Credit Card Fraud Detection project

Presented by:-

Valiveti Manikanta Bhuvanesh-19BCD7088

Tarini Guttula -19BCE7758

Popuri Harsha Vardhan -19BCI7039

G.Balasubramanyam -19BCE7681

Rokkam Anirudh -19BCD7081

Guided by:-

Dr.GopiKrishnan

VIT-AP UNIVERSITY

ABSTRACT

- Credit card fraud has increased dramatically in recent months. It is, in fact, one of the most common threats facing the BFSI industry. The objective of this R project is to create a classifier capable of accurately detecting credit card fraud. The credit card transaction dataset with a mix of non-fraudulent and fraudulent transactions will be used for research. Decision trees, logistic regression, artificial neural networks and the gradient classifier will be used in the project. a fraudulent and non-fraudulent call by applying these ML algorithms.This project will show you how to classify data using machine learning techniques in a real context.

INTRODUCTION

ABOUT DATASET

- Transactions by European cardholders in September 2013 are included in this dataset. There are 492 fraudulent transactions out of a total of 2.84.807. As there are fewer fraud cases than transactions, the data is out of balance. The dataset has been transformed into a PCA transformation and includes only numeric values. For the sake of privacy and confidentiality, many basic details are hidden, simply leaving the converted PCA data.Only time and money are not converted into PCA; all other values provided (v1, v2, v3, v4, v5, v6, v7, v8, etc.) are PCA transformed numeric values. The fraudulent entity class has a value of 1 and the regular transaction has a value of 0.

