Mini Project: Supervised Machine Learning

Model Report

CART, Random Forest, Neural Network

Table of Contents

1 Project Objective

The data-set provides details from MyBank about a Personal Loan Campaign that was executed by the bank. 20000 customers were targeted with an offer of personal loan on 10% interest rate, out of which 2512 customers responded positively. The data needs to be used to create classification model(s) in order to predict the response of new set of customers in the future, depending on the attributes available in the data.

Students are expected to build classification Models using following Supervised Machine Learning Techniques:

1. Classification and Regression Tree
2. Random Forest
3. Artificial Neural Network

Once the Classification Models are built, compare the three models with each other and provide your observations / recommendations.

2 Database overview

Details about the Personal Loan Campaign Dataset are as follows:

Sr. No.	Feature Name	Feature Type	Description
1	CUST_ID	Factor	Unique Customer ID
2	TARGET	Factor	Target Field - 1: Responder, 0: Non-Responder
3	AGE	Integer	Customer Age in years
4	GENDER	Factor	Gender: Male / Female / Other
5	BALANCE	Numeric	Monthly Average Balance
6	OCCUPATION	Factor	Occupation: Professional / Salaried / Self Employed / Self Employed Non Professional.
7	AGE_BKT	Factor	Age Bucket
8	SCR	Integer	Marketing Score
9	HOLDING_PERIOD	Integer	Duration in days to hold the money
10	ACC_TYPE	Factor	Account Type: Current Account / Saving Account
11	ACC_OP_DATE	Date	Account Open Date
12	LEN_OF_RLTN_IN_MNTH	Integer	Length of Relationship in Months
13	NO_OF_L_CR_TXNS	Integer	Number of Credit Transactions
14	NO_OF_L_DR_TXNS	Integer	Number of Debit Transactions
15	TOT_NO_OF_L_TXNS	Integer	Total Number of Transactions
16	NO_OF_BR_CSH_WDL_DR_TXNS	Integer	Branch Cash Withdrawal Debit Transactions
17	NO_OF_ATM_DR_TXNS	Integer	Number of ATM Debit Transactions

Sr. No.	Feature Name	Feature Type	Description
18	NO_OF_NET_DR_TXNS	Integer	Number of Net Banking Debit Transactions
19	NO_OF_MOB_DR_TXNS	Integer	Number of Mobile Banking Debit Transactions
20	NO_OF_CHQ_DR_TXNS	Integer	Number of Cheque Debit Transactions
21	FLG_HAS_CC	Integer	Has Credit Card - 1: Yes, 0: No
22	AMT_ATM_DR	Integer	Amount Withdrawn from ATM
23	AMT_BR_CSH_WDL_DR	Integer	Amount cash withdrawn from Branch
24	AMT_CHQ_DR	Integer	Amount debited by Cheque Transactions
25	AMT_NET_DR	Numeric	Amount debited by Net Transactions
26	AMT_MOB_DR	Integer	Amount debited by Mobile Transactions
27	AMT_L_DR	Numeric	Total Amount Debited
28	FLG_HAS_ANY_CHGS	Integer	Flag: Has any banking charges
29	AMT_OTH_BK_ATM_USG_CHGS	Integer	Charges for using Other Bank ATM
30	AMT_MIN_BAL_NMC_CHGS	Integer	Charges for not maintaining Min Balance
31	NO_OF_IW_CHQ_BNC_TXNS	Integer	Charges for Inward Cheque Bounce
32	NO_OF_OW_CHQ_BNC_TXNS	Integer	Charges for Outward Cheque Bounce
33	AVG_AMT_PER_ATM_TXN	Numeric	Average Amount per ATM Transaction
34	AVG_AMT_PER_CSH_WDL_TXN	Numeric	Average Amount Per Cash Withdrawal
35	AVG_AMT_PER_CHQ_TXN	Numeric	Average Amount Per Cheque Transaction
36	AVG_AMT_PER_NET_TXN	Numeric	Average Amount per Net banking Transaction
37	AVG_AMT_PER_MOB_TXN	Numeric	Average Amount per Mobile Transaction
38	FLG_HAS_NOMINEE	Integer	Flag: Has Nominee - 1: Yes, 0: No
39	FLG_HAS_OLD_LOAN	Integer	Flag: Has any earlier loan - 1: Yes, 0: No
40	random	Numeric	Random Number

3 Exploratory Data Analysis and Descriptive Statistics

3.1 Key observations:

Number of Rows and Columns:

- The number of rows in the dataset is 20,000
- The number of columns (Features) in the dataset is 40

3.2 Missing values check:

No Missing values present in the dataset.

3.3 Proportion of Responders Vs Non Responders:

- Total Responder Records: 2512 (12.56%)
- Total Non-Responder Records: 17,488 (87.44%)

Inference: Non Responder class is highly dominating. We might need to Balance out the dataset to achieve better classification results.

3.4 Summary Statistics of the dataset:

BALANCE	OCCUPATION	AGE_BKT	SCR
Min. : 0.00	PROF :5417	<25 :1753	Min. :100.0000
1st Qu.: 64754.03	SAL :5855	>50 :3035	1st Qu.:227.000
Median : 231675.85	SELF-EMP:3568	26-30:3434	Median :364.00
Mean : 511362.19	SENP :5160	31-35:3404	Mean :440.150
3rd Qu.: 653876.85		36-40:2814	3rd Qu.:644.000
Max. :8360430.57		41-45:3067	Max. :999.000
		46-50:2493	

HOLDING_PERIOD	ACC_TYPE	ACC_OP_DATE	LEN_OF_RLTN_IN_MNTH
Min. : 1.00000	CA: 4241	11/16/2010: 24	Min. : 29.0000
1st Qu.: 7.00000	SA:15759	04-03-09 : 23	1st Qu.: 79.0000
Median :15.00000		7/25/2010 : 22	Median :125.0000
Mean :14.95565		05-06-13 : 21	Mean :125.2393
3rd Qu.:22.00000		02-07-07 : 20	3rd Qu.:172.0000
Max. :31.00000		8/24/2010 : 20	Max. :221.0000
		(Other) :19870	

NO_OF_L_CR_TXNS	NO_OF_L_DR_TXNS	TOT_NO_OF_L_TXNS	NO_OF_BR_CSH_WDL_DR_TXNS
Min. : 0.00000	Min. : 0.0000	Min. : 0.0000	Min. : 0.000
1st Qu.: 6.00000	1st Qu.: 2.0000	1st Qu.: 9.0000	1st Qu.: 1.000
Median :10.00000	Median : 5.0000	Median : 14.0000	Median : 1.000
Mean :12.34805	Mean : 6.6337	Mean : 18.9754	Mean : 1.883
3rd Qu.:14.00000	3rd Qu.: 7.0000	3rd Qu.: 21.0000	3rd Qu.: 2.000
Max. :75.00000	Max. :74.0000	Max. :149.0000	Max. :15.000

NO_OF_ATM_DR_TXNS	NO_OF_NET_DR_TXNS	NO_OF_MOB_DR_TXNS	NO_OF_CHQ_DR_TXNS
Min. : 0.00000	Min. : 0.00000	Min. : 0.00000	Min. : 0.00000
1st Qu.: 0.00000	1st Qu.: 0.00000	1st Qu.: 0.00000	1st Qu.: 0.00000
Median : 1.00000	Median : 0.00000	Median : 0.00000	Median : 2.00000
Mean : 1.02895	Mean : 1.17245	Mean : 0.41175	Mean : 2.13755
3rd Qu.: 1.00000	3rd Qu.: 1.00000	3rd Qu.: 0.00000	3rd Qu.: 4.00000
Max. :25.00000	Max. :22.00000	Max. :25.00000	Max. :15.00000

FLG_HAS_CC	AMT_ATM_DR	AMT_BR_CSH_WDL_DR	AMT_CHQ_DR
Min. :0.0000	Min. : 0.00	Min. : 0.0	Min. : 0
1st Qu.:0.0000	1st Qu.: 0.00	1st Qu.: 2990.0	1st Qu.: 0
Median :0.0000	Median : 6900.00	Median :340150.0	Median : 23840
Mean :0.3054	Mean : 10990.01	Mean :378474.5	Mean : 124520
3rd Qu.:1.0000	3rd Qu.: 15800.00	3rd Qu.:674675.0	3rd Qu.: 72470
Max. :1.0000	Max. :199300.00	Max. :999930.0	Max. :4928640

AMT_NET_DR	AMT_MOB_DR	AMT_L_DR	FLG_HAS_ANY_CHGS
Min. : 0.0	Min. : 0.0	Min. : 0.0	Min. :0.0000
1st Qu.: 0.0	1st Qu.: 0.0	1st Qu.: 237935.5	1st Qu.:0.0000
Median : 0.0	Median : 0.0	Median : 695115.0	Median :0.0000
Mean :237307.8	Mean : 22424.7	Mean : 773717.0	Mean :0.1106
3rd Qu.:473970.5	3rd Qu.: 0.0	3rd Qu.:1078927.0	3rd Qu.:0.0000
Max. :999854.0	Max. :199667.0	Max. :6514921.0	Max. :1.0000

AMT_OTH_BK_ATM_USG_CHGS	AMT_MIN_BAL_NMC_CHGS	NO_OF_IW_CHQ_BNC_TXNS	NO_OF_OW_CHQ_BNC_TXNS
Min. : 0.0000	Min. : 0.000	Min. :0.00000	Min. :0.0000
1st Qu.: 0.0000	1st Qu.: 0.000	1st Qu.:0.00000	1st Qu.:0.0000
Median : 0.0000	Median : 0.000	Median :0.00000	Median :0.0000
Mean : 1.0995	Mean : 1.292	Mean :0.04275	Mean :0.0444
3rd Qu.: 0.0000	3rd Qu.: 0.000	3rd Qu.:0.00000	3rd Qu.:0.0000
Max. :250.0000	Max. :170.000	Max. :2.00000	Max. :2.0000

AVG_AMT_PER_ATM_TXN	AVG_AMT_PER_CSH_WDL_TXN	AVG_AMT_PER_CHQ_TXN	AVG_AMT_PER_NET_TXN
Min. : 0.00	Min. : 0.00	Min. : 0.00	Min. : 0
1st Qu.: 0.00	1st Qu.: 1265.45	1st Qu.: 0.00	1st Qu.: 0
Median : 6000.00	Median :147095.00	Median : 8645.00	Median : 0
Mean : 7408.84	Mean :242236.48	Mean : 25092.48	Mean :179059
3rd Qu.:13500.00	3rd Qu.:385000.00	3rd Qu.: 28605.00	3rd Qu.:257699
Max. :25000.00	Max. :999640.00	Max. :537842.22	Max. :999854

AVG_AMT_PER_MOB_TXN	FLG_HAS_NOMINEE	FLG_HAS_OLD_LOAN	random
Min. : 0.00	Min. :0.00000	Min. :0.00000	Min. :0.0000114
1st Qu.: 0.00	1st Qu.:1.00000	1st Qu.:0.00000	1st Qu.:0.2481866
Median : 0.00	Median :1.00000	Median :0.00000	Median :0.5061214
Mean : 20303.92	Mean :0.90115	Mean :0.49295	Mean :0.5019330
3rd Qu.: 0.00	3rd Qu.:1.00000	3rd Qu.:1.00000	3rd Qu.:0.7535712
Max. :199667.00	Max. :1.00000	Max. :1.00000	Max. :0.9999471

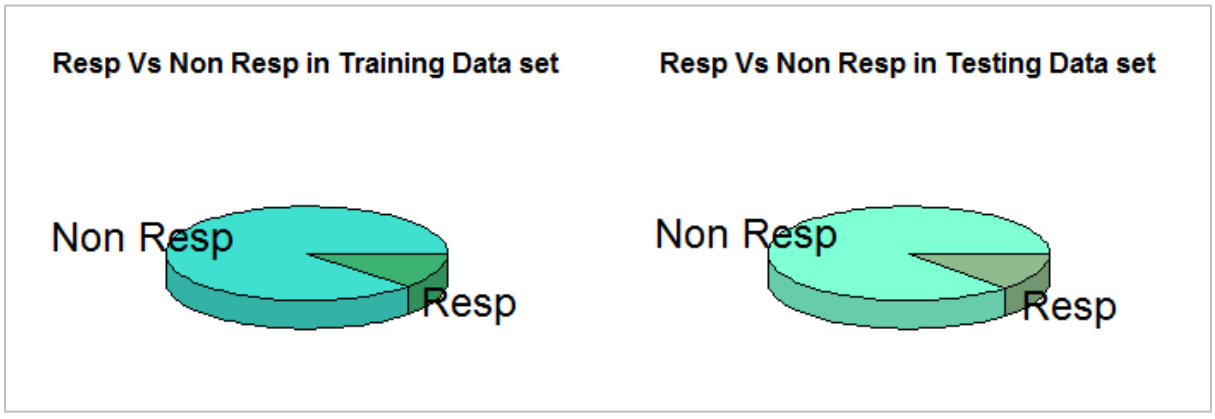
4 Creation of Training and Testing Data set

The given data set is divided into Training and Testing data set, with 70:30 proportion.

The distribution of Responder and Non Responder Class is verified in both the data sets, and ensured it's close to equal.

Following Table summarises the total number of records as well as percentage distribution of Responder Vs Non Responder Records.

Parameter	Training Data	Testing Data
No of Records	14000	6000
No of Features	40	40
% of Responder	87.65%	87%
% of Non Resp.	12.34%	13%



5 Model Building – CART

Decision Trees are commonly used in data mining with the objective of creating a model that predicts the value of a target (or dependent variable) based on the values of several input (or independent variables).

As an Umbrella term, the Classification and Regression Tree refers to the following decision trees:

- **Classification Trees:** where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.
- **Regression Trees:** where the target variable is continuous and tree is used to predict its value.

The CART algorithm is structured as a sequence of questions, the answers to which determine what the next question, if any should be. The result of these questions is a tree like structure where the ends are terminal nodes at which point there are no more questions.

5.1 The CART Model output

The initial CART Model is built by setting up Control Parameters as follows:

```
# Setting the control parameter inputs for rpart.
r.ctrl = rpart.control(minsplit=100,
                       minbucket = 10,
                       cp = 0, xval = 10)

# Build the Model on Training Dataset.
# Exclude Columns Customer ID and Acct Opening Date.
# names(cart.train)
m1 <- rpart(formula = TARGET ~ .,
            data = cart.train[,-c(1,11)], method = "class",
#            data = cart.train[,-c(1)], method = "class",
            control = r.ctrl)
```

