Credit Card Fraud Detection

Aman Sharma

2023-06-10

Project Details:

This is a Credit Card Fraud Detection project for **Codeclause**. The dataset available on *Kaggle* has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

Loading Packages

```
library(tidymodels)
library(corrplot)
library(caret)
library(rpart)
library(rpart.plot)
library(ggplot2)
library(pROC)
library(neuralnet)
```

Importing Data and exploratory analysis

'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. We have 492 frauds out of 284,807 transactions.

```
data<- read.csv("C:/Users/91835/Desktop/CodeClause_data/creditcard.csv")
data<-na.omit(data)
head(data)
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 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
## 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)
```

```
##       Time             V1                  V2                  V3
##  Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256
##  1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904
##  Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799
##  Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000
##  3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272
##  Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826
##        V4                 V5                  V6                  V7
##  Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605   Min.   :-43.5572
##  1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.5541
##  Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.0401
##  Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.0000
##  3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.5704
##  Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.5895
##        V8                  V9                 V10                 V11
##  Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747
##  1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249
##  Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276
##  Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959
##  Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891
##       V12                 V13                 V14                 V15
##  Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894
##  1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288
##  Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807
##  Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000
##  3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882
##  Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774
```

```
##       V16                V17                V18          
##  Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746  
##  1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850  
##  Median :  0.06641   Median : -0.06568   Median :-0.003636  
##  Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000  
##  3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807  
##  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
```

```
dim(data)
```

```
## [1] 284807     31
```

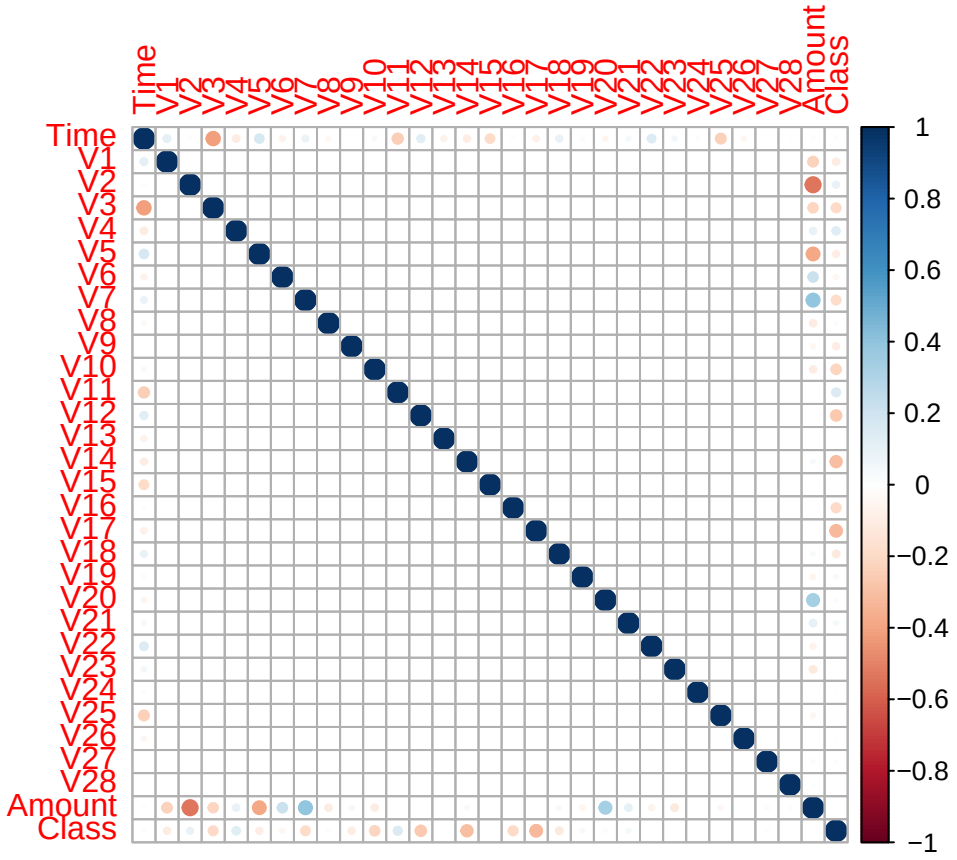
```
data %>%
  filter(Class==1) %>%
  count()
```

```
##     n
## 1 492
```

Observing Correlation

Visible correlation can be observed between V5, V6, V7 and Amount.