DATASET

Time	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	Amount	Class
0	-1.3598	-0.0728	2.53635	1.37816	-0.3383	0.46239	0.2396	0.0987	0.36379	0.09079	-0.5516	-0.6178	-0.9914	-0.3112	1.46818	-0.4704	0.20797	0.02579	0.40399	0.25141	-0.0183	0.27784	-0.1105	0.06693	0.12854	-0.1891	0.13356	-0.0211	149.62	0
0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.0824	-0.0788	0.0851	-0.2554	-0.167	1.61273	1.06524	0.4891	-0.1438	0.63556	0.46392	-0.1148	-0.1834	-0.1458	-0.0691	-0.2258	-0.6387	0.10129	-0.3398	0.16717	0.12589	-0.009	0.01472	2.69	0
1	-1.3584	-1.3402	1.77321	0.37978	-0.5032	1.8005	0.79146	0.24768	-1.5147	0.20764	0.6245	0.06608	0.71729	-0.1659	2.34586	-2.8901	1.10997	-0.1214	-2.2619	0.52498	0.248	0.77168	0.90941	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
1	-0.9663	-0.1852	1.79299	-0.8633	-0.0103	1.2472	0.23761	0.37744	-1.387	-0.055	-0.2265	0.17823	0.50776	-0.2879	-0.6314	-1.0596	-0.6841	1.96578	-1.2326	-0.208	-0.1083	0.00527	-0.1903	-1.1756	0.64738	-0.2219	0.06272	0.06146	123.5	0
2	-1.1582	0.87774	1.54872	0.40303	-0.4072	0.09592	0.59294	-0.2705	0.81774	0.75307	-0.8228	0.5382	1.34585	-1.1197	0.17512	-0.4514	-0.237	-0.0382	0.80349	0.40854	-0.0094	0.79828	-0.1375	0.14127	-0.206	0.50229	0.21942	0.21515	69.99	0
2	-0.426	0.96052	1.14111	-0.1683	0.42099	-0.0297	0.4762	0.26031	-0.5687	-0.3714	1.34126	0.35989	-0.3581	-0.1371	0.51762	0.40173	-0.0581	0.06865	-0.0332	0.08497	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.10591	0.25384	0.08108	3.67	0
4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.0052	0.08121	0.46496	-0.0993	-1.4169	-0.1538	-0.7511	0.16737	0.05014	-0.4436	0.00282	-0.612	-0.0456	-0.2196	-0.1677	-0.2707	-0.1541	-0.7801	0.75014	-0.2572	0.03451	0.00517	4.99	0
7	-0.6443	1.41796	1.07438	-0.4922	0.94893	0.42812	1.12063	-3.8079	0.61537	1.24938	-0.6195	0.29147	1.75796	-1.3239	0.68613	-0.0761	-1.2221	-0.3582	0.3245	-0.1567	1.94347	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
7	-0.8943	0.28616	-0.1132	-0.2715	2.6696	3.72182	0.37015	0.85108	-0.392	-0.4104	-0.7051	-0.1105	-0.2863	0.07436	-0.3288	-0.2101	-0.4998	0.11876	0.57033	0.05274	-0.0734	-0.2681	-0.2042	1.01159	0.3732	-0.3842	0.01175	0.1424	93.2	0
9	-0.3383	1.11959	1.04437	-0.2222	0.49936	-0.2468	0.65158	0.06954	-0.7367	-0.3668	1.01761	0.83639	1.00684	-0.4435	0.15022	0.73945	-0.541	0.47668	0.45177	0.20371	-0.2469	-0.6338	-0.1208	-0.385	-0.0697	0.0942	0.24622	0.08308	3.68	0
10	1.44904	-1.1763	0.91386	-1.3757	-1.9714	-0.6292	-1.4232	0.04846	-1.7204	1.62666	1.19964	-0.6714	-0.5139	-0.095	0.23093	0.03197	0.25341	0.85434	-0.2214	-0.3872	-0.0093	0.31389	0.02774	0.50051	0.25137	-0.1295	0.04285	0.01625	7.8	0
10	0.38498	0.61611	-0.8743	-0.094	2.92458	3.31703	0.47045	0.53825	-0.5589	0.30976	-0.2591	-0.3261	-0.09	0.36283	0.9289	-0.1295	-0.81	0.35999	0.70766	0.12599	0.04992	0.23842	0.00913	0.99671	-0.7673	-0.4922	0.04247	-0.0543	9.99	0
10	1.25	-1.2216	0.38393	-1.2349	-1.4854	-0.7532	-0.6894	-0.2275	-2.094	1.32373	0.22767	-0.2427	1.20542	-0.3176	0.72567	-0.8156	0.87394	-0.8478	-0.6832	-0.1028	-0.2318	-0.4833	0.08467	0.39283	0.16113	-0.355	0.02642	0.04242	121.5	0
11	1.06937	0.28772	0.82861	2.71252	-0.1784	0.33754	-0.0967	0.11598	-0.2211	0.46023	-0.7737	0.32339	-0.0111	-0.1785	-0.6556	-0.1999	0.12401	-0.9805	-0.9829	-0.1532	-0.0369	0.07441	-0.0714	0.10474	0.54826	0.10409	0.02149	0.02129	27.5	0
12	-2.7919	-0.3278	1.64175	1.76747	-0.1366	0.8076	-0.4229	-1.9071	0.75571	1.15109	0.84456	0.79294	0.37045	-0.735	0.4068	-0.3031	-0.1559	0.77827	2.22187	-1.5821	1.15166	0.22218	1.02059	0.02832	-0.2327	-0.2356	-0.1648	-0.0302	58.8	0
12	-0.7524	0.34549	2.05732	-1.4686	-1.1584	-0.0778	-0.6086	0.0036	-0.4362	0.74773	-0.794	-0.7704	1.04763	-1.0666	1.10695	1.66011	-0.2793	-0.42	0.43254	0.26345	0.49962	1.35365	-0.2566	-0.0651	-0.0391	-0.0871	-0.181	0.12939	15.99	0
12	1.10322	-0.0403	1.26733	1.28909	-0.736	0.28807	-0.5861	0.18938	0.78233	-0.268	-0.4503	0.93671	0.70838	-0.4686	0.35457	-0.2466	-0.0092	-0.5959	-0.5757	-0.1139	-0.0246	0.196	0.0138	0.10376	0.3643	-0.3823	0.09281	0.03705	12.99	0
