

Mini Project 4

Personal loan Campaign



March 24, 2019

Saurabh MUDGal

**Part 1 - Classification Tree ‘**

Reading data file

> setwd("C:/BACP/Module 4 - Machine learning/Project/")

> loan\_cart=read.table("PL\_XSELL.csv",sep = ",",header = T)

> View(loan\_cart)

> table(loan\_cart$TARGET)

0 1

17488 2512

>

> names(loan\_cart)

[1] "CUST\_ID" "TARGET"

[3] "AGE" "GENDER"

[5] "BALANCE" "OCCUPATION"

[7] "AGE\_BKT" "SCR"

[9] "HOLDING\_PERIOD" "ACC\_TYPE"

[11] "ACC\_OP\_DATE" "LEN\_OF\_RLTN\_IN\_MNTH"

[13] "NO\_OF\_L\_CR\_TXNS" "NO\_OF\_L\_DR\_TXNS"

[15] "TOT\_NO\_OF\_L\_TXNS" "NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS"

[17] "NO\_OF\_ATM\_DR\_TXNS" "NO\_OF\_NET\_DR\_TXNS"

[19] "NO\_OF\_MOB\_DR\_TXNS" "NO\_OF\_CHQ\_DR\_TXNS"

[21] "FLG\_HAS\_CC" "AMT\_ATM\_DR"

[23] "AMT\_BR\_CSH\_WDL\_DR" "AMT\_CHQ\_DR"

[25] "AMT\_NET\_DR" "AMT\_MOB\_DR"

[27] "AMT\_L\_DR" "FLG\_HAS\_ANY\_CHGS"

[29] "AMT\_OTH\_BK\_ATM\_USG\_CHGS" "AMT\_MIN\_BAL\_NMC\_CHGS"

[31] "NO\_OF\_IW\_CHQ\_BNC\_TXNS" "NO\_OF\_OW\_CHQ\_BNC\_TXNS"

[33] "AVG\_AMT\_PER\_ATM\_TXN" "AVG\_AMT\_PER\_CSH\_WDL\_TXN"

[35] "AVG\_AMT\_PER\_CHQ\_TXN" "AVG\_AMT\_PER\_NET\_TXN"

[37] "AVG\_AMT\_PER\_MOB\_TXN" "FLG\_HAS\_NOMINEE"

[39] "FLG\_HAS\_OLD\_LOAN" "random"

> str(loan\_cart)

'data.frame': 20000 obs. of 40 variables:

$ CUST\_ID : Factor w/ 20000 levels "C1","C10","C100",..: 17699 16532 11027 17984 2363 11747 18115 15556 15216 12494 ...

$ TARGET : int 0 0 0 0 0 0 0 0 0 0 ...

$ AGE : int 27 47 40 53 36 42 30 53 42 30 ...

$ GENDER : Factor w/ 3 levels "F","M","O": 2 2 2 2 2 1 2 1 1 2 ...

$ BALANCE : num 3384 287489 18217 71720 1671623 ...

$ OCCUPATION : Factor w/ 4 levels "PROF","SAL","SELF-EMP",..: 3 2 3 2 1 1 1 2 3 1 ...

$ AGE\_BKT : Factor w/ 7 levels "<25",">50","26-30",..: 3 7 5 2 5 6 3 2 6 3 ...

$ SCR : int 776 324 603 196 167 493 479 562 105 170 ...

$ HOLDING\_PERIOD : int 30 28 2 13 24 26 14 25 15 13 ...

$ ACC\_TYPE : Factor w/ 2 levels "CA","SA": 2 2 2 1 2 2 2 1 2 2 ...

$ ACC\_OP\_DATE : Factor w/ 4869 levels "01-01-00","01-01-01",..: 3270 1806 3575 993 2861 862 4533 3160 257 334 ...

$ LEN\_OF\_RLTN\_IN\_MNTH : int 146 104 61 107 185 192 177 99 88 111 ...

$ NO\_OF\_L\_CR\_TXNS : int 7 8 10 36 20 5 6 14 18 14 ...

$ NO\_OF\_L\_DR\_TXNS : int 3 2 5 14 1 2 6 3 14 8 ...

$ TOT\_NO\_OF\_L\_TXNS : int 10 10 15 50 21 7 12 17 32 22 ...

$ NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS: int 0 0 1 4 1 1 0 3 6 3 ...

$ NO\_OF\_ATM\_DR\_TXNS : int 1 1 1 2 0 1 1 0 2 1 ...