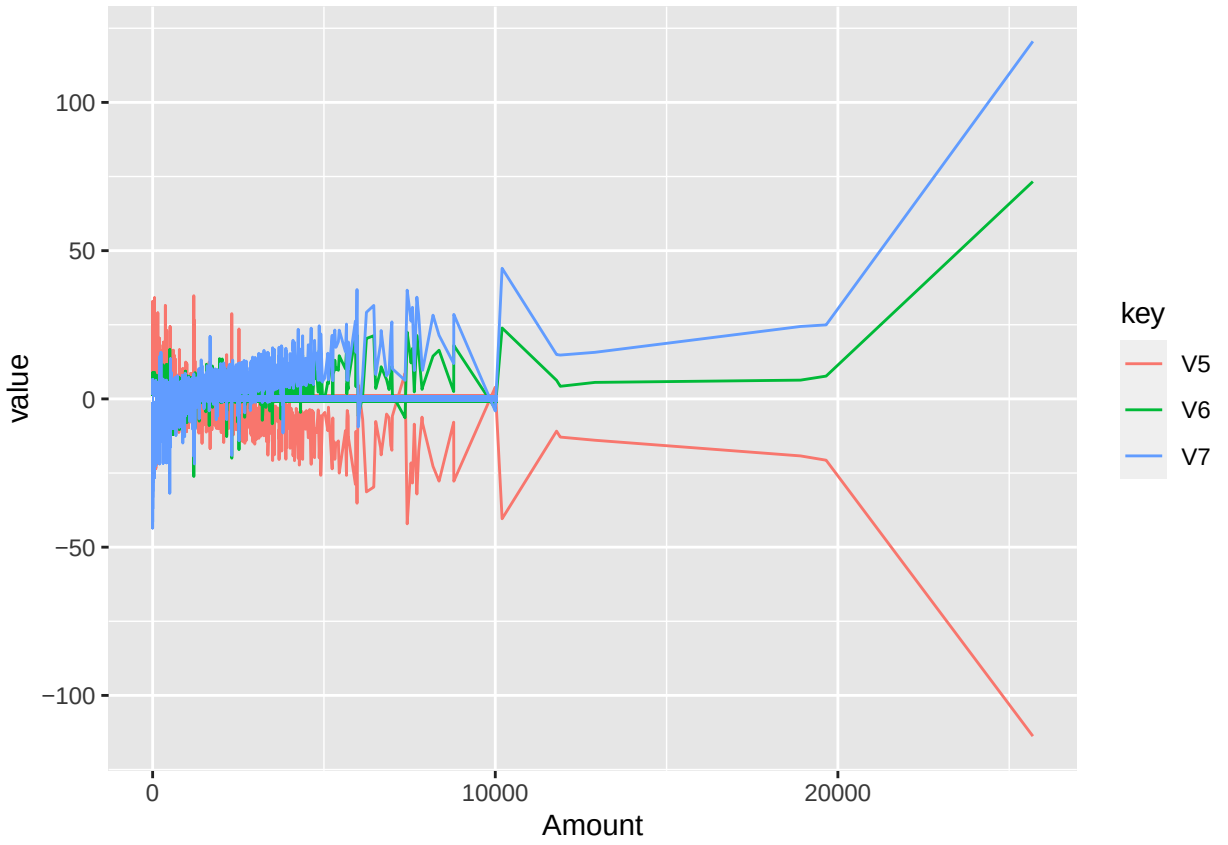
```
new_data<- cor(data)
corrplot(new_data)
```



Plotting them together.

```
df<-data %>%
  select(Amount, V5, V6, V7)
dfframe<-data.frame(df)
dfplot <- dfframe %>% gather(key, value, -Amount)

ggplot(dfplot, mapping = aes(x = Amount, y = value, color = key) ) + geom_line()
```



Modelling

Splitting the data to train and test sets. In addition to use of different models, we have also plotted the ROC curve to measure the model accuracy. The ROC-Curves can be compared to choose the best model and subsequently using it to predict new data.

```
set.seed(1234)
var_split <- initial_split(data, prop = .8)
train_data<- training(var_split)
test_data<- testing(var_split)
```