13	-0.4369	0.91897	0.92459	-0.7272	0.91568	-0.1279	0.70764	0.08796	-0.6653	-0.738	0.3241	0.27719	0.25262	-0.2919	-0.1845	1.14317	-0.9287	0.68047	0.02544	-0.047	-0.1948	-0.6726	-0.1569	-0.8884	-0.3424	-0.049	0.07969	0.13102	0.89	0
14	-5.4013	-5.4501	1.1863	1.73624	3.04911	-1.7634	-1.5597	0.16084	1.23309	0.34517	0.91723	0.97012	-0.2666	-0.4791	-0.5266	0.472	-0.7255	0.07508	-0.4069	-2.1968	-0.5036	0.98446	2.45859	0.04212	-0.4816	-0.6213	0.39205	0.94959	46.8	0
15	1.49294	-1.0293	0.45479	-1.438	-1.5554	-0.721	-1.0807	-0.0531	-1.9787	1.63808	1.07754	-0.632	-0.417	0.05201	-0.043	-0.1664	0.30424	0.55443	0.05423	-0.3879	-0.1776	-0.1751	0.04	0.29581	0.33293	-0.2204	0.0223	0.0076	5	0
16	0.69488	-1.3618	1.02922	0.83416	-1.1912	1.30911	-0.8786	0.44529	-0.4462	0.56852	1.01915	1.29833	0.42048	-0.3727	-0.808	-2.0446	0.51566	0.62585	-1.3004	-0.1383	-0.2956	-0.572	-0.0509	-0.3042	0.072	-0.4222	0.08655	0.0635	231.71	0
17	0.9625	0.32846	-0.1715	2.1092	1.12957	1.69604	0.10771	0.5215	-1.1913	0.7244	1.69033	0.40677	-0.9364	0.98374	0.71091	-0.6022	0.40248	-1.7372	-2.0276	-0.2693	0.144	0.40249	-0.0485	-1.3719	0.39081	0.19996	0.01637	-0.0146	34.09	0
18	1.16662	0.50212	-0.0673	2.26157	0.4288	0.08947	0.24115	0.13808	-0.9892	0.92217	0.74479	-0.5314	-2.1053	1.12687	0.00308	0.42442	-0.4545	-0.0989	-0.8166	-0.3072	0.0187	-0.062	-0.1039	-0.3704	0.6032	0.10856	-0.0405	-0.0114	2.28	0
18	0.24749	0.27767	1.18547	-0.0926	-1.3144	-0.1501	-0.9464	-1.6179	1.54407	-0.8299	-0.5832	0.52493	-0.4534	0.08139	1.5552	-1.3969	0.78313	0.43662	2.17781	-0.231	1.65018	0.20045	-0.1854	0.42307	0.82059	-0.2276	0.33663	0.25048	22.75	0
22	-1.9465	-0.0449	-0.4056	-1.0131	2.94197	2.95505	-0.0631	0.85555	0.04997	0.57374	-0.0813	-0.2157	0.04416	0.0339	1.19072	0.57884	-0.9757	0.04406	0.4886	-0.2167	-0.5795	-0.7992	0.8703	0.98342	0.3212	0.14965	0.70752	0.0146	0.89	0
22	-2.0743	-0.1215	1.32202	0.41001	0.2952	-0.9595	0.54399	-0.1046	0.47566	0.14945	-0.8566	-0.1805	-0.6552	-0.2798	-0.2117	-0.3333	0.01075	-0.4885	0.50575	-0.3867	-0.4036	-0.2274	0.74243	0.39853	0.24921	0.2744	0.35997	0.24323	26.43	0
23	1.17328	0.3535	0.28391	1.13356	-0.1726	-0.9161	0.36902	-0.3273	-0.2467	-0.0461	-0.1434	0.97935	1.49229	0.10142	0.76148	-0.0146	-0.5116	-0.3251	-0.3909	0.02788	0.067	0.22781	-0.1505	0.43505	0.72482	-0.3371	0.01637	0.03004	41.88	0
23	1.32271	-0.174	0.43456	0.57604	-0.8368	-0.8311	-0.2649	-0.221	-1.0714	0.86856	-0.6415	-0.1113	0.36149	0.17195	0.78217	-1.3559	-0.2169	1.27177	-1.2406	-0.523	-0.2844	-0.3234	-0.0377	0.34715	0.55964	-0.2802	0.04234	0.02882	16	0
23	-0.4143	0.90544	1.72745	1.47347	0.00744	-0.2003	0.74023	-0.0292	-0.5934	-0.3462	-0.0121	0.7868	0.63595	-0.0863	0.0768	-1.4059	0.77559	-0.9429	0.54397	0.09731	0.07724	0.45733	-0.0385	0.64252	-0.1839	-0.2775	0.18269	0.15266	33	0
23	1.05939	-0.1753	1.26613	1.18611	-0.786	0.57844	-0.7671	0.40105	0.6995	-0.0647	1.04829	1.00562	-0.542	-0.0399	-0.2187	0.00448	-0.1936	0.04239	-0.2778	-0.178	0.01368	0.21373	0.01446	0.00295	0.29464	-0.3951	0.08146	0.02422	12.99	0
24	1.23743	0.06104	0.38053	0.76156	-0.3598	-0.4941	0.00649	-0.1339	0.43881	-0.2074	-0.9292	0.52711	0.34868	-0.1525	-0.2184	-0.1916	-0.1166	-0.6338	0.34842	-0.0664	-0.2457	-0.5309	-0.0443	0.07917	0.50914	0.28886	-0.0227	0.01184	17.28	0
25	1.11401	0.08555	0.4937	1.33576	-0.3002	-0.0108	-0.1188	0.18862	0.20569	0.08226	1.13356	0.6267	-1.4928	0.52079	-0.6746	-0.5291	0.15826	-0.3988	-0.1457	-0.2738	-0.0532	-0.0048	-0.0315	0.19805	0.56501	-0.3377	0.02906	0.00445	4.45	0
26	-0.5299	0.87389	1.34725	0.14546	0.41421	0.10022	0.71121	0.17607	-0.2867	-0.4847	0.87249	0.85164	-0.5717	0.10097	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.04695	0.2081	-0.1855	0.00103	0.09882	-0.5529	-0.0733	0.02331	6.14	0
26	-0.5299	0.87389	1.34725	0.14546	0.41421	0.10022	0.71121	0.17607	-0.2867	-0.4847	0.87249	0.85164	-0.5717	0.10097	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.04695	0.2081	-0.1855	0.00103	0.09882	-0.5529	-0.0733	0.02331	6.14	0

