Credit Card Fraud Detection

Puneeth Kondarasi, Gagan Karnam

2025-05-13

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## 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
```

```r
library(ranger) #random forest
```

```
## Warning: package 'ranger' was built under R version 4.3.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

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

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

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

```r
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.3.3
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(ggplot2)
library(ROSE) #Data balancing
```

```
## Warning: package 'ROSE' was built under R version 4.3.3
```

```
## Loaded ROSE 0.0-4
```

```r
library(pROC) #ROC
```

```
## Warning: package 'pROC' was built under R version 4.3.3
```

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

```
##
## Attaching package: 'pROC'
```

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

```r
library(rpart) #decision tree
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```r
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
```

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

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

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.3.3
```

```r
dataset <- setDT(read.csv("C:/Users/punee/Downloads/creditcard.csv"))



# Data exploration
head(dataset)
```

```
##     Time         V1          V2        V3         V4          V5          V6
##    <num>      <num>       <num>     <num>      <num>       <num>       <num>
## 1:     0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2:     0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3:     1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4:     1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5:     2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6:     2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##             V7          V8         V9         V10        V11         V12
##          <num>       <num>      <num>       <num>      <num>       <num>
## 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
##          <num>      <num>      <num>      <num>       <num>       <num>
## 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
##           <num>       <num>        <num>        <num>       <num>       <num>
## 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
##          <num>      <num>        <num>       <num>  <num> <int>
## 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
```

```r
tail(dataset)
```

```
##       Time          V1          V2         V3         V4          V5         V6
##      <num>       <num>       <num>      <num>      <num>       <num>      <num>
## 1: 172785   0.1203164  0.93100513 -0.5460121 -0.7450968  1.13031398 -0.2359732
## 2: 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278 -2.6068373
## 3: 172787  -0.7327887 -0.05508049  2.0350297 -0.7385886  0.86822940  1.0584153
## 4: 172788   1.9195650 -0.30125385 -3.2496398 -0.5578281  2.63051512  3.0312601
## 5: 172788  -0.2404400  0.53048251  0.7025102  0.6897992 -0.37796113  0.6237077
## 6: 172792  -0.5334125 -0.18973334  0.7033374 -0.5062712 -0.01254568 -0.6496167
##            V7         V8         V9        V10        V11         V12
##         <num>      <num>      <num>      <num>      <num>       <num>
## 1:  0.8127221  0.1150929 -0.2040635 -0.6574221  0.6448373  0.19091623
## 2: -4.9182154  7.3053340  1.9144283  4.3561704 -1.5931053  2.71194079
## 3:  0.0243297  0.2948687  0.5848000 -0.9759261 -0.1501888  0.91580191
## 4: -0.2968265  0.7084172  0.4324540 -0.4847818  0.4116137  0.06311886
## 5: -0.6861800  0.6791455  0.3920867 -0.3991257 -1.9338488 -0.96288614
## 6:  1.5770063 -0.4146504  0.4861795 -0.9154266 -1.0404583 -0.03151305
##           V13         V14         V15        V16         V17        V18
##         <num>       <num>       <num>      <num>       <num>      <num>
## 1: -0.5463289 -0.73170658 -0.80803553  0.5996281  0.07044075  0.3731103
## 2: -0.6892556  4.62694203 -0.92445871  1.1076406  1.99169111  0.5106323
## 3:  1.2147558 -0.67514296  1.16493091 -0.7117573 -0.02569286 -1.2211789
## 4: -0.1836987 -0.51060184  1.32928351  0.1407160  0.31350179  0.3956525
## 5: -1.0420817  0.44962444  1.96256312 -0.6085771  0.50992846  1.1139806
## 6: -0.1880929 -0.08431647  0.04133346 -0.3026201 -0.66037665  0.1674299
##           V19          V20        V21        V22         V23          V24
##         <num>        <num>      <num>      <num>       <num>        <num>
## 1:  0.1289038 0.0006758329 -0.3142046 -0.8085204  0.05034266  0.102799590
## 2: -0.6829197 1.4758291347  0.2134541  0.1118637  1.01447990 -0.509348453
## 3: -1.5455561 0.0596158999  0.2142053  0.9243836  0.01246304 -1.016225669
## 4: -0.5772518 0.0013959703  0.2320450  0.5782290 -0.03750086  0.640133881
## 5:  2.8978488 0.1274335158  0.2652449  0.8000487 -0.16329794  0.123205244
## 6: -0.2561169 0.3829481049  0.2610573  0.6430784  0.37677701  0.008797379
##           V25        V26          V27         V28 Amount Class
##         <num>      <num>        <num>       <num>  <num> <int>
## 1: -0.4358701  0.1240789  0.217939865  0.06880333   2.69     0
## 2:  1.4368069  0.2500343  0.943651172  0.82373096   0.77     0
## 3: -0.6066240 -0.3952551  0.068472470 -0.05352739  24.79     0
## 4:  0.2657455 -0.0873706  0.004454772 -0.02656083  67.88     0
## 5: -0.5691589  0.5466685  0.108820735  0.10453282  10.00     0
## 6: -0.4736487 -0.8182671 -0.002415309  0.01364891 217.00     0
```