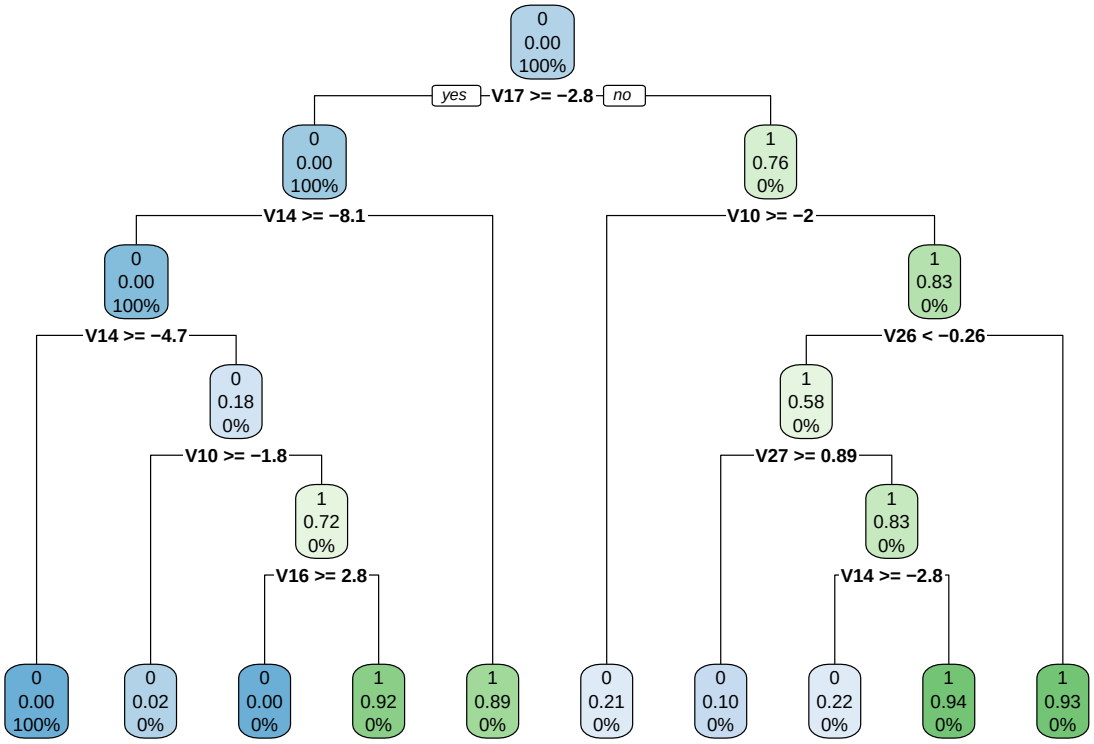
Decision Tree model

Building a Decision Tree model and using ROC curve to determine accuracy.

```
decisionTree_model <- rpart(Class ~ . ,train_data, method = "class")
predicted_decision <- predict(decisionTree_model, test_data, type = "class")
tail(predicted_decision)
```

```
## 284770 284774 284784 284787 284801 284807
##      0      0      0      0      0      0
## Levels: 0 1
```

