Hw-4

Homework 4

Q1

Below we import the data, perform principle component analysis and train a basic liniear model with the first four components (highest variance explained).

```
# load the libraries
library(caret)
```

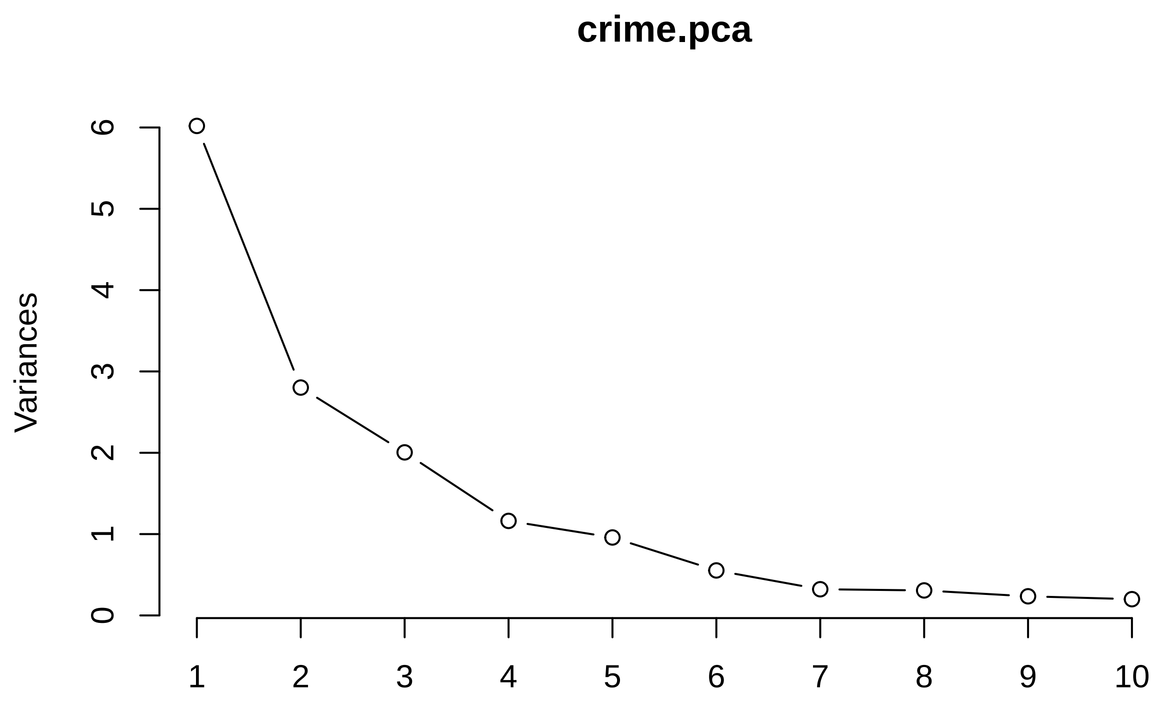
```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
crime = read.table('../Hw-3/uscrime.txt', header = TRUE)
crime.pca <- prcomp(crime[1:15], scale. = TRUE)
summary(crime.pca)
```

