Implementation : Credit Card Fraud Detection

```
library(data.table)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(ggplot2)
library(DMwR)
```

```
## Warning: package 'DMwR' was built under R version 3.3.3
```

```
## Loading required package: grid
```

```
library(ROSE)
```

```
## Warning: package 'ROSE' was built under R version 3.3.3
```

```
## Loaded ROSE 0.0-3
```

```
library(e1071)
library(class)
library(gmodels)
library(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
```

```
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

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

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

```
## The following object is masked from 'package:graphics':
##
##     layout
```

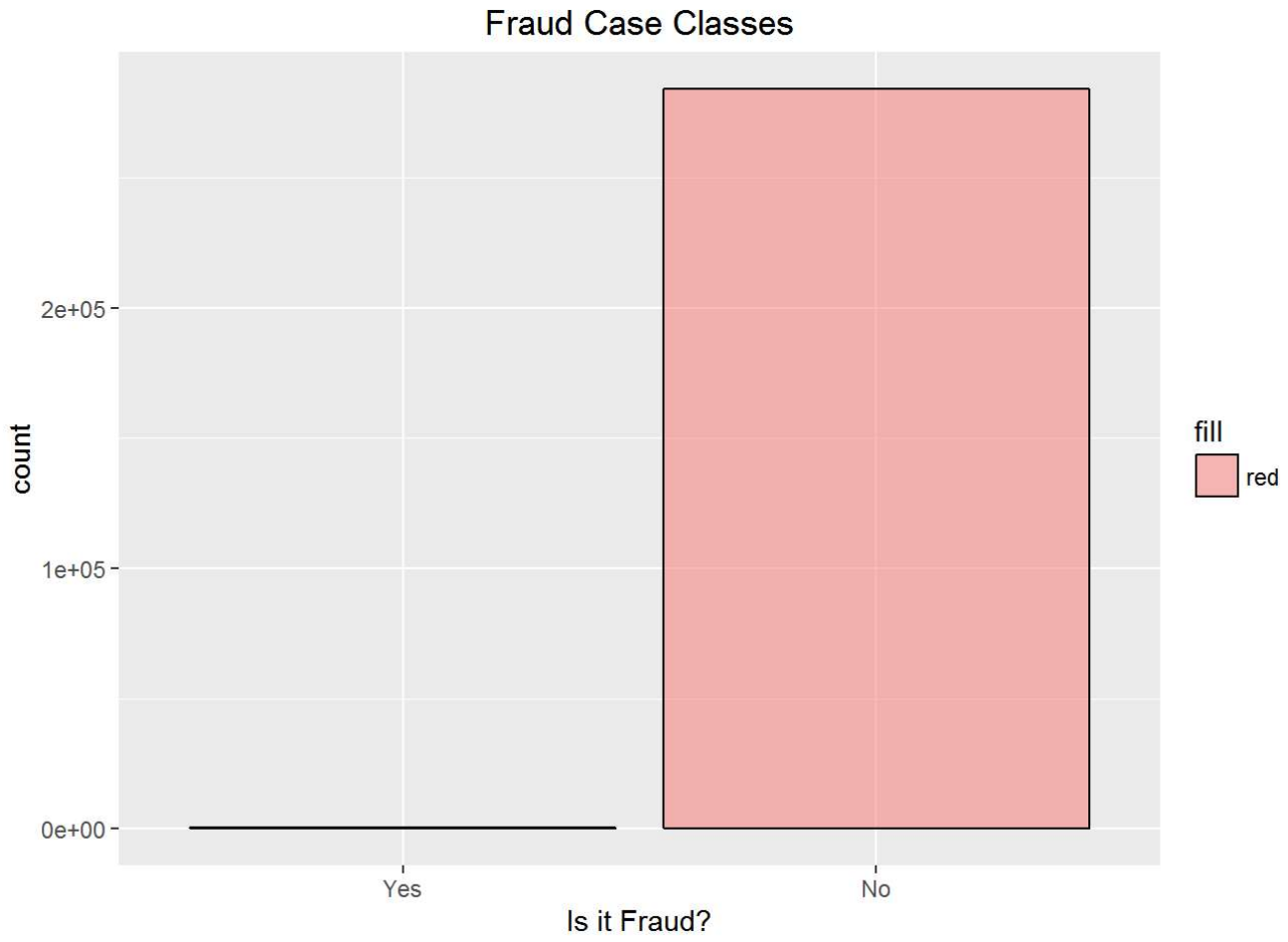
```
library(stats)
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.3.2
```

```
library(C50)
library(gmodels)
library(cluster)
library(fpc)
###  Importing the dataset.
creditcard <- read.csv("creditcard.csv")
### Get the feel of the data
creditcard$Class <- factor(creditcard$Class, levels = c("1", "0"), labels = c("Yes", "No"))
creditcard <- creditcard[sample(nrow(creditcard)),]
table(creditcard$Class)
```

```
##
##    Yes     No
##    492 284315
```

```
#Plot the data
ggplot(creditcard,aes(x = creditcard$Class, fill="red")) +
  geom_bar(position = "dodge", alpha = 0.5, col ="black") +
  scale_x_discrete( name = "Is it Fraud?") +
  scale_y_continuous() +
  ggtitle("Fraud Case Classes") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
### Dividing data into fraud and non fraud classes
fraudData = creditcard[creditcard$Class=="Yes",]
NofraudData = creditcard[creditcard$Class=="No",]

#############################################################################
### New dataset with 25% fraud data
fraud25 = fraudData[1:123,]
dataWith25Fraud = rbind(NofraudData,fraud25)

#Randomize the data for 10% fraud
dataFor25 <- dataWith25Fraud[sample(nrow(dataWith25Fraud)),]
table(dataFor25$Class)
```

```
##
##    Yes     No
##    123 284315
```

```
###removing unbalanced classification problemand making data balanced
set.seed(4356)
smote_data <- SMOTE(Class ~ ., data  = dataFor25, perc.over = 300, perc.under = 150, k=5)
new.data <- sample(2, nrow(smote_data), replace = TRUE, prob = c(0.7, 0.3))
trainSplit2 <-smote_data[new.data==1,]
testSplit2 <-smote_data[new.data==2,]
table(trainSplit2$Class)
```

```
##
## Yes  No
## 346 386
```

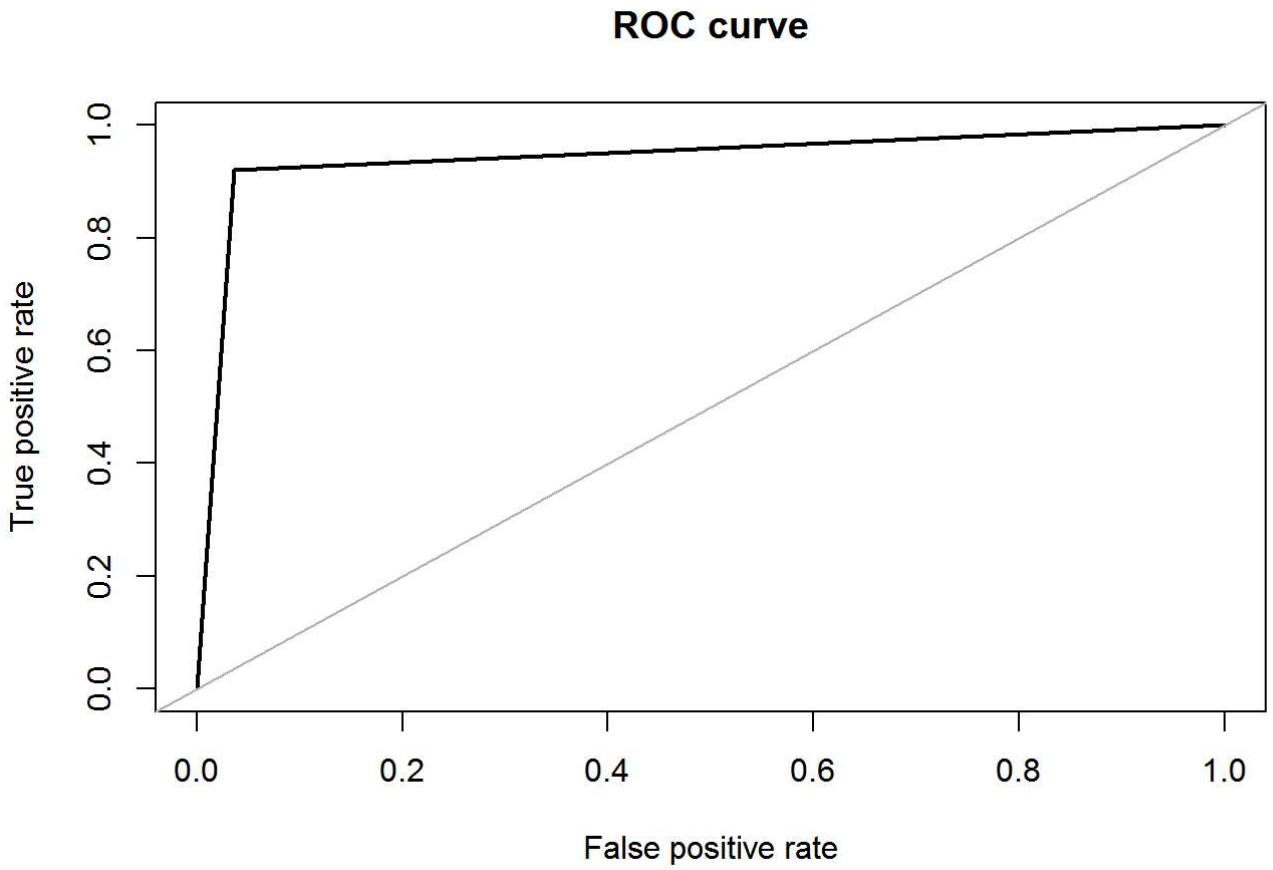
```
table(testSplit2$Class)
```

```
##
## Yes  No
## 146 167
```

```
### Applying SVM algorithm for 25% fraud data
svm.model <- svm(Class ~ ., data = trainSplit2, kernel = "radial", cost = 1, gamma = 0.1)
svm.predict <- predict(svm.model, testSplit2)
confusionMatrix(testSplit2$Class, svm.predict)
```

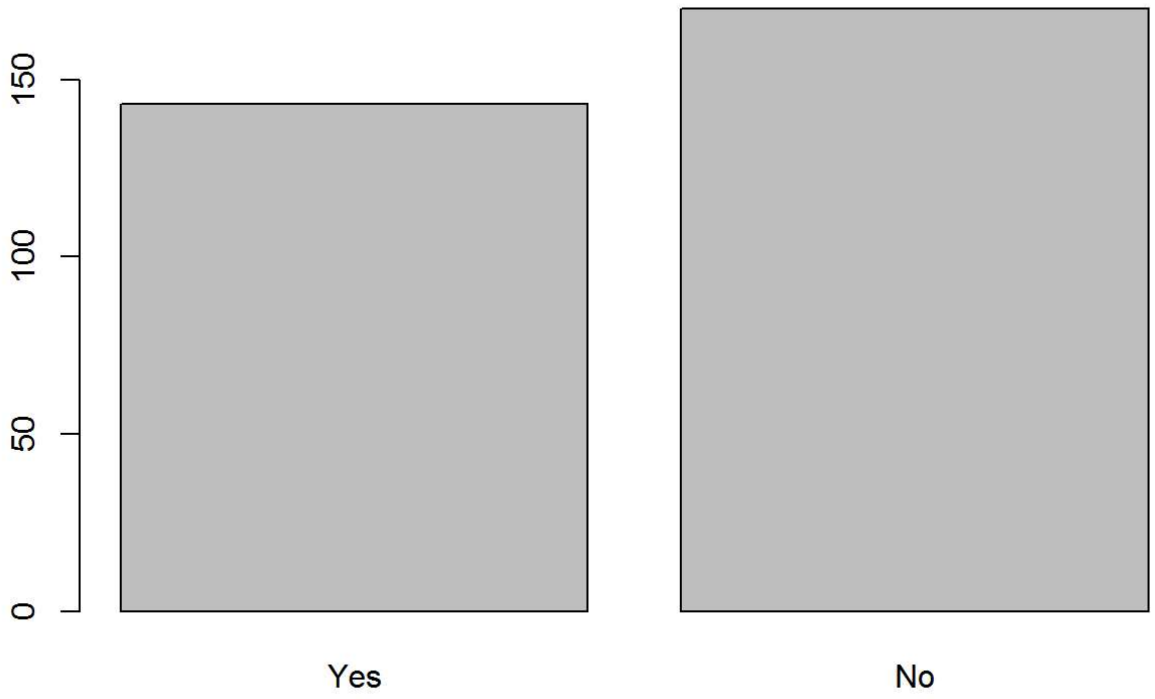
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 132  14
##        No    5 162
##
##                Accuracy : 0.9393
##                  95% CI : (0.9068, 0.9631)
##     No Information Rate : 0.5623
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8776
##  Mcnemar's Test P-Value : 0.06646
##
##             Sensitivity : 0.9635
##             Specificity : 0.9205
##          Pos Pred Value : 0.9041
##          Neg Pred Value : 0.9701
##              Prevalence : 0.4377
##          Detection Rate : 0.4217
##    Detection Prevalence : 0.4665
##       Balanced Accuracy : 0.9420
##
##        'Positive' Class : Yes
##
```

```
roc.curve(svm.predict, testSplit2$Class, plotit = T)
```




```
## Area under the curve (AUC): 0.942
```

```
## Applying KNN Algorithm
knn.model <- knn(train = trainSplit2[,1:30],
    test = testSplit2[,1:30],
    cl = trainSplit2$Class)
plot(knn.model)
```



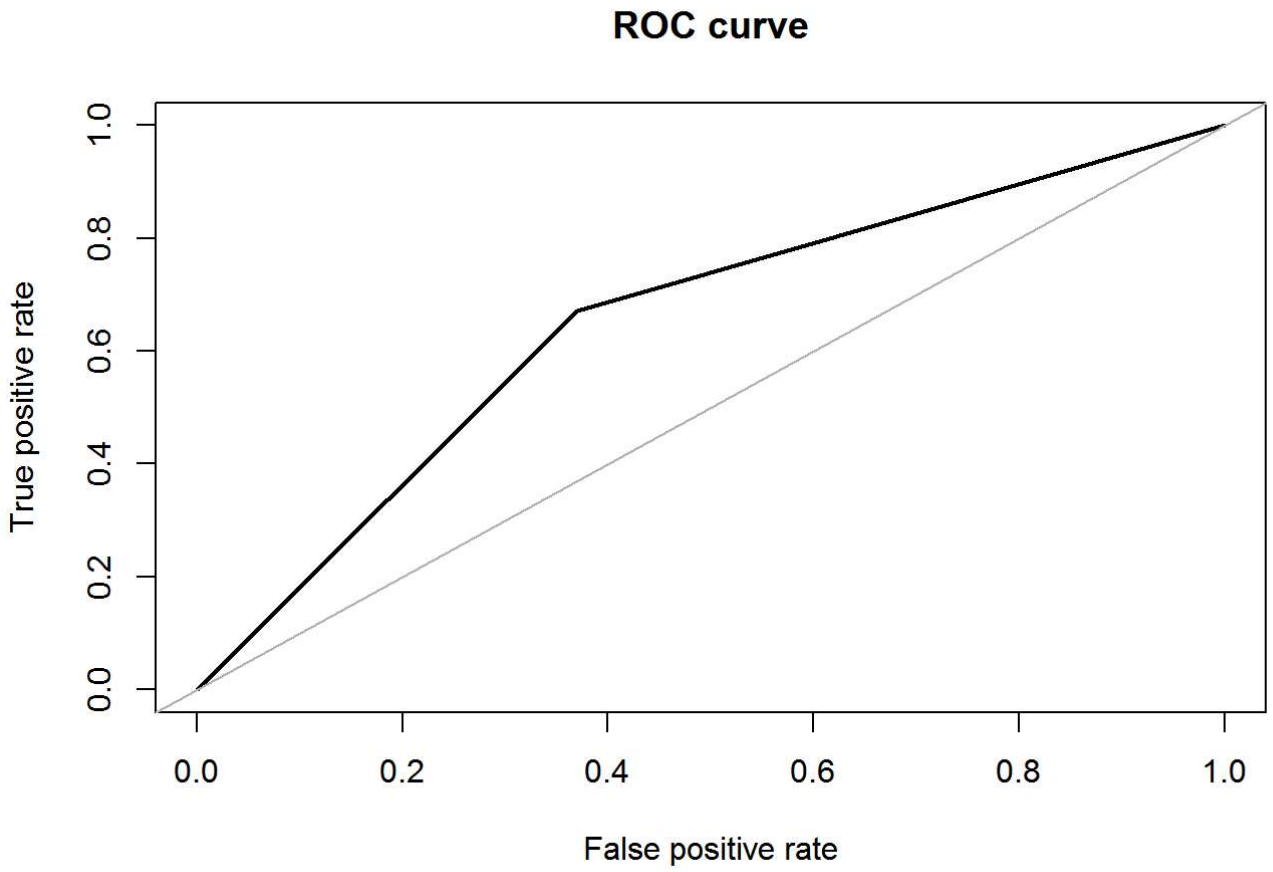

```
table(knn.model, testSplit2$Class)
```

```
##
## knn.model Yes  No
##       Yes  90  53
##       No   56 114
```

```
confusionMatrix(knn.model, testSplit2[,31])
```