PROCEDURE

Step1:Loading dataset

- We import datasets containing credit card transactions

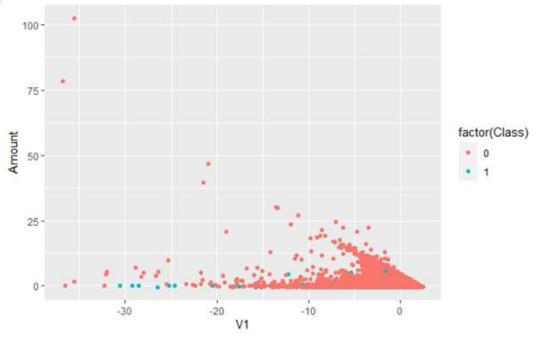
```
> df=read.csv("creditcard.csv")
> head(df)
  Time         V1          V2        V3         V4          V5          V6          V7          V8         V9         V10        V11
1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995
2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267
3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015
4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873
5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429
6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620
         V12        V13        V14        V15        V16         V17         V18         V19         V20          V21          V22
1 -0.61780086 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296  0.25141210 -0.018306778  0.277837576
2  1.06523531  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304 -0.06908314 -0.225775248 -0.638671953
3  0.06608369  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185710  0.52497973  0.247998153  0.771679402
4  0.17822823  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197 -0.20803778 -0.108300452  0.005273597
5  0.53819555  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692  0.40854236 -0.009430697  0.798278495
6  0.35989384 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379  0.08496767 -0.208253515 -0.559824796
         V23         V24        V25        V26         V27         V28 Amount Class
1 -0.11047391  0.06692807  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
2  0.10128802 -0.33984648  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
3  0.90941226 -0.68928096 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
4 -0.19032052 -1.17557533  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
5 -0.13745808  0.14126698 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
6 -0.02639767 -0.37142658 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

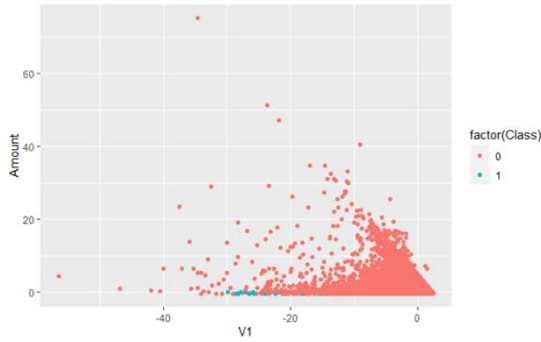
Step 2:Data Exploration

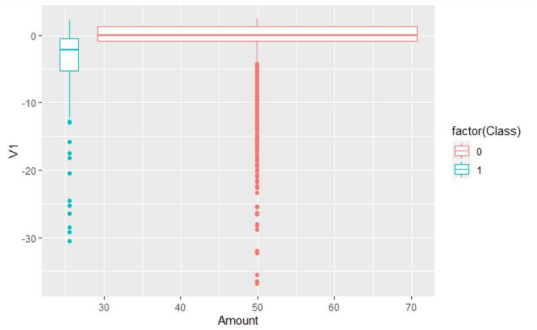
- In this step of the fraud detection ML project, we are able to discover the information this is contained withinside the credit score card information. We will continue through showing the credit score card information the usage of the head() feature in addition to the tail() feature. We will then continue to discover the opposite additives of this data frame.

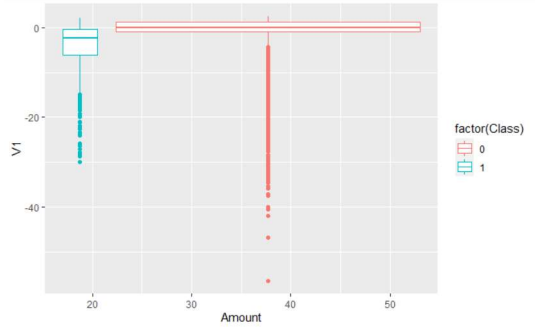
```
> dim(df)
[1] 284807     31
> colnames(df)
 [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"     "V8"     "V9"     "V10"    "V11"    "V12"    "V13"    "V14"    "V15"
[17] "V16"    "V17"    "V18"    "V19"    "V20"    "V21"    "V22"    "V23"    "V24"    "V25"    "V26"    "V27"    "V28"    "Amount" "Class"
> table(df$Class)

