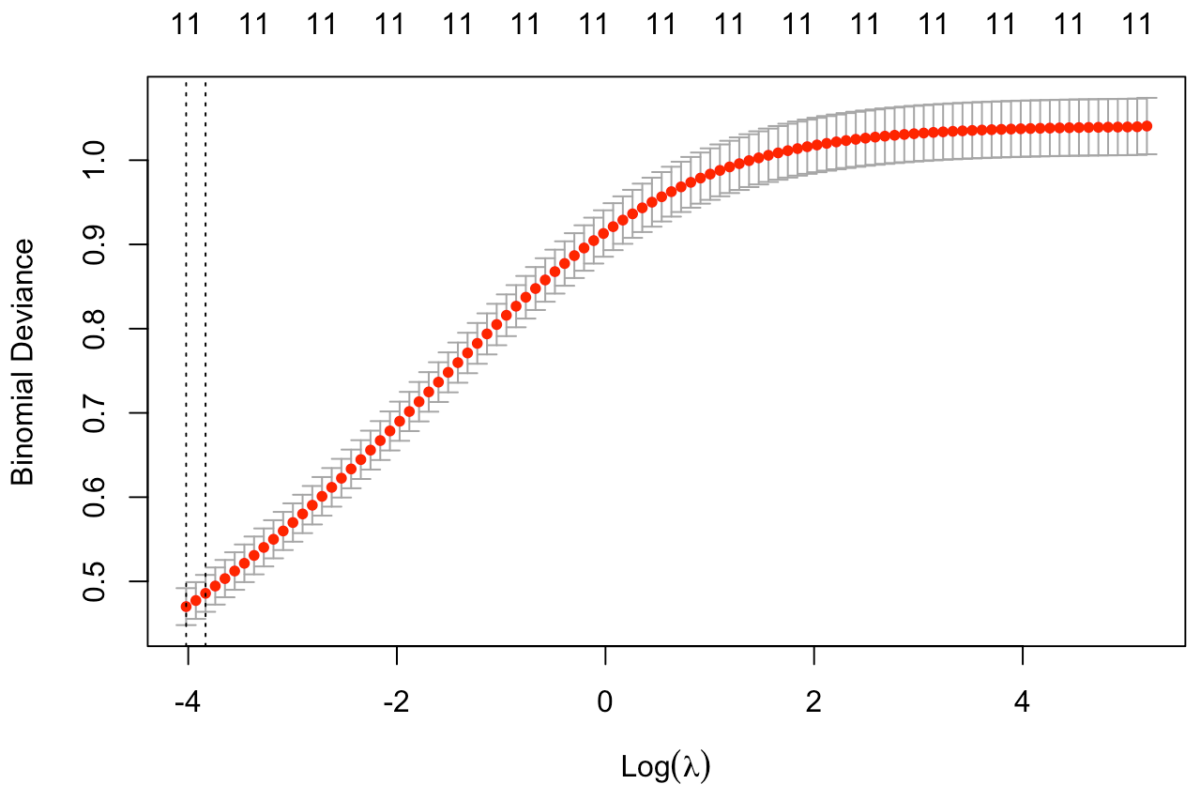
```
bestlam = cv.out$lambda.min  # Select lamda that minimizes training MSE
bestlam
```

```
## [1] 0.0179514
```

The best model is gonna be close to an OLS model since my lambda is 0, but I will choose 1.

2c. Show the cross-validation error and the chosen lambda in a graph

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



In this graph, it is showing me the values of lambda that results in the smallest cross validation for 0. This plot will allow me to plot the MSE as a function of lambda.