```
table(dataset$Class)
```

```
##
##      0      1
## 284315    492
```

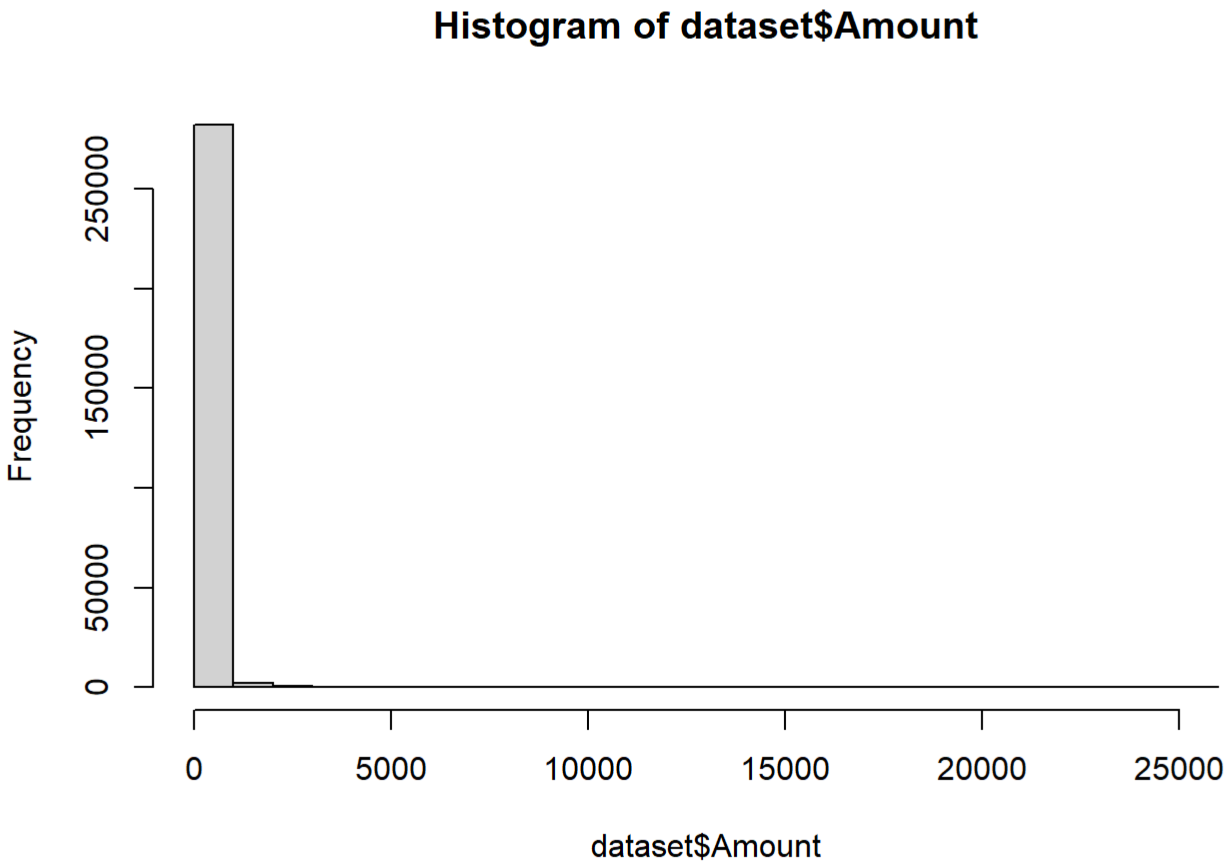
```
summary(dataset$Amount)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     5.60    22.00    88.35    77.17 25691.16
```

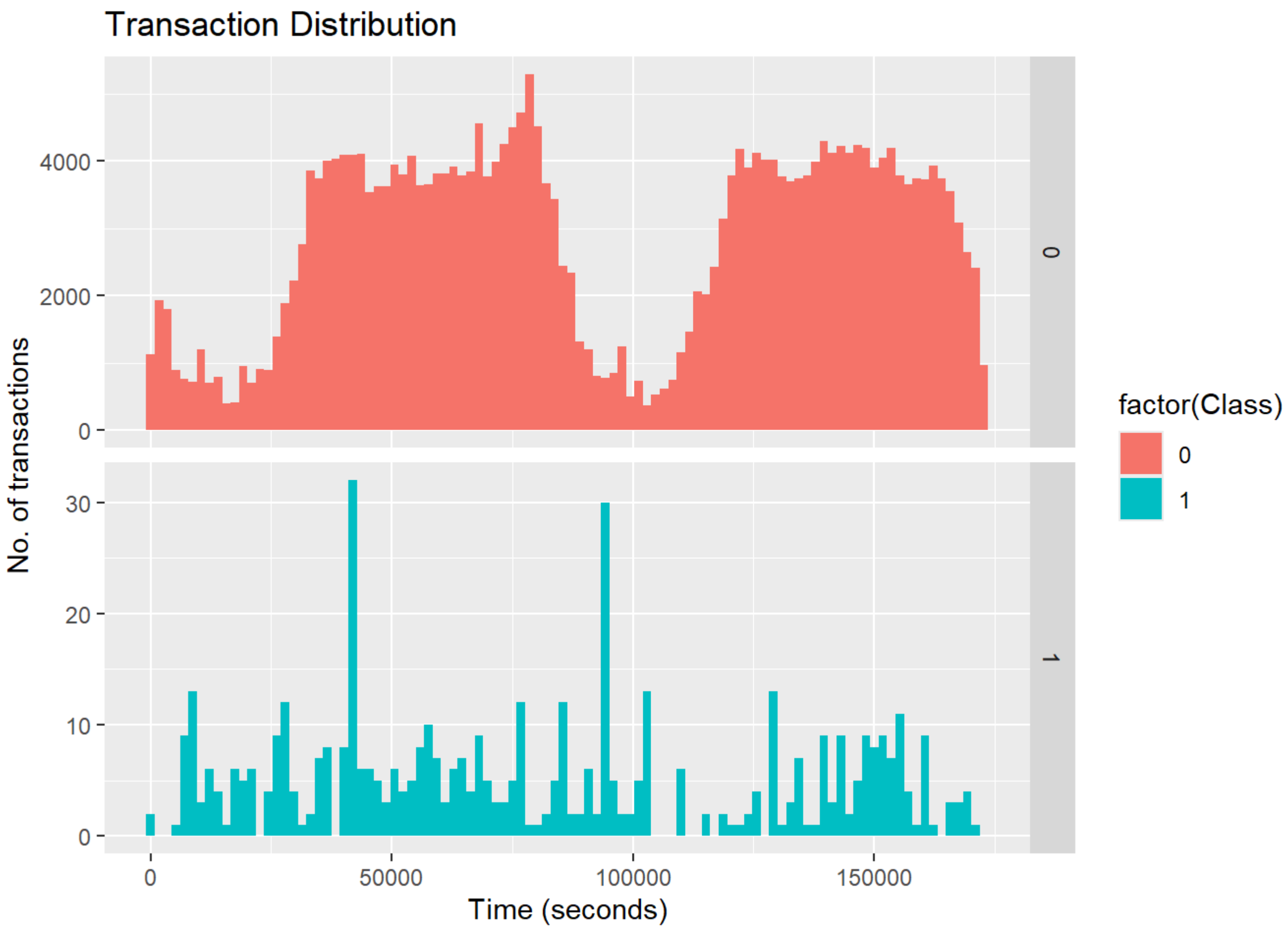
```
colSums(is.na(dataset))
```

```
##    Time      V1      V2      V3      V4      V5      V6      V7      V8      V9     V10
##       0       0       0       0       0       0       0       0       0       0       0
##     V11     V12     V13     V14     V15     V16     V17     V18     V19     V20     V21
##       0       0       0       0       0       0       0       0       0       0       0
##     V22     V23     V24     V25     V26     V27     V28  Amount   Class
##       0       0       0       0       0       0       0       0       0
```

```
hist(dataset$Amount)
```




```
# Data visualization
dataset %>%
  ggplot(aes(x = Time, fill = factor(Class))) +
  geom_histogram(bins = 100) +
  labs(x = "Time (seconds)", y = "No. of transactions", title = "Transaction Distribution") +
  facet_grid(Class ~ ., scales = 'free_y') + theme()
```




```
# Feature Scaling
dataset$Amount <- scale(dataset$Amount)


# Prepare dataset
new_data <- dataset[, -c(1)]
new_data$Class <- as.factor(new_data$Class)
levels(new_data$Class) <- c("Not Fraud", "Fraud")


# Train-test split
set.seed(101)
split <- sample.split(new_data$Class, SplitRatio = 0.8)
train_data <- subset(new_data, split == TRUE)
test_data <- subset(new_data, split == FALSE)


# Show train and test data samples
print("Train Data Sample:")
```

```
## [1] "Train Data Sample:"
```

```
head(train_data)
```

```
##            V1          V2        V3         V4          V5          V6
##         <num>       <num>     <num>      <num>       <num>       <num>
## 1: -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2:  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3: -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4: -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5: -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6: -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##             V7          V8         V9         V10        V11         V12
##          <num>       <num>      <num>       <num>      <num>       <num>
## 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
##         <num>      <num>      <num>      <num>       <num>       <num>
## 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
##          <num>       <num>        <num>        <num>       <num>       <num>
## 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
##         <num>      <num>        <num>       <num>       <num>    <fctr>
## 1:  0.1285394 -0.1891148  0.133558377 -0.02105305  0.24496383 Not Fraud
## 2:  0.1671704  0.1258945 -0.008983099  0.01472417 -0.34247394 Not Fraud
## 3: -0.3276418 -0.1390966 -0.055352794 -0.05975184  1.16068389 Not Fraud
## 4:  0.6473760 -0.2219288  0.062722849  0.06145763  0.14053401 Not Fraud
## 5: -0.2060096  0.5022922  0.219422230  0.21515315 -0.07340321 Not Fraud
## 6: -0.2327938  0.1059148  0.253844225  0.08108026 -0.33855582 Not Fraud
```

