Assignment 4 Data Analytics Project

Akash Chauhan

12/11/2021

##Business Problem The business problem that we are trying to resolve in this project is credit card fraud detection. It is important for credit card companies to identify any fraudulent transaction made on an account so that the customers are charged only for the items or services they actually purchased.

## Dataset origin

To do the analysis, we are using a data set that contains various credit card transactions made by various card holders in European countries in the month September of the year 2013, that contains a mix of fraud as well as non-fraudulent transactions and are marked accordingly. The data set has been collected and analysed during a research collaboration of World line and the Machine Learning Group (<http://mlg.ulb.ac.be>) of ULB (University Libre de Bruxelles) on big data mining and fraud detection.

## importing Dataset

library(caret)

## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.2

## Loading required package: lattice

library()  
library(data.table)

## Warning: package 'data.table' was built under R version 4.1.2

creditcard\_data <- fread('/Users/Sahil Motwani/Downloads/creditcard.csv')

## Data exploration

nrow(creditcard\_data) #shows number of rows in data

## [1] 284807

dim(creditcard\_data) #shows number of rows and columns in dataset

## [1] 284807 31

head(creditcard\_data, 5) #display top 5 rows

## 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  
## 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  
## V13 V14 V15 V16 V17 V18  
## 1: -0.9913898 -0.3111694 1.4681770 -0.4704005 0.2079712 0.02579058  
## 2: 0.4890950 -0.1437723 0.6355581 0.4639170 -0.1148047 -0.18336127  
## 3: 0.7172927 -0.1659459 2.3458649 -2.8900832 1.1099694 -0.12135931  
## 4: 0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.6840928 1.96577500  
## 5: 1.3458516 -1.1196698 0.1751211 -0.4514492 -0.2370332 -0.03819479  
## V19 V20 V21 V22 V23 V24  
## 1: 0.4039930 0.25141210 -0.018306778 0.277837576 -0.1104739 0.06692807  
## 2: -0.1457830 -0.06908314 -0.225775248 -0.638671953 0.1012880 -0.33984648  
## 3: -2.2618571 0.52497973 0.247998153 0.771679402 0.9094123 -0.68928096  
## 4: -1.2326220 -0.20803778 -0.108300452 0.005273597 -0.1903205 -1.17557533  
## 5: 0.8034869 0.40854236 -0.009430697 0.798278495 -0.1374581 0.14126698  
## 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