$ NO\_OF\_NET\_DR\_TXNS : int 2 1 1 3 0 0 1 0 4 0 ...

$ NO\_OF\_MOB\_DR\_TXNS : int 0 0 0 1 0 0 0 0 1 0 ...

$ NO\_OF\_CHQ\_DR\_TXNS : int 0 0 2 4 0 0 4 0 1 4 ...

$ FLG\_HAS\_CC : int 0 0 0 0 0 1 0 0 1 0 ...

$ AMT\_ATM\_DR : int 13100 6600 11200 26100 0 18500 6200 0 35400 18000 ...

$ AMT\_BR\_CSH\_WDL\_DR : int 0 0 561120 673590 808480 379310 0 945160 198430 869880 ...

$ AMT\_CHQ\_DR : int 0 0 49320 60780 0 0 10580 0 51490 32610 ...

$ AMT\_NET\_DR : num 973557 799813 997570 741506 0 ...

$ AMT\_MOB\_DR : int 0 0 0 71388 0 0 0 0 170332 0 ...

$ AMT\_L\_DR : num 986657 806413 1619210 1573364 808480 ...

$ FLG\_HAS\_ANY\_CHGS : int 0 1 1 0 0 0 1 0 0 0 ...

$ AMT\_OTH\_BK\_ATM\_USG\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ AMT\_MIN\_BAL\_NMC\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_IW\_CHQ\_BNC\_TXNS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_OW\_CHQ\_BNC\_TXNS : int 0 0 1 0 0 0 0 0 0 0 ...

$ AVG\_AMT\_PER\_ATM\_TXN : num 13100 6600 11200 13050 0 ...

$ AVG\_AMT\_PER\_CSH\_WDL\_TXN : num 0 0 561120 168398 808480 ...

$ AVG\_AMT\_PER\_CHQ\_TXN : num 0 0 24660 15195 0 ...

$ AVG\_AMT\_PER\_NET\_TXN : num 486779 799813 997570 247169 0 ...

$ AVG\_AMT\_PER\_MOB\_TXN : num 0 0 0 71388 0 ...

$ FLG\_HAS\_NOMINEE : int 1 1 1 1 1 1 0 1 1 0 ...

$ FLG\_HAS\_OLD\_LOAN : int 1 0 1 0 0 1 1 1 1 0 ...

$ random : num 1.14e-05 1.11e-04 1.20e-04 1.37e-04 1.74e-04 ...

> #converting acc\_op\_date to date format

> loan\_cart$ACC\_OP\_DATE = as.Date(loan\_cart$ACC\_OP\_DATE, format="%m/%d/%Y")

> str(loan\_cart)

'data.frame': 20000 obs. of 40 variables:

$ CUST\_ID : Factor w/ 20000 levels "C1","C10","C100",..: 17699 16532 11027 17984 2363 11747 18115 15556 15216 12494 ...

$ TARGET : int 0 0 0 0 0 0 0 0 0 0 ...

$ AGE : int 27 47 40 53 36 42 30 53 42 30 ...

$ GENDER : Factor w/ 3 levels "F","M","O": 2 2 2 2 2 1 2 1 1 2 ...

$ BALANCE : num 3384 287489 18217 71720 1671623 ...

$ OCCUPATION : Factor w/ 4 levels "PROF","SAL","SELF-EMP",..: 3 2 3 2 1 1 1 2 3 1 ...

$ AGE\_BKT : Factor w/ 7 levels "<25",">50","26-30",..: 3 7 5 2 5 6 3 2 6 3 ...

$ SCR : int 776 324 603 196 167 493 479 562 105 170 ...

$ HOLDING\_PERIOD : int 30 28 2 13 24 26 14 25 15 13 ...

$ ACC\_TYPE : Factor w/ 2 levels "CA","SA": 2 2 2 1 2 2 2 1 2 2 ...

$ ACC\_OP\_DATE : Date, format: "2005-03-23" ...

$ LEN\_OF\_RLTN\_IN\_MNTH : int 146 104 61 107 185 192 177 99 88 111 ...

$ NO\_OF\_L\_CR\_TXNS : int 7 8 10 36 20 5 6 14 18 14 ...

$ NO\_OF\_L\_DR\_TXNS : int 3 2 5 14 1 2 6 3 14 8 ...

$ TOT\_NO\_OF\_L\_TXNS : int 10 10 15 50 21 7 12 17 32 22 ...

$ NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS: int 0 0 1 4 1 1 0 3 6 3 ...