```
table(train_data$Class)
```

```
##
## Not Fraud     Fraud
##    227452       394
```

```
print("Test Data Sample:")
```

```
## [1] "Test Data Sample:"
```

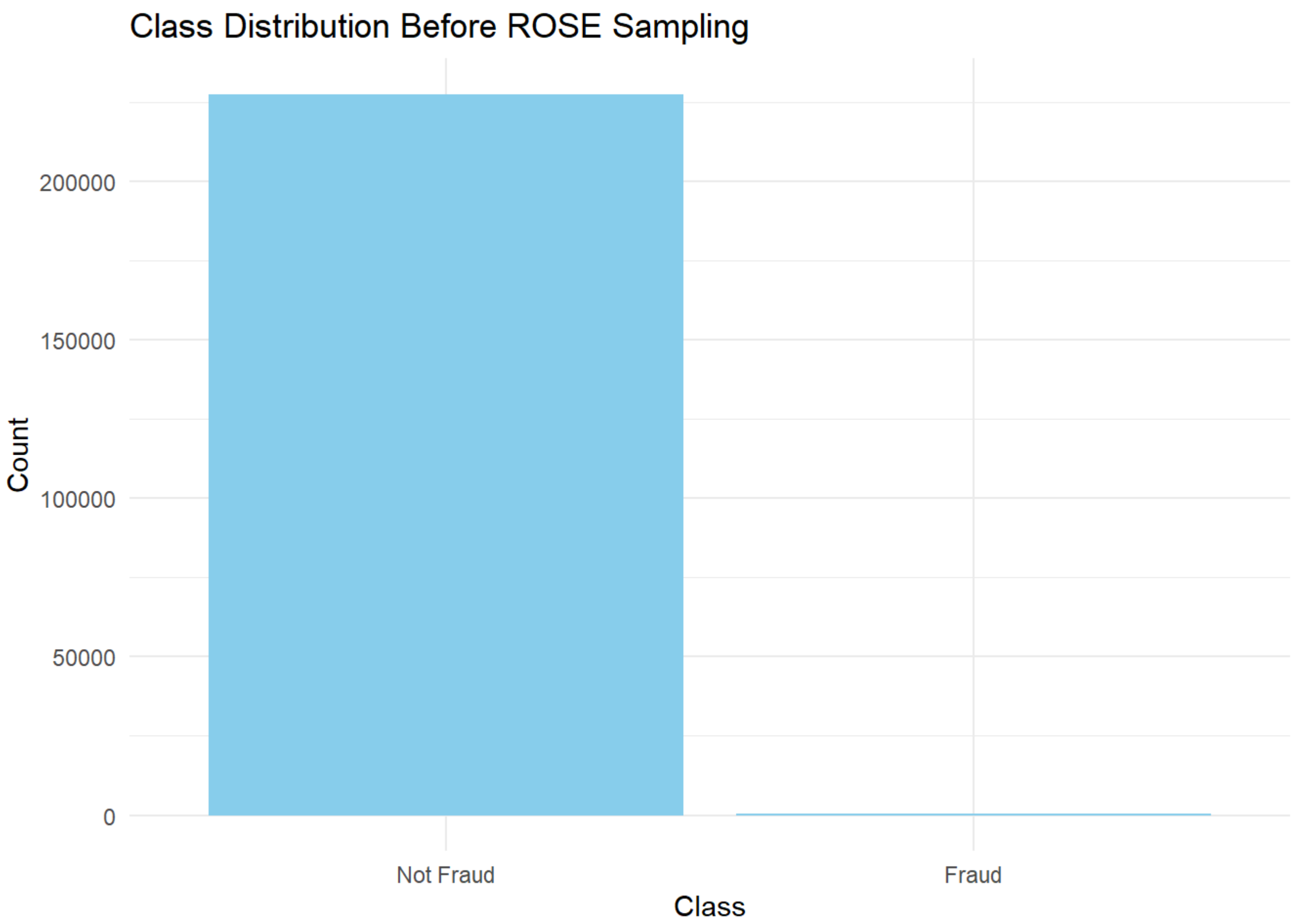
```
head(test_data)
```