The Model outcome is as follows:

```
> m1
n= 24544

node), split, n, loss, yval, (yprob)
      * denotes terminal node

     1) root 24544 12272 0 (0.50000000 0.50000000)
       2) NO_OF_L_DR_TXNS< 5.5 13088  5433 0 (0.58488692 0.41511308)
         4) FLG_HAS_CC< 0.5 8342  2899 0 (0.65248142 0.34751858)
           8) NO_OF_L_CR_TXNS< 16.5 6904  2112 0 (0.69409038 0.30590962)
            16) HOLDING_PERIOD>=12.5 4158   841 0 (0.79773930 0.20226070)
              32) SCR< 995.5 4124   809 0 (0.80383123 0.19616877)
                64) OCCUPATION=PROF,SAL,SENP 3253   520 0 (0.84014756 0.15985244)
                 128) AMT_MOB_DR< 73570 2982   406 0 (0.86384977 0.13615023)
                   256) NO_OF_L_CR_TXNS< 5.5 1083    73 0 (0.93259464 0.06740536)
                     512) AMT_BR_CSH_WDL_DR< 885060 1007    51 0 (0.94935452 0.05064548)
```

```
                  1024) AGE_BKT=<25,>50,26-30,31-35,36-40,41-45 868    20 0 (0.97695853 0.02304147) *
                  1025) AGE_BKT=46-50 139    31 0 (0.77697842 0.22302158)
                    2050) NO_OF_L_DR_TXNS>=0.5 93     0 0 (1.00000000 0.00000000) *
                    2051) NO_OF_L_DR_TXNS< 0.5 46    15 1 (0.32608696 0.67391304) *
                513) AMT_BR_CSH_WDL_DR>=885060 76    22 0 (0.71052632 0.28947368) *
              257) NO_OF_L_CR_TXNS>=5.5 1899   333 0 (0.82464455 0.17535545)
                514) AMT_L_DR>=8250 1819   292 0 (0.83947224 0.16052776)
                  1028) SCR< 194.5 299     0 0 (1.00000000 0.00000000) *
                  1029) SCR>=194.5 1520   292 0 (0.80789474 0.19210526)
                    2058) ACC_TYPE=SA 1373   225 0 (0.83612527 0.16387473)
                      4116) AVG_AMT_PER_CSH_WDL_TXN< 688815 1145   143 0 (0.87510917 0.12489083)
                        8232) AVG_AMT_PER_CHQ_TXN< 91355 1102   119 0 (0.89201452 0.10798548)
                          16464) NO_OF_L_CR_TXNS< 10.5 698    32 0 (0.95415473 0.04584527) *
                          16465) NO_OF_L_CR_TXNS>=10.5 404    87 0 (0.78465347 0.21534653)
                            32930) SCR< 382 142     0 0 (1.00000000 0.00000000) *
                            32931) SCR>=382 262    87 0 (0.66793893 0.33206107)
                              65862) TOT_NO_OF_L_TXNS>=14.5 154    24 0 (0.84415584 0.15584416)
                                131724) SCR>=463.5 111     0 0 (1.00000000 0.00000000) *
                                131725) SCR< 463.5 43    19 1 (0.44186047 0.55813953) *
                              65863) TOT_NO_OF_L_TXNS< 14.5 108    45 1 (0.41666667 0.58333333)
                                131726) AGE_BKT=>50,26-30,36-40,46-50 32     0 0 (1.00000000 0.00000000)
*
                                131727) AGE_BKT=<25,31-35,41-45 76    13 1 (0.17105263 0.82894737) *
                        8233) AVG_AMT_PER_CHQ_TXN>=91355 43    19 1 (0.44186047 0.55813953) *
                      4117) AVG_AMT_PER_CSH_WDL_TXN>=688815 228    82 0 (0.64035088 0.35964912)
                        8234) AMT_BR_CSH_WDL_DR>=698060 186    40 0 (0.78494624 0.21505376)
                          16468) NO_OF_ATM_DR_TXNS< 2.5 160    17 0 (0.89375000 0.10625000)
                            32936) HOLDING_PERIOD>=14.5 139     0 0 (1.00000000 0.00000000) *
                            32937) HOLDING_PERIOD< 14.5 21     4 1 (0.19047619 0.80952381) *
                          16469) NO_OF_ATM_DR_TXNS>=2.5 26     3 1 (0.11538462 0.88461538) *
                        8235) AMT_BR_CSH_WDL_DR< 698060 42     0 1 (0.00000000 1.00000000) *
                    2059) ACC_TYPE=CA 147    67 0 (0.54421769 0.45578231)
                      4118) AGE< 37.5 46     0 0 (1.00000000 0.00000000) *
                      4119) AGE>=37.5 101    34 1 (0.33663366 0.66336634) *
                515) AMT_L_DR< 8250 80    39 1 (0.48750000 0.51250000) *
            129) AMT_MOB_DR>=73570 271   114 0 (0.57933579 0.42066421)
              258) NO_OF_ATM_DR_TXNS< 1.5 197    46 0 (0.76649746 0.23350254)
                516) AMT_CHQ_DR< 53600 175    26 0 (0.85142857 0.14857143) *
                517) AMT_CHQ_DR>=53600 22     2 1 (0.09090909 0.90909091) *
              259) NO_OF_ATM_DR_TXNS>=1.5 74     6 1 (0.08108108 0.91891892) *
          65) OCCUPATION=SELF-EMP 871   289 0 (0.66819747 0.33180253)
            130) LEN_OF_RLTN_IN_MNTH>=40.5 747   187 0 (0.74966533 0.25033467)
              260) SCR< 787 613   102 0 (0.83360522 0.16639478)
                520) AMT_BR_CSH_WDL_DR< 882390 537    61 0 (0.88640596 0.11359404)
                  1040) AMT_CHQ_DR< 79420 480    34 0 (0.92916667 0.07083333)
                    2080) NO_OF_L_CR_TXNS>=2.5 358     4 0 (0.98882682 0.01117318) *
                    2081) NO_OF_L_CR_TXNS< 2.5 122    30 0 (0.75409836 0.24590164)
                      4162) SCR< 422.5 69     0 0 (1.00000000 0.00000000) *
                      4163) SCR>=422.5 53    23 1 (0.43396226 0.56603774) *
                  1041) AMT_CHQ_DR>=79420 57    27 0 (0.52631579 0.47368421) *
                521) AMT_BR_CSH_WDL_DR>=882390 76    35 1 (0.46052632 0.53947368) *
              261) SCR>=787 134    49 1 (0.36567164 0.63432836)
                522) BALANCE>=177903.3 34     3 0 (0.91176471 0.08823529) *
                523) BALANCE< 177903.3 100    18 1 (0.18000000 0.82000000) *
            131) LEN_OF_RLTN_IN_MNTH< 40.5 124    22 1 (0.17741935 0.82258065) *
        33) SCR>=995.5 34     2 1 (0.05882353 0.94117647) *
      17) HOLDING_PERIOD< 12.5 2746  1271 0 (0.53714494 0.46285506)
        34) AMT_ATM_DR< 13350 2089   816 0 (0.60938248 0.39061752)
          68) SCR< 210.5 348    60 0 (0.82758621 0.17241379)
            136) AMT_NET_DR< 867563.5 315    37 0 (0.88253968 0.11746032) *
            137) AMT_NET_DR>=867563.5 33    10 1 (0.30303030 0.69696970) *
          69) SCR>=210.5 1741   756 0 (0.56576680 0.43423320)
            138) AGE_BKT=>50,26-30,36-40,46-50 879   296 0 (0.66325370 0.33674630)
              276) AMT_ATM_DR>=400 237    20 0 (0.91561181 0.08438819) *
              277) AMT_ATM_DR< 400 642   276 0 (0.57009346 0.42990654)
                554) NO_OF_L_CR_TXNS< 8 575   209 0 (0.63652174 0.36347826)
                  1108) OCCUPATION=PROF,SAL 235    34 0 (0.85531915 0.14468085) *
                  1109) OCCUPATION=SELF-EMP,SENP 340   165 1 (0.48529412 0.51470588)
                    2218) GENDER=F 72     7 0 (0.90277778 0.09722222) *
                    2219) GENDER=M 268   100 1 (0.37313433 0.62686567)
                      4438) BALANCE>=290982.1 85    24 0 (0.71764706 0.28235294) *
                      4439) BALANCE< 290982.1 183    39 1 (0.21311475 0.78688525)
                        8878) AVG_AMT_PER_CHQ_TXN< 32277.5 114    36 1 (0.31578947 0.68421053)
                          17756) NO_OF_L_DR_TXNS< 2.5 29     8 0 (0.72413793 0.27586207) *
                          17757) NO_OF_L_DR_TXNS>=2.5 85    15 1 (0.17647059 0.82352941) *
                        8879) AVG_AMT_PER_CHQ_TXN>=32277.5 69     3 1 (0.04347826 0.95652174) *
                555) NO_OF_L_CR_TXNS>=8 67     0 1 (0.00000000 1.00000000) *
            139) AGE_BKT=<25,31-35,41-45 862   402 1 (0.46635731 0.53364269)
              278) AVG_AMT_PER_CSH_WDL_TXN>=628750 68     7 0 (0.89705882 0.10294118) *
              279) AVG_AMT_PER_CSH_WDL_TXN< 628750 794   341 1 (0.42947103 0.57052897)
```

```
                558) LEN_OF_RLTN_IN_MNTH>=176.5 115    32 0 (0.72173913 0.27826087)
                 1116) AVG_AMT_PER_CHQ_TXN< 34637.5 94    13 0 (0.86170213 0.13829787) *
                 1117) AVG_AMT_PER_CHQ_TXN>=34637.5 21     2 1 (0.09523810 0.90476190) *
                559) LEN_OF_RLTN_IN_MNTH< 176.5 679   258 1 (0.37997054 0.62002946)
                 1118) NO_OF_CHQ_DR_TXNS>=2.5 96    33 0 (0.65625000 0.34375000) *
                 1119) NO_OF_CHQ_DR_TXNS< 2.5 583   195 1 (0.33447684 0.66552316)
                   2238) SCR>=447.5 279   134 1 (0.48028674 0.51971326)
                     4476) GENDER=F 46     0 0 (1.00000000 0.00000000) *
                     4477) GENDER=M 233    88 1 (0.37768240 0.62231760)
                       8954) LEN_OF_RLTN_IN_MNTH>=137.5 28     0 0 (1.00000000 0.00000000) *
                       8955) LEN_OF_RLTN_IN_MNTH< 137.5 205    60 1 (0.29268293 0.70731707)
                        17910) LEN_OF_RLTN_IN_MNTH< 77 42    14 0 (0.66666667 0.33333333) *
                        17911) LEN_OF_RLTN_IN_MNTH>=77 163    32 1 (0.19631902 0.80368098) *
                   2239) SCR< 447.5 304    61 1 (0.20065789 0.79934211)
                     4478) BALANCE< 460279.6 80    37 0 (0.53750000 0.46250000) *
                     4479) BALANCE>=460279.6 224    18 1 (0.08035714 0.91964286) *
            35) AMT_ATM_DR>=13350 657   202 1 (0.30745814 0.69254186)
              70) AGE_BKT=26-30 43     0 0 (1.00000000 0.00000000) *
              71) AGE_BKT=<25,>50,31-35,36-40,41-45,46-50 614   159 1 (0.25895765 0.74104235)
               142) AGE>=47.5 74    26 0 (0.64864865 0.35135135) *
               143) AGE< 47.5 540   111 1 (0.20555556 0.79444444)
                 286) BALANCE>=885810 40    16 0 (0.60000000 0.40000000) *
                 287) BALANCE< 885810 500    87 1 (0.17400000 0.82600000)
                   574) LEN_OF_RLTN_IN_MNTH< 62.5 36    17 0 (0.52777778 0.47222222) *
                   575) LEN_OF_RLTN_IN_MNTH>=62.5 464    68 1 (0.14655172 0.85344828) *
          9) NO_OF_L_CR_TXNS>=16.5 1438   651 1 (0.45271210 0.54728790)
           18) OCCUPATION=SAL 321    89 0 (0.72274143 0.27725857)
             36) AGE_BKT=<25,>50,26-30,31-35,36-40,41-45 226    22 0 (0.90265487 0.09734513) *
             37) AGE_BKT=46-50 95    28 1 (0.29473684 0.70526316) *
           19) OCCUPATION=PROF,SELF-EMP,SENP 1117   419 1 (0.37511191 0.62488809)
             38) BALANCE>=105898.3 587   290 1 (0.49403748 0.50596252)
               76) SCR< 201 79     4 0 (0.94936709 0.05063291) *
               77) SCR>=201 508   215 1 (0.42322835 0.57677165)
               154) AGE_BKT=31-35,36-40 125    36 0 (0.71200000 0.28800000)
                 308) AMT_NET_DR< 469244 74     0 0 (1.00000000 0.00000000) *
                 309) AMT_NET_DR>=469244 51    15 1 (0.29411765 0.70588235) *
               155) AGE_BKT=<25,>50,26-30,41-45,46-50 383   126 1 (0.32898172 0.67101828)
                 310) AGE< 24.5 23     0 0 (1.00000000 0.00000000) *
                 311) AGE>=24.5 360   103 1 (0.28611111 0.71388889)
                   622) LEN_OF_RLTN_IN_MNTH< 158 150    67 1 (0.44666667 0.55333333)
                    1244) LEN_OF_RLTN_IN_MNTH>=100 41     0 0 (1.00000000 0.00000000) *
                    1245) LEN_OF_RLTN_IN_MNTH< 100 109    26 1 (0.23853211 0.76146789)
                      2490) AMT_ATM_DR< 2200 22     4 0 (0.81818182 0.18181818) *
                      2491) AMT_ATM_DR>=2200 87     8 1 (0.09195402 0.90804598) *
                   623) LEN_OF_RLTN_IN_MNTH>=158 210    36 1 (0.17142857 0.82857143) *
             39) BALANCE< 105898.3 530   129 1 (0.24339623 0.75660377)
               78) AMT_L_DR< 124566 30     0 0 (1.00000000 0.00000000) *
               79) AMT_L_DR>=124566 500    99 1 (0.19800000 0.80200000)
               158) AMT_BR_CSH_WDL_DR>=767370 25     3 0 (0.88000000 0.12000000) *
               159) AMT_BR_CSH_WDL_DR< 767370 475    77 1 (0.16210526 0.83789474)
                 318) AGE>=46.5 38    13 0 (0.65789474 0.34210526) *
                 319) AGE< 46.5 437    52 1 (0.11899314 0.88100686)
                   638) BALANCE< 13188.08 22     8 0 (0.63636364 0.36363636) *
                   639) BALANCE>=13188.08 415    38 1 (0.09156627 0.90843373) *
        5) FLG_HAS_CC>=0.5 4746  2212 1 (0.46607670 0.53392330)
         10) NO_OF_L_DR_TXNS< 1.5 820   288 0 (0.64878049 0.35121951)
           20) OCCUPATION=PROF,SAL,SENP 630   155 0 (0.75396825 0.24603175)
             40) HOLDING_PERIOD>=15.5 377    34 0 (0.90981432 0.09018568)
               80) OCCUPATION=PROF,SENP 268     5 0 (0.98134328 0.01865672) *
               81) OCCUPATION=SAL 109    29 0 (0.73394495 0.26605505)
               162) AGE_BKT=<25,>50,26-30,31-35,41-45,46-50 82    10 0 (0.87804878 0.12195122) *
               163) AGE_BKT=36-40 27     8 1 (0.29629630 0.70370370) *
             41) HOLDING_PERIOD< 15.5 253   121 0 (0.52173913 0.47826087)
               82) SCR< 582 127    34 0 (0.73228346 0.26771654)
               164) AGE>=28 91     4 0 (0.95604396 0.04395604) *
               165) AGE< 28 36     6 1 (0.16666667 0.83333333) *
               83) SCR>=582 126    39 1 (0.30952381 0.69047619)
               166) FLG_HAS_OLD_LOAN>=0.5 40    14 0 (0.65000000 0.35000000) *
               167) FLG_HAS_OLD_LOAN< 0.5 86    13 1 (0.15116279 0.84883721) *
           21) OCCUPATION=SELF-EMP 190    57 1 (0.30000000 0.70000000)
             42) AGE_BKT=>50,26-30,31-35,36-40,46-50 74    23 0 (0.68918919 0.31081081) *
             43) AGE_BKT=<25,41-45 116     6 1 (0.05172414 0.94827586) *
         11) NO_OF_L_DR_TXNS>=1.5 3926  1680 1 (0.42791645 0.57208355)
           22) AGE< 26.5 273    86 0 (0.68498168 0.31501832)
             44) HOLDING_PERIOD>=6.5 181    24 0 (0.86740331 0.13259669)
               88) NO_OF_IW_CHQ_BNC_TXNS< 0.5 147     0 0 (1.00000000 0.00000000) *
               89) NO_OF_IW_CHQ_BNC_TXNS>=0.5 34    10 1 (0.29411765 0.70588235) *
             45) HOLDING_PERIOD< 6.5 92    30 1 (0.32608696 0.67391304) *
           23) AGE>=26.5 3653  1493 1 (0.40870517 0.59129483)
             46) AMT_L_DR< 435905 1281   634 0 (0.50507416 0.49492584)
```

