Credit Card Fraud Detection

1. Load the creditcard.csv dataset

```
# read the dataset
credit_card <- read.csv('creditcard.csv')
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
1	0	-1.35980713	-0.072781173	2.53634674	1.37815522	-0.338320770	0.462387778	0.239598554	0.098697901
2	0	1.19185711	0.266150712	0.16648011	0.44815408	0.060017649	-0.082360809	-0.078802983	0.085101655
3	1	-1.35835406	-1.340163075	1.77320934	0.37977959	-0.503198133	1.800499381	0.791460956	0.247675787
4	1	-0.96627171	-0.185226008	1.79299334	-0.86329128	-0.010308880	1.247203168	0.237608940	0.377435875
5	2	-1.15823309	0.877736755	1.54871785	0.40303393	-0.407193377	0.095921462	0.592940745	-0.270532677
6	2	-0.42596588	0.960523045	1.14110934	-0.16825208	0.420986881	-0.029727552	0.476200949	0.260314333
7	4	1.22965763	0.141003507	0.04537077	1.20261274	0.191880989	0.272708123	-0.005159003	0.081212940
8	7	-0.64426944	1.417963545	1.07438038	-0.49219902	0.948934095	0.428118463	1.120631358	-3.807864239
9	7	-0.89428608	0.286157196	-0.11319221	-0.27152613	2.669598660	3.721818061	0.370145128	0.851084443
10	9	-0.33826175	1.119593376	1.04436655	-0.22218728	0.499360806	-0.246761101	0.651583206	0.069538587
11	10	1.44904378	-1.176338825	0.91385983	-1.37566665	-1.971383165	-0.629152139	-1.423235601	0.048455888
12	10	0.38497822	0.616109459	-0.87429970	-0.09401863	2.924584378	3.317027168	0.470454672	0.538247228
13	10	1.24999874	-1.221636809	0.38393015	-1.23489869	-1.485419474	-0.753230165	-0.689404975	-0.227487228
14	11	1.06937359	0.287722129	0.82861273	2.71252043	-0.178398016	0.337543730	-0.096716862	0.115981736
15	12	-2.79185477	-0.327770757	1.64175016	1.76747274	-0.136588446	0.807596468	-0.422911390	-1.907107476
16	12	-0.75241704	0.345485415	2.05732291	-1.46864330	-1.158393680	-0.077849829	-0.608581418	0.003603484
17	12	1.10321544	-0.040296215	1.26733209	1.28909147	-0.735997164	0.288069163	-0.586056786	0.189379714
18	13	-0.43690507	0.918966213	0.92459077	-0.72721905	0.915678718	-0.127867352	0.707641607	0.087962355
19	14	-5.40125766	-5.450147834	1.18630463	1.73623880	3.049105878	-1.763405574	-1.559737699	0.160841747

The dataset includes 31 columns from V1 to V28, Time, Amount and Class. Class is the most important column as it determines which transaction is fraudulent. V1 to V28 contains private information of the customers, which have already been converted to numerical variables by using PCA transformation. Time is when the fraudulent transaction occurs. Amount is how much money involves in the transaction.

2. Examine the structure of the dataset

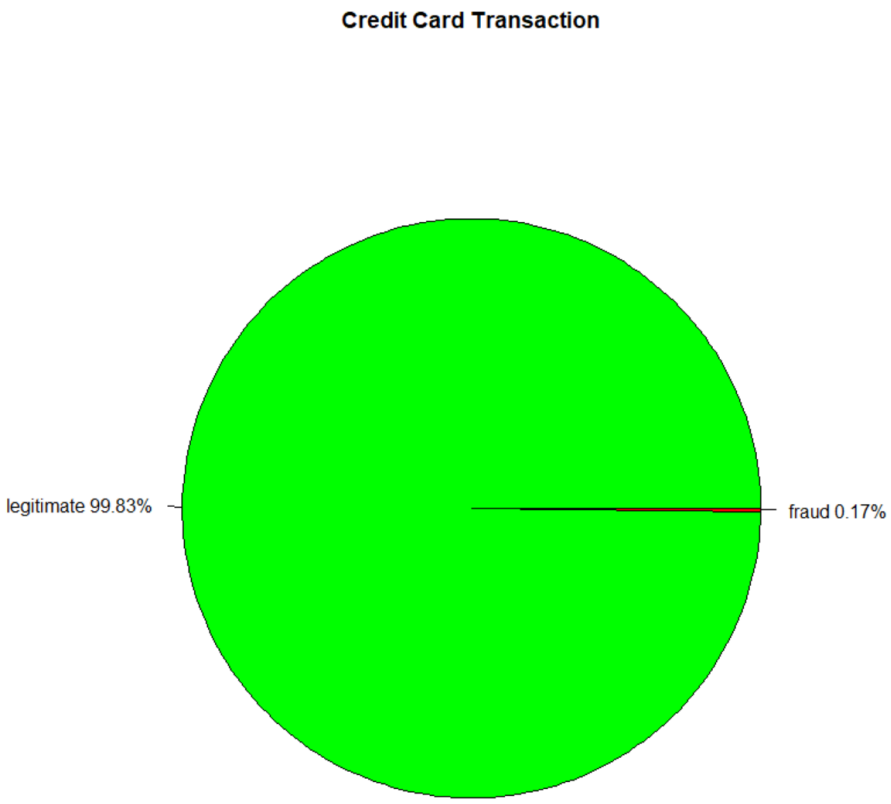
```
'data.frame':   284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
 $ 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 ...
```

3. The Class column has two values (0 and 1), convert this column to vector variable