```
##            V1          V2         V3         V4         V5          V6
##         <num>       <num>      <num>      <num>      <num>       <num>
## 1:  1.4490438 -1.17633883  0.9138598 -1.3756667 -1.9713832 -0.62915214
## 2:  1.0693736  0.28772213  0.8286127  2.7125204 -0.1783980  0.33754373
## 3:  1.1032154 -0.04029621  1.2673321  1.2890915 -0.7359972  0.28806916
## 4:  0.9624961  0.32846103 -0.1714791  2.1092041  1.1295656  1.69603769
## 5: -1.9465251 -0.04490051 -0.4055701 -1.0130573  2.9419677  2.95505340
## 6: -0.5353878  0.86526781  1.3510763  0.1475755  0.4336802  0.08698294
##             V7          V8         V9        V10         V11        V12
##          <num>       <num>      <num>      <num>       <num>      <num>
## 1: -1.42323560 0.04845589 -1.7204084  1.6266591  1.19964395 -0.6714398
## 2: -0.09671686 0.11598174 -0.2210826  0.4602304 -0.77365693  0.3233872
## 3: -0.58605679 0.18937971  0.7823329 -0.2679751 -0.45031128  0.9367077
## 4:  0.10771161 0.52150216 -1.1913111  0.7243963  1.69032992  0.4067736
## 5: -0.06306315 0.85554631  0.0499669  0.5737425 -0.08125651 -0.2157450
## 6:  0.69303931 0.17974226 -0.2856419 -0.4824745  0.87179958  0.8534474
##            V13         V14        V15         V16          V17         V18
##          <num>       <num>      <num>       <num>        <num>       <num>
## 1: -0.51394715 -0.09504505  0.2309304  0.03196747  0.253414716  0.85434381
## 2: -0.01107589 -0.17848518 -0.6555643 -0.19992517  0.124005415 -0.98049620
## 3:  0.70838041 -0.46864729  0.3545741 -0.24663466 -0.009212378 -0.59591241
## 4: -0.93642130  0.98373942  0.7109108 -0.60223177  0.402484376 -1.73716203
## 5:  0.04416063  0.03389776  1.1907177  0.57884348 -0.975667025  0.04406282
## 6: -0.57182189  0.10225210 -1.5199912 -0.28591250 -0.309633387 -0.40390199
##           V19        V20          V21        V22         V23           V24
##         <num>      <num>        <num>      <num>       <num>         <num>
## 1: -0.2213654 -0.3872265 -0.009301897  0.3138944  0.02774016  0.5005122871
## 2: -0.9829161 -0.1531972 -0.036875532  0.0744124 -0.07140743  0.1047437526
## 3: -0.5756816 -0.1139102 -0.024612006  0.1960020  0.01380165  0.1037583310
## 4: -2.0276123 -0.2693210  0.143997423  0.4024917 -0.04850822 -1.3718662945
## 5:  0.4886029 -0.2167153 -0.579525934 -0.7992290  0.87030022  0.9834214925
## 6: -0.8237430 -0.2832638  0.049525687  0.2065365 -0.18710807  0.0007530143
##           V25        V26         V27         V28     Amount     Class
##         <num>      <num>       <num>       <num>      <num>    <fctr>
## 1: 0.25136736 -0.1294780  0.04284987  0.01625326 -0.3220438 Not Fraud
## 2: 0.54826473  0.1040942  0.02149106  0.02129331 -0.2432816 Not Fraud
## 3: 0.36429754 -0.3822606  0.09280919  0.03705052 -0.3012937 Not Fraud
## 4: 0.39081389  0.1999637  0.01637064 -0.01460533 -0.2169343 Not Fraud
## 5: 0.32120113  0.1496499  0.70751884  0.01459975 -0.3496705 Not Fraud
## 6: 0.09811661 -0.5534710 -0.07830550  0.02542738 -0.3461522 Not Fraud
```

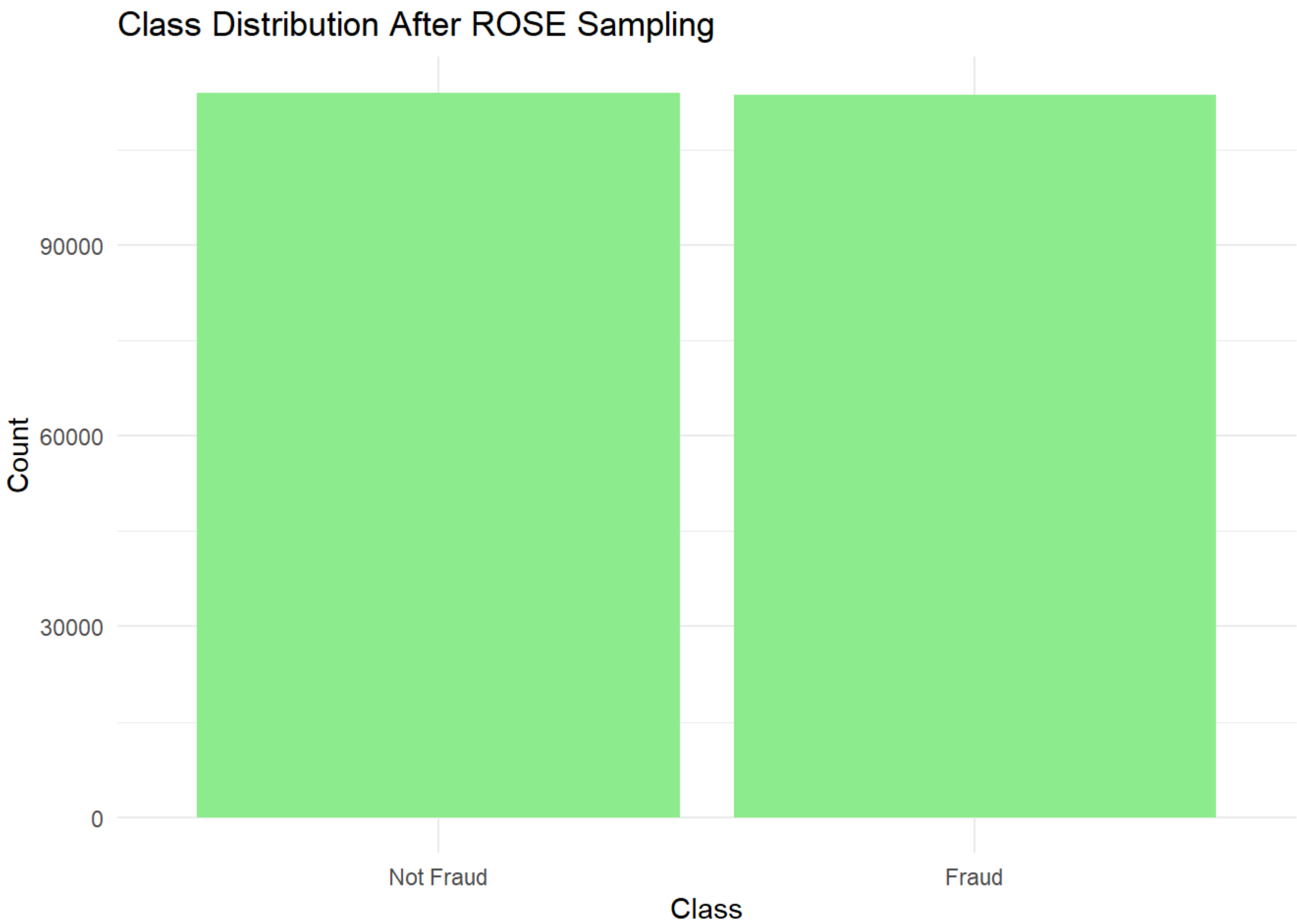
```
table(test_data$Class)
```

```
## 
## Not Fraud     Fraud 
##     56863        98
```

```
# Handle Class Imbalance using ROSE
set.seed(9560)
rose_train_data <- ROSE(Class ~ ., data = train_data)$data

# Plot class distribution before ROSE sampling
ggplot(train_data, aes(x = factor(Class))) +
  geom_bar(fill = "skyblue") +
  ggtitle("Class Distribution Before ROSE Sampling") +
  xlab("Class") +
  ylab("Count") +
  theme_minimal()
```