$ NO\_OF\_ATM\_DR\_TXNS : int 1 1 1 2 0 1 1 0 2 1 ...

$ NO\_OF\_NET\_DR\_TXNS : int 2 1 1 3 0 0 1 0 4 0 ...

$ NO\_OF\_MOB\_DR\_TXNS : int 0 0 0 1 0 0 0 0 1 0 ...

$ NO\_OF\_CHQ\_DR\_TXNS : int 0 0 2 4 0 0 4 0 1 4 ...

$ FLG\_HAS\_CC : int 0 0 0 0 0 1 0 0 1 0 ...

$ AMT\_ATM\_DR : int 13100 6600 11200 26100 0 18500 6200 0 35400 18000 ...

$ AMT\_BR\_CSH\_WDL\_DR : int 0 0 561120 673590 808480 379310 0 945160 198430 869880 ...

$ AMT\_CHQ\_DR : int 0 0 49320 60780 0 0 10580 0 51490 32610 ...

$ AMT\_NET\_DR : num 973557 799813 997570 741506 0 ...

$ AMT\_MOB\_DR : int 0 0 0 71388 0 0 0 0 170332 0 ...

$ AMT\_L\_DR : num 986657 806413 1619210 1573364 808480 ...

$ FLG\_HAS\_ANY\_CHGS : int 0 1 1 0 0 0 1 0 0 0 ...

$ AMT\_OTH\_BK\_ATM\_USG\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ AMT\_MIN\_BAL\_NMC\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_IW\_CHQ\_BNC\_TXNS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_OW\_CHQ\_BNC\_TXNS : int 0 0 1 0 0 0 0 0 0 0 ...

$ AVG\_AMT\_PER\_ATM\_TXN : num 13100 6600 11200 13050 0 ...

$ AVG\_AMT\_PER\_CSH\_WDL\_TXN : num 0 0 561120 168398 808480 ...

$ AVG\_AMT\_PER\_CHQ\_TXN : num 0 0 24660 15195 0 ...

$ AVG\_AMT\_PER\_NET\_TXN : num 486779 799813 997570 247169 0 ...

$ AVG\_AMT\_PER\_MOB\_TXN : num 0 0 0 71388 0 ...

$ FLG\_HAS\_NOMINEE : int 1 1 1 1 1 1 0 1 1 0 ...

$ FLG\_HAS\_OLD\_LOAN : int 1 0 1 0 0 1 1 1 1 0 ...

$ random : num 1.14e-05 1.11e-04 1.20e-04 1.37e-04 1.74e-04 ...

**Split data into Development (70%) and Hold-out (30%) Sample**

> #split data in test and train data set

> set.seed(123) ## to get reapeatable data

>

> ind = sample(2,nrow(loan\_cart),replace = TRUE ,prob = c(.7,.3))

> dev\_cart = loan\_cart[ind==1,]

> hold\_out\_cart = loan\_cart[ind==2,]

> table(dev\_cart$TARGET)

0 1

12319 1761

> table(hold\_out\_cart$TARGET)

0 1

5169 751

**•Build Classification Tree using CART technique**

#lib for CART

> library(rpart)

> #lib for visual flow chart of tree

> library(rpart.plot)

>

> #setting up control parameter

> r.ctrl = rpart.control(minsplit = 100,minbucket = 10,cp=0,xval=10)

>

> #build tree

> m\_cart=rpart(formula=TARGET ~.,data = dev\_cart[,-1], method = "class", control = r.ctrl)

> m\_cart

n= 14080

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 14080 1761 0 (0.87492898 0.12507102)

2) NO\_OF\_L\_CR\_TXNS< 48.5 13599 1597 0 (0.88256489 0.11743511)

4) HOLDING\_PERIOD>=15.5 6554 527 0 (0.91959109 0.08040891)

8) FLG\_HAS\_CC< 0.5 4562 270 0 (0.94081543 0.05918457)

16) NO\_OF\_L\_CR\_TXNS< 16.5 3774 165 0 (0.95627981 0.04372019)

32) SCR< 994.5 3763 159 0 (0.95774648 0.04225352)

64) OCCUPATION=PROF,SAL,SENP 3028 92 0 (0.96961691 0.03038309) \*

65) OCCUPATION=SELF-EMP 735 67 0 (0.90884354 0.09115646)

130) ACC\_OP\_DATE< 15991 691 50 0 (0.92764110 0.07235890)

260) NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS< 5.5 681 43 0 (0.93685756 0.06314244)

520) BALANCE>=42785.68 541 19 0 (0.96487985 0.03512015) \*

521) BALANCE< 42785.68 140 24 0 (0.82857143 0.17142857)

1042) AGE< 53.5 130 14 0 (0.89230769 0.10769231)

2084) AMT\_L\_DR< 1264747 116 6 0 (0.94827586 0.05172414) \*

2085) AMT\_L\_DR>=1264747 14 6 1 (0.42857143 0.57142857) \*

1043) AGE>=53.5 10 0 1 (0.00000000 1.00000000) \*

261) NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS>=5.5 10 3 1 (0.30000000 0.70000000) \*

131) ACC\_OP\_DATE>=15991 44 17 0 (0.61363636 0.38636364) \*

33) SCR>=994.5 11 5 1 (0.45454545 0.54545455) \*

17) NO\_OF\_L\_CR\_TXNS>=16.5 788 105 0 (0.86675127 0.13324873)

34) BALANCE>=63162.51 587 57 0 (0.90289608 0.09710392)

68) NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS< 9.5 574 50 0 (0.91289199 0.08710801) \*

69) NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS>=9.5 13 6 1 (0.46153846 0.53846154) \*

35) BALANCE< 63162.51 201 48 0 (0.76119403 0.23880597) \*

9) FLG\_HAS\_CC>=0.5 1992 257 0 (0.87098394 0.12901606)

18) NO\_OF\_ATM\_DR\_TXNS< 0.5 802 58 0 (0.92768080 0.07231920)

36) FLG\_HAS\_ANY\_CHGS< 0.5 733 43 0 (0.94133697 0.05866303)

72) NO\_OF\_CHQ\_DR\_TXNS< 2.5 712 37 0 (0.94803371 0.05196629)

144) AGE\_BKT=26-30,31-35 256 3 0 (0.98828125 0.01171875) \*

145) AGE\_BKT=<25,>50,36-40,41-45,46-50 456 34 0 (0.92543860 0.07456140)

290) BALANCE>=328616.4 196 4 0 (0.97959184 0.02040816) \*

291) BALANCE< 328616.4 260 30 0 (0.88461538 0.11538462)

582) ACC\_OP\_DATE>=14788.5 78 2 0 (0.97435897 0.02564103) \*

583) ACC\_OP\_DATE< 14788.5 182 28 0 (0.84615385 0.15384615)

1166) ACC\_OP\_DATE< 14358 172 21 0 (0.87790698 0.12209302) \*

1167) ACC\_OP\_DATE>=14358 10 3 1 (0.30000000 0.70000000) \*

73) NO\_OF\_CHQ\_DR\_TXNS>=2.5 21 6 0 (0.71428571 0.28571429) \*

37) FLG\_HAS\_ANY\_CHGS>=0.5 69 15 0 (0.78260870 0.21739130) \*

19) NO\_OF\_ATM\_DR\_TXNS>=0.5 1190 199 0 (0.83277311 0.16722689)

38) AMT\_ATM\_DR>=50 1166 175 0 (0.84991424 0.15008576)

76) AVG\_AMT\_PER\_NET\_TXN< 922368.5 1135 158 0 (0.86079295 0.13920705)

152) AVG\_AMT\_PER\_MOB\_TXN< 151369.5 1094 139 0 (0.87294333 0.12705667)

304) AVG\_AMT\_PER\_NET\_TXN< 258902.5 812 77 0 (0.90517241 0.09482759)

608) AGE\_BKT=<25,>50,31-35,46-50 431 18 0 (0.95823666 0.04176334) \*

609) AGE\_BKT=26-30,36-40,41-45 381 59 0 (0.84514436 0.15485564)

1218) NO\_OF\_CHQ\_DR\_TXNS< 4.5 358 47 0 (0.86871508 0.13128492) \*

1219) NO\_OF\_CHQ\_DR\_TXNS>=4.5 23 11 1 (0.47826087 0.52173913) \*

305) AVG\_AMT\_PER\_NET\_TXN>=258902.5 282 62 0 (0.78014184 0.21985816)

610) AMT\_NET\_DR>=679381.5 105 4 0 (0.96190476 0.03809524) \*

611) AMT\_NET\_DR< 679381.5 177 58 0 (0.67231638 0.32768362)

1222) AVG\_AMT\_PER\_CHQ\_TXN>=22730 46 2 0 (0.95652174 0.04347826) \*

1223) AVG\_AMT\_PER\_CHQ\_TXN< 22730 131 56 0 (0.57251908 0.42748092)