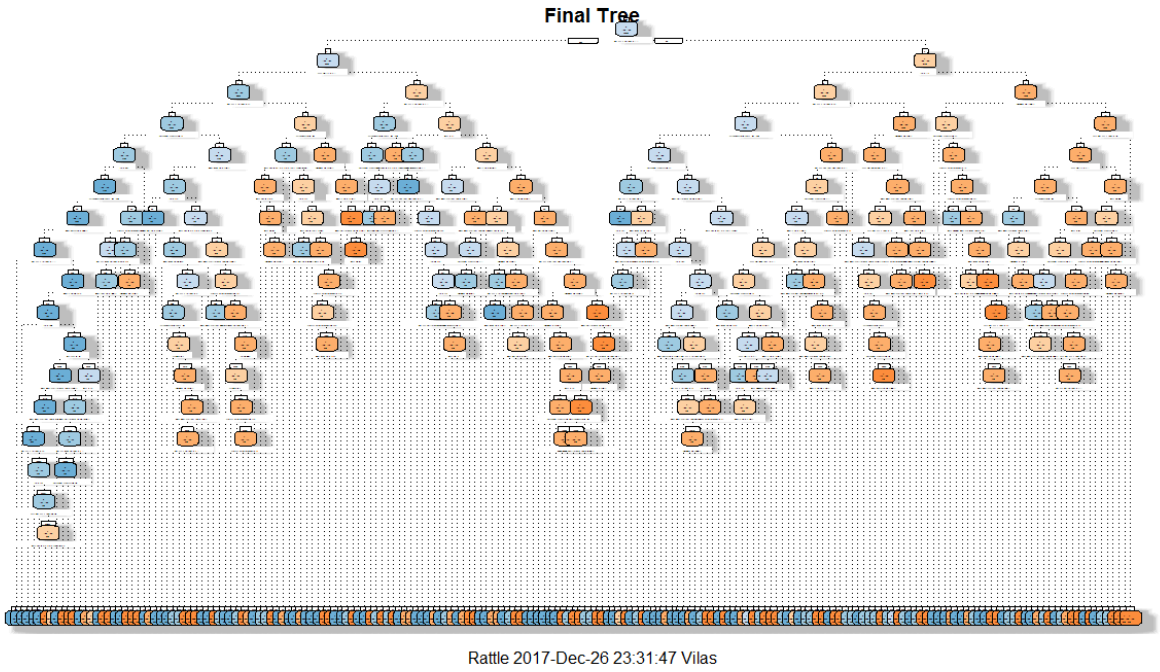
```
            92) AVG_AMT_PER_CHQ_TXN< 19730.88 865   353 0 (0.59190751 0.40809249)
             184) LEN_OF_RLTN_IN_MNTH>=187.5 79     0 0 (1.00000000 0.00000000) *
             185) LEN_OF_RLTN_IN_MNTH< 187.5 786   353 0 (0.55089059 0.44910941)
               370) SCR>=780.5 95     9 0 (0.90526316 0.09473684) *
               371) SCR< 780.5 691   344 0 (0.50217077 0.49782923)
                 742) SCR< 367 268    73 0 (0.72761194 0.27238806)
                  1484) NO_OF_BR_CSH_WDL_DR_TXNS< 2.5 206    32 0 (0.84466019 0.15533981) *
                  1485) NO_OF_BR_CSH_WDL_DR_TXNS>=2.5 62    21 1 (0.33870968 0.66129032) *
                 743) SCR>=367 423   152 1 (0.35933806 0.64066194)
                  1486) HOLDING_PERIOD>=26.5 34     0 0 (1.00000000 0.00000000) *
                  1487) HOLDING_PERIOD< 26.5 389   118 1 (0.30334190 0.69665810)
                    2974) AGE>=51.5 36     9 0 (0.75000000 0.25000000) *
                    2975) AGE< 51.5 353    91 1 (0.25779037 0.74220963) *
            93) AVG_AMT_PER_CHQ_TXN>=19730.88 416   135 1 (0.32451923 0.67548077)
             186) AVG_AMT_PER_CHQ_TXN>=28145 180    90 0 (0.50000000 0.50000000)
               372) HOLDING_PERIOD>=3.5 136    47 0 (0.65441176 0.34558824)
                 744) BALANCE>=42436.2 89    13 0 (0.85393258 0.14606742) *
                 745) BALANCE< 42436.2 47    13 1 (0.27659574 0.72340426) *
               373) HOLDING_PERIOD< 3.5 44     1 1 (0.02272727 0.97727273) *
             187) AVG_AMT_PER_CHQ_TXN< 28145 236    45 1 (0.19067797 0.80932203) *
          47) AMT_L_DR>=435905 2372   846 1 (0.35666105 0.64333895)
            94) AMT_L_DR>=971710 685   333 1 (0.48613139 0.51386861)
             188) AGE_BKT=31-35 78     0 0 (1.00000000 0.00000000) *
             189) AGE_BKT=>50,26-30,36-40,41-45,46-50 607   255 1 (0.42009885 0.57990115)
               378) BALANCE>=279978.4 180    57 0 (0.68333333 0.31666667)
                 756) NO_OF_L_CR_TXNS< 10.5 115    18 0 (0.84347826 0.15652174)
                  1512) AMT_ATM_DR< 17350 90     0 0 (1.00000000 0.00000000) *
                  1513) AMT_ATM_DR>=17350 25     7 1 (0.28000000 0.72000000) *
                 757) NO_OF_L_CR_TXNS>=10.5 65    26 1 (0.40000000 0.60000000) *
               379) BALANCE< 279978.4 427   132 1 (0.30913349 0.69086651)
                 758) AMT_NET_DR< 32073 26     0 0 (1.00000000 0.00000000) *
                 759) AMT_NET_DR>=32073 401   106 1 (0.26433915 0.73566085)
                  1518) NO_OF_L_CR_TXNS< 8.5 177    71 1 (0.40112994 0.59887006)
                    3036) AMT_NET_DR< 950856.5 99    32 0 (0.67676768 0.32323232) *
                    3037) AMT_NET_DR>=950856.5 78     4 1 (0.05128205 0.94871795) *
                  1519) NO_OF_L_CR_TXNS>=8.5 224    35 1 (0.15625000 0.84375000)
                    3038) AGE>=45 40    17 0 (0.57500000 0.42500000) *
                    3039) AGE< 45 184    12 1 (0.06521739 0.93478261) *
            95) AMT_L_DR< 971710 1687   513 1 (0.30409010 0.69590990)
             190) AMT_NET_DR>=638496.5 51     7 0 (0.86274510 0.13725490) *
             191) AMT_NET_DR< 638496.5 1636   469 1 (0.28667482 0.71332518)
               382) NO_OF_OW_CHQ_BNC_TXNS>=0.5 20     0 0 (1.00000000 0.00000000) *
               383) NO_OF_OW_CHQ_BNC_TXNS< 0.5 1616   449 1 (0.27784653 0.72215347)
                 766) BALANCE>=100250.1 972   329 1 (0.33847737 0.66152263)
                  1532) BALANCE< 182252.4 93    15 0 (0.83870968 0.16129032) *
                  1533) BALANCE>=182252.4 879   251 1 (0.28555176 0.71444824)
                    3066) AMT_MOB_DR>=90949.5 28     0 0 (1.00000000 0.00000000) *
                    3067) AMT_MOB_DR< 90949.5 851   223 1 (0.26204465 0.73795535)
                      6134) AGE< 38.5 314   121 1 (0.38535032 0.61464968)
                       12268) HOLDING_PERIOD>=24.5 73    12 0 (0.83561644 0.16438356) *
                       12269) HOLDING_PERIOD< 24.5 241    60 1 (0.24896266 0.75103734)
                         24538) AGE>=32.5 24     0 0 (1.00000000 0.00000000) *
                         24539) AGE< 32.5 217    36 1 (0.16589862 0.83410138) *
                      6135) AGE>=38.5 537   102 1 (0.18994413 0.81005587)
                       12270) OCCUPATION=SAL 114    43 1 (0.37719298 0.62280702)
                         24540) AVG_AMT_PER_CSH_WDL_TXN< 438515 42    13 0 (0.69047619 0.30952381) *
                         24541) AVG_AMT_PER_CSH_WDL_TXN>=438515 72    14 1 (0.19444444 0.80555556) *
                       12271) OCCUPATION=PROF,SELF-EMP,SENP 423    59 1 (0.13947991 0.86052009) *
                 767) BALANCE< 100250.1 644   120 1 (0.18633540 0.81366460)
                  1534) AMT_BR_CSH_WDL_DR>=836725 23     6 0 (0.73913043 0.26086957) *
                  1535) AMT_BR_CSH_WDL_DR< 836725 621   103 1 (0.16586151 0.83413849)
                    3070) NO_OF_BR_CSH_WDL_DR_TXNS>=1.5 189    60 1 (0.31746032 0.68253968)
                      6140) BALANCE< 67708.63 75    21 0 (0.72000000 0.28000000) *
                      6141) BALANCE>=67708.63 114     6 1 (0.05263158 0.94736842) *
                    3071) NO_OF_BR_CSH_WDL_DR_TXNS< 1.5 432    43 1 (0.09953704 0.90046296) *
3) NO_OF_L_DR_TXNS>=5.5 11456  4617 1 (0.40302025 0.59697975)
  6) SCR< 552.5 6853  3176 1 (0.46344667 0.53655333)
   12) NO_OF_L_CR_TXNS< 17.5 4494  2130 0 (0.52603471 0.47396529)
     24) OCCUPATION=PROF,SAL,SENP 3527  1521 0 (0.56875532 0.43124468)
       48) HOLDING_PERIOD>=15.5 737   194 0 (0.73677069 0.26322931)
         96) NO_OF_L_CR_TXNS>=6.5 559    93 0 (0.83363148 0.16636852)
          192) SCR< 386 364    10 0 (0.97252747 0.02747253) *
          193) SCR>=386 195    83 0 (0.57435897 0.42564103)
            386) AMT_MOB_DR< 127453.5 150    41 0 (0.72666667 0.27333333)
              772) NO_OF_L_CR_TXNS>=11.5 78     0 0 (1.00000000 0.00000000) *
              773) NO_OF_L_CR_TXNS< 11.5 72    31 1 (0.43055556 0.56944444) *
            387) AMT_MOB_DR>=127453.5 45     3 1 (0.06666667 0.93333333) *
         97) NO_OF_L_CR_TXNS< 6.5 178    77 1 (0.43258427 0.56741573)
          194) AMT_ATM_DR>=11850 45     0 0 (1.00000000 0.00000000) *
          195) AMT_ATM_DR< 11850 133    32 1 (0.24060150 0.75939850)
```

```
          390) LEN_OF_RLTN_IN_MNTH< 158 37    14 0 (0.62162162 0.37837838) *
          391) LEN_OF_RLTN_IN_MNTH>=158 96     9 1 (0.09375000 0.90625000) *
       49) HOLDING_PERIOD< 15.5 2790  1327 0 (0.52437276 0.47562724)
         98) AMT_BR_CSH_WDL_DR>=952465 66     0 0 (1.00000000 0.00000000) *
         99) AMT_BR_CSH_WDL_DR< 952465 2724  1327 0 (0.51284875 0.48715125)
        198) AGE_BKT=<25,>50,31-35,36-40 1411   587 0 (0.58398299 0.41601701)
          396) SCR< 198.5 283    52 0 (0.81625442 0.18374558)
            792) HOLDING_PERIOD>=1.5 246    31 0 (0.87398374 0.12601626) *
            793) HOLDING_PERIOD< 1.5 37    16 1 (0.43243243 0.56756757) *
          397) SCR>=198.5 1128   535 0 (0.52570922 0.47429078)
            794) SCR>=206.5 1013   440 0 (0.56564659 0.43435341)
             1588) AVG_AMT_PER_NET_TXN< 64397.25 287    72 0 (0.74912892 0.25087108)
               3176) AMT_MOB_DR< 130096.5 215    22 0 (0.89767442 0.10232558) *
               3177) AMT_MOB_DR>=130096.5 72    22 1 (0.30555556 0.69444444) *
             1589) AVG_AMT_PER_NET_TXN>=64397.25 726   358 1 (0.49311295 0.50688705)
               3178) NO_OF_L_CR_TXNS>=10.5 405   151 0 (0.62716049 0.37283951)
                 6356) NO_OF_L_CR_TXNS< 12.5 118     6 0 (0.94915254 0.05084746) *
                 6357) NO_OF_L_CR_TXNS>=12.5 287   142 1 (0.49477352 0.50522648)
                  12714) AVG_AMT_PER_CHQ_TXN>=55890 52     0 0 (1.00000000 0.00000000) *
                  12715) AVG_AMT_PER_CHQ_TXN< 55890 235    90 1 (0.38297872 0.61702128)
                    25430) BALANCE>=313647.8 52    10 0 (0.80769231 0.19230769) *
                    25431) BALANCE< 313647.8 183    48 1 (0.26229508 0.73770492) *
               3179) NO_OF_L_CR_TXNS< 10.5 321   104 1 (0.32398754 0.67601246)
                 6358) SCR>=401 34     0 0 (1.00000000 0.00000000) *
                 6359) SCR< 401 287    70 1 (0.24390244 0.75609756)
                  12718) LEN_OF_RLTN_IN_MNTH< 75.5 33    11 0 (0.66666667 0.33333333) *
                  12719) LEN_OF_RLTN_IN_MNTH>=75.5 254    48 1 (0.18897638 0.81102362)
                    25438) AMT_L_DR< 540952 29    12 0 (0.58620690 0.41379310) *
                    25439) AMT_L_DR>=540952 225    31 1 (0.13777778 0.86222222) *
          795) SCR< 206.5 115    20 1 (0.17391304 0.82608696) *
        199) AGE_BKT=26-30,41-45,46-50 1313   573 1 (0.43640518 0.56359482)
          398) FLG_HAS_ANY_CHGS< 0.5 1080   532 1 (0.49259259 0.50740741)
            796) HOLDING_PERIOD>=12.5 146    30 0 (0.79452055 0.20547945)
             1592) AMT_ATM_DR>=1700 120     7 0 (0.94166667 0.05833333) *
             1593) AMT_ATM_DR< 1700 26     3 1 (0.11538462 0.88461538) *
            797) HOLDING_PERIOD< 12.5 934   416 1 (0.44539615 0.55460385)
             1594) AMT_L_DR>=714539 464   199 0 (0.57112069 0.42887931)
               3188) FLG_HAS_CC< 0.5 254    63 0 (0.75196850 0.24803150)
                 6376) SCR< 372.5 152     9 0 (0.94078947 0.05921053) *
                 6377) SCR>=372.5 102    48 1 (0.47058824 0.52941176)
                  12754) AMT_L_DR< 1059334 30     0 0 (1.00000000 0.00000000) *
                  12755) AMT_L_DR>=1059334 72    18 1 (0.25000000 0.75000000) *
               3189) FLG_HAS_CC>=0.5 210    74 1 (0.35238095 0.64761905)
                 6378) AGE>=29.5 97    41 0 (0.57731959 0.42268041) *
                 6379) AGE< 29.5 113    18 1 (0.15929204 0.84070796) *
             1595) AMT_L_DR< 714539 470   151 1 (0.32127660 0.67872340)
               3190) AMT_L_DR< 570388 227   113 0 (0.50220264 0.49779736)
                 6380) AMT_L_DR>=244320 140    42 0 (0.70000000 0.30000000) *
                 6381) AMT_L_DR< 244320 87    16 1 (0.18390805 0.81609195) *
               3191) AMT_L_DR>=570388 243    37 1 (0.15226337 0.84773663)
                 6382) SCR< 317 102    28 1 (0.27450980 0.72549020)
                  12764) BALANCE>=68073.69 41    16 0 (0.60975610 0.39024390) *
                  12765) BALANCE< 68073.69 61     3 1 (0.04918033 0.95081967) *
                 6383) SCR>=317 141     9 1 (0.06382979 0.93617021) *
          399) FLG_HAS_ANY_CHGS>=0.5 233    41 1 (0.17596567 0.82403433) *
     25) OCCUPATION=SELF-EMP 967   358 1 (0.37021717 0.62978283)
       50) NO_OF_IW_CHQ_BNC_TXNS< 0.5 796   336 1 (0.42211055 0.57788945)
        100) HOLDING_PERIOD>=3.5 550   271 0 (0.50727273 0.49272727)
          200) AMT_L_DR< 759040 136    36 0 (0.73529412 0.26470588)
            400) AMT_CHQ_DR< 56105 69     0 0 (1.00000000 0.00000000) *
            401) AMT_CHQ_DR>=56105 67    31 1 (0.46268657 0.53731343) *
          201) AMT_L_DR>=759040 414   179 1 (0.43236715 0.56763285)
            402) AMT_ATM_DR>=14800 107    31 0 (0.71028037 0.28971963)
              804) AVG_AMT_PER_NET_TXN>=71241.64 52     0 0 (1.00000000 0.00000000) *
              805) AVG_AMT_PER_NET_TXN< 71241.64 55    24 1 (0.43636364 0.56363636) *
            403) AMT_ATM_DR< 14800 307   103 1 (0.33550489 0.66449511)
              806) AGE_BKT=<25,>50 25     0 0 (1.00000000 0.00000000) *
              807) AGE_BKT=26-30,31-35,36-40,41-45,46-50 282    78 1 (0.27659574 0.72340426)
               1614) AMT_ATM_DR< 11050 157    65 1 (0.41401274 0.58598726)
                 3228) AVG_AMT_PER_ATM_TXN>=6000 46    11 0 (0.76086957 0.23913043) *
                 3229) AVG_AMT_PER_ATM_TXN< 6000 111    30 1 (0.27027027 0.72972973)
                   6458) AMT_L_DR>=1099355 32    12 0 (0.62500000 0.37500000) *
                   6459) AMT_L_DR< 1099355 79    10 1 (0.12658228 0.87341772) *
               1615) AMT_ATM_DR>=11050 125    13 1 (0.10400000 0.89600000) *
        101) HOLDING_PERIOD< 3.5 246    57 1 (0.23170732 0.76829268)
          202) LEN_OF_RLTN_IN_MNTH< 72.5 21     3 0 (0.85714286 0.14285714) *
          203) LEN_OF_RLTN_IN_MNTH>=72.5 225    39 1 (0.17333333 0.82666667) *
       51) NO_OF_IW_CHQ_BNC_TXNS>=0.5 171    22 1 (0.12865497 0.87134503)
        102) NO_OF_BR_CSH_WDL_DR_TXNS>=1.5 29    12 0 (0.58620690 0.41379310) *
        103) NO_OF_BR_CSH_WDL_DR_TXNS< 1.5 142     5 1 (0.03521127 0.96478873) *
```

```
          13) NO_OF_L_CR_TXNS>=17.5 2359   812 1 (0.34421365 0.65578635)
            26) BALANCE>=10171.98 1968   772 1 (0.39227642 0.60772358)
              52) FLG_HAS_NOMINEE< 0.5 121    30 0 (0.75206612 0.24793388)
               104) HOLDING_PERIOD>=6.5 64     0 0 (1.00000000 0.00000000) *
               105) HOLDING_PERIOD< 6.5 57    27 1 (0.47368421 0.52631579) *
              53) FLG_HAS_NOMINEE>=0.5 1847   681 1 (0.36870601 0.63129399)
               106) AVG_AMT_PER_CHQ_TXN>=23126.25 1140   496 1 (0.43508772 0.56491228)
                 212) TOT_NO_OF_L_TXNS< 56.5 642   279 0 (0.56542056 0.43457944)
                   424) AVG_AMT_PER_CSH_WDL_TXN< 105977.5 270    56 0 (0.79259259 0.20740741)
                     848) HOLDING_PERIOD>=2.5 212    24 0 (0.88679245 0.11320755) *
                     849) HOLDING_PERIOD< 2.5 58    26 1 (0.44827586 0.55172414) *
                   425) AVG_AMT_PER_CSH_WDL_TXN>=105977.5 372   149 1 (0.40053763 0.59946237)
                     850) SCR< 195 48     5 0 (0.89583333 0.10416667) *
                     851) SCR>=195 324   106 1 (0.32716049 0.67283951)
                      1702) OCCUPATION=PROF 75    27 0 (0.64000000 0.36000000) *
                      1703) OCCUPATION=SAL,SELF-EMP,SENP 249    58 1 (0.23293173 0.76706827)
                        3406) HOLDING_PERIOD>=14.5 27     6 0 (0.77777778 0.22222222) *
                        3407) HOLDING_PERIOD< 14.5 222    37 1 (0.16666667 0.83333333)
                          6814) AMT_MOB_DR>=154223 21     7 0 (0.66666667 0.33333333) *
                          6815) AMT_MOB_DR< 154223 201    23 1 (0.11442786 0.88557214) *
                 213) TOT_NO_OF_L_TXNS>=56.5 498   133 1 (0.26706827 0.73293173)
                   426) LEN_OF_RLTN_IN_MNTH>=194.5 55    20 0 (0.63636364 0.36363636) *
                   427) LEN_OF_RLTN_IN_MNTH< 194.5 443    98 1 (0.22121896 0.77878104)
                     854) AVG_AMT_PER_CHQ_TXN< 65377.5 75    33 0 (0.56000000 0.44000000) *
                     855) AVG_AMT_PER_CHQ_TXN>=65377.5 368    56 1 (0.15217391 0.84782609)
                      1710) LEN_OF_RLTN_IN_MNTH< 106.5 101    32 1 (0.31683168 0.68316832)
                        3420) SCR< 291 24     6 0 (0.75000000 0.25000000) *
                        3421) SCR>=291 77    14 1 (0.18181818 0.81818182) *
                      1711) LEN_OF_RLTN_IN_MNTH>=106.5 267    24 1 (0.08988764 0.91011236) *
               107) AVG_AMT_PER_CHQ_TXN< 23126.25 707   185 1 (0.26166902 0.73833098)
                 214) AMT_CHQ_DR< 11245 81    24 0 (0.70370370 0.29629630) *
                 215) AMT_CHQ_DR>=11245 626   128 1 (0.20447284 0.79552716)
                   430) AVG_AMT_PER_NET_TXN>=183514.5 23     4 0 (0.82608696 0.17391304) *
                   431) AVG_AMT_PER_NET_TXN< 183514.5 603   109 1 (0.18076285 0.81923715)
                     862) AGE< 27.5 29     7 0 (0.75862069 0.24137931) *
                     863) AGE>=27.5 574    87 1 (0.15156794 0.84843206)
                      1726) AGE_BKT=>50,31-35,41-45 205    56 1 (0.27317073 0.72682927)
                        3452) NO_OF_BR_CSH_WDL_DR_TXNS< 8.5 75    37 1 (0.49333333 0.50666667) *
                        3453) NO_OF_BR_CSH_WDL_DR_TXNS>=8.5 130    19 1 (0.14615385 0.85384615)
                          6906) BALANCE>=614419.1 20     9 0 (0.55000000 0.45000000) *
                          6907) BALANCE< 614419.1 110     8 1 (0.07272727 0.92727273) *
                      1727) AGE_BKT=26-30,36-40,46-50 369    31 1 (0.08401084 0.91598916) *
            27) BALANCE< 10171.98 391    40 1 (0.10230179 0.89769821) *
          7) SCR>=552.5 4603  1441 1 (0.31305670 0.68694330)
           14) BALANCE>=538766 855   391 1 (0.45730994 0.54269006)
             28) HOLDING_PERIOD>=16.5 104     0 0 (1.00000000 0.00000000) *
             29) HOLDING_PERIOD< 16.5 751   287 1 (0.38215712 0.61784288)
               58) LEN_OF_RLTN_IN_MNTH< 58.5 47     5 0 (0.89361702 0.10638298) *
               59) LEN_OF_RLTN_IN_MNTH>=58.5 704   245 1 (0.34801136 0.65198864)
                118) LEN_OF_RLTN_IN_MNTH>=185 105    31 0 (0.70476190 0.29523810)
                  236) AGE>=24.5 74     7 0 (0.90540541 0.09459459) *
                  237) AGE< 24.5 31     7 1 (0.22580645 0.77419355) *
                119) LEN_OF_RLTN_IN_MNTH< 185 599   171 1 (0.28547579 0.71452421)
                  238) AVG_AMT_PER_ATM_TXN< 2695 20     0 0 (1.00000000 0.00000000) *
                  239) AVG_AMT_PER_ATM_TXN>=2695 579   151 1 (0.26079447 0.73920553)
                    478) AMT_ATM_DR>=17950 130    62 1 (0.47692308 0.52307692)
                      956) AGE_BKT=<25,>50,36-40,41-45 42     4 0 (0.90476190 0.09523810) *
                      957) AGE_BKT=26-30,31-35,46-50 88    24 1 (0.27272727 0.72727273) *
                    479) AMT_ATM_DR< 17950 449    89 1 (0.19821826 0.80178174)
                      958) AMT_NET_DR>=734444 21     5 0 (0.76190476 0.23809524) *
                      959) AMT_NET_DR< 734444 428    73 1 (0.17056075 0.82943925)
                       1918) TOT_NO_OF_L_TXNS< 20.5 181    55 1 (0.30386740 0.69613260)
                         3836) AMT_ATM_DR>=14450 21     0 0 (1.00000000 0.00000000) *
                         3837) AMT_ATM_DR< 14450 160    34 1 (0.21250000 0.78750000)
                           7674) AVG_AMT_PER_CHQ_TXN>=18610 25     5 0 (0.80000000 0.20000000) *
                           7675) AVG_AMT_PER_CHQ_TXN< 18610 135    14 1 (0.10370370 0.89629630) *
                       1919) TOT_NO_OF_L_TXNS>=20.5 247    18 1 (0.07287449 0.92712551) *
           15) BALANCE< 538766 3748  1050 1 (0.28014941 0.71985059)
             30) TOT_NO_OF_L_TXNS< 53.5 3122   974 1 (0.31197950 0.68802050)
               60) FLG_HAS_CC< 0.5 1843   687 1 (0.37276180 0.62723820)
                120) OCCUPATION=SAL 343   122 0 (0.64431487 0.35568513)
                  240) AVG_AMT_PER_ATM_TXN< 13300 169    27 0 (0.84023669 0.15976331) *
                  241) AVG_AMT_PER_ATM_TXN>=13300 174    79 1 (0.45402299 0.54597701)
                    482) BALANCE< 106410.3 66    15 0 (0.77272727 0.22727273) *
                    483) BALANCE>=106410.3 108    28 1 (0.25925926 0.74074074)
                      966) AGE_BKT=>50,36-40,41-45 21     7 0 (0.66666667 0.33333333) *
                      967) AGE_BKT=26-30,31-35,46-50 87    14 1 (0.16091954 0.83908046) *
                121) OCCUPATION=PROF,SELF-EMP,SENP 1500   466 1 (0.31066667 0.68933333)
                  242) AGE< 41.5 776   317 1 (0.40850515 0.59149485)
                    484) HOLDING_PERIOD>=8.5 348   147 0 (0.57758621 0.42241379)
```

```
              968) AMT_BR_CSH_WDL_DR< 810905 245    69 0 (0.71836735 0.28163265)
               1936) BALANCE>=69414.92 121     6 0 (0.95041322 0.04958678) *
               1937) BALANCE< 69414.92 124    61 1 (0.49193548 0.50806452)
                 3874) AVG_AMT_PER_CSH_WDL_TXN>=74447.17 45     0 0 (1.00000000 0.00000000) *
                 3875) AVG_AMT_PER_CSH_WDL_TXN< 74447.17 79    16 1 (0.20253165 0.79746835) *
              969) AMT_BR_CSH_WDL_DR>=810905 103    25 1 (0.24271845 0.75728155)
               1938) AMT_ATM_DR>=12300 35    14 0 (0.60000000 0.40000000) *
               1939) AMT_ATM_DR< 12300 68     4 1 (0.05882353 0.94117647) *
            485) HOLDING_PERIOD< 8.5 428   116 1 (0.27102804 0.72897196)
              970) LEN_OF_RLTN_IN_MNTH>=49 332   113 1 (0.34036145 0.65963855)
               1940) LEN_OF_RLTN_IN_MNTH< 70.5 27     5 0 (0.81481481 0.18518519) *
               1941) LEN_OF_RLTN_IN_MNTH>=70.5 305    91 1 (0.29836066 0.70163934)
                 3882) TOT_NO_OF_L_TXNS>=17.5 173    68 1 (0.39306358 0.60693642)
                   7764) NO_OF_NET_DR_TXNS< 3.5 74    26 0 (0.64864865 0.35135135) *
                   7765) NO_OF_NET_DR_TXNS>=3.5 99    20 1 (0.20202020 0.79797980) *
                 3883) TOT_NO_OF_L_TXNS< 17.5 132    23 1 (0.17424242 0.82575758) *
              971) LEN_OF_RLTN_IN_MNTH< 49 96     3 1 (0.03125000 0.96875000) *
         243) AGE>=41.5 724   149 1 (0.20580110 0.79419890)
           486) LEN_OF_RLTN_IN_MNTH>=198.5 30     6 0 (0.80000000 0.20000000) *
           487) LEN_OF_RLTN_IN_MNTH< 198.5 694   125 1 (0.18011527 0.81988473)
             974) GENDER=F 69    33 1 (0.47826087 0.52173913) *
             975) GENDER=M,O 625    92 1 (0.14720000 0.85280000)
              1950) AMT_ATM_DR< 2250 25    10 0 (0.60000000 0.40000000) *
              1951) AMT_ATM_DR>=2250 600    77 1 (0.12833333 0.87166667)
                3902) OCCUPATION=PROF 170    39 1 (0.22941176 0.77058824)
                  7804) HOLDING_PERIOD>=8.5 35    13 0 (0.62857143 0.37142857) *
                  7805) HOLDING_PERIOD< 8.5 135    17 1 (0.12592593 0.87407407) *
                3903) OCCUPATION=SELF-EMP,SENP 430    38 1 (0.08837209 0.91162791)
                  7806) AVG_AMT_PER_NET_TXN>=196358.8 110    23 1 (0.20909091 0.79090909)
                   15612) HOLDING_PERIOD>=11.5 29    14 0 (0.51724138 0.48275862) *
                   15613) HOLDING_PERIOD< 11.5 81     8 1 (0.09876543 0.90123457) *
                  7807) AVG_AMT_PER_NET_TXN< 196358.8 320    15 1 (0.04687500 0.95312500) *
      61) FLG_HAS_CC>=0.5 1279   287 1 (0.22439406 0.77560594)
       122) AGE>=44.5 276   111 1 (0.40217391 0.59782609)
         244) HOLDING_PERIOD>=16.5 21     0 0 (1.00000000 0.00000000) *
         245) HOLDING_PERIOD< 16.5 255    90 1 (0.35294118 0.64705882)
           490) AMT_ATM_DR>=18500 46    13 0 (0.71739130 0.28260870) *
           491) AMT_ATM_DR< 18500 209    57 1 (0.27272727 0.72727273)
             982) BALANCE>=146586.3 45    15 0 (0.66666667 0.33333333) *
             983) BALANCE< 146586.3 164    27 1 (0.16463415 0.83536585) *
       123) AGE< 44.5 1003   176 1 (0.17547358 0.82452642)
         246) NO_OF_L_CR_TXNS>=5.5 805   170 1 (0.21118012 0.78881988)
           492) NO_OF_L_CR_TXNS< 7.5 42    15 0 (0.64285714 0.35714286) *
           493) NO_OF_L_CR_TXNS>=7.5 763   143 1 (0.18741809 0.81258191)
             986) OCCUPATION=PROF,SENP 298    86 1 (0.28859060 0.71140940)
              1972) NO_OF_L_CR_TXNS< 10.5 33     6 0 (0.81818182 0.18181818) *
              1973) NO_OF_L_CR_TXNS>=10.5 265    59 1 (0.22264151 0.77735849)
                3946) BALANCE>=72956.27 113    47 1 (0.41592920 0.58407080)
                  7892) TOT_NO_OF_L_TXNS>=22.5 25     0 0 (1.00000000 0.00000000) *
                  7893) TOT_NO_OF_L_TXNS< 22.5 88    22 1 (0.25000000 0.75000000) *
                3947) BALANCE< 72956.27 152    12 1 (0.07894737 0.92105263) *
             987) OCCUPATION=SAL,SELF-EMP 465    57 1 (0.12258065 0.87741935)
              1974) AMT_L_DR>=1650502 26    10 0 (0.61538462 0.38461538) *
              1975) AMT_L_DR< 1650502 439    41 1 (0.09339408 0.90660592)
                3950) AMT_L_DR< 969307.5 150    25 1 (0.16666667 0.83333333)
                  7900) AVG_AMT_PER_CHQ_TXN>=15806.25 24    11 0 (0.54166667 0.45833333) *
                  7901) AVG_AMT_PER_CHQ_TXN< 15806.25 126    12 1 (0.09523810 0.90476190) *
                3951) AMT_L_DR>=969307.5 289    16 1 (0.05536332 0.94463668) *
         247) NO_OF_L_CR_TXNS< 5.5 198     6 1 (0.03030303 0.96969697) *
    31) TOT_NO_OF_L_TXNS>=53.5 626    76 1 (0.12140575 0.87859425)
      62) BALANCE>=359018.6 20     7 0 (0.65000000 0.35000000) *
      63) BALANCE< 359018.6 606    63 1 (0.10396040 0.89603960) *
```