```
# Plot class distribution after ROSE sampling
ggplot(rose_train_data, aes(x = factor(Class))) +
  geom_bar(fill = "lightgreen") +
  ggtitle("Class Distribution After ROSE Sampling") +
  xlab("Class") +
  ylab("Count") +
  theme_minimal()
```



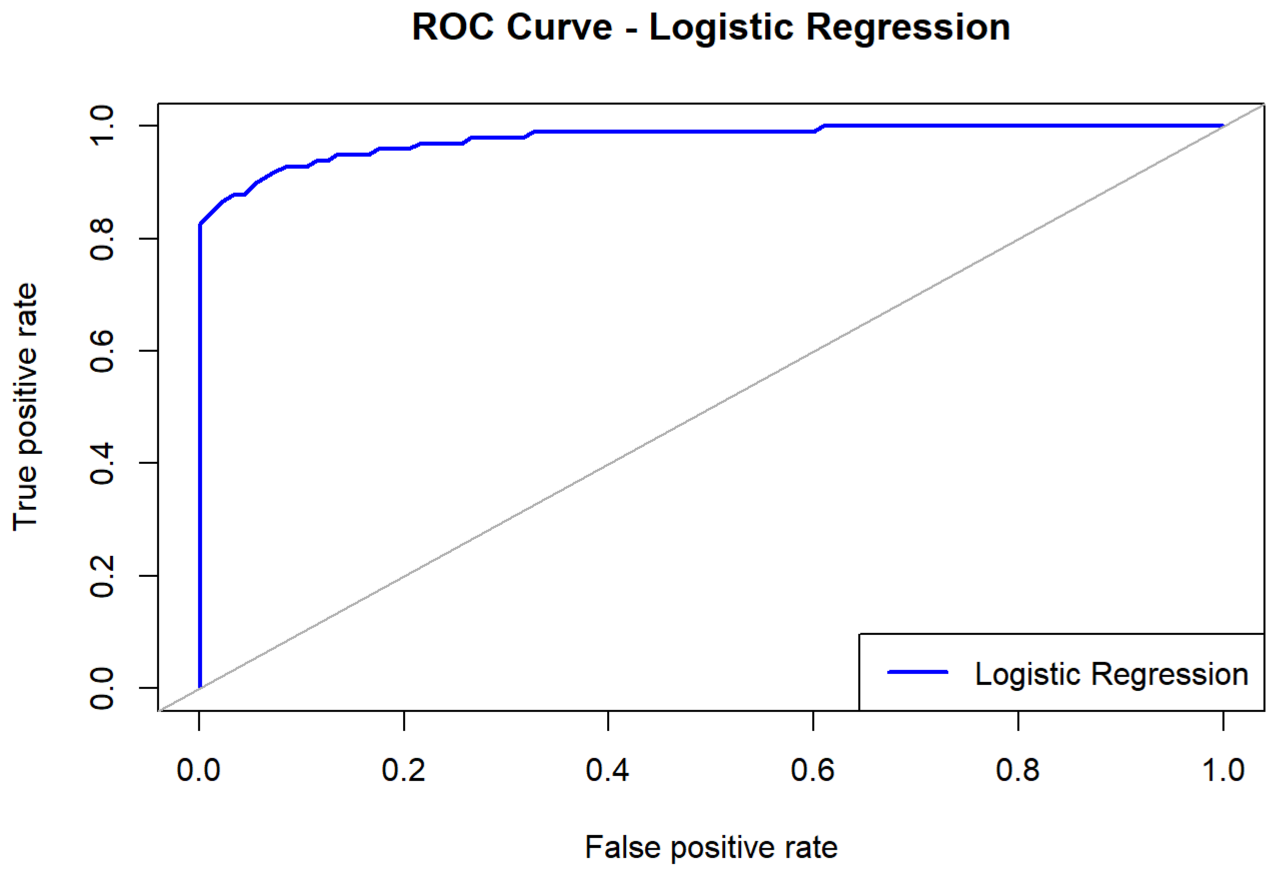

```
# ----------------- LOGISTIC REGRESSION -----------------
logistic_model <- glm(Class ~ ., data = rose_train_data, family = 'binomial')
```

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

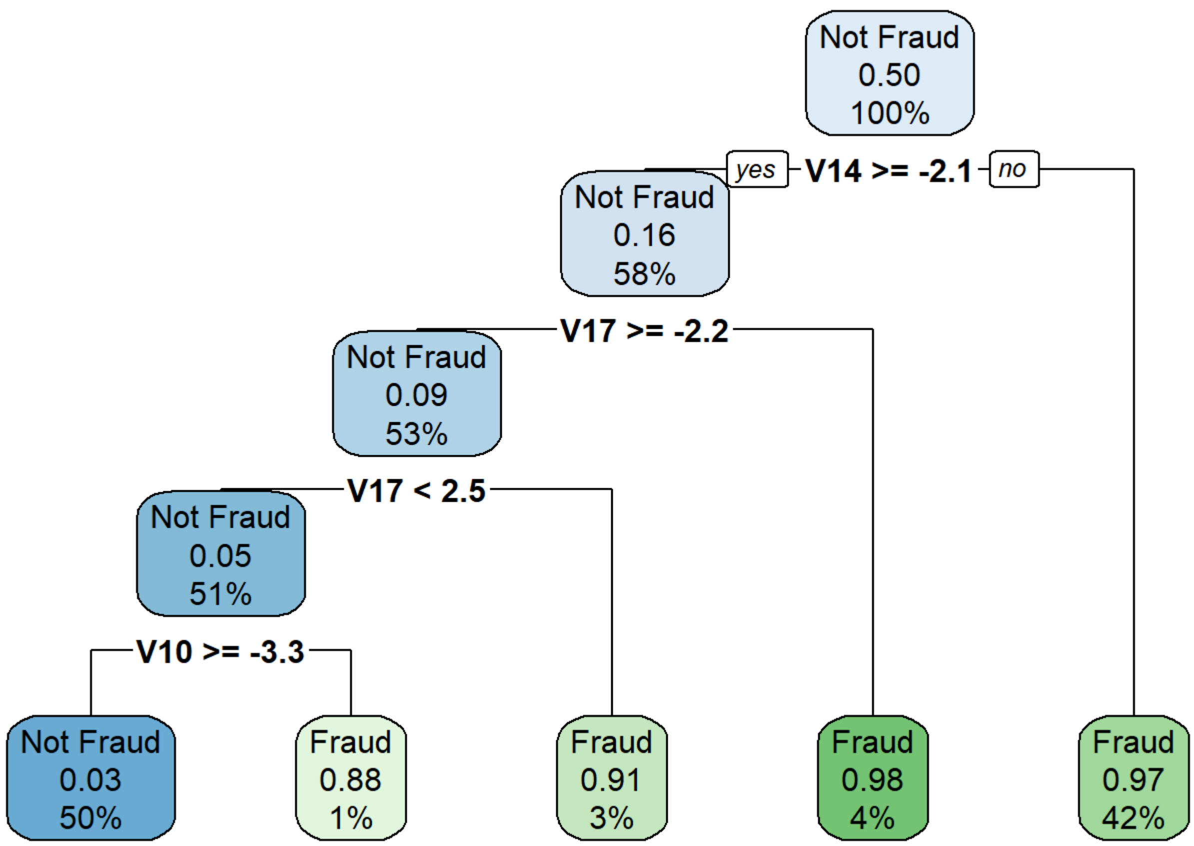
```
logistic_predictions <- predict(logistic_model, test_data, type = 'response')

# ROC Curve for Logistic Regression
roc.curve(test_data$Class, logistic_predictions,
          plotit = TRUE,
          col = "blue",
          main = "ROC Curve - Logistic Regression")
```