tail(creditcard\_data, 5) #display end 5 rows

## Time V1 V2 V3 V4 V5 V6  
## 1: 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278 -2.6068373  
## 2: 172787 -0.7327887 -0.05508049 2.0350297 -0.7385886 0.86822940 1.0584153  
## 3: 172788 1.9195650 -0.30125385 -3.2496398 -0.5578281 2.63051512 3.0312601  
## 4: 172788 -0.2404400 0.53048251 0.7025102 0.6897992 -0.37796113 0.6237077  
## 5: 172792 -0.5334125 -0.18973334 0.7033374 -0.5062712 -0.01254568 -0.6496167  
## V7 V8 V9 V10 V11 V12 V13  
## 1: -4.9182154 7.3053340 1.9144283 4.3561704 -1.5931053 2.71194079 -0.6892556  
## 2: 0.0243297 0.2948687 0.5848000 -0.9759261 -0.1501888 0.91580191 1.2147558  
## 3: -0.2968265 0.7084172 0.4324540 -0.4847818 0.4116137 0.06311886 -0.1836987  
## 4: -0.6861800 0.6791455 0.3920867 -0.3991257 -1.9338488 -0.96288614 -1.0420817  
## 5: 1.5770063 -0.4146504 0.4861795 -0.9154266 -1.0404583 -0.03151305 -0.1880929  
## V14 V15 V16 V17 V18 V19  
## 1: 4.62694203 -0.92445871 1.1076406 1.99169111 0.5106323 -0.6829197  
## 2: -0.67514296 1.16493091 -0.7117573 -0.02569286 -1.2211789 -1.5455561  
## 3: -0.51060184 1.32928351 0.1407160 0.31350179 0.3956525 -0.5772518  
## 4: 0.44962444 1.96256312 -0.6085771 0.50992846 1.1139806 2.8978488  
## 5: -0.08431647 0.04133346 -0.3026201 -0.66037665 0.1674299 -0.2561169  
## V20 V21 V22 V23 V24 V25  
## 1: 1.47582913 0.2134541 0.1118637 1.01447990 -0.509348453 1.4368069  
## 2: 0.05961590 0.2142053 0.9243836 0.01246304 -1.016225669 -0.6066240  
## 3: 0.00139597 0.2320450 0.5782290 -0.03750086 0.640133881 0.2657455  
## 4: 0.12743352 0.2652449 0.8000487 -0.16329794 0.123205244 -0.5691589  
## 5: 0.38294810 0.2610573 0.6430784 0.37677701 0.008797379 -0.4736487  
## V26 V27 V28 Amount Class  
## 1: 0.2500343 0.943651172 0.82373096 0.77 0  
## 2: -0.3952551 0.068472470 -0.05352739 24.79 0  
## 3: -0.0873706 0.004454772 -0.02656083 67.88 0  
## 4: 0.5466685 0.108820735 0.10453282 10.00 0  
## 5: -0.8182671 -0.002415309 0.01364891 217.00 0

colnames(x = creditcard\_data) #shows all column names

## [1] "Time" "V1" "V2" "V3" "V4" "V5" "V6" "V7"   
## [9] "V8" "V9" "V10" "V11" "V12" "V13" "V14" "V15"   
## [17] "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"   
## [25] "V24" "V25" "V26" "V27" "V28" "Amount" "Class"

table(creditcard\_data$Class) #shows how many are marked as fraud and how many are genuine

##   
## 0 1   
## 284315 492

summary(creditcard\_data$Amount) #shows min, max, and summary values for the Amount column

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

names(creditcard\_data) #display names of all the columns

## [1] "Time" "V1" "V2" "V3" "V4" "V5" "V6" "V7"   
## [9] "V8" "V9" "V10" "V11" "V12" "V13" "V14" "V15"   
## [17] "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"   
## [25] "V24" "V25" "V26" "V27" "V28" "Amount" "Class"

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are ‘Time’ and ‘Amount’. Feature ‘Time’ contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature ‘Amount’ is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature ‘Class’ is the response variable and it takes value 1 in case of fraud and 0 otherwise.

## Data Preparation

check for any NULL values in the dataset.

nullTime <- creditcard\_data[creditcard\_data$Time == "null", ]  
head(nullTime, 5)

## Empty data.table (0 rows and 31 cols): Time,V1,V2,V3,V4,V5...

nullAmount <- creditcard\_data[creditcard\_data$Amount == "null", ]  
head(nullAmount, 5)

## Empty data.table (0 rows and 31 cols): Time,V1,V2,V3,V4,V5...

nullClass <- creditcard\_data[creditcard\_data$Class == "null", ]  
head(nullClass, 5)

## Empty data.table (0 rows and 31 cols): Time,V1,V2,V3,V4,V5...

We will scale our data using the **scale()** function. We will apply this to the amount component of our creditcard\_data amount. Scaling is also known as feature standardization. With the help of scaling, the data is structured according to a specified range. Therefore, there are no extreme values in our dataset that might interfere with the functioning of our model.

creditcard\_data$Amount=scale(creditcard\_data$Amount)  
head(creditcard\_data, 5)