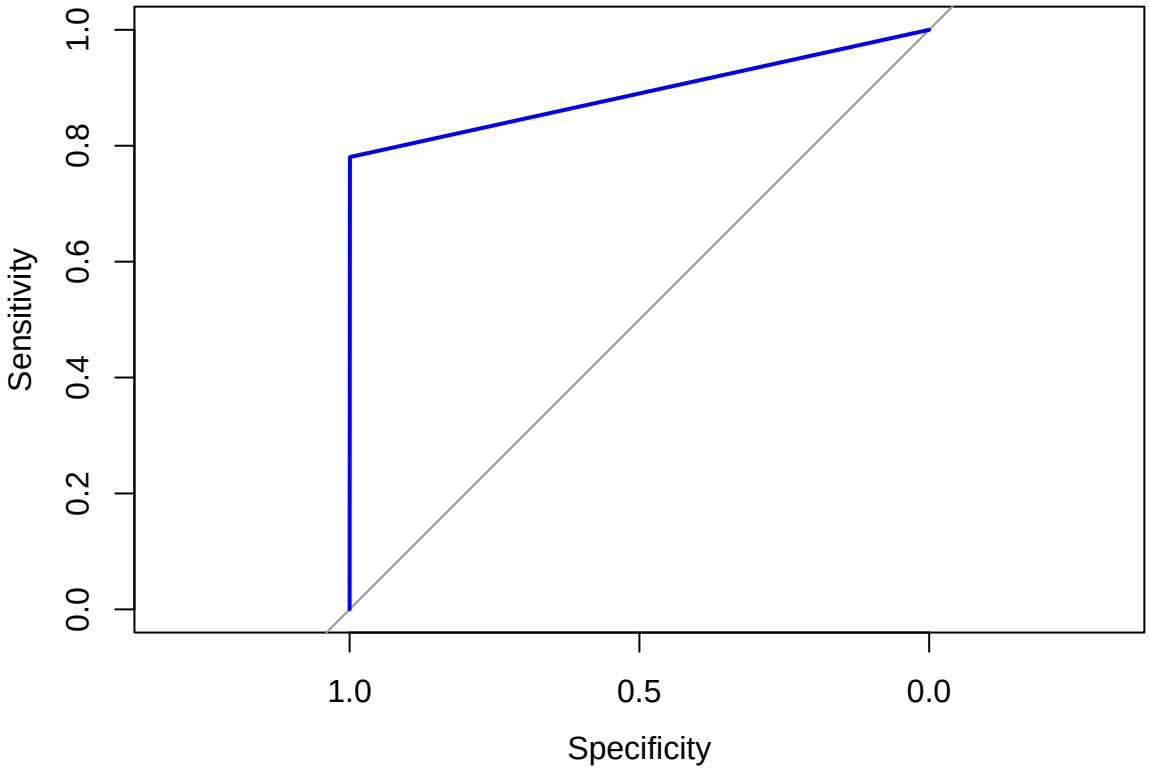
```
rpart.plot(decisionTree_model)
```




```
#ROC
model= train(Class~., data=train_data, method="rpart")
var2<-scale(test_data)
var_predict<- predict(model, newdata = var2)
auc.gbm = roc(test_data$Class, var_predict, plot = TRUE, col = "blue")
```

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

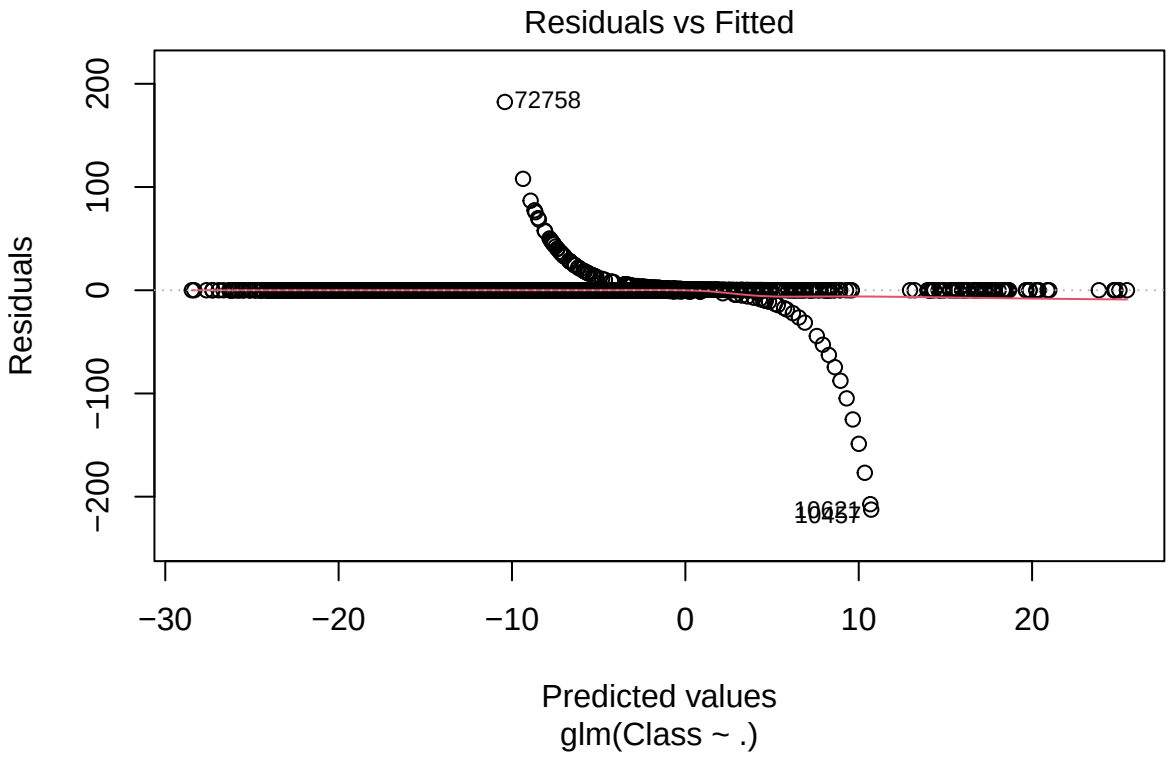
```
## Setting direction: controls < cases
```