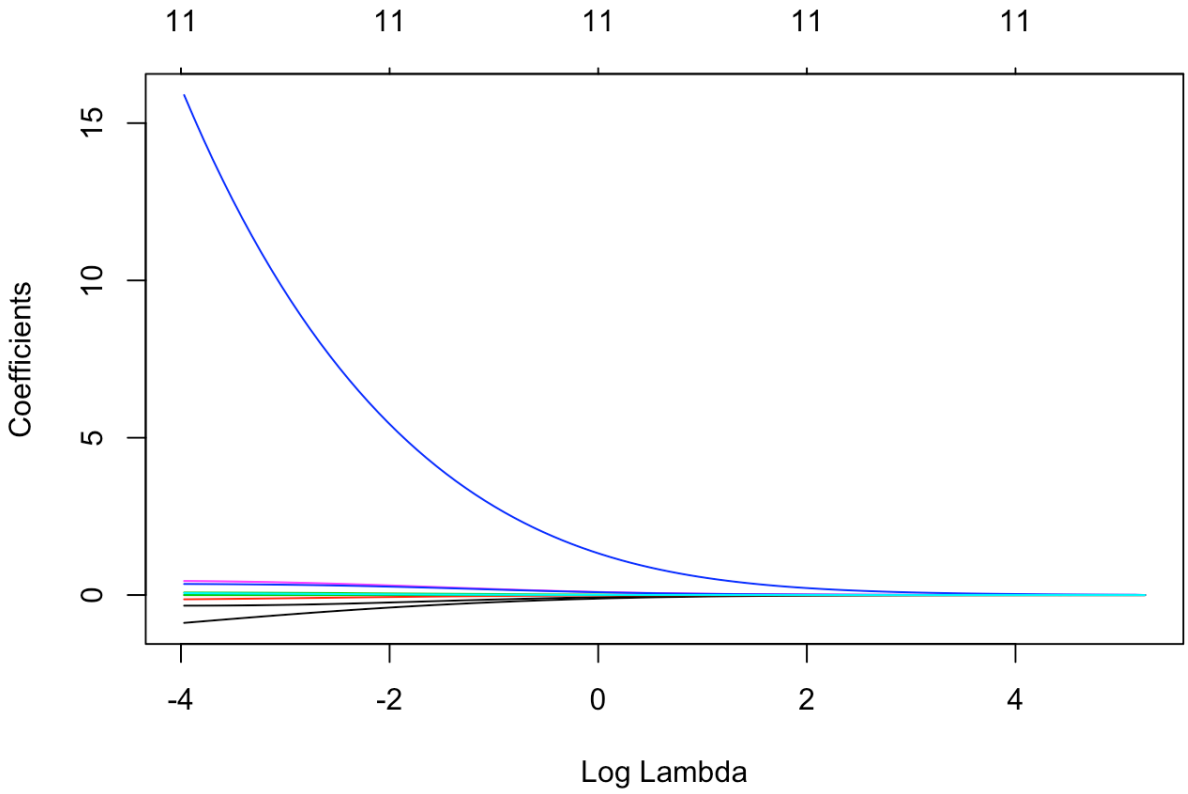
2d. Show how the coefficients vary with lambda in a graph

```
ridge_pred = predict(ridge_mod, s = 1, newx = x_test) # Use best lambda to predict test data
mean((ridge_pred - y_test)^2) # Calculate test MSE
```

```
## Warning in Ops.factor(ridge_pred, y_test): '-' not meaningful for factors
```

```
## [1] NA
```

```
out = glmnet(x, y, alpha = 0, family = "binomial") # Fit ridge regression model on the FULL datase
t (train and test)
plot(out, xvar = "lambda")
```



In this graph, we see that none of the coefficients are exactly zero, which is correct as shown below. We are plotting the coefficients for the different values of lambda.

2e. Report the coefficients correspondent with the chosen l

```
predict(out, type = "coefficients", s = bestlam)[1:12,] # Display coefficients using lambda chosen
by CV
```

```
##   (Intercept)       reports           age        income         share
## -0.2172574440 -0.8850259823 -0.0011027159  0.0818480789 15.8922215919
##   expenditure      owneryes    selfempyes    dependents        months
##  0.0048072023  0.4458577211 -0.3376107430 -0.1357754099 -0.0003058545
##    majorcards        active
##  0.3515142631  0.0611774102
```

2f. Report the MSE (Error Rate for classification) in the test subset

```
ridge_pred = predict(ridge_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
mean((ridge_pred - y_test)^2) # Calculate test MSE
```

```
## Warning in Ops.factor(ridge_pred, y_test): '-' not meaningful for factors
```

```
## [1] NA
```

THE MSE is close to 101 when compared to the test data. #start of lasso regression

LASSO REGRESSION

2a.Tuning the model: Cross-Validation

```
lasso_mod = glmnet(x_train,
                   y_train,
                   alpha = 1,
                   family="binomial") # Fit lasso model on training data
```