5.2 The CART Model Graphical Output:



5.3 Performance Measures on Training Data Set

The following model performance measures will be calculated on the training set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

5.3.1 Model Performance Measure - Rank Ordering

The rank order table is provided below:

deciles	cnt	cnt_resp	cnt_ non_resp	rrate	cum_resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
10	2604	2394	210	92%	2394	210	20%	2%	17.8
9	2448	2123	325	87%	4517	535	37%	4%	32.45
8	2578	2154	424	84%	6671	959	54%	8%	46.55
7	2371	1951	420	82%	8622	1379	70%	11%	59.02
6	2294	1777	517	78%	10399	1896	85%	15%	69.29
5	2435	1004	1431	41%	11403	3327	93%	27%	65.81
4	2561	463	2098	18%	11866	5425	97%	44%	52.48
3	2365	255	2110	11%	12121	7535	99%	61%	37.37
2	2462	141	2321	6%	12262	9856	100%	80%	19.61
1	2426	10	2416	0%	12272	12272	100%	100%	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top deciles is above 78%.
- The KS is above 69%, indicating it to be a good model.

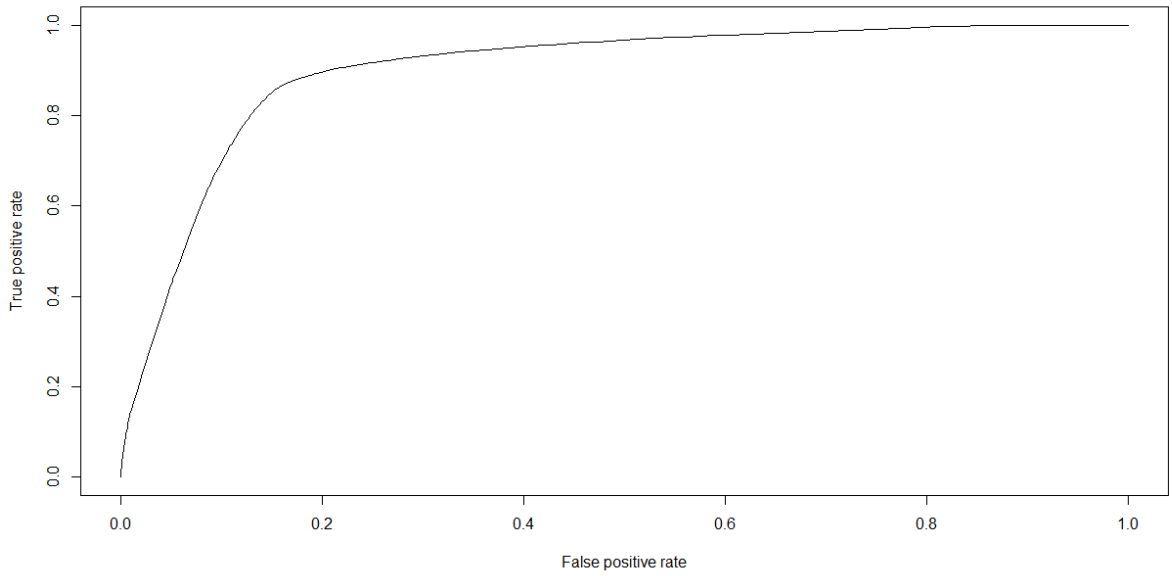
5.3.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	70.80%
Area Under Curve	90%

The AUC value around 90% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



5.3.3 Model Performance Measure – Gini Coefficient

For our given dataset and the model we built, the Gini Coefficient is 40%, indicating a good model, with scope for improvement.

5.3.4 Model Performance Measure – Confusion Matrix

Confusion Matrix for our given Model is as follows:

```
##        predict.class
## TARGET     0     1
##      0 10257  2015
##      1  1568 10704
```

Accuracy = (10257+10704)/(10257+10704+2015+1568) = 85.40%

Classification Error Rate = 1- Accuracy = 14.59%.

The lower the classification error rate, higher the model accuracy, resulting in a better model. The classification error rate can be reduced if there were more independent variables were present for modeling.

5.3.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	70.80%
Area Under Curve	90%
Gini Coefficient	40%
Accuracy	85.40%
Classification Error Rate	14.59%

The model observed to perform as per expectations on majority of the model performance measures, indicating it to be a good model.

5.4 Performance Measures on Validation Data Set

The following model performance measures will be calculated on the validation data set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

5.4.1 Model Performance Measure - Rank Ordering

deciles	cnt	cnt_resp	cnt_ non_resp	rrate	cum_resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
10	612	230	382	38%	230	382	29%	7%	22.02
9	592	217	375	37%	447	757	57%	15%	42.51
8	596	121	475	20%	568	1232	72%	24%	48.83
7	618	55	563	9%	623	1795	80%	34%	45.05
6	590	34	556	6%	657	2351	84%	45%	38.73
5	636	35	601	6%	692	2952	88%	57%	31.67
4	1193	36	1157	3%	728	4109	93%	79%	14.08
2	643	39	604	6%	767	4713	98%	90%	7.47
1	520	17	503	3%	784	5216	100%	100%	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top three deciles is 38%, 37%, 20% respectively.
- With top 3 deciles, the KS is 48%, indicating it to be a good model.

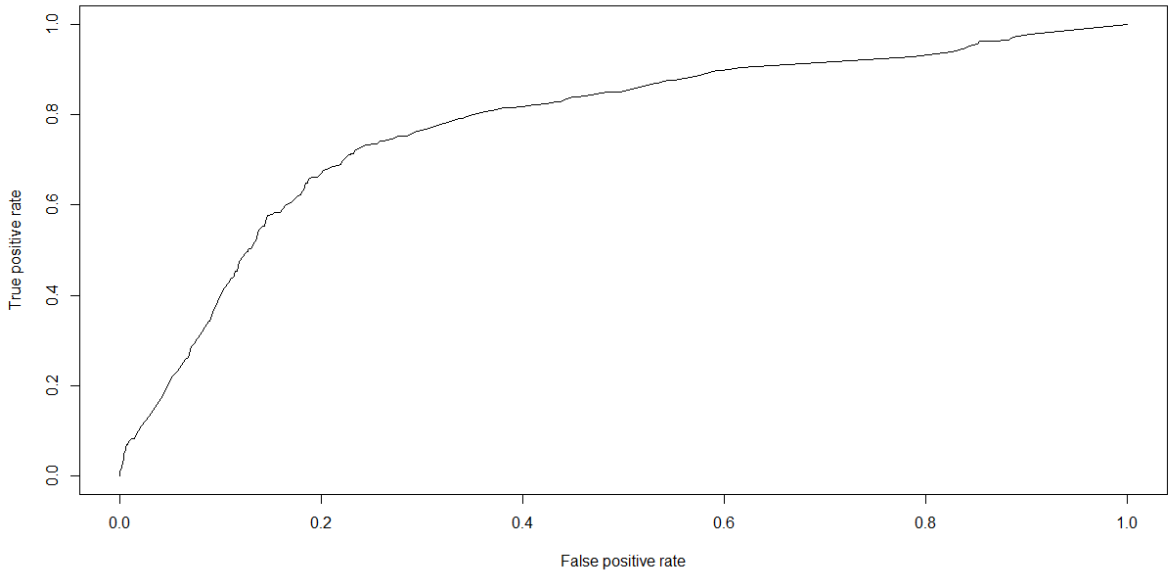
5.4.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	48.85%
Area Under Curve	77.77%

The AUC value around 77% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



5.4.3 Model Performance Measure – Gini Coefficient

For our given dataset and the model we built, the Gini Coefficient on testing data set is 54.18%, indicating a good model.

5.4.4 Model Performance Measure – Confusion Matrix

Confusion Matrix for our given Model on testing dataset is as follows:

```
##        predict.class
## TARGET     0     1
##      0  4174  1042
##      1   258   526
```

Accuracy = (4174+526)/(4174+526+1042+258) = 78.33%

Classification Error Rate = 1- Accuracy = 21.67%.

5.4.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	48.85%
Area Under Curve	77.77%
Gini Coefficient	54.18%
Accuracy	78.33%
Classification Error Rate	21.67%

The model observed to perform above expectations on majority of the model performance measures, indicating it to be a good model.

5.5 CART – Conclusion

Comparative Summary of the Random Forest Model on Training and Testing Dataset is as follows:

Measure	Result on Training DS	Result on Testing DS	% Deviation
KS	70.80%	48.85%	31%
Area Under Curve	90%	77.77%	13.58%
Gini Coefficient	40%	54.18%	-35.45%
Accuracy	85.40%	78.33%	8%
Classification Error Rate	14.59%	21.67%	50%

It can be observed that most of the Model Performance values for Training & Testing sets are above the maximum tolerance deviation of +/- 10%. Hence, the model is over-fitting.

The good performance on the model performance measures indicates good prediction making capabilities of the developed Random Forest model.

6 Model Building – Random Forest

A Supervised Classification Algorithm, as the name suggests, this algorithm creates the forest with a number of trees in random order.

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

Some advantages of using Random Forest are as follows:

- The same random forest algorithm or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will handle the missing values.
- When we have more trees in the forest, random forest classifier won't over fit the model.
- Can model the random forest classifier for categorical values also.

6.1 The initial build & Optimal No of Trees

The model is built with dependant variable as TARGET, and considering all independent variables except Customer ID and Account Opening Date.

```
# Random Forest
RF = randomForest( as.factor(TARGET) ~ .,
                   data = rf.train[, -c(1, 11)],
                   ntree = 501, mtry = 3, nodesize = 10,
                   importance = TRUE )

print(RF)

##
## Call:
##  randomForest(formula = as.factor(TARGET) ~ ., data = rf.train[,
-c(1, 11)], ntree = 501, mtry = 3, nodesize = 10, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 501
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 7.48%
## Confusion matrix:
##       0   1  class.error
## 0 12269   3 0.0002444589
## 1  1044 684 0.6041666667
```

Out of Bag Error Rate:

Random Forests algorithm is a classifier based on primarily two methods - bagging and random subspace method.

Suppose we decide to have S number of trees in our forest then we first create S datasets of "same size as original" created from random resampling

of data with-replacement. Each of these datasets is called a bootstrap dataset.

Due to "with-replacement" option, every dataset can have duplicate data records and at the same time, can be missing several data records from original datasets. This is called Bagging.

The algorithm uses m (=sqrt(M) random sub features out of M possible features to create any tree. This is called random subspace method.

After creating the classifiers (S trees), there is a subset of records which does not include any of the records part of the classifier tree. This subset, is a set of boostrap datasets which does not contain a particular record from the original dataset. This set is called out-of-bag examples. There are n such subsets (one for each data record in original dataset T). OOB classifier is the aggregation of all such records.