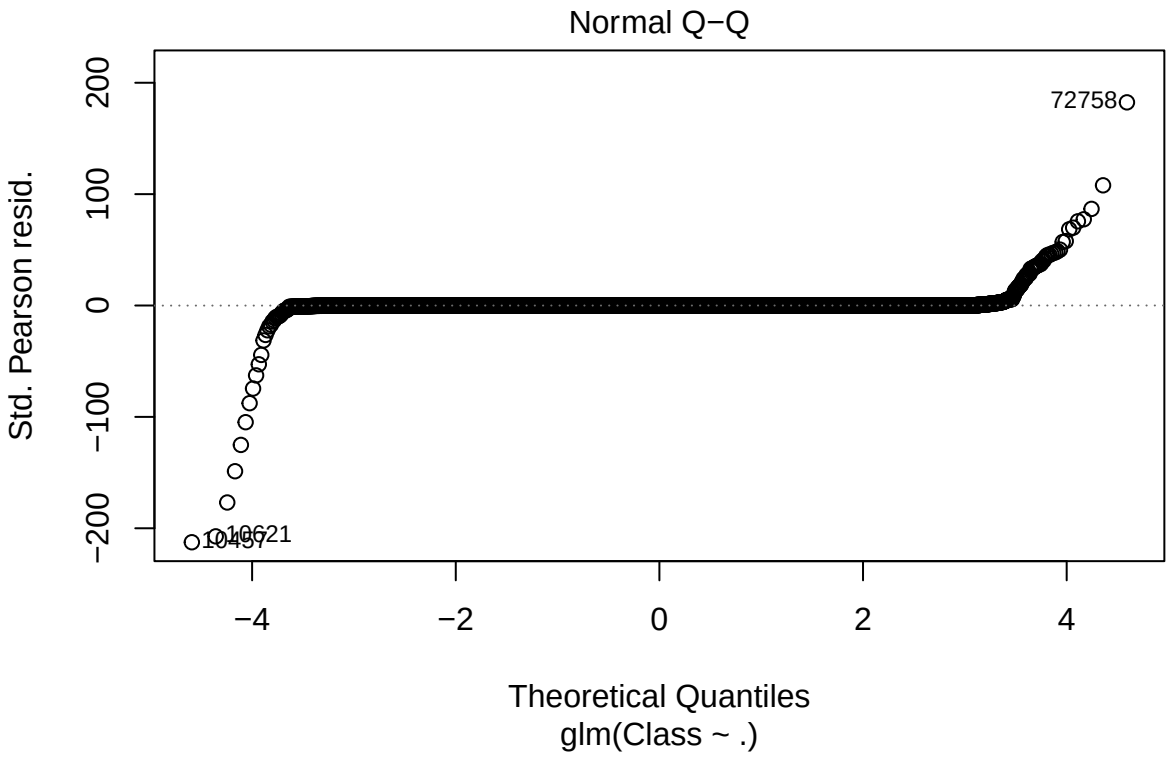


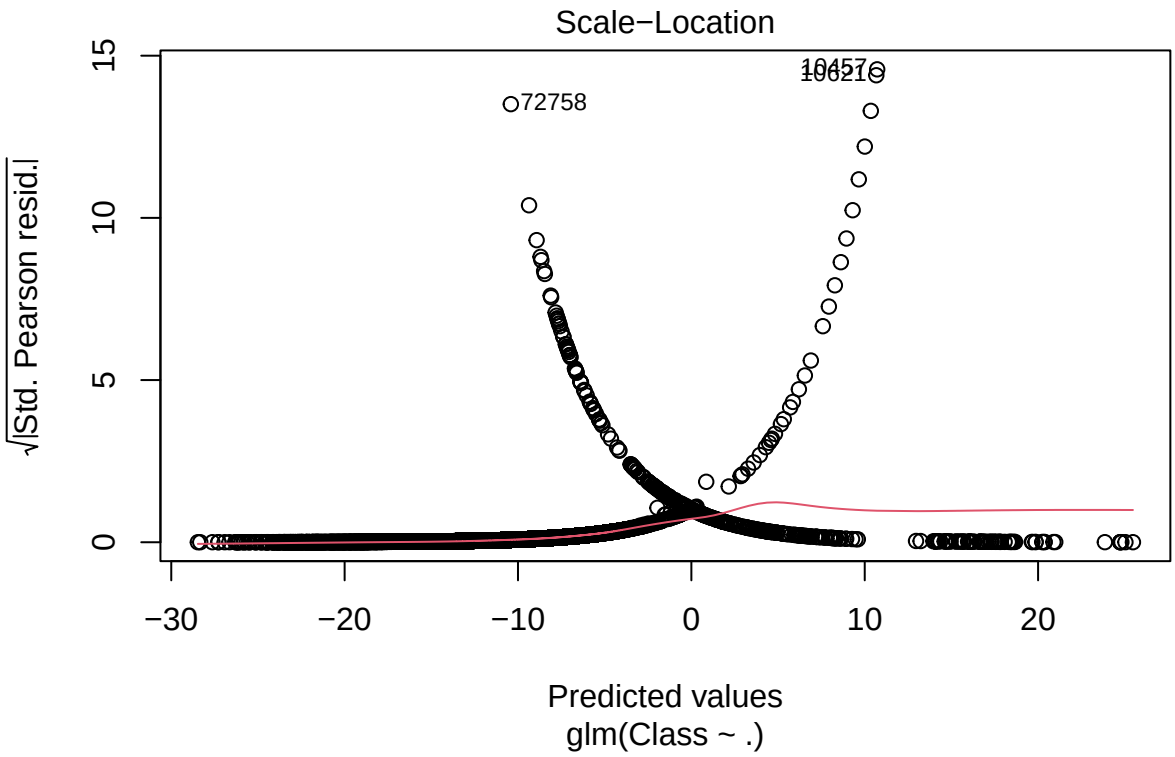
Logistic Model

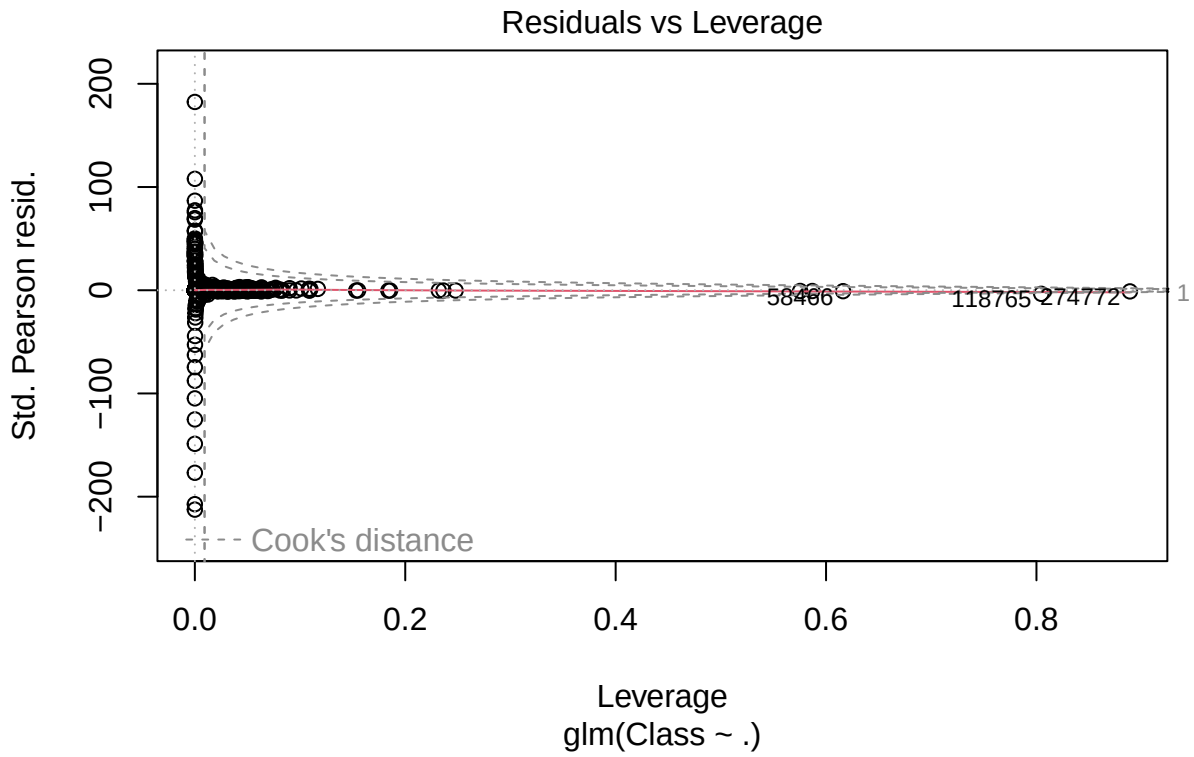
Building Logistic Model with ROC curve.

```
Logistic_Model=glm(Class~.,train_data,family=binomial())
plot(Logistic_Model)
```