     0      1
284315    492
> summary(df)
      Time              V1                  V2                  V3                  V4                  V5                  V6
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985   Median :  -0.05434   Median : -0.2742
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000
 3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016
       V7                  V8                  V9                 V10                 V11                 V12                 V13
 Min.   :-43.5572   Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188
 1st Qu.: -0.5541   1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854
 Median :  0.0401   Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276   Median :  0.1400   Median :-0.01357
 Mean   :  0.0000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000
 3rd Qu.:  0.5704   3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251
 Max.   :120.5895   Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688
       V14                 V15                 V16                 V17                 V18                 V19                 V20
 Min.   :-19.2143   Min.   :-4.49894   Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772
 1st Qu.: -0.4256   1st Qu.:-0.58288   1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172
 Median :  0.0506   Median : 0.04807   Median :  0.06641   Median : -0.06568   Median :-0.003636   Median : 0.003735   Median : -0.06248
 Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000
 3rd Qu.:  0.4931   3rd Qu.: 0.64882   3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304
 Max.   : 10.5268   Max.   : 8.87774   Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090
       V21                 V22                 V23                 V24                 V25                 V26                 V27
 Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679
 1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840
 Median : -0.02945   Median :  0.006782   Median : -0.01119   Median : 0.04098   Median :  0.01659   Median :-0.05214   Median :  0.001342
 Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000
 3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045
 Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455   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
> var(df$Amount)
[1] 62560.07
> sd(df$Amount)
[1] 250.1201
>
```

```
> names(df)
 [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"     "V8"     "V9"
[11] "V10"    "V11"    "V12"    "V13"    "V14"    "V15"    "V16"    "V17"    "V18"    "V19"
[21] "V20"    "V21"    "V22"    "V23"    "V24"    "V25"    "V26"    "V27"    "V28"    "Amount"
[31] "Class"
> str(df)
'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 [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 ...
> class(df)
[1] "data.frame"
> ggplot(test,aes(x=V1,y=Amount))+geom_point(aes(color=factor(Class)))
> ggplot(train,aes(x=V1,y=Amount))+geom_point(aes(color=factor(Class)))
> ggplot(test,aes(x=Amount,y=V1))+geom_boxplot(aes(color=factor(Class)))
> ggplot(train,aes(x=Amount,y=V1))+geom_boxplot(aes(color=factor(Class)))
>
```









Step 3 :Data Manipulation

- In this step, we'll scale our data using the scale () function. We will apply this to the amount component of the amount of our credit card information. Scaling is also known as feature normalization. Using scaling, data is structured according to a specified range. Therefore, there are no extreme values in our dataset that could interfere with the functioning of our model.

