hw1_ml

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
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

library(ggplot2)
library(grid)
library(caTools)
library(caret)

## Loading required package: lattice

library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(ROCit)
library(rpart)
library(rpart.plot)
library(Rborist)

## Rborist 0.2-3
```

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

library(class)
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-3

library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

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:ggplot2':
##
##     margin

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

data <- read.csv("/home/yuvraj/Downloads/EDA/CC.csv", head
=T,stringsAsFactors=F)
data <- as.data.frame(data)
head(data)

##   X Time         V1          V2        V3         V4          V5
V6
## 1 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077
0.46238778
## 2 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -
0.08236081
## 3 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813
1.80049938
## 4 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888
1.24720317
## 5 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338
0.09592146
## 6 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -
```

```
0.02972755
##           V7          V8         V9         V10        V11
V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -
0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267
1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015
0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873
0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429
0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620
0.35989384
##          V13        V14        V15        V16         V17
V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124
0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -
0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -
0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279
1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -
0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282
0.06865315
##           V19         V20          V21          V22         V23
V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -
0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -
0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -
1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -
0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

```
summary(data)

##        X               Time              V1                  V2

##  Min.   :     1   Min.   :     0   Min.   :-56.40751   Min.   :-
72.71573
##  1st Qu.: 71202   1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -
0.59855
##  Median :142404   Median : 84692   Median :  0.01811   Median :
0.06549
##  Mean   :142404   Mean   : 94814   Mean   :  0.00000   Mean   :
0.00000
##  3rd Qu.:213606   3rd Qu.:139320   3rd Qu.:  1.31564   3rd Qu.:
0.80372
##  Max.   :284807   Max.   :172792   Max.   :  2.45493   Max.   :
22.05773
##        V3                  V4                  V5
V6
##  Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331
Min.   :-26.1605
##  1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160   1st
Qu.: -0.7683
##  Median :  0.1799   Median :-0.01985   Median :  -0.05434
Median : -0.2742
##  Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000
Mean   :  0.0000
##  3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd
Qu.:  0.3986
##  Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167
Max.   : 73.3016
##        V7                  V8                   V9
V10
##  Min.   :-43.5572   Min.   :-73.21672   Min.   :-13.43407
Min.   :-24.58826
##  1st Qu.: -0.5541   1st Qu.: -0.20863   1st Qu.: -0.64310   1st
Qu.: -0.53543
##  Median :  0.0401   Median :  0.02236   Median : -0.05143
Median : -0.09292
##  Mean   :  0.0000   Mean   :  0.00000   Mean   :  0.00000
Mean   :  0.00000
##  3rd Qu.:  0.5704   3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd
Qu.:  0.45392
##  Max.   :120.5895   Max.   : 20.00721   Max.   : 15.59500
Max.   : 23.74514
##       V11                 V12                 V13                V14

##  Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188   Min.   :-
19.2143
##  1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.:
-0.4256
```

```
##  Median :-0.03276   Median :  0.1400   Median :-0.01357   Median :
0.0506
##  Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :
0.0000
##  3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:
0.4931
##  Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688   Max.   :
10.5268
##       V15                V16                 V17
V18
##  Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280
Min.   :-9.498746
##  1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375   1st
Qu.:-0.498850
##  Median : 0.04807   Median :  0.06641   Median : -0.06568
Median :-0.003636
##  Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000
Mean   : 0.000000
##  3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd
Qu.: 0.500807
##  Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353
Max.   : 5.041069
##       V19                 V20                 V21
##  Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038
##  1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839
##  Median : 0.003735   Median : -0.06248   Median : -0.02945
##  Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000
##  3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638
##  Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284
##       V22                  V23                 V24
##  Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663
##  1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459
##  Median :  0.006782   Median : -0.01119   Median : 0.04098
##  Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953
##  Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455
##       V25                 V26                V27
##  Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679
##  1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840
##  Median :  0.01659   Median :-0.05214   Median :  0.001342
##  Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000
##  3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045
##  Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198
##       V28                Amount             Class
##  Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000
##  1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000
##  Median :  0.01124   Median :   22.00   Median :0.000000
##  Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728
##  3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000
##  Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000
```

```
data <- data[,-1]
data$Class <- as.factor(data$Class)
set.seed(69)
split <- sample.split(data$Class, SplitRatio = 0.7)
train <-  subset(data, split == TRUE)
test <- subset(data, split == FALSE)

down_sample <- downSample(x = train[,-ncol(train)],y = train$Class)
table(down_sample$Class)

##
##   0   1
## 344 344

up_sample <- upSample(x = train[,-ncol(train)], y=train$Class)
table(up_sample$Class)

##
##      0      1
## 199020 199020

final <- c()
```

Before we try to create a model to predict fraud we must think about our consumer The following are the problems with the data set: Highly unbalanced due to samll number of frauds that occur. We do not want to categorize a non fraud and predict it as a fraud as that will cause inconvenience to the consumer. We want to maximise fraud detection. Fraud detection for large amounts can have False Positives because of the risks involved. For small amounts the risk is not enough to actually create a problem to the consumer false positives should be almost negligible. Time is not a determining factor so do not take that into consideration as fraud can occur anytime.

The following are the needs of a bank for credit card fraud detection Maximize TP and TN. Minimize FP for amount that is small. Minimize FN for amount that is large.

We will try logistic regression to begin our analasis Using Down Sample for out training set

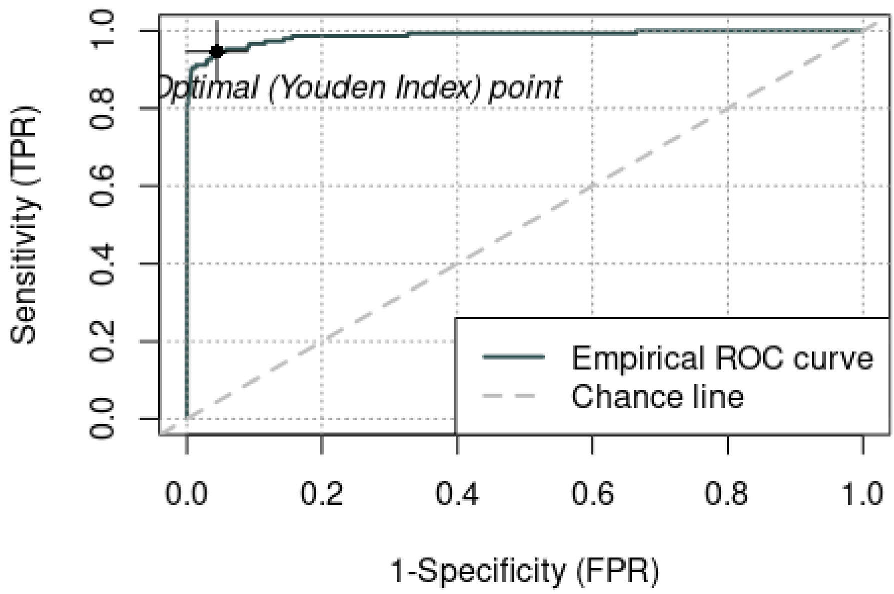
```
model <- glm(Class ~ ., data = down_sample, family = 'binomial')

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

prediction <- predict(model, newdata = test, type = 'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)
typeof(prediction)

## [1] "double"

result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 82206    11
##          1  3089   137
##
##                Accuracy : 0.9637
##                  95% CI : (0.9624, 0.965)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0782
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96378
##             Specificity : 0.92568
##          Pos Pred Value : 0.99987
```

```
##          Neg Pred Value : 0.04247
##              Prevalence : 0.99827
##          Detection Rate : 0.96212
##    Detection Prevalence : 0.96224
##       Balanced Accuracy : 0.94473
##
##        'Positive' Class : 0
##
```

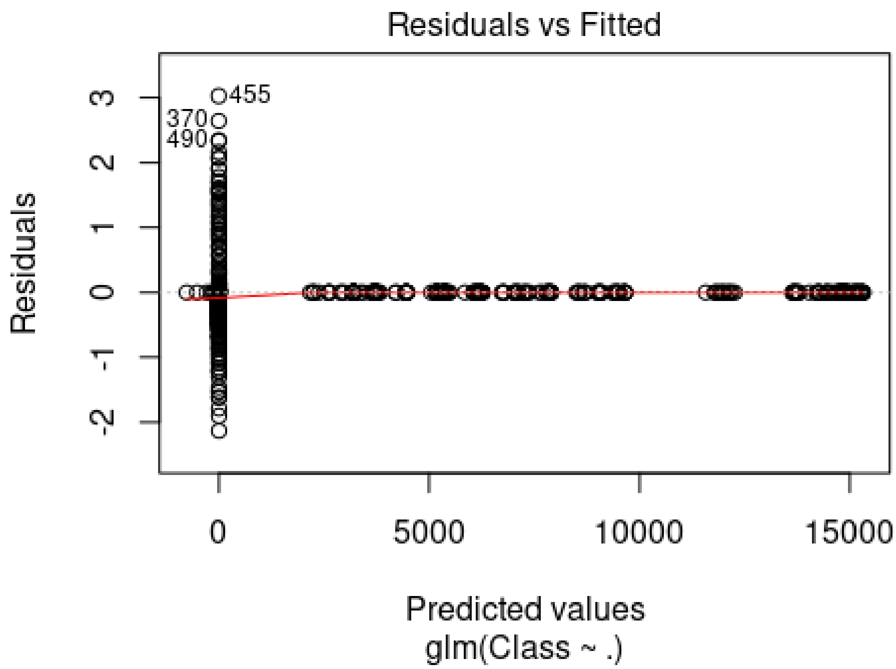
```
auc(test$Class, prediction)
```

```
## Setting levels: control = 0, case = 1
```

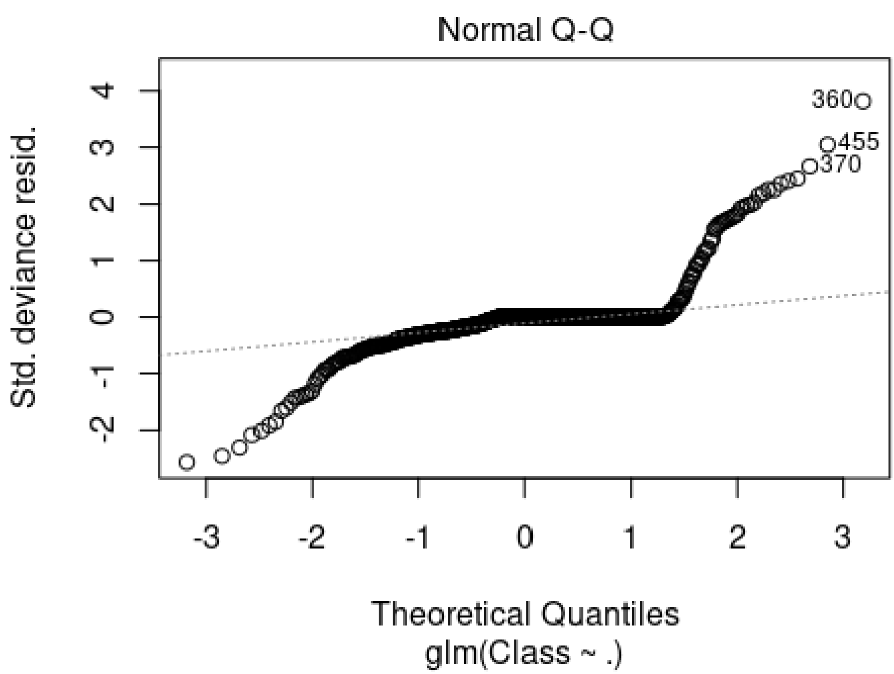
```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.9447
```

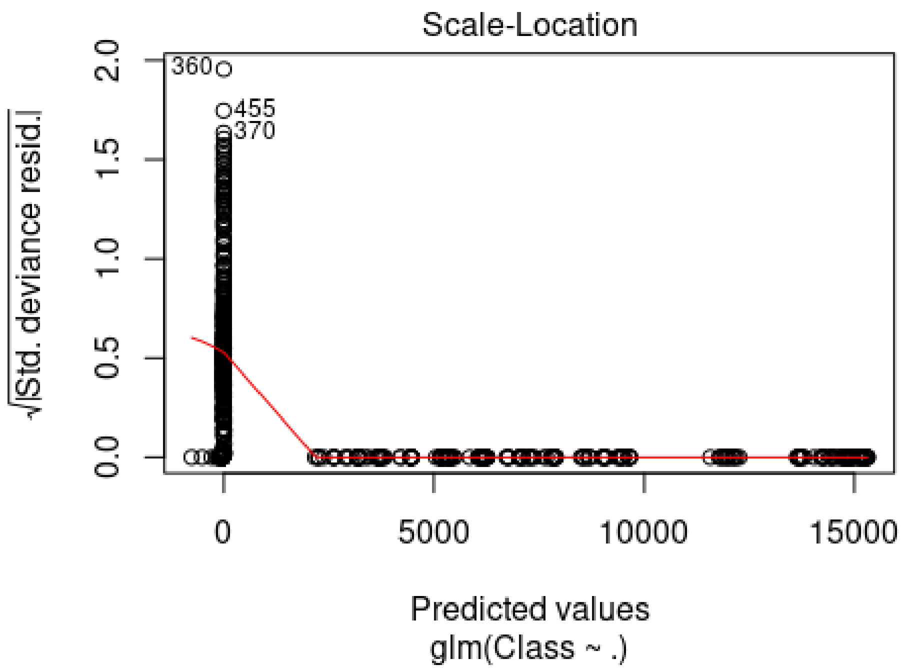
```
plot(model)
```

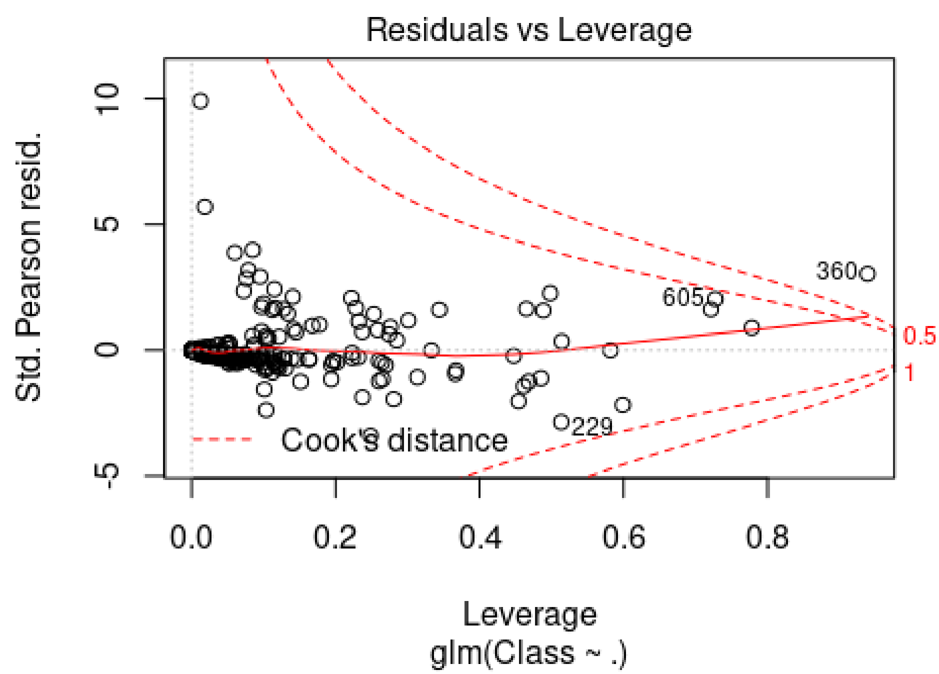


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
455
370
490
3
2
1
0
-1
-2
0
5000
10000
15000



Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(Class ~ .)
360
455
370
4
3
2
1
0
-1
-2
-3
-2
-1
0
1
2
3






```
final <- append(final, prediction)
```

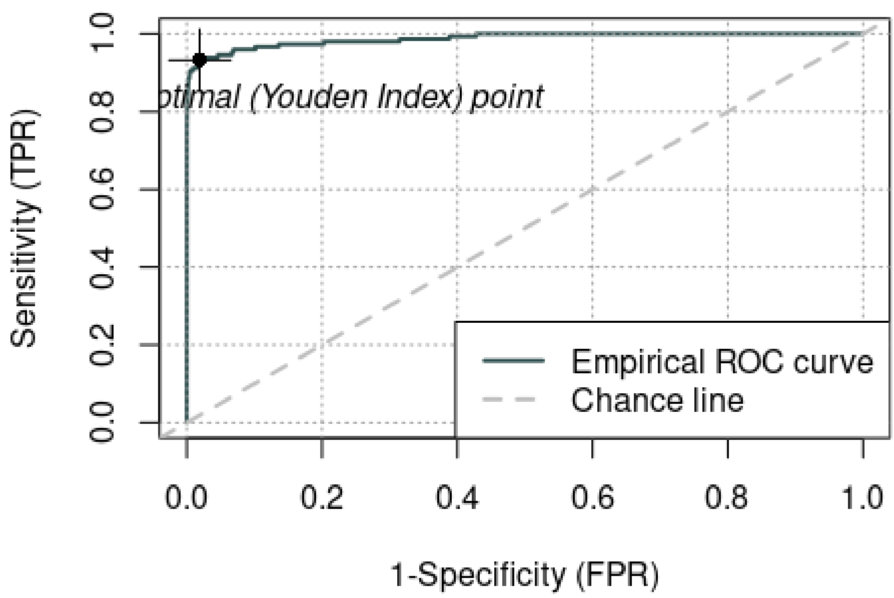
Summary : Acceptable FN Not acceptable FP rate but can work as a good model for large money credit statements The downsampling of data has caused a huge amount of important data that was necessary for making prediction to be completely removed from the sample due to which the model cannot comprehend complexity of test data which caused the model to be too simple to be used We will now try an upsample training data

```
model <- glm(Class ~ ., data = up_sample, family = 'binomial')