```
## Area under the curve (AUC): 0.977
```

```
legend("bottomright", legend = "Logistic Regression", col = "blue", lwd = 2)
```



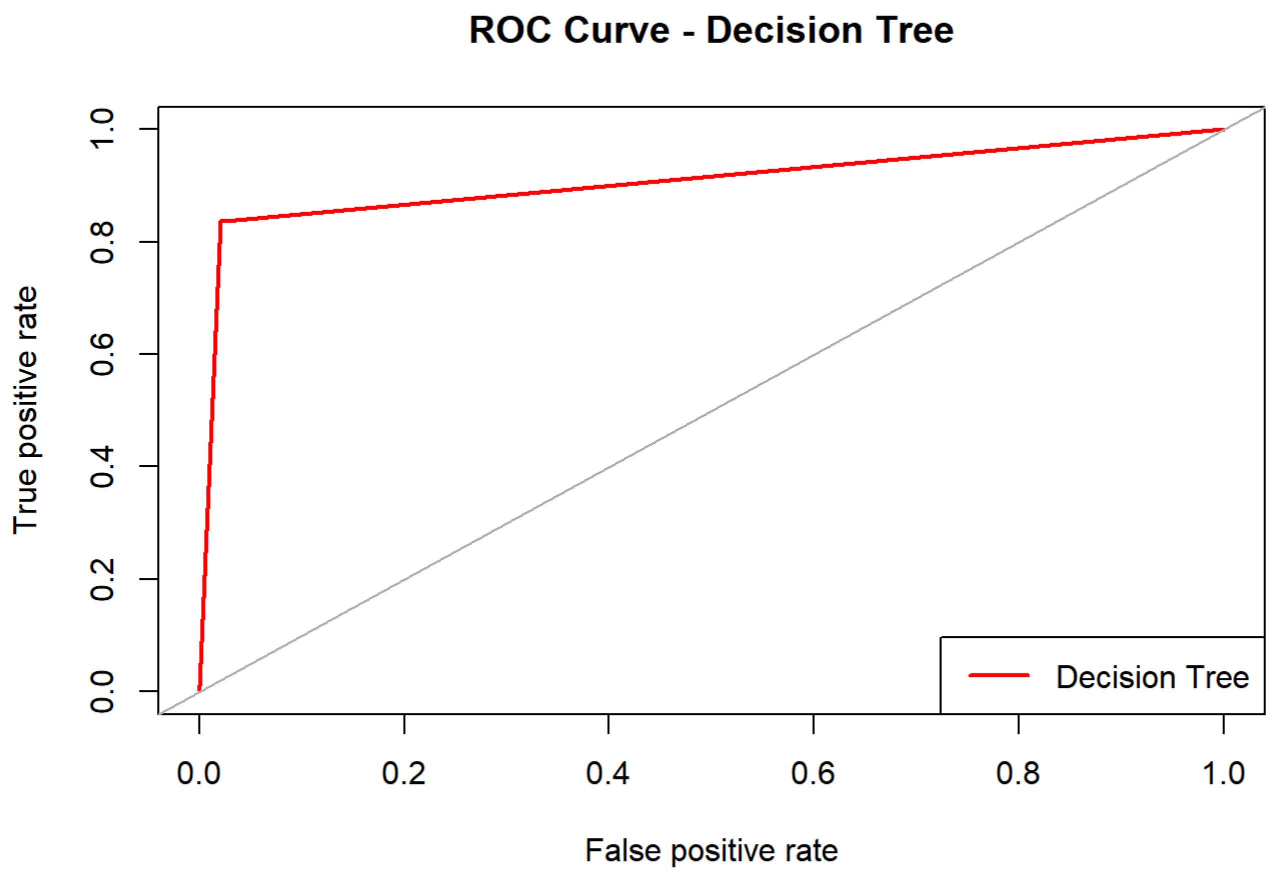

```
# ----------------- DECISION TREE -----------------
decisionTree_model <- rpart(Class ~ ., data = rose_train_data, method = 'class')
rpart.plot(decisionTree_model)
```



```
decisionTree_predictions <- predict(decisionTree_model, test_data, type = 'prob')[,2]

# ROC Curve for Decision Tree
roc.curve(test_data$Class, decisionTree_predictions,
          plotit = TRUE,
          col = "red",
          main = "ROC Curve - Decision Tree")
```

```
## Area under the curve (AUC): 0.908
```

```
legend("bottomright", legend = "Decision Tree", col = "red", lwd = 2)
```



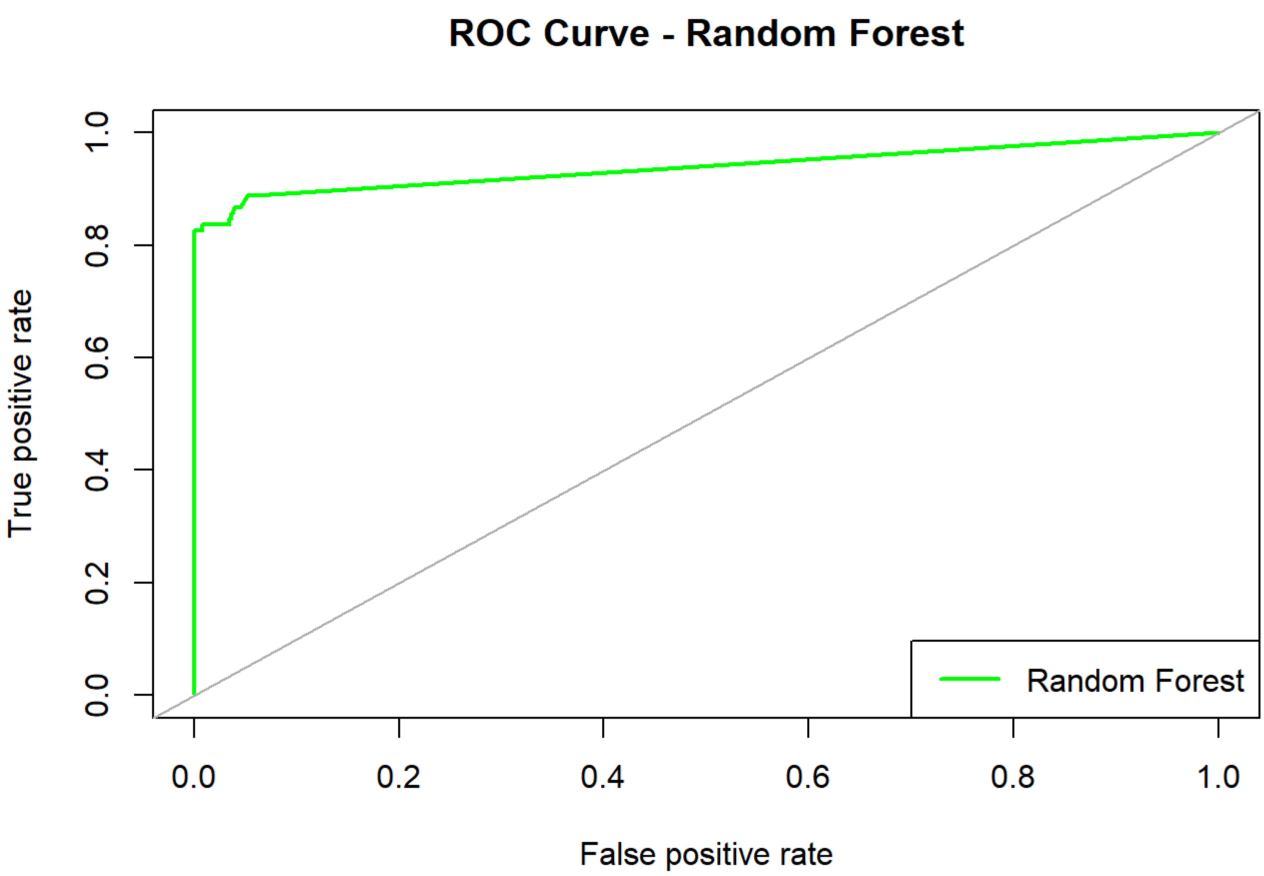

```
# ----------------- RANDOM FOREST -----------------
rf_fit <- ranger(Class ~ .,
                 data = rose_train_data,
                 num.trees = 200,
                 mtry = 5,
                 min.node.size = 10,
                 importance = 'impurity',
                 probability = TRUE)
```

```
## Growing trees.. Progress: 11%. Estimated remaining time: 4 minutes, 41 seconds.
## Growing trees.. Progress: 22%. Estimated remaining time: 4 minutes, 0 seconds.
## Growing trees.. Progress: 40%. Estimated remaining time: 2 minutes, 28 seconds.
## Growing trees.. Progress: 53%. Estimated remaining time: 1 minute, 57 seconds.
## Growing trees.. Progress: 76%. Estimated remaining time: 52 seconds.
```