## 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  
## 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  
## V13 V14 V15 V16 V17 V18  
## 1: -0.9913898 -0.3111694 1.4681770 -0.4704005 0.2079712 0.02579058  
## 2: 0.4890950 -0.1437723 0.6355581 0.4639170 -0.1148047 -0.18336127  
## 3: 0.7172927 -0.1659459 2.3458649 -2.8900832 1.1099694 -0.12135931  
## 4: 0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.6840928 1.96577500  
## 5: 1.3458516 -1.1196698 0.1751211 -0.4514492 -0.2370332 -0.03819479  
## V19 V20 V21 V22 V23 V24  
## 1: 0.4039930 0.25141210 -0.018306778 0.277837576 -0.1104739 0.06692807  
## 2: -0.1457830 -0.06908314 -0.225775248 -0.638671953 0.1012880 -0.33984648  
## 3: -2.2618571 0.52497973 0.247998153 0.771679402 0.9094123 -0.68928096  
## 4: -1.2326220 -0.20803778 -0.108300452 0.005273597 -0.1903205 -1.17557533  
## 5: 0.8034869 0.40854236 -0.009430697 0.798278495 -0.1374581 0.14126698  
## V25 V26 V27 V28 Amount Class  
## 1: 0.1285394 -0.1891148 0.133558377 -0.02105305 0.24496383 0  
## 2: 0.1671704 0.1258945 -0.008983099 0.01472417 -0.34247394 0  
## 3: -0.3276418 -0.1390966 -0.055352794 -0.05975184 1.16068389 0  
## 4: 0.6473760 -0.2219288 0.062722849 0.06145763 0.14053401 0  
## 5: -0.2060096 0.5022922 0.219422230 0.21515315 -0.07340321 0

tail(creditcard\_data, 5)

## Time V1 V2 V3 V4 V5 V6  
## 1: 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278 -2.6068373  
## 2: 172787 -0.7327887 -0.05508049 2.0350297 -0.7385886 0.86822940 1.0584153  
## 3: 172788 1.9195650 -0.30125385 -3.2496398 -0.5578281 2.63051512 3.0312601  
## 4: 172788 -0.2404400 0.53048251 0.7025102 0.6897992 -0.37796113 0.6237077  
## 5: 172792 -0.5334125 -0.18973334 0.7033374 -0.5062712 -0.01254568 -0.6496167  
## V7 V8 V9 V10 V11 V12 V13  
## 1: -4.9182154 7.3053340 1.9144283 4.3561704 -1.5931053 2.71194079 -0.6892556  
## 2: 0.0243297 0.2948687 0.5848000 -0.9759261 -0.1501888 0.91580191 1.2147558  
## 3: -0.2968265 0.7084172 0.4324540 -0.4847818 0.4116137 0.06311886 -0.1836987  
## 4: -0.6861800 0.6791455 0.3920867 -0.3991257 -1.9338488 -0.96288614 -1.0420817  
## 5: 1.5770063 -0.4146504 0.4861795 -0.9154266 -1.0404583 -0.03151305 -0.1880929  
## V14 V15 V16 V17 V18 V19  
## 1: 4.62694203 -0.92445871 1.1076406 1.99169111 0.5106323 -0.6829197  
## 2: -0.67514296 1.16493091 -0.7117573 -0.02569286 -1.2211789 -1.5455561  
## 3: -0.51060184 1.32928351 0.1407160 0.31350179 0.3956525 -0.5772518  
## 4: 0.44962444 1.96256312 -0.6085771 0.50992846 1.1139806 2.8978488  
## 5: -0.08431647 0.04133346 -0.3026201 -0.66037665 0.1674299 -0.2561169  
## V20 V21 V22 V23 V24 V25  
## 1: 1.47582913 0.2134541 0.1118637 1.01447990 -0.509348453 1.4368069  
## 2: 0.05961590 0.2142053 0.9243836 0.01246304 -1.016225669 -0.6066240  
## 3: 0.00139597 0.2320450 0.5782290 -0.03750086 0.640133881 0.2657455  
## 4: 0.12743352 0.2652449 0.8000487 -0.16329794 0.123205244 -0.5691589  
## 5: 0.38294810 0.2610573 0.6430784 0.37677701 0.008797379 -0.4736487  
## V26 V27 V28 Amount Class  
## 1: 0.2500343 0.943651172 0.82373096 -0.35015025 0  
## 2: -0.3952551 0.068472470 -0.05352739 -0.25411639 0  
## 3: -0.0873706 0.004454772 -0.02656083 -0.08183916 0  
## 4: 0.5466685 0.108820735 0.10453282 -0.31324798 0  
## 5: -0.8182671 -0.002415309 0.01364891 0.51435441 0

summary(creditcard\_data$Amount)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.35323 -0.33084 -0.26527 0.00000 -0.04472 102.36206

NewData1 <- creditcard\_data[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
NewData1 <- NewData1[,-c(2)]  
head(NewData1)

## Time Amount Class  
## 1: 0 0.24496383 0  
## 2: 0 -0.34247394 0  
## 3: 1 1.16068389 0  
## 4: 1 0.14053401 0  
## 5: 2 -0.07340321 0  
## 6: 2 -0.33855582 0

dim(NewData1)

## [1] 284807 3

NewData=creditcard\_data[,-c(1)]  
head(NewData)

## V1 V2 V3 V4 V5 V6  
## 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  
## 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 0.24496383 0  
## 2: 0.1671704 0.1258945 -0.008983099 0.01472417 -0.34247394 0  
## 3: -0.3276418 -0.1390966 -0.055352794 -0.05975184 1.16068389 0  
## 4: 0.6473760 -0.2219288 0.062722849 0.06145763 0.14053401 0  
## 5: -0.2060096 0.5022922 0.219422230 0.21515315 -0.07340321 0  
## 6: -0.2327938 0.1059148 0.253844225 0.08108026 -0.33855582 0

## Data Modelling

We will **split** our dataset into training set as well as test set with a split ratio of 0.80. This means that 80% of our data will be attributed to the **train\_data** whereas 20% will be attributed to the **test\_data**.

library(caTools)

## Warning: package 'caTools' was built under R version 4.1.2

set.seed(123)  
data\_sample = sample.split(NewData$Class,SplitRatio=0.80)  
train\_data = subset(NewData,data\_sample==TRUE)  
test\_data = subset(NewData,data\_sample==FALSE)  
dim(train\_data)

## [1] 227846 30

dim(test\_data)

## [1] 56961 30

## Code and graphs

#### Logistic Regression

A logistic regression is used for modeling the outcome probability of a class such as pass/fail, positive/negative and in our case – fraud/not fraud.

Logistic\_Model=glm(Class~.,test\_data,family=binomial())

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

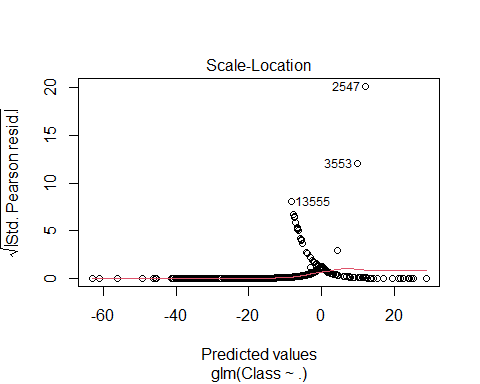
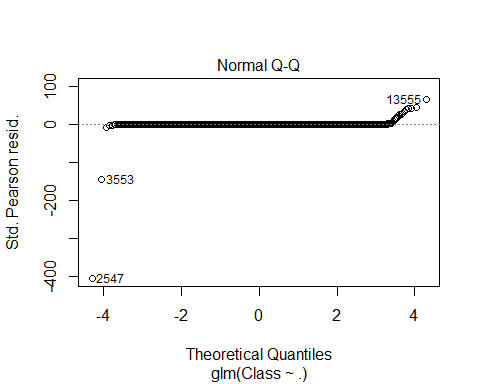
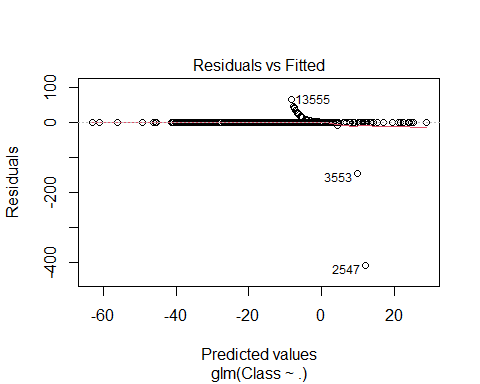
summary(Logistic\_Model)

##   
## Call:  
## glm(formula = Class ~ ., family = binomial(), data = test\_data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.9019 -0.0254 -0.0156 -0.0078 4.0877   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -12.52800 10.30537 -1.216 0.2241   
## V1 -0.17299 1.27381 -0.136 0.8920   
## V2 1.44512 4.23062 0.342 0.7327   
## V3 0.17897 0.24058 0.744 0.4569   
## V4 3.13593 7.17768 0.437 0.6622   
## V5 1.49014 3.80369 0.392 0.6952   
## V6 -0.12428 0.22202 -0.560 0.5756   
## V7 1.40903 4.22644 0.333 0.7388   
## V8 -0.35254 0.17462 -2.019 0.0435 \*  
## V9 3.02176 8.67262 0.348 0.7275   
## V10 -2.89571 6.62383 -0.437 0.6620   
## V11 -0.09769 0.28270 -0.346 0.7297   
## V12 1.97992 6.56699 0.301 0.7630   
## V13 -0.71674 1.25649 -0.570 0.5684   
## V14 0.19316 3.28868 0.059 0.9532   
## V15 1.03868 2.89256 0.359 0.7195   
## V16 -2.98194 7.11391 -0.419 0.6751   
## V17 -1.81809 4.99764 -0.364 0.7160   
## V18 2.74772 8.13188 0.338 0.7354   
## V19 -1.63246 4.77228 -0.342 0.7323   
## V20 -0.69925 1.15114 -0.607 0.5436   
## V21 -0.45082 1.99182 -0.226 0.8209   
## V22 -1.40395 5.18980 -0.271 0.7868   
## V23 0.19026 0.61195 0.311 0.7559   
## V24 -0.12889 0.44701 -0.288 0.7731   
## V25 -0.57835 1.94988 -0.297 0.7668   
## V26 2.65938 9.34957 0.284 0.7761   
## V27 -0.45396 0.81502 -0.557 0.5775   
## V28 -0.06639 0.35730 -0.186 0.8526   
## Amount 0.22576 0.71892 0.314 0.7535   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1443.40 on 56960 degrees of freedom  
## Residual deviance: 378.59 on 56931 degrees of freedom  
## AIC: 438.59  
##   
## Number of Fisher Scoring iterations: 17

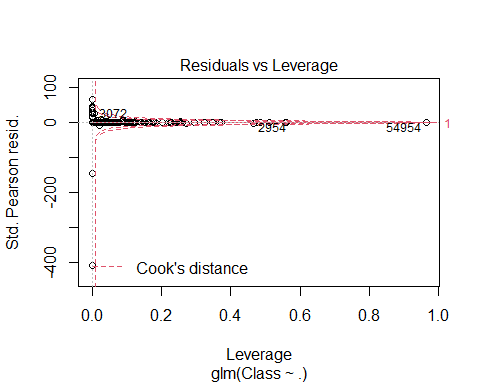
#### Plots

We will visual it through the following plots:

plot(Logistic\_Model)



## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced  
  
## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced

 We learnt how to develop our credit card fraud detection model using machine learning. We used a Logistic Regression to implement this model and also plotted the respective performance curves for the model. We learnt how data can be analyzed and visualized to discern fraudulent transactions from other types of data.