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

prediction <- predict(model, newdata = test, type = 'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)

result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 83296    10
##          1  1999   138
##
##                Accuracy : 0.9765
##                  95% CI : (0.9754, 0.9775)
```

```
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1179
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.97656
##             Specificity : 0.93243
##          Pos Pred Value : 0.99988
##          Neg Pred Value : 0.06458
##              Prevalence : 0.99827
##          Detection Rate : 0.97487
##    Detection Prevalence : 0.97499
##       Balanced Accuracy : 0.95450
##
##        'Positive' Class : 0
##
```

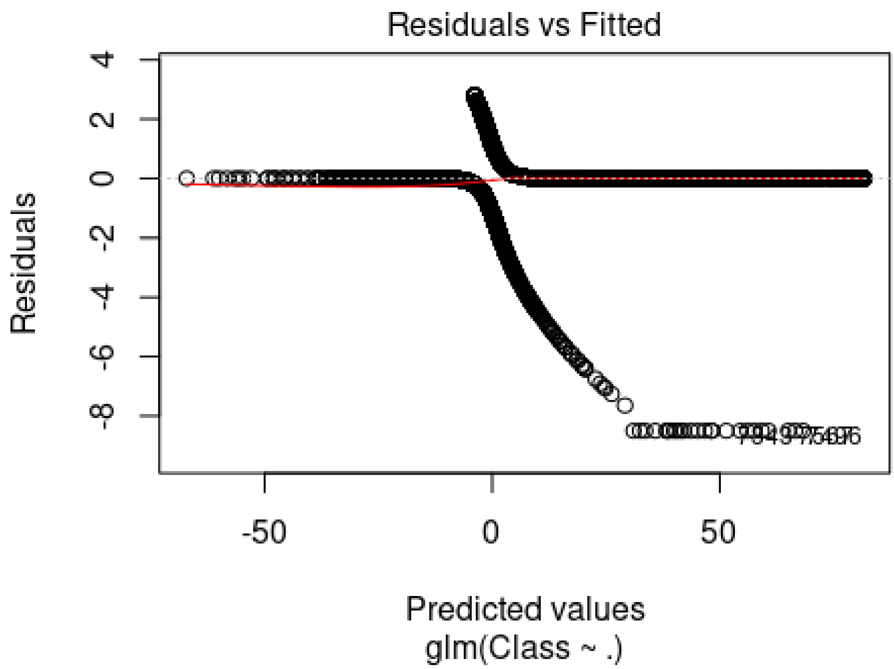
```
auc(test$Class, prediction)
```

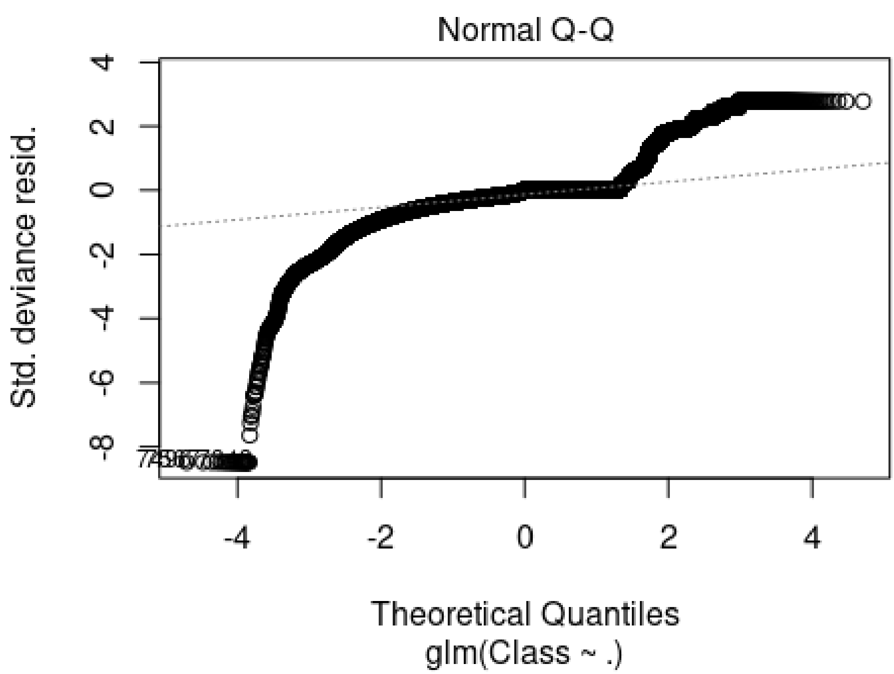
```
## Setting levels: control = 0, case = 1
```

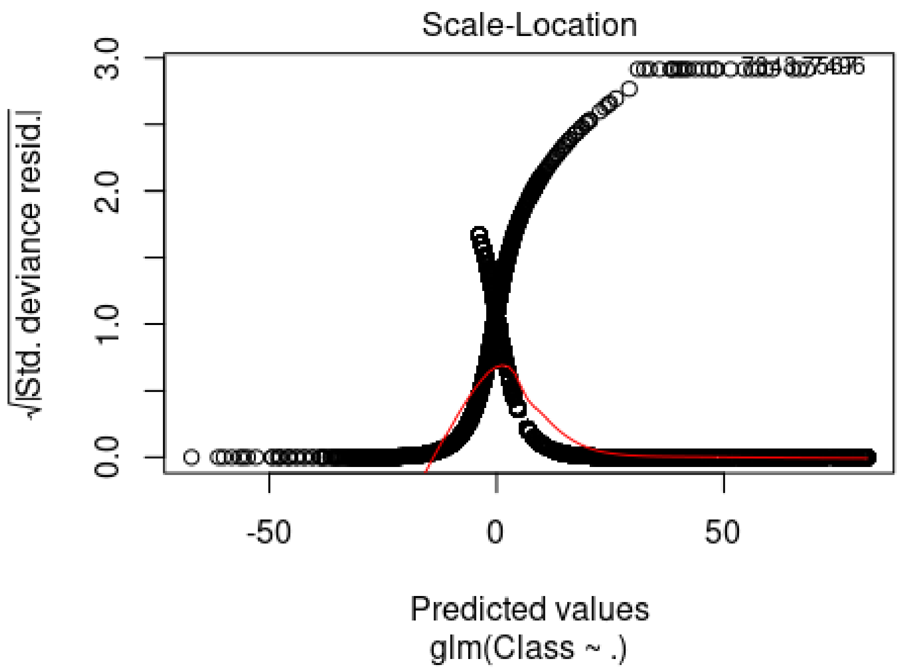
```
## Setting direction: controls < cases
```

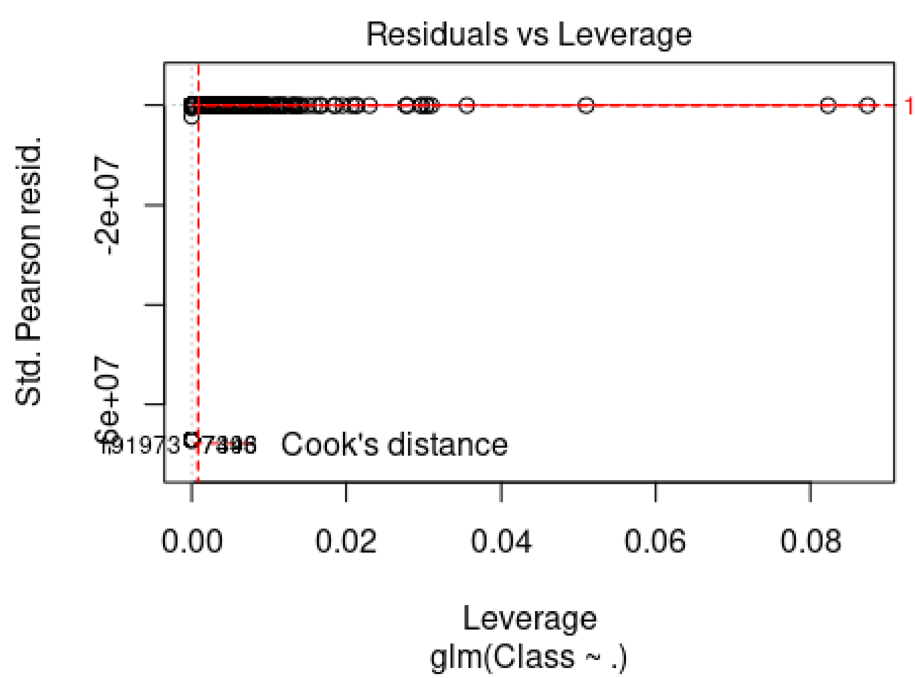
```
## Area under the curve: 0.9545
```

```
plot(model)
```


Residuals vs Fitted
Residuals
4
2
0
-2
-4
-6
-8
-50
0
50
Predicted values
glm(Class ~ .)


Normal Q-Q
Std. deviance resid.
4
2
0
-2
-4
-6
-8
-4
-2
0
2
4
Theoretical Quantiles
glm(Class ~ .)






```
final <- append(final, prediction)
```

Clearly a better model trained than upsample the number of FP has decreased a bit but stll not acceptable Upsampling has created a bias towards the positive class however due to it there are a lot of negatives being classified as positives however the trend has created negatives to be classified as positives thus making model too complex

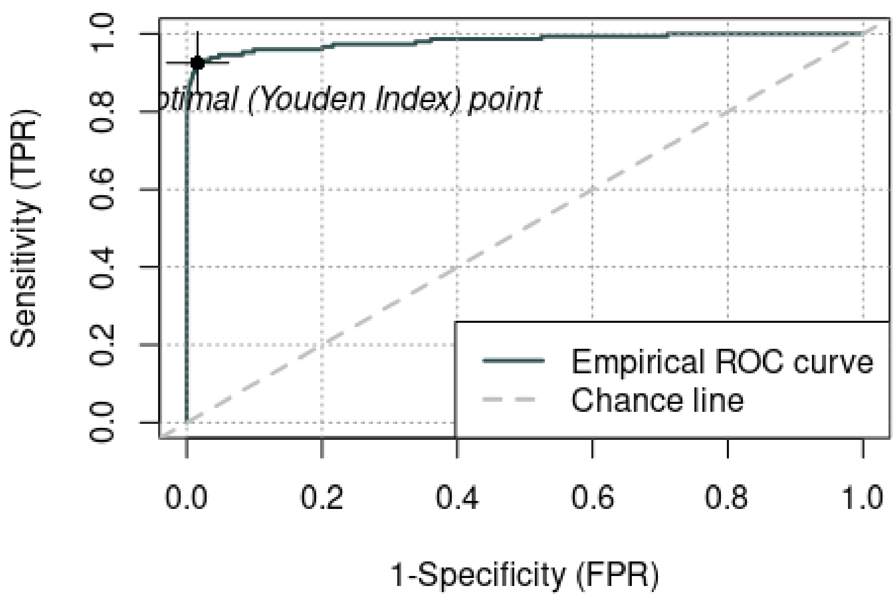
Training a model on entire dataset

```
model <- glm(Class ~ ., data = train, family = 'binomial')

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

prediction <- predict(model, newdata = test, type = 'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)

result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85285    58
##          1    10    90
##
##                Accuracy : 0.9992
##                  95% CI : (0.999, 0.9994)
```

```
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1.426e-13
##
##                   Kappa : 0.7254
##
##  Mcnemar's Test P-Value : 1.201e-08
##
##             Sensitivity : 0.9999
##             Specificity : 0.6081
##          Pos Pred Value : 0.9993
##          Neg Pred Value : 0.9000
##              Prevalence : 0.9983
##          Detection Rate : 0.9982
##    Detection Prevalence : 0.9988
##       Balanced Accuracy : 0.8040
##
##        'Positive' Class : 0
##
```

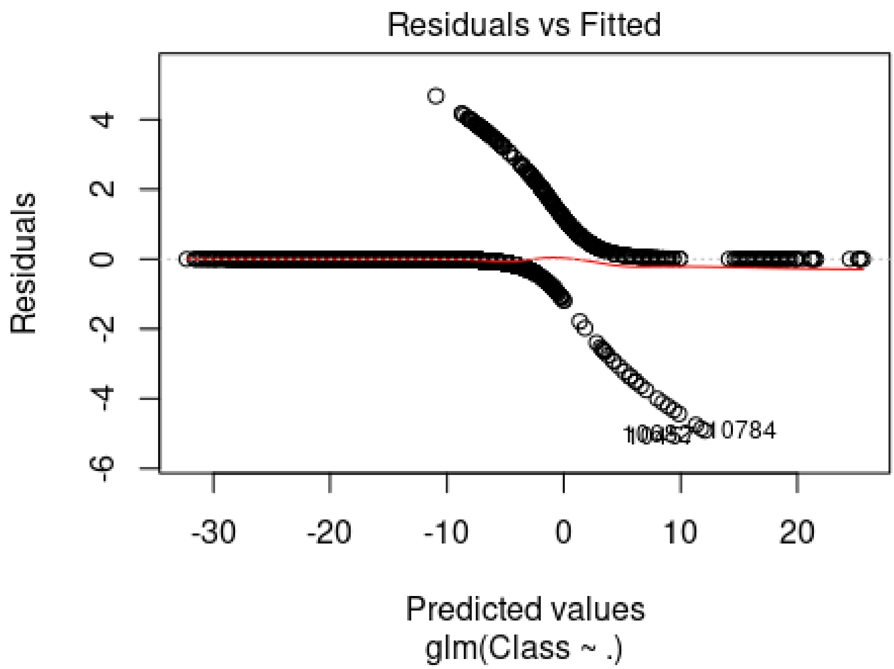
```
auc(test$Class, prediction)
```

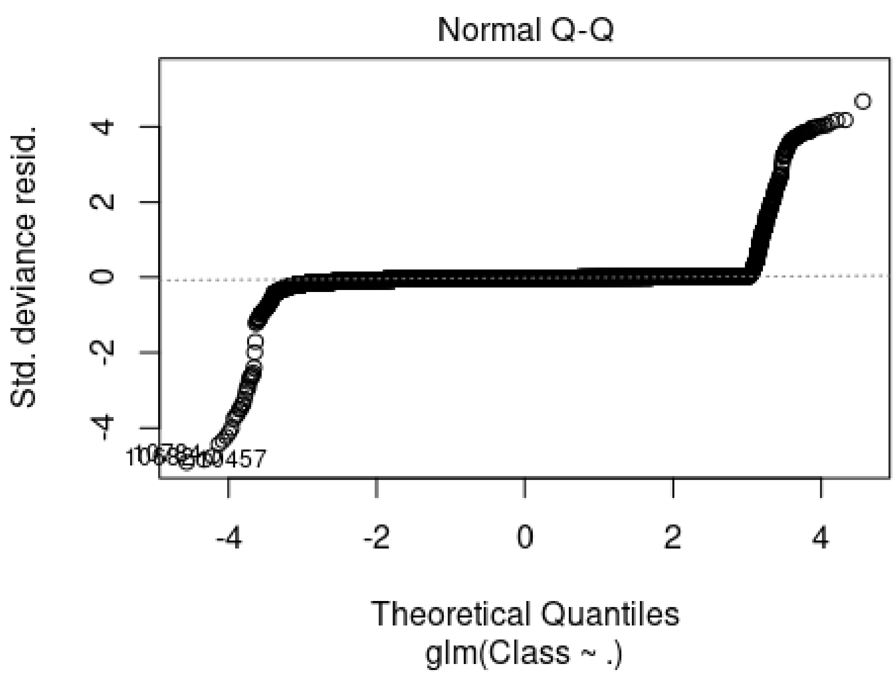
```
## Setting levels: control = 0, case = 1
```

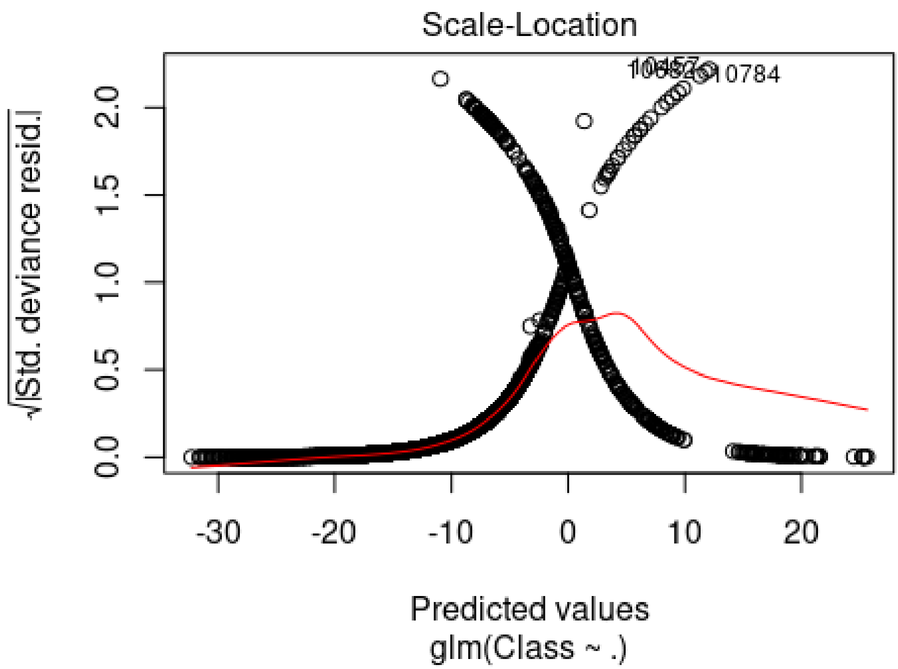
```
## Setting direction: controls < cases
```

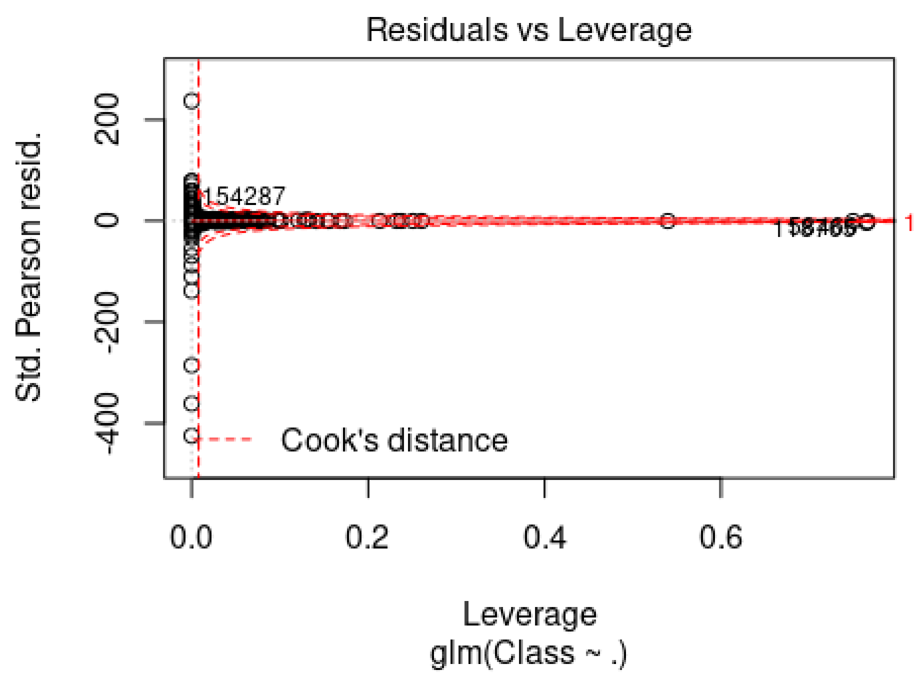
```
## Area under the curve: 0.804
```