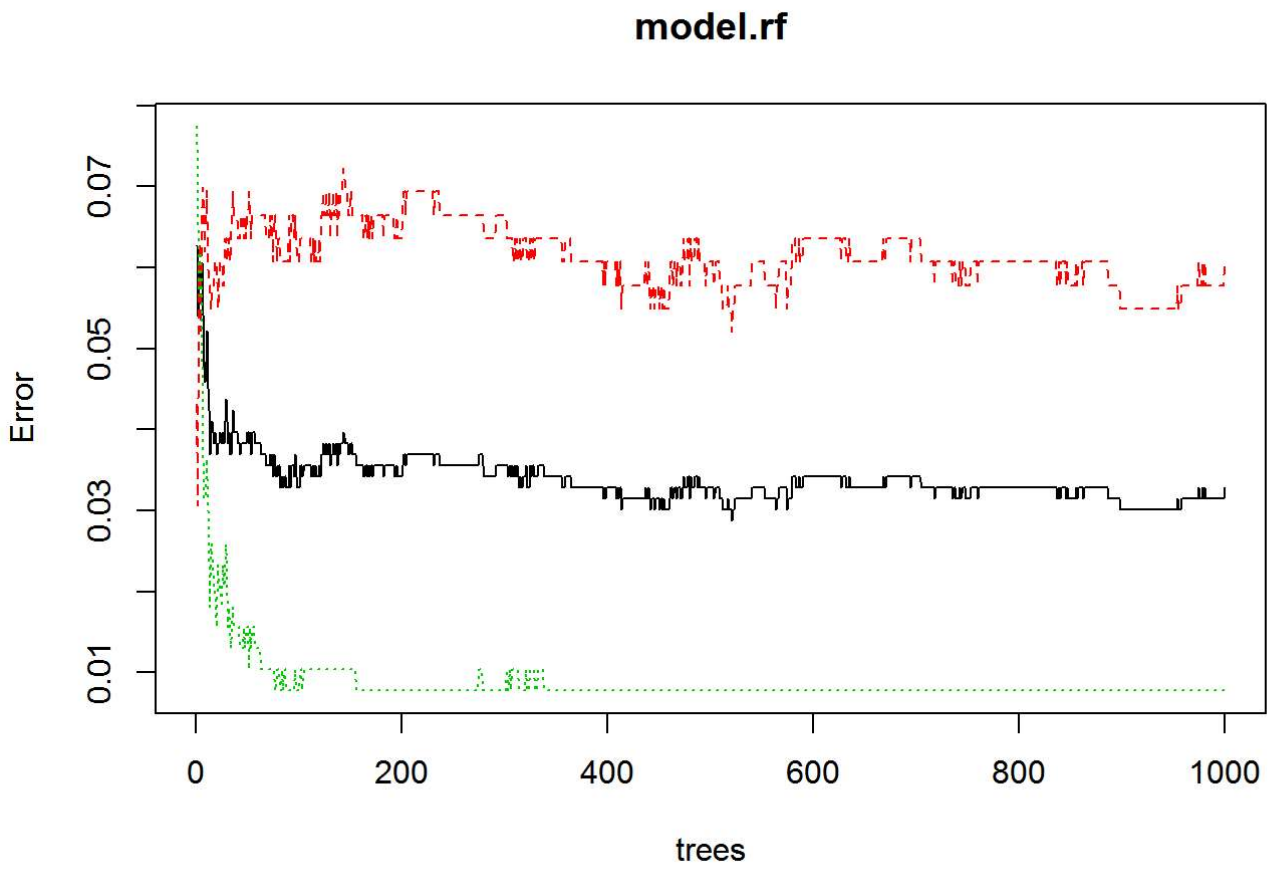
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes  90  53
##        No   56 114
##
##                Accuracy : 0.6518
##                  95% CI : (0.5961, 0.7045)
##     No Information Rate : 0.5335
##     P-Value [Acc > NIR] : 1.49e-05
##
##                   Kappa : 0.2995
##  Mcnemar's Test P-Value : 0.8481
##
##             Sensitivity : 0.6164
##             Specificity : 0.6826
##          Pos Pred Value : 0.6294
##          Neg Pred Value : 0.6706
##              Prevalence : 0.4665
##          Detection Rate : 0.2875
##    Detection Prevalence : 0.4569
##       Balanced Accuracy : 0.6495
##
##        'Positive' Class : Yes
##
```

```
roc.curve(knn.model, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.650
```

```
### Applying Random Forest algorithm for 25% fraud data
model.rf <- randomForest(Class ~ ., data =trainSplit2 , ntree = 1000, importance = TRUE)
plot(model.rf)
```



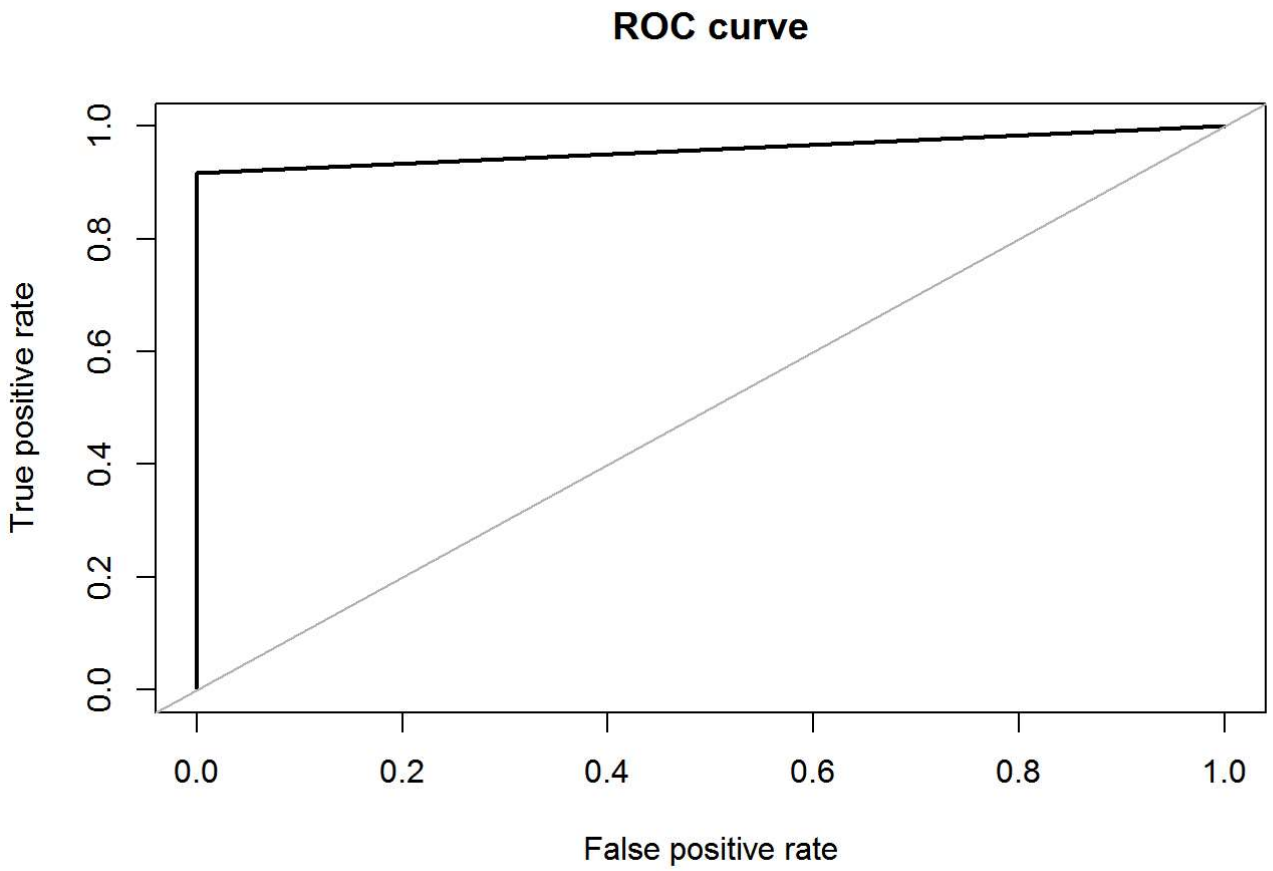

```
cv.tree.pred2 <- predict(model.rf, testSplit2)

# Making table of Confusion matrix
CrossTable(cv.tree.pred2, testSplit2$Class,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual class', 'predicted class'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  313
##
##
##              | predicted class
## actual class |       Yes |        No | Row Total |
## -------------|-----------|-----------|-----------|
##          Yes |       131 |         0 |       131 |
##              |     0.419 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##           No |        15 |       167 |       182 |
##              |     0.048 |     0.534 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       146 |       167 |       313 |
## -------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(cv.tree.pred2, testSplit2$Class)
```

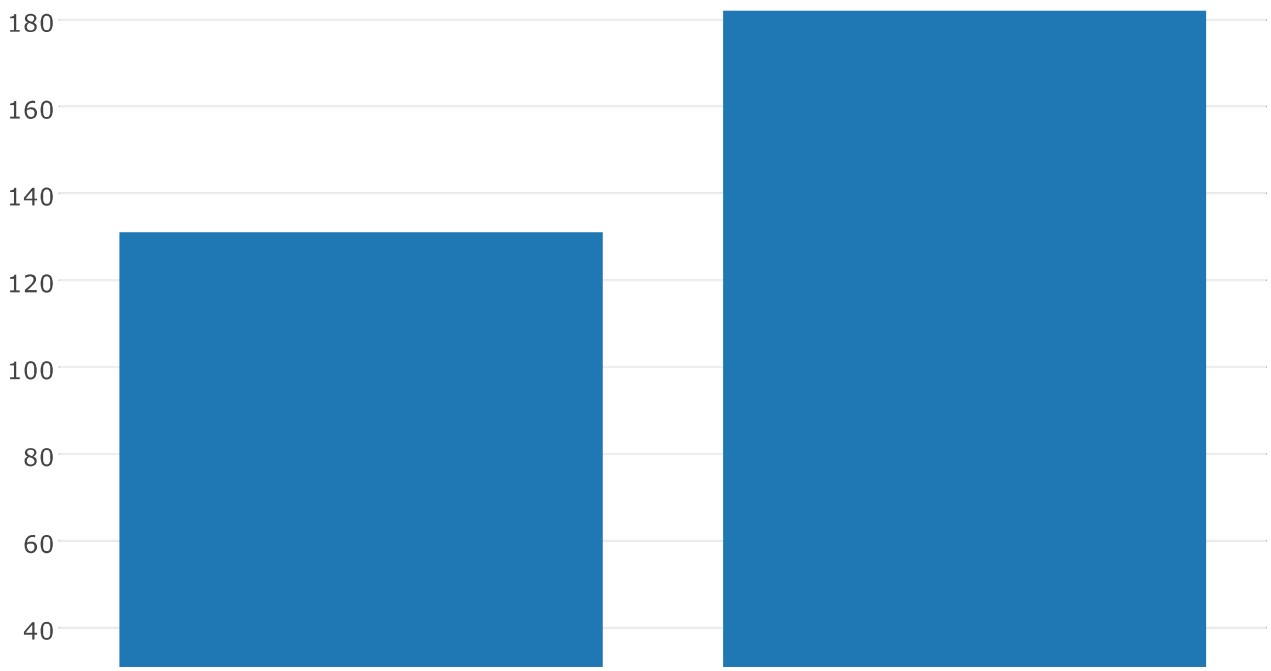
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 131   0
##        No   15 167
##
##                Accuracy : 0.9521
##                  95% CI : (0.9222, 0.9729)
##     No Information Rate : 0.5335
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9031
##  Mcnemar's Test P-Value : 0.0003006
##
##             Sensitivity : 0.8973
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9176
##              Prevalence : 0.4665
##          Detection Rate : 0.4185
##    Detection Prevalence : 0.4185
##       Balanced Accuracy : 0.9486
##
##        'Positive' Class : Yes
##
```

```
### ploting results
roc.curve(cv.tree.pred2, testSplit2$Class, plotit = T)
```




```
## Area under the curve (AUC): 0.959
```

```
x<-rchisq(1000,5,0)
plot_ly(x=cv.tree.pred2,type = 'histogram')
```




```
### Applying Logistic Regression Algorithm for 25% fraud data
logist <- glm(Class ~ ., data = trainSplit2, family = "binomial")
```

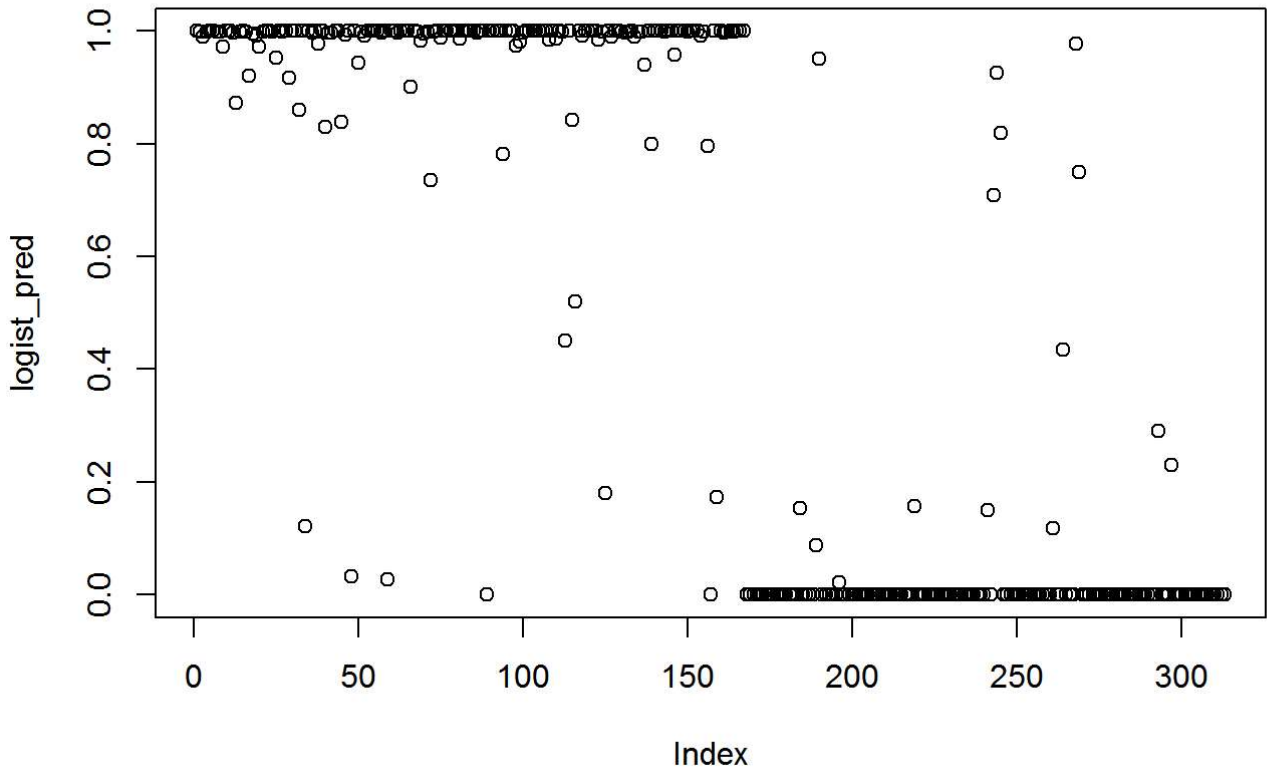
```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(logist)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = trainSplit2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.96293   0.00000   0.00000   0.01889   3.02022
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.368e+01  9.313e+01   0.362    0.718
## Time         1.960e-05  1.599e-05   1.226    0.220
## V1           1.244e+01  3.560e+01   0.349    0.727
## V2          -7.519e+01  1.383e+02  -0.544    0.587
## V3           6.193e+01  7.513e+01   0.824    0.410
## V4          -4.168e+01  5.145e+01  -0.810    0.418
## V5           2.845e+01  3.711e+01   0.766    0.443
## V6           3.646e+01  6.112e+01   0.597    0.551
## V7           1.412e+02  2.299e+02   0.614    0.539
## V8          -2.451e+01  6.237e+01  -0.393    0.694
## V9           5.302e+01  7.594e+01   0.698    0.485
## V10          1.204e+02  1.717e+02   0.701    0.483
## V11         -9.061e+01  1.333e+02  -0.680    0.497
## V12          1.648e+02  2.391e+02   0.689    0.491
## V13          2.064e+00  8.756e+00   0.236    0.814
## V14          1.749e+02  2.575e+02   0.679    0.497
## V15          6.362e+00  9.844e+00   0.646    0.518
## V16          1.562e+02  2.298e+02   0.680    0.497
## V17          2.782e+02  4.072e+02   0.683    0.495
## V18          1.036e+02  1.551e+02   0.668    0.504
## V19         -4.000e+01  6.171e+01  -0.648    0.517
## V20          8.462e+00  5.497e+01   0.154    0.878
## V21         -1.122e+01  6.325e+01  -0.177    0.859
## V22         -1.573e+01  3.733e+01  -0.421    0.673
## V23         -3.286e+01  8.132e+01  -0.404    0.686
## V24          4.328e+00  7.944e+00   0.545    0.586
## V25         -1.928e+01  3.678e+01  -0.524    0.600
## V26         -1.129e+00  9.347e+00  -0.121    0.904
## V27         -2.481e+01  3.112e+01  -0.797    0.425
## V28         -3.478e+01  9.784e+01  -0.355    0.722
## Amount      -3.910e-01  9.375e-01  -0.417    0.677
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1012.581  on 731  degrees of freedom
## Residual deviance:   77.153  on 701  degrees of freedom
## AIC: 139.15
##
## Number of Fisher Scoring iterations: 25
```

```
logist_pred <- predict(logist, newdata=testSplit2,type = "response")
plot(logist_pred)
```