```
> df$Amount=scale(df$Amount)
> data=df[,-c(1)]
> head(data)
         V1          V2        V3         V4          V5          V6          V7          V8         V9         V10        V11         V12        V13
1 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086 -0.9913898
2  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531  0.4890950
3 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369  0.7172927
4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823  0.5077569
5 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555  1.3458516
6 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384 -0.3580907
         V14        V15        V16         V17         V18         V19         V20          V21          V22         V23         V24         V25
1 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807  0.1285394
2 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704
3 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418
4 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760
5 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096
6 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938
         V26          V27         V28      Amount Class
1 -0.1891148  0.133558377 -0.02105305  0.24496383     0
2  0.1258945 -0.008983099  0.01472417 -0.34247394     0
3 -0.1390966 -0.055352794 -0.05975184  1.16068389     0
4 -0.2219288  0.062722849  0.06145763  0.14053401     0
5  0.5022922  0.219422230  0.21515315 -0.07340321     0
6  0.1059148  0.253844225  0.08108026 -0.33855582     0
```

Step 4 :Data Modeling

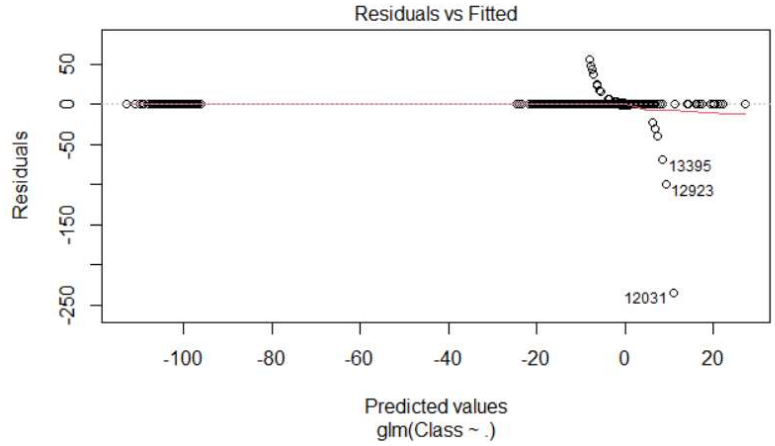
- Once we have standardized the dataset, we will divide our dataset into training sets and test sets with a divide ratio of 0.80. This means that 80% of our data will be attributed to train data while 20% will be attributed to test data.

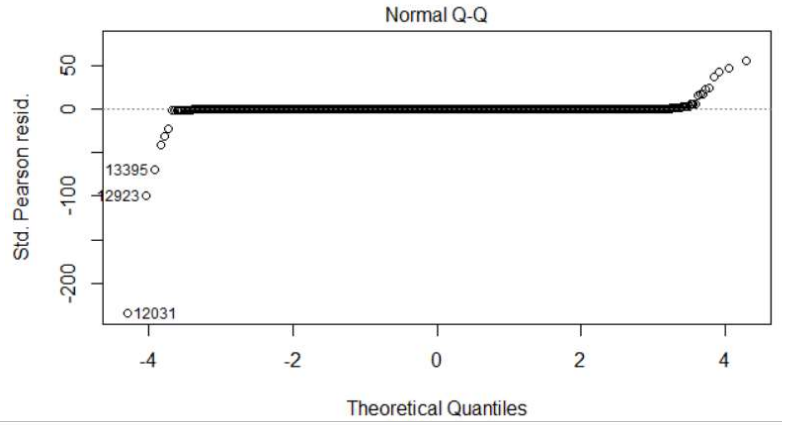
```
> sample = sample.split(data$Class,SplitRatio=0.80)
> train= subset(data,sample==TRUE)
> test= subset(data,sample==FALSE)
> dim(train)
[1] 227846     30
> dim(test)
[1] 56961    30
>
```

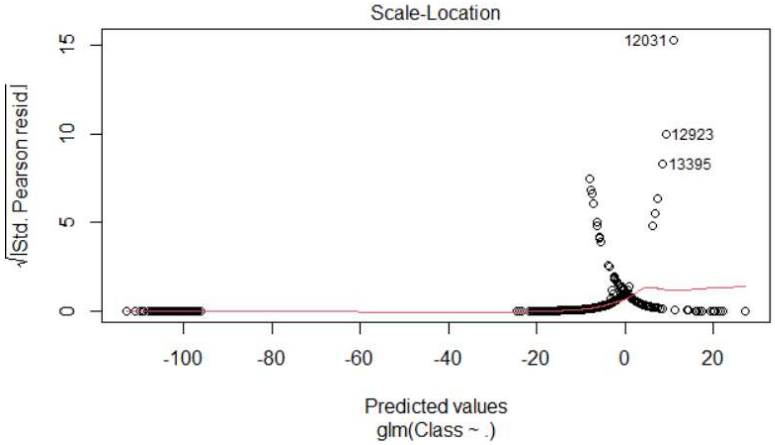
Fitting Logistic Regression Model

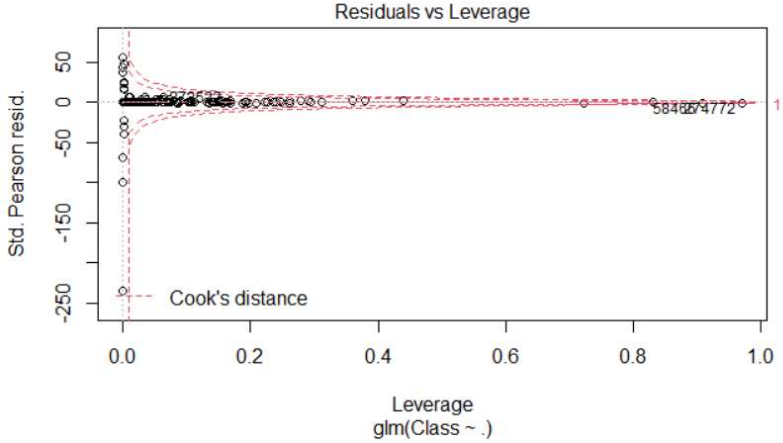
- In this section , we will adapt our first model. We will start with the logistic regression. Logistic regression is used to model the probability of a class outcome as pass / fail, pass / fail, and in our case - fraud / non-fraud. We are implementing this model on our test data.

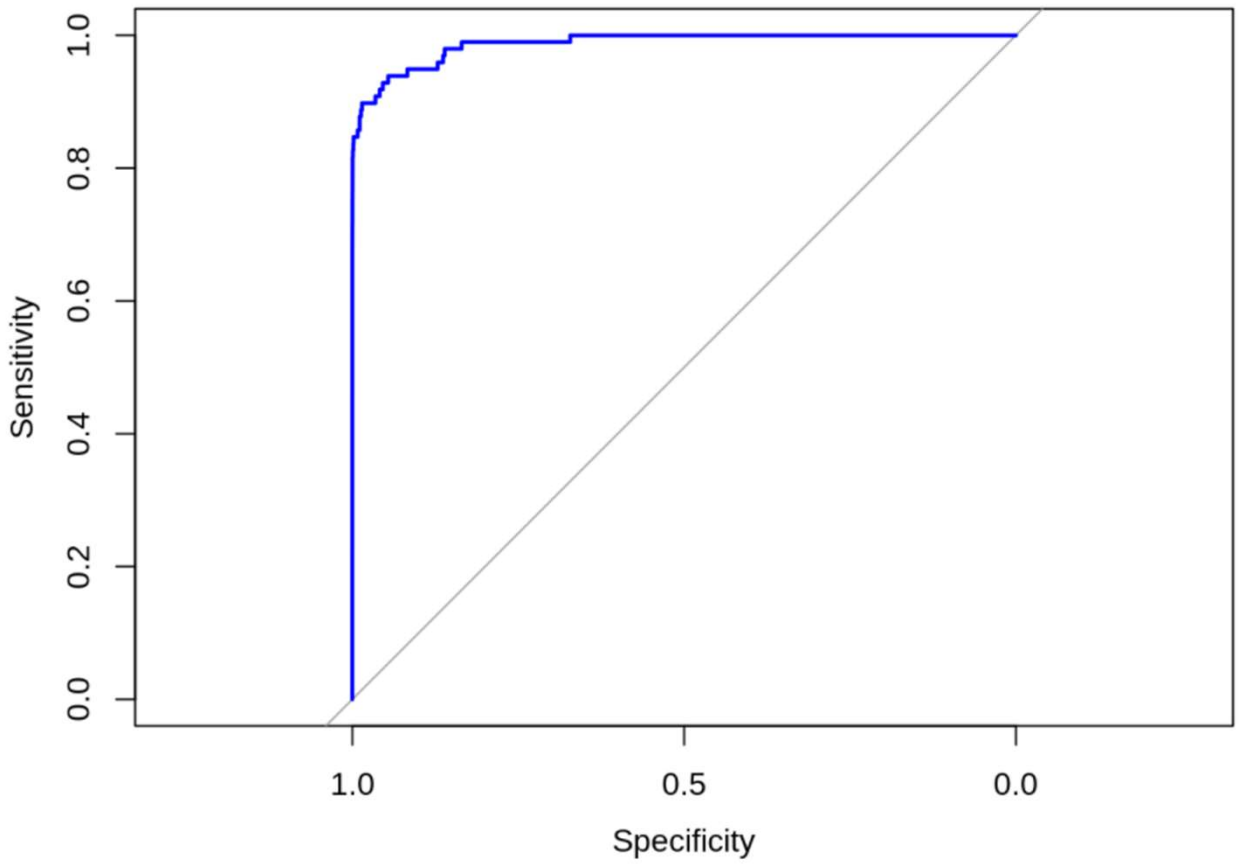
```
> Logistic_regression=glm(Class~.,test,family=binomial())
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(Logistic_regression)