```
plot(model)
```


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
10784
-30
-20
-10
0
10
20
4
2
0
-2
-4
-6


Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(Class ~ .)
10457
-4
-2
0
2
4






```
final <- append(final, prediction)
```

The FN have decreased substansially however FP havee increased to about 40% The model was able to learn a lot about predicting negatives correctly however due to bias caused by a huge number of negative training subjects the model was not able to calssify positives correctly and tends to classify them as negatives instead

Trying to form a model based on feature importance to try select better features

```
tree.data<-rpart(Class~.,data=data)
tree.data$variable.importance
```

```
##        V17        V12        V16        V10        V11        V18
V14
## 514.763411 395.257448 313.679409 279.201624 238.709736 217.447820
85.681800
##         V7        V27         V9         V6       Time         V8
V20
##  37.374873  32.694915  31.028687  26.791072  25.611488  24.642402
17.740322
##         V3         V4        V21        V26        V15         V2
V19
##  14.924998  14.694106  14.368522  13.641260   8.551106   8.358353
4.863039
##        V22
##   4.828564
```

These are the top 5 features according to feature importance V17, V12, V16, V10, V11, V18

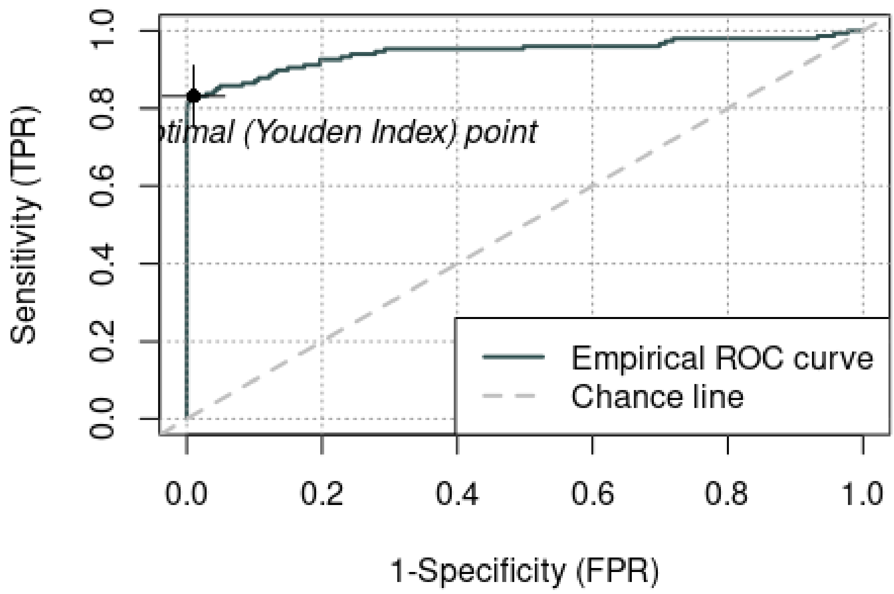
Training a logistic regression model based on these features: We will use entire dataset

```
log_reg_train = select(train,V17, V12, V16, V10, V11, V18, Class)
log_reg_test = select(test,V17, V12, V16, V10, V11, V18, Class)

model <- glm(Class ~ ., data = log_reg_train, family = 'binomial')

prediction <- predict(model, newdata = log_reg_test, type =
'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=log_reg_test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)
result <-
confusionMatrix(factor(prediction),factor(log_reg_test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85284    66
##          1    11    82
##
##                Accuracy : 0.9991
##                  95% CI : (0.9989, 0.9993)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 9.384e-11
##
##                   Kappa : 0.6801
##
##  Mcnemar's Test P-Value : 7.561e-10
##
##             Sensitivity : 0.9999
##             Specificity : 0.5541
##          Pos Pred Value : 0.9992
##          Neg Pred Value : 0.8817
##              Prevalence : 0.9983
##          Detection Rate : 0.9981
```

```
##    Detection Prevalence : 0.9989
##       Balanced Accuracy : 0.7770
##
##        'Positive' Class : 0
##
```

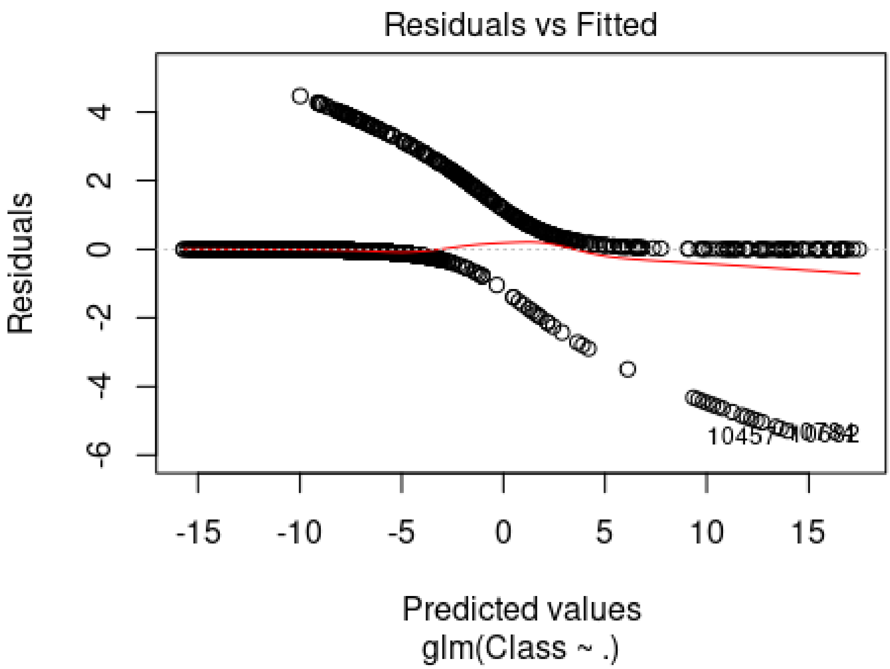
```
auc(log_reg_test$Class, prediction)
```

```
## Setting levels: control = 0, case = 1
```

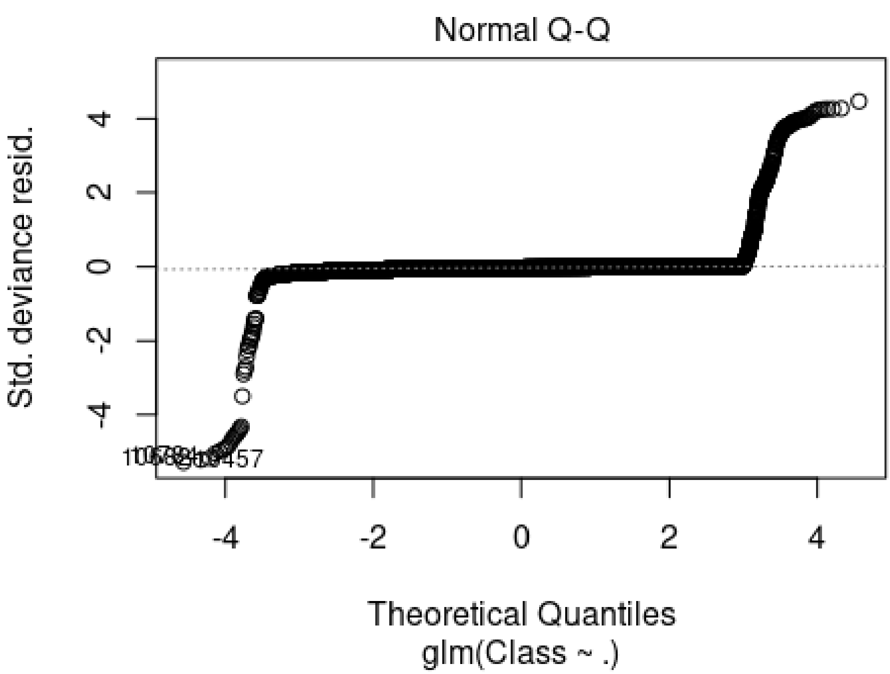
```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.777
```

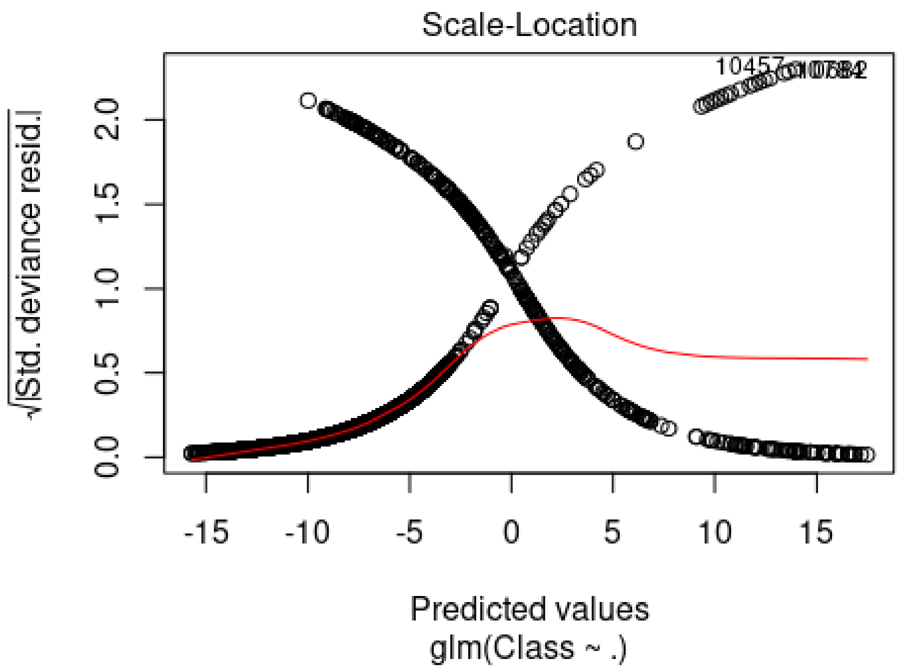
```
plot(model)
```

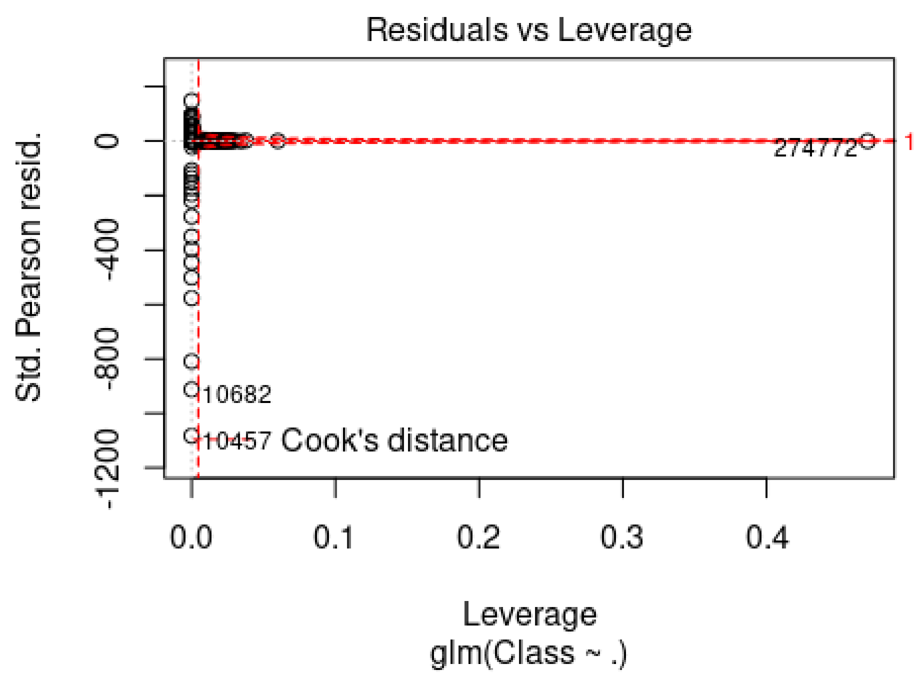


Residuals vs Fitted
Residuals
4
2
0
-2
-4
-6
10457
-15
-10
-5
0
5
10
15
Predicted values
glm(Class ~ .)



Normal Q-Q
Std. deviance resid.
4
2
0
-2
-4
10457
-4
-2
0
2
4
Theoretical Quantiles
glm(Class ~ .)






```
final <- append(final, prediction)
```

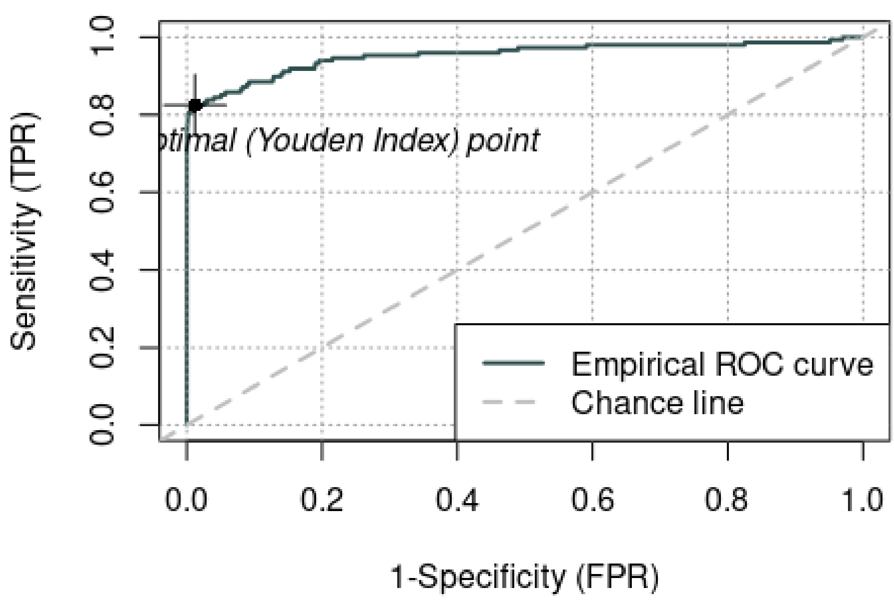
Clearly there is no further enhancement in the model using feature selection The kappa values decreased further so model not at all better in any way The model was able to perform similar to the normal model without variable importance

Trying logistic regression classifier with upsample data

```
log_reg_train = select(up_sample,V17, V12, V16, V10, V11, V18, Class)
log_reg_test = select(test,V17, V12, V16, V10, V11, V18, Class)
model <- glm(Class ~ ., data = log_reg_train, family = 'binomial')

prediction <- predict(model, newdata = log_reg_test, type =
'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=log_reg_test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)
result <-
confusionMatrix(factor(prediction),factor(log_reg_test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 80509    22
##          1  4786   126
```