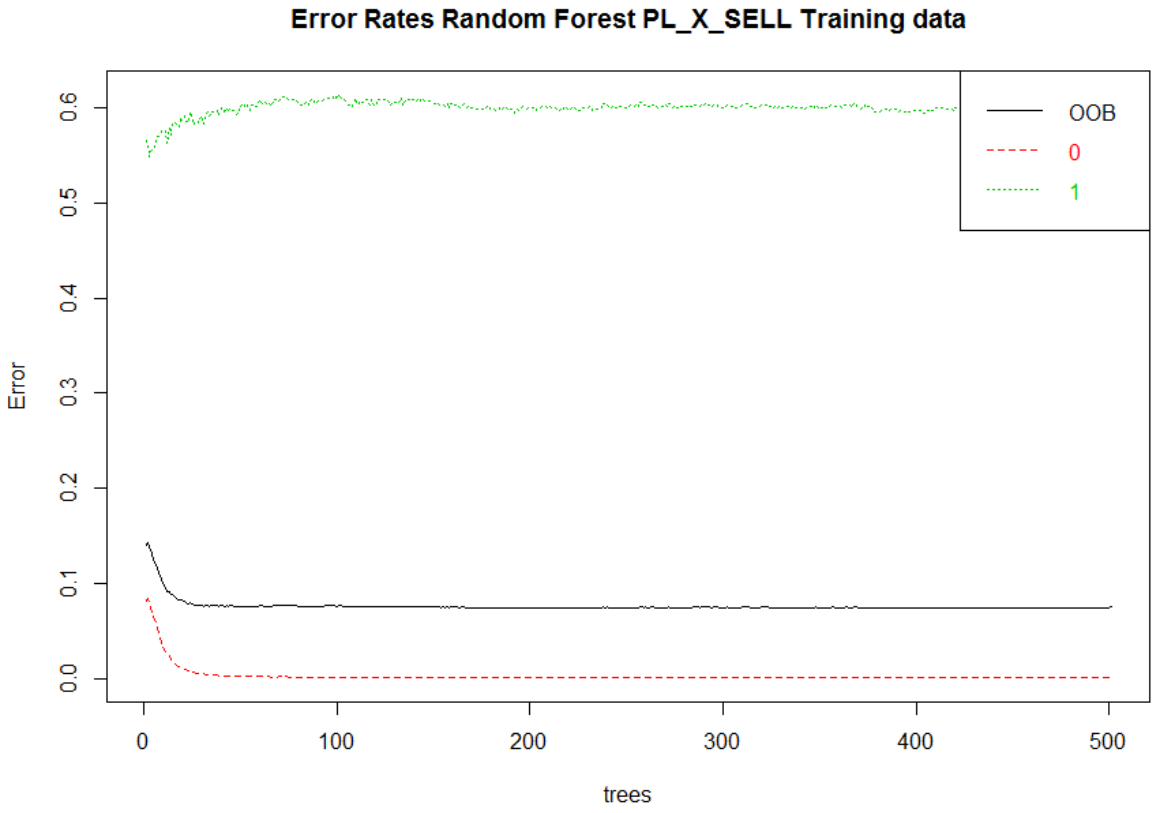
Out-of-bag estimate for the generalization error is the error rate of the out-of-bag classifier on the training set (compare it with known yi's).

Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing **bootstrap aggregating** to sub-sample data samples used for training.

Out-of-bag estimates help in avoiding the need for an independent validation dataset.

The Out-of-Bag Estimate of Error Rate for our given Random Forest in our case is 7.48%

The graphical output for the OOB estimate of error rate is provided below:



The output in tabular form for the OOB estimate of error rate is provided below:

```
RF$err.rate

##                OOB            0         1
##   [1,] 0.13971014 8.109890e-02 0.5664000
##   [2,] 0.14284002 8.402787e-02 0.5623782
##   [3,] 0.13861004 7.997794e-02 0.5490347
##   [4,] 0.13105510 7.010350e-02 0.5577580
##   [5,] 0.12556805 6.484176e-02 0.5572075
##   [6,] 0.11994485 5.752251e-02 0.5615337
##   [7,] 0.11741464 5.334808e-02 0.5707405
##   [8,] 0.11059671 4.577849e-02 0.5704938
##   [9,] 0.10496072 3.898474e-02 0.5750591
##  [10,] 0.09914011 3.189385e-02 0.5777126
.

.

## [150,] 0.07500000 4.074316e-04 0.6047454
## [151,] 0.07500000 4.074316e-04 0.6047454
## [152,] 0.07507143 4.074316e-04 0.6053241
## [153,] 0.07478571 3.259452e-04 0.6035880
## [154,] 0.07457143 3.259452e-04 0.6018519
## [155,] 0.07471429 3.259452e-04 0.6030093
## [156,] 0.07457143 3.259452e-04 0.6018519
## [157,] 0.07485714 3.259452e-04 0.6041667
```

```
## [158,] 0.07428571 3.259452e-04 0.5995370
## [159,] 0.07492857 3.259452e-04 0.6047454
## [160,] 0.07500000 3.259452e-04 0.6053241
.

.

## [490,] 0.07378571 1.629726e-04 0.5966435
## [491,] 0.07371429 8.148631e-05 0.5966435
## [492,] 0.07385714 8.148631e-05 0.5978009
## [493,] 0.07392857 1.629726e-04 0.5978009
## [494,] 0.07407143 1.629726e-04 0.5989583
## [495,] 0.07428571 1.629726e-04 0.6006944
## [496,] 0.07435714 1.629726e-04 0.6012731
## [497,] 0.07435714 1.629726e-04 0.6012731
## [498,] 0.07414286 1.629726e-04 0.5995370
## [499,] 0.07428571 1.629726e-04 0.6006944
## [500,] 0.07428571 1.629726e-04 0.6006944
## [501,] 0.07478571 2.444589e-04 0.6041667
```

It is observed that as the number of tress increases, the OOB error rate starts decreasing till it reaches around 150th tree with OOB = 0.074 (the minimum value). After this, the OOB doesn't decrease further and remains largely steady. **Hence, the optimal number of trees would be 155.**

6.2 Variable Importance

To understand the important variables in Random Forest, the following measures are generally used:

- Mean Decrease in Accuracy is based on permutation
  - Randomly permute values of a variable for which importance is to be computed in the OOB sample
  - Compute the Error Rate with permuted values
  - Compute decrease in OOB Error rate (Permuted - Not permuted)
  - Average the decrease over all the trees
- Mean Decrease in Gini is computed as "total decrease in node impurities from splitting on the variable, averaged over all trees"

The variables importance is computed as follows:

```
# List the importance of the variables.
impVar <- round(randomForest::importance(RF), 2)
impVar[order(impVar[,3], decreasing=TRUE),]
```

```
##                            0     1 MeanDecreaseAccuracy MeanDecreaseGini
## OCCUPATION             36.56 41.67                42.80            65.24
## AGE_BKT                36.41 49.59                47.73            88.57
## BALANCE                35.16 43.35                43.08           121.90
## AMT_L_DR               34.80 29.52                40.10           102.69
## SCR                    33.89 42.89                41.49           117.55
## AGE                    32.62 39.85                42.05            75.89
## LEN_OF_RLTN_IN_MNTH    31.94 44.77                43.96            99.28
## NO_OF_L_CR_TXNS        31.50 29.55                36.66            96.31
## HOLDING_PERIOD         30.52 37.74                37.31            97.33
```

```
## FLG_HAS_CC              30.27 34.71                34.74             30.04
## TOT_NO_OF_L_TXNS        29.70 24.42                34.84             94.60
## AMT_CHQ_DR              28.12 25.77                31.54             75.57
## AVG_AMT_PER_CHQ_TXN     27.90 22.98                31.85             75.09
## AVG_AMT_PER_CSH_WDL_TXN  26.68 38.33               33.33             85.45
## AMT_BR_CSH_WDL_DR       26.66 33.93                33.12             86.13
## AVG_AMT_PER_ATM_TXN     21.40 22.50                26.27             82.69
## AMT_ATM_DR              21.37 21.45                26.17             84.56
## NO_OF_L_DR_TXNS         20.64 22.74                22.81             58.80
## NO_OF_BR_CSH_WDL_DR_TXNS 20.13 27.91               25.25             43.80
## NO_OF_CHQ_DR_TXNS          19.63 21.94             22.17             36.75
## GENDER                     17.80 20.62             20.45             20.53
## FLG_HAS_OLD_LOAN           17.65 25.31             24.81             14.80
## AMT_NET_DR                 17.46 19.72             20.92             61.53
## NO_OF_ATM_DR_TXNS          16.90  6.78             17.50             26.22
## AVG_AMT_PER_NET_TXN        16.40 18.94             20.21             62.40
## FLG_HAS_ANY_CHGS           15.53 21.60             20.47             13.27
## NO_OF_NET_DR_TXNS          13.95 17.61             16.37             21.00
## ACC_TYPE                   13.67 12.47             16.04             12.84
## FLG_HAS_NOMINEE            13.12 18.45             18.46              9.91
## AMT_MOB_DR                 13.00 15.82             16.38             39.44
## AVG_AMT_PER_MOB_TXN        11.85 16.55             15.11             39.16
## NO_OF_OW_CHQ_BNC_TXNS       9.37 19.07             17.18              8.43
## NO_OF_IW_CHQ_BNC_TXNS       8.95 17.50             17.68              8.00
## NO_OF_MOB_DR_TXNS           7.64  8.10              8.78              9.76
## AMT_MIN_BAL_NMC_CHGS        5.60 10.04              9.89              2.90
## AMT_OTH_BK_ATM_USG_CHGS     1.34  4.32              3.20              1.37
## random                     -1.49 -0.61             -1.60             57.38
```

The variable importance order (Top 10) on the basis of **Mean Decrease Accuracy** would be as follows:

```
##                           0     1 MeanDecreaseAccuracy MeanDecreaseGini
## AGE_BKT                36.41 49.59                47.73             88.57
## LEN_OF_RLTN_IN_MNTH    31.94 44.77                43.96             99.28
## BALANCE                35.16 43.35                43.08            121.90
## OCCUPATION             36.56 41.67                42.80             65.24
## AGE                    32.62 39.85                42.05             75.89
## SCR                    33.89 42.89                41.49            117.55
## AMT_L_DR               34.80 29.52                40.10            102.69
## HOLDING_PERIOD         30.52 37.74                37.31             97.33
## NO_OF_L_CR_TXNS        31.50 29.55                36.66             96.31
## TOT_NO_OF_L_TXNS       29.70 24.42                34.84             94.60
```

Also, the variable importance order (Top 10) on the basis of **Mean Decrease Gini** would be as follows:

```
##                           0     1 MeanDecreaseAccuracy MeanDecreaseGini
## BALANCE                35.16 43.35                43.08            121.90
## SCR                    33.89 42.89                41.49            117.55
## AMT_L_DR               34.80 29.52                40.10            102.69
## LEN_OF_RLTN_IN_MNTH    31.94 44.77                43.96             99.28
## HOLDING_PERIOD         30.52 37.74                37.31             97.33
```

```
## NO_OF_L_CR_TXNS        31.50 29.55                36.66                96.31
## TOT_NO_OF_L_TXNS       29.70 24.42                34.84                94.60
## AGE_BKT                36.41 49.59                47.73                88.57
## AMT_BR_CSH_WDL_DR      26.66 33.93                33.12                86.13
## AVG_AMT_PER_CSH_WDL_TXN  26.68 38.33              33.33                85.45
```

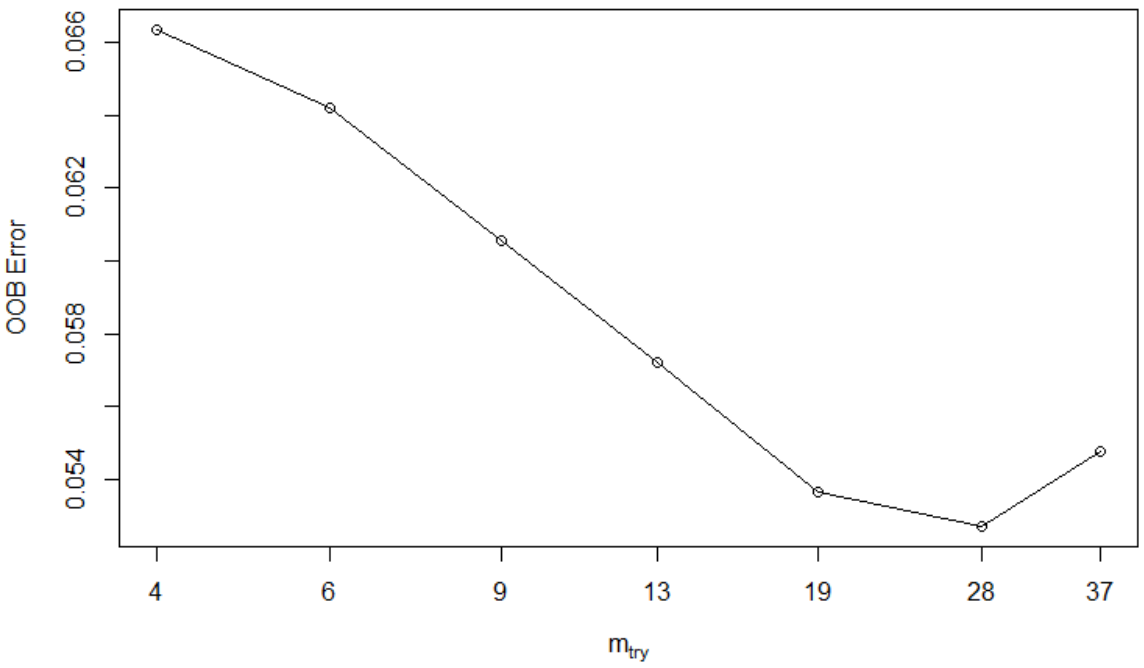
6.3 Optimal mtry value

In the random forests literature, the number of variables available for splitting at each tree node is referred to as the **mtry parameter.**

The optimum number of variables is obtained using tuneRF function as follows:

```
# Tuning Random Forest
tRF <- tuneRF(x = rf.train[,-c(1,2,11)],
              y=as.factor(rf.train$TARGET),
              mtryStart = 6, # Approx. Sqrt of Tot. No of Variables
              ntreeTry=155,
              stepFactor = 1.5,
              improve = 0.0001,
              trace=TRUE,
              plot = TRUE,
              doBest = TRUE,
              nodesize = 10,
              importance=TRUE
)
```

Output: OOB Error Vs Mtry:



As can be seen, the optimum number of Variables is 28.

6.4 Performance Measures on Training Data Set

The following model performance measures will be calculated on the training set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

6.4.1 Model Performance Measure - Rank Ordering

Rank Ordering as model performance measures helps:

- Assess the ability of the model to relatively rank the customers
- How well the model separates the Responder Class from the Non-Responder
- Track the utility of the model on an ongoing basis

The rank order table is provided below:

Sr. No.	deciles	cnt	cnt_resp	cnt_ non_resp	rrate	cum_ resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
1	10	1410	1022	388	72%	1022	388	59%	3%	0.56
2	9	1415	464	951	33%	1486	1339	86%	11%	0.75
3	8	1492	168	1324	11%	1654	2663	96%	22%	0.74
4	7	1436	46	1390	3%	1700	4053	98%	33%	0.65
5	6	1300	13	1287	1%	1713	5340	99%	44%	0.55
6	5	1446	12	1434	1%	1725	6774	100%	55%	0.45
7	4	1454	2	1452	0%	1727	8226	100%	67%	0.33
8	3	1358	1	1357	0%	1728	9583	100%	78%	0.22
9	2	2689	0	2689	0%	1728	12272	100%	100%	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top two deciles is 72% and 33% respectively.
- With top 2 deciles, the KS is 72%, indicating it to be a good model.

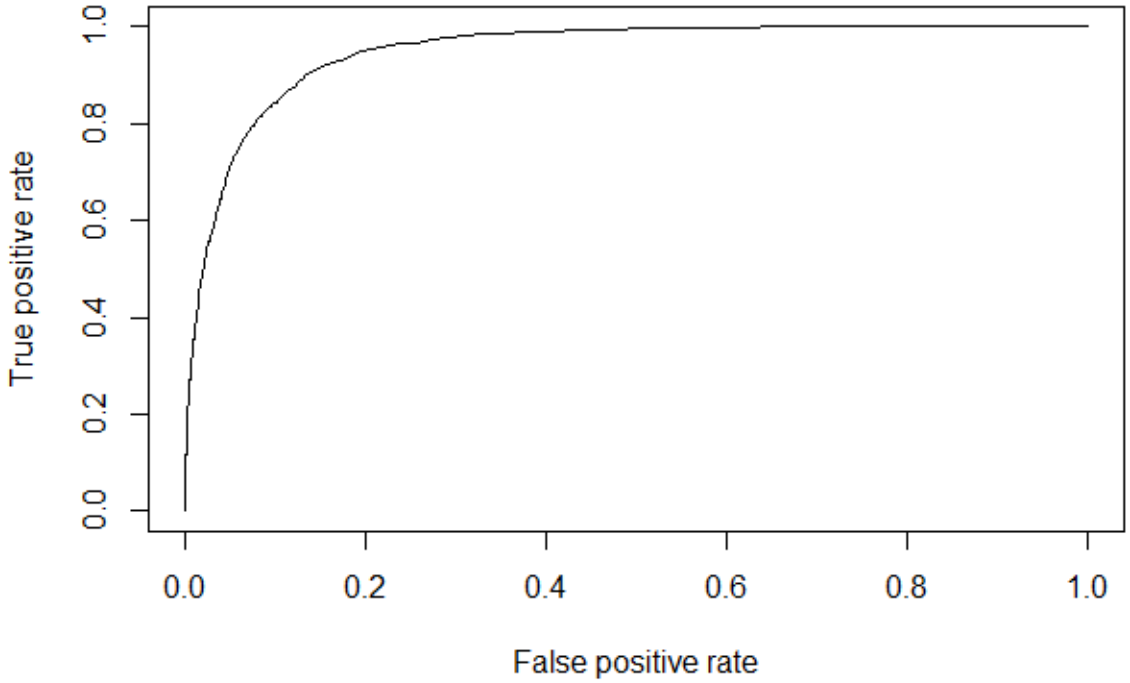
6.4.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	76.79%
Area Under Curve	94.97%

The AUC value above 90% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



6.4.3 Model Performance Measure – Gini Coefficient

The Gini Coefficient is the ratio of the area between the line of perfect equality and the observed Lorenz curve to the area between the line of perfect equality and the line of perfect inequality. The higher the coefficient, the more unequal the distribution is.

Gini coefficient can be straight away derived from the AUC ROC number.

Gini above 60% is a good model.

For our given dataset and the model we built, the Gini Coefficient is 72.29%, indicating a robust model.

6.4.4 Model Performance Measure – Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix.

Few performance parameters we can obtain with the help of confusion matrix are as follows:

- **Accuracy:** the proportion of the total number of predictions that were correct.

- **Positive Predictive Value or Precision:** the proportion of positive cases that were correctly identified.
- **Negative Predictive Value:** the proportion of negative cases that were correctly identified.
- **Sensitivity or Recall:** the proportion of actual positive cases which are correctly identified.
- **Specificity:** the proportion of actual negative cases which are correctly identified.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Confusion Matrix for our given Model is as follows:

```
##       predict.class
## TARGET     0     1
##      0 12262    10
##      1  1533   195
```

Accuracy = (12262+195)/(12262+10+1533+195) = 88.97%

Classification Error Rate = 1- Accuracy = 11%.

The lower the classification error rate, higher the model accuracy, resulting in a better model. The classification error rate can be reduced if there were more independent variables were present for modeling.

6.4.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	76.79%
Area Under Curve	94.97%
Gini Coefficient	72.29%
Accuracy	88.97%
Classification Error Rate	11.00%

The model observed to perform above expectations on majority of the model performance measures, indicating it to be a good model.

6.5 Performance Measures on Validation Data Set

The following model performance measures will be calculated on the validation data set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

6.5.1 Model Performance Measure - Rank Ordering

Deciles	cnt	cnt_resp	cnt_ non_resp	rrate	cum_resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
10	604	332	272	55%	332	272	42%	5%	0.37
9	593	193	400	33%	525	672	67%	13%	0.54
8	612	115	497	19%	640	1169	82%	22%	0.6
7	612	56	556	9%	696	1725	89%	33%	0.56
6	706	49	657	7%	745	2382	95%	46%	0.49
5	669	24	645	4%	769	3027	98%	58%	0.4
4	438	9	429	2%	778	3456	99%	66%	0.33
3	797	4	793	1%	782	4249	100%	81%	0.19
2	969	2	967	0%	784	5216	100%	100%	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top three deciles is 55%, 33%, 19% respectively.
- With top 3 deciles, the KS is 60%, indicating it to be a good model.