2446) BALANCE>=539045.2 39 7 0 (0.82051282 0.17948718) \*

2447) BALANCE< 539045.2 92 43 1 (0.46739130 0.53260870) \*

153) AVG\_AMT\_PER\_MOB\_TXN>=151369.5 41 19 0 (0.53658537 0.46341463) \*

77) AVG\_AMT\_PER\_NET\_TXN>=922368.5 31 14 1 (0.45161290 0.54838710) \*

39) AMT\_ATM\_DR< 50 24 0 1 (0.00000000 1.00000000) \*

5) HOLDING\_PERIOD< 15.5 7045 1070 0 (0.84811923 0.15188077)

10) OCCUPATION=PROF,SAL,SENP 5816 772 0 (0.86726272 0.13273728)

20) FLG\_HAS\_CC< 0.5 3967 433 0 (0.89084951 0.10915049)

40) AVG\_AMT\_PER\_ATM\_TXN< 11416.67 2550 220 0 (0.91372549 0.08627451)

80) AVG\_AMT\_PER\_CHQ\_TXN< 203842 2514 208 0 (0.91726333 0.08273667)

160) AGE\_BKT=<25,>50,26-30,36-40 1414 85 0 (0.93988685 0.06011315) \*

161) AGE\_BKT=31-35,41-45,46-50 1100 123 0 (0.88818182 0.11181818)

322) AVG\_AMT\_PER\_NET\_TXN< 499582.5 917 86 0 (0.90621592 0.09378408) \*

323) AVG\_AMT\_PER\_NET\_TXN>=499582.5 183 37 0 (0.79781421 0.20218579)

646) TOT\_NO\_OF\_L\_TXNS>=10.5 169 29 0 (0.82840237 0.17159763) \*

647) TOT\_NO\_OF\_L\_TXNS< 10.5 14 6 1 (0.42857143 0.57142857) \*

81) AVG\_AMT\_PER\_CHQ\_TXN>=203842 36 12 0 (0.66666667 0.33333333) \*

41) AVG\_AMT\_PER\_ATM\_TXN>=11416.67 1417 213 0 (0.84968243 0.15031757)

82) AMT\_BR\_CSH\_WDL\_DR< 981400 1398 200 0 (0.85693848 0.14306152)

164) ACC\_OP\_DATE< 12525 467 36 0 (0.92291221 0.07708779) \*

165) ACC\_OP\_DATE>=12525 931 164 0 (0.82384533 0.17615467)

330) SCR< 535.5 603 79 0 (0.86898839 0.13101161) \*

331) SCR>=535.5 328 85 0 (0.74085366 0.25914634)

662) NO\_OF\_CHQ\_DR\_TXNS< 2.5 139 16 0 (0.88489209 0.11510791) \*

663) NO\_OF\_CHQ\_DR\_TXNS>=2.5 189 69 0 (0.63492063 0.36507937)

1326) AMT\_BR\_CSH\_WDL\_DR< 322070 79 14 0 (0.82278481 0.17721519) \*

1327) AMT\_BR\_CSH\_WDL\_DR>=322070 110 55 0 (0.50000000 0.50000000)

2654) HOLDING\_PERIOD>=12.5 19 1 0 (0.94736842 0.05263158) \*

2655) HOLDING\_PERIOD< 12.5 91 37 1 (0.40659341 0.59340659) \*

83) AMT\_BR\_CSH\_WDL\_DR>=981400 19 6 1 (0.31578947 0.68421053) \*

21) FLG\_HAS\_CC>=0.5 1849 339 0 (0.81665765 0.18334235)

42) AVG\_AMT\_PER\_MOB\_TXN< 143313.5 1735 295 0 (0.82997118 0.17002882)

84) AVG\_AMT\_PER\_CSH\_WDL\_TXN< 545650 1459 222 0 (0.84784099 0.15215901)

168) HOLDING\_PERIOD>=4.5 1054 135 0 (0.87191651 0.12808349)

336) AVG\_AMT\_PER\_ATM\_TXN< 23600 1040 128 0 (0.87692308 0.12307692)

672) SCR< 849 936 102 0 (0.89102564 0.10897436)

1344) AMT\_CHQ\_DR< 981390 916 94 0 (0.89737991 0.10262009)