Call:
glm(formula = Class ~ ., family = binomial(), data = test)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6721  -0.0243  -0.0138  -0.0059   4.0076

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -23.35723   10.46345  -2.232   0.0256 *
V1           -1.58484    1.27413  -1.244   0.2136
V2            6.08293    4.30739   1.412   0.1579
V3           -0.12974    0.24262  -0.535   0.5928
V4           10.51008    7.24934   1.450   0.1471
V5            5.29508    3.86330   1.371   0.1705
V6           -0.28563    0.27563  -1.036   0.3001
V7            5.47408    4.24678   1.289   0.1974
V8            0.01824    0.19476   0.094   0.9254
V9           11.69057    8.77347   1.332   0.1827
V10          -9.64105    6.71795  -1.435   0.1513
V11          -0.33408    0.29440  -1.135   0.2565
V12           9.30174    6.66803   1.395   0.1630
V13          -2.23434    1.30503  -1.712   0.0869 .
V14           3.75140    3.31695   1.131   0.2581
V15           3.74282    2.94050   1.273   0.2031
V16         -10.46380    7.19279  -1.455   0.1457
V17          -6.73060    5.06457  -1.329   0.1839
V18          11.09331    8.22161   1.349   0.1772
V19          -6.66437    4.83521  -1.378   0.1681
V20          -2.10079    1.19215  -1.762   0.0780 .
V21          -2.50719    2.02720  -1.237   0.2162
V22          -6.48370    5.26233  -1.232   0.2179
V23           0.97618    0.65580   1.489   0.1366
V24           0.74325    0.53334   1.394   0.1634
V25          -2.44598    2.01639  -1.213   0.2251
V26          13.01188    9.42821   1.380   0.1676
V27          -1.78501    0.79534  -2.244   0.0248 *
V28          -0.36543    0.36923  -0.990   0.3223
```


Residuals vs Fitted
Residuals
Predicted values
glm(Class ~ .)
13395
12923
12031


Normal Q-Q
Std. Pearson resid.
Theoretical Quantiles
13395
12923
12031

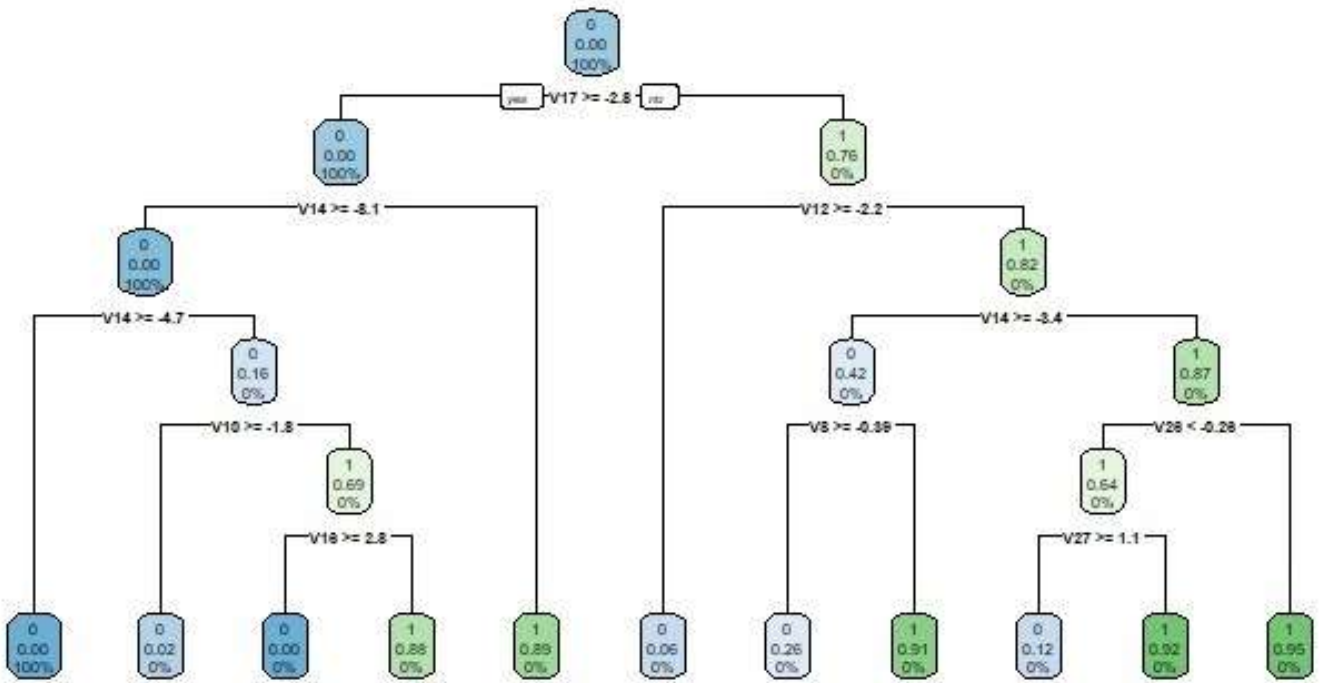

Scale-Location
√|Std. Pearson resid|
Predicted values
glm(Class ~ .)
12031
12923
13395


Residuals vs Leverage
Std. Pearson resid.
Leverage
glm(Class ~ .)
Cook's distance


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

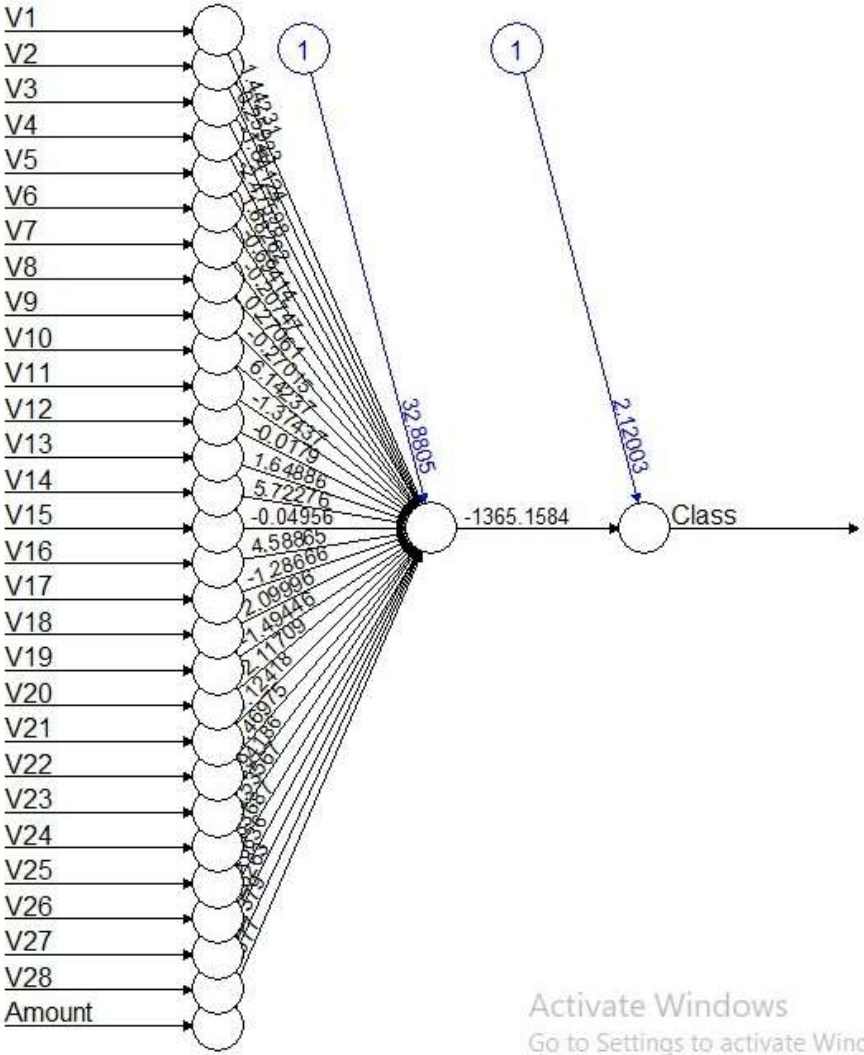
Fitting a Decision Tree Model

- In this section, we will implement a decision tree algorithm. Decision trees to track the results of a decision. These results are essentially a consequence by which we can conclude to which class the object belongs. We will now implement our decision tree model and plot it using the rpart.plot () function. We will specifically use recursive division to plot the decision tree.


0
0.00
100%
yes
V17 >= -2.8
no
0
0.00
100%
1
0.76
0%
V14 >= -8.1
V12 >= -2.2
0
0.00
100%
1
0.82
0%
V14 >= -4.7
V14 >= -3.4
0
0.16
0%
0
0.42
0%
1
0.87
0%
V10 >= -1.8
V8 >= -0.39
V28 < -0.28
1
0.69
0%
1
0.64
0%
V18 >= 2.8
V27 >= 1.1
0
0.00
100%
0
0.02
0%
0
0.00
0%
1
0.88
0%
1
0.89
0%
0
0.06
0%
0
0.26
0%
1
0.91
0%
0
0.12
0%
1
0.92
0%
1
0.95
0%

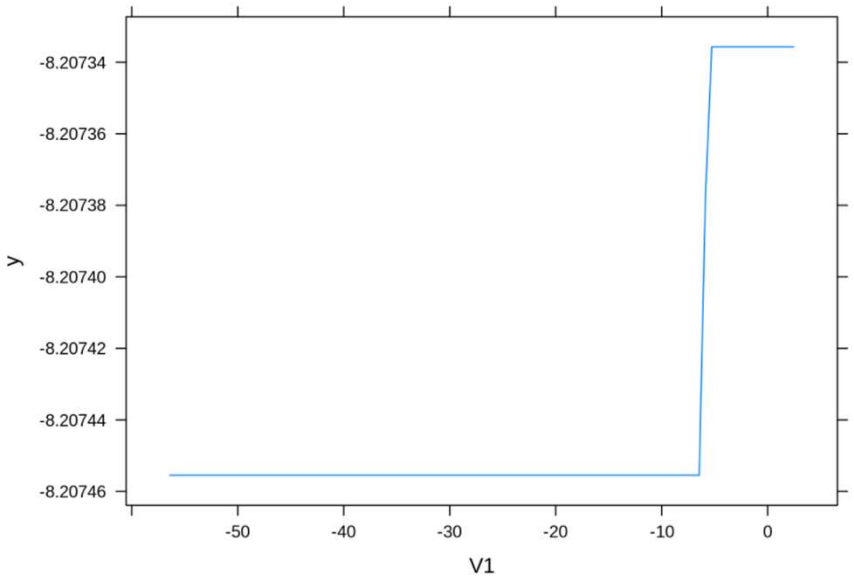
Fitting a Artificial Neural Network Model

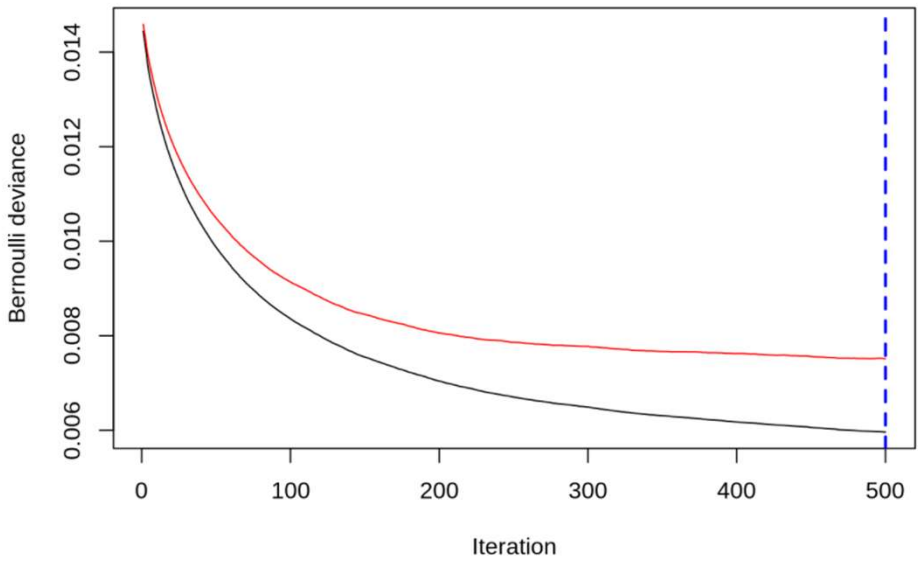
- We import the neuralnet package that would allow us to implement our ANNs. We therefore proceeded to plot it using the plot () function. Now, in the case of artificial neural networks, there is a range of values between 1 and 0. We set a threshold at 0.5, that is, values greater than 0.5 will correspond to 1 and the rest will be 0.



Fitting a Gradient Boosting Model

- Gradient Boosting is a popular machine learning algorithm used to perform classification and regression tasks. This model includes several underlying ensemble models such as weak decision trees. These decision trees combine to form a strong gradient augmentation model. We will implement the gradient descent algorithm in our model


-8.20734
-8.20736
-8.20738
-8.20740
-8.20742
-8.20744
-8.20746
y
-50
-40
-30
-20
-10
0
V1


0.006
0.008
0.010
0.012
0.014
Bernoulli deviance
0
100
200
300
400
500
Iteration

Summary

- Finally, we learned how to use machine learning to develop our credit card fraud detection model. We implemented this model using a variety of ML algorithms and plotted the models' respective performance curves. We learned how to analyse and visualise data in order to distinguish fraudulent transactions from other types of data.