Residuals vs Fitted
72758
10457
Residuals
200
100
0
−100
−200
−30
−20
−10
0
10
20
Predicted values
glm(Class ~ .)


Normal Q−Q
Std. Pearson resid.
200
100
0
−100
−200
72758
10457
10621
−4
−2
0
2
4
Theoretical Quantiles
glm(Class ~ .)

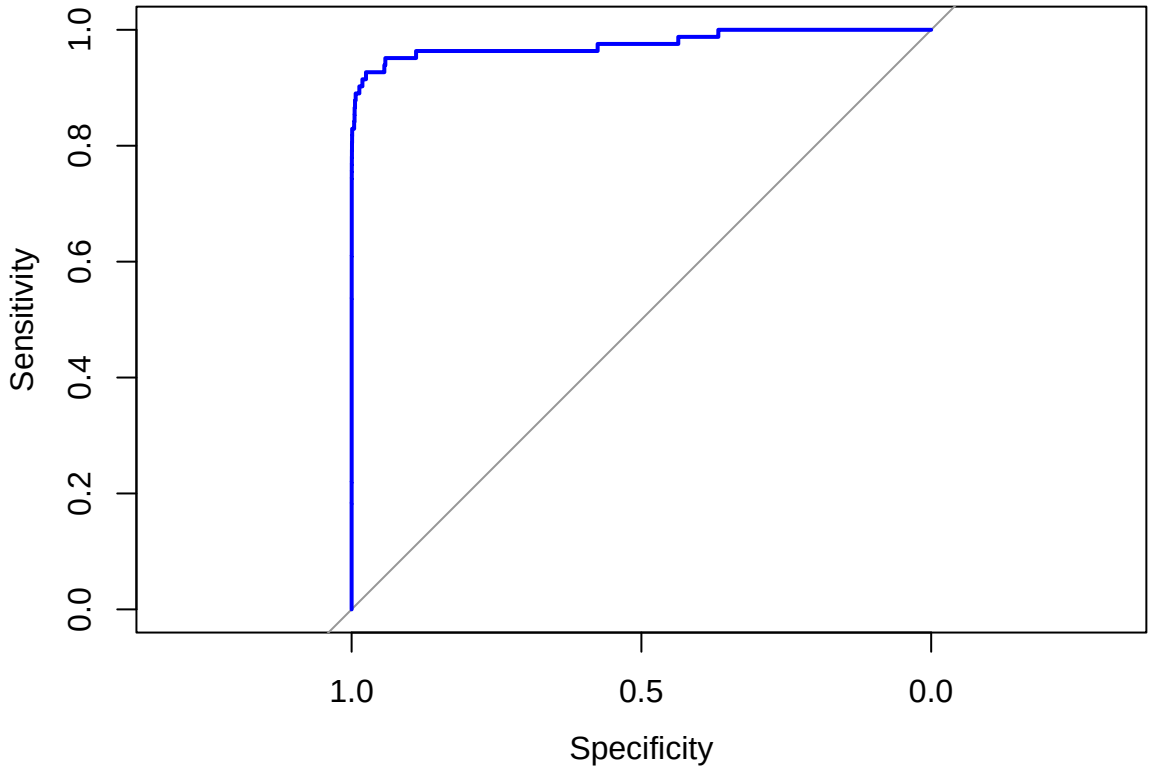

Scale−Location
√|Std. Pearson resid.|
0
5
10
15
72758
10457
10621
−30
−20
−10
0
10
20
Predicted values
glm(Class ~ .)




```
lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
#ROC TEST
auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
```

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