```
     Time              V1                  V2                  V3                  V4                  V5
Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331
1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160
Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985   Median :  -0.05434
Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000
3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193
Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167
       V6                V7                  V8                  V9                 V10                 V11
Min.   :-26.1605   Min.   :-43.5572   Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747
1st Qu.: -0.7683   1st Qu.: -0.5541   1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249
Median : -0.2742   Median :  0.0401   Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276
Mean   :  0.0000   Mean   :  0.0000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000
3rd Qu.:  0.3986   3rd Qu.:  0.5704   3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959
Max.   : 73.3016   Max.   :120.5895   Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891
      V12                V13                 V14                 V15                 V16                 V17
Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280
1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375
Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807   Median :  0.06641   Median : -0.06568
Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000
3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968
Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353
      V18                V19                 V20                 V21                 V22
Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038   Min.   :-10.933144
1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839   1st Qu.: -0.542350
Median :-0.003636   Median : 0.003735   Median : -0.06248   Median : -0.02945   Median :  0.006782
Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.000000
3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638   3rd Qu.:  0.528554
Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284   Max.   : 10.503090
      V23                V24                 V25                 V26                 V27                 V28
Min.   :-44.80774   Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679   Min.   :-15.43008
1st Qu.: -0.16185   1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840   1st Qu.: -0.05296
Median : -0.01119   Median : 0.04098   Median :  0.01659   Median :-0.05214   Median :  0.001342   Median :  0.01124
Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000   Mean   :  0.00000
3rd Qu.:  0.14764   3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045   3rd Qu.:  0.07828
Max.   : 22.52841   Max.   : 4.58455   Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198   Max.   : 33.84781
     Amount          Class
Min.   :    0.00   0:284315
1st Qu.:    5.60   1:   492
Median :   22.00
Mean   :   88.35
3rd Qu.:   77.17
Max.   :25691.16
```

In the Class column, there are 284,315 legitimate transactions and 492 fraudulent transactions.

4. The number of missing values is 0. Therefore, the data is good to proceed.
5. Data visualization of the percentage of both legitimate and fraudulent transactions

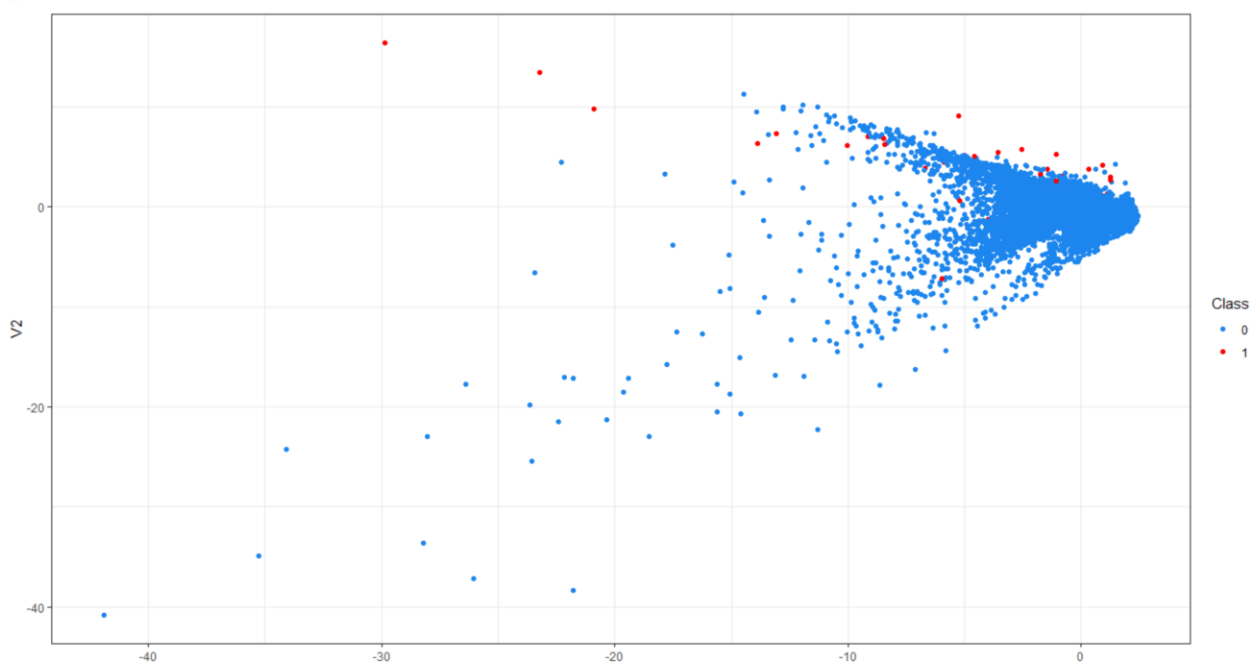


There is 99.83% legitimate transaction and 0.17% fraudulent transaction in this dataset.

6. Use a Confusion Matrix and statistics for the model prediction

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 284315    492
         1      0      0

               Accuracy : 0.9983
                 95% CI : (0.9981, 0.9984)
    No Information Rate : 0.9983
    P-Value [Acc > NIR] : 0.512

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.9983
         Neg Pred Value :    NaN
             Prevalence : 0.9983
         Detection Rate : 0.9983
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

7. Predict and visualize the frauds based on V1 and V2

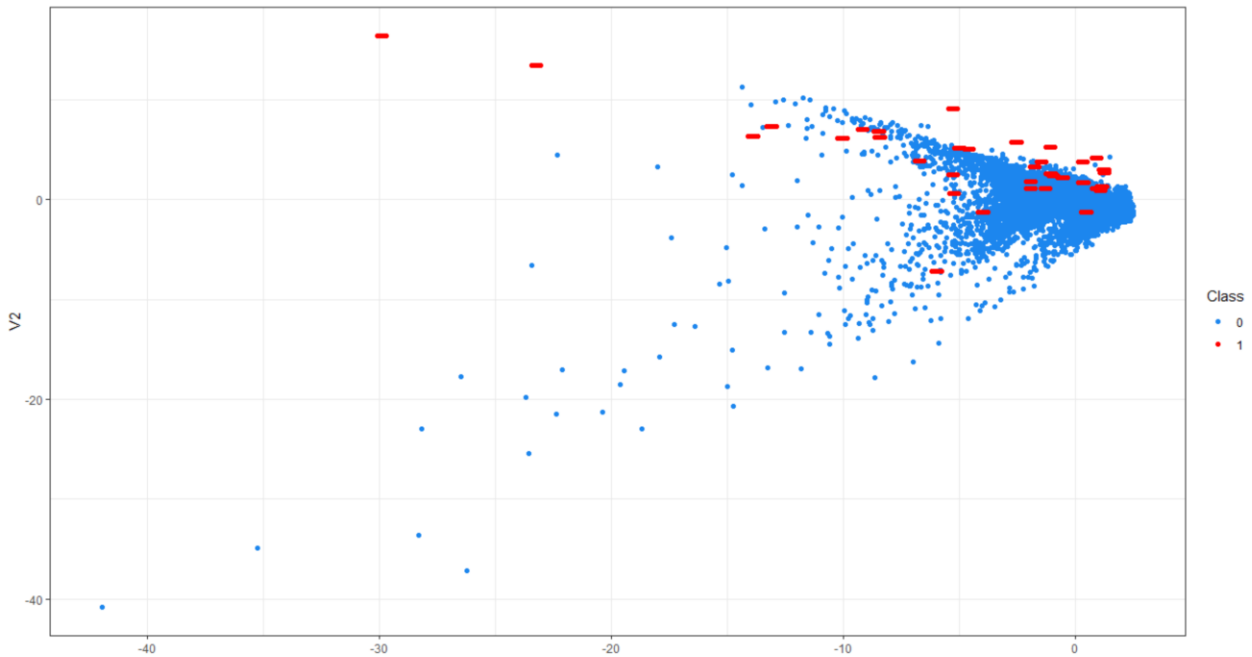


The red dots represent the fraudulent transactions. The blue dots represent legitimate transactions.

8. Split the dataset into training and test set. Training set takes 80% of the dataset, the test set takes 20% of the dataset.

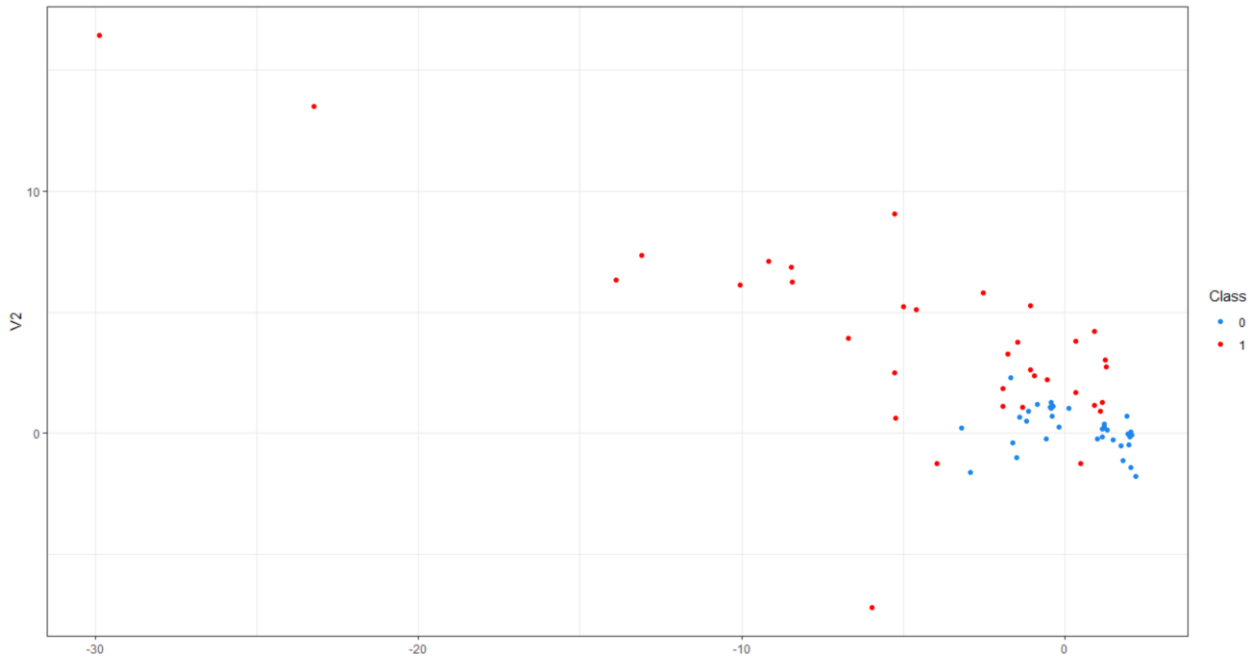
```
> dim(train_data)
[1] 22785    31
> dim(test_data)
[1] 5696   31
```

9. Method 1: Random Over-Sampling (ROS)



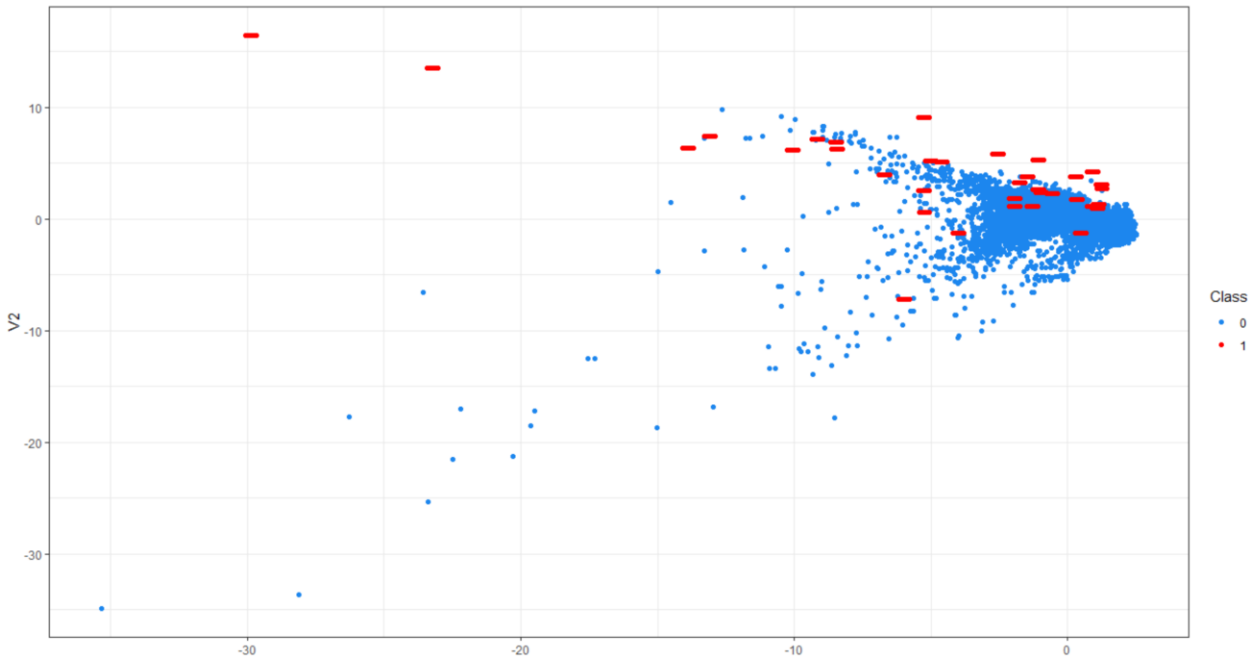
The result of ROS is fairly similar to the previous predicted result. However, due to the over-sampling, the red dots appear to be more spread-out.

10. Method 2: Random Under-Sampling (RUS)



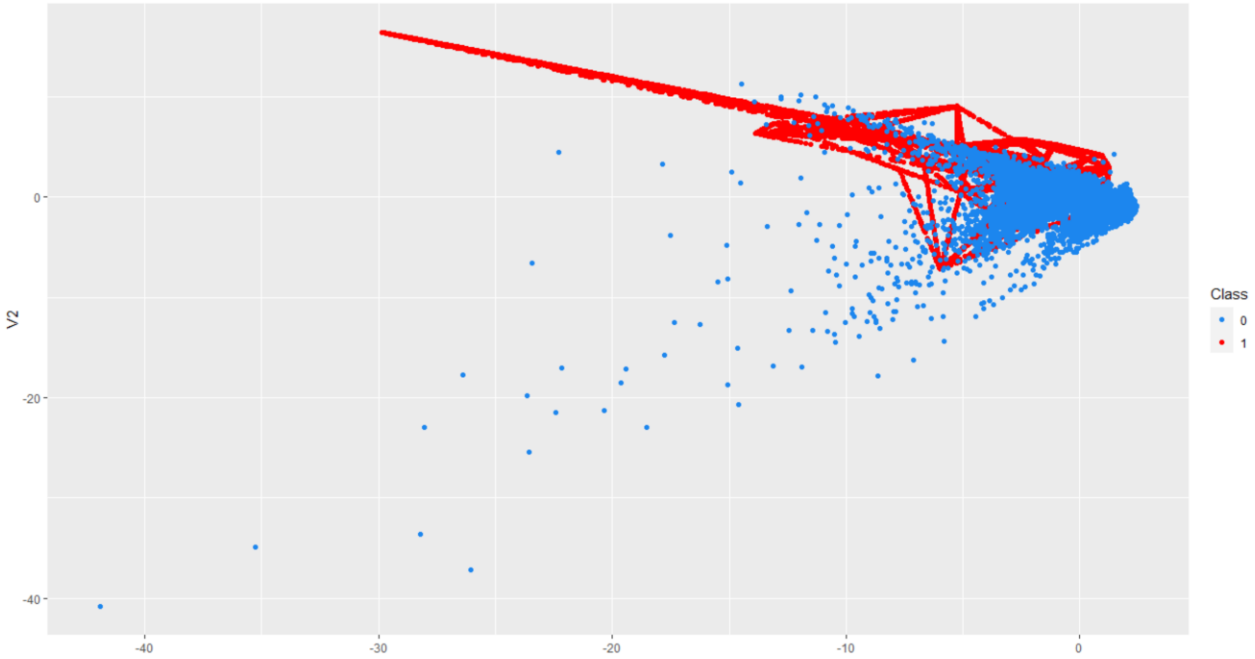
The result of RUS method is rather interesting as we see the majority of the transactions to be fraudulent. This is not conclusive and more investigations need to be done.

11. Method 3: Combination of ROS and RUS



There is some similarity between the third method and the ROS method.

12. Balance the dataset using the smotefamily package. This will be the fourth method to predict the fraudulent transactions.



Since this method boosts the amount of 'potential' fraudulent transactions up to 40%, we can see the amount of red dots has increased significantly.

13. Examine the predictions of the model and the test data

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5620    2
         1   67    7

               Accuracy : 0.9879
                 95% CI : (0.9847, 0.9906)
    No Information Rate : 0.9984
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1663

 Mcnemar's Test P-Value : 1.312e-14

            Sensitivity : 0.98822
            Specificity : 0.77778
         Pos Pred Value : 0.99964
         Neg Pred Value : 0.09459
             Prevalence : 0.99842
         Detection Rate : 0.98666
   Detection Prevalence : 0.98701
      Balanced Accuracy : 0.88300

       'Positive' Class : 0
```

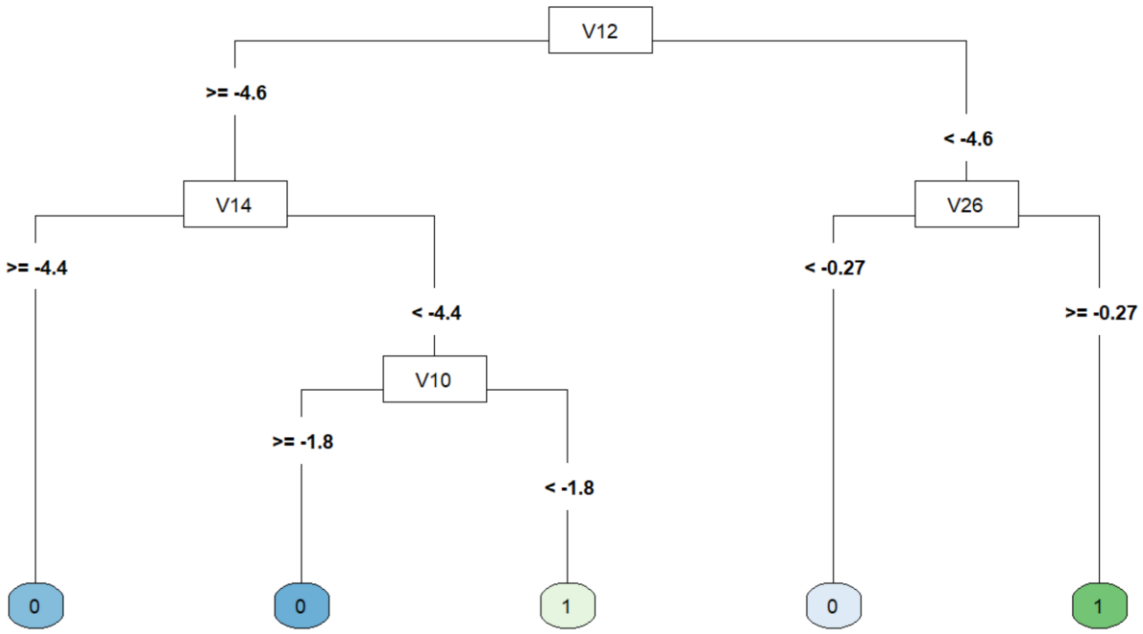
Out of 9 fraudulent cases, the model has correctly predicted 7. Out of 5,687 legitimate cases, the model has correctly predicted 5,620 cases. Overall, this is a pretty good result.

14. Examine the predictions of the model and the original data

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 28122     4
         1   315    40

               Accuracy : 0.9888
                 95% CI : (0.9875, 0.99)
    No Information Rate : 0.9985
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1983

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9889
            Specificity : 0.9091
         Pos Pred Value : 0.9999
         Neg Pred Value : 0.1127
             Prevalence : 0.9985
         Detection Rate : 0.9874
   Detection Prevalence : 0.9875
      Balanced Accuracy : 0.9490

       'Positive' Class : 0
```

The model has correctly predicted 40 cases out of 44 fraudulent cases, and 28,122 cases out of 28,437 cases. This is even better than the predictions for the test data

15. Build a decision tree without using SMOTE



16. Examine the prediction result without SMOTE for the test data

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5686    3
         1    1    6

               Accuracy : 0.9993
                 95% CI : (0.9982, 0.9998)
    No Information Rate : 0.9984
    P-Value [Acc > NIR] : 0.05483

                  Kappa : 0.7497

 Mcnemar's Test P-Value : 0.61708

            Sensitivity : 0.9998
            Specificity : 0.6667
         Pos Pred Value : 0.9995
         Neg Pred Value : 0.8571
             Prevalence : 0.9984
         Detection Rate : 0.9982
   Detection Prevalence : 0.9988
      Balanced Accuracy : 0.8332

       'Positive' Class : 0
```

The model only made 50% correct prediction for positive cases. This shows that the model has failed when it is built without using SMOTE.

17. Examine the result without using SMOTE for the original data

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 28433     9
         1     4    35

               Accuracy : 0.9995
                 95% CI : (0.9992, 0.9998)
    No Information Rate : 0.9985
    P-Value [Acc > NIR] : 3.99e-08

                  Kappa : 0.8431

 Mcnemar's Test P-Value : 0.2673

            Sensitivity : 0.9999
            Specificity : 0.7955
         Pos Pred Value : 0.9997
         Neg Pred Value : 0.8974
             Prevalence : 0.9985
         Detection Rate : 0.9983
   Detection Prevalence : 0.9986
      Balanced Accuracy : 0.8977

       'Positive' Class : 0
```

Again, the model without SMOTE shows much less accuracy than with SMOTE.

Overall, the credit card fraud detection works best when it is built using SMOTE. The accuracy is fairly high and acceptable; however, this model still needs more work before it can be implemented in production.