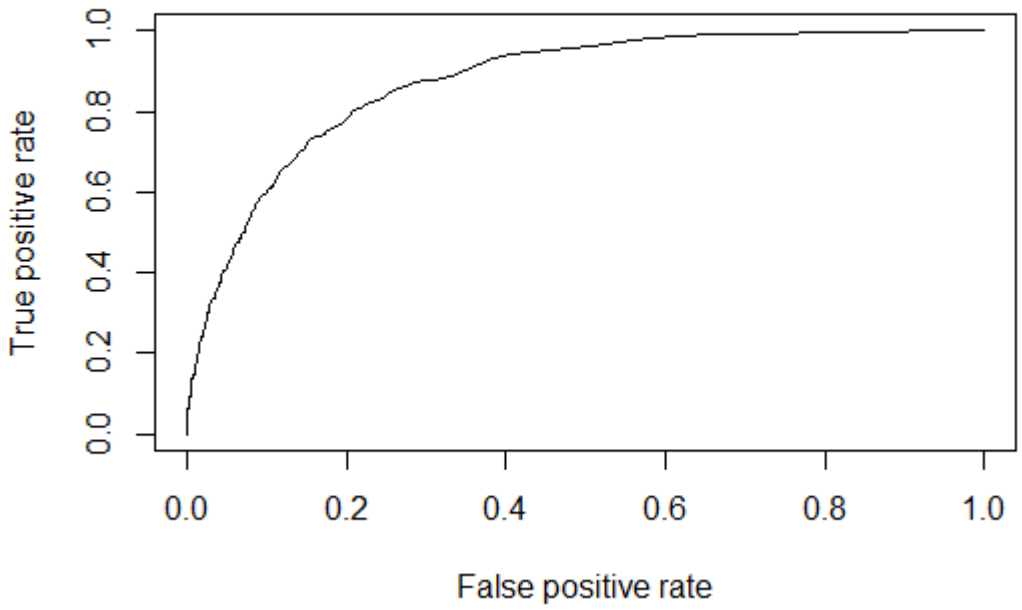
6.5.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	59.35%
Area Under Curve	87.50%

The AUC value around 90% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



6.5.3 Model Performance Measure – Gini Coefficient

For our given dataset and the model we built, the Gini Coefficient on testing data set is 71.18%, indicating a robust model.

6.5.4 Model Performance Measure – Confusion Matrix

Confusion Matrix for our given Model on testing dataset is as follows:

```
##        predict.class
## TARGET     0     1
##      0  5208     8
##      1   730    54
```

Accuracy = (5208+54)/(5208+8+730+54) = 87.58%

Classification Error Rate = 1- Accuracy = 12.41%.

6.5.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	59.35%
Area Under Curve	87.50
Gini Coefficient	71.18

Accuracy	87.58%
Classification Error Rate	12.41%

The model observed to perform above expectations on majority of the model performance measures, indicating it to be a good model.

6.6 Random Forest – Conclusion

Comparative Summary of the Random Forest Model on Training and Testing Dataset is as follows:

Measure	Result on Training DS	Result on Testing DS	% Deviation
KS	76.79%	59.35%	22.71%
Area Under Curve	94.97%	87.50%	7.87%
Gini Coefficient	72.29%	71.18%	1.54%
Accuracy	88.97%	87.58%	1.56%
Classification Error Rate	11.00%	12.41%	-12.82%

It can be observed that most of the Model Performance values for Training & Testing sets are within the maximum tolerance deviation of +/- 10%. Hence, the model is not over-fitting.

At an overall level, the model observed to perform above expectations on the various model performance measures such as KS value > 40, Gini coefficient value > 50 etc.

The good performance on the model performance measures indicates good prediction making capabilities of the developed Random Forest model.

7 Model Building – Artificial Neural Network

Artificial neural networks (ANNs) are statistical models directly inspired by, and partially modeled on biological neural networks. They are capable of modeling and processing nonlinear relationships between inputs and outputs in parallel.

Artificial neural networks are characterized by containing adaptive weights along paths between neurons that can be tuned by a learning algorithm that learns from observed data in order to improve the model. In addition to the learning algorithm itself, one must choose an appropriate cost function.

The cost function is what's used to learn the optimal solution to the problem being solved. This involves determining the best values for all of the tuneable model parameters, with neuron path adaptive weights being the primary target, along with algorithm tuning parameters such as the learning rate. It's usually done through optimization techniques such as gradient descent or stochastic gradient descent.

These optimization techniques basically try to make the ANN solution be as close as possible to the optimal solution, which when successful means that the ANN is able to solve the intended problem with high performance.

7.1 Creation of Dummy variables

Create Dummy Variables for Categorical Variables.

```
# Create Dummy Variables for Categorical Variables
#
# Gender
GEN.matrix <- model.matrix(~ GENDER - 1, data = NNInput)
NNInput <- data.frame(NNInput, GEN.matrix)
#
# Occupation
occ.matrix <- model.matrix(~ OCCUPATION - 1, data = NNInput)
NNInput <- data.frame(NNInput, occ.matrix)
#
# AGE_BKT
AGEBKT.matrix <- model.matrix(~ AGE_BKT - 1, data = NNInput)
NNInput <- data.frame(NNInput, AGEBKT.matrix)
#
# ACC_TYPE
ACCTYP.matrix <- model.matrix(~ ACC_TYPE - 1, data = NNInput)
NNInput <- data.frame(NNInput, ACCTYP.matrix)

#
dim(NNInput)

## [1] 20000    56
```

7.2 Scaling of the variables

```
#
# Scaling the Dataset and Variables
#
#names(NN.train.data)

x <- subset(NN.train.data,
            select = c("AGE",
              "BALANCE",
              "SCR",
              "HOLDING_PERIOD",
              "LEN_OF_RLTN_IN_MNTH",
              "NO_OF_L_CR_TXNS",
              "NO_OF_L_DR_TXNS",
              "TOT_NO_OF_L_TXNS",
              "NO_OF_BR_CSH_WDL_DR_TXNS",
              "NO_OF_ATM_DR_TXNS",
              "NO_OF_NET_DR_TXNS",
              "NO_OF_MOB_DR_TXNS",
              "NO_OF_CHQ_DR_TXNS",
              "FLG_HAS_CC",
              "AMT_ATM_DR",
              "AMT_BR_CSH_WDL_DR",
              "AMT_CHQ_DR",
              "AMT_NET_DR",
              "AMT_MOB_DR",
              "AMT_L_DR",
              "FLG_HAS_ANY_CHGS",
              "AMT_OTH_BK_ATM_USG_CHGS",
              "AMT_MIN_BAL_NMC_CHGS",
              "NO_OF_IW_CHQ_BNC_TXNS",
              "NO_OF_OW_CHQ_BNC_TXNS",
              "AVG_AMT_PER_ATM_TXN",
              "AVG_AMT_PER_CSH_WDL_TXN",
              "AVG_AMT_PER_CHQ_TXN",
              "AVG_AMT_PER_NET_TXN",
              "AVG_AMT_PER_MOB_TXN",
              "FLG_HAS_NOMINEE",
              "FLG_HAS_OLD_LOAN",
              "random",
              "GENDERF",
              "GENDERM",
              "GENDERO",
              "OCCUPATIONPROF",
              "OCCUPATIONSAL",
              "OCCUPATIONSELF.EMP",
              "OCCUPATIONSENP",
              "AGE_BKT.25",
              "AGE_BKT.50",
              "AGE_BKT26.30",
              "AGE_BKT31.35",
              "AGE_BKT36.40",
              "AGE_BKT41.45",
              "AGE_BKT46.50",
              "ACC_TYPECA",
```

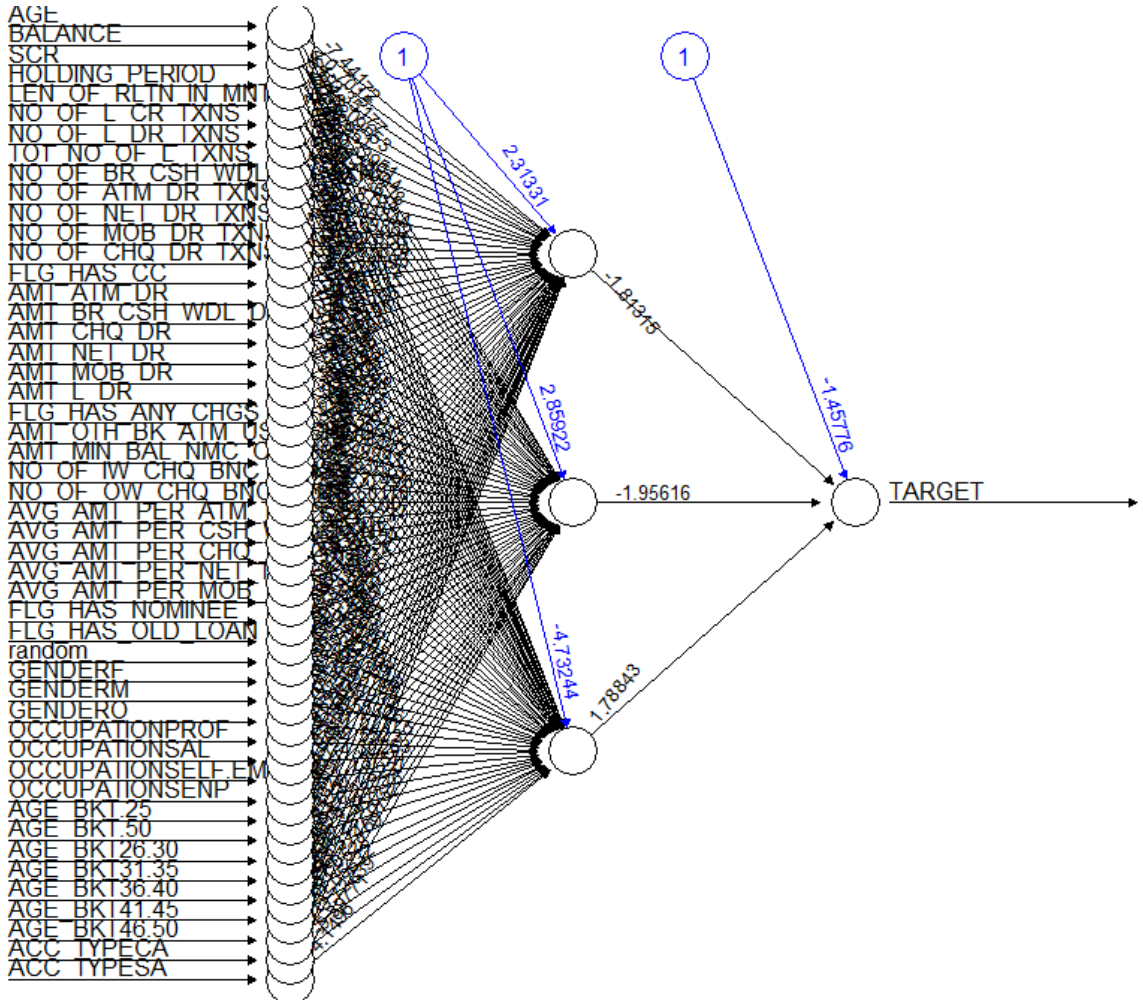
```
                "ACC_TYPESA"
                )
)
#
nn.devscaled <- scale(x)
nn.devscaled <- cbind(NN.train.data[2], nn.devscaled)
# names(nn.devscaled)
```

7.3 Building the Artificial Neural Network

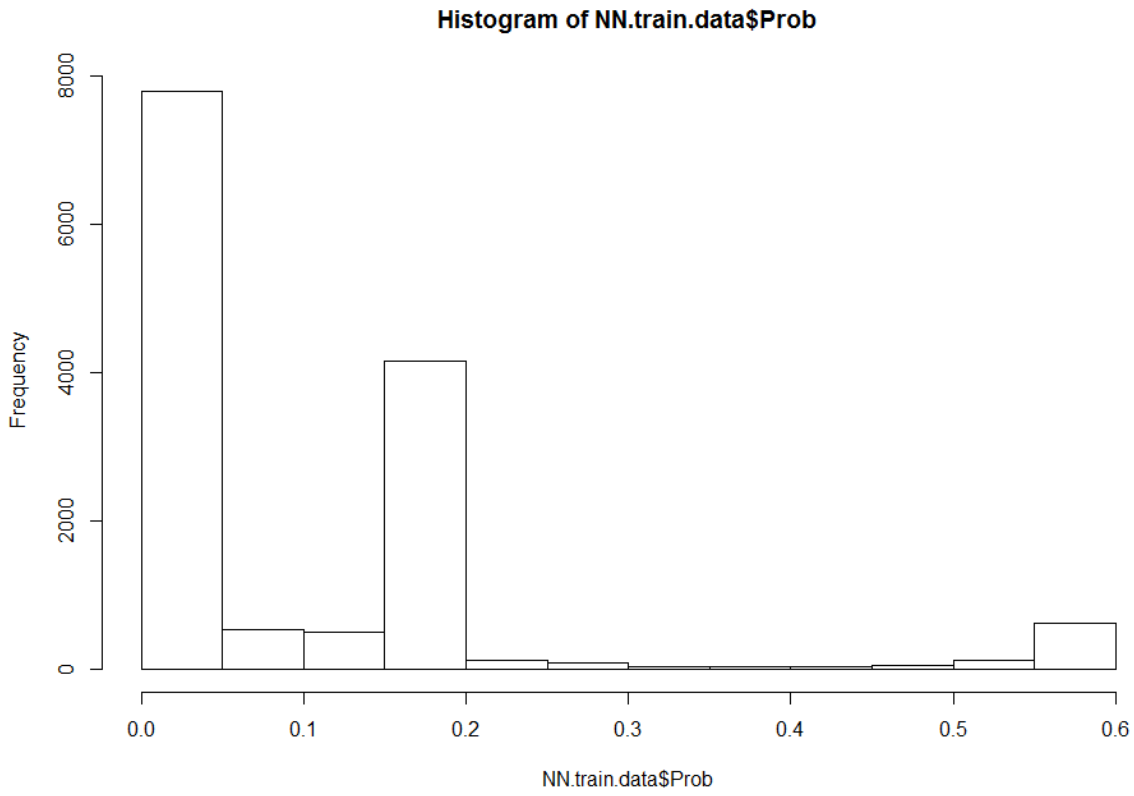
```
#
nn2 <- neuralnet(formula = TARGET ~
                    AGE +
                    BALANCE +
                    SCR +
                    HOLDING_PERIOD +
                    LEN_OF_RLTN_IN_MNTH +
                    NO_OF_L_CR_TXNS +
                    NO_OF_L_DR_TXNS +
                    TOT_NO_OF_L_TXNS +
                    NO_OF_BR_CSH_WDL_DR_TXNS +
                    NO_OF_ATM_DR_TXNS +
                    NO_OF_NET_DR_TXNS +
                    NO_OF_MOB_DR_TXNS +
                    NO_OF_CHQ_DR_TXNS +
                    FLG_HAS_CC +
                    AMT_ATM_DR +
                    AMT_BR_CSH_WDL_DR +
                    AMT_CHQ_DR +
                    AMT_NET_DR +
                    AMT_MOB_DR +
                    AMT_L_DR +
                    FLG_HAS_ANY_CHGS +
                    AMT_OTH_BK_ATM_USG_CHGS +
                    AMT_MIN_BAL_NMC_CHGS +
                    NO_OF_IW_CHQ_BNC_TXNS +
                    NO_OF_OW_CHQ_BNC_TXNS +
                    AVG_AMT_PER_ATM_TXN +
                    AVG_AMT_PER_CSH_WDL_TXN +
                    AVG_AMT_PER_CHQ_TXN +
                    AVG_AMT_PER_NET_TXN +
                    AVG_AMT_PER_MOB_TXN +
                    FLG_HAS_NOMINEE +
                    FLG_HAS_OLD_LOAN +
                    random +
                    GENDERF +
                    GENDERM +
                    GENDERO +
                    OCCUPATIONPROF +
                    OCCUPATIONSAL +
                    OCCUPATIONSELF.EMP +
                    OCCUPATIONSENP +
                    AGE_BKT.25 +
```

```
                  AGE_BKT.50 +
                  AGE_BKT26.30 +
                  AGE_BKT31.35 +
                  AGE_BKT36.40 +
                  AGE_BKT41.45 +
                  AGE_BKT46.50 +
                  ACC_TYPECA +
                  ACC_TYPESA ,
                data = nn.devscaled,
                hidden = 3,
                err.fct = "sse",
                linear.output = FALSE,
                lifesign = "full",
                lifesign.step = 10,
                threshold = 0.1,
                stepmax = 2000
)
```

7.4 The Artificial Neural Network – Graphical Representation



7.5 Histogram of Probabilities in Training Dataset



7.6 Performance Measures on Training Data Set

The following model performance measures will be calculated on the training set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

7.6.1 Model Performance Measure - Rank Ordering

The rank order table is provided below:

Sr. No.	deciles	cnt	cnt_ resp	cnt_ non_resp	rrate	cum_ resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
1	10	1400	549	851	0.39	549	851	0.32	0.07	0.25
2	9	1400	251	1149	0.18	800	2000	0.46	0.16	0.3
3	8	1400	251	1149	0.18	1051	3149	0.61	0.26	0.35
4	7	1400	228	1172	0.16	1279	4321	0.74	0.35	0.39
5	6	1400	62	1338	0.04	1341	5659	0.78	0.46	0.32
6	5	1400	74	1326	0.05	1415	6985	0.82	0.57	0.25
7	4	1400	72	1328	0.05	1487	8313	0.86	0.68	0.18
8	3	1400	96	1304	0.07	1583	9617	0.92	0.78	0.14

9	2	1400	85	1315	0.06	1668	10932	0.97	0.89	0.08
10	1	1400	60	1340	0.04	1728	12272	1	1	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top three deciles is 39%, 18% and 18%respectively.
- With top 4 deciles, the KS is 39%, which is close to a good fitness model indicator..