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



Neural Net Model

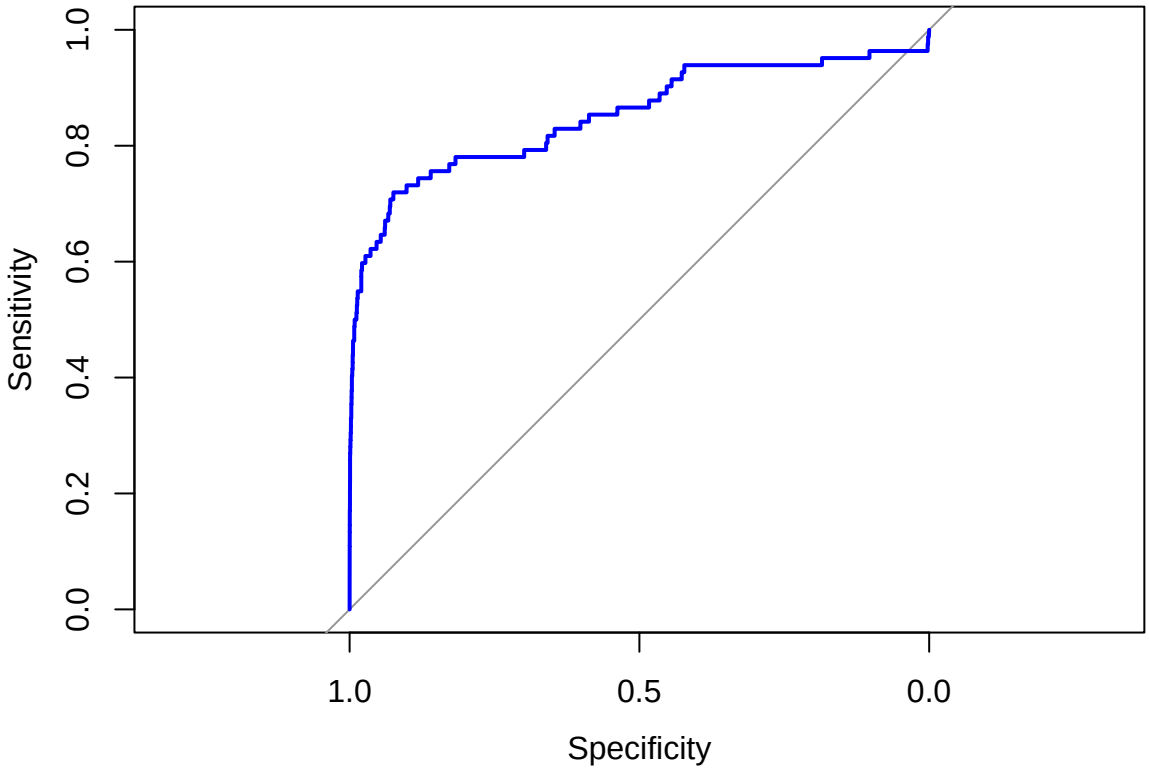
Building Neural Net Model with ROC curve.

```
neural_model<- neuralnet(Class ~ ., data=train_data)
# plot(neural_model)

neural_predict<- predict(neural_model, newdata= scale(test_data))
auc.gbm<- roc(test_data$Class, neural_predict, plot=TRUE, col="blue")
```

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

```
## Setting direction: controls > cases
```


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity