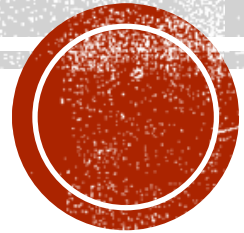
DETECTING CREDIT-CARD FRAUD



By:
Mohammad Sherif Mousa Dabour

PROBLEM STATEMENT

- The main aim of the project is to detect the fraudulent transaction in credit card
- It includes modeling past transaction that turned out to be fraud
- The new model is used to detect, if the new transaction is fraud or not
- It will minimize the fraud transaction while 100% detecting the fraudulent transaction

SAMPLE DATA SET

- https://drive.google.com/file/d/1CTAlmlREFRaEN3NoHHitewpqAtWS5cVQ/view
- Data set consist of mix data with total 284,807 observations
- 28 numerical value, namely V1 to V28
- There is no missing value in dataset
- We will be importing the dataset, and performing exploration, data manipulation followed by data modelling

ALGORITHM/METHODOLOGY

- There are various algorithm that will come into place, namely

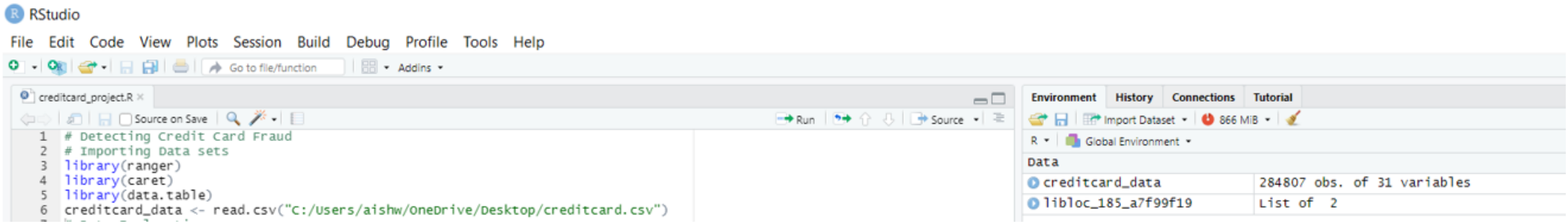
1. Decision tree
2. Logistic regression
3. Artificial Neural Network
4. Gradient boosting classifier

SOFTWARE

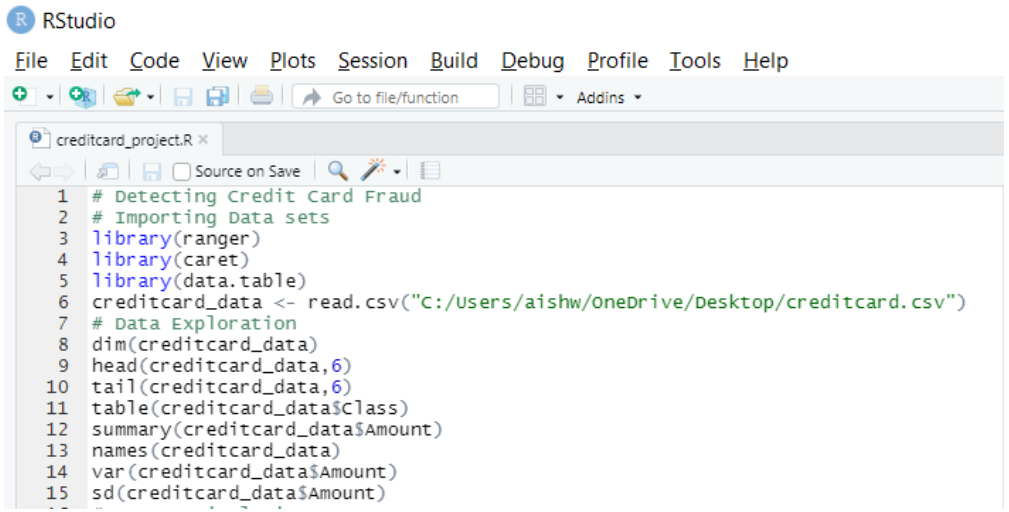
- To execute the specified task, we will be using R programming language

IMPORTING DATASET THAT CONTAIN THE TRANSACTIONS MADE BY CREDIT CARD

- We can see that the data set has 284807 observations of 31 variables



DATA EXPLORATION



- We are looking at the data in the credit card data frame in this section of the fraud detection-machine learning project.
- The credit card data will be displayed using the head() and tail() functions.
- After that, we will look at the other elements of this data frame
- Here is the code for data exploration..

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Source

Console Terminal Jobs

R 4.1.1 · C:/Users/aishw/AppData/Local/Temp/RtmpUbb1D4/

```
> # Detecting Credit Card Fraud
> # Importing Data sets
> library(ranger)
> library(caret)
> library(data.table)
> creditcard_data <- read.csv("C:/Users/aishw/OneDrive/Desktop/creditcard.csv")
> # Data Exploration
> dim(creditcard_data)
[1] 284807     31
> head(creditcard_data,6)
  Time         V1          V2        V3         V4          V5          V6          V7          V8         V9
1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870
2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251
3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543
4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241
5    2 -1.1582331  0.87773676 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393
6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714
         V10        V11        V12        V13        V14        V15        V16        V17        V18         V19
1  0.09079417 -0.5515995 -0.61780086 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296
2 -0.16697441  1.6127267  1.06523531  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304
3  0.20764287  0.6245015  0.06608369  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185709
4 -0.05495192 -0.2264873  0.17822823  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197
5  0.75307443 -0.8228429  0.53819555  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692
6 -0.37140720  1.3412620  0.35989384 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379
         V20          V21         V22         V23         V24        V25         V26          V27         V28 Amount
1  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692808  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62
2 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704  0.1258945 -0.008983099  0.01472417   2.69
3  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66
4 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760 -0.2219288  0.062722849  0.06145763 123.50
5  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096  0.5022922  0.219422230  0.21515315  69.99
6  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938  0.1059148  0.253844225  0.08108026   3.67
  Class
1     0
2     0
3     0
4     0
5     0
6     0
> tail(creditcard_data,6)
         Time          V1          V2         V3         V4          V5         V6         V7          V8         V9
284802 172785   0.1203164  0.93100513 -0.5460121 -0.7450968  1.13031398 -0.2359732  0.8127221  0.1150929 -0.2040635
284803 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278 -2.6068373 -4.9182154  7.3053340  1.9144283
284804 172787  -0.7327887 -0.05508049  2.0350297 -0.7385886  0.86822940  1.0584153  0.0243297  0.2948687  0.5848000
284805 172788   1.9195650 -0.30125385 -3.2496398 -0.5578281  2.63051512  3.0312601 -0.2968265  0.7084172  0.4324540
284806 172788  -0.2404400  0.53048251  0.7025102  0.6897992 -0.37796113  0.6237077 -0.6861800  0.6791455  0.3920867
284807 172792  -0.5334125 -0.18973334  0.7033374 -0.5062712 -0.01254568 -0.6496167  1.5770063 -0.4146504  0.4861795
             V10        V11        V12        V13        V14        V15        V16        V17        V18
284802 -0.6574221  0.6448373  0.19091623 -0.5463289 -0.7317065 -0.80803553  0.5996281  0.07044075  0.3731103
284803  4.3561704 -1.5931053  2.71194079 -0.6892556  4.62694202 -0.92445871  1.1076406  1.99169111  0.5106323
284804 -0.9759261 -0.1501888  0.91580191  1.2147558 -0.67514296  1.16493091 -0.7117573 -0.02569286 -1.2211789
284805 -0.4847818  0.4116137  0.06311886 -0.1836987 -0.51060184  1.32928351  0.1407160  0.31350179  0.3956525
284806 -0.3991257 -1.9338488 -0.96288614 -1.0420817  0.44962444  1.96256312 -0.6085771  0.50992846  1.1139806
284807 -0.9154266 -1.0404583 -0.03151305 -0.1880929 -0.08431647  0.04133345 -0.3026201 -0.66037665  0.1674299
             V19         V20         V21         V22         V23          V24         V25         V26          V27
284802  0.1289038 0.000675833 -0.3142046 -0.8085204  0.05034266  0.102799590 -0.4358701  0.1240789  0.217939865
284803 -0.6829197 1.475829135  0.2134541  0.1118637  1.01447990 -0.509348453  1.4368069  0.2500343  0.943651172
284804 -1.5455561 0.059615900  0.2142053  0.9243836  0.01246304 -1.016225669 -0.6066240 -0.3952551  0.068472470
```

Environment History Connections Tutorial

Import Dataset | 897 MiB | List | R | Global Environment

```
 $ 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 [1:284807, 1] 0.245 -0.3425 1.1607 0.1405 -0.0734 ...
  ..- attr(*, "scaled:center")= num 88.3
  ..- attr(*, "scaled:scale")= num 250
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

libloc_185_a7f99f19	List of 2
NewData	284807 obs. of 30 variables
test_data	56961 obs. of 30 variables
train_data	227846 obs. of 30 variables

Files Plots Packages Help Viewer

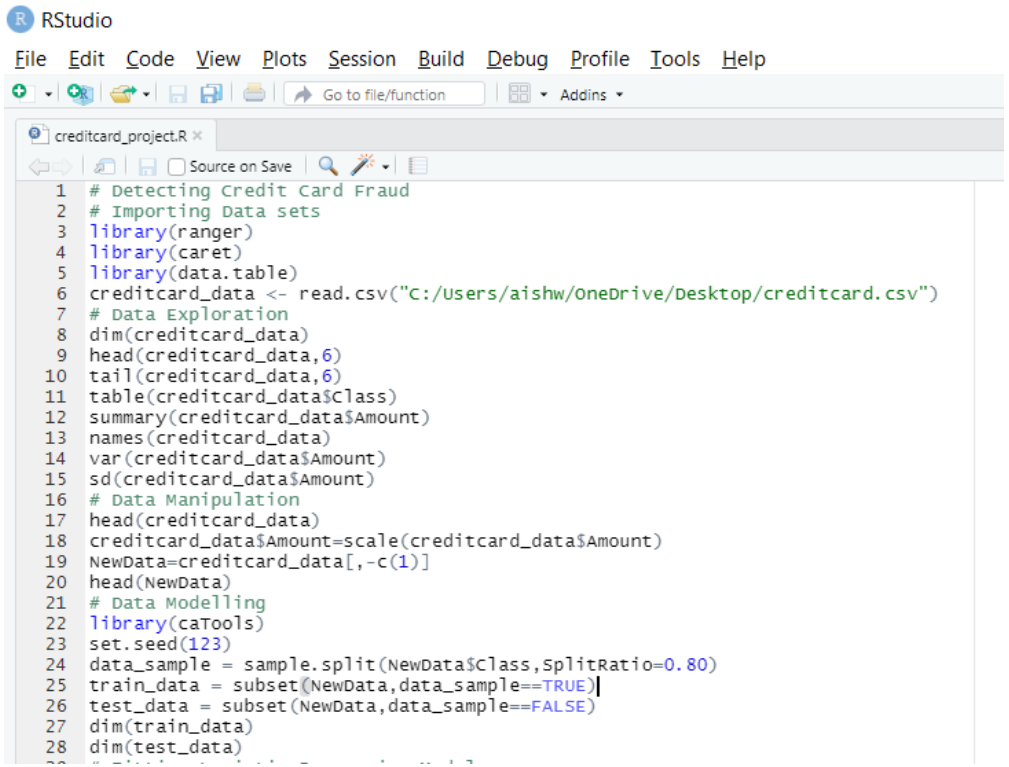
Zoom | Export

Type here to search

Raining now

2:54 PM
5/6/2022

DATA MANIPULATION



- We use the scale() function to scale our data in this phase of the R data science project. This will be applied to the credit card data amount's amount component
- Feature standardization is another term for scaling. The data is organized according to a given range with the help of scaling
- As a result, our dataset contains no extreme values that could cause our model to malfunction
- This is the code for data manipulation

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Source

Console Terminal Jobs

R 4.1.1 · C:/Users/aishw/AppData/Local/Temp/RtmpUbb1D4/

```
> sd(creditcard_data$Amount)
[1] 250.1201
> # Data Manipulation
> head(creditcard_data)
  Time         V1          V2        V3         V4          V5          V6          V7          V8         V9
1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870
2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251
3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543
4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241
5    2 -1.1582331  0.87773676 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393
6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714
         V10        V11        V12        V13        V14        V15        V16        V17         V18         V19
1  0.09079417 -0.5515995 -0.61780086 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296
2 -0.16697441  1.6127267  1.06523531  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304
3  0.20764287  0.6245015  0.06608369  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185709
4 -0.05495192 -0.2264873  0.17822823  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197
5  0.75307443 -0.8228429  0.53819555  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692
6 -0.37140720  1.3412620  0.35989384 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379
         V20         V21         V22         V23         V24         V25         V26         V27         V28 Amount
1  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692808  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62
2 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704  0.1258945 -0.008983099  0.01472417   2.69
3  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66
4 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760 -0.2219288  0.062722849  0.06145763 123.50
5  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096  0.5022922  0.219422230  0.21515315  69.99
6  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938  0.1059148  0.253844225  0.08108026   3.67
  Class
1     0
2     0
3     0
4     0
5     0
6     0
> creditcard_data$Amount=scale(creditcard_data$Amount)
> NewData=creditcard_data[,-c(1)]
> head(NewData)
         V1          V2        V3         V4          V5          V6          V7          V8         V9         V10
1 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870  0.09079417
2  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251 -0.16697441
3 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543  0.20764287
4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241 -0.05495192
5 -1.1582331  0.87773676 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393  0.75307443
6 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714 -0.37140720
         V11        V12        V13        V14        V15        V16        V17         V18         V19         V20
1 -0.5515995 -0.61780086 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296  0.25141210
2  1.6127267  1.06523531  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304 -0.06908314
3  0.6245015  0.06608369  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185709  0.52497973
4 -0.2264873  0.17822823  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197 -0.20803778
5 -0.8228429  0.53819555  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692  0.40854236
6  1.3412620  0.35989384 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379  0.08496767
          V21         V22         V23         V24         V25         V26         V27         V28      Amount Class
1 -0.018306778  0.277837576 -0.11047391  0.06692808  0.1285394 -0.1891148  0.133558377 -0.02105305  0.24496383     0
2 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704  0.1258945 -0.008983099  0.01472417 -0.34247394     0
3  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418 -0.1390966 -0.055352794 -0.05975184  1.16068389     0
4 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760 -0.2219288  0.062722849  0.06145763  0.14053401     0
5 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096  0.5022922  0.219422230  0.21515315 -0.07340321     0
6 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938  0.1059148  0.253844225  0.08108026 -0.33855582     0
> # Data Modelling
```

Environment History Connections Tutorial

Import Dataset | 897 MiB | List

R | Global Environment

```
 $ 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 [1:284807, 1] 0.245 -0.3425 1.1607 0.1405 -0.0734 ...
  ..- attr(*, "scaled:center")= num 88.3
  ..- attr(*, "scaled:scale")= num 250
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

libloc_185_a7f99f19	List of 2
NewData	284807 obs. of 30 variables
test_data	56961 obs. of 30 variables
train_data	227846 obs. of 30 variables

Files Plots Packages Help Viewer

Zoom | Export

Type here to search

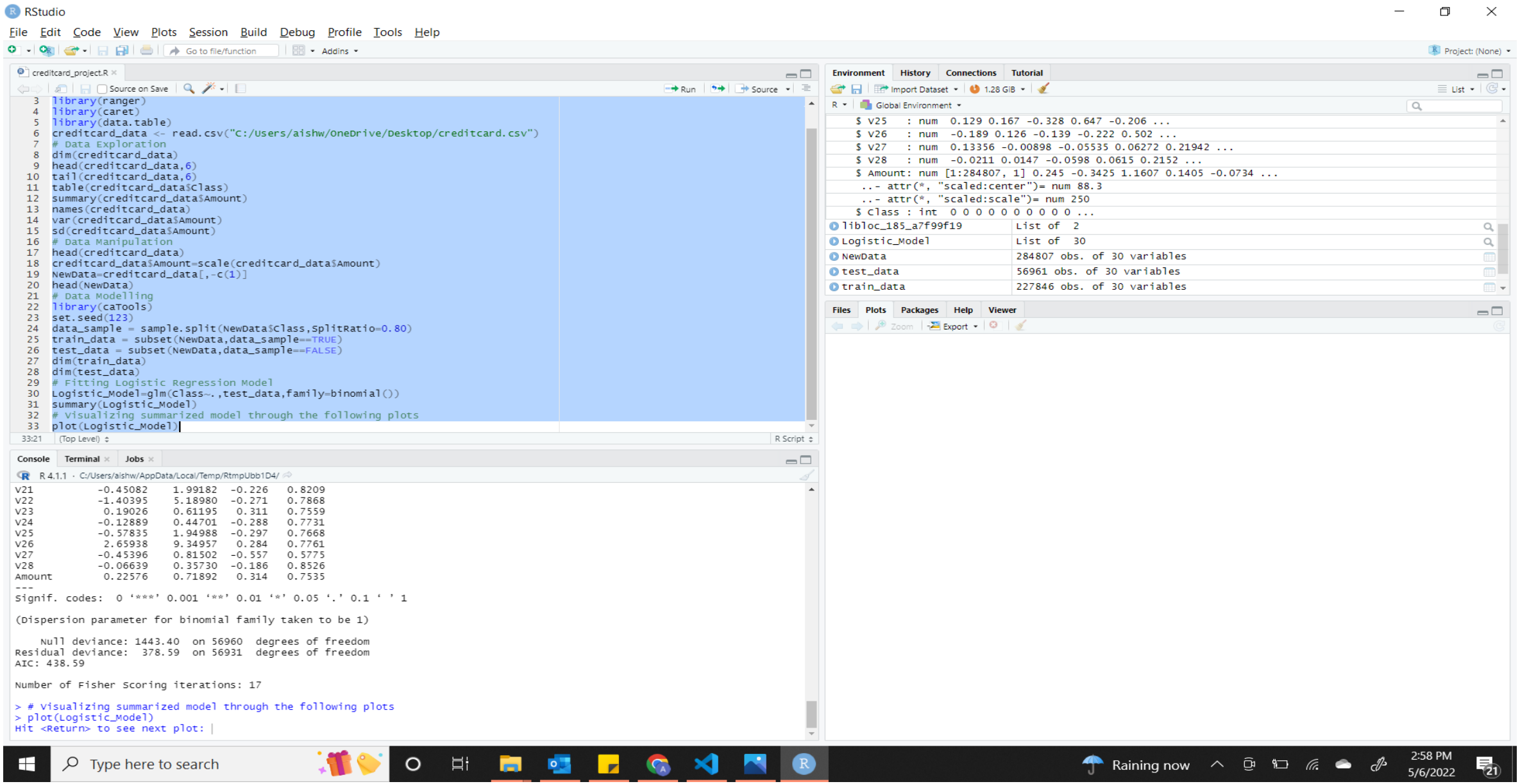
57°F Rain

2:54 PM
5/6/2022

DATA MODELLING

- We will split our dataset into training and test sets with a split ratio of 0.80 after we have standardized our entire dataset
- This indicates that the train data will account for 80% of our data, while the test data will account for 20%
- The dimensions will be found using the dim() function

DATA MODELLING



FITTING LOGISTIC REGRESSION MODEL

- We will fit our first model in this phase of the credit card fraud detection project
- We will get started with logistic regression
- A logistic regression is used to model the probability of a class's outcome, such as pass/fail, positive/negative, and fraud/not fraud in our project
- We use the following procedure to apply this model to our data

FITTING LOGISTIC REGRESSION MODEL

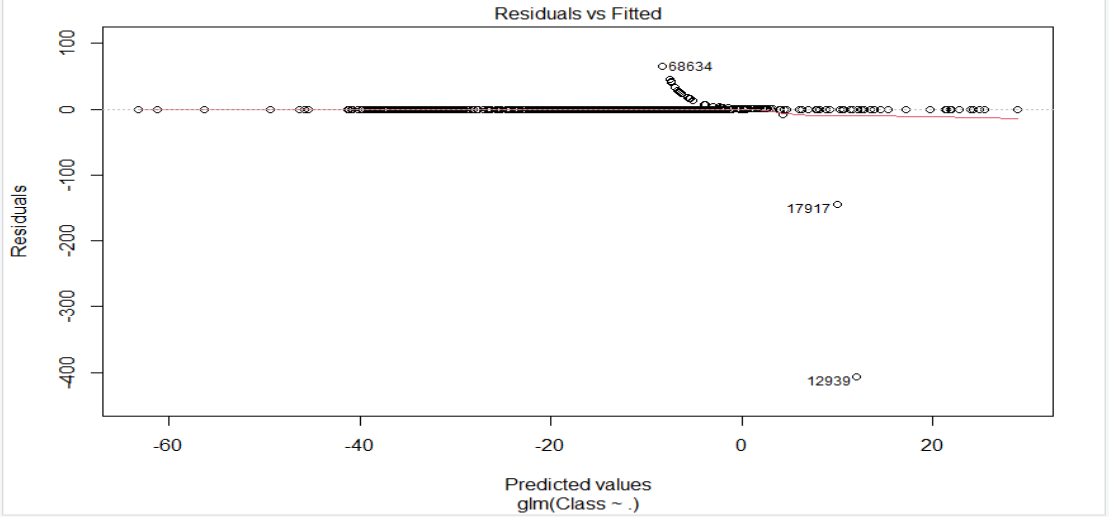
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
> 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

> # Visualizing summarized model through the following plots
> plot(Logistic_Model)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
```

```
$ 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 [1:284807, 1] 0.245 -0.3425 1.1607 0.1405 -0.0734 ...
 ..- attr(*, "scaled:center")= num 88.3
 ..- attr(*, "scaled:scale")= num 250
$ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

libloc_185_a7f99f19	List of 2
Logistic_Model	List of 30
NewData	284807 obs. of 30 variables
test_data	56961 obs. of 30 variables
train_data	227846 obs. of 30 variables

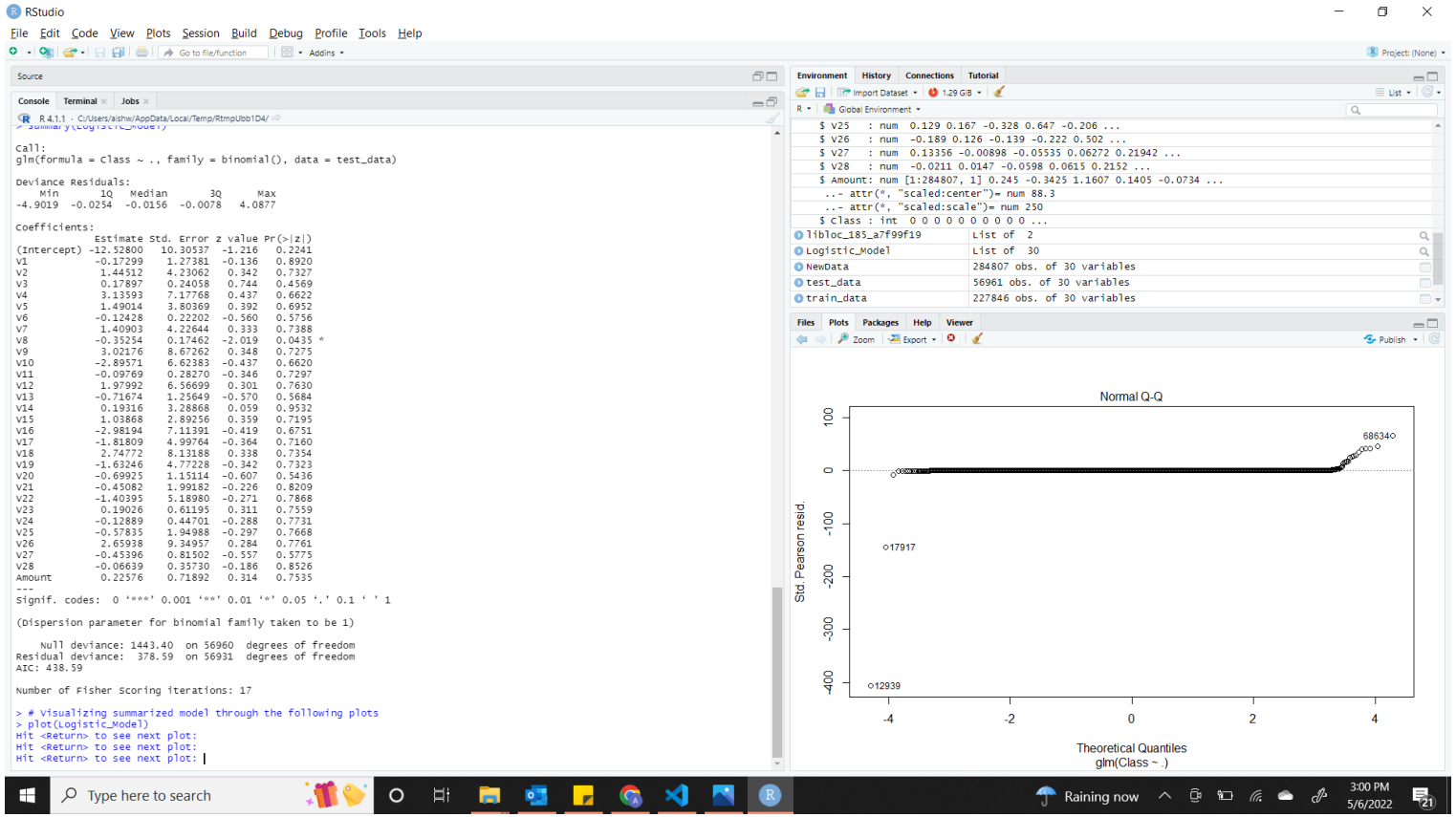


Type here to search

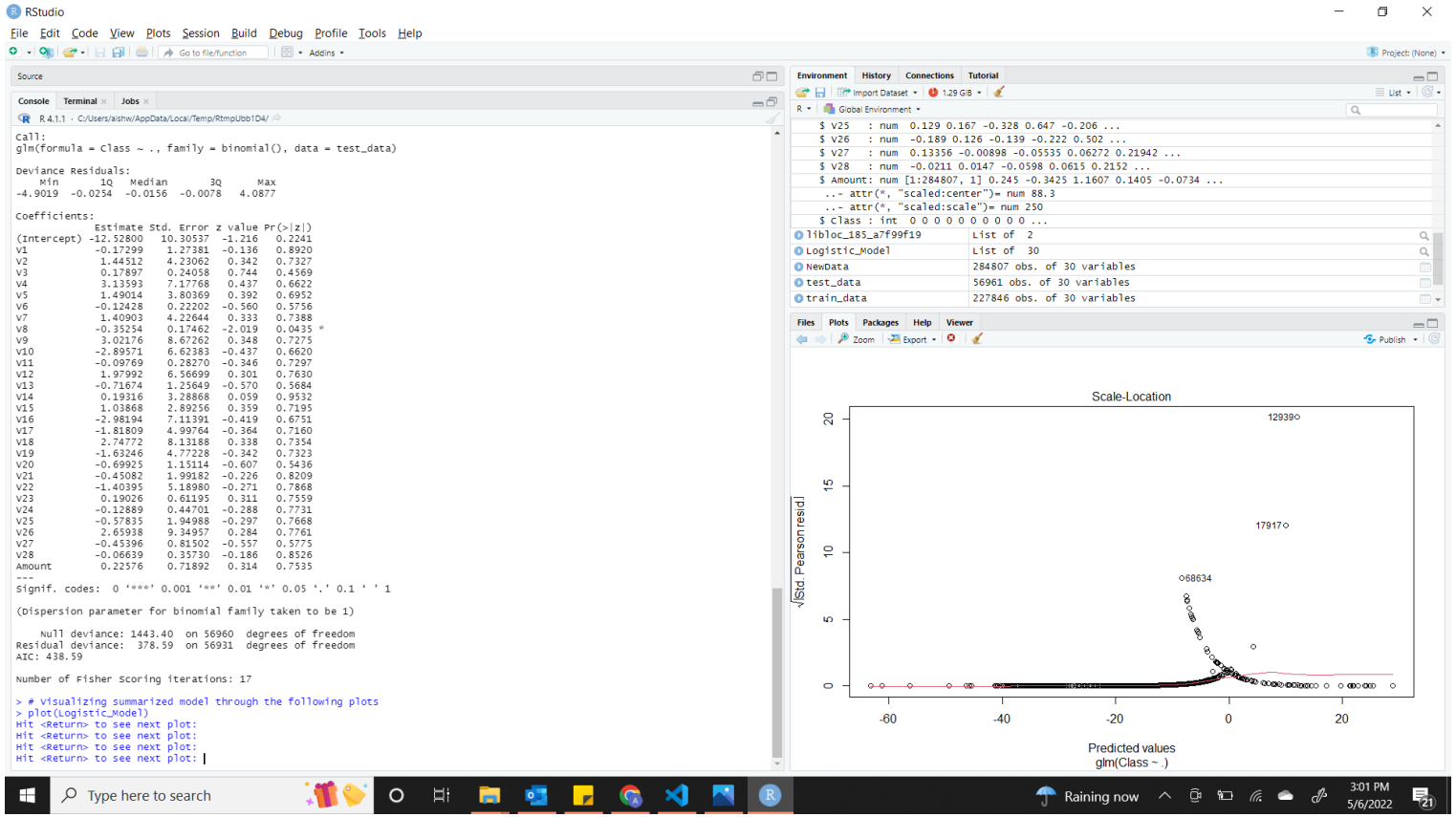
Raining now

2:59 PM
5/6/2022

FITTING LOGISTIC REGRESSION MODEL



FITTING LOGISTIC REGRESSION MODEL



FITTING LOGISTIC REGRESSION MODEL

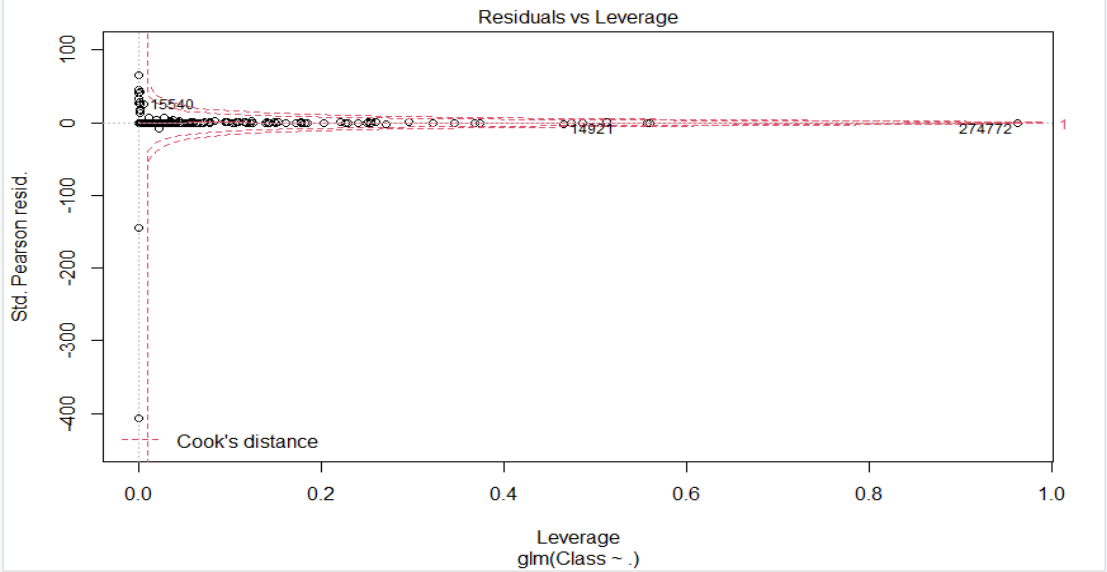
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
    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

> # Visualizing summarized model through the following plots
> plot(Logistic_Model)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Warning messages:
1: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced
2: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced
>
```

```
    $ 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 [1:284807, 1] 0.245 -0.3425 1.1607 0.1405 -0.0734 ...
     ..- attr(*, "scaled:center")= num 88.3
     ..- attr(*, "scaled:scale")= num 250
    $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

libloc_185_a7f99f19	List of 2
Logistic_Model	List of 30
NewData	284807 obs. of 30 variables
test_data	56961 obs. of 30 variables
train_data	227846 obs. of 30 variables



Type here to search

Raining now

3:02 PM
5/6/2022

FITTING LOGISTIC REGRESSION MODEL

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

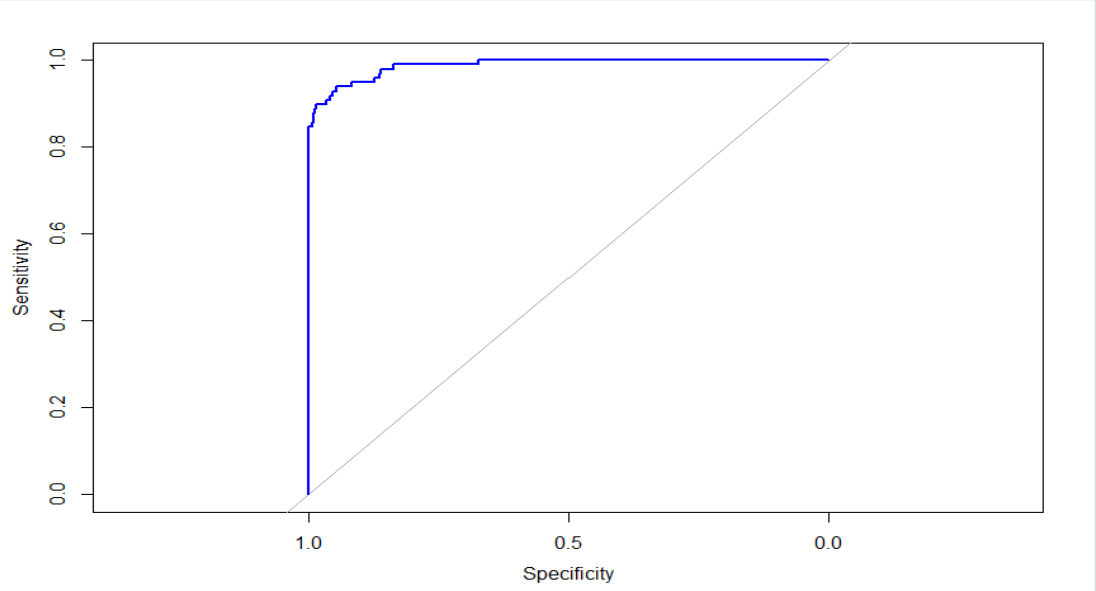
creditcard_project.R

```
# Data Exploration
dim(creditcard_data)
head(creditcard_data,6)
tail(creditcard_data,6)
table(creditcard_data$Class)
summary(creditcard_data$Amount)
names(creditcard_data)
var(creditcard_data$Amount)
sd(creditcard_data$Amount)
# Data Manipulation
head(creditcard_data)
creditcard_data$Amount=scale(creditcard_data$Amount)
NewData=creditcard_data[,-c(1)]
head(NewData)
# Data Modelling
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
dim(test_data)
# Fitting Logistic Regression Model
Logistic_Model=glm(Class~.,test_data,family=binomial())
summary(Logistic_Model)
# Visualizing summarized model through the following plots
plot(Logistic_Model)
# ROC Curve to assess the performance of the model
library(pROC)
lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
```

```
Hit <Return> to see next plot: # ROC Curve to assess the performance of the model
Hit <Return> to see next plot: library(pROC)
Hit <Return> to see next plot: lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
Hit <Return> to see next plot: auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
Warning messages:
1: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced
2: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced
> library(pROC)
Type 'citation("pROC")' for a citation.

Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var

Warning message:
package 'pROC' was built under R version 4.1.3
> lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
> auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
Setting levels: control = 0, case = 1
Setting direction: controls < cases
>
```

Environment | History | Connections | Tutorial

$ Amount: num [1:284807, 1] 0.245 -0.3425 1.1607 0.1405 -0.0734 ...	
..- attr(*, "scaled:center")= num 88.3	
..- attr(*, "scaled:scale")= num 250	
$ Class : int 0 0 0 0 0 0 0 0 0 0 ...	
libloc_185_a7f99f19	List of 2
Logistic_Model	List of 30
NewData	284807 obs. of 30 variables
test_data	56961 obs. of 30 variables
train_data	227846 obs. of 30 variables
Values	
data_sample	Large logical (284807 elements, 1.1 MB)
lr.predict	Large numeric (56961 elements, 4.1 MB)

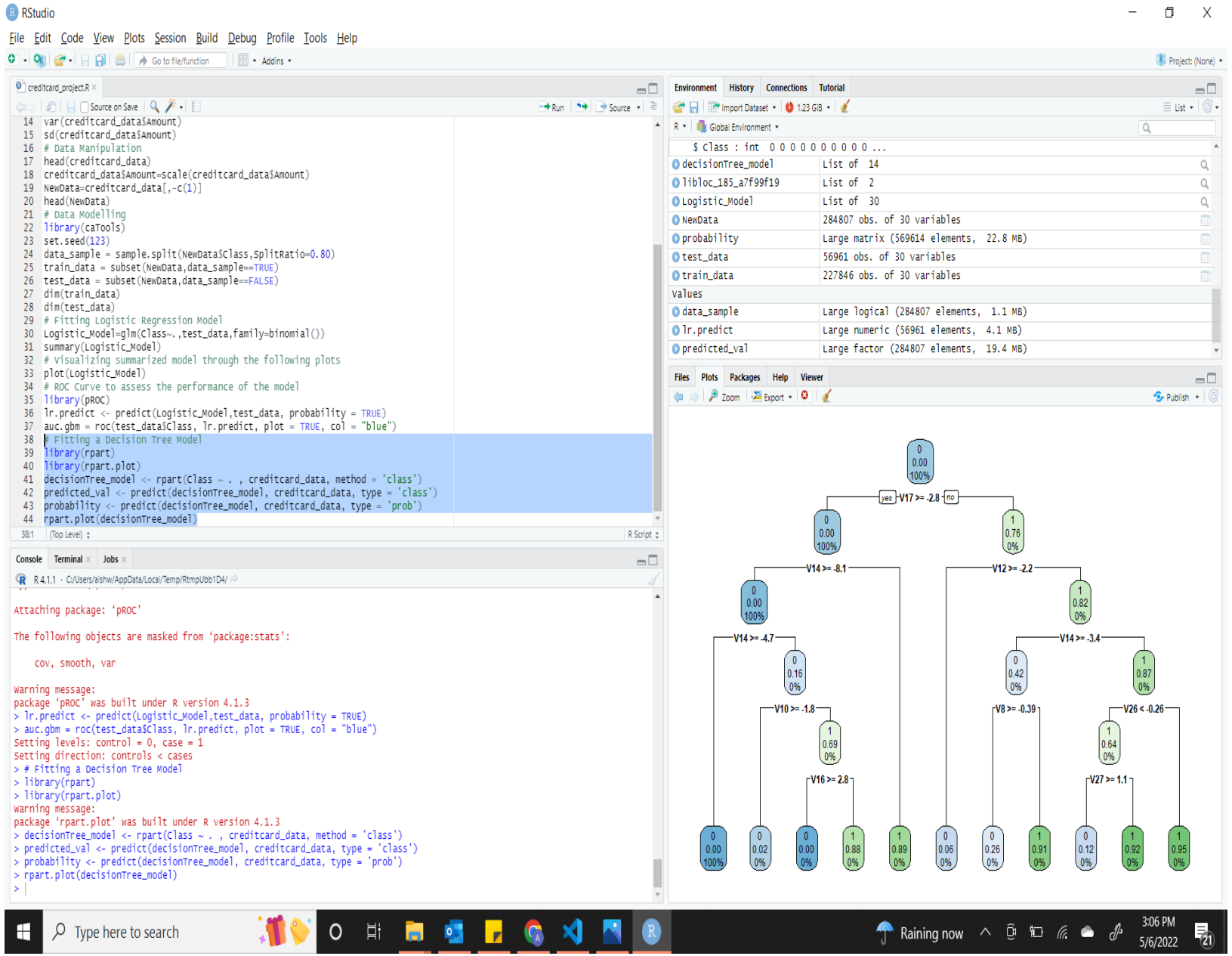


Type here to search

Raining now

3:04 PM
5/6/2022

DECISION TREE



- We will implement a decision tree algorithm in this part
- To plot the outcomes of a decision, use Decision Trees, these outcomes are essentially a result that allows us to determine which class the object belongs too
- Our decision tree model will now be implemented and plotted using the rpart.plot() function
- The recursive splitting will be used to plot the decision tree particularly.

ARTIFICIAL NEURAL NETWORK

- Artificial Neural Networks (ANNs) are a form of machine learning algorithm inspired by the human nervous system
- The ANN models are capable of learning patterns from past data and performing classification on the input data
- We import the neural net package, which will allow us to create our artificial neural networks
- Then, using the plot() function, we plotted it.
- In the case of Artificial Neural Networks, there is now a value range between 1 and 0
- We set a threshold of 0.5, which means that values greater than 0.5 correspond to 1 and the remainder to 0

ARTIFICIAL NEURAL NETWORK

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

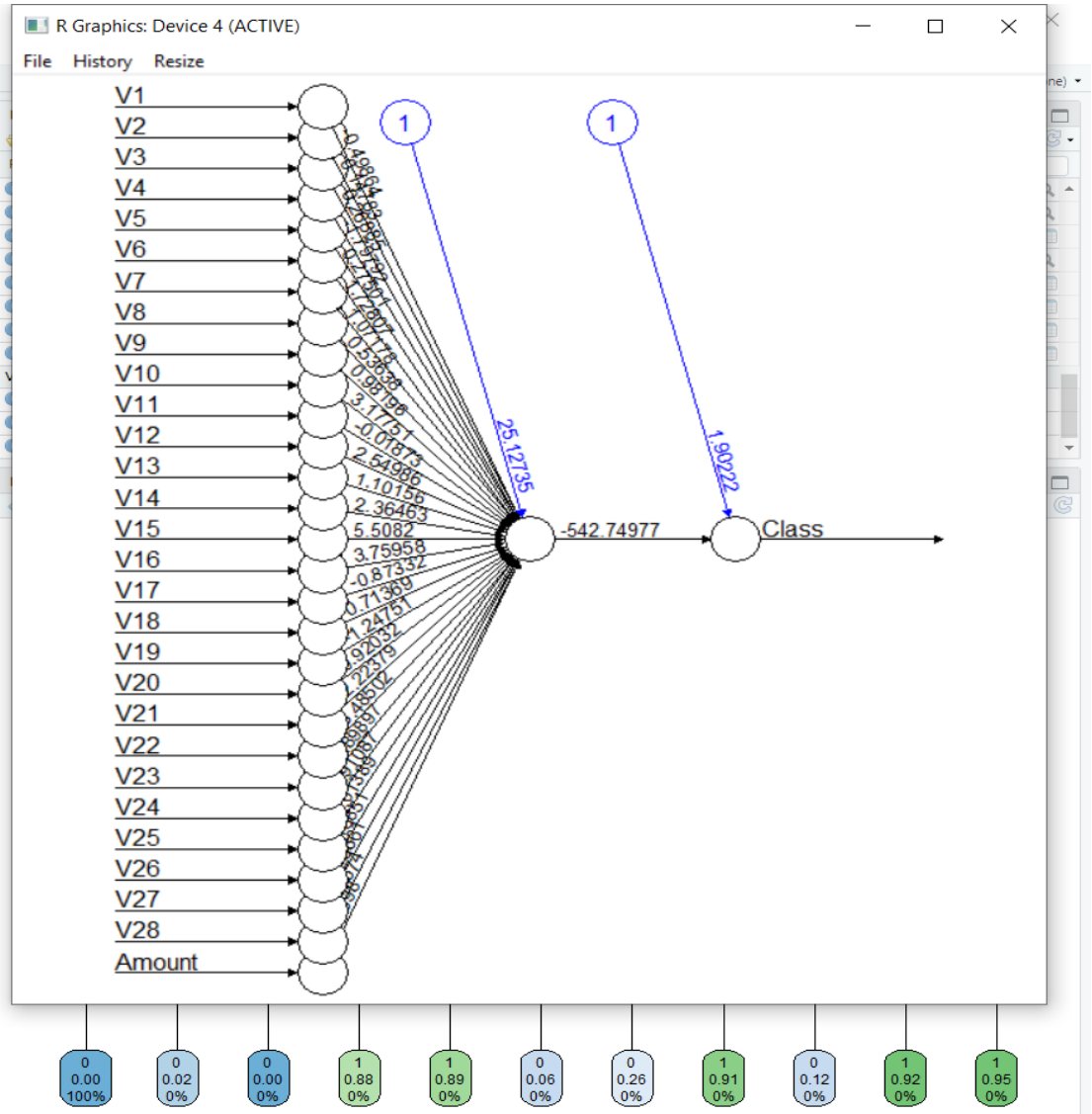
creditcard_project.R

```
# Data Modelling
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
dim(test_data)
# Fitting Logistic Regression Model
Logistic_Model=glm(Class~.,test_data,family=binomial())
summary(Logistic_Model)
# Visualizing summarized model through the following plots
plot(Logistic_Model)
# ROC Curve to assess the performance of the model
library(pROC)
lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
# Fitting a Decision Tree Model
library(rpart)
library(rpart.plot)
decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')
predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')
probability <- predict(decisionTree_model, creditcard_data, type = 'prob')
rpart.plot(decisionTree_model)
# Artificial Neural Network
library(neuralnet)
ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)
plot(ANN_model)
predANN=compute(ANN_model,test_data)
resultANN=predANN$net.result
resultANN=ifelse(resultANN>0.5,1,0)
```

Console Terminal Jobs

R 4.1.1 · C:/Users/aishw/AppData/Local/Temp/RtmpUbb1D4/

```
> lr.predict <- predict(Logistic_Model,test_data, probability = TRUE)
> auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> # Fitting a Decision Tree Model
> library(rpart)
> library(rpart.plot)
warning message:
package 'rpart.plot' was built under R version 4.1.3
> decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')
> predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')
> probability <- predict(decisionTree_model, creditcard_data, type = 'prob')
> rpart.plot(decisionTree_model)
> # Artificial Neural Network
> library(neuralnet)
warning message:
package 'neuralnet' was built under R version 4.1.3
> ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)
> plot(ANN_model)
> predANN=compute(ANN_model,test_data)
> resultANN=predANN$net.result
> resultANN=ifelse(resultANN>0.5,1,0)
>
```



Type here to search

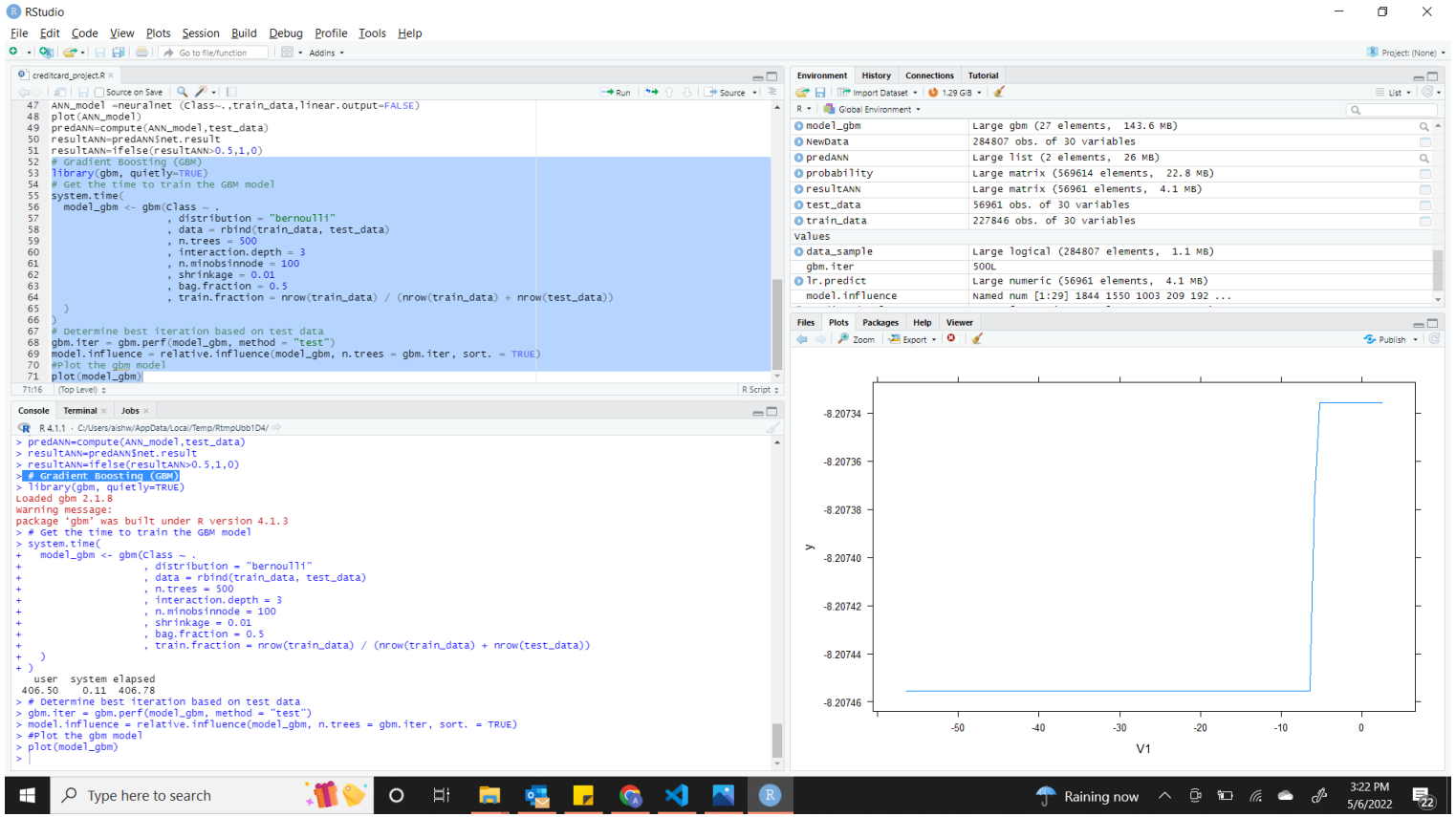
Raining now

3:13 PM
5/6/2022

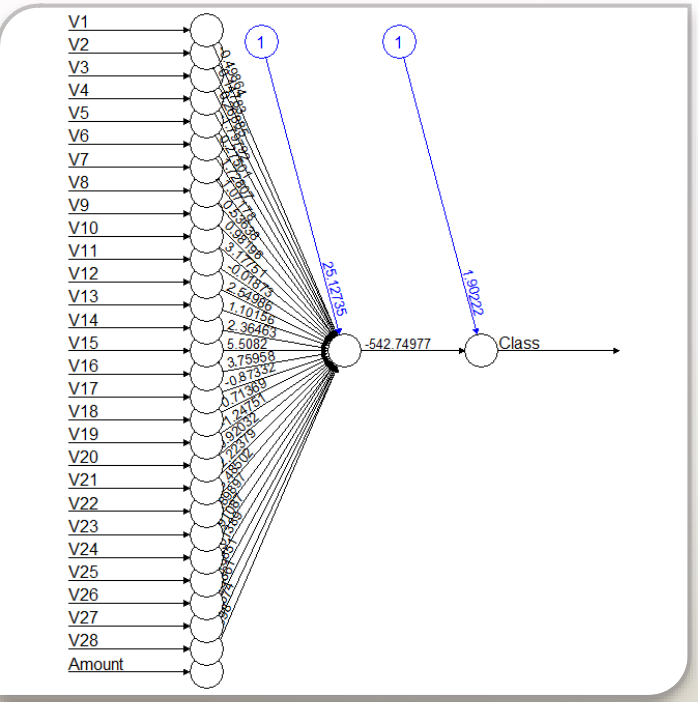
GRADIENT BOOSTING(GBM)

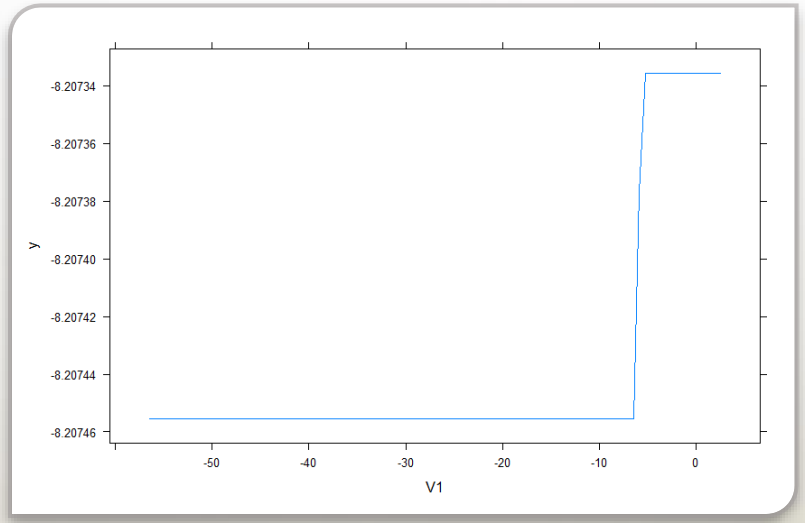
- Gradient Boosting is a well-known machine learning approach for classification and regression problems
- Several underlying ensemble models, such as weak decision trees, make up this model
- These decision trees are combined to produce a powerful gradient boosting model
- In our model, we use the gradient descent

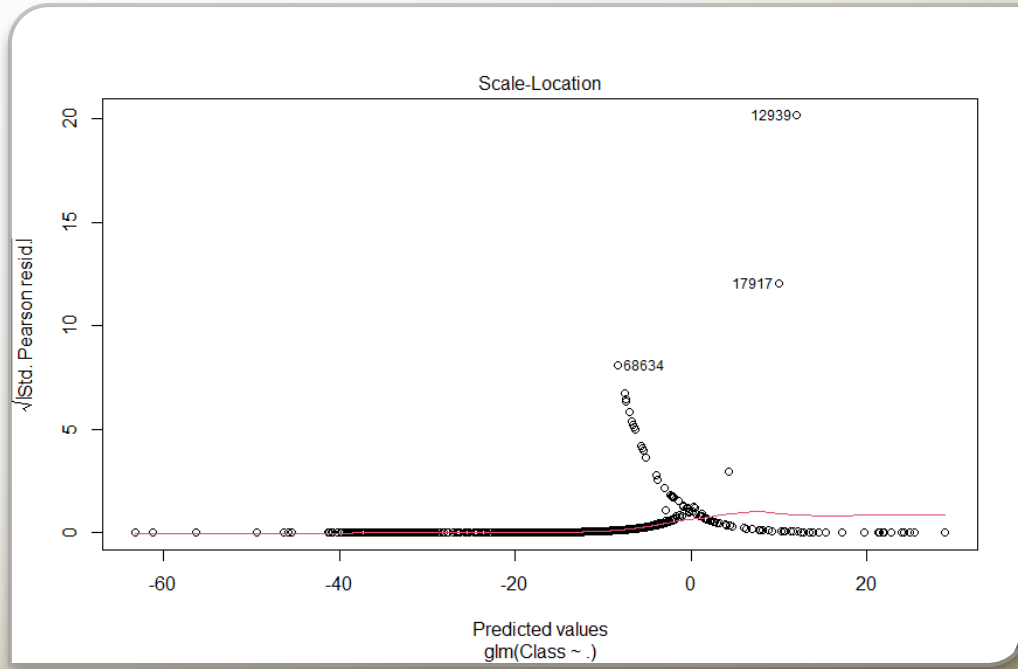
GRADIENT BOOSTING(GBM)

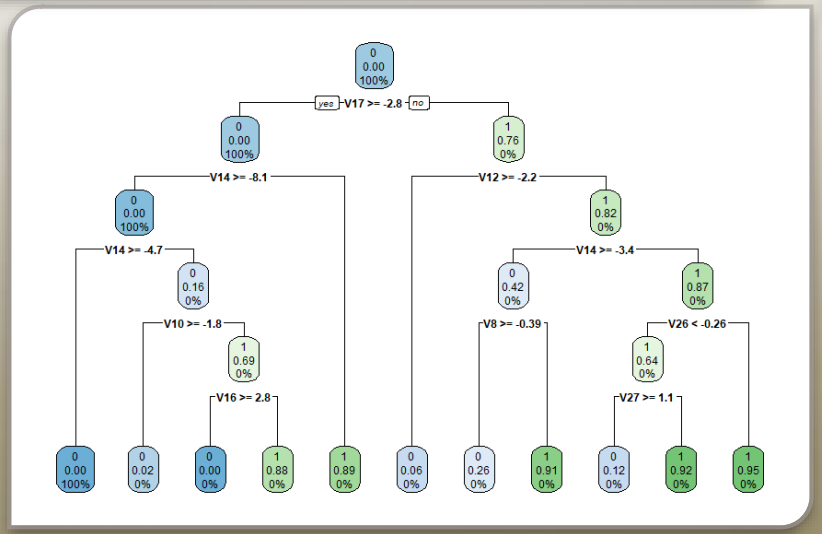


DATA VISUALIZATION









CONCLUSION

- We learned how to use machine learning to construct our credit card fraud detection model as part of our project
- We implemented this model using several machine learning algorithms and plotted the model's performance curves
- We learned how to analyze and interpret data in order to distinguish fraudulent transactions from other sorts of information

THANK YOU