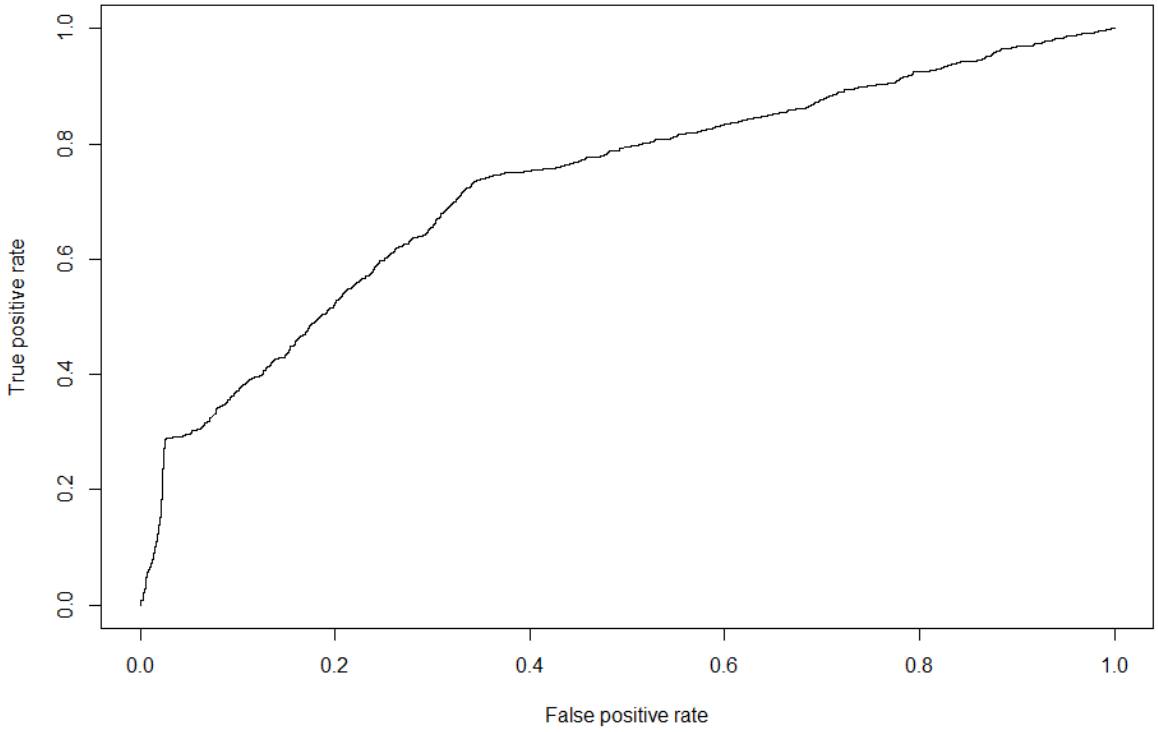
7.6.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	39%
Area Under Curve	73%

The AUC value of 73% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



7.6.3 Model Performance Measure – Gini Coefficient

The Gini Coefficient is the ratio of the area between the line of perfect equality and the observed Lorenz curve to the area between the line of

perfect equality and the line of perfect inequality. The higher the coefficient, the more unequal the distribution is.

Gini coefficient can be straight away derived from the AUC ROC number.

Gini above 60% is a good model.

For our given dataset and the model we built, the Gini Coefficient is 54.70%.

7.6.4 Model Performance Measure – Confusion Matrix

Confusion Matrix for our given Model is as follows:

```
##       predict.class
## TARGET     0     1
##      0 11981   291
##      1  1296   432
```

Accuracy = (11981+432)/(11981+291+1296+432) = 88.66%

Classification Error Rate = 1- Accuracy = 11%.

The lower the classification error rate, higher the model accuracy, resulting in a better model. The classification error rate can be reduced if there were more independent variables were present for modeling.

7.6.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	39%
Area Under Curve	73%
Gini Coefficient	54.70%
Accuracy	88.66%
Classification Error Rate	11.00%

The model observed to perform at par expectations on majority of the model performance measures, indicating it to be a good model, with scope for improvement.

7.7 Performance Measures on Validation Data Set

The following model performance measures will be calculated on the validation data set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient

- Classification Error

7.7.1 Model Performance Measure - Rank Ordering

deciles	cnt	cnt_ resp	cnt_ non_resp	rrate	cum_ resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
10	600	226	374	38%	226	374	29%	7%	0.22
9	600	118	482	20%	344	856	44%	16%	0.28
8	600	103	497	17%	447	1353	57%	26%	0.31
7	600	103	497	17%	550	1850	70%	35%	0.35
6	600	53	547	9%	603	2397	77%	46%	0.31
5	600	48	552	8%	651	2949	83%	57%	0.26
4	600	42	558	7%	693	3507	88%	67%	0.21
3	600	33	567	6%	726	4074	93%	78%	0.15
2	600	35	565	6%	761	4639	97%	89%	0.08
1	600	23	577	4%	784	5216	100%	100%	0

Interpretation:

- The baseline Response Rate is 12.34%, whereas the response rate in top three deciles is 38%, 20%, 17% respectively.
- With top 4 deciles, the KS is 35%, indicating scope for improvement.

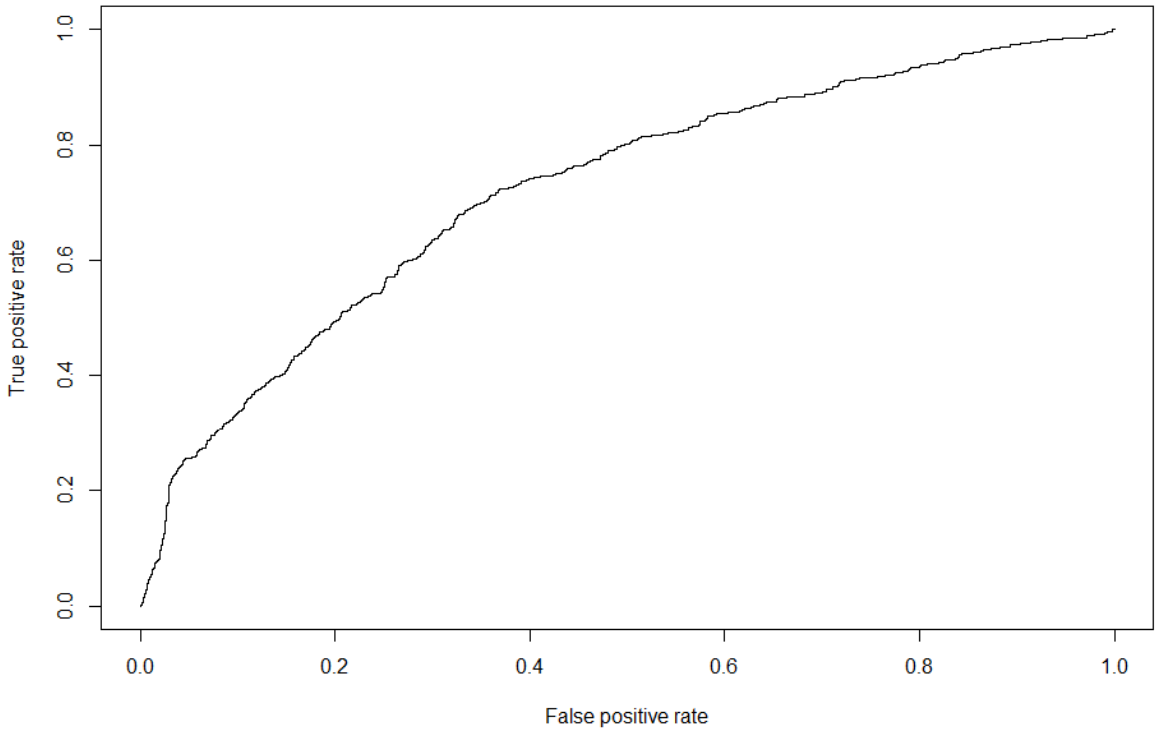
7.7.2 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	35.45%
Area Under Curve	72.13%

The AUC value around 72% indicates the good performance of the model, with further scope for improvement.

Graphical representation of the Area Under Curve is as follows:



7.7.3 Model Performance Measure – Gini Coefficient

For our given dataset and the model we built, the Gini Coefficient on testing data set is 54.62%.

7.7.4 Model Performance Measure – Confusion Matrix

Confusion Matrix for our given Model on testing dataset is as follows:

```
##        predict.class
## TARGET     0     1
##      0  5065   151
##      1   639   145
```

Accuracy = (5065+145)/(5065+145+151+639) = 86.83%

Classification Error Rate = 1- Accuracy = 13.16%.

7.7.5 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Result
KS	35.45%

Area Under Curve	72.13%
Gini Coefficient	54.62%
Accuracy	86.83%
Classification Error Rate	13.16%

The model observed to perform "normal" on majority of the model performance measures, indicating it to be a good model, with scope for improvement.

7.8 Artificial Neural Network – Conclusion

Comparative Summary of the Neural Network Model on Training and Testing Dataset is as follows:

Measure	Result on Training DS	Result on Testing DS	% Deviation
KS	39%	35.45%	9.10%
Area Under Curve	73%	72.13%	1.19%
Gini Coefficient	54.70%	54.62%	0.14%
Accuracy	88.66%	86.83%	2.06%
Classification Error Rate	11.00%	13.16%	-19.63%

It can be observed that most of the Model Performance values for Training & Testing sets are within the maximum tolerance deviation of +/- 10%. Hence, the model is not over-fitting.

At an overall level, the model observed to perform "normal" on the various model performance measures such as KS value close to 40, Gini coefficient value > 50 etc.

The good performance on the model performance measures indicates good prediction making capabilities of the developed Neural Network model.

8 Model Comparison

The main objective of the project was to develop a predictive model to predict if MyBank customers will respond positively to a promotion or an offer using tools of Machine Learning. In this context, the key parameter for model evaluation was 'Accuracy', i.e., the proportion of the total number of predictions that were correct (i.e. % of the customers that were correctly predicted).

The predictive models was be developed using the following Machine Learning techniques:

- Classification Tree – CART
- Random Forest
- Neural Network

The snap shot of the performance of all the models on accuracy, over-fitting and other model performance measures is provided below:

	CART			Random Forest			Artificial Neural Network		
Measure	Training DS	Testing DS	% Dev.	Training DS	Testing DS	% Dev.	Training DS	Testing DS	% Dev
KS	71%	49%	31%	77%	59%	23%	39%	35%	9%
Area Under Curve	90%	78%	14%	95%	88%	8%	73%	72%	1%
Gini Coefficient	40%	54%	-35%	72%	71%	2%	55%	55%	0%
Accuracy	85%	78%	8%	89%	88%	2%	89%	87%	2%
Classification Error Rate	15%	22%	50%	11%	12%	-13%	11%	13%	-20%

Interpretation:

- The CART method has given poor performance compared to Random Forest and ANN. Looking at the percentage deviation between Training and Testing Dataset, it looks like the Model is over fit.
- The Random Forest method has the best performance among all the three models. The percentage deviation between Training and Testing Dataset also is reasonably under control, suggesting a robust model.
- Neural Network has given relatively secondary performance compared to Random Forest, however, better than CART. However, the percentage deviation between Training and Testing Data set is minimal among three models.
- **Random Forest seems to be the overall winner.**

9 Appendix – Sample Source Code

```
#=====================================================================
#
#   Mini Project 4: Supervised Machine Learning
#
#   1. Pre-Processing for Model Building
#   2. Model Building: CART
#   3. Model Building: Random Forest
#   4. Model Building: Neural Network
#   5. Model Comparison: CART VS RF VS NN
#
#=====================================================================
# 1. Environment Set up and Data Import
#=====================================================================
# Install Libraries and Packages
#=====================================================================
library(caret)
library(rpart)
#install.packages("rpart.plot")
library(rpart.plot)
library(randomForest)
#
# Setup Working Directory
#=====================================================================
setwd("D:\\04 Module 4 Machine Learning\\04 Week 4")
getwd()
# Read Input File
PL_X_SELL=read.csv("PL_XSELL.csv")
attach(PL_X_SELL)
# detach(PL_X_SELL)
#=====================================================================
# 2. Exploratory Data Analysis
#=====================================================================
#
# Find out Total Number of Rows and Columns
dim(PL_X_SELL)
# Find out Names of the Columns (Features)
names(PL_X_SELL)
# Find out Class of each Feature, along with internal structure
str(PL_X_SELL)
#
# Check top 6 and bottom 6 Rows of the Dataset
#head(PL_X_SELL)
#tail(PL_X_SELL)
# head(PL_X_SELL,10) # To obtain desired number of rows, here 10.
#
#Check for Missing Values
colSums(is.na(PL_X_SELL))
#
table(TARGET)
prop.table(table(TARGET))

library(plotrix)
```

```
pie3D(prop.table((table(PL_X_SELL$TARGET))),
      main='Resp Vs Non Resp in Input Data set',
      #explode=0.1,
      labels=c("Non Resp", "Resp"),
      col = c("Turquoise", "Medium Sea Green")
)


#prop.table(table(GENDER))
#prop.table(table(GENDER,TARGET),margin = 1)

# Provide Summary statistics of a Dataset.
summary(PL_X_SELL)
#

#Creating Training and Testing Datasets
set.seed(111)
trainIndex <- createDataPartition(TARGET,
                                  p = .7,
                                  list = FALSE,
                                  times = 1)

train.data <- PL_X_SELL[trainIndex,]
test.data  <- PL_X_SELL[-trainIndex,]

dim(train.data)
dim(test.data)

#Check if distribution of partition data is correct
prop.table((table(train.data$TARGET)))
#0.8749286 0.1250714
prop.table((table(test.data$TARGET)))
#0.8731667 0.1268333
#
# So the data is well distributed in the training and validation sets
# if not change the seed used above

library(plotrix)
par(mfrow=c(1,2))
pie3D(prop.table((table(train.data$TARGET))),
      main='Resp Vs Non Resp in Training Data set',
      #explode=0.1,
      labels=c("Non Resp", "Resp"),
      col = c("Turquoise", "Medium Sea Green")
      )

pie3D(prop.table((table(test.data$TARGET))),
      main='Resp Vs Non Resp in Testing Data set',
      #explode=0.1,
      labels=c("Non Resp", "Resp"),
      col = c("Aquamarine", "Dark Sea Green")
      )
#=============================================================================
# MODEL BUILDING - CART.
```

```
#=============================================================================
# Setting the control paramter inputs for rpart.
r.ctrl = rpart.control(minsplit=100,
                       minbucket = 10,
                       cp = 0, xval = 10)

# Build the Model on Training Dataset.
# Exclude Columns Customer ID and Acct Opening Date.
names(cart.train)
m1 <- rpart(formula = TARGET ~ .,
            data = cart.train[,-c(1,11)], method = "class",
#            data = cart.train[,-c(1)], method = "class",
            control = r.ctrl)
m1

## install.packages("rattle")
## install.packages("RColorBrewer")
library(rattle)
library(RColorBrewer)
# fancyRpartPlot(m1)

printcp(m1)
plotcp(m1)
# Pruning the Tree
## Pruning Code
ptree<- prune(m1, cp= 0.0017 ,"CP")
printcp(ptree)

fancyRpartPlot(ptree, uniform=TRUE,  main="Pruned Classification Tree")

fancyRpartPlot(ptree,
               uniform = TRUE,
               main = "Final Tree",
               palettes = c("Blues", "Oranges"))

# Measure Model Performance

cart.train$predict.class <- predict(ptree, cart.train, type="class")
cart.train$predict.score <- predict(ptree, cart.train, type="prob")

# head(cart.train)

## deciling code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
```

```
                                        ifelse(x<deciles[6], 6,
                                               ifelse(x<deciles[7], 7,
                                                      ifelse(x<deciles[8],
8,

ifelse(x<deciles[9], 9, 10
                                                                    ))))))))))
}

class(cart.train$predict.score)
## deciling
cart.train$deciles <- decile(cart.train$predict.score[,2])
# View(cart.train)

## Ranking code
##install.packages("data.table")
##install.packages("scales")
library(data.table)
library(scales)
tmp_DT = data.table(cart.train)
rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_resp / rank$cnt,4);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),4);
rank$cum_rel_non_resp <- round(rank$cum_non_resp /
sum(rank$cnt_non_resp),4);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp) * 100;
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

#install.packages("ROCR")
#install.packages("ineq")
library(ROCR)
library(ineq)
pred <- prediction(cart.train$predict.score[,2], cart.train$TARGET)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(cart.train$predict.score[,2], type="Gini")

with(cart.train, table(TARGET, predict.class))
auc
KS
```

```
gini

View(rank)

## Syntax to get the node path
tree.path <- path.rpart(ptree, node = c(2, 12))
nrow(cart.test)

## Scoring Holdout sample
cart.test$predict.class <- predict(ptree, cart.test, type="class")
cart.test$predict.score <- predict(ptree, cart.test, type="prob")


cart.test$deciles <- decile(cart.test$predict.score[,2])
# View(cart.test)

# Ranking code
tmp_DT = data.table(cart.test)
h_rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
h_rank$rrate <- round(h_rank$cnt_resp / h_rank$cnt,4);
h_rank$cum_resp <- cumsum(h_rank$cnt_resp)
h_rank$cum_non_resp <- cumsum(h_rank$cnt_non_resp)
h_rank$cum_rel_resp <- round(h_rank$cum_resp / sum(h_rank$cnt_resp),4);
h_rank$cum_rel_non_resp <- round(h_rank$cum_non_resp /
sum(h_rank$cnt_non_resp),4);
h_rank$ks <- abs(h_rank$cum_rel_resp - h_rank$cum_rel_non_resp)*100;
h_rank$rrate <- percent(h_rank$rrate)
h_rank$cum_rel_resp <- percent(h_rank$cum_rel_resp)
h_rank$cum_rel_non_resp <- percent(h_rank$cum_rel_non_resp)

View(h_rank)

pred <- prediction(cart.test$predict.score[,2], cart.test$TARGET)
perf <- performance(pred, "tpr", "fpr")
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(cart.test$predict.score[,2], type="Gini")

with(cart.test, table(TARGET, predict.class))
auc
KS
gini
#

# Oversampling the data (using ROSE Algorithm)
#=============================================================================
library(ROSE)
table(cart.train$TARGET)
```

```
N=12272*2
N
cart.train.over <- ovun.sample(TARGET~.,data=cart.train,
                               method="over",N=24544)$data

table(cart.train.over$TARGET)
#
#=============================================================================
# MODEL BUILDING - CART - With Oversampling
#=============================================================================
# Setting the control paramter inputs for rpart.
r.ctrl = rpart.control(minsplit=100,
                       minbucket = 20,
                       cp = 0, xval = 10)

# Build the Model on Training Dataset.
# Exclude Columns Customer ID and Acct Opening Date.
names(cart.train.over)
m1 <- rpart(formula = TARGET ~ .,
            data = cart.train.over[,-c(1,11,41,42,43)], method = "class",
            #            data = cart.train.over[,-c(1)], method = "class",
            control = r.ctrl)
m1

## install.packages("rattle")
## install.packages("RColorBrewer")
library(rattle)
##install.packages
library(RColorBrewer)
# fancyRpartPlot(m1)

printcp(m1)
plotcp(m1)
# Pruning the Tree
## Pruning Code
ptree<- prune(m1, cp= 0.00051 ,"CP")
printcp(ptree)

fancyRpartPlot(ptree, uniform=TRUE,  main="Pruned Classification Tree")

fancyRpartPlot(ptree,
               uniform = TRUE,
               main = "Final Tree",
               palettes = c("Blues", "Oranges"))

# Measure Model Performance

cart.train.over$predict.class <- predict(ptree, cart.train.over,
type="class")
cart.train.over$predict.score <- predict(ptree, cart.train.over,
type="prob")

# head(cart.train.over)
```

```
## deciling code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8],
8,

ifelse(x<deciles[9], 9, 10
                                                                    ))))))))))
}

class(cart.train.over$predict.score)
## deciling
cart.train.over$deciles <- decile(cart.train.over$predict.score[,2])
View(cart.train.over)

## Ranking code
##install.packages("data.table")
##install.packages("scales")
library(data.table)
library(scales)
tmp_DT = data.table(cart.train.over)
rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_resp / rank$cnt,4);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),4);
rank$cum_rel_non_resp <- round(rank$cum_non_resp /
sum(rank$cnt_non_resp),4);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp) * 100;
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

#install.packages("ROCR")
#install.packages("ineq")
library(ROCR)
library(ineq)
```

```
pred <- prediction(cart.train.over$predict.score[,2],
cart.train.over$TARGET)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(cart.train.over$predict.score[,2], type="Gini")

with(cart.train.over, table(TARGET, predict.class))
auc
KS
gini

View(rank)

## Syntax to get the node path
tree.path <- path.rpart(ptree, node = c(2, 12))
nrow(cart.test)

## Scoring Holdout sample
cart.test$predict.class <- predict(ptree, cart.test, type="class")
cart.test$predict.score <- predict(ptree, cart.test, type="prob")


cart.test$deciles <- decile(cart.test$predict.score[,2])
# View(cart.test)

# Ranking code
tmp_DT = data.table(cart.test)
h_rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
h_rank$rrate <- round(h_rank$cnt_resp / h_rank$cnt,4);
h_rank$cum_resp <- cumsum(h_rank$cnt_resp)
h_rank$cum_non_resp <- cumsum(h_rank$cnt_non_resp)
h_rank$cum_rel_resp <- round(h_rank$cum_resp / sum(h_rank$cnt_resp),4);
h_rank$cum_rel_non_resp <- round(h_rank$cum_non_resp /
sum(h_rank$cnt_non_resp),4);
h_rank$ks <- abs(h_rank$cum_rel_resp - h_rank$cum_rel_non_resp)*100;
h_rank$rrate <- percent(h_rank$rrate)
h_rank$cum_rel_resp <- percent(h_rank$cum_rel_resp)
h_rank$cum_rel_non_resp <- percent(h_rank$cum_rel_non_resp)

View(h_rank)

pred <- prediction(cart.test$predict.score[,2], cart.test$TARGET)
perf <- performance(pred, "tpr", "fpr")
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)
```

```
gini = ineq(cart.test$predict.score[,2], type="Gini")

with(cart.test, table(TARGET, predict.class))
auc
KS
gini
#
#============================================================================
# MODEL BUILDING - RANDOM FOREST
#============================================================================
#install.packages("randomForest")
library(randomForest)
#
# Copy datasets for RF

rf.train <- train.data
rf.test <- test.data

dim(rf.train)
dim(rf.test)
names(rf.train)

str(rf.train)

# Random Forest
RF = randomForest( as.factor(TARGET) ~ .,
                   data = rf.train[, -c(1, 11)],
                   ntree = 500, mtry = 7, nodesize = 100,
                   importance = TRUE )

print(RF)
#
dev.off()
plot(RF, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest PL_X_SELL Training data")

RF$err.rate

# List the importance of the variables.
impVar <- round(randomForest::importance(RF), 2)
# impVar[order(impVar[,3], decreasing=TRUE),]
# impVar[order(impVar[,4], decreasing=TRUE),]
impVar[order(impVar[,1], decreasing=TRUE),]

# Tuning Random Forest
tRF <- tuneRF(x = rf.train[,-c(1,2,11)],
              y=as.factor(rf.train$TARGET),
              mtryStart = 6, # Approx. Sqrt of Tot. No of Variables
              ntreeTry=51,
              stepFactor = 1.5,
              improve = 0.0001,
              trace=TRUE,
```

```
              plot = TRUE,
              doBest = TRUE,
              nodesize = 100,
              importance=TRUE
)
#
# Scoring syntax
rf.train$predict.class <- predict(tRF, rf.train, type="class")
rf.train$predict.score <- predict(tRF, rf.train, type="prob")
# head(rf.train)
# class(rf.train$predict.score)

# deciling code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8],
8,

ifelse(x<deciles[9], 9, 10
                                                                    ))))))))))
}


rf.train$deciles <- decile(rf.train$predict.score[,2])

library(data.table)
tmp_DT = data.table(rf.train)
rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round (rank$cnt_resp / rank$cnt,2);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),2);
rank$cum_rel_non_resp <- round(rank$cum_non_resp /
sum(rank$cnt_non_resp),2);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp);

library(scales)
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
```

```
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

# rank
# Baseline Response Rate
sum(rf.train$TARGET) / nrow(rf.train)


library(ROCR)
pred <- prediction(rf.train$predict.score[,2], rf.train$TARGET)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
KS

## Area Under Curve
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)
auc

## Gini Coefficient
library(ineq)
gini = ineq(rf.train$predict.score[,2], type="Gini")
gini

## Classification Error
with(rf.train, table(TARGET, predict.class))
#
#========================================================================
# Model Performance on Testing Data Set
#========================================================================
# Rank Order
#
## Scoring syntax
rf.test$predict.class <- predict(tRF, rf.test, type="class")
rf.test$predict.score <- predict(tRF, rf.test, type="prob")

rf.test$deciles <- decile(rf.test$predict.score[,2])

tmp_DT = data.table(rf.test)
h_rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
h_rank$rrate <- round (h_rank$cnt_resp / h_rank$cnt,2);
h_rank$cum_resp <- cumsum(h_rank$cnt_resp)
h_rank$cum_non_resp <- cumsum(h_rank$cnt_non_resp)
h_rank$cum_rel_resp <- round(h_rank$cum_resp / sum(h_rank$cnt_resp),2);
h_rank$cum_rel_non_resp <- round(h_rank$cum_non_resp /
sum(h_rank$cnt_non_resp),2);
h_rank$ks <- abs(h_rank$cum_rel_resp - h_rank$cum_rel_non_resp);
```

```
library(scales)
h_rank$rrate <- percent(h_rank$rrate)
h_rank$cum_rel_resp <- percent(h_rank$cum_rel_resp)
h_rank$cum_rel_non_resp <- percent(h_rank$cum_rel_non_resp)

View(h_rank)

h_rank

pred1 <- prediction(rf.test$predict.score[,2], rf.test$TARGET)
perf1 <- performance(pred1, "tpr", "fpr")
plot(perf1)
KS1 <- max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
KS1

## Area Under Curve
auc1 <- performance(pred1,"auc");
auc1 <- as.numeric(auc1@y.values)
auc1

## Gini Coefficient
library(ineq)
gini1 = ineq(rf.test$predict.score[,2], type="Gini")
gini1

## Classification Error
with(rf.test, table(TARGET, predict.class))

#
#=============================================================================
# MODEL BUILDING - NEURAL NETWORK
#=============================================================================
#
library(neuralnet)
#
# Read Input File
# detach(PL_X_SELL)
NNInput=read.csv("PL_XSELL.csv")
attach(NNInput)
dim(NNInput)
str(NNInput)
#
# Create Dummy Variables for Categorical Variables
#
# Gender
GEN.matrix <- model.matrix(~ GENDER - 1, data = NNInput)
NNInput <- data.frame(NNInput, GEN.matrix)
#
# Occupation
occ.matrix <- model.matrix(~ OCCUPATION - 1, data = NNInput)
NNInput <- data.frame(NNInput, occ.matrix)
#
# AGE_BKT
AGEBKT.matrix <- model.matrix(~ AGE_BKT - 1, data = NNInput)
```

```
NNInput <- data.frame(NNInput, AGEBKT.matrix)
#
# ACC_TYPE
ACCTYP.matrix <- model.matrix(~ ACC_TYPE - 1, data = NNInput)
NNInput <- data.frame(NNInput, ACCTYP.matrix)

names(NNInput)
#
dim(NNInput)

#Creating Training and Testing Datasets
set.seed(111)
trainIndex <- createDataPartition(TARGET,
                                  p = .7,
                                  list = FALSE,
                                  times = 1)

NN.train.data <- NNInput[trainIndex,]
NN.test.data  <- NNInput[-trainIndex,]

dim(NN.train.data)
dim(NN.test.data)
#
# Scaling the Dataset and Variables
#
#names(NN.train.data)

x <- subset(NN.train.data,
            select = c("AGE",
              "BALANCE",
              "SCR",
              "HOLDING_PERIOD",
              "LEN_OF_RLTN_IN_MNTH",
              "NO_OF_L_CR_TXNS",
              "NO_OF_L_DR_TXNS",
              "TOT_NO_OF_L_TXNS",
              "NO_OF_BR_CSH_WDL_DR_TXNS",
              "NO_OF_ATM_DR_TXNS",
              "NO_OF_NET_DR_TXNS",
              "NO_OF_MOB_DR_TXNS",
              "NO_OF_CHQ_DR_TXNS",
              "FLG_HAS_CC",
              "AMT_ATM_DR",
              "AMT_BR_CSH_WDL_DR",
              "AMT_CHQ_DR",
              "AMT_NET_DR",
              "AMT_MOB_DR",
              "AMT_L_DR",
              "FLG_HAS_ANY_CHGS",
              "AMT_OTH_BK_ATM_USG_CHGS",
              "AMT_MIN_BAL_NMC_CHGS",
              "NO_OF_IW_CHQ_BNC_TXNS",
              "NO_OF_OW_CHQ_BNC_TXNS",
              "AVG_AMT_PER_ATM_TXN",
```

```
                "AVG_AMT_PER_CSH_WDL_TXN",
                "AVG_AMT_PER_CHQ_TXN",
                "AVG_AMT_PER_NET_TXN",
                "AVG_AMT_PER_MOB_TXN",
                "FLG_HAS_NOMINEE",
                "FLG_HAS_OLD_LOAN",
                "random",
                "GENDERF",
                "GENDERM",
                "GENDERO",
                "OCCUPATIONPROF",
                "OCCUPATIONSAL",
                "OCCUPATIONSELF.EMP",
                "OCCUPATIONSENP",
                "AGE_BKT.25",
                "AGE_BKT.50",
                "AGE_BKT26.30",
                "AGE_BKT31.35",
                "AGE_BKT36.40",
                "AGE_BKT41.45",
                "AGE_BKT46.50",
                "ACC_TYPECA",
                "ACC_TYPESA"
                )
)
#
nn.devscaled <- scale(x)
nn.devscaled <- cbind(NN.train.data[2], nn.devscaled)
# names(nn.devscaled)

#
nn2 <- neuralnet(formula = TARGET ~
                    AGE +
                    BALANCE +
                    SCR +
                    HOLDING_PERIOD +
                    LEN_OF_RLTN_IN_MNTH +
                    NO_OF_L_CR_TXNS +
                    NO_OF_L_DR_TXNS +
                    TOT_NO_OF_L_TXNS +
                    NO_OF_BR_CSH_WDL_DR_TXNS +
                    NO_OF_ATM_DR_TXNS +
                    NO_OF_NET_DR_TXNS +
                    NO_OF_MOB_DR_TXNS +
                    NO_OF_CHQ_DR_TXNS +
                    FLG_HAS_CC +
                    AMT_ATM_DR +
                    AMT_BR_CSH_WDL_DR +
                    AMT_CHQ_DR +
                    AMT_NET_DR +
                    AMT_MOB_DR +
                    AMT_L_DR +
                    FLG_HAS_ANY_CHGS +
                    AMT_OTH_BK_ATM_USG_CHGS +
```

```
                    AMT_MIN_BAL_NMC_CHGS +
                    NO_OF_IW_CHQ_BNC_TXNS +
                    NO_OF_OW_CHQ_BNC_TXNS +
                    AVG_AMT_PER_ATM_TXN +
                    AVG_AMT_PER_CSH_WDL_TXN +
                    AVG_AMT_PER_CHQ_TXN +
                    AVG_AMT_PER_NET_TXN +
                    AVG_AMT_PER_MOB_TXN +
                    FLG_HAS_NOMINEE +
                    FLG_HAS_OLD_LOAN +
                    random +
                    GENDERF +
                    GENDERM +
                    GENDERO +
                    OCCUPATIONPROF +
                    OCCUPATIONSAL +
                    OCCUPATIONSELF.EMP +
                    OCCUPATIONSENP +
                    AGE_BKT.25 +
                    AGE_BKT.50 +
                    AGE_BKT26.30 +
                    AGE_BKT31.35 +
                    AGE_BKT36.40 +
                    AGE_BKT41.45 +
                    AGE_BKT46.50 +
                    ACC_TYPECA +
                    ACC_TYPESA ,
                  data = nn.devscaled,
                  hidden = 3,
                  err.fct = "sse",
                  linear.output = FALSE,
                  lifesign = "full",
                  lifesign.step = 10,
                  threshold = 0.1,
                  stepmax = 2000
)
plot (nn2)
# Assigning the Probabilities to Dev Sample
NN.train.data$Prob = nn2$net.result[[1]]

# The distribution of the estimated probabilities
quantile(NN.train.data$Prob, c(0,1,5,10,25,50,75,90,95,98,99,100)/100)
hist(NN.train.data$Prob)
#

## deciling
NN.train.data$deciles <- decile(NN.train.data$Prob)
#
# Ranking code
##install.packages("data.table")
library(data.table)
library(scales)

tmp_DT = data.table(NN.train.data)
```

```
rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round (rank$cnt_resp / rank$cnt,2);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),2);
rank$cum_rel_non_resp <- round(rank$cum_non_resp /
sum(rank$cnt_non_resp),2);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp)

library(scales)
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

# Assgining 0 / 1 class based on certain threshold
NN.train.data$Class = ifelse(NN.train.data$Prob>0.5,1,0)
with( NN.train.data, table(TARGET, as.factor(Class)  ))

# We can use the confusionMatrix function of the caret package
#install.packages("caret")
library(caret)
confusionMatrix(NN.train.data$TARGET, NN.train.data$Class)


## Error Computation
sum((NN.train.data$TARGET - NN.train.data$Prob)^2)/2

## Other Model Performance Measures

library(ROCR)

# str(NN.train.data)
pred3 <- prediction(NN.train.data$Prob, NN.train.data$TARGET)
perf3 <- performance(pred3, "tpr", "fpr")
plot(perf3)
KS3 <- max(attr(perf3, 'y.values')[[1]]-attr(perf3, 'x.values')[[1]])
KS3
auc3 <- performance(pred3,"auc");
auc3 <- as.numeric(auc3@y.values)

auc3

library(ineq)
gini3 = ineq(NN.train.data$Prob, type="Gini")

auc3
KS3
gini3
```

```
# Scoring another dataset using the Neural Net Model Object
# To score we will use the compute function
#
x <- subset(NN.test.data,
            select = c("AGE",
                       "BALANCE",
                       "SCR",
                       "HOLDING_PERIOD",
                       "LEN_OF_RLTN_IN_MNTH",
                       "NO_OF_L_CR_TXNS",
                       "NO_OF_L_DR_TXNS",
                       "TOT_NO_OF_L_TXNS",
                       "NO_OF_BR_CSH_WDL_DR_TXNS",
                       "NO_OF_ATM_DR_TXNS",
                       "NO_OF_NET_DR_TXNS",
                       "NO_OF_MOB_DR_TXNS",
                       "NO_OF_CHQ_DR_TXNS",
                       "FLG_HAS_CC",
                       "AMT_ATM_DR",
                       "AMT_BR_CSH_WDL_DR",
                       "AMT_CHQ_DR",
                       "AMT_NET_DR",
                       "AMT_MOB_DR",
                       "AMT_L_DR",
                       "FLG_HAS_ANY_CHGS",
                       "AMT_OTH_BK_ATM_USG_CHGS",
                       "AMT_MIN_BAL_NMC_CHGS",
                       "NO_OF_IW_CHQ_BNC_TXNS",
                       "NO_OF_OW_CHQ_BNC_TXNS",
                       "AVG_AMT_PER_ATM_TXN",
                       "AVG_AMT_PER_CSH_WDL_TXN",
                       "AVG_AMT_PER_CHQ_TXN",
                       "AVG_AMT_PER_NET_TXN",
                       "AVG_AMT_PER_MOB_TXN",
                       "FLG_HAS_NOMINEE",
                       "FLG_HAS_OLD_LOAN",
                       "random",
                       "GENDERF",
                       "GENDERM",
                       "GENDERO",
                       "OCCUPATIONPROF",
                       "OCCUPATIONSAL",
                       "OCCUPATIONSELF.EMP",
                       "OCCUPATIONSENP",
                       "AGE_BKT.25",
                       "AGE_BKT.50",
                       "AGE_BKT26.30",
                       "AGE_BKT31.35",
                       "AGE_BKT36.40",
                       "AGE_BKT41.45",
                       "AGE_BKT46.50",
                       "ACC_TYPECA",
                       "ACC_TYPESA"
```

```
            )
)
x.scaled <- scale(x)

compute.output = compute(nn2, x.scaled)

# compute.output
NN.test.data$Predict.score = compute.output$net.result
# View(NN.test.data)

names(NN.train.data)

quantile(NN.test.data$Predict.score,
c(0,1,5,10,25,50,75,90,95,99,100)/100)
NN.test.data$deciles <- decile(NN.test.data$Predict.score)

library(data.table)
tmp_DT = data.table(NN.test.data)
h_rank <- tmp_DT[, list(
  cnt = length(TARGET),
  cnt_resp = sum(TARGET),
  cnt_non_resp = sum(TARGET == 0)) ,
  by=deciles][order(-deciles)]
h_rank$rrate <- round (h_rank$cnt_resp / h_rank$cnt,2);
h_rank$cum_resp <- cumsum(h_rank$cnt_resp)
h_rank$cum_non_resp <- cumsum(h_rank$cnt_non_resp)
h_rank$cum_rel_resp <- round(h_rank$cum_resp / sum(h_rank$cnt_resp),2);
h_rank$cum_rel_non_resp <- round(h_rank$cum_non_resp /
sum(h_rank$cnt_non_resp),2);
h_rank$ks <- abs(h_rank$cum_rel_resp - h_rank$cum_rel_non_resp);
#
library(scales)
h_rank$rrate <- percent(h_rank$rrate)
h_rank$cum_rel_resp <- percent(h_rank$cum_rel_resp)
h_rank$cum_rel_non_resp <- percent(h_rank$cum_rel_non_resp)

View(h_rank)
#
#============================================================
# Working on Holdout Sample
#============================================================

# Assigning the Probabilities to Holdout Sample
NN.test.data$Prob = compute.output$net.result

# Assgining 0 / 1 class based on certain threshold
NN.test.data$Class = ifelse(NN.test.data$Prob>0.5,1,0)
with( NN.test.data, table(TARGET, as.factor(Class)  ))

# We can use the confusionMatrix function of the caret package
#install.packages("caret")
library(caret)
confusionMatrix(NN.test.data$TARGET, NN.test.data$Class)
```

```
## Error Computation
sum((NN.test.data$TARGET - NN.test.data$Prob)^2)/2

## Other Model Performance Measures

library(ROCR)

# str(NN.test.data)
pred4 <- prediction(NN.test.data$Prob, NN.test.data$TARGET)
perf4 <- performance(pred4, "tpr", "fpr")
plot(perf4)
KS4 <- max(attr(perf4, 'y.values')[[1]]-attr(perf4, 'x.values')[[1]])
KS4
auc4 <- performance(pred4,"auc");
auc4 <- as.numeric(auc4@y.values)

auc4

library(ineq)
gini4 = ineq(NN.test.data$Prob, type="Gini")

auc4
KS4
gini4
```