```
rf_pred <- predict(rf_fit, test_data)$predictions[,2]

roc.curve(test_data$Class, rf_pred,
          plotit = TRUE,
          col = 'green',
          main = "ROC Curve - Random Forest")
```

```
## Area under the curve (AUC): 0.938
```

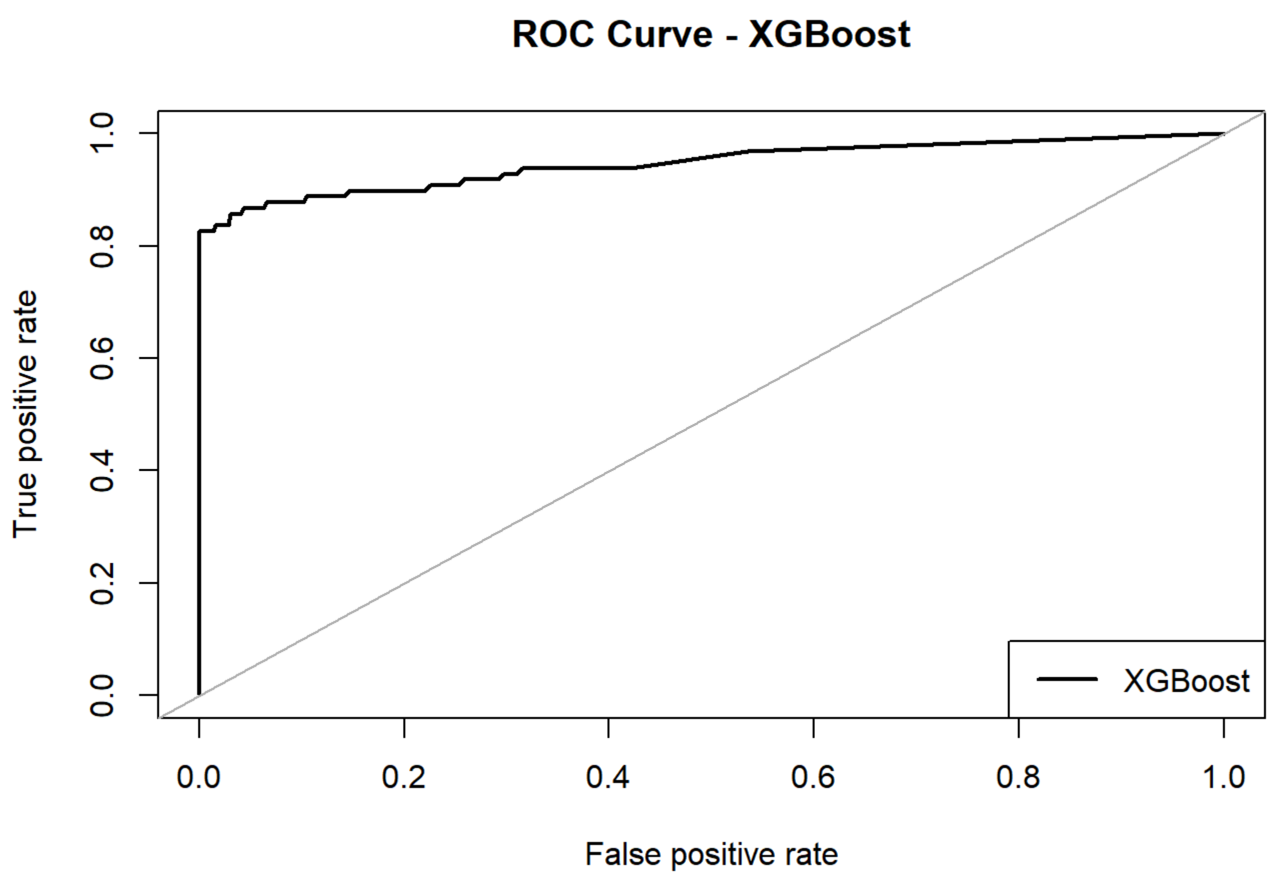
```
legend("bottomright", legend = "Random Forest", col = "green", lwd = 2)
```



```r
# ----------------- HYPERPARAMETER TUNING FOR XGBOOST -----------------
labels <- rose_train_data$Class
y <- ifelse(labels == "Not Fraud", 0, 1)

set.seed(42)
xgb <- xgboost(
  data = data.matrix(rose_train_data[,-30]),
  label = y,
  eta = 0.05,
  gamma = 0.1,
  max_depth = 6,
  nrounds = 150,
  objective = "binary:logistic",
  colsample_bytree = 0.7,
  subsample = 0.7,
  verbose = 0,
  nthread = 2
)

xgb_pred <- predict(xgb, data.matrix(test_data[,-30]))

# ROC Curve for XGBoost
roc.curve(test_data$Class, xgb_pred,
          plotit = TRUE,
          col = "black",
          main = "ROC Curve - XGBoost")
```

```
## Area under the curve (AUC): 0.946
```

```r
legend("bottomright", legend = "XGBoost", col = "black", lwd = 2)
```



```r
# ----------------- EVALUATION FOR LOGISTIC REGRESSION -----------------
logistic_pred_class <- ifelse(logistic_predictions > 0.5, "Fraud", "Not Fraud")
logistic_pred_class <- factor(logistic_pred_class, levels = c("Not Fraud", "Fraud"))
logistic_cm <- confusionMatrix(logistic_pred_class, test_data$Class)
logistic_precision <- logistic_cm$byClass['Precision']
logistic_recall <- logistic_cm$byClass['Recall']
logistic_auc <- roc(test_data$Class, logistic_predictions)$auc
```

```
## Setting levels: control = Not Fraud, case = Fraud
```

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

```r
# ----------------- EVALUATION FOR DECISION TREE -----------------
decisionTree_pred_class <- ifelse(decisionTree_predictions > 0.5, "Fraud", "Not Fraud")
decisionTree_pred_class <- factor(decisionTree_pred_class, levels = c("Not Fraud", "Fraud"))
decisionTree_cm <- confusionMatrix(decisionTree_pred_class, test_data$Class)
decisionTree_precision <- decisionTree_cm$byClass['Precision']
decisionTree_recall <- decisionTree_cm$byClass['Recall']
decisionTree_auc <- roc(test_data$Class, decisionTree_predictions)$auc
```

