Credit Card Fraud Detection

SAGNIK SAMANTA

Codsoft Task-3

We have dataset on Credit Card Transactions.Here our main purpose is to build a Machine Learning model which can detect fraudulent credit transactions. Description is given at the beginning of our analysis.Here the response variable is of dichotomous type i.e. it can take values 0 and 1.We have performed logistic regression.At the beginning dataset is divided into two parts,one is training and the other is testing.Train the classification algorithm using training dataset and predict the result using test dataset.For the evaluation of model's performance ROC curve method is used which says that AUC(The area under the curve)is 0.9687 ~ 1,that means the model performed here fits the data well.

Importing Dataset

```
Creditcard=read.csv("C:/Users/SAGNIK SAMANTA/OneDrive/Desktop/Datasets/CreditCard.csv",sep=",",header = TRUE)
```

Print First 6 rows of the Dataset

```
head(Creditcard)
```

```
##   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  0.87773676 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         V2
4
```

```
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.0669280
8
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.3398464
8
## 3 -2.26185709  0.52497973  0.247998153  0.771679402  0.90941226 -0.6892809
6
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.1755753
3
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.1412669
8
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.3714265
8
##          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
```

Check the Dimension of the Dataset

```
dim(Creditcard)
```

```
[1] 284807     31
```

Therefore the Dataset(CreditCard) has 284807 Rows and 31 Columns.

```
str(Creditcard)
```

```
'data.frame':    284807 obs. of  31 variables:
$ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
$ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
$ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
$ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
$ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
$ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
$ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
$ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
$ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
$ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
$ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
$ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
$ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
$ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
$ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
$ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
$ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
$ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
$ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
$ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
$ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
```

```
$ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
$ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
$ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
$ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
$ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
$ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
$ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
$ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
$ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
$ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
Creditcard$Class=as.factor(Creditcard$Class)
```

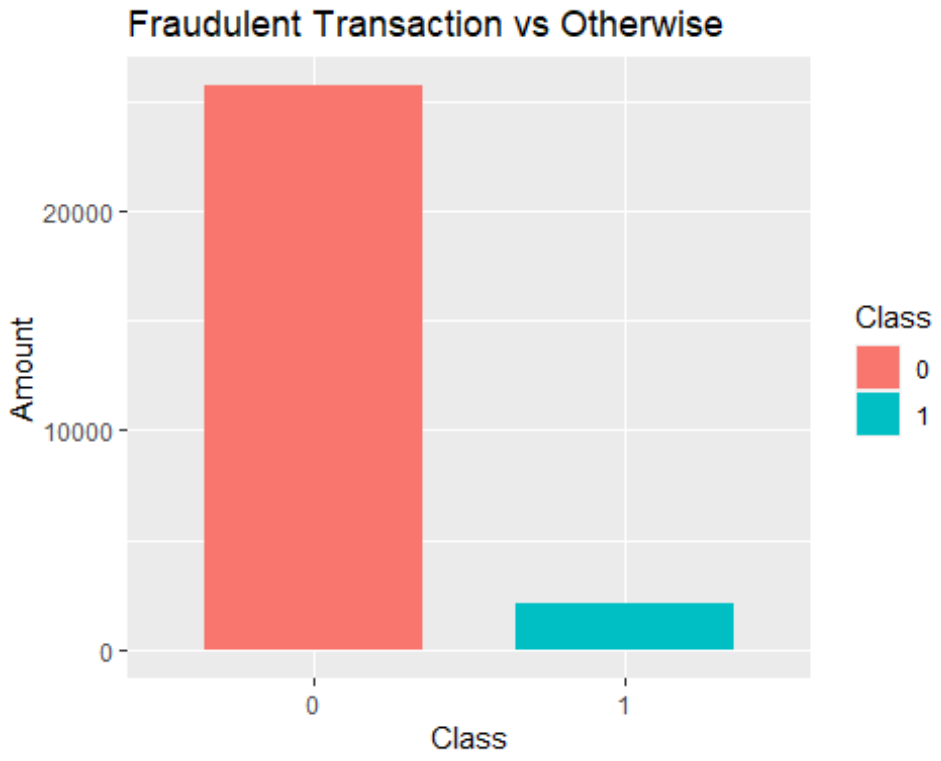
```
Summary of dataset in package
summary(Creditcard)
```

```
##       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.   :-43.55
72
##  1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.55
41
##  Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.04
01
##  Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.00
00
##  3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.57
04
##  Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.58
95
##        V8                  V9                  V10                 V11
##  Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79
747
##  1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76
249
##  Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03
276
##  Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00
000
##  3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73
959
##  Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01
891
##       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   0:284315
##  1st Qu.: -0.05296   1st Qu.:    5.60   1:   492
##  Median :  0.01124   Median :   22.00
##  Mean   :  0.00000   Mean   :   88.35
##  3rd Qu.:  0.07828   3rd Qu.:   77.17
##  Max.   : 33.84781   Max.   :25691.16
```

```
Percentage Calculation of Fraudulent Transaction or Otherwise
attach(Creditcard)
my_table=Class

Percentage of Fraudulent Transaction
(sum(my_table==1)/(sum(my_table==1)+sum(my_table==0)))*100

[1] 0.1727486
```

```
Percentage of Otherwise
(sum(my_table==0)/(sum(my_table==1)+sum(my_table==0)))*100

[1] 99.82725

Class == 1 represents fradulent transactions
Class == 0 represents Otherwise
As shown in the above result, out of 284807 transactions 492   transactions a
re fraudulent credit card transactions.
99.82725% are Non-fraudulent and 0.1727486% are fraudulent transactions.

Data Visualizations Using Barplot
install.packages("ggplot2")
library(ggplot2)

ggplot(Creditcard, aes(y=Amount, x=Class,fill=Class)) +
  geom_bar(position="dodge", stat="identity",width=0.7)+
  ggtitle("Fraudulent Transaction vs Otherwise") +
  xlab("Class") +
  ylab("Amount")
```




```
For Logistic regression
install.packages("caTools")
library(caTools)

Splitting dataset
split=sample.split(Creditcard, SplitRatio = 0.8)
```

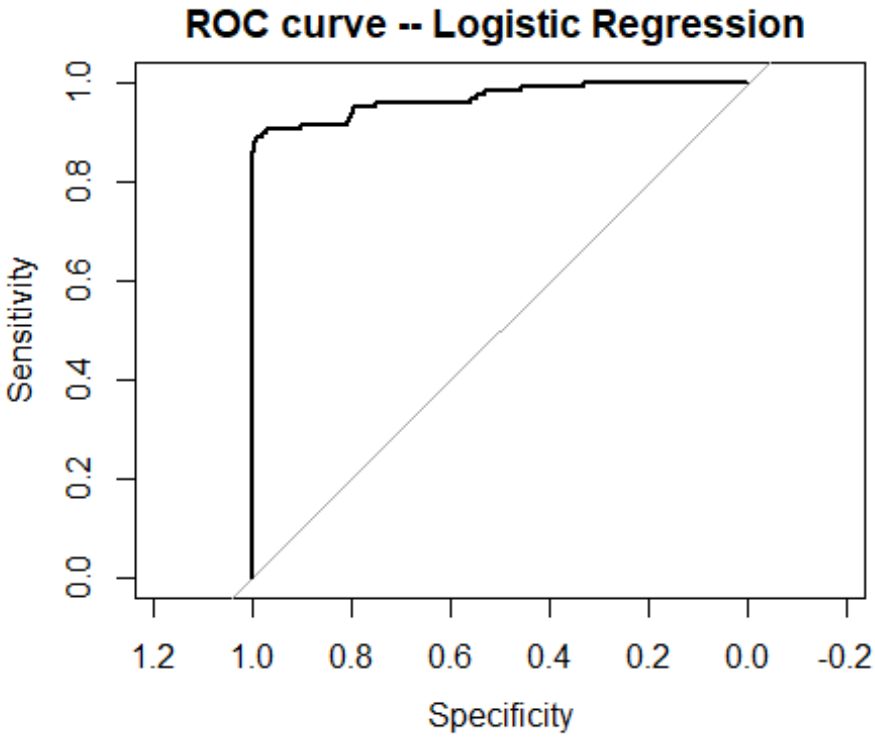
```r
train_reg=subset(Creditcard, split == "TRUE")
test_reg=subset(Creditcard, split == "FALSE")

Training model
logistic_model = glm(Class ~ Time+Amount+V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V
12+V13+V14+V15+V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28,data = tra
in_reg,family = "binomial"(link = logit))

Summary
summary(logistic_model)

##
## Call:
## glm(formula = Class ~ Time + Amount + V1 + V2 + V3 + V4 + V5 +
##     V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 +
##     V17 + V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 +
##     V27 + V28, family = binomial(link = logit), data = train_reg)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.598e+00  2.920e-01 -29.446  < 2e-16 ***
## Time        -2.533e-06  2.568e-06  -0.986 0.323896
## Amount       9.635e-04  3.677e-04   2.620 0.008784 **
## V1           4.414e-02  4.667e-02   0.946 0.344259
## V2           5.842e-02  6.466e-02   0.904 0.366240
## V3          -2.299e-02  5.901e-02  -0.390 0.696872
## V4           7.111e-01  8.814e-02   8.069 7.11e-16 ***
## V5           1.637e-01  7.553e-02   2.167 0.030224 *
## V6          -1.166e-01  8.120e-02  -1.436 0.151084
## V7          -5.964e-02  7.385e-02  -0.808 0.419348
## V8          -1.781e-01  3.318e-02  -5.366 8.04e-08 ***
## V9          -1.892e-01  1.310e-01  -1.444 0.148630
## V10         -6.979e-01  1.120e-01  -6.232 4.60e-10 ***
## V11         -5.663e-02  9.147e-02  -0.619 0.535868
## V12          1.373e-01  1.038e-01   1.323 0.185914
## V13         -4.145e-01  9.414e-02  -4.403 1.07e-05 ***
## V14         -5.837e-01  7.308e-02  -7.987 1.38e-15 ***
## V15         -7.407e-02  9.678e-02  -0.765 0.444077
## V16         -2.534e-01  1.457e-01  -1.739 0.082008 .
## V17         -2.887e-02  8.329e-02  -0.347 0.728920
## V18          1.789e-02  1.510e-01   0.118 0.905692
## V19          2.386e-02  1.112e-01   0.215 0.830072
## V20         -4.102e-01  8.698e-02  -4.717 2.40e-06 ***
## V21          3.988e-01  6.825e-02   5.843 5.13e-09 ***
## V22          7.545e-01  1.545e-01   4.884 1.04e-06 ***
## V23         -9.598e-02  6.332e-02  -1.516 0.129540
## V24          2.527e-01  1.662e-01   1.520 0.128441
## V25         -6.897e-02  1.456e-01  -0.474 0.635708
## V26         -1.677e-02  2.228e-01  -0.075 0.939982
```

```
## V27         -8.670e-01  1.227e-01  -7.065 1.61e-12 ***
## V28         -4.418e-01  1.167e-01  -3.786 0.000153 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5557.4  on 220495  degrees of freedom
## Residual deviance: 1748.4  on 220465  degrees of freedom
## AIC: 1810.4
##
## Number of Fisher Scoring iterations: 12
```

```
Predict test data based on model
predict_reg=predict(logistic_model,test_reg, type = "response")

Transaction=c()
for(i in 1:length(predict_reg)){
  if(predict_reg[i] > 0.9){
    Transaction[i] = "Fraudulent Transaction"
  } else {
    Transaction[i] = "Non-Fraudulent Transaction"
  }
}

Final_data=cbind(Class=test_reg$Class,Predicted=Transaction)
Final_data=as.data.frame(Final_data)
View(Final_data)

Checking The Model Performance
ROC-curve using pROC library
install.packages("pROC")
library(pROC)

roc_score = roc(test_reg$Class, predict_reg)   ## AUC score

roc_score
```

```
Call:
roc.default(response = test_reg$Class, predictor = predict_reg)
Data: predict_reg in 64196 controls (test_reg$Class 0) < 115 cases (test_reg$
Class 1).
Area under the curve: 0.9671
```

```
plot(roc_score, main="ROC curve -- Logistic Regression ")
```


ROC curve -- Logistic Regression
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Specificity