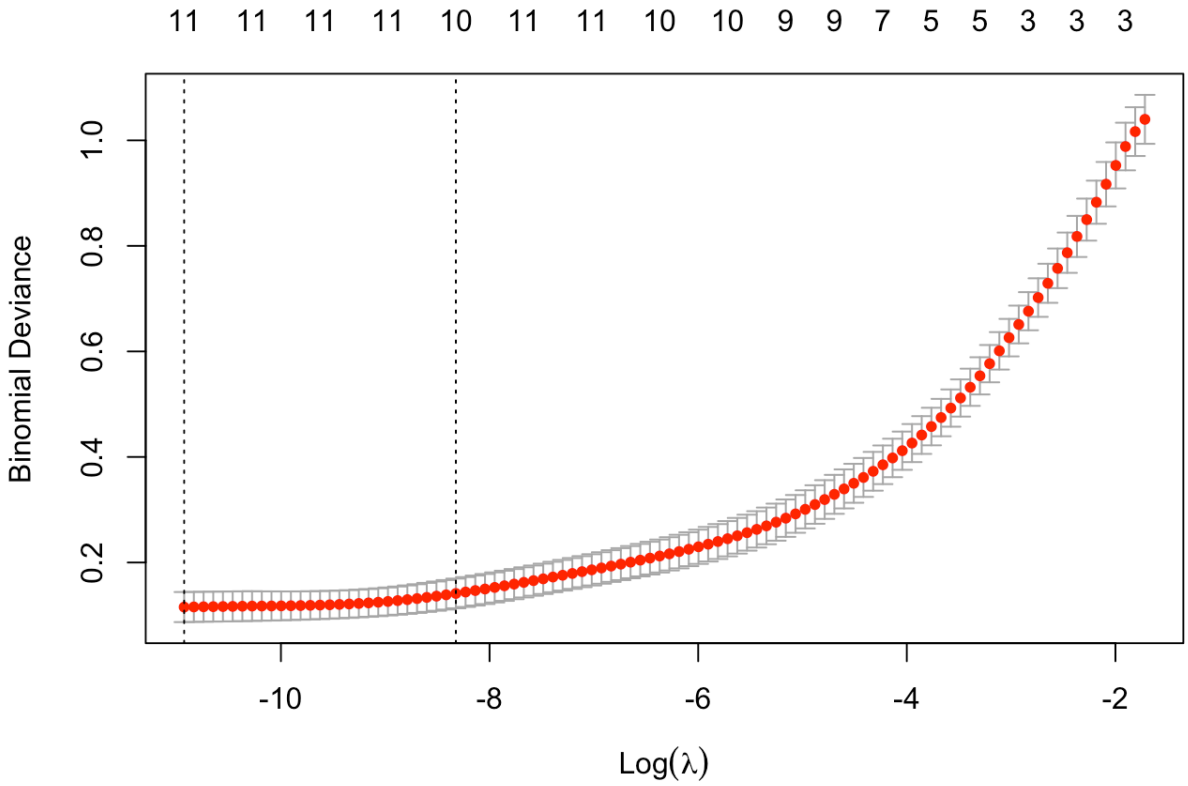
2b. Choosing lambda within one standard error of minimum lambda

```
set.seed(1)
cv.out = cv.glmnet(x_train, y_train, alpha = 1, family="binomial") # Fit lasso model on training d
ata
```

```
## Warning: from glmnet Fortran code (error code -99); Convergence for 99th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet Fortran code (error code -98); Convergence for 98th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
plot(cv.out) # Draw plot of training MSE as a function of lambda
```

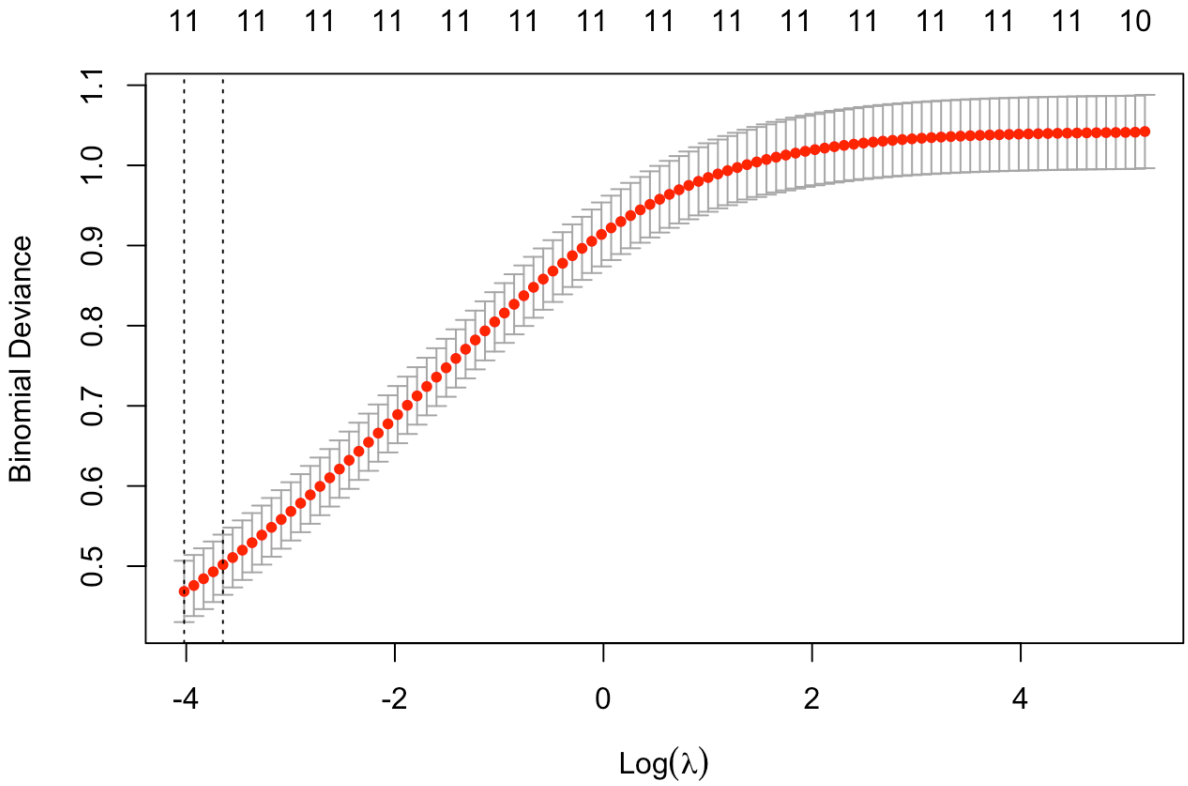



```
bestlam = cv.out$lambda.min # Select lamda that minimizes training MSE
bestlam
```

```
## [1] 1.79514e-05
```

2c. Show the cross-validation error and the chosen l in a graph

```
set.seed(1)
cv.out = cv.glmnet(x_train, y_train, alpha = bestlam, family="binomial") # Fit lasso model on trai
ning data
plot(cv.out) # Draw plot of training MSE as a function of lambda
```

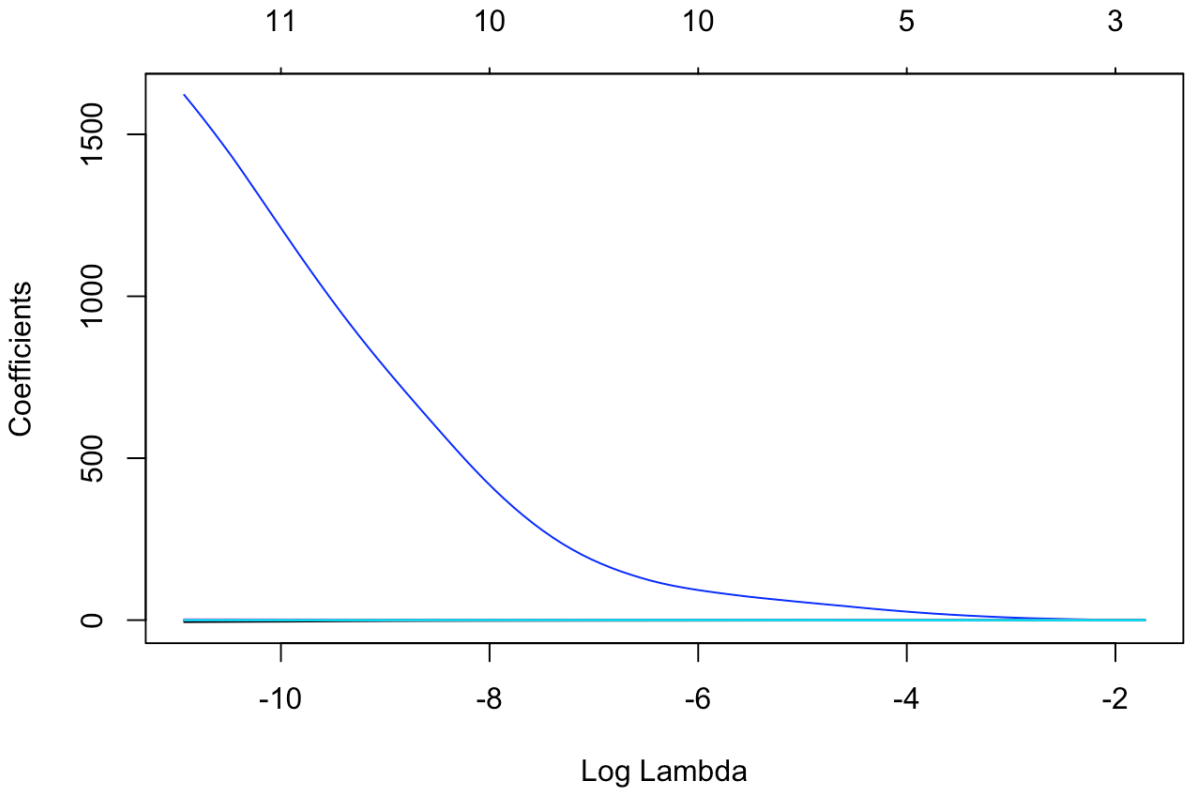


2d. Show how the coefficients vary with l in a graph

```
ls(lasso_mod)
```

```
##  [1] "a0"         "beta"       "call"       "classnames" "dev.ratio"
##  [6] "df"         "dim"        "jerr"       "lambda"     "nobs"
## [11] "npasses"    "nulldev"    "offset"
```

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



In this graph, we see that there is a lot of It looks like a lot of the coefficients are overlapping each other.

2e. Report the coefficients correspondent with the chosen l

```
out = glmnet(x, y, alpha = 1, lambda = grid, family = "binomial") # Fit lasso model on full datase
t
lasso_coef = predict(out, type = "coefficients", s = bestlam)[1:12,] # Display coefficients using
lambda chosen by CV
lasso_coef
```

```
## (Intercept)     reports         age      income       share expenditure
## -0.73851677 -1.05336272  0.00000000  0.07419872 70.23998753  0.00000000
##    owneryes  selfempyes  dependents      months  majorcards      active
##  0.22440783  0.00000000 -0.01879572  0.00000000  0.17443795  0.06705179
```

```
lasso_coef[lasso_coef != 0] # Display only non-zero coefficients
```

```
## (Intercept)     reports      income       share    owneryes  dependents
## -0.73851677 -1.05336272  0.07419872 70.23998753  0.22440783 -0.01879572
##  majorcards      active
##  0.17443795  0.06705179
```

```
bestlam
```

```
## [1] 1.79514e-05
```

2f. Report the MSE (Error Rate for classification) in the test subset

```
lasso_pred = predict(lasso_mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
mean((lasso_pred - y_test)^2) # Calculate test MSE
```

```
## Warning in Ops.factor(lasso_pred, y_test): '-' not meaningful for factors
```

```
## [1] NA
```

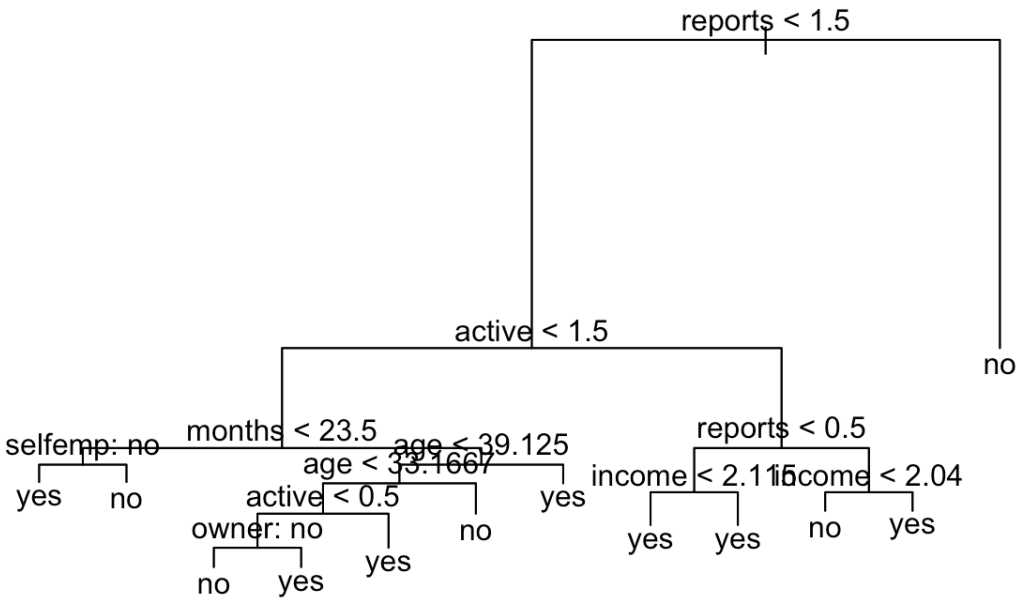
Running Classification Tree

3a. Fit and plotting a tree

```
library(tree)
tree_creditcard = tree(card ~ (active+majorcards+months+dependents+selfemp+owner+income+age+report
s), train)
summary(tree_creditcard)
```

```
##
## Classification tree:
## tree(formula = card ~ (active + majorcards + months + dependents +
##     selfemp + owner + income + age + reports), data = train)
## Variables actually used in tree construction:
## [1] "reports" "active"  "months"  "selfemp" "age"     "owner"   "income"
## Number of terminal nodes:  12
## Residual mean deviance:  0.6367 = 408.8 / 642
## Misclassification error rate: 0.1101 = 72 / 654
```

```
plot(tree_creditcard)
text(tree_creditcard, pretty = 0)
```



3b. Error rate and mse on tree

```
tree_pred = predict(tree_creditcard, test, type = "class")
table(tree_pred, test$card)
```

```
##
## tree_pred  no yes
##       no   72  27
##       yes  84 482
```

```
mean((tree_pred - CreditCard$card)^2)
```

```
## Warning in Ops.factor(tree_pred, CreditCard$card): '-' not meaningful for
## factors
```

```
## [1] NA
```

```
errorrate = (72+482)/(72+27+84+482)
errorrate
```

```
## [1] 0.8330827
```

The error rate is about 83% so we see that the pruned tree is classifying it correctly 83% of the time which is around the same as our shrinkage methods.

Use cross validation to prune your tree

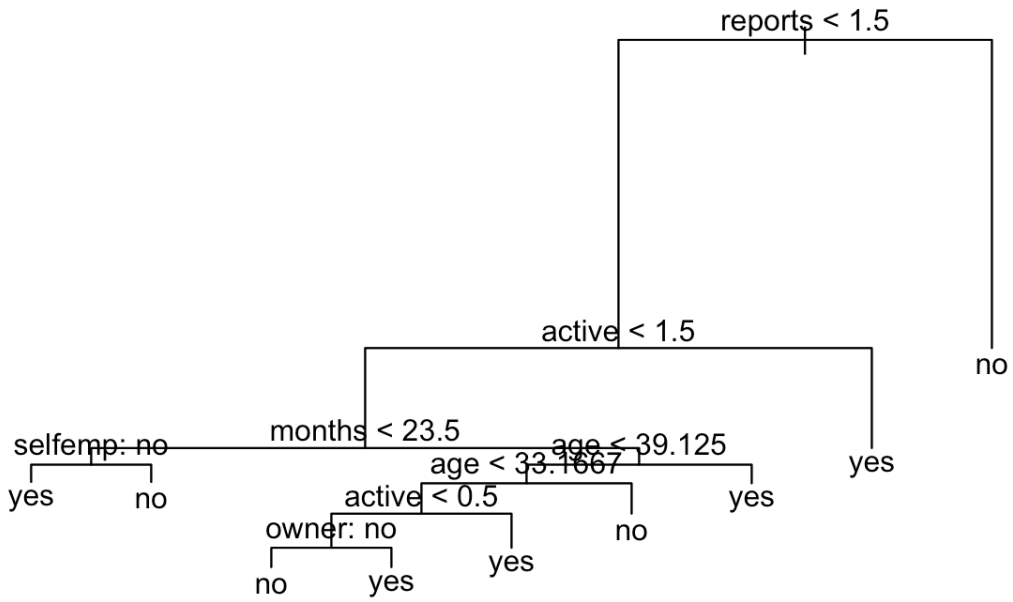
```
set.seed(5)
cv.creditcard = cv.tree(tree_creditcard, FUN = prune.misclass)
cv.creditcard
```

```
## $size
## [1] 12 11  9  7  2  1
##
## $dev
## [1] 112 115 106  98  97 140
##
## $k
## [1] -Inf  0.0  1.5  3.0  3.2 43.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
#plot(cv.creditcard$card, cv.creditcard$dev, type = "b")
```

3d.Plotting the pruned tree

```
prune_creditcard = prune.misclass(tree_creditcard, best = 8)
plot(prune_creditcard)
text(prune_creditcard, pretty = 0)
```




```
#plot(cv.creditcard$card, cv.creditcard$dev, type = "b")
```

{r} #tree_pred = predict(prune_creditcard, test, type = "class") #table(tree_pred, CreditCard$card)

Extra Credit: using the boot strap