```
## Setting levels: control = Not Fraud, case = Fraud
## Setting direction: controls < cases
```

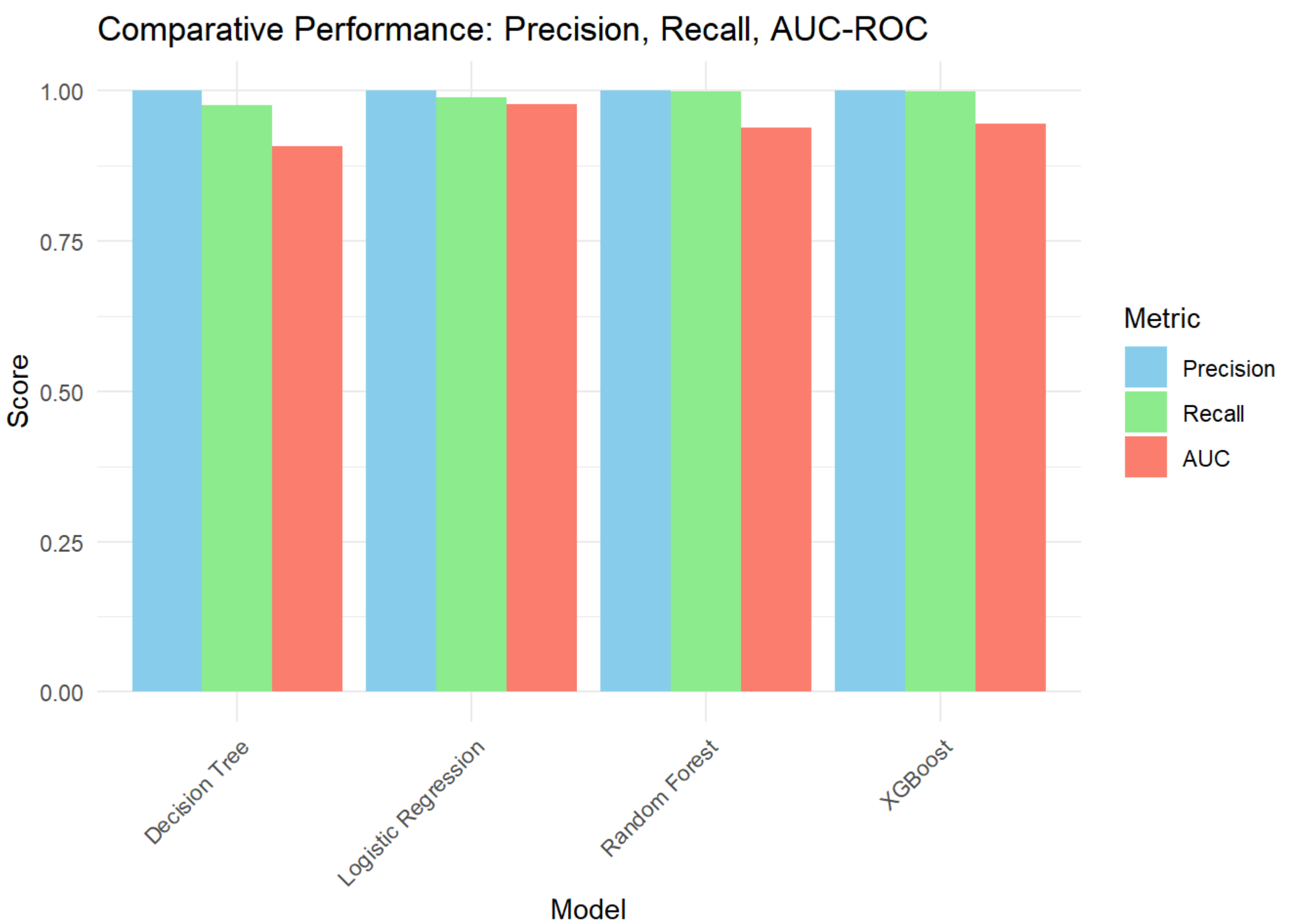
```r
# ----------------- EVALUATION FOR RANDOM FOREST -----------------
rf_pred_class <- ifelse(rf_pred > 0.5, "Fraud", "Not Fraud")
rf_pred_class <- factor(rf_pred_class, levels = c("Not Fraud", "Fraud"))
rf_cm <- confusionMatrix(rf_pred_class, test_data$Class)
rf_precision <- rf_cm$byClass['Precision']
rf_recall <- rf_cm$byClass['Recall']
rf_auc <- roc(test_data$Class, rf_pred)$auc
```

```
## Setting levels: control = Not Fraud, case = Fraud
## Setting direction: controls < cases
```

```r
# ----------------- EVALUATION FOR XGBOOST -----------------
xgb_pred_class <- ifelse(xgb_pred > 0.5, "Fraud", "Not Fraud")
xgb_pred_class <- factor(xgb_pred_class, levels = c("Not Fraud", "Fraud"))
xgb_cm <- confusionMatrix(xgb_pred_class, test_data$Class)
xgb_precision <- xgb_cm$byClass['Precision']
xgb_recall <- xgb_cm$byClass['Recall']
xgb_auc <- roc(test_data$Class, xgb_pred)$auc
```

```
## Setting levels: control = Not Fraud, case = Fraud
## Setting direction: controls < cases
```

```r
# ----------------- COMPARATIVE DATAFRAME -----------------
model_comparison <- data.frame(
  Model = c("Logistic Regression", "Decision Tree", "Random Forest", "XGBoost"),
  Precision = c(logistic_precision, decisionTree_precision, rf_precision, xgb_precision),
  Recall = c(logistic_recall, decisionTree_recall, rf_recall, xgb_recall),
  AUC = c(logistic_auc, decisionTree_auc, rf_auc, xgb_auc)
)

# Reshape the dataframe for ggplot
model_comparison_long <- reshape2::melt(model_comparison, id.vars = "Model", variable.name = "Metric",
value.name = "Value")


# ----------------- PLOTTING COMPARATIVE BAR CHART -----------------
ggplot(model_comparison_long, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Comparative Performance: Precision, Recall, AUC-ROC",
       x = "Model",
       y = "Score") +
  scale_fill_manual(values = c("skyblue", "lightgreen", "salmon")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
# ----------------- THRESHOLD OPTIMIZATION -----------------
test_data$Class <- ifelse(test_data$Class == "Fraud", 1, 0)
pred_obj <- prediction(xgb_pred, test_data$Class)
perf <- performance(pred_obj, "tpr", "fpr")

# Find the best threshold (Closest to perfect TPR/FPR balance)
best_threshold_index <- which.max(perf@y.values[[1]] - perf@x.values[[1]])
best_threshold <- perf@alpha.values[[1]][best_threshold_index]

optimized_predictions <- ifelse(xgb_pred > best_threshold, "Fraud", "Not Fraud")
optimized_predictions <- factor(optimized_predictions, levels = c("Not Fraud", "Fraud"))

print(paste("Optimized Threshold:", best_threshold))
```

```
## [1] "Optimized Threshold: 0.0149605302140117"
```

```r
# Evaluate Model
test_data$Class <- factor(test_data$Class, levels = c(0, 1), labels = c("Not Fraud", "Fraud"))
conf_matrix <- confusionMatrix(optimized_predictions, test_data$Class)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Not Fraud Fraud
##   Not Fraud     55120    15
##   Fraud          1743    83
##
##                Accuracy : 0.9691
##                  95% CI : (0.9677, 0.9705)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0833
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96935
##             Specificity : 0.84694
##          Pos Pred Value : 0.99973
##          Neg Pred Value : 0.04545
##              Prevalence : 0.99828
##          Detection Rate : 0.96768
##    Detection Prevalence : 0.96794
##       Balanced Accuracy : 0.90814
##
##        'Positive' Class : Not Fraud
##
```

```
true_labels <- ifelse(test_data$Class == "Fraud", 1, 0)
pred_obj <- prediction(xgb_pred, true_labels)
perf <- performance(pred_obj, "tpr", "fpr")
thresholds <- perf@alpha.values[[1]]
tpr <- perf@y.values[[1]]
fpr <- perf@x.values[[1]]

plot(thresholds, tpr, type = "l", col = "blue", ylim = c(0, 1),
     xlab = "Threshold", ylab = "Rate", main = "TPR and FPR vs. Threshold")
lines(thresholds, fpr, col = "red")
legend("bottomleft", legend = c("TPR", "FPR"), col = c("blue", "red"), lty = 1)
```