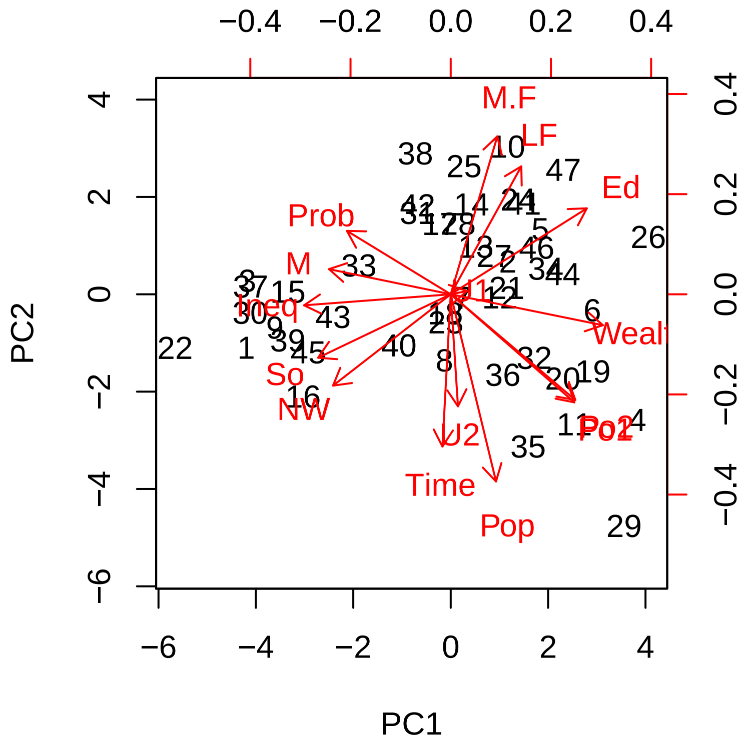
```
## Importance of components%s:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.56729 0.55444 0.48493 0.44708 0.41915 0.35804
## Proportion of Variance 0.02145 0.02049 0.01568 0.01333 0.01171 0.00855
## Cumulative Proportion  0.92142 0.94191 0.95759 0.97091 0.98263 0.99117
##                           PC13   PC14    PC15
## Standard deviation     0.26333 0.2418 0.06793
## Proportion of Variance 0.00462 0.0039 0.00031
## Cumulative Proportion  0.99579 0.9997 1.00000
```

The first 4 elements explain 80% of the variance. Lets have a look at the breakdown graphically:

```
plot(crime.pca, type = "l")
```



```r
biplot(crime.pca, scale = 0)
```



From the elbow graph we see the diminishing return each component contributes. From the biplot we see the rotation and level of importance.

```r
sd <- crime.pca$sdev
loadings <- crime.pca$rotation
rownames(loadings) <- colnames(crime[1:15])
scores <- crime.pca$x

crime.train <- as.data.frame(crime.pca$x[,1:4], header = T)
crime.train$Y <- crime$Crime

model.pca <- lm(Y~., data=crime.train)
summary(model.pca)
```

```
##
## Call:
## lm(formula = Y ~ ., data = crime.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -557.76 -210.91  -29.08  197.26  810.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      49.07  18.443  < 2e-16 ***
## PC1            65.22      20.22   3.225  0.00244 **
## PC2           -70.08      29.63  -2.365  0.02273 *
## PC3            25.19      35.03   0.719  0.47602
## PC4            69.45      46.01   1.509  0.13872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 336.4 on 42 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.2433
## F-statistic: 4.698 on 4 and 42 DF,  p-value: 0.003178
```

With an r^2 of 30.1% we explain less variance than last weeks homework. Below we can explore preprocessing the parameters in order to reverse engineer the model back into the original factors. I will use 10-fold cross validation in the training.

```r
# calculate the pre-process parameters from the dataset
preprocessParams <- preProcess(crime[1:15], method=c("center", "scale", "pca"), pcaComp = 4)
# preprocessParams$rotation

# transform the dataset using the parameters
transformed <- predict(preprocessParams, crime[1:15])
# summarize the transformed dataset
# summary(transformed)
transformed$Y <- crime$Crime

control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
# lm
set.seed(7)
fit.lm <- train(Y~., data=transformed, method="lm", metric=metric, trControl=control)
final.model <- fit.lm$finalModel

model.inter<-as.data.frame(preprocessParams$rotation)
for (i in 1:4){
  model.inter[,i] <- model.inter[,i]*final.model$coefficients[i+1]
}

coefs <- data.frame(row.names = c(colnames(crime[1:15]), "intercept"))
for (i in 1:15){
  coefs[i,1] <- sum(model.inter[i,])
}
coefs[16,1]<-final.model$coefficients[1]
coefs
```

```
##                 V1
## M       -21.277963
## So       10.223091
## Ed       14.352610
## Po1      63.456426
## Po2      64.557974
## LF      -14.005349
## M.F     -24.437572
## Pop      39.830667
## NW       15.434545
## U1      -27.222281
## U2        1.425902
## Wealth   38.607855
## Ineq      -27.536348
## Prob        3.295707
## Time       -6.612616
## intercept 905.085106
```

So this is our model translated back into the original elements in via "un-rotation"

Q2

Below we build the models for the regression tree and the random forest:

```
library(caret)
library(rpart)
library(corrplot)

crime = read.table('../Hw-3/uscrime.txt', header = TRUE)

# Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"

set.seed(42)
grid <- expand.grid(.cp=c(0, 0.05, 0.1))
fit.cart <- train(Crime~., data=crime, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center"
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

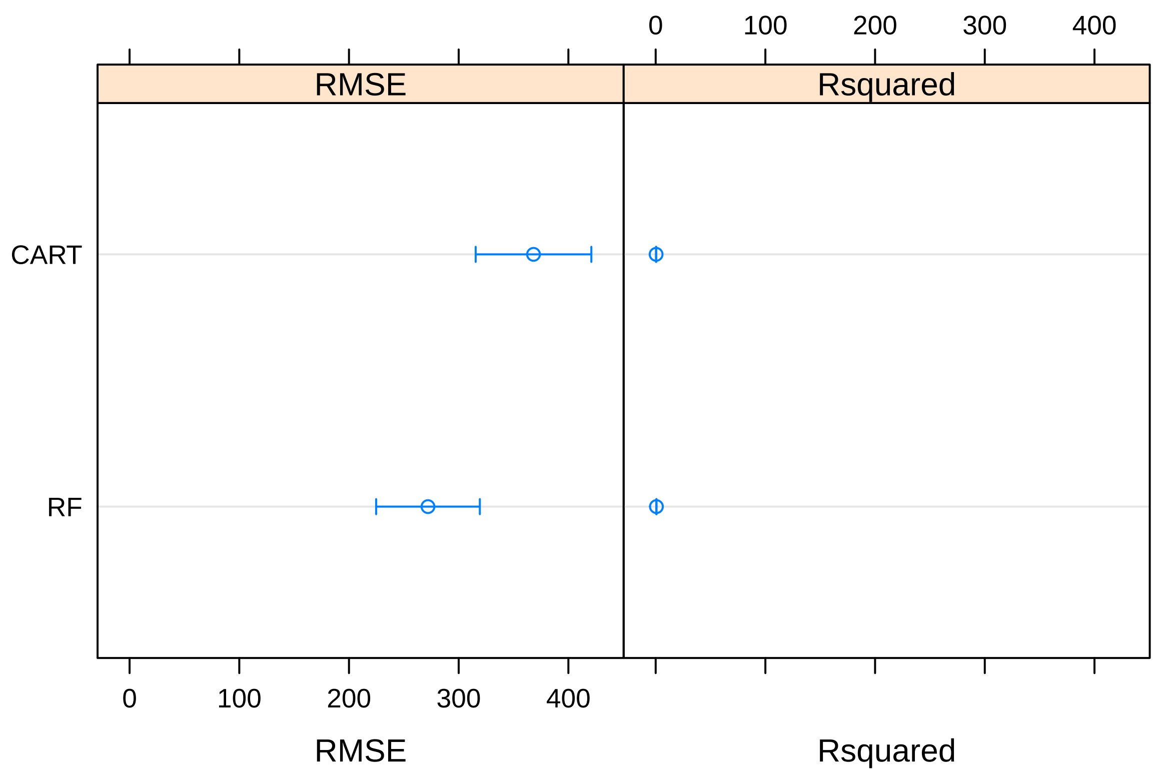
```
# Random Forest
set.seed(42)
fit.rf <- train(Crime~., data=crime, method="rf", metric=metric, preProc=c("BoxCox"), trControl=control
```

```
## Loading required package: randomForest

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

```
# Compare algorithms
transform_results <- resamples(list(CART=fit.cart, RF=fit.rf))
summary(transform_results)
```

```
##
## Call:
## summary.resamples(object = transform_results)
##
## Models: CART, RF
## Number of resamples: 30
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## CART 143.56524 247.7585 342.3559 368.2294 467.7376 682.7395    0
## RF    75.71638 176.4595 260.3188 272.0210 346.3054 537.0275    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## CART 0.0001149384 0.1016129 0.3311003 0.3735410 0.6683652 0.9818932    3
## RF   0.0008630734 0.4658528 0.7460322 0.6717879 0.9498149 0.9901082    0
```

```r
dotplot(transform_results)
```



Confidence Level: 0.95

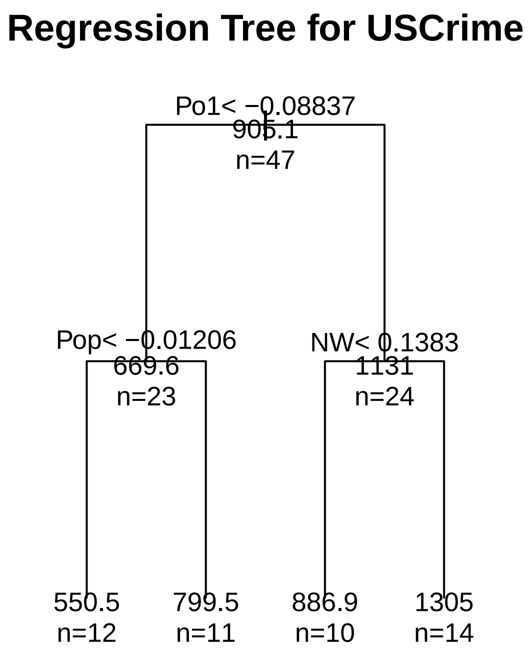
The RF model had a slightly lower RMSE, They had very high R-sq values. Lets have a look at the look at each of these to understand the results:

```r
par(mfrow = c(1,2), xpd = NA)
plot(fit.cart$finalModel, uniform=TRUE,
    main="Regression Tree for USCrime")
text(fit.cart$finalModel, use.n=TRUE, all=TRUE, cex=.8)

print(fit.rf$finalModel) # view results
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 88026.1
##                      % Var explained: 39.87
```

```
importance(fit.rf$finalModel, type=2) # importance of each predictor
```

```
##          IncNodePurity
## M            247503.28
## So            18299.65
## Ed           197872.76
## Po1         1296476.21
## Po2         1284771.18
## LF           201132.67
## M.F          244774.20
## Pop          274764.11
## NW           494086.68
## U1           111766.71
## U2           152780.11
## Wealth       718980.85
## Ineq         212007.22
## Prob         912158.35
## Time         157958.49
```



We can see that there were quite a lot of impurity in certain nodes. As with previous analysis we can probably prune down some of the fetures to get a higher performing model. From the tree structure, we can see that the decisions are Po1, Pop and NW.

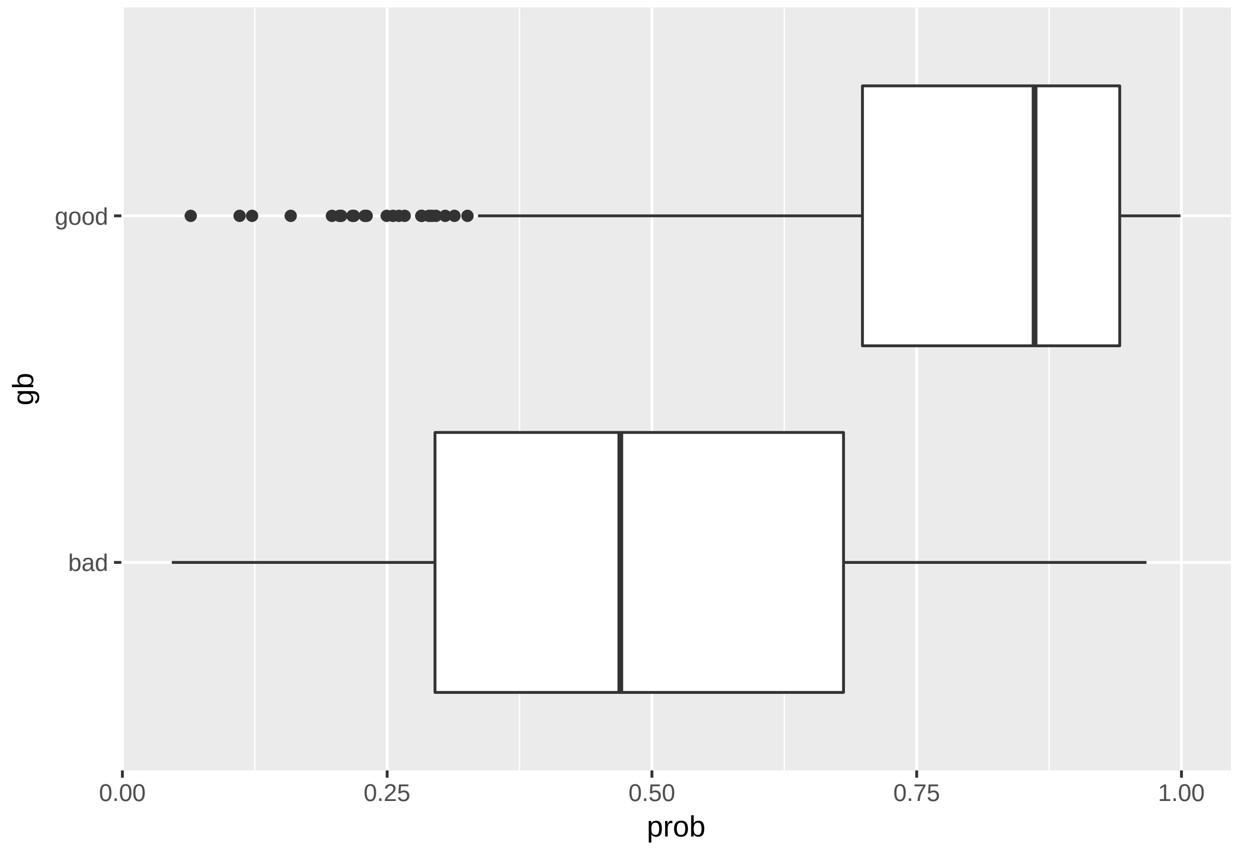
Q3

We use logistic regression nodes in deep learning algorithims in combination with relus to do certain computer vision tasks. So looking at a sequence of pixels it might say this image is a possible case of fraud. Another example may be when considering which sites to drill for oil. Features may include: mineral sample levels, location of previous oil sites etc.

Q4

Lets produce our logistic regression model

```
credit = read.table("germancredit.txt")

library(ggplot2)

names(credit) <- c('ca_status','mob','credit_history','purpose','credit_amount','savings',
                   'present_employment_since','status_sex','installment_rate_income','other_debtors
                   'present_residence_since','property','age','other_installment','housing','existi
                   'job','liable_maintenance_people','telephone','foreign_worker','gb')

credit$gb <- factor(credit$gb,levels=c(2,1),labels=c("bad","good"))
model <- glm(data=credit,formula=gb~.,family=binomial(link="logit"))
credit$prob <- predict(model,newdata=credit, type="response")

ggplot(data=credit) + geom_boxplot(aes(y=prob,x=gb))  + coord_flip()
```



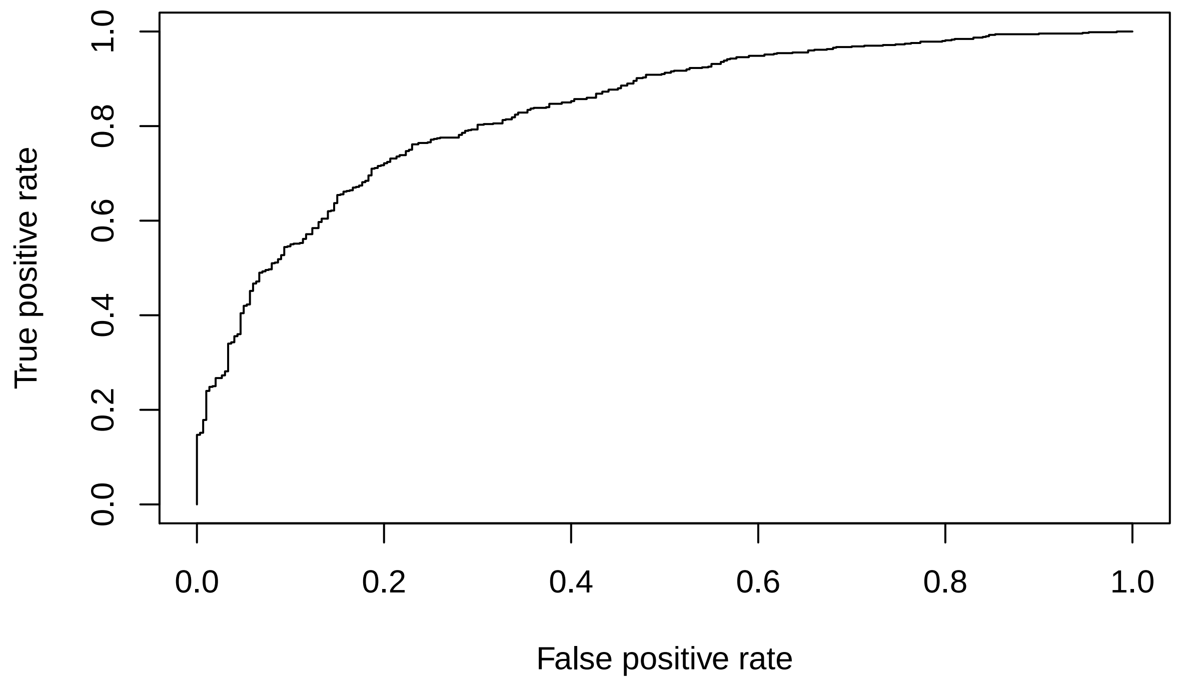
Just from looking at the distribution the model is doing a reasonable job of seperating the data with more than 75% of the good credit ratings above 75% of the bad credit ratings.

Now we will have a look at the ROC and AUC measures:

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
pr <- prediction(credit$prob, credit$gb)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```



```r
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.833781
```

this looks pretty good, we see a good climb in performance, and an auc of 83% certainly suggests this may be a useful model.

Finally, lets go about threshilding this. We will define a cost function and then just desend this from 50% through to where cost starts to increase:

```
thresh <- 0.5
cost <- Inf

repeat{
  fitted.results <- ifelse(credit$prob > thresh,"good","bad")
  CM <- table(credit$gb, fitted.results)
  new_cost = 5*CM[1,2]+CM[2,1]
  if(new_cost < cost){
    cost <- new_cost
    thresh <- thresh +0.01
  }
  else{
    thresh <- thresh - 0.01
    break
  }
}
thresh
```

```
## [1] 0.69
```

We produce a threshold of 69% (lol), lets have a look at what the model performance is (on the training data) at this threshold:

```
fitted.results <- ifelse(credit$prob > thresh,"good","bad")
CM <- table(credit$gb, fitted.results)
thresh
```

```
## [1] 0.69
```

```
CM
```

```
##       fitted.results
##        bad good
##   bad  231   69
##   good 167  533
```

```
misClasificError <- mean(fitted.results != credit$gb)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.764"
```

76.4% accuracy however this largely consists of good results classified as bad, rather than bad classified as good.

What a great time.