```
##
##                Accuracy : 0.9437
##                  95% CI : (0.9422, 0.9453)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0466
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.94389
##             Specificity : 0.85135
##          Pos Pred Value : 0.99973
##          Neg Pred Value : 0.02565
##              Prevalence : 0.99827
##          Detection Rate : 0.94225
##    Detection Prevalence : 0.94251
##       Balanced Accuracy : 0.89762
##
##        'Positive' Class : 0
##
```

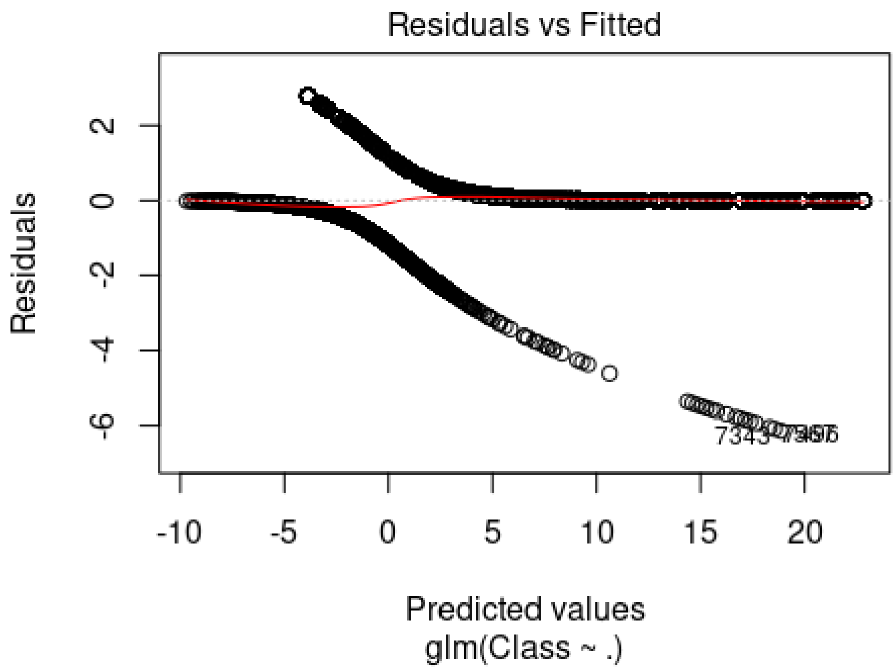
```
auc(log_reg_test$Class, prediction)
```

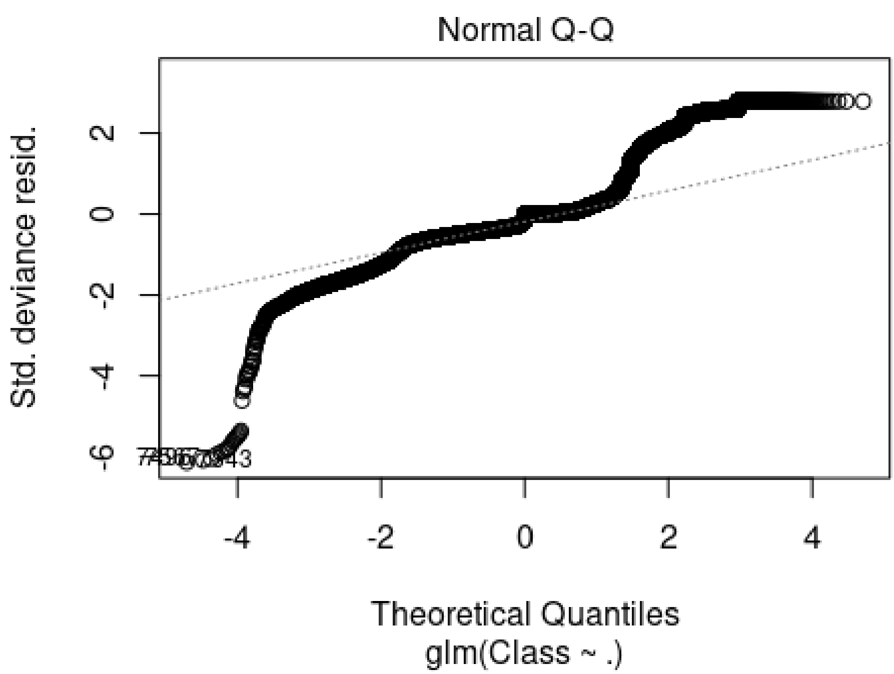
```
## Setting levels: control = 0, case = 1
```

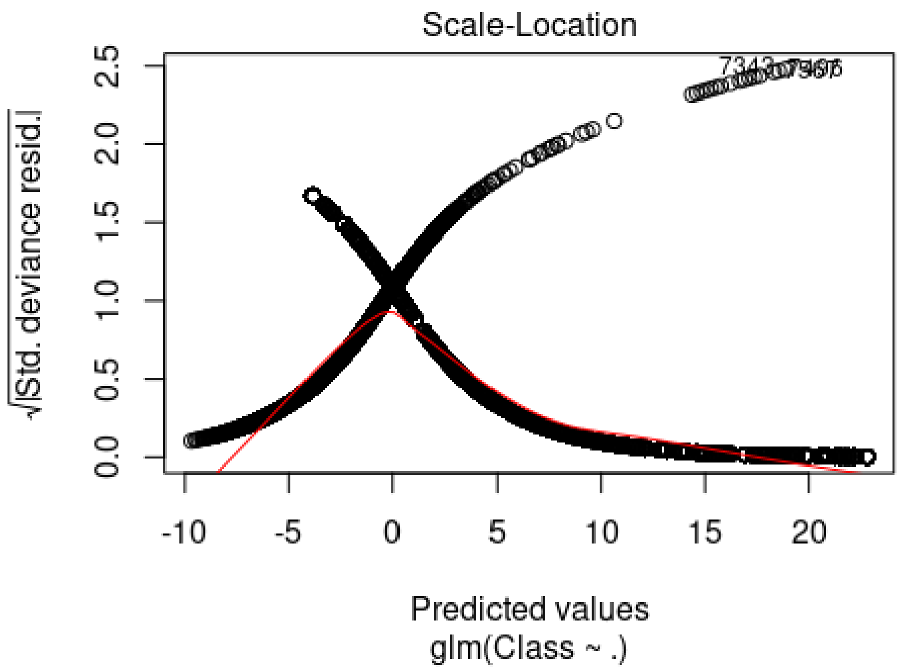
```
## Setting direction: controls < cases
```

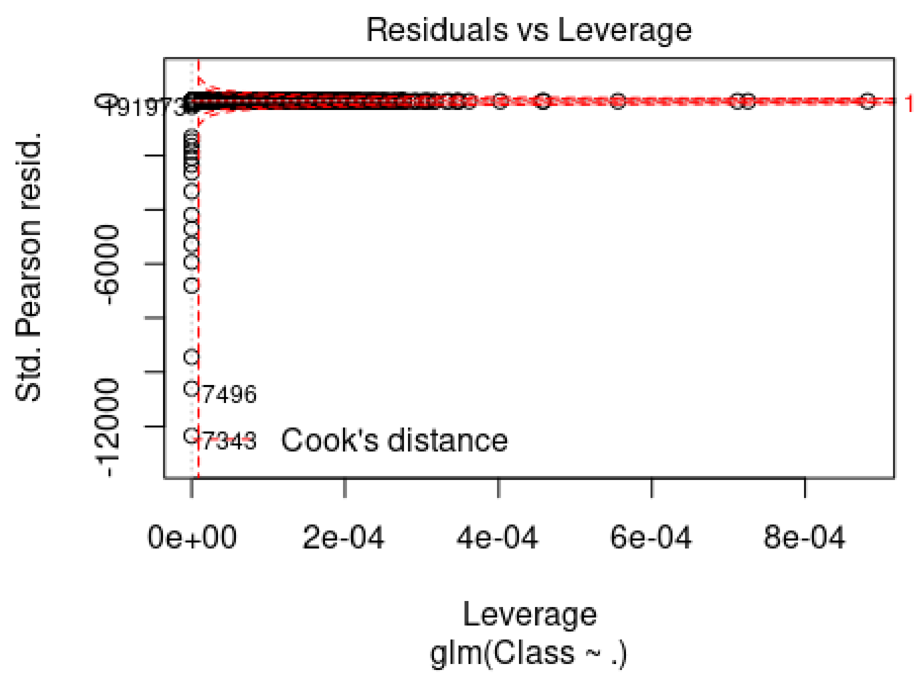
```
## Area under the curve: 0.8976
```

```
plot(model)
```


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
7343
-10
-5
0
5
10
15
20
-6
-4
-2
0
2


Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(Class ~ .)
-4
-2
0
2
4
-6
-4
-2
0
2






```
final <- append(final, prediction)
```

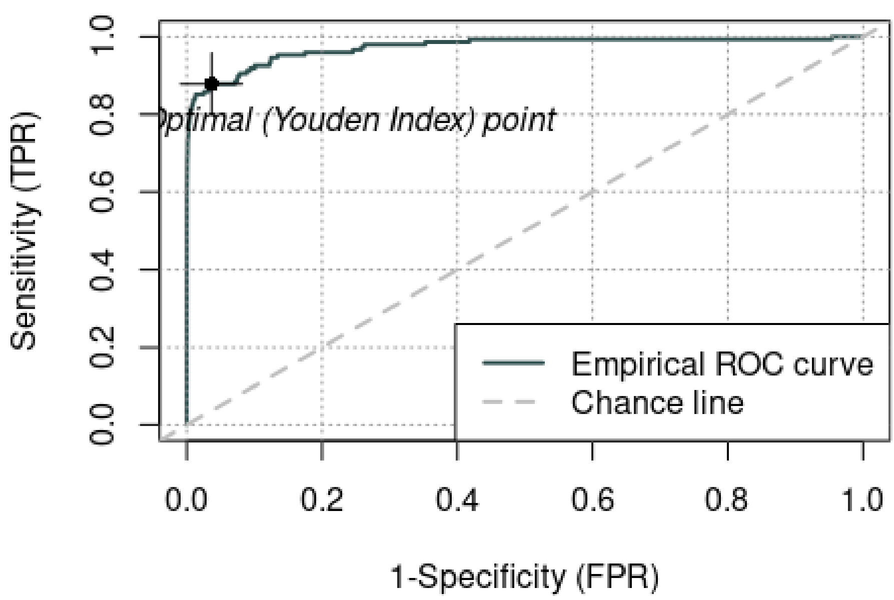
Worse performance kappa values bad not at all acceptable for credit card fraud detection The bias created by adding new samples equaling the Negative samples was large enough to create a more positive oriented model

We will now try the best features from the EDA and try to create a model for logistic regression, with up sample

```
log_reg_train = select(up_sample,V4,V11,V12, Class)
log_reg_test = select(test,V4, V11, V12, Class)
model <- glm(Class ~ ., data = log_reg_train, family = 'binomial')

prediction <- predict(model, newdata = log_reg_test, type =
'response')
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=log_reg_test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)
result <-
confusionMatrix(factor(prediction),factor(log_reg_test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 81054    18
```

```
##          1  4241   130
##
##                Accuracy : 0.9502
##                  95% CI : (0.9487, 0.9516)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0544
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.95028
##             Specificity : 0.87838
##          Pos Pred Value : 0.99978
##          Neg Pred Value : 0.02974
##              Prevalence : 0.99827
##          Detection Rate : 0.94863
##    Detection Prevalence : 0.94884
##       Balanced Accuracy : 0.91433
##
##        'Positive' Class : 0
##
```

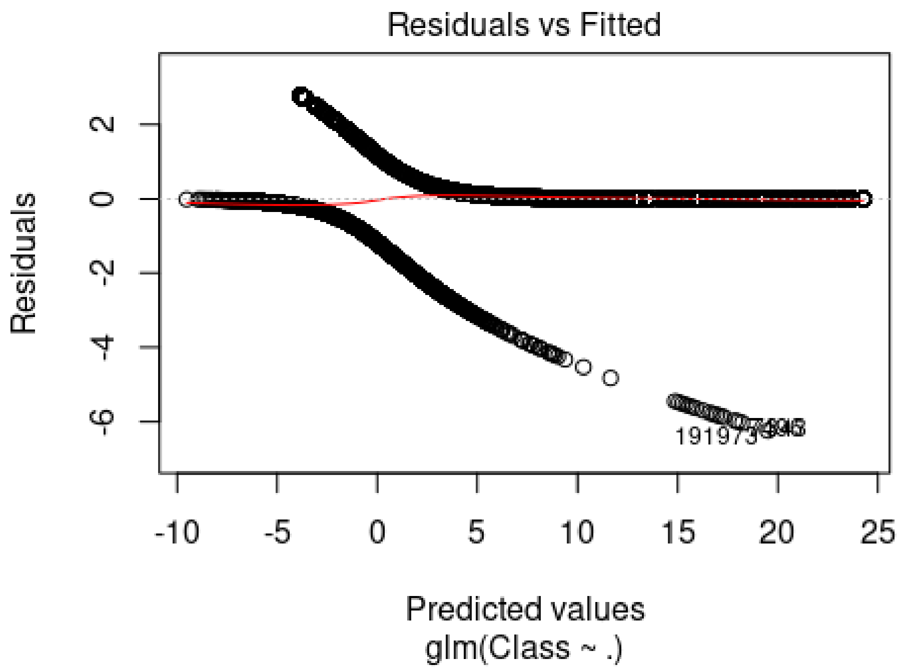
```
auc(log_reg_test$Class, prediction)
```

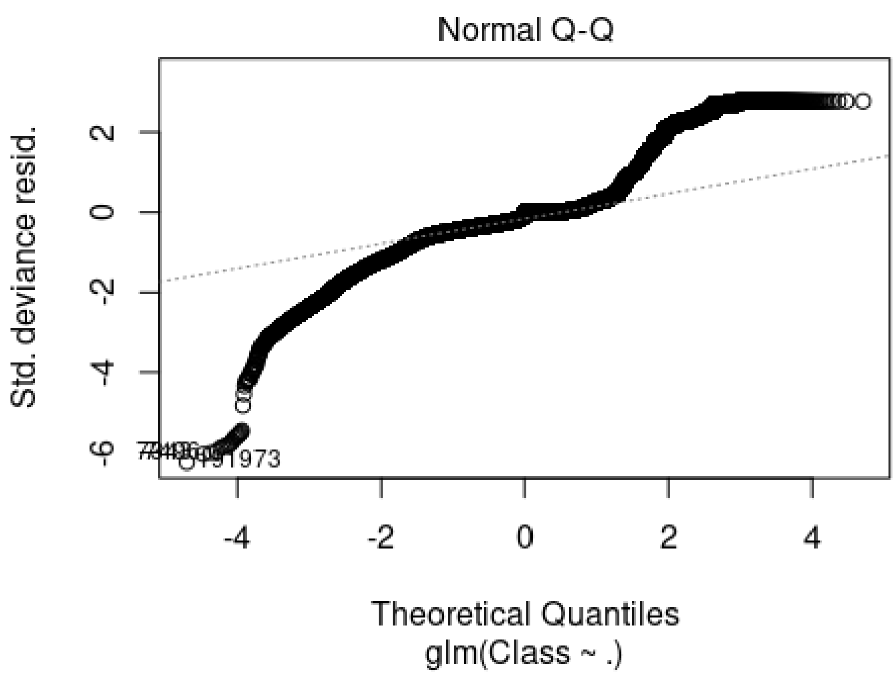
```
## Setting levels: control = 0, case = 1
```

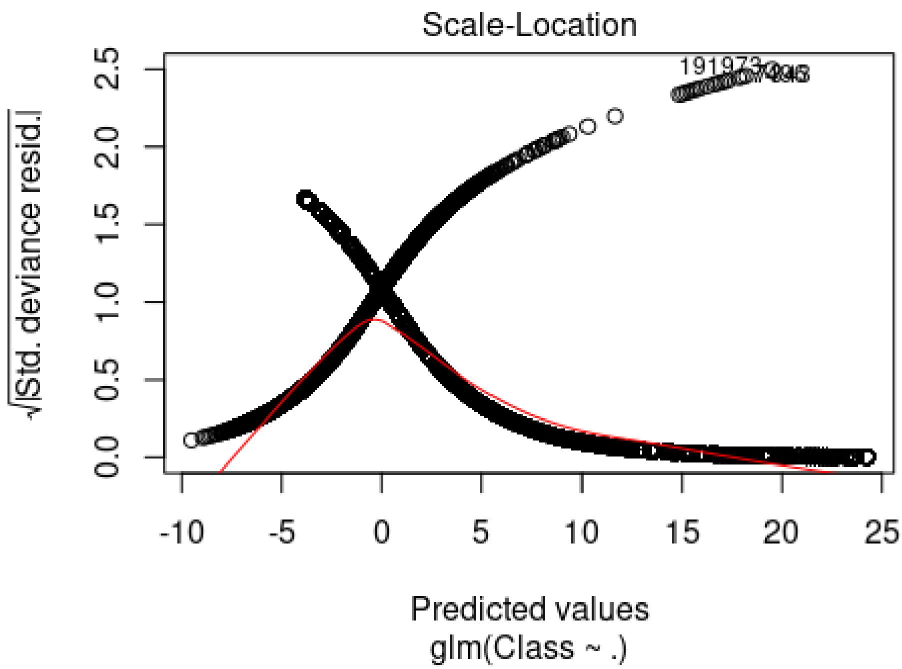
```
## Setting direction: controls < cases
```

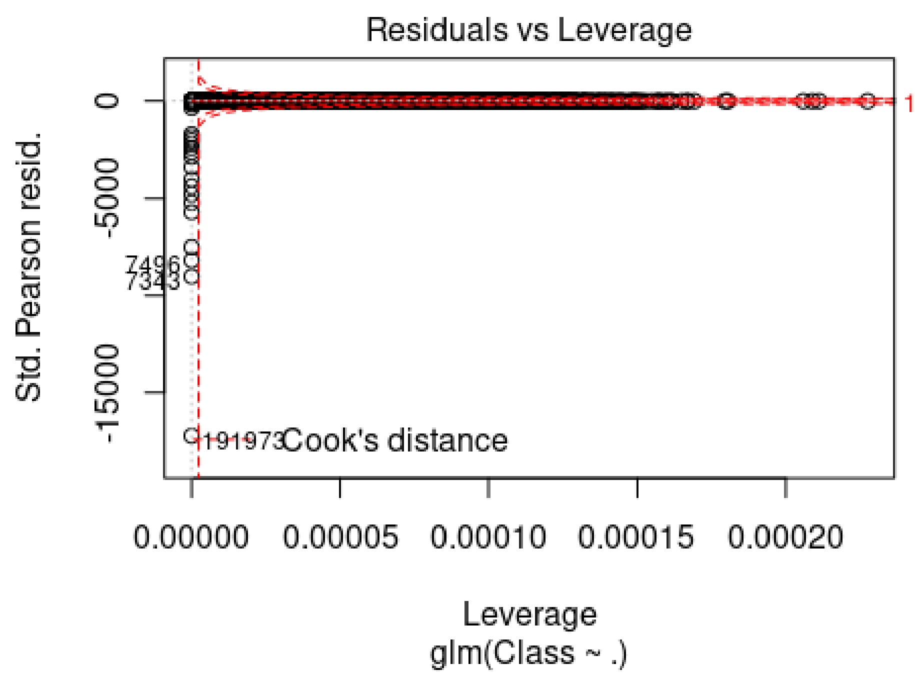
```
## Area under the curve: 0.9143
```