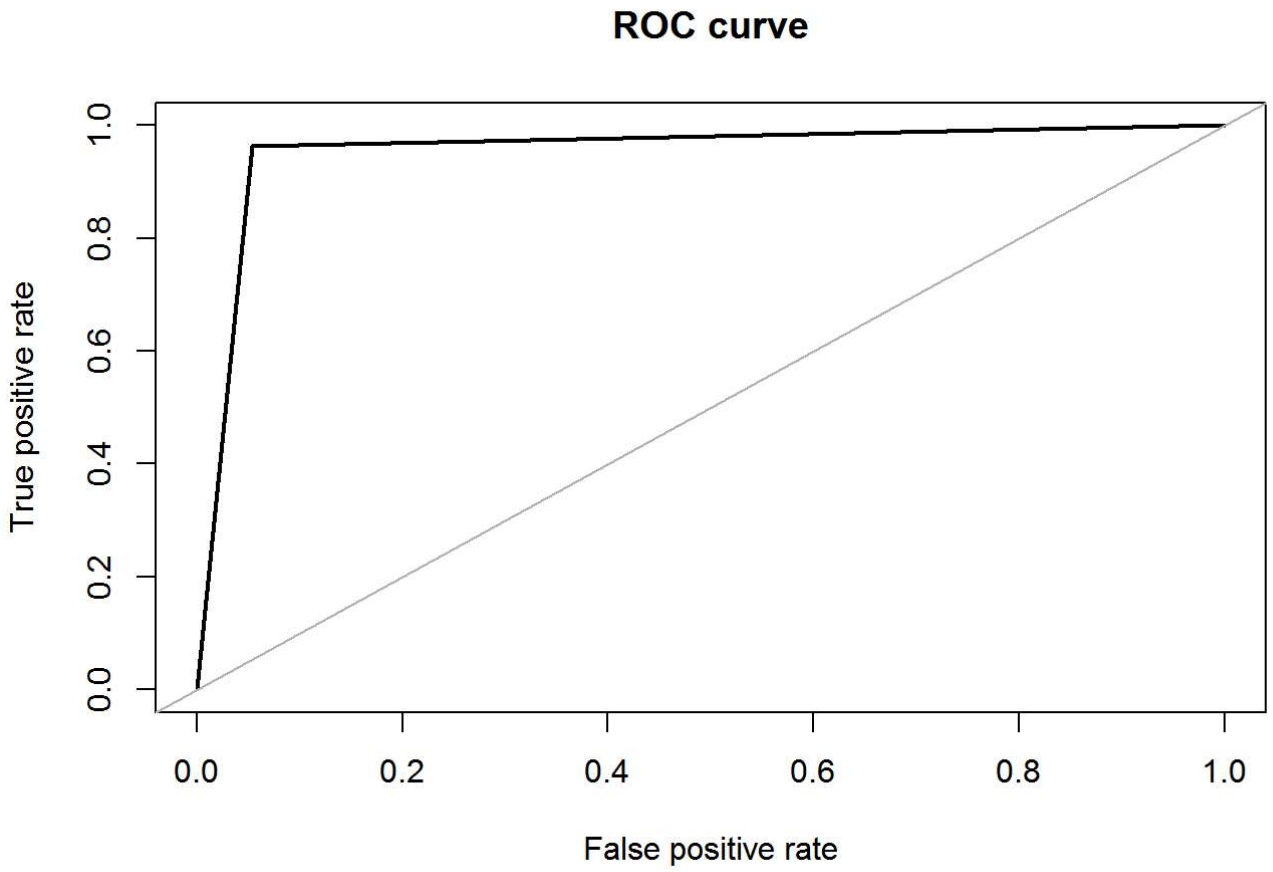



```
pred=rep("Yes",length(logist_pred))
pred[logist_pred > 0.5] = "No"
confusionMatrix(testSplit2$Class, pred)
```

```
## Warning in confusionMatrix.default(testSplit2$Class, pred): Levels are not
## in the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  159   8
##        Yes   6 140
##
##                Accuracy : 0.9553
##                  95% CI : (0.9261, 0.9753)
##     No Information Rate : 0.5272
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9102
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9636
##             Specificity : 0.9459
##          Pos Pred Value : 0.9521
##          Neg Pred Value : 0.9589
##              Prevalence : 0.5272
##          Detection Rate : 0.5080
##    Detection Prevalence : 0.5335
##       Balanced Accuracy : 0.9548
##
##        'Positive' Class : No
##
```

```
roc.curve(pred, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.955
```

```
#############################################################################
### New dataset with 50% fraud data
fraud50 = fraudData[1:246,]
dataWith50Fraud = rbind(NofraudData,fraud50)

#Randomize the data for 50% fraud
dataFor50 <- dataWith50Fraud[sample(nrow(dataWith50Fraud)),]
table(dataFor50$Class)
```

```
##
##    Yes     No
##    246 284315
```

```
###removing unbalanced classification problemand making data balanced
set.seed(4356)
smote_data <- SMOTE(Class ~ ., data  = dataFor50, perc.over = 300, perc.under = 150, k=5)
new.data <- sample(2, nrow(smote_data), replace = TRUE, prob = c(0.7, 0.3))
trainSplit2 <-smote_data[new.data==1,]
testSplit2 <-smote_data[new.data==2,]
table(trainSplit2$Class)
```

```
##
## Yes  No
## 682 772
```

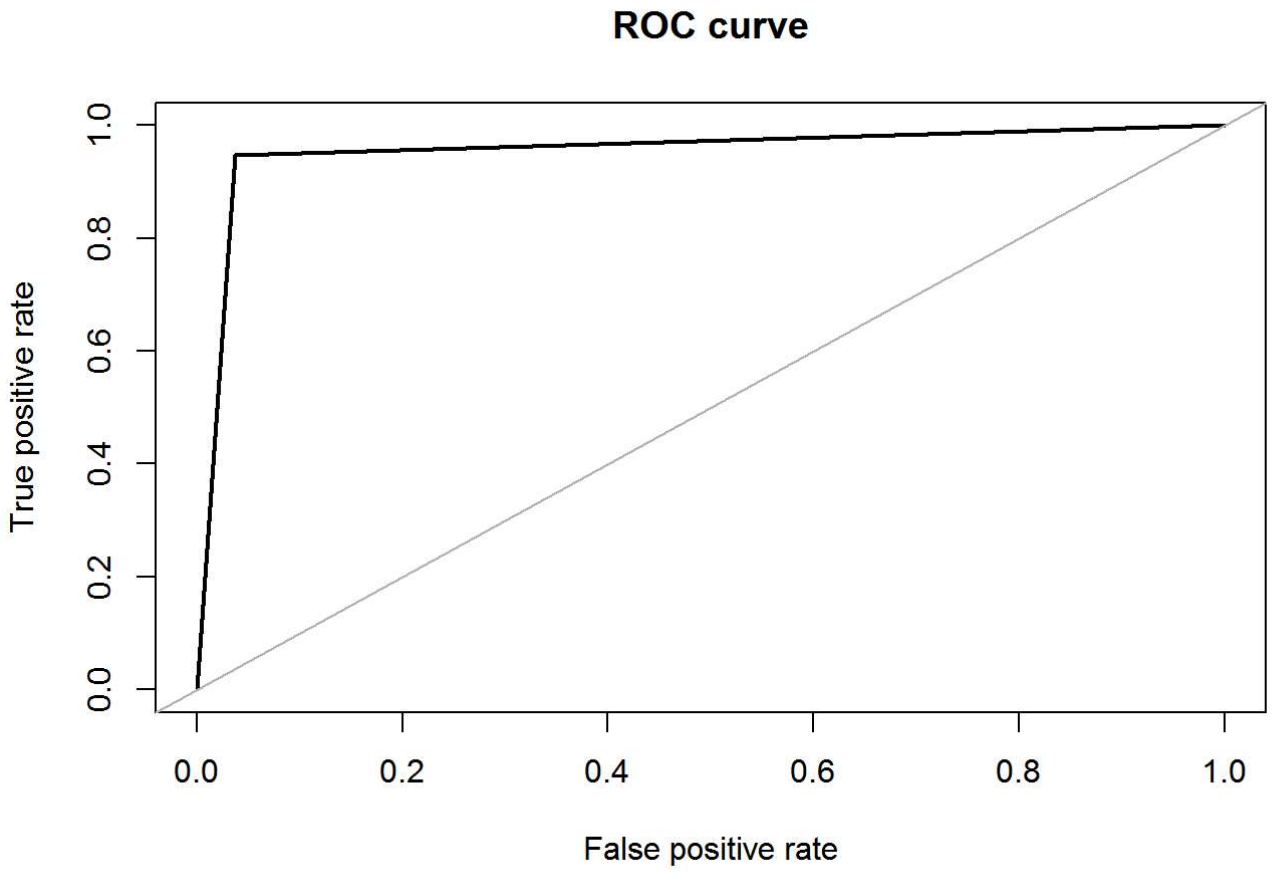
```
table(testSplit2$Class)
```

```
##
## Yes  No
## 302 335
```

```
### Applying SVM algorithm for 50% fraud data
svm.model <- svm(Class ~ ., data = trainSplit2, kernel = "radial", cost = 1, gamma = 0.1)
svm.predict <- predict(svm.model, testSplit2)
confusionMatrix(testSplit2$Class, svm.predict)
```

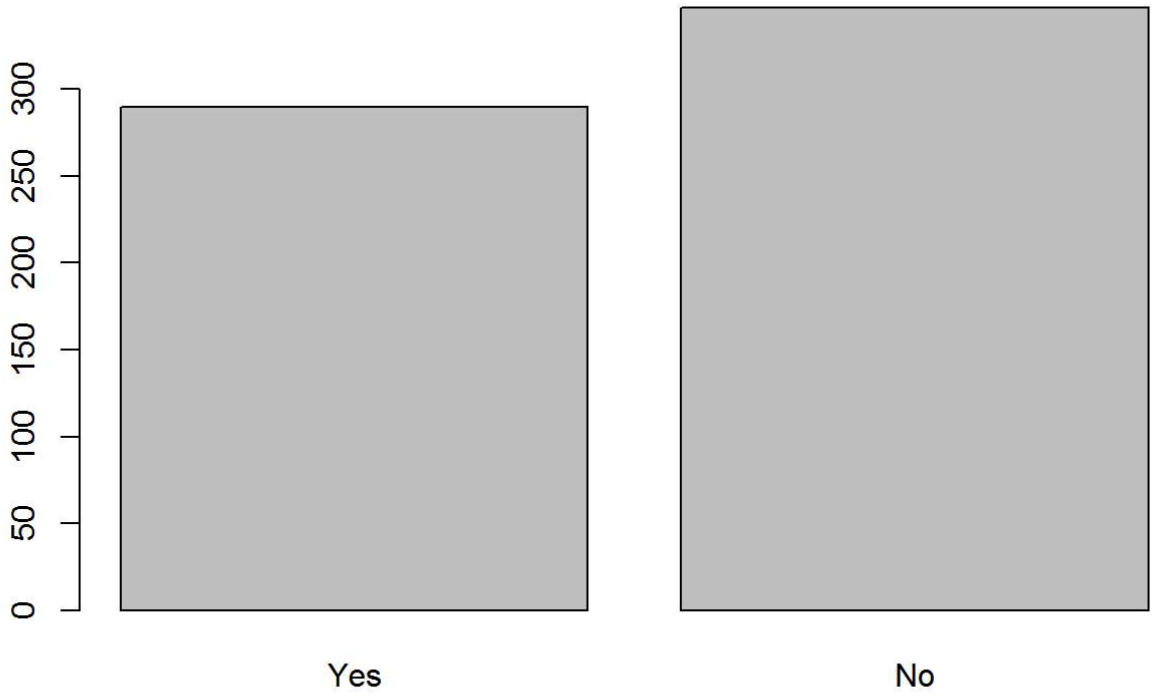
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 284  18
##        No   11 324
##
##                Accuracy : 0.9545
##                  95% CI : (0.9353, 0.9693)
##     No Information Rate : 0.5369
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9086
##  Mcnemar's Test P-Value : 0.2652
##
##             Sensitivity : 0.9627
##             Specificity : 0.9474
##          Pos Pred Value : 0.9404
##          Neg Pred Value : 0.9672
##              Prevalence : 0.4631
##          Detection Rate : 0.4458
##    Detection Prevalence : 0.4741
##       Balanced Accuracy : 0.9550
##
##        'Positive' Class : Yes
##
```

```
roc.curve(svm.predict, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.955
```

```
## Applying KNN Algorithm
knn.model <- knn(train = trainSplit2[,1:30],
    test = testSplit2[,1:30],
    cl = trainSplit2$Class)
plot(knn.model)
```



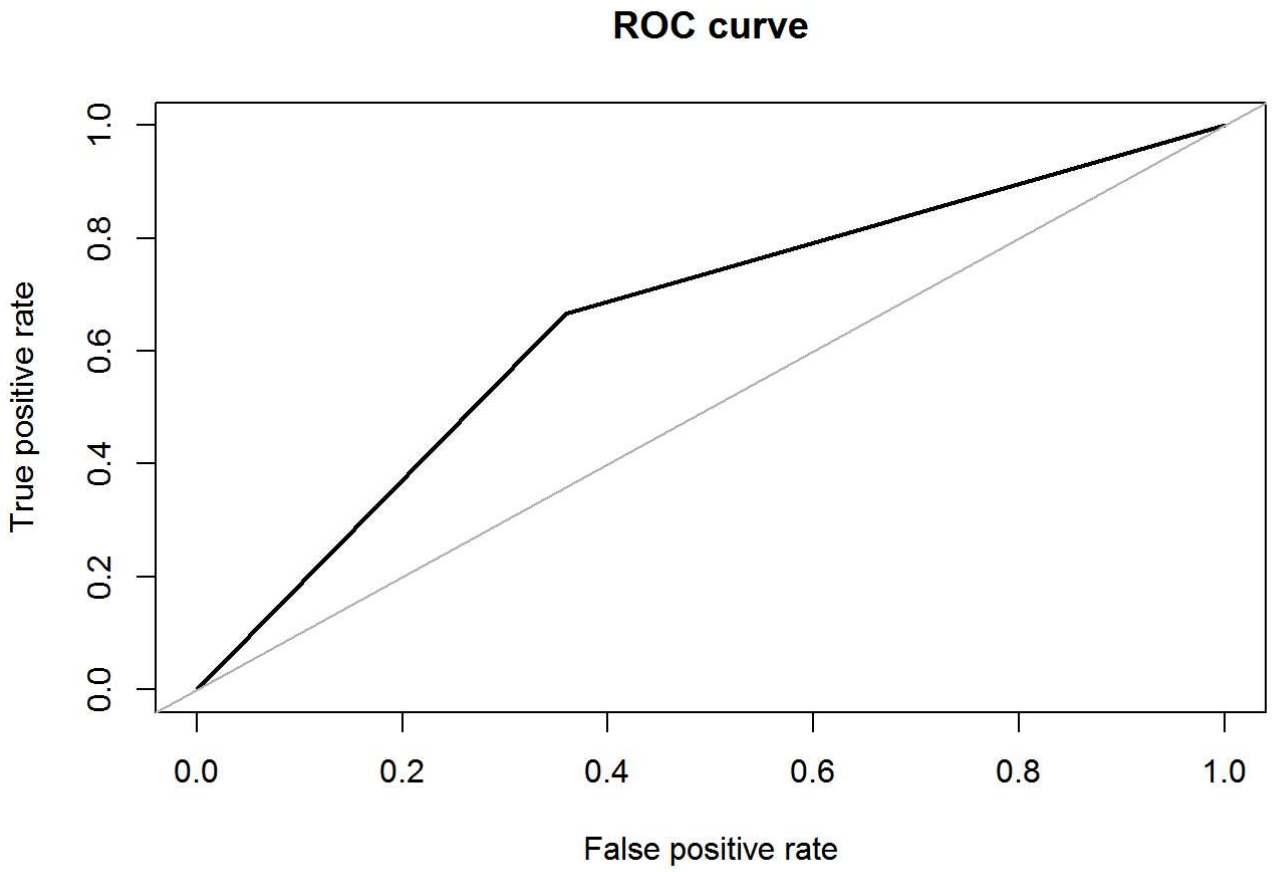

```
table(knn.model, testSplit2$Class)
```

```
##
## knn.model Yes  No
##       Yes 186 104
##       No  116 231
```

```
confusionMatrix(knn.model, testSplit2[,31])
```

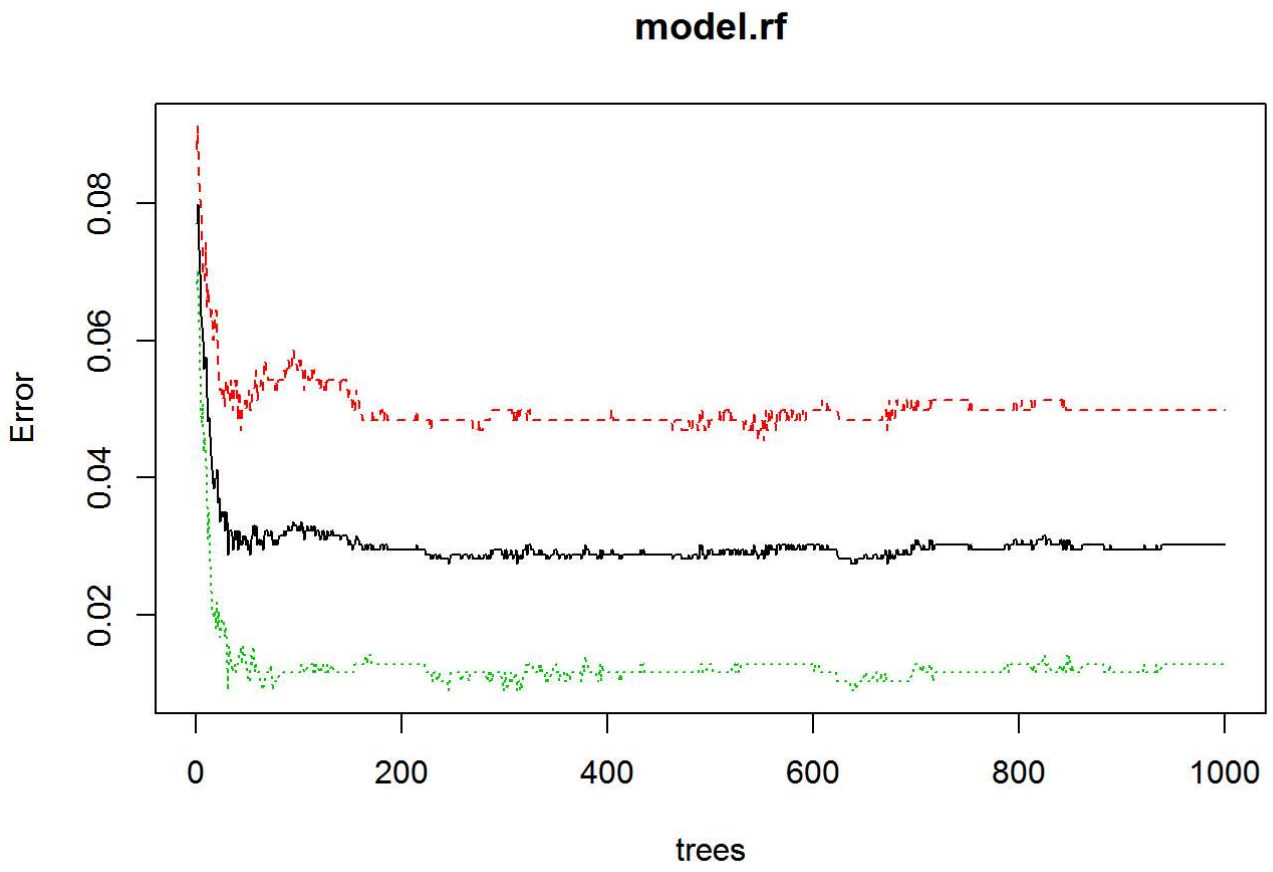
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 186 104
##        No  116 231
##
##                Accuracy : 0.6546
##                  95% CI : (0.6163, 0.6916)
##     No Information Rate : 0.5259
##     P-Value [Acc > NIR] : 3.252e-11
##
##                   Kappa : 0.306
##  Mcnemar's Test P-Value : 0.4583
##
##             Sensitivity : 0.6159
##             Specificity : 0.6896
##          Pos Pred Value : 0.6414
##          Neg Pred Value : 0.6657
##              Prevalence : 0.4741
##          Detection Rate : 0.2920
##    Detection Prevalence : 0.4553
##       Balanced Accuracy : 0.6527
##
##        'Positive' Class : Yes
##
```

```
roc.curve(knn.model, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.654
```

```
### Applying Random Forest algorithm for 50% fraud data
model.rf <- randomForest(Class ~ ., data =trainSplit2 , ntree = 1000, importance = TRUE)
plot(model.rf)
```



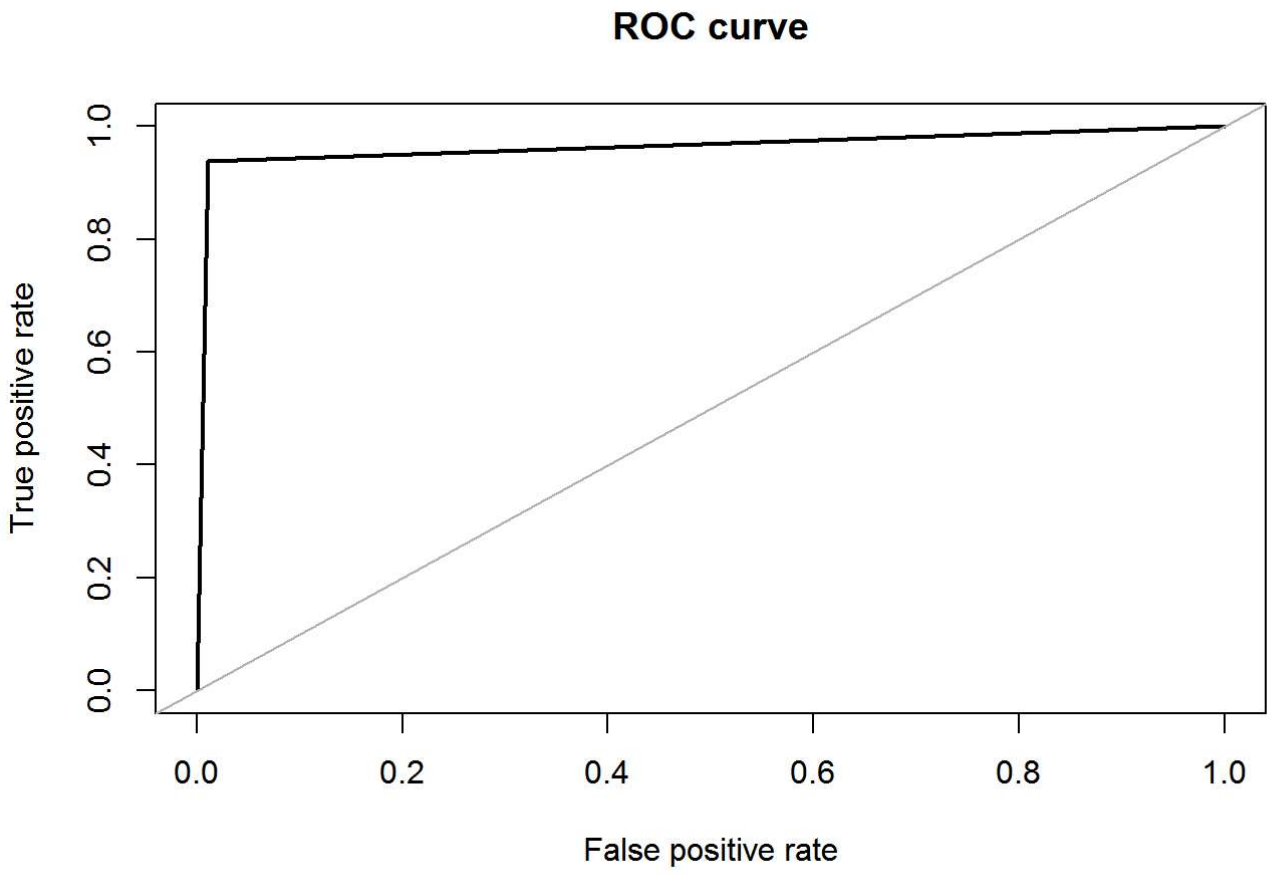

```
cv.tree.pred2 <- predict(model.rf, testSplit2)

# Making table of Confusion matrix
CrossTable(cv.tree.pred2, testSplit2$Class,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual class', 'predicted class'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  637
##
##
##              | predicted class
## actual class |       Yes |        No | Row Total |
## -------------|-----------|-----------|-----------|
##          Yes |       280 |         3 |       283 |
##              |     0.440 |     0.005 |           |
## -------------|-----------|-----------|-----------|
##           No |        22 |       332 |       354 |
##              |     0.035 |     0.521 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       302 |       335 |       637 |
## -------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(cv.tree.pred2, testSplit2$Class)
```

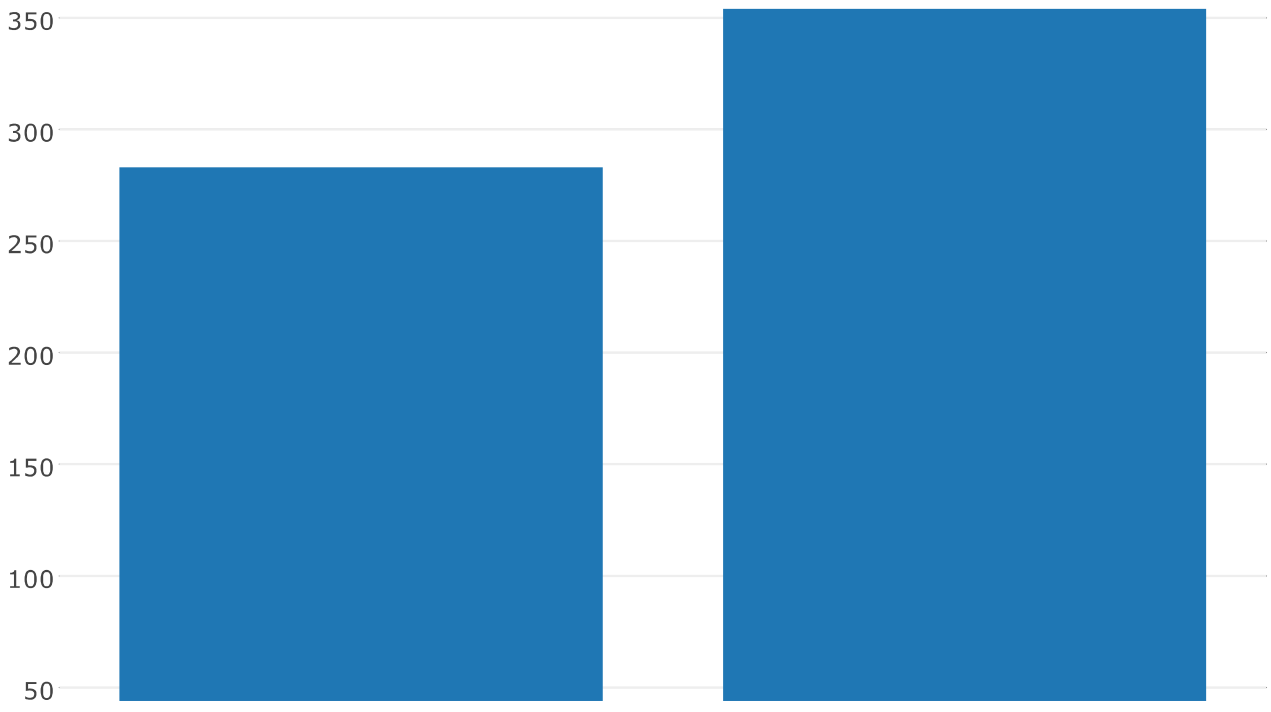
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 280   3
##        No   22 332
##
##                Accuracy : 0.9608
##                  95% CI : (0.9426, 0.9744)
##     No Information Rate : 0.5259
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9211
##  Mcnemar's Test P-Value : 0.0003182
##
##             Sensitivity : 0.9272
##             Specificity : 0.9910
##          Pos Pred Value : 0.9894
##          Neg Pred Value : 0.9379
##              Prevalence : 0.4741
##          Detection Rate : 0.4396
##    Detection Prevalence : 0.4443
##       Balanced Accuracy : 0.9591
##
##        'Positive' Class : Yes
##
```

```
roc.curve(cv.tree.pred2, testSplit2$Class, plotit = T)
```




```
## Area under the curve (AUC): 0.964
```

```
x<-rchisq(1000,5,0)
plot_ly(x=cv.tree.pred2,type = 'histogram')
```



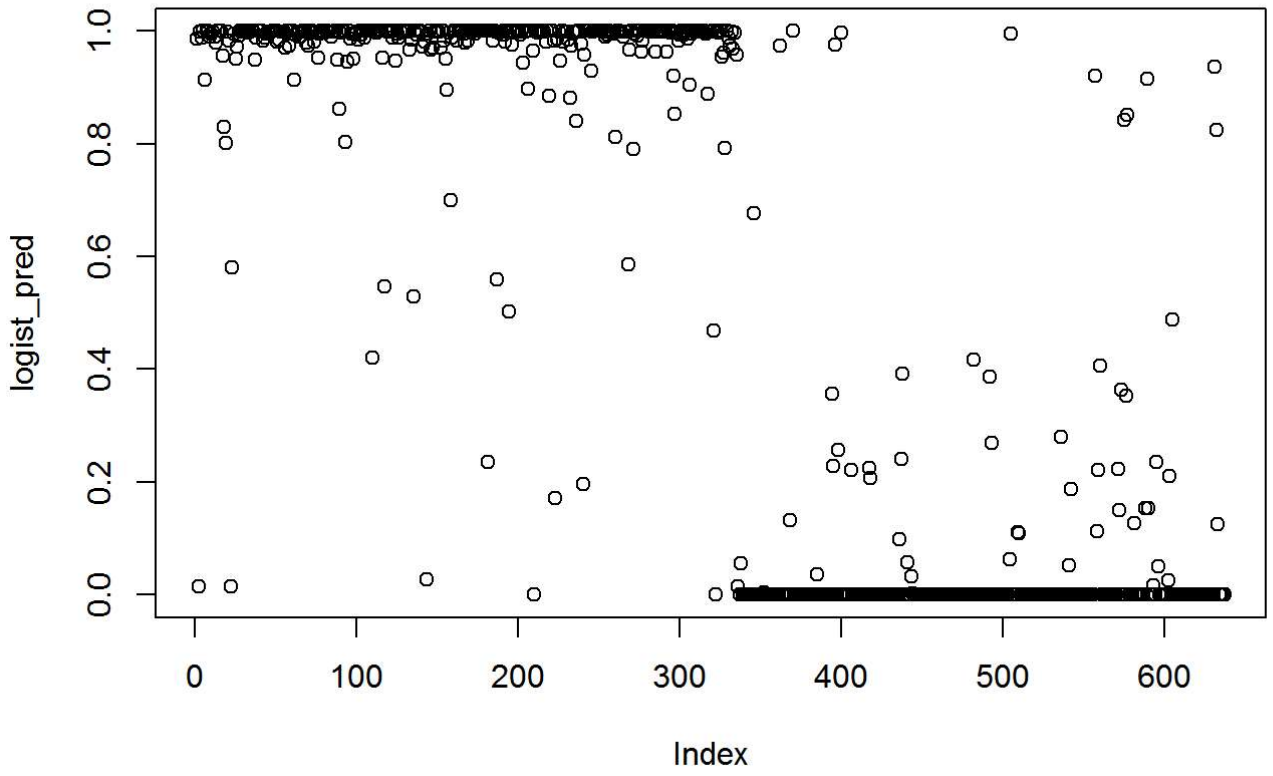

```
### Applying Logistic Regression Algorithm for 50% fraud data
logist <- glm(Class ~ ., data = trainSplit2, family = "binomial")
```

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

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

```
summary(logist)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = trainSplit2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.96702   0.00000   0.00000   0.05639   2.99552
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.935e+01  8.519e+01   0.462  0.64412
## Time         4.322e-05  1.069e-05   4.041 5.32e-05 ***
## V1           1.328e+01  1.934e+01   0.686  0.49245
## V2          -9.543e+01  1.257e+02  -0.759  0.44766
## V3           7.865e+01  5.023e+01   1.566  0.11740
## V4          -5.247e+01  4.030e+01  -1.302  0.19294
## V5           3.194e+01  9.969e+00   3.204  0.00136 **
## V6           4.568e+01  5.717e+01   0.799  0.42427
## V7           1.783e+02  1.972e+02   0.904  0.36601
## V8          -3.569e+01  3.359e+01  -1.062  0.28810
## V9           6.577e+01  6.051e+01   1.087  0.27707
## V10          1.525e+02  1.391e+02   1.096  0.27323
## V11         -1.165e+02  1.178e+02  -0.990  0.32232
## V12          2.107e+02  2.115e+02   0.996  0.31915
## V13          2.256e+00  5.561e+00   0.406  0.68492
## V14          2.253e+02  2.307e+02   0.977  0.32879
## V15          6.782e+00  8.228e+00   0.824  0.40981
## V16          2.004e+02  2.036e+02   0.984  0.32500
## V17          3.570e+02  3.574e+02   0.999  0.31787
## V18          1.358e+02  1.366e+02   0.994  0.32025
## V19         -5.114e+01  5.654e+01  -0.905  0.36567
## V20          1.215e+01  3.828e+01   0.317  0.75093
## V21         -1.792e+01  9.703e+00  -1.847  0.06470 .
## V22         -1.663e+01  2.500e+01  -0.665  0.50579
## V23         -4.205e+01  7.549e+01  -0.557  0.57753
## V24          5.465e+00  7.228e+00   0.756  0.44956
## V25         -2.234e+01  3.437e+01  -0.650  0.51569
## V26         -3.689e+00  8.678e+00  -0.425  0.67075
## V27         -2.954e+01  2.718e+01  -1.087  0.27714
## V28         -5.311e+01  9.139e+01  -0.581  0.56114
## Amount      -5.067e-01  8.720e-01  -0.581  0.56118
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2010.10  on 1453  degrees of freedom
## Residual deviance:  171.25  on 1423  degrees of freedom
## AIC: 233.25
##
## Number of Fisher Scoring iterations: 25
```

```
logist_pred <- predict(logist, newdata=testSplit2,type = "response")
plot(logist_pred)
```

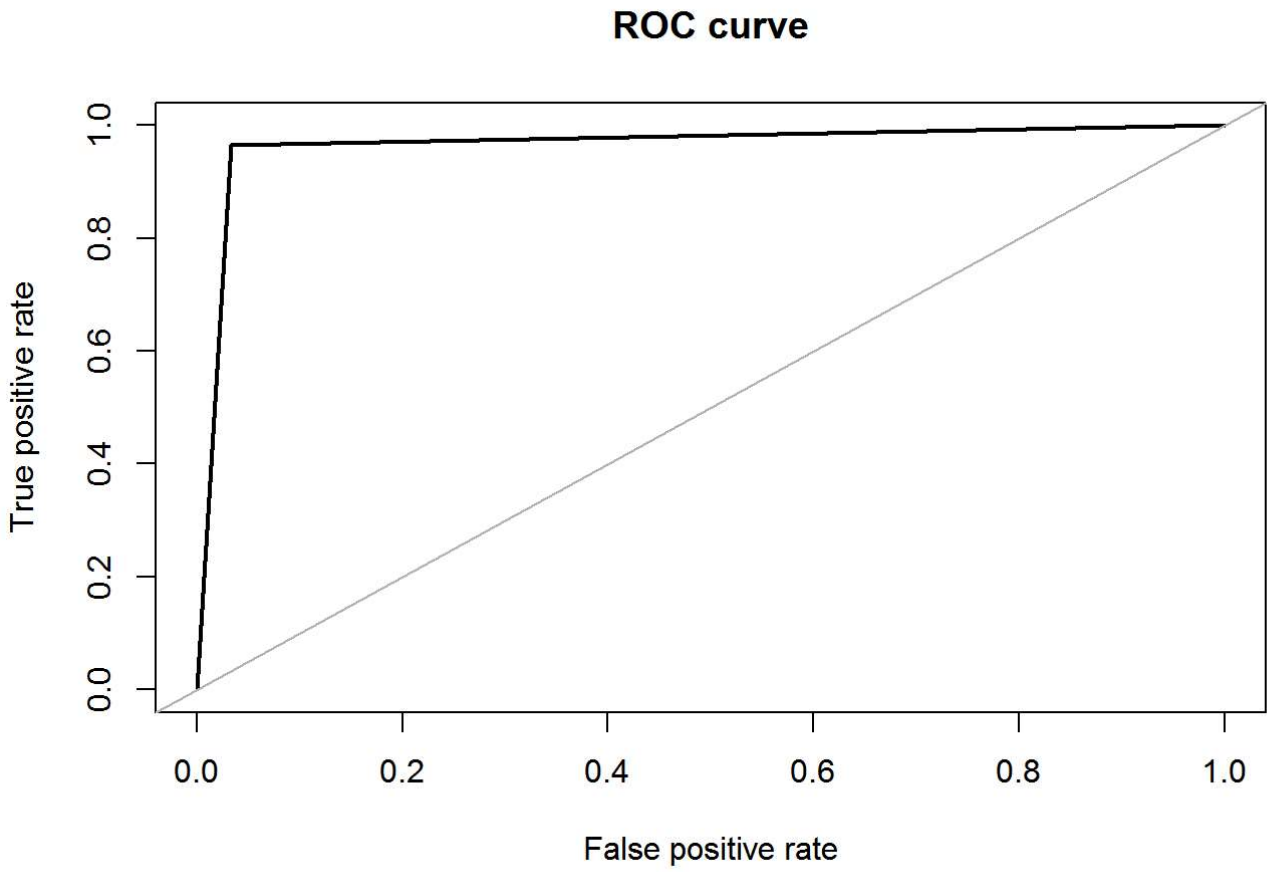



```
pred=rep("Yes",length(logist_pred))
pred[logist_pred > 0.5] = "No"
confusionMatrix(testSplit2$Class, pred)
```

```
## Warning in confusionMatrix.default(testSplit2$Class, pred): Levels are not
## in the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  325  10
##        Yes  12 290
##
##                Accuracy : 0.9655
##                  95% CI : (0.9482, 0.9782)
##     No Information Rate : 0.529
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9307
##  Mcnemar's Test P-Value : 0.8312
##
##             Sensitivity : 0.9644
##             Specificity : 0.9667
##          Pos Pred Value : 0.9701
##          Neg Pred Value : 0.9603
##              Prevalence : 0.5290
##          Detection Rate : 0.5102
##    Detection Prevalence : 0.5259
##       Balanced Accuracy : 0.9655
##
##        'Positive' Class : No
##
```

```
roc.curve(pred, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.966
```

```
#############################################################################
### New dataset with 75% fraud data
fraud75 = fraudData[1:369,]
dataWith75Fraud = rbind(NofraudData,fraud75)

#Randomize the data for 75% fraud
dataFor75 <- dataWith75Fraud[sample(nrow(dataWith75Fraud)),]
table(dataFor75$Class)
```

```
##
##    Yes     No
##    369 284315
```

```
###removing unbalanced classification problemand making data balanced
set.seed(4356)
smote_data <- SMOTE(Class ~ ., data  = dataFor75, perc.over = 300, perc.under = 150, k=5)
new.data <- sample(2, nrow(smote_data), replace = TRUE, prob = c(0.7, 0.3))
trainSplit2 <-smote_data[new.data==1,]
testSplit2 <-smote_data[new.data==2,]
table(trainSplit2$Class)
```

```
##
##  Yes   No
## 1026 1173
```

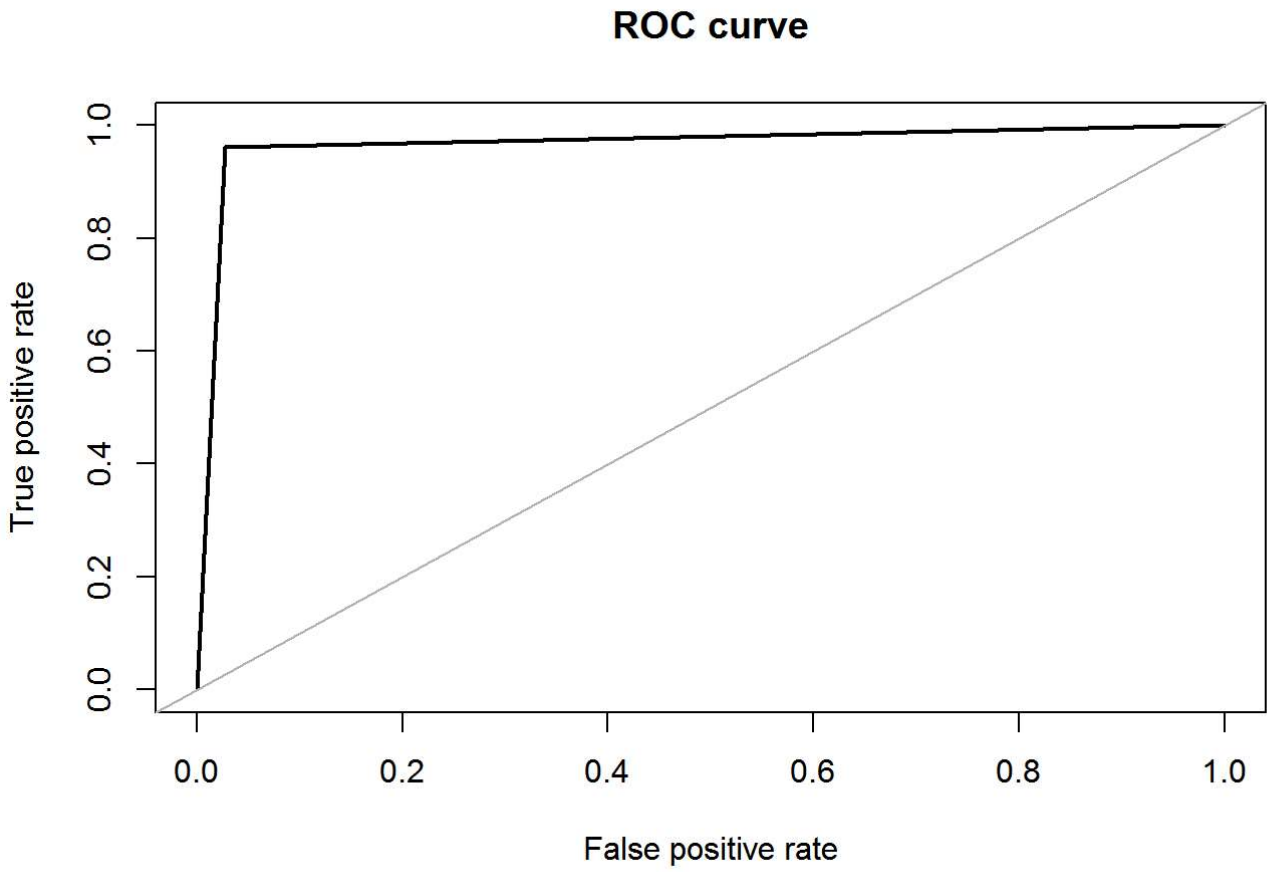
```
table(testSplit2$Class)
```

```
##
## Yes  No
## 450 487
```

```
### Applying SVM algorithm for 75% fraud data
svm.model <- svm(Class ~ ., data = trainSplit2, kernel = "radial", cost = 1, gamma = 0.1)
svm.predict <- predict(svm.model, testSplit2)
confusionMatrix(testSplit2$Class, svm.predict)
```

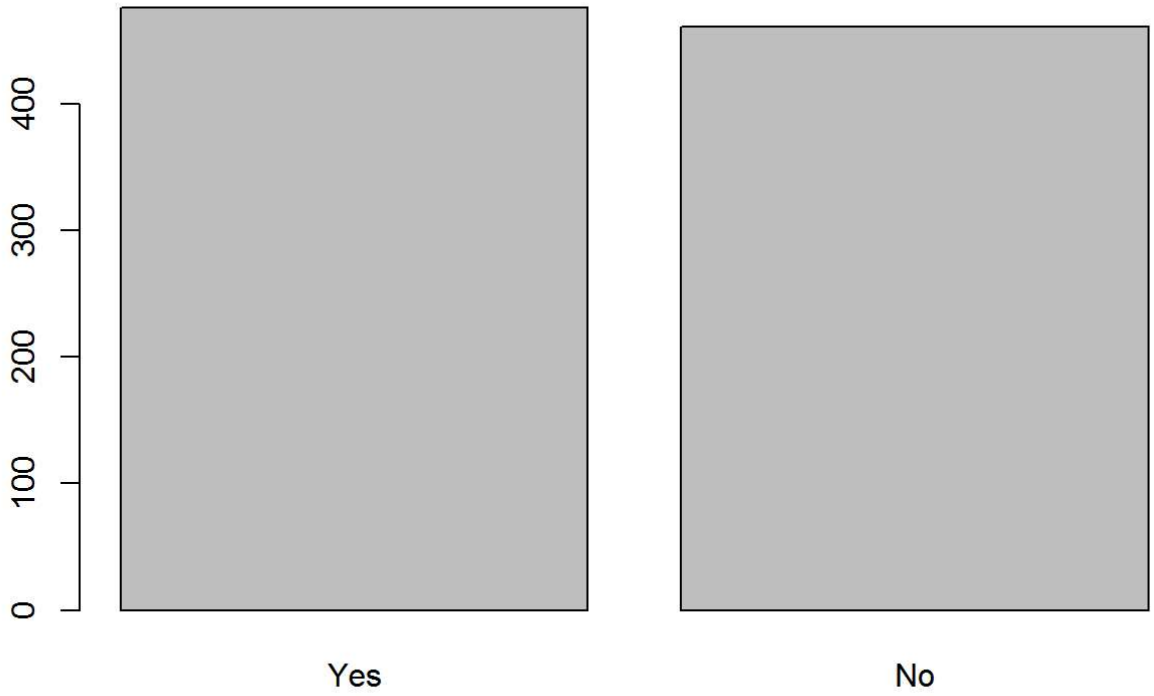
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 431  19
##        No   12 475
##
##                Accuracy : 0.9669
##                  95% CI : (0.9534, 0.9774)
##     No Information Rate : 0.5272
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9337
##  Mcnemar's Test P-Value : 0.2812
##
##             Sensitivity : 0.9729
##             Specificity : 0.9615
##          Pos Pred Value : 0.9578
##          Neg Pred Value : 0.9754
##              Prevalence : 0.4728
##          Detection Rate : 0.4600
##    Detection Prevalence : 0.4803
##       Balanced Accuracy : 0.9672
##
##        'Positive' Class : Yes
##
```

```
roc.curve(svm.predict, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.967
```

```
## Applying KNN Algorithm
knn.model <- knn(train = trainSplit2[,1:30],
    test = testSplit2[,1:30],
    cl = trainSplit2$Class)
plot(knn.model)
```



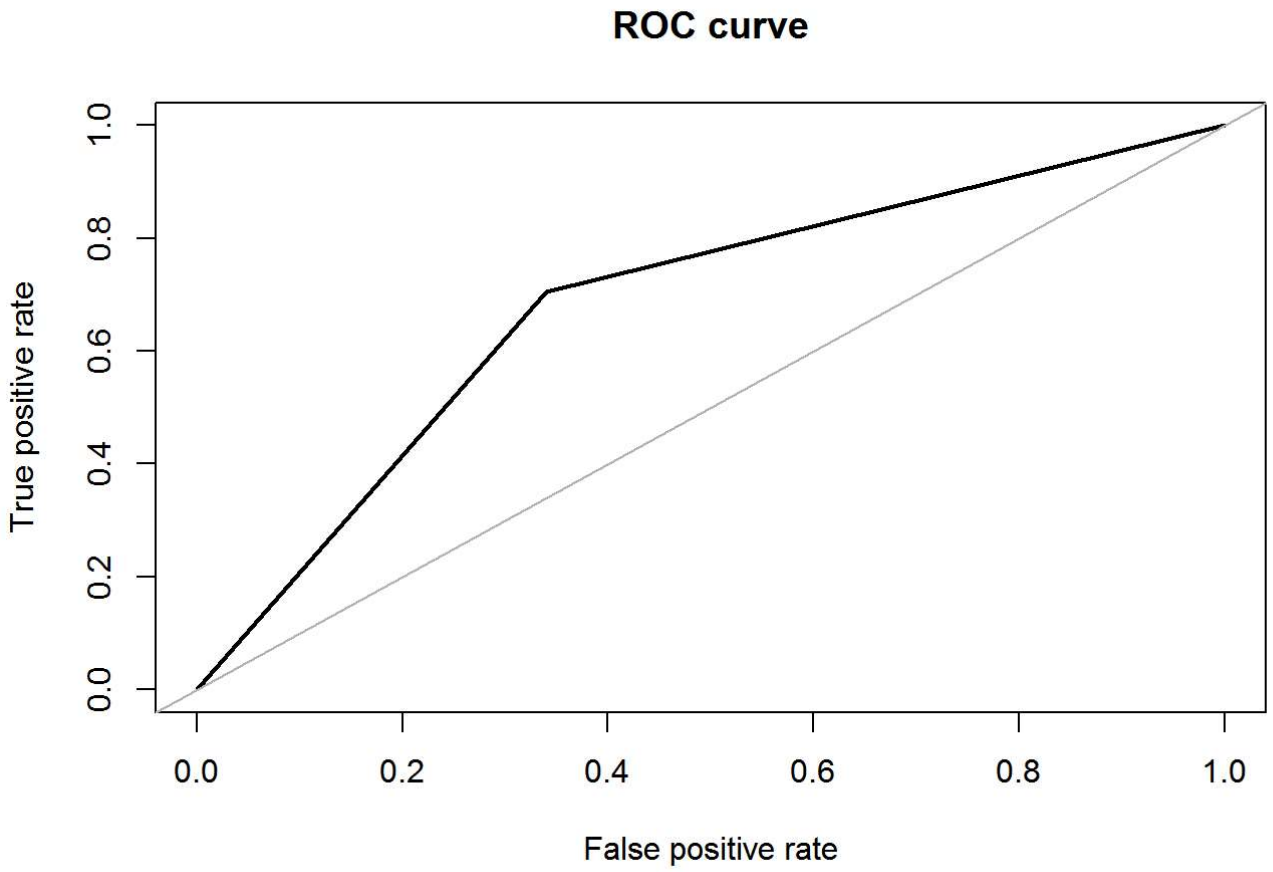

```
table(knn.model, testSplit2$Class)
```

```
##
## knn.model Yes  No
##       Yes 314 162
##       No  136 325
```

```
confusionMatrix(knn.model, testSplit2[,31])
```

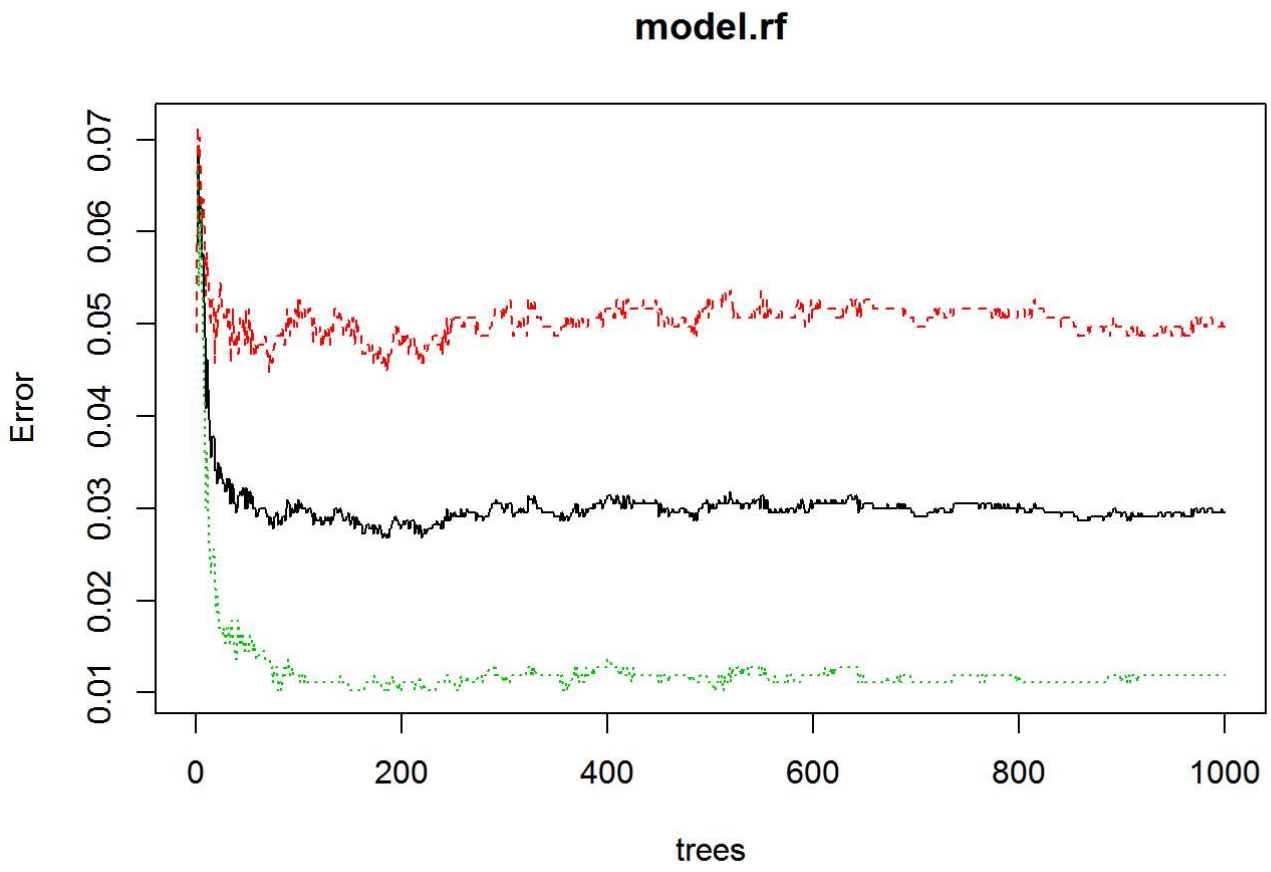
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 314 162
##        No  136 325
##
##                Accuracy : 0.682
##                  95% CI : (0.6511, 0.7117)
##     No Information Rate : 0.5197
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.3643
##  Mcnemar's Test P-Value : 0.1476
##
##             Sensitivity : 0.6978
##             Specificity : 0.6674
##          Pos Pred Value : 0.6597
##          Neg Pred Value : 0.7050
##              Prevalence : 0.4803
##          Detection Rate : 0.3351
##    Detection Prevalence : 0.5080
##       Balanced Accuracy : 0.6826
##
##        'Positive' Class : Yes
##
```

```
roc.curve(knn.model, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.682
```

```
### Applying Random Forest algorithm for 75% fraud data
model.rf <- randomForest(Class ~ ., data =trainSplit2 , ntree = 1000, importance = TRUE)
plot(model.rf)
```



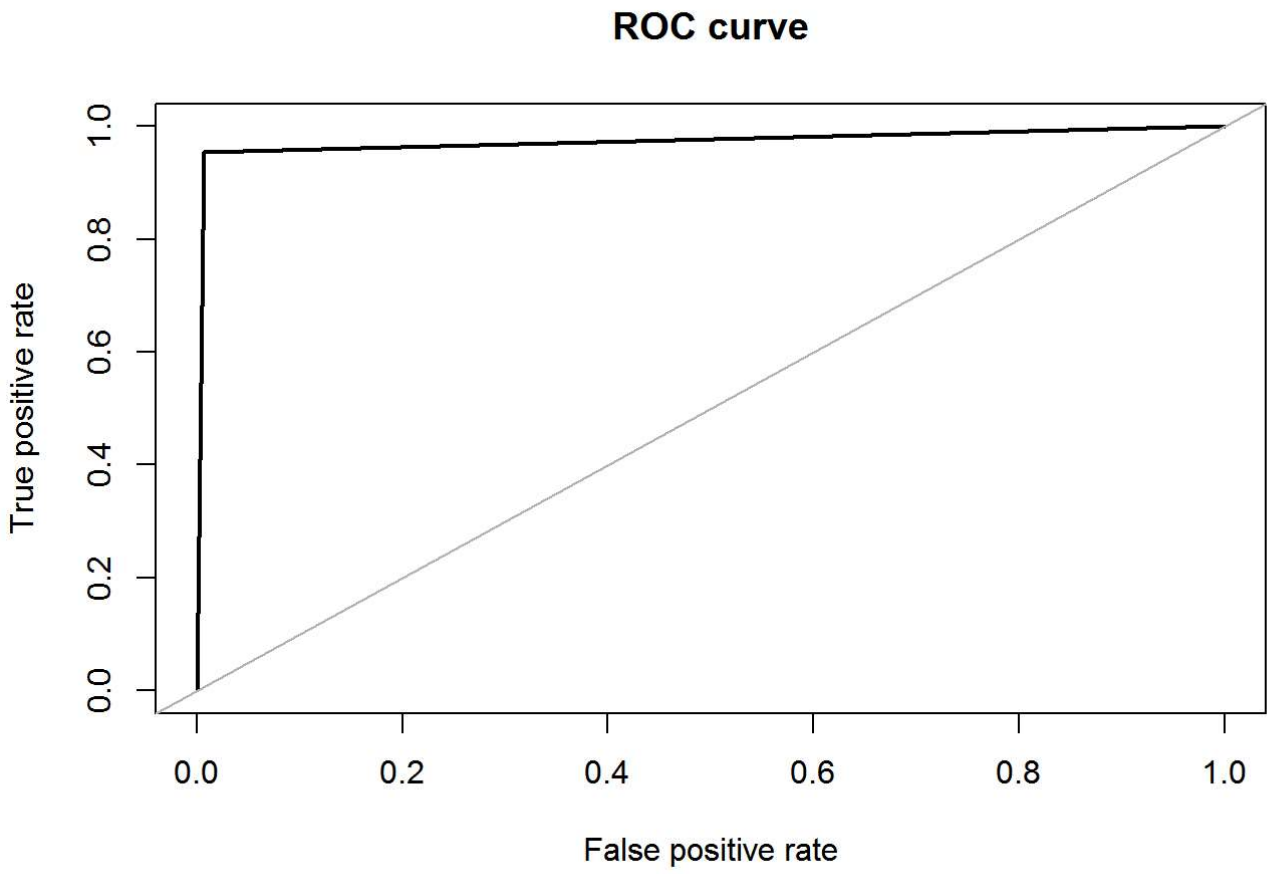

```
cv.tree.pred2 <- predict(model.rf, testSplit2)

# Making table of Confusion matrix
CrossTable(cv.tree.pred2, testSplit2$Class,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual class', 'predicted class'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  937
##
##
##              | predicted class
## actual class |       Yes |        No | Row Total |
## -------------|-----------|-----------|-----------|
##          Yes |       427 |         3 |       430 |
##              |     0.456 |     0.003 |           |
## -------------|-----------|-----------|-----------|
##           No |        23 |       484 |       507 |
##              |     0.025 |     0.517 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       450 |       487 |       937 |
## -------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(cv.tree.pred2, testSplit2$Class)
```

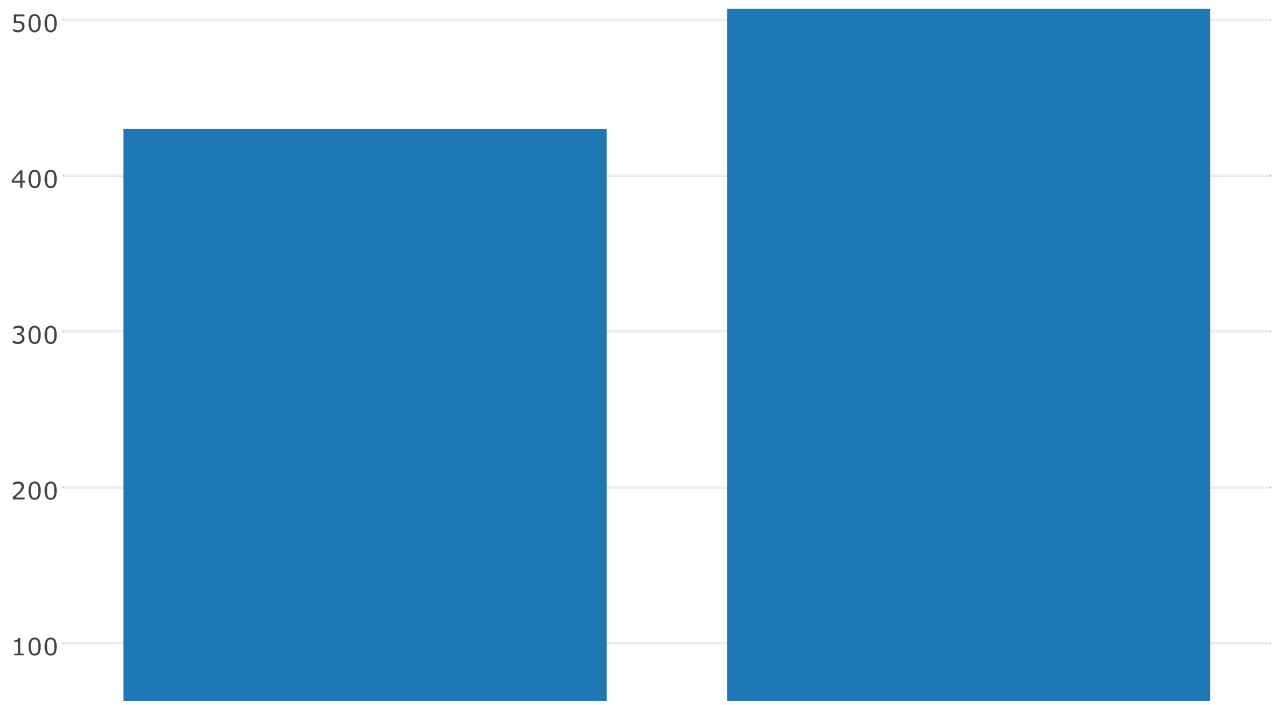
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 427   3
##        No   23 484
##
##                Accuracy : 0.9723
##                  95% CI : (0.9596, 0.9818)
##     No Information Rate : 0.5197
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9443
##  Mcnemar's Test P-Value : 0.0001944
##
##             Sensitivity : 0.9489
##             Specificity : 0.9938
##          Pos Pred Value : 0.9930
##          Neg Pred Value : 0.9546
##              Prevalence : 0.4803
##          Detection Rate : 0.4557
##    Detection Prevalence : 0.4589
##       Balanced Accuracy : 0.9714
##
##        'Positive' Class : Yes
##
```

```
roc.curve(cv.tree.pred2, testSplit2$Class, plotit = T)
```




```
## Area under the curve (AUC): 0.974
```

```
x<-rchisq(1000,5,0)
plot_ly(x=cv.tree.pred2,type = 'histogram')
```



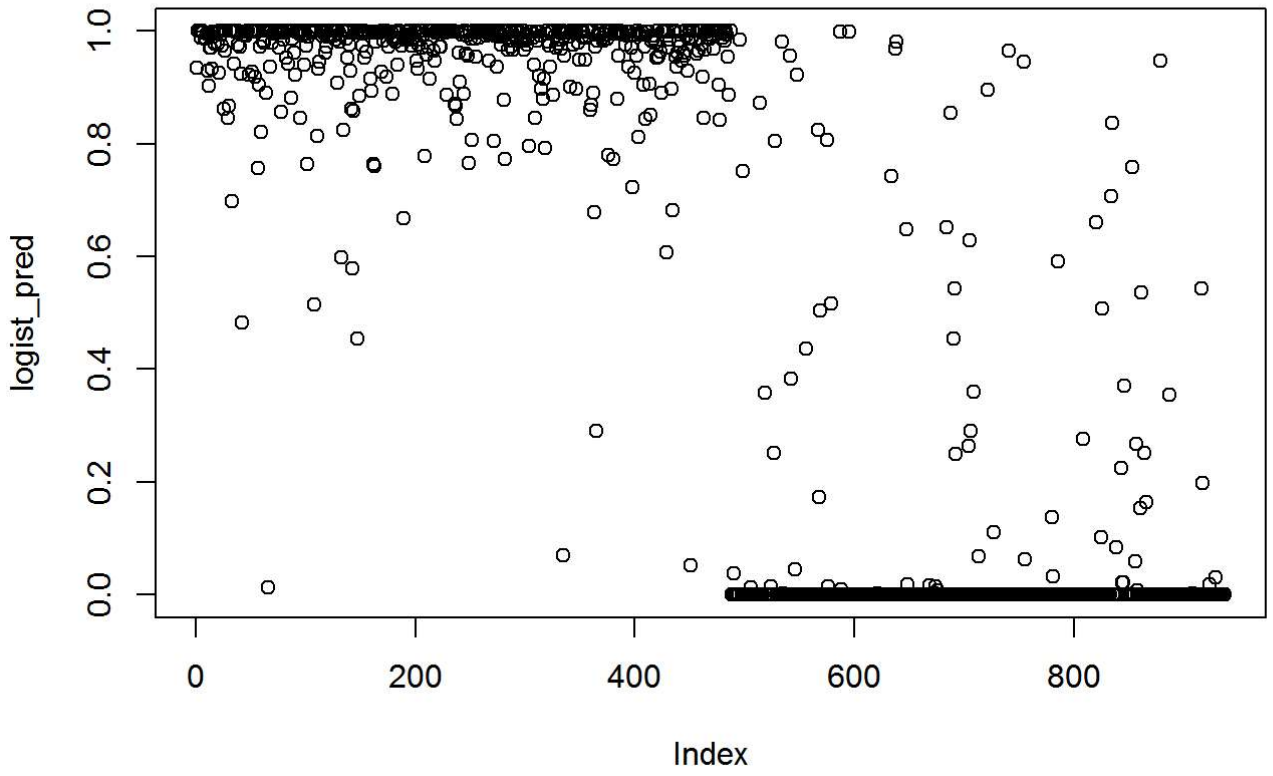

```
### Applying Logistic Regression Algorithm for 75% fraud data
logist <- glm(Class ~ ., data = trainSplit2, family = "binomial")
```

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

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

```
summary(logist)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = trainSplit2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6113   0.0000   0.0000   0.1607   3.0528
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.515e+01  2.701e+01   0.931 0.351820
## Time         1.015e-05  4.314e-06   2.354 0.018594 *
## V1           7.164e+00  6.644e+00   1.078 0.280949
## V2          -5.338e+01  3.983e+01  -1.340 0.180183
## V3           4.507e+01  1.681e+01   2.681 0.007337 **
## V4          -3.114e+01  1.318e+01  -2.362 0.018155 *
## V5           1.847e+01  5.132e+00   3.599 0.000319 ***
## V6           2.537e+01  1.812e+01   1.400 0.161523
## V7           1.014e+02  6.275e+01   1.616 0.106192
## V8          -1.917e+01  1.074e+01  -1.785 0.074200 .
## V9           3.830e+01  1.945e+01   1.969 0.048993 *
## V10          8.962e+01  4.477e+01   2.002 0.045296 *
## V11         -6.810e+01  3.767e+01  -1.808 0.070625 .
## V12          1.227e+02  6.765e+01   1.814 0.069634 .
## V13          1.480e+00  1.768e+00   0.837 0.402575
## V14          1.312e+02  7.371e+01   1.779 0.075185 .
## V15          3.495e+00  2.612e+00   1.338 0.180820
## V16          1.158e+02  6.508e+01   1.780 0.075051 .
## V17          2.077e+02  1.144e+02   1.816 0.069324 .
## V18          7.892e+01  4.367e+01   1.807 0.070737 .
## V19         -2.973e+01  1.801e+01  -1.651 0.098723 .
## V20          6.410e+00  1.225e+01   0.523 0.600936
## V21         -1.082e+01  3.448e+00  -3.138 0.001699 **
## V22         -9.449e+00  7.923e+00  -1.193 0.233020
## V23         -2.372e+01  2.392e+01  -0.991 0.321507
## V24          2.870e+00  2.291e+00   1.253 0.210274
## V25         -1.291e+01  1.089e+01  -1.186 0.235554
## V26         -2.388e+00  2.760e+00  -0.865 0.386868
## V27         -1.735e+01  8.843e+00  -1.962 0.049756 *
## V28         -3.276e+01  2.893e+01  -1.132 0.257567
## Amount      -2.825e-01  2.763e-01  -1.022 0.306585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3038.63  on 2198  degrees of freedom
## Residual deviance:  431.95  on 2168  degrees of freedom
## AIC: 493.95
##
## Number of Fisher Scoring iterations: 25
```

```
logist_pred <- predict(logist, newdata=testSplit2,type = "response")
plot(logist_pred)
```

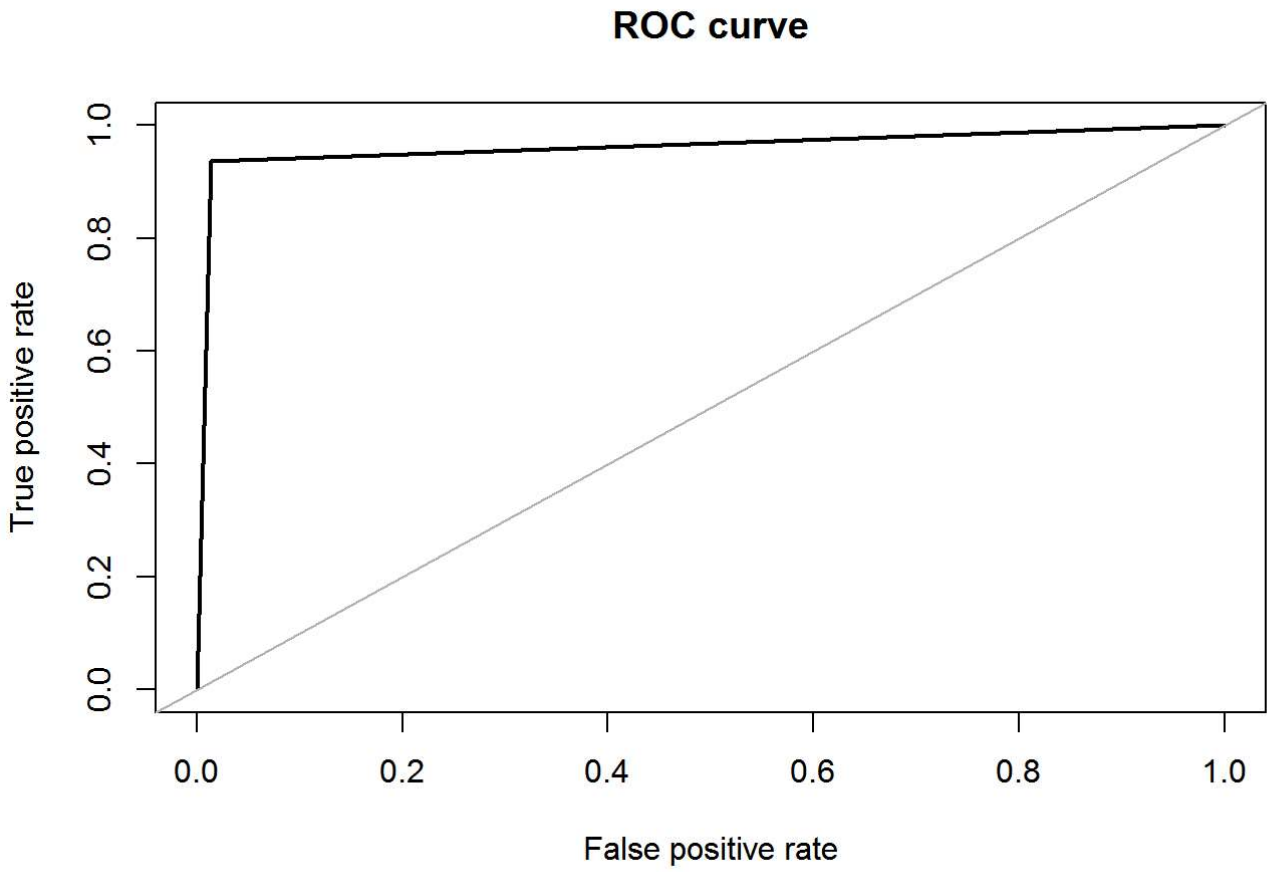



```
pred=rep("Yes",length(logist_pred))
pred[logist_pred > 0.5] = "No"
confusionMatrix(testSplit2$Class, pred)
```

```
## Warning in confusionMatrix.default(testSplit2$Class, pred): Levels are not
## in the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  481   6
##        Yes  33 417
##
##                Accuracy : 0.9584
##                  95% CI : (0.9435, 0.9702)
##     No Information Rate : 0.5486
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9164
##  Mcnemar's Test P-Value : 3.136e-05
##
##             Sensitivity : 0.9358
##             Specificity : 0.9858
##          Pos Pred Value : 0.9877
##          Neg Pred Value : 0.9267
##              Prevalence : 0.5486
##          Detection Rate : 0.5133
##    Detection Prevalence : 0.5197
##       Balanced Accuracy : 0.9608
##
##        'Positive' Class : No
##
```

```
roc.curve(pred, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.961
```

```
#############################################################################
### New dataset with 100% fraud data
fraud100 = fraudData[1:492,]

dataWith100Fraud = rbind(NofraudData,fraud100)

#Randomize the data for 100% fraud
dataFor100 <- dataWith100Fraud[sample(nrow(dataWith100Fraud)),]
table(dataFor100$Class)
```

```
##
##    Yes     No
##    492 284315
```

```
###removing unbalanced classification problemand making data balanced
set.seed(4356)
smote_data <- SMOTE(Class ~ ., data  = dataFor100, perc.over = 300, perc.under = 150, k=5)
new.data <- sample(2, nrow(smote_data), replace = TRUE, prob = c(0.7, 0.3))
trainSplit2 <-smote_data[new.data==1,]
testSplit2 <-smote_data[new.data==2,]
table(trainSplit2$Class)
```

```
##
##  Yes   No
## 1360 1560
```

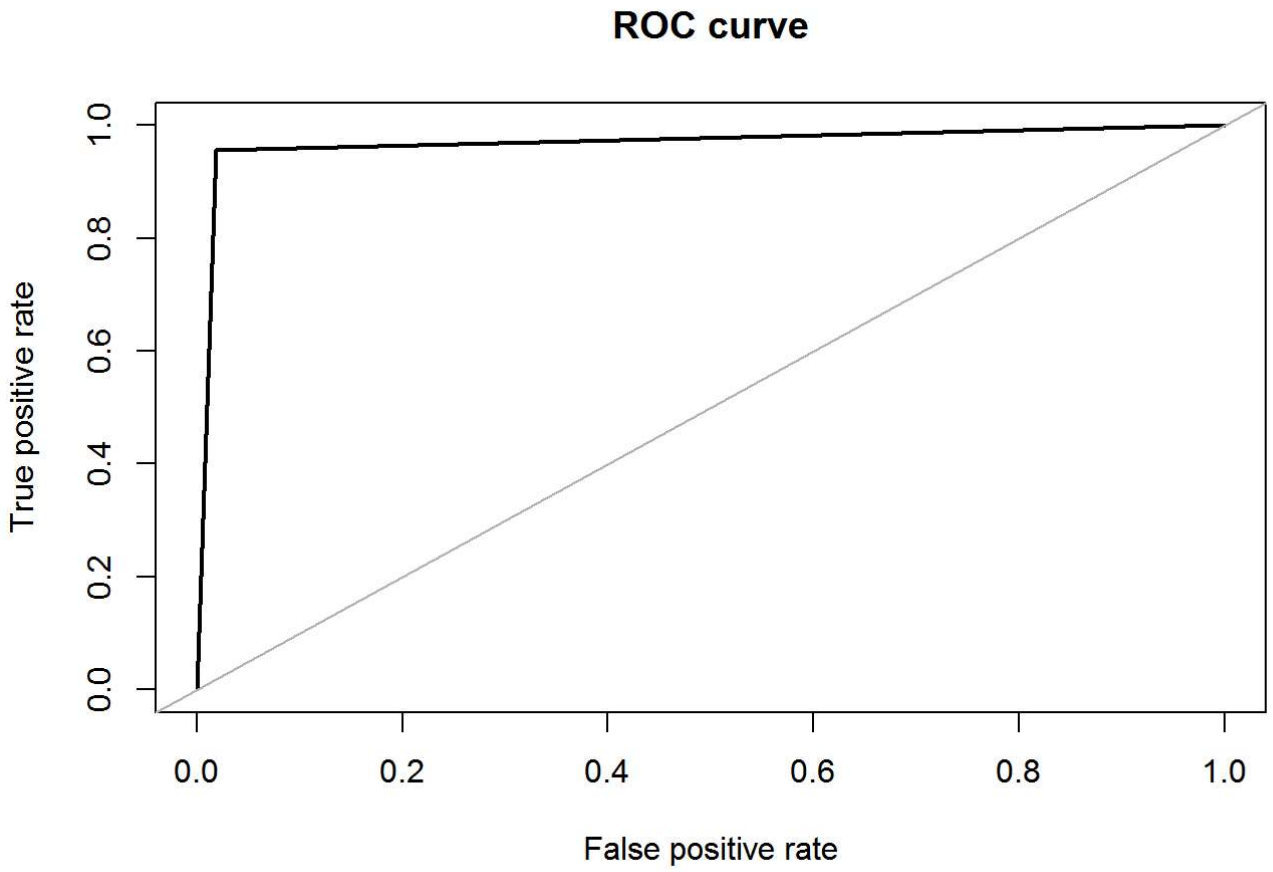
```
table(testSplit2$Class)
```

```
##
## Yes  No
## 608 654
```

```
### Applying SVM algorithm for 100% fraud data
svm.model <- svm(Class ~ ., data = trainSplit2, kernel = "radial", cost = 1, gamma = 0.1)
svm.predict <- predict(svm.model, testSplit2)
confusionMatrix(testSplit2$Class, svm.predict)
```

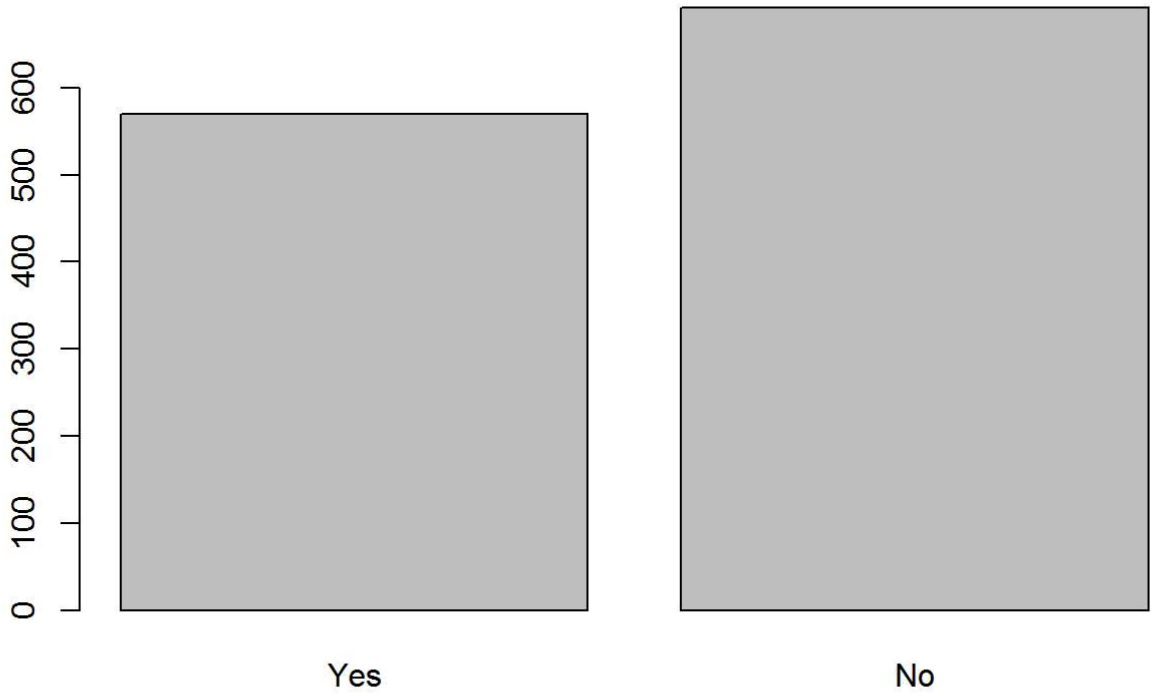
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 579  29
##        No   11 643
##
##                Accuracy : 0.9683
##                  95% CI : (0.9571, 0.9773)
##     No Information Rate : 0.5325
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9365
##  Mcnemar's Test P-Value : 0.00719
##
##             Sensitivity : 0.9814
##             Specificity : 0.9568
##          Pos Pred Value : 0.9523
##          Neg Pred Value : 0.9832
##              Prevalence : 0.4675
##          Detection Rate : 0.4588
##    Detection Prevalence : 0.4818
##       Balanced Accuracy : 0.9691
##
##        'Positive' Class : Yes
##
```

```
roc.curve(svm.predict, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.969
```

```
## Applying KNN Algorithm
knn.model <- knn(train = trainSplit2[,1:30],
    test = testSplit2[,1:30],
    cl = trainSplit2$Class)
plot(knn.model)
```



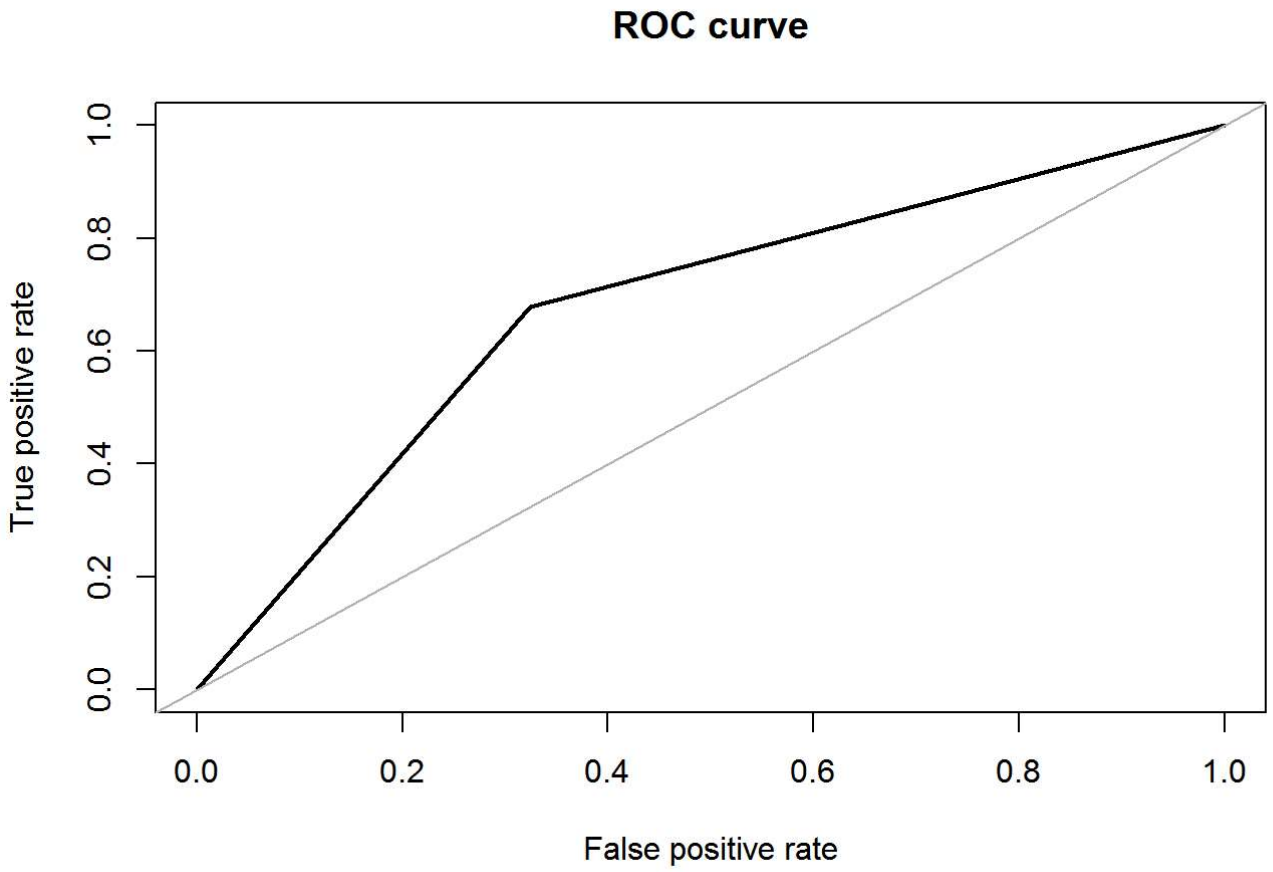

```
table(knn.model, testSplit2$Class)
```

```
##
## knn.model Yes  No
##       Yes 385 185
##       No  223 469
```

```
confusionMatrix(knn.model, testSplit2[,31])
```

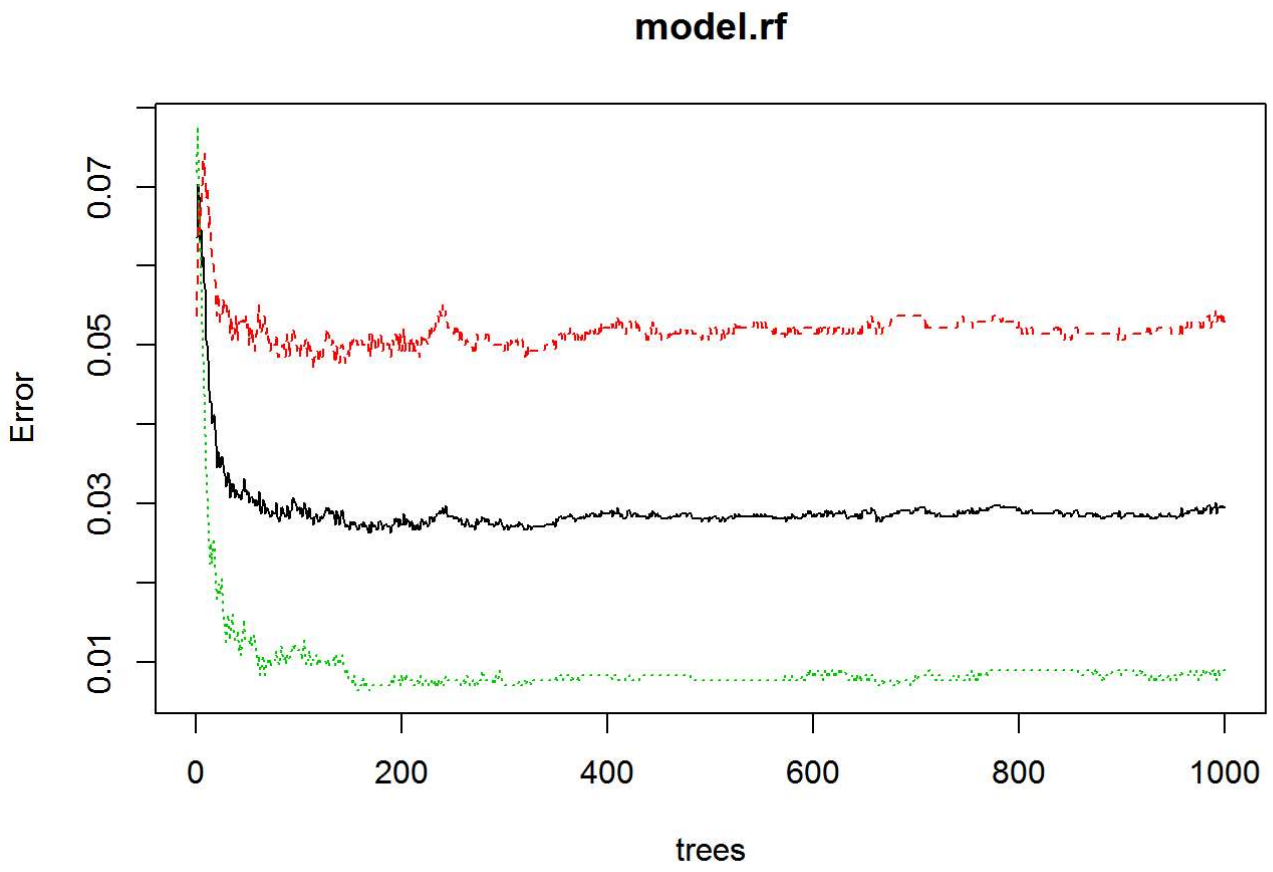
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 385 185
##        No  223 469
##
##                Accuracy : 0.6767
##                  95% CI : (0.6501, 0.7025)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.3511
##  Mcnemar's Test P-Value : 0.06699
##
##             Sensitivity : 0.6332
##             Specificity : 0.7171
##          Pos Pred Value : 0.6754
##          Neg Pred Value : 0.6777
##              Prevalence : 0.4818
##          Detection Rate : 0.3051
##    Detection Prevalence : 0.4517
##       Balanced Accuracy : 0.6752
##
##        'Positive' Class : Yes
##
```

```
roc.curve(knn.model, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.677
```

```
### Applying Random Forest algorithm for 100% fraud data
model.rf <- randomForest(Class ~ ., data =trainSplit2 , ntree = 1000, importance = TRUE)
plot(model.rf)
```



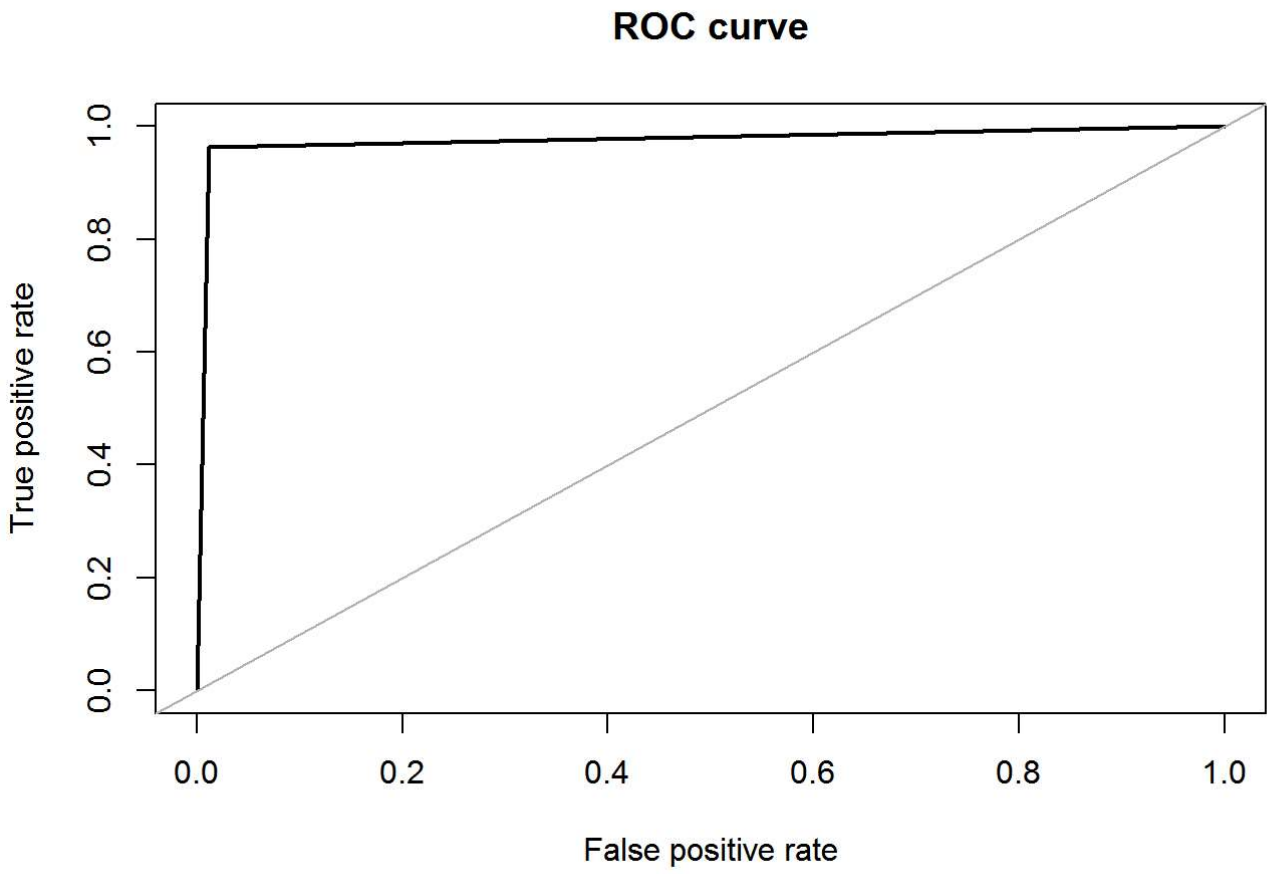

```
cv.tree.pred2 <- predict(model.rf, testSplit2)

# Making table of Confusion matrix
CrossTable(cv.tree.pred2, testSplit2$Class,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual class', 'predicted class'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1262
##
##
##              | predicted class
## actual class |       Yes |        No | Row Total |
## -------------|-----------|-----------|-----------|
##          Yes |       584 |         7 |       591 |
##              |     0.463 |     0.006 |           |
## -------------|-----------|-----------|-----------|
##           No |        24 |       647 |       671 |
##              |     0.019 |     0.513 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       608 |       654 |      1262 |
## -------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(cv.tree.pred2, testSplit2$Class)
```

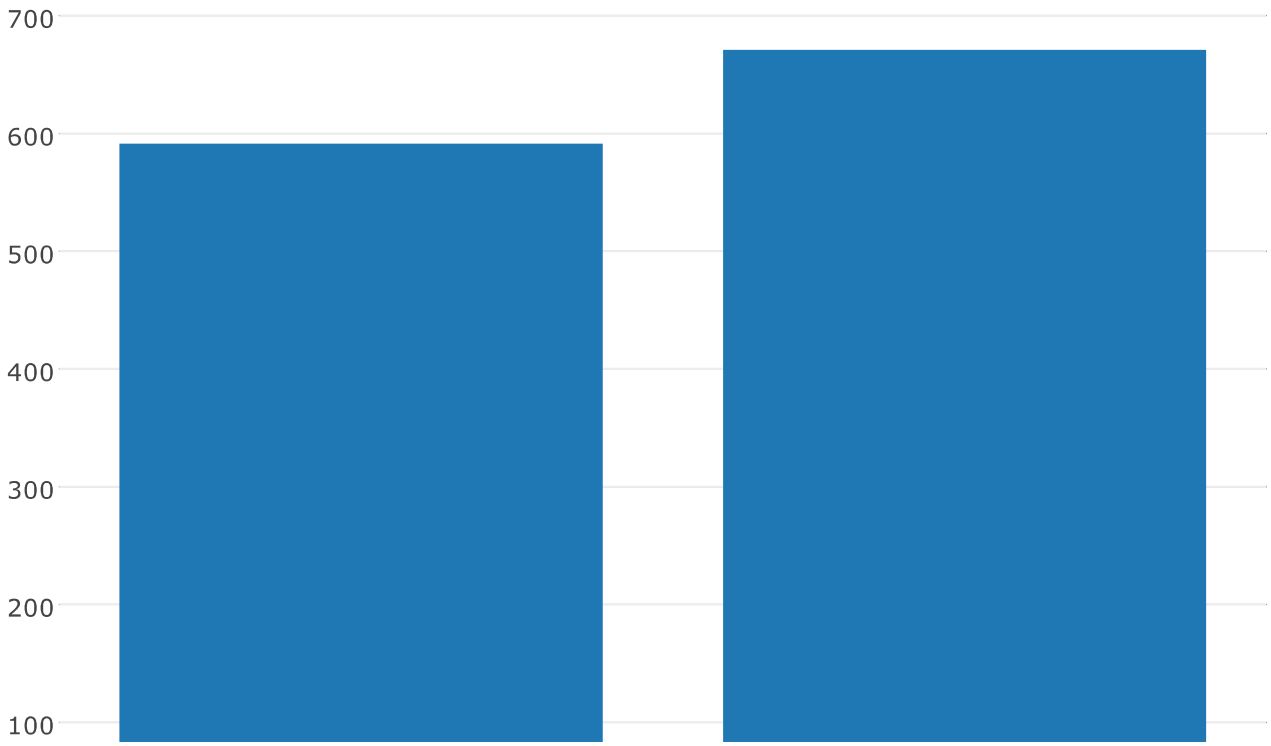
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 584   7
##        No   24 647
##
##                Accuracy : 0.9754
##                  95% CI : (0.9653, 0.9833)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9508
##  Mcnemar's Test P-Value : 0.004057
##
##             Sensitivity : 0.9605
##             Specificity : 0.9893
##          Pos Pred Value : 0.9882
##          Neg Pred Value : 0.9642
##              Prevalence : 0.4818
##          Detection Rate : 0.4628
##    Detection Prevalence : 0.4683
##       Balanced Accuracy : 0.9749
##
##        'Positive' Class : Yes
##
```

```
roc.curve(cv.tree.pred2, testSplit2$Class, plotit = T)
```




```
## Area under the curve (AUC): 0.976
```

```
x<-rchisq(1000,5,0)
plot_ly(x=cv.tree.pred2,type = 'histogram')
```



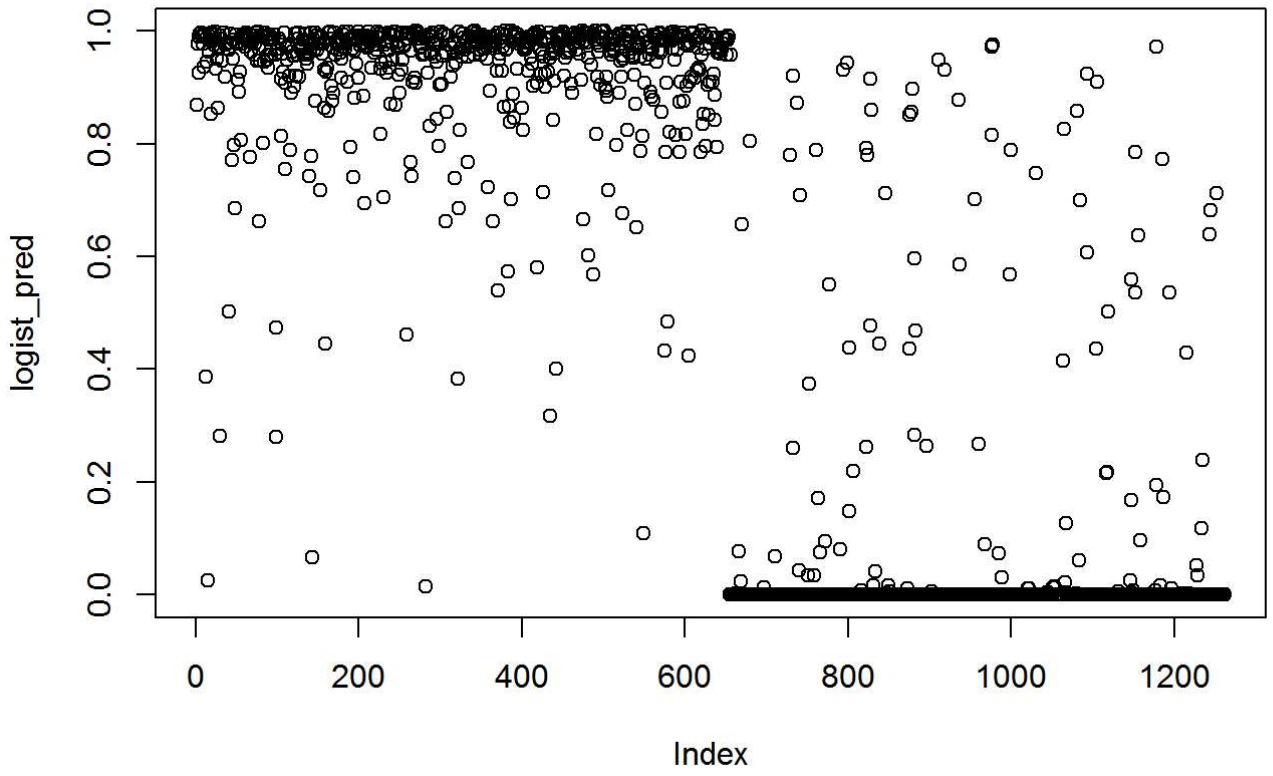

```
### Applying Logistic Regression Algorithm for 100% fraud data
logist <- glm(Class ~ ., data = trainSplit2, family = "binomial")
```

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

```
summary(logist)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = trainSplit2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1263  -0.0002   0.0758   0.2410   6.7890
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.014e+00  3.686e-01   8.179 2.86e-16 ***
## Time         8.538e-06  3.227e-06   2.646 0.008146 **
## V1          -2.193e-01  9.681e-02  -2.265 0.023519 *
## V2          -1.267e-01  2.085e-01  -0.608 0.543189
## V3           5.379e-02  9.909e-02   0.543 0.587224
## V4          -7.667e-01  9.896e-02  -7.748 9.37e-15 ***
## V5          -2.678e-01  1.304e-01  -2.054 0.039987 *
## V6           2.812e-01  1.261e-01   2.230 0.025773 *
## V7          -2.300e-02  1.544e-01  -0.149 0.881616
## V8           4.041e-01  1.043e-01   3.875 0.000107 ***
## V9           1.914e-01  1.434e-01   1.334 0.182217
## V10          3.864e-01  1.943e-01   1.988 0.046770 *
## V11         -3.108e-01  1.145e-01  -2.715 0.006618 **
## V12          8.151e-01  1.401e-01   5.819 5.92e-09 ***
## V13          3.726e-01  9.637e-02   3.866 0.000111 ***
## V14          1.050e+00  1.451e-01   7.237 4.58e-13 ***
## V15          4.615e-02  1.098e-01   0.420 0.674227
## V16          1.932e-01  1.570e-01   1.231 0.218441
## V17          2.124e-01  1.701e-01   1.249 0.211704
## V18          2.920e-02  1.634e-01   0.179 0.858154
## V19         -2.369e-01  1.262e-01  -1.877 0.060470 .
## V20          3.399e-01  2.352e-01   1.445 0.148519
## V21         -9.448e-02  1.187e-01  -0.796 0.426045
## V22         -5.936e-01  1.655e-01  -3.586 0.000336 ***
## V23          1.945e-01  1.629e-01   1.194 0.232509
## V24          1.060e-01  1.993e-01   0.532 0.594905
## V25          1.918e-02  2.377e-01   0.081 0.935698
## V26          7.133e-01  2.558e-01   2.789 0.005295 **
## V27          1.495e-01  3.095e-01   0.483 0.629016
## V28         -2.511e-01  4.531e-01  -0.554 0.579483
## Amount      -2.797e-03  1.819e-03  -1.538 0.124080
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4034.3  on 2919  degrees of freedom
## Residual deviance:  764.1  on 2889  degrees of freedom
## AIC: 826.1
##
## Number of Fisher Scoring iterations: 12
```

```
logist_pred <- predict(logist, newdata=testSplit2,type = "response")
plot(logist_pred)
```

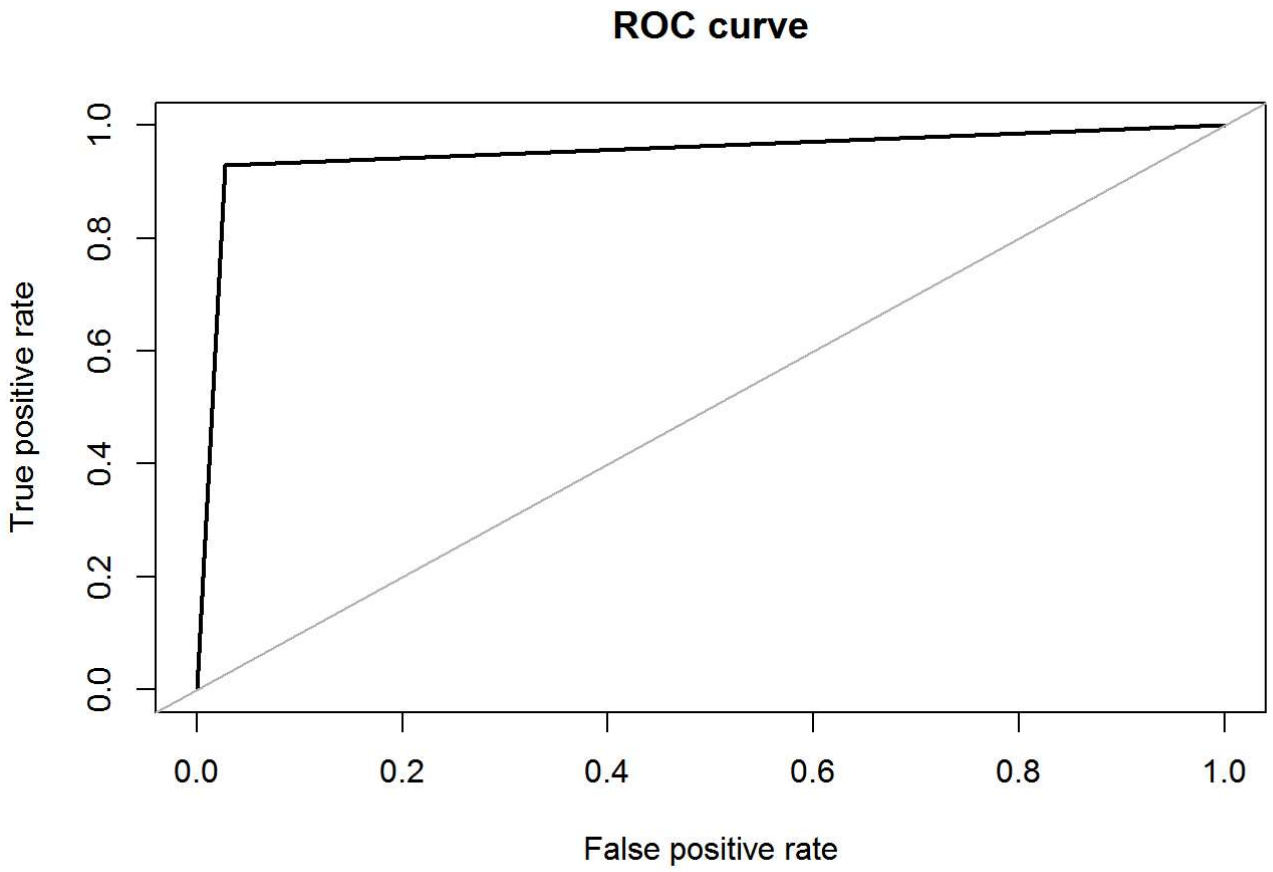



```
pred=rep("Yes",length(logist_pred))
pred[logist_pred > 0.5] = "No"
confusionMatrix(testSplit2$Class, pred)
```

```
## Warning in confusionMatrix.default(testSplit2$Class, pred): Levels are not
## in the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  638  16
##        Yes  48 560
##
##                Accuracy : 0.9493
##                  95% CI : (0.9357, 0.9607)
##     No Information Rate : 0.5436
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8983
##  Mcnemar's Test P-Value : 0.0001066
##
##             Sensitivity : 0.9300
##             Specificity : 0.9722
##          Pos Pred Value : 0.9755
##          Neg Pred Value : 0.9211
##              Prevalence : 0.5436
##          Detection Rate : 0.5055
##    Detection Prevalence : 0.5182
##       Balanced Accuracy : 0.9511
##
##        'Positive' Class : No
##
```

```
roc.curve(pred, testSplit2$Class, plotit = T)
```



```
## Area under the curve (AUC): 0.951
```