```
plot(model)
```


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
191973


Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(Class ~ .)
191973





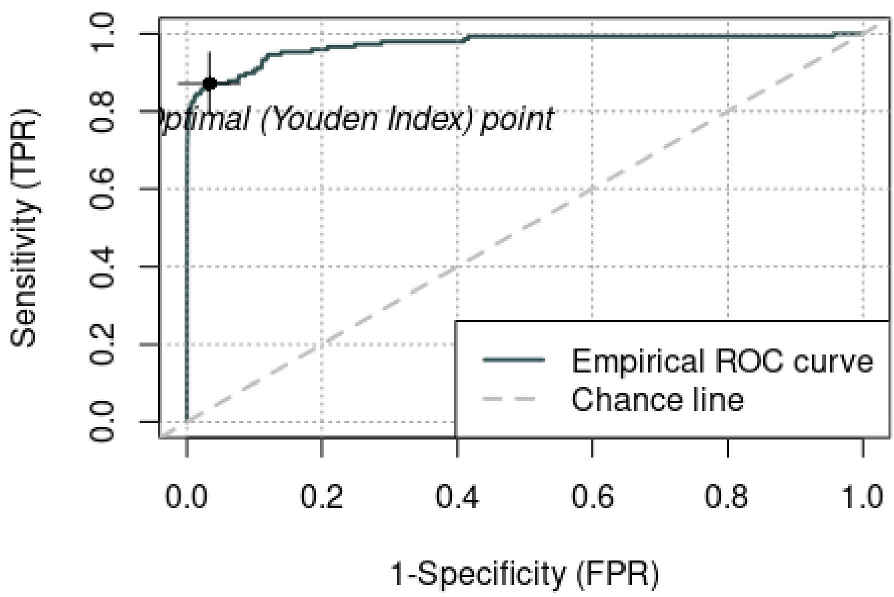

```
final <- append(final, prediction)
```

Still lacks the ability to make a good model so we will try with the entire dataset Trying with the EDA samples it is noticed that not much of difference occured except for the fact that FN decreased a bit indicating that the features selected from EDA have a good potiential to classify the model

```
log_reg_train = select(train,V4,V11,V12, Class)
log_reg_test = select(test,V4, V11, V12, Class)
model <- glm(Class ~ ., data = log_reg_train, family = 'binomial')

prediction <- predict(model, newdata = log_reg_test)
#roc.curve(test$Class, prediction, plotit = TRUE)

plot(rocit(score=prediction,class=log_reg_test$Class))
```




```
prediction <- ifelse(prediction > 0.5, 1, 0)
result <-
confusionMatrix(factor(prediction),factor(log_reg_test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85282    82
##          1    13    66
##
##                Accuracy : 0.9989
##                  95% CI : (0.9986, 0.9991)
##     No Information Rate : 0.9983
```

```
##     P-Value [Acc > NIR] : 2.045e-06
##
##                   Kappa : 0.581
##
##  Mcnemar's Test P-Value : 3.023e-12
##
##             Sensitivity : 0.9998
##             Specificity : 0.4459
##          Pos Pred Value : 0.9990
##          Neg Pred Value : 0.8354
##              Prevalence : 0.9983
##          Detection Rate : 0.9981
##    Detection Prevalence : 0.9991
##       Balanced Accuracy : 0.7229
##
##        'Positive' Class : 0
##
```

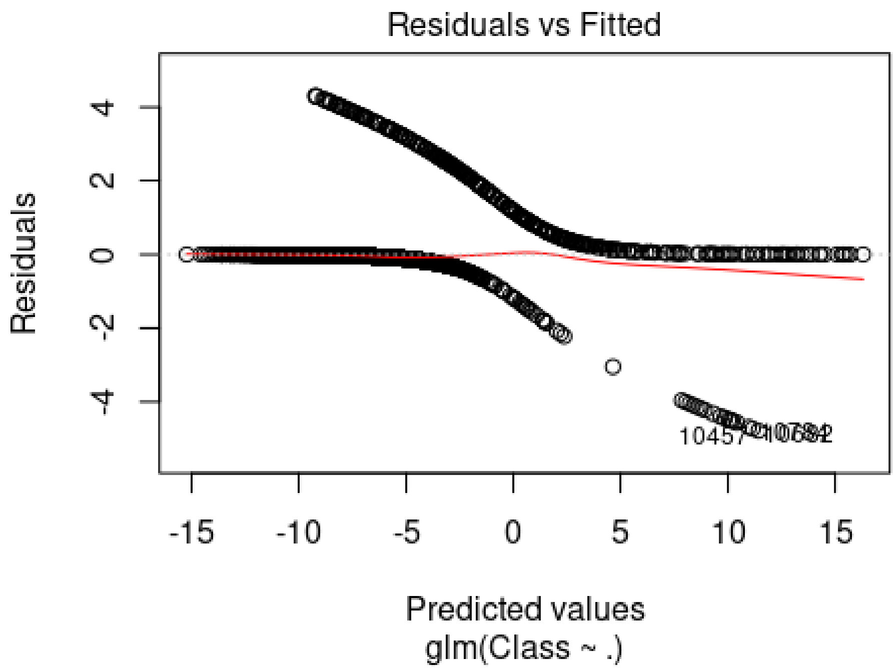
```
auc(log_reg_test$Class, prediction)
```

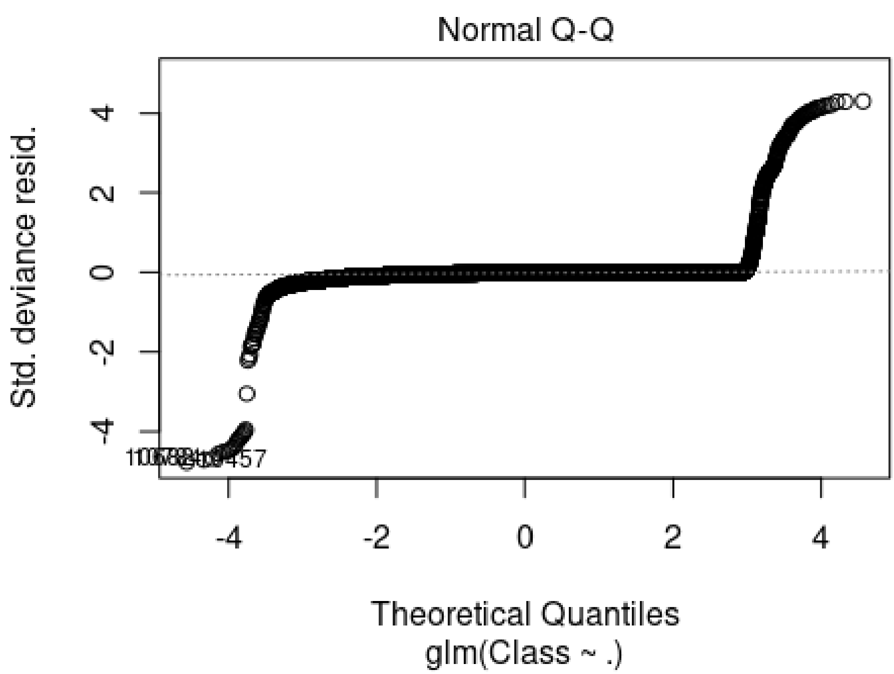
```
## Setting levels: control = 0, case = 1
```

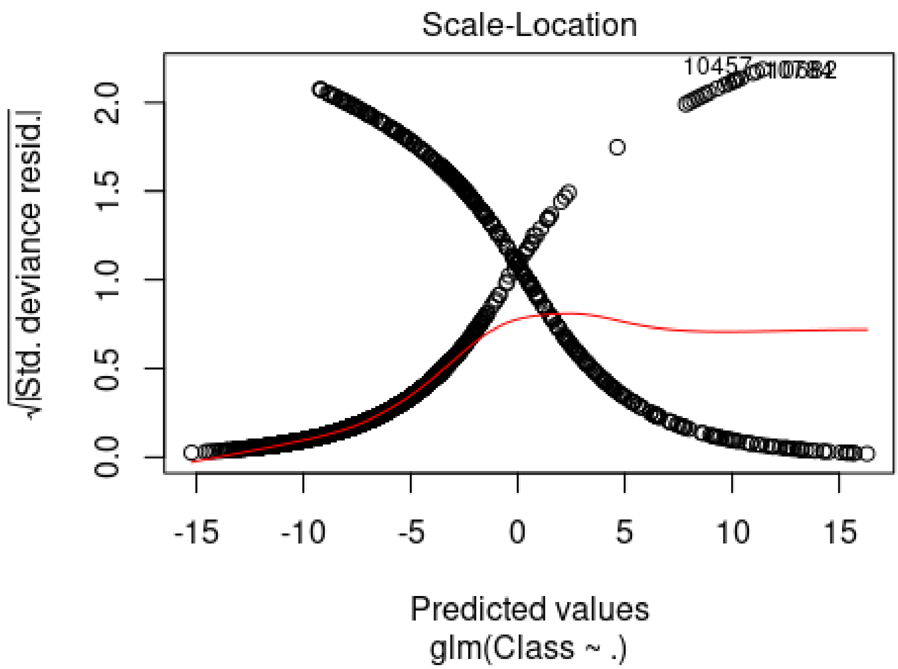
```
## Setting direction: controls < cases
```

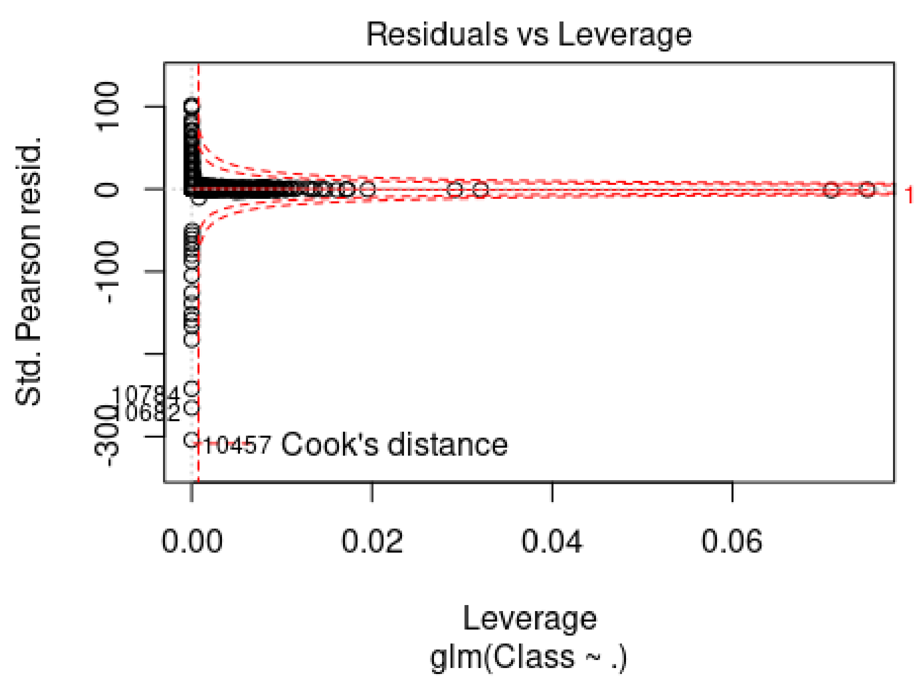
```
## Area under the curve: 0.7229
```

```
plot(model)
```


Residuals vs Fitted
Residuals
4
2
0
-2
-4
10457
-15
-10
-5
0
5
10
15
Predicted values
glm(Class ~ .)


Normal Q-Q
Std. deviance resid.
4
2
0
-2
-4
-4
-2
0
2
4
Theoretical Quantiles
glm(Class ~ .)






```
final <- append(final, prediction)
```

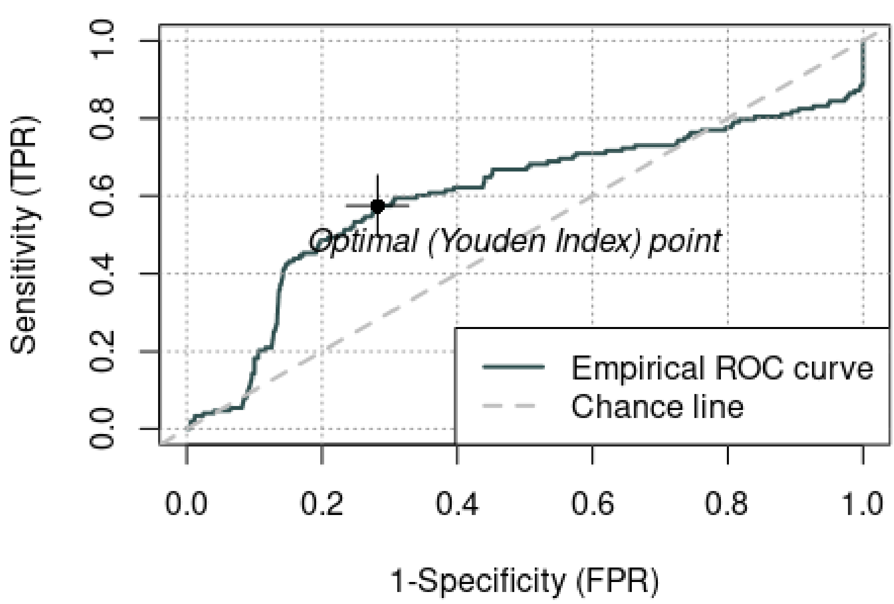
The model performed well and can be used, however the FN is still too high for realistic use The performance was amazing indeed however the bias due to having a lot of negative samples is still causing the model to be negative oriented

Creating RBORST Random Forest trees for finding fraud

```
x = up_sample[, -30]
y = up_sample[,30]

rf_fit <- Rborist(x, y, ntree = 1000, minNode = 20, maxLeaf = 13)


rf_pred <- predict(rf_fit, test[,-30], ctgCensus = "prob")
prob <- rf_pred$yPred
plot(rocit(score=prob,class=test$Class))
```




```
prediction <- ifelse(prob > 50, 1, 0)
result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 42015    47
##          1 43280   101
##
```

```
##                Accuracy : 0.4929
##                  95% CI : (0.4896, 0.4963)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0012
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.492585
##             Specificity : 0.682432
##          Pos Pred Value : 0.998883
##          Neg Pred Value : 0.002328
##              Prevalence : 0.998268
##          Detection Rate : 0.491731
##    Detection Prevalence : 0.492281
##       Balanced Accuracy : 0.587508
##
##        'Positive' Class : 0
##
```

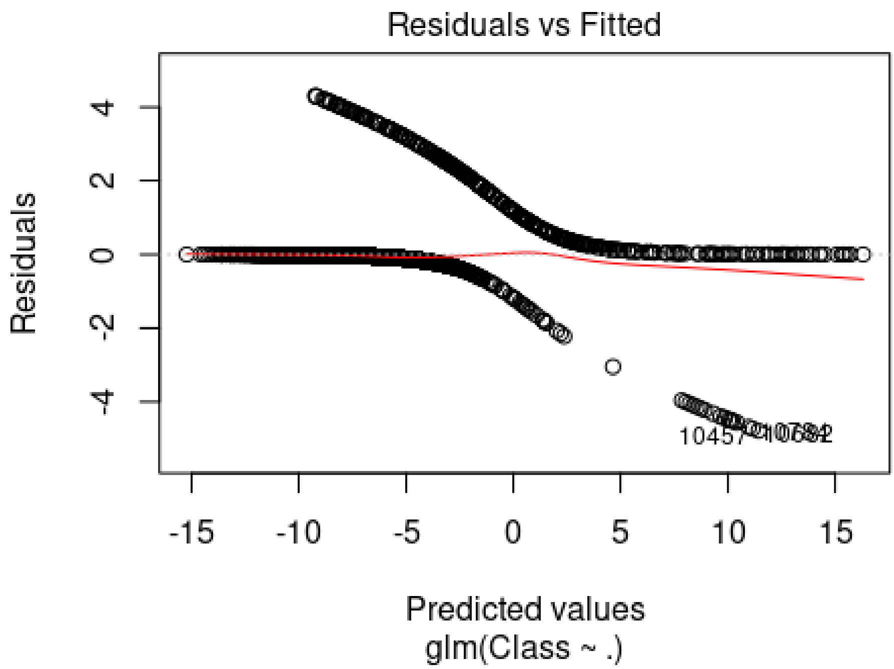
```
auc(test$Class, prediction)
```

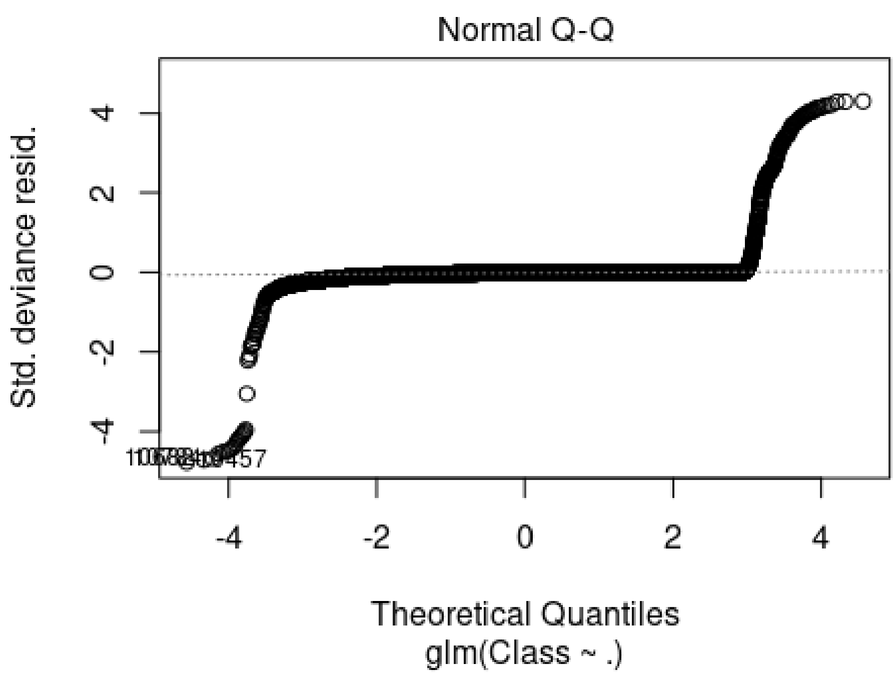
```
## Setting levels: control = 0, case = 1
```

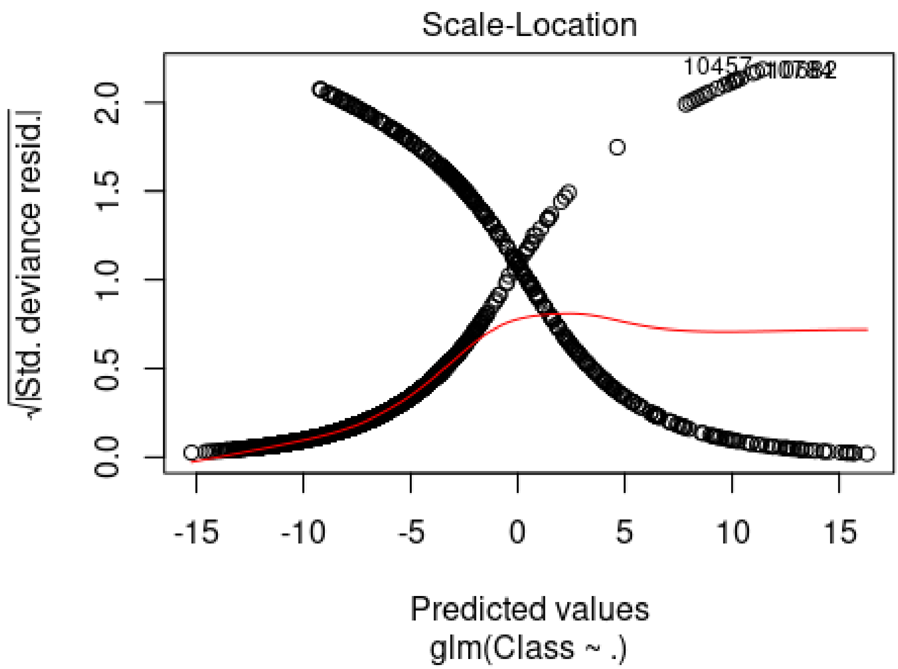
```
## Setting direction: controls < cases
```

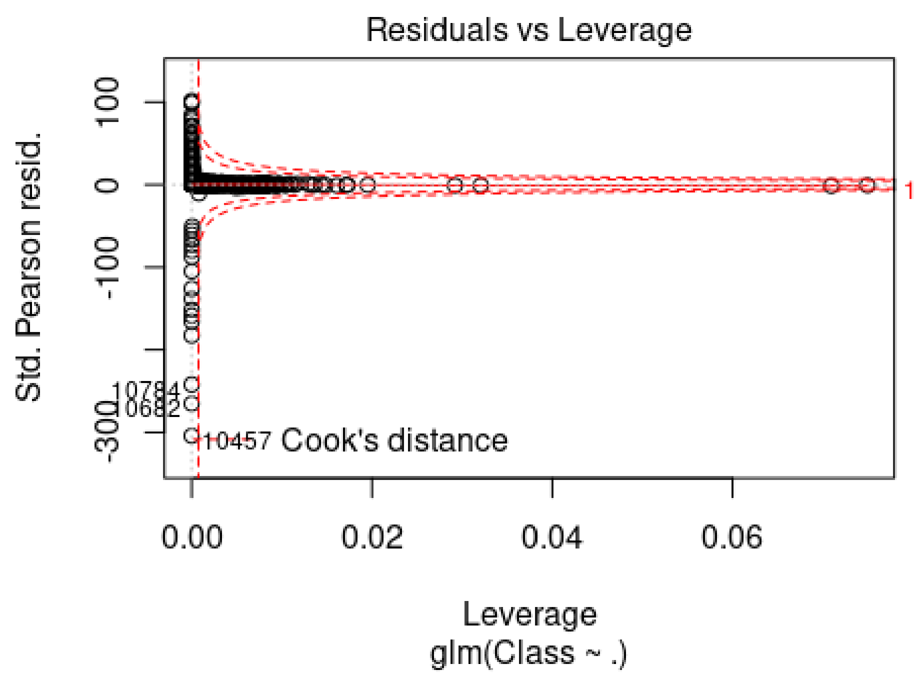
```
## Area under the curve: 0.5875
```

```
plot(model)
```


Residuals vs Fitted
Residuals
4
2
0
-2
-4
10457
10784
-15
-10
-5
0
5
10
15
Predicted values
glm(Class ~ .)


Normal Q-Q
Std. deviance resid.
4
2
0
-2
-4
10457
-4
-2
0
2
4
Theoretical Quantiles
glm(Class ~ .)





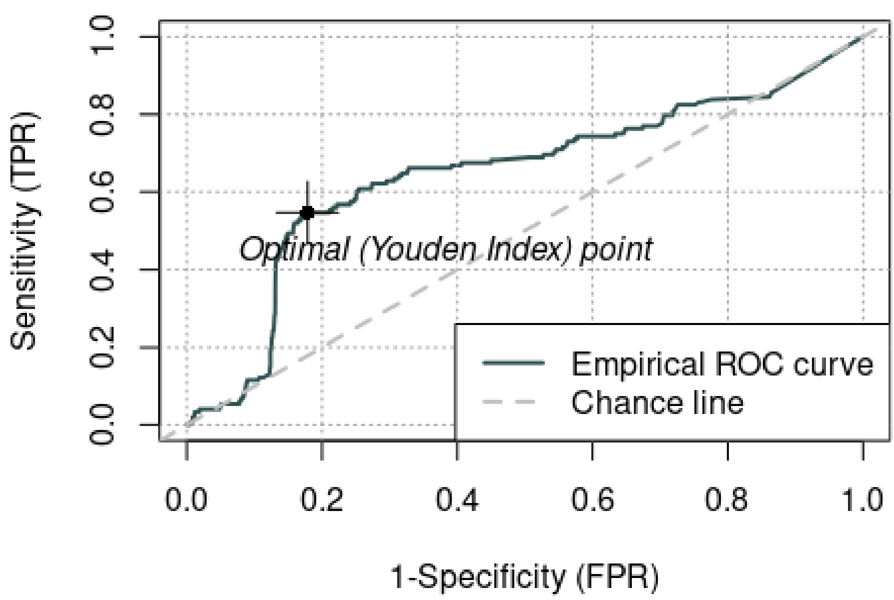

```
final <- append(final, prediction)
```

The FN are low but the FP are amost equal to TN which is concerning and model will not work The RBORST random forests might seem a good option but they are alwas bad at classification with highly unbalanced data, this can be easily observed by the fact that model was not able to bring a good classification boundary even at very complex forest Trying a smaaler forest

```
x = up_sample[, -30]
y = up_sample[,30]

rf_fit <- Rborist(x, y, ntree = 100, minNode = 5, maxLeaf = 4)


rf_pred <- predict(rf_fit, test[,-30], ctgCensus = "prob")
prob <- rf_pred$yPred
plot(rocit(score=prob,class=test$Class))
```




```
prediction <- ifelse(prob > 50, 1, 0)
result <- confusionMatrix(factor(prediction),factor(test$Class))

## Warning in confusionMatrix.default(factor(prediction),
factor(test$Class)):
## Levels are not in the same order for reference and data.
Refactoring data to
## match.

result

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction     0     1
##          0     0     0
##          1 85295   148
##
##                Accuracy : 0.0017
##                  95% CI : (0.0015, 0.002)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.000000
##             Specificity : 1.000000
##          Pos Pred Value :      NaN
##          Neg Pred Value : 0.001732
##              Prevalence : 0.998268
##          Detection Rate : 0.000000
##    Detection Prevalence : 0.000000
##       Balanced Accuracy : 0.500000
##
##        'Positive' Class : 0
##
```

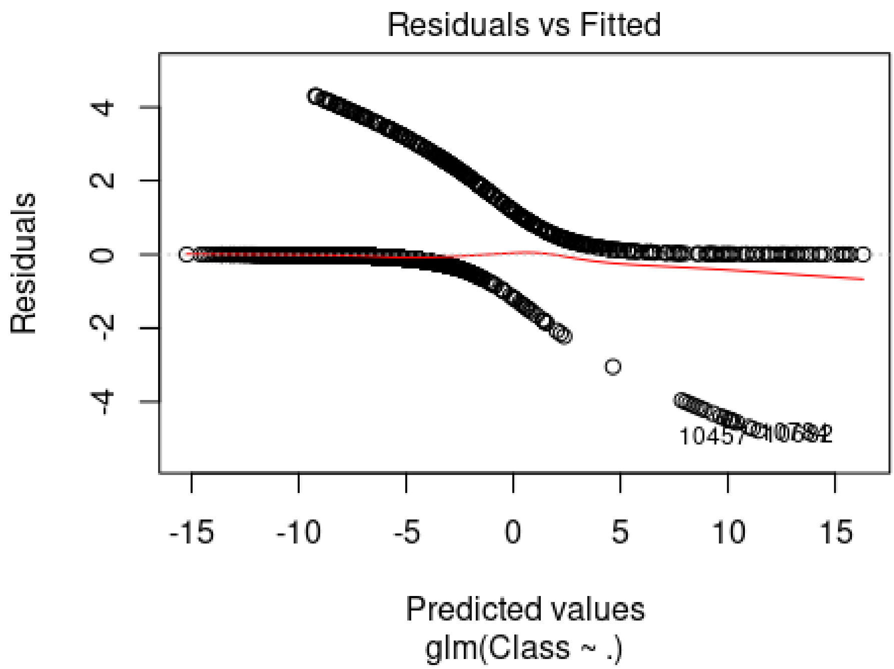
```
auc(test$Class, prediction)
```

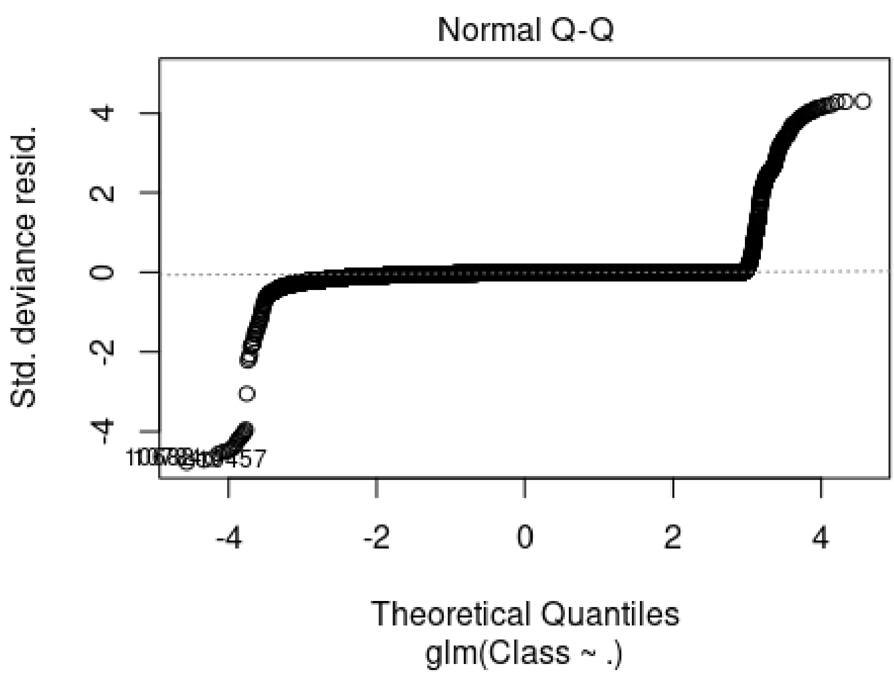
```
## Setting levels: control = 0, case = 1
```

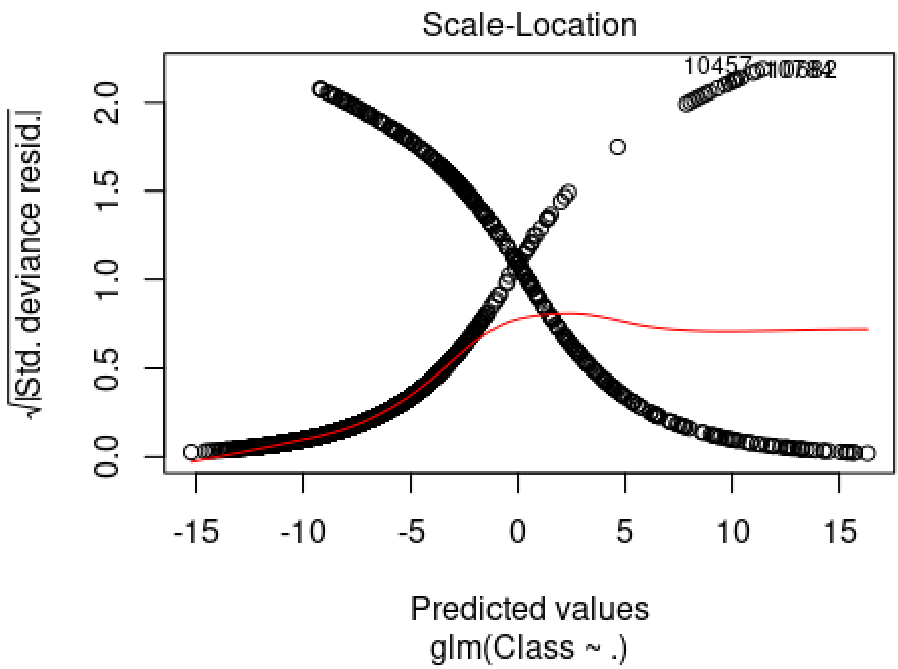
```
## Setting direction: controls < cases
```

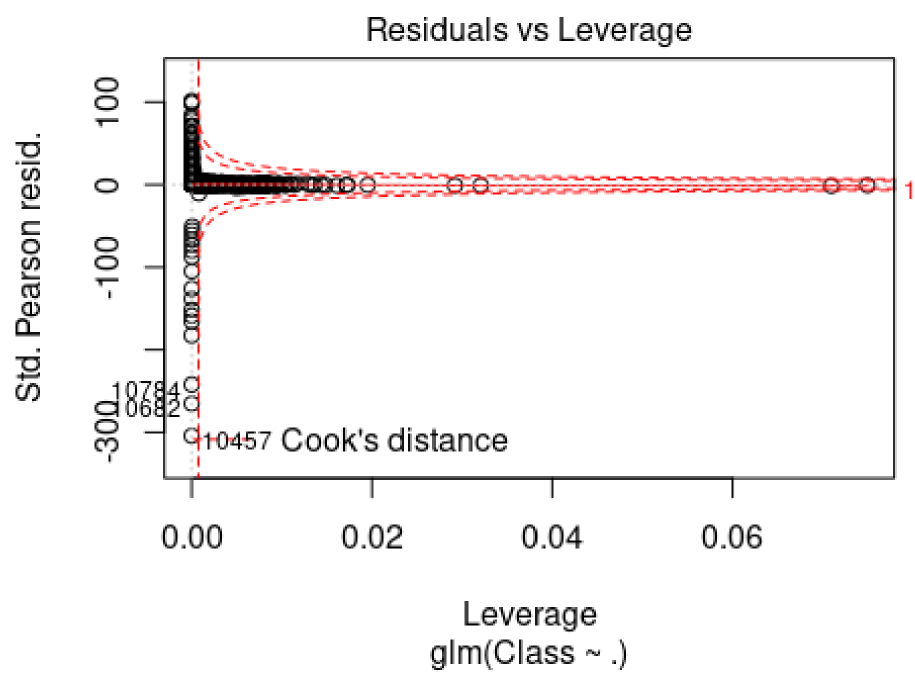
```
## Area under the curve: 0.5
```

```
plot(model)
```


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
10457


Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(Class ~ .)






```
final <- append(final, prediction)
```

Still the same problem cannot be accepted but enhanced FN

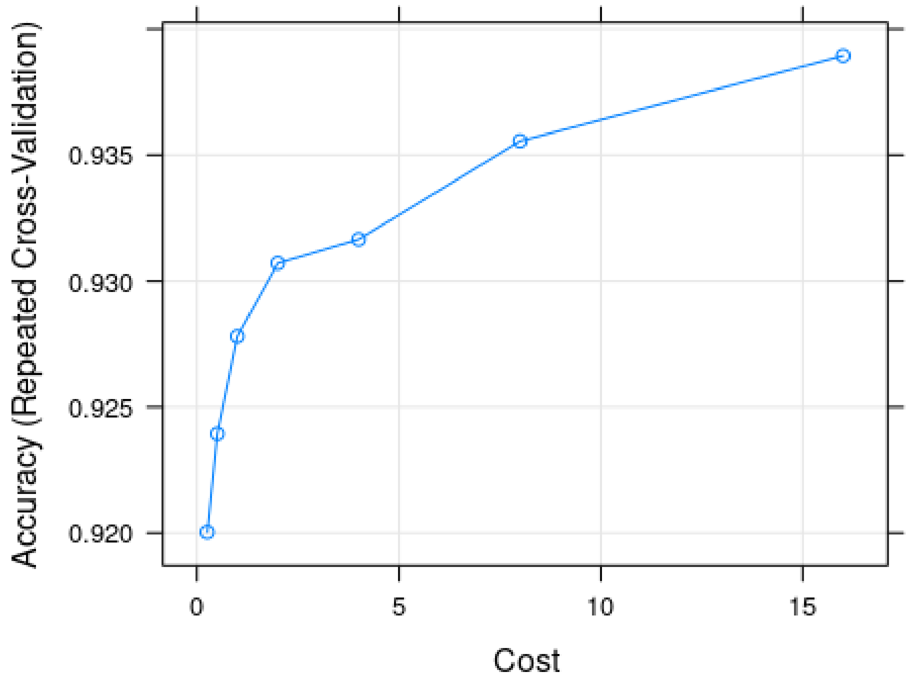
The model with 100 forest was just too simple for fitting in 25 features and making decision hence classifing everythin as positive this is a main problem with Random forest where unbalanced data cannot give any sort of good result with simple forest approach It was also noticed that AUC and ROC are not good indicators for this particular case

Trying SVM Radial for with down sample due to constrains on the computer which im working on this is only one possible The model can give good results as SVM is good with multiple dimensions

```
model <- train(Class ~ ., data = down_sample, method = 'svmRadial',
trControl = trainControl(method = 'repeatedcv', number = 10, repeats =
3),tuneLength = 7)
prediction <- predict(model, newdata = test)
result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 80250    10
##          1  5045   138
##
##                Accuracy : 0.9408
##                  95% CI : (0.9392, 0.9424)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0486
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.94085
##             Specificity : 0.93243
##          Pos Pred Value : 0.99988
##          Neg Pred Value : 0.02663
##              Prevalence : 0.99827
##          Detection Rate : 0.93922
##    Detection Prevalence : 0.93934
##       Balanced Accuracy : 0.93664
##
##        'Positive' Class : 0
##

plot(model)
```



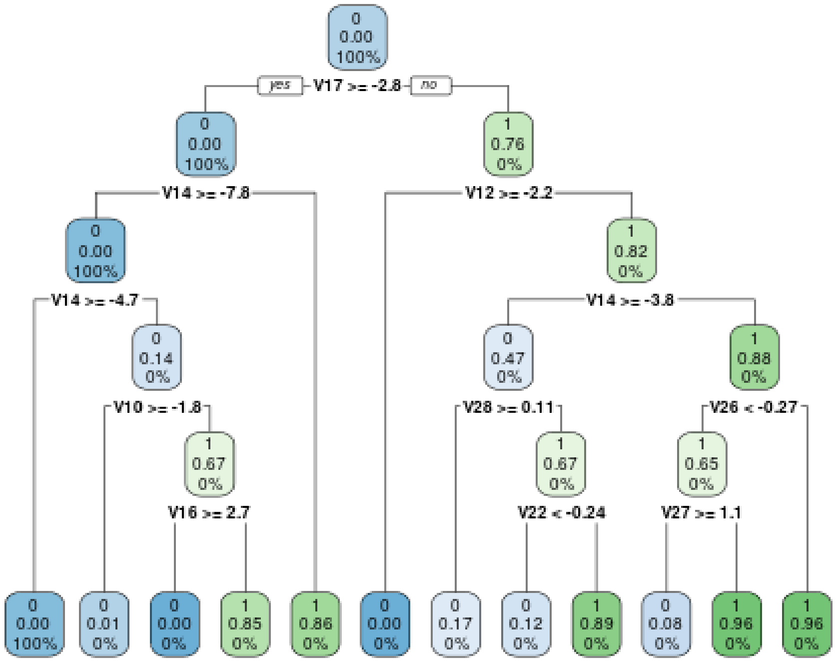

```
final <- append(final, prediction)
```

The results were good especially the FN at down sample are good but still too high FP Not acceptable Radial SVM was selected for the following reasons: Good with high dimensional classification Good in cases where there are clear boundaries seperating the positives and negatives Memory efficiency as the models are trained for low specs ATM machines

Problems Encountered: Dataset was large and highly unbalanced even when down sampling was done. Target classes are overlapping as was noticed in EDA for most of the features This caused the SVM to perform badly in FP cases however that being true it was notied that the FN were very less which means that there is clear classification boundary between positives and negatives with some features and these features we will target

While the Fn increased a bit the FP almost one third still not accetable We will try decision Tree models

```
decisionTree <- rpart(Class ~ . , train, method = 'class')
prediction <- predict(decisionTree, test, type = 'class')
probability <- predict(decisionTree, test, type = 'prob')
rpart.plot(decisionTree)
```



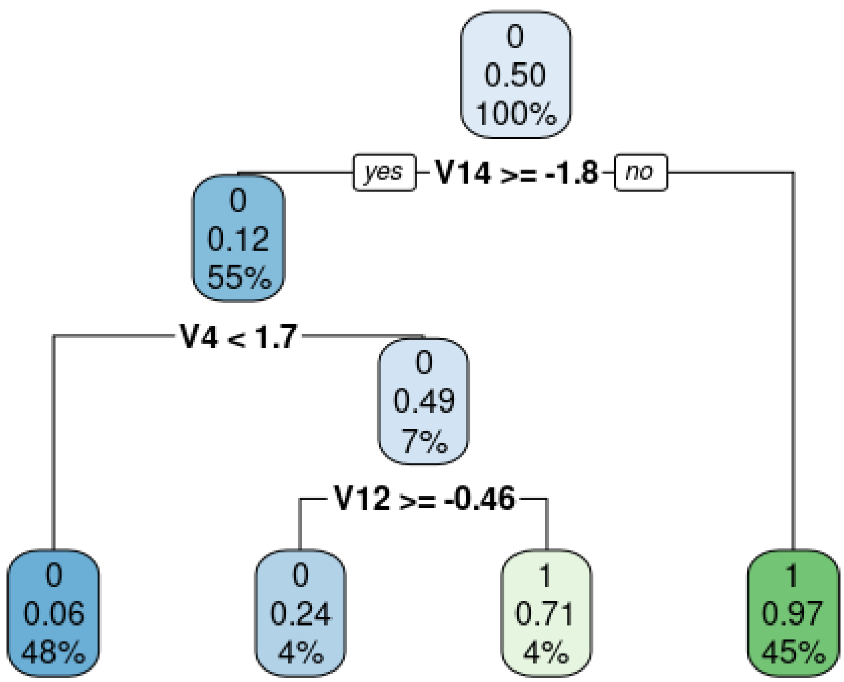

```
confMat <- confusionMatrix(prediction, test$Class)
confMat

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85285    33
##          1    10   115
##
##                Accuracy : 0.9995
##                  95% CI : (0.9993, 0.9996)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8422
##
##  Mcnemar's Test P-Value : 0.0007937
##
##             Sensitivity : 0.9999
##             Specificity : 0.7770
##          Pos Pred Value : 0.9996
##          Neg Pred Value : 0.9200
##              Prevalence : 0.9983
##          Detection Rate : 0.9982
##    Detection Prevalence : 0.9985
##       Balanced Accuracy : 0.8885
```

```
##
##        'Positive' Class : 0
##
```

```
final <- append(final, prediction)
```

```
decisionTree <- rpart(Class ~ . , up_sample, method = 'class')
prediction <- predict(decisionTree, test, type = 'class')
probability <- predict(decisionTree, test, type = 'prob')
rpart.plot(decisionTree)
```



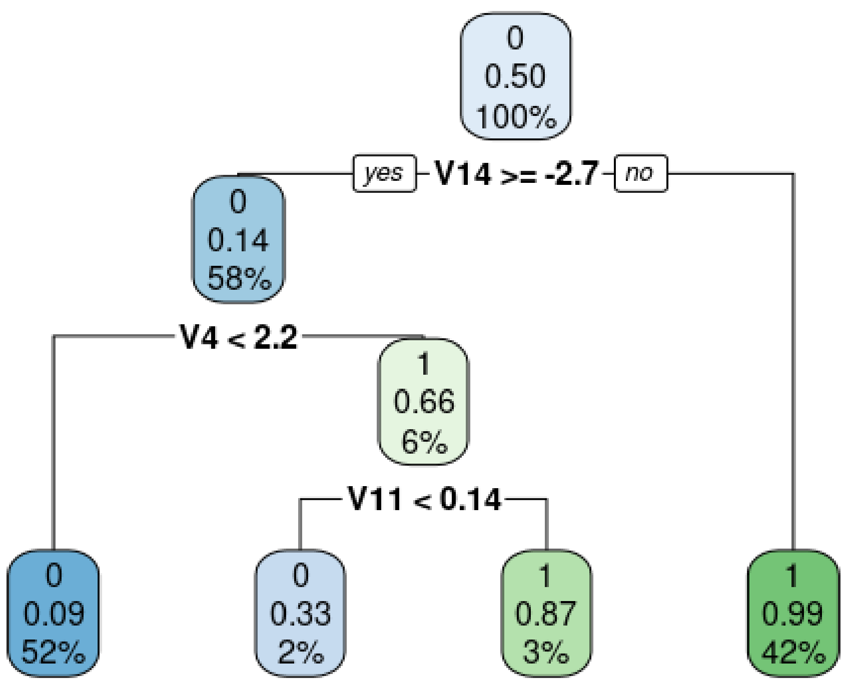

```
confMat <- confusionMatrix(prediction, test$Class)
confMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 80761    11
##          1  4534   137
##
##                Accuracy : 0.9468
##                  95% CI : (0.9453, 0.9483)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0537
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.94684
##             Specificity : 0.92568
##          Pos Pred Value : 0.99986
##          Neg Pred Value : 0.02933
##              Prevalence : 0.99827
##          Detection Rate : 0.94520
##    Detection Prevalence : 0.94533
##       Balanced Accuracy : 0.93626
##
##        'Positive' Class : 0
##
```

```
final <- append(final, prediction)
```

```
decisionTree <- rpart(Class ~ . , down_sample, method = 'class')
prediction <- predict(decisionTree, test, type = 'class')
probability <- predict(decisionTree, test, type = 'prob')
rpart.plot(decisionTree)
```




```
confMat <- confusionMatrix(prediction, test$Class)
confMat
```

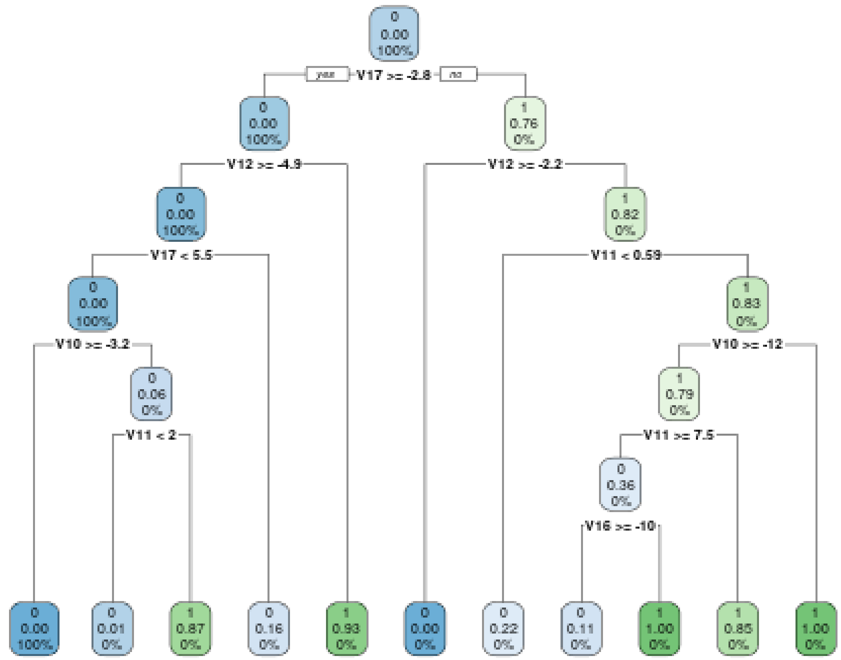
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
```

```
##           0 82808    16
##           1  2487   132
##
##                Accuracy : 0.9707
##                  95% CI : (0.9696, 0.9718)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0924
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9708
##             Specificity : 0.8919
##          Pos Pred Value : 0.9998
##          Neg Pred Value : 0.0504
##              Prevalence : 0.9983
##          Detection Rate : 0.9692
##    Detection Prevalence : 0.9693
##       Balanced Accuracy : 0.9314
##
##        'Positive' Class : 0
##

final <- append(final, prediction)
```

trying with feature importance

```
log_reg_train = select(train,V17, V12, V16, V10, V11, V18, Class)
log_reg_test = select(test,V17, V12, V16, V10, V11, V18, Class)
decisionTree <- rpart(Class ~ . , log_reg_train, method = 'class')
prediction <- predict(decisionTree, log_reg_test, type = 'class')
probability <- predict(decisionTree, log_reg_test, type = 'prob')
rpart.plot(decisionTree)
```



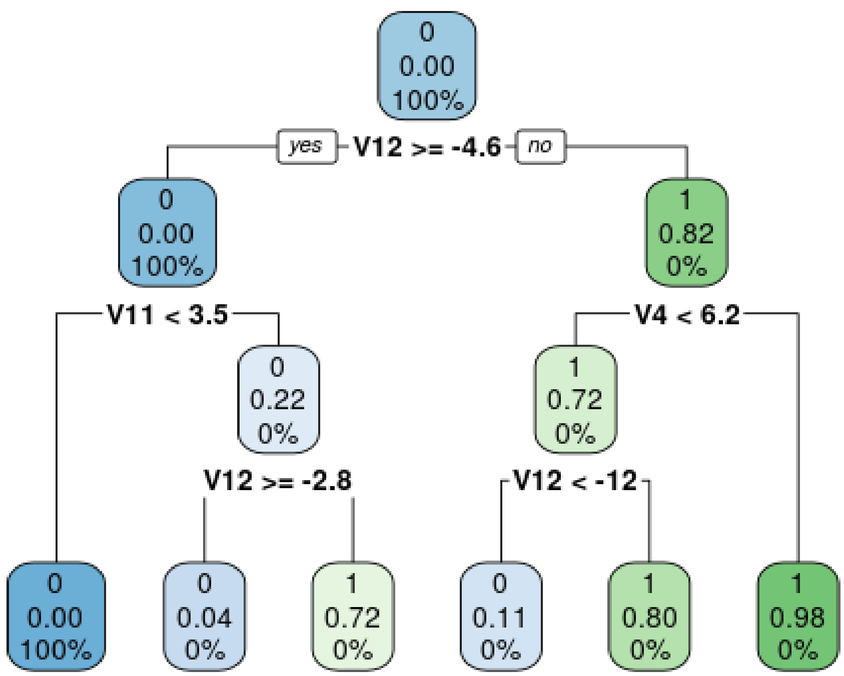

```
confMat <- confusionMatrix(prediction, test$Class)
confMat

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85281    39
##          1    14   109
##
##                Accuracy : 0.9994
##                  95% CI : (0.9992, 0.9995)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8041
##
##  Mcnemar's Test P-Value : 0.0009784
##
##             Sensitivity : 0.9998
##             Specificity : 0.7365
##          Pos Pred Value : 0.9995
##          Neg Pred Value : 0.8862
##              Prevalence : 0.9983
##          Detection Rate : 0.9981
##    Detection Prevalence : 0.9986
##       Balanced Accuracy : 0.8682
```

```
##
##        'Positive' Class : 0
##
```

trying with EDA selected

```
log_reg_train <- select(train,V4,V11,V12, Class)
log_reg_test <- select(test,V4, V11, V12, Class)
decisionTree <- rpart(Class ~ . , log_reg_train, method = 'class')
prediction <- predict(decisionTree, log_reg_test, type = 'class')
probability <- predict(decisionTree, log_reg_test, type = 'prob')
rpart.plot(decisionTree)
```




```
confMat <- confusionMatrix(prediction, test$Class)
confMat

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85278    55
##          1    17    93
##
##                Accuracy : 0.9992
##                  95% CI : (0.9989, 0.9993)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 2.922e-12
##
```

```
##                  Kappa : 0.7205
##
##  Mcnemar's Test P-Value : 1.298e-05
##
##             Sensitivity : 0.9998
##             Specificity : 0.6284
##          Pos Pred Value : 0.9994
##          Neg Pred Value : 0.8455
##              Prevalence : 0.9983
##          Detection Rate : 0.9981
##    Detection Prevalence : 0.9987
##       Balanced Accuracy : 0.8141
##
##        'Positive' Class : 0
##
```

```
final <- append(final, prediction)
```

Trying KNN with feature importance, KNN on real dataset will be impossible due to KNN being bad with multiple dimension handling

```
log_reg_train = select(train,V17, V12, V16, V10, V11, V18, Class)
log_reg_test = select(test,V17, V12, V16, V10, V11, V18, Class)
prediction <- knn(log_reg_train, log_reg_test, log_reg_train$Class, k
=5)
result <- confusionMatrix(factor(prediction),factor(test$Class))
result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85289    31
##          1     6   117
##
##                Accuracy : 0.9996
##                  95% CI : (0.9994, 0.9997)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8633
##
##  Mcnemar's Test P-Value : 7.961e-05
##
##             Sensitivity : 0.9999
##             Specificity : 0.7905
##          Pos Pred Value : 0.9996
##          Neg Pred Value : 0.9512
##              Prevalence : 0.9983
##          Detection Rate : 0.9982
##    Detection Prevalence : 0.9986
```

```
##       Balanced Accuracy : 0.8952
##
##        'Positive' Class : 0
##
```

```
final <- append(final, prediction)
```

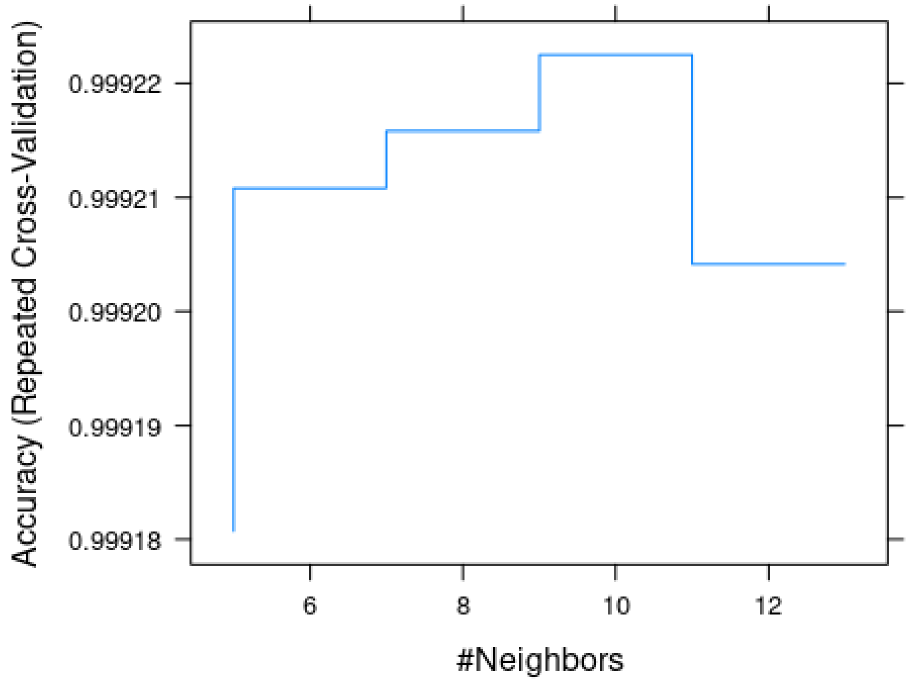
It was obseved that the data was amazingly identified by decision trees however that being true due to a large imbalance in the data set the model still was not able to understand the concept of fraud correctly as there is a huge bias towards the not fraud data

With newfound confidence in KNN trying to find a good model for EDA dataset

```
log_reg_train <- select(train,V4,V11,V12, Class)
log_reg_test <- select(test,V4, V11, V12, Class)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
#,classProbs=TRUE,summaryFunction = twoClassSummary)
knn_model <- train(Class ~ ., data = log_reg_train, method = "knn",
trControl = ctrl, preProcess = c("center","scale"), tuneLength = 5)
knnPredict <- predict(knn_model,newdata = log_reg_test)
result <-
confusionMatrix(factor(knnPredict),factor(log_reg_test$Class))
result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85283    52
##          1    12    96
##
##                Accuracy : 0.9993
##                  95% CI : (0.999, 0.9994)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 5.519e-15
##
##                   Kappa : 0.7496
##
##  Mcnemar's Test P-Value : 1.088e-06
##
##             Sensitivity : 0.9999
##             Specificity : 0.6486
##          Pos Pred Value : 0.9994
##          Neg Pred Value : 0.8889
##              Prevalence : 0.9983
##          Detection Rate : 0.9981
##    Detection Prevalence : 0.9987
##       Balanced Accuracy : 0.8243
##
##        'Positive' Class : 0
##
```

```
plot(knn_model, print.thres = 0.5, type="S")
```




```
final <- append(final, prediction)
```

Good performance will now try EDA features with k=1 as the dataset is very unbalanced anything more than k=1 will make knn worse

```
log_reg_train <- select(train,V4,V11,V12, Class)
log_reg_test <- select(test,V4, V11, V12, Class)
prediction <- knn(log_reg_train, log_reg_test, log_reg_train$Class, k
=1, prob = TRUE)
result <- confusionMatrix(factor(prediction),factor(test$Class))
result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85293     7
##          1     2   141
##
##                Accuracy : 0.9999
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.969
```

```
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 1.0000
##             Specificity : 0.9527
##          Pos Pred Value : 0.9999
##          Neg Pred Value : 0.9860
##              Prevalence : 0.9983
##          Detection Rate : 0.9982
##    Detection Prevalence : 0.9983
##       Balanced Accuracy : 0.9763
##
##        'Positive' Class : 0
##

final <- append(final, prediction)
```

Practically amazing results were obtained this is due to there being a clear classification boundary between the features selected in EDA a was observed in EDA the model is simple and easy to deploy with marginal error on both the side

We now try to build models specializing for their own range of data, we build two different models that work in their space of either more than +1 standard deviation or less than that

```
data1 <-filter(data, Amount >=77.17)
head(data1)

##   Time         V1          V2         V3         V4          V5
V6
## 1    0 -1.3598071 -0.07278117  2.5363467  1.3781552 -0.33832077
0.4623878
## 2    1 -1.3583541 -1.34016307  1.7732093  0.3797796 -0.50319813
1.8004994
## 3    1 -0.9662717 -0.18522601  1.7929933 -0.8632913 -0.01030888
1.2472032
## 4    7 -0.8942861  0.28615720 -0.1131922 -0.2715261  2.66959866
3.7218181
## 5   10  1.2499987 -1.22163681  0.3839302 -1.2348987 -1.48541947 -
0.7532302
## 6   16  0.6948848 -1.36181910  1.0292210  0.8341593 -1.19120879
1.3091088
##           V7         V8         V9         V10        V11
V12
## 1  0.2395986  0.0986979  0.3637870  0.09079417 -0.5515995 -
0.61780086
## 2  0.7914610  0.2476758 -1.5146543  0.20764287  0.6245015
0.06608369
## 3  0.2376089  0.3774359 -1.3870241 -0.05495192 -0.2264873
0.17822823
## 4  0.3701451  0.8510844 -0.3920476 -0.41043043 -0.7051166 -
0.11045226
```

```
## 5 -0.6894050 -0.2274872 -2.0940106  1.32372927  0.2276662 -
0.24268200
## 6 -0.8785859  0.4452901 -0.4461958  0.56852074  1.0191506
1.29832870
##          V13         V14        V15        V16        V17
V18
## 1 -0.9913898 -0.31116935  1.4681770 -0.4704005  0.2079712
0.02579058
## 2  0.7172927 -0.16594592  2.3458649 -2.8900832  1.1099694 -
0.12135931
## 3  0.5077569 -0.28792375 -0.6314181 -1.0596472 -0.6840928
1.96577500
## 4 -0.2862536  0.07435536 -0.3287831 -0.2100773 -0.4997680
0.11876486
## 5  1.2054168 -0.31763053  0.7256750 -0.8156122  0.8739364 -
0.84778860
## 6  0.4204803 -0.37265100 -0.8079795 -2.0445575  0.5156635
0.62584730
##          V19         V20         V21          V22         V23
V24
## 1  0.4039930  0.25141210 -0.01830678  0.277837576 -0.11047391
0.06692807
## 2 -2.2618571  0.52497973  0.24799815  0.771679402  0.90941226 -
0.68928096
## 3 -1.2326220 -0.20803778 -0.10830045  0.005273597 -0.19032052 -
1.17557533
## 4  0.5703282  0.05273567 -0.07342510 -0.268091632 -0.20423267
1.01159180
## 5 -0.6831926 -0.10275594 -0.23180924 -0.483285330  0.08466769
0.39283089
## 6 -1.3004082 -0.13833394 -0.29558293 -0.571955007 -0.05088070 -
0.30421450
##           V25        V26         V27         V28 Amount Class
## 1  0.12853936 -0.1891148  0.13355838 -0.02105305 149.62     0
## 2 -0.32764183 -0.1390966 -0.05535279 -0.05975184 378.66     0
## 3  0.64737603 -0.2219288  0.06272285  0.06145763 123.50     0
## 4  0.37320468 -0.3841573  0.01174736  0.14240433  93.20     0
## 5  0.16113455 -0.3549900  0.02641555  0.04242209 121.50     0
## 6  0.07200101 -0.4222344  0.08655340  0.06349865 231.71     0

split <- sample.split(data1$Class, SplitRatio = 0.7)
high_train <-  subset(data1, split == TRUE)
high_test <- subset(data1, split == FALSE)
log_reg_train <- select(high_train,V4,V11,V12, Class)
log_reg_test <- select(high_test,V4, V11, V12, Class)
prediction <- knn(log_reg_train, log_reg_test, log_reg_train$Class, k
=1, prob = TRUE)
result <- confusionMatrix(prediction,factor(high_test$Class))
result
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction     0     1
##          0 21308     2
##          1     0    50
##                                           
##                Accuracy : 0.9999          
##                  95% CI : (0.9997, 1)     
##     No Information Rate : 0.9976          
##     P-Value [Acc > NIR] : <2e-16          
##                                           
##                   Kappa : 0.9803          
##                                           
##  Mcnemar's Test P-Value : 0.4795          
##                                           
##             Sensitivity : 1.0000          
##             Specificity : 0.9615          
##          Pos Pred Value : 0.9999          
##          Neg Pred Value : 1.0000          
##              Prevalence : 0.9976          
##          Detection Rate : 0.9976          
##    Detection Prevalence : 0.9977          
##       Balanced Accuracy : 0.9808          
##                                           
##        'Positive' Class : 0               
## 
```

```
final <- append(final, prediction)
```

thisgreatly improves prediction now we are absolutely sure that there is something fishy about a flagged transanction and not someone who would be stuck without any reason

Clear observation is made that seperatingg the models for the third quarile has helped to reduce the error completely this is due to the fact that some large transanction which were present in feature V11 from EDA were overlapping with medium transanctions from the same feature hence making the model create some problems FP have been eliminated entirely

```
data1 <-filter(data, Amount <=77.17)
head(data1)
```

```
##   Time         V1        V2         V3         V4          V5
V6
## 1    0  1.1918571 0.2661507 0.16648011  0.4481541  0.06001765 -
0.08236081
## 2    2 -1.1582331 0.8777368 1.54871785  0.4030339 -0.40719338
0.09592146
## 3    2 -0.4259659 0.9605230 1.14110934 -0.1682521  0.42098688 -
0.02972755
## 4    4  1.2296576 0.1410035 0.04537077  1.2026127  0.19188099
```

```
0.27270812
## 5    7 -0.6442694 1.4179635 1.07438038 -0.4921990  0.94893409
0.42811846
## 6    9 -0.3382618 1.1195934 1.04436655 -0.2221873  0.49936081 -
0.24676110
##             V7          V8         V9         V10        V11
V12
## 1 -0.078802983  0.08510165 -0.2554251 -0.16697441  1.6127267
1.0652353
## 2  0.592940745 -0.27053268  0.8177393  0.75307443 -0.8228429
0.5381956
## 3  0.476200949  0.26031433 -0.5686714 -0.37140720  1.3412620
0.3598938
## 4 -0.005159003  0.08121294  0.4649600 -0.09925432 -1.4169072 -
0.1538258
## 5  1.120631358 -3.80786424  0.6153747  1.24937618 -0.6194678
0.2914744
## 6  0.651583206  0.06953859 -0.7367273 -0.36684564  1.0176145
0.8363896
##          V13        V14        V15        V16          V17
V18
## 1  0.4890950 -0.1437723 0.63555809  0.4639170 -0.114804663 -
0.18336127
## 2  1.3458516 -1.1196698 0.17512113 -0.4514492 -0.237033239 -
0.03819479
## 3 -0.3580907 -0.1371337 0.51761681  0.4017259 -0.058132823
0.06865315
## 4 -0.7510627  0.1673720 0.05014359 -0.4435868  0.002820512 -
0.61198734
## 5  1.7579642 -1.3238652 0.68613250 -0.0761270 -1.222127345 -
0.35822157
## 6  1.0068435 -0.4435228 0.15021910  0.7394528 -0.540979922
0.47667726
##           V19         V20          V21        V22         V23
V24
## 1 -0.14578304 -0.06908314 -0.225775248 -0.6386720  0.10128802 -
0.3398465
## 2  0.80348692  0.40854236 -0.009430697  0.7982785 -0.13745808
0.1412670
## 3 -0.03319379  0.08496767 -0.208253515 -0.5598248 -0.02639767 -
0.3714266
## 4 -0.04557504 -0.21963255 -0.167716266 -0.2707097 -0.15410379 -
0.7800554
## 5  0.32450473 -0.15674185  1.943465340 -1.0154547  0.05750353 -
0.6497090
## 6  0.45177296  0.20371145 -0.246913937 -0.6337526 -0.12079408 -
0.3850499
##           V25         V26          V27          V28 Amount Class
## 1  0.16717040  0.12589453 -0.008983099  0.014724169   2.69     0
## 2 -0.20600959  0.50229222  0.219422230  0.215153147  69.99     0
```

```
## 3 -0.23279382  0.10591478  0.253844225  0.081080257   3.67     0
## 4  0.75013694 -0.25723685  0.034507430  0.005167769   4.99     0
## 5 -0.41526657 -0.05163430 -1.206921081 -1.085339188  40.80     0
## 6 -0.06973305  0.09419883  0.246219305  0.083075649   3.68     0
```

```
split <- sample.split(data1$Class, SplitRatio = 0.7)
high_train <-  subset(data1, split == TRUE)
high_test <- subset(data1, split == FALSE)
log_reg_train <- select(high_train,V4,V11,V12, Class)
log_reg_test <- select(high_test,V4, V11, V12, Class)
prediction <- knn(log_reg_train, log_reg_test, log_reg_train$Class, k
=1, prob = TRUE)
result <- confusionMatrix(prediction,factor(high_test$Class))
result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 63985     4
##          1     2    91
##
##                Accuracy : 0.9999
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.9985
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.968
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 1.0000
##             Specificity : 0.9579
##          Pos Pred Value : 0.9999
##          Neg Pred Value : 0.9785
##              Prevalence : 0.9985
##          Detection Rate : 0.9985
##    Detection Prevalence : 0.9985
##       Balanced Accuracy : 0.9789
##
##        'Positive' Class : 0
##
```

```
final <- append(final, prediction)
```

This is the matrix containing the values for all the models

```
final <- matrix(final, nrow = 18)
```

```
## Warning in matrix(final, nrow = 18): data length [1537973] is not a
sub-multiple
## or multiple of the number of rows [18]
```

```
rownames(final) <- c('Logistic Regression down_scale', 'Logistic
Regression up_scale', 'logistic regresion', 'logistic regression
Feature importance', 'Logistic Regression up_scale Feature
Importance', 'logistic Regression up_sample EDA', 'logistic regression
EDA', 'Random Forest', 'Random Forest Simple', 'SVM radial Down
Sample', 'Decision Tree', 'Decision Tree down_sample', 'Decision Tree
Up Sample', 'Decision Tree FI', 'Decision Tree EDA', 'KNN Featurre
Importance', 'KNN EDA', 'KNN EDA quartile based')

final.table<- as.table(final)
```

Further building upon the model now for transactions that are less than 3 quartile of Amount are also classified as legit with no FP we can be certain that any flagged transaction is actually really a fraud.

This model has no FP which means that the model was able to flag all normal transanction as not fraud and mostly all fraud transanction as fraud which is great for our system

Building comparison based on our observations we have:

this matrix contains value for all

```
final <-
matrix(c(0.0782,0.1179,0.7254,0.6801,0.0466,0.0544,0.581,0.0012,0,0.00
1732,0,0.969,0.9794,0.99987,0.99988,0.9993,0.9992,0.99973,0.99978,0.99
90,0.998905,NaN,0.99985,0.9996,0.9999,1.0000,0.04247,0.06458,0.9000,0.
8817,0.02565,0.02974,0.8354,0.002347,0,0.99985,0.9996,0.9860,0.9596,0.
7205,0.9994,0.8455),nrow=14)
colnames(final) <- c('Kappa', 'Positive Prediction Value', 'Negative
Prediction Value')
rownames(final) <- c('Logistic Regression down_scale', 'Logistic
Regression up_scale', 'logistic regresion', 'logistic regression
Feature importance', 'Logistic Regression up_scale Feature
Importance', 'logistic Regression up_sample EDA', 'logistic regression
EDA', 'Random Forest', 'Random Forest Simple', 'SVM radial Down
Sample', 'KNN Featurre Importance', 'KNN EDA', 'KNN EDA quartile
based', 'decision tree EDA')
final.table <- as.table(final)
final
```

```
##                                                     Kappa
## Logistic Regression down_scale                   0.078200
## Logistic Regression up_scale                     0.117900
## logistic regresion                               0.725400
## logistic regression Feature importance           0.680100
## Logistic Regression up_scale Feature Importance 0.046600
## logistic Regression up_sample EDA                0.054400
```

```
## logistic regression EDA                                  0.581000
## Random Forest                                            0.001200
## Random Forest Simple                                     0.000000
## SVM radial Down Sample                                   0.001732
## KNN Featurre Importance                                  0.000000
## KNN EDA                                                  0.969000
## KNN EDA quartile based                                   0.979400
## decision tree EDA                                        0.999870
##                                                          Positive Prediction
Value
## Logistic Regression down_scale
0.999880
## Logistic Regression up_scale
0.999300
## logistic regresion
0.999200
## logistic regression Feature importance
0.999730
## Logistic Regression up_scale Feature Importance
0.999780
## logistic Regression up_sample EDA
0.999000
## logistic regression EDA
0.998905
## Random Forest
NaN
## Random Forest Simple
0.999850
## SVM radial Down Sample
0.999600
## KNN Featurre Importance
0.999900
## KNN EDA
1.000000
## KNN EDA quartile based
0.042470
## decision tree EDA
0.064580
##                                                          Negative Prediction
Value
## Logistic Regression down_scale
0.900000
## Logistic Regression up_scale
0.881700
## logistic regresion
0.025650
## logistic regression Feature importance
0.029740
## Logistic Regression up_scale Feature Importance
0.835400
```

```
## logistic Regression up_sample EDA
0.002347
## logistic regression EDA
0.000000
## Random Forest
0.999850
## Random Forest Simple
0.999600
## SVM radial Down Sample
0.986000
## KNN Featurre Importance
0.959600
## KNN EDA
0.720500
## KNN EDA quartile based
0.999400
## decision tree EDA
0.845500
```

building a KNN model for prediction of fraud has been completeted where the following points from the start have been accomplished: Minimize the Fn and FP and if possible remove FP whatsoever Build a model that can be deployed for small ATM machines with very small amount of memory

along with that another important point has been acheived which by using knn allows us to very importantly use a fact that model can be easily updated real time as all transanctions that keep occuring can be added up to training set in Real Time without delay which allows us to modify ourselves to newer ways of coommiting fraud easily rather that again and again building models and updating them.