2688) AMT\_BR\_CSH\_WDL\_DR>=695095 165 2 0 (0.98787879 0.01212121) \*

2689) AMT\_BR\_CSH\_WDL\_DR< 695095 751 92 0 (0.87749667 0.12250333)

5378) AMT\_BR\_CSH\_WDL\_DR< 680740 741 86 0 (0.88394062 0.11605938)

10756) AGE\_BKT=<25,>50,31-35,41-45,46-50 474 41 0 (0.91350211 0.08649789)

21512) AVG\_AMT\_PER\_CSH\_WDL\_TXN>=96048 208 7 0 (0.96634615 0.03365385) \*

21513) AVG\_AMT\_PER\_CSH\_WDL\_TXN< 96048 266 34 0 (0.87218045 0.12781955)

43026) NO\_OF\_L\_DR\_TXNS< 2.5 74 1 0 (0.98648649 0.01351351) \*

43027) NO\_OF\_L\_DR\_TXNS>=2.5 192 33 0 (0.82812500 0.17187500)

86054) NO\_OF\_L\_CR\_TXNS>=4.5 171 20 0 (0.88304094 0.11695906) \*

86055) NO\_OF\_L\_CR\_TXNS< 4.5 21 8 1 (0.38095238 0.61904762) \*

10757) AGE\_BKT=26-30,36-40 267 45 0 (0.83146067 0.16853933) \*

5379) AMT\_BR\_CSH\_WDL\_DR>=680740 10 4 1 (0.40000000 0.60000000) \*

1345) AMT\_CHQ\_DR>=981390 20 8 0 (0.60000000 0.40000000) \*

673) SCR>=849 104 26 0 (0.75000000 0.25000000)

1346) ACC\_OP\_DATE>=10709 92 18 0 (0.80434783 0.19565217) \*

1347) ACC\_OP\_DATE< 10709 12 4 1 (0.33333333 0.66666667) \*

337) AVG\_AMT\_PER\_ATM\_TXN>=23600 14 7 0 (0.50000000 0.50000000) \*

169) HOLDING\_PERIOD< 4.5 405 87 0 (0.78518519 0.21481481)

338) AMT\_L\_DR< 534404.5 156 15 0 (0.90384615 0.09615385) \*

339) AMT\_L\_DR>=534404.5 249 72 0 (0.71084337 0.28915663)

678) random>=0.1741624 206 49 0 (0.76213592 0.23786408)

1356) AMT\_BR\_CSH\_WDL\_DR< 769800 163 31 0 (0.80981595 0.19018405)

2712) AVG\_AMT\_PER\_CSH\_WDL\_TXN>=188897.5 60 4 0 (0.93333333 0.06666667) \*

2713) AVG\_AMT\_PER\_CSH\_WDL\_TXN< 188897.5 103 27 0 (0.73786408 0.26213592)

5426) AMT\_BR\_CSH\_WDL\_DR< 543710 87 17 0 (0.80459770 0.19540230) \*

5427) AMT\_BR\_CSH\_WDL\_DR>=543710 16 6 1 (0.37500000 0.62500000) \*

1357) AMT\_BR\_CSH\_WDL\_DR>=769800 43 18 0 (0.58139535 0.41860465) \*

679) random< 0.1741624 43 20 1 (0.46511628 0.53488372) \*

85) AVG\_AMT\_PER\_CSH\_WDL\_TXN>=545650 276 73 0 (0.73550725 0.26449275)

170) AGE\_BKT=<25,31-35,36-40 119 16 0 (0.86554622 0.13445378)

340) HOLDING\_PERIOD>=1.5 109 10 0 (0.90825688 0.09174312) \*

341) HOLDING\_PERIOD< 1.5 10 4 1 (0.40000000 0.60000000) \*

171) AGE\_BKT=>50,26-30,41-45,46-50 157 57 0 (0.63694268 0.36305732)

342) AMT\_L\_DR>=892195 101 24 0 (0.76237624 0.23762376) \*

343) AMT\_L\_DR< 892195 56 23 1 (0.41071429 0.58928571) \*

43) AVG\_AMT\_PER\_MOB\_TXN>=143313.5 114 44 0 (0.61403509 0.38596491)

86) AMT\_MOB\_DR>=148926.5 99 29 0 (0.70707071 0.29292929) \*

87) AMT\_MOB\_DR< 148926.5 15 0 1 (0.00000000 1.00000000) \*

11) OCCUPATION=SELF-EMP 1229 298 0 (0.75752644 0.24247356)

22) BALANCE>=50334.8 919 171 0 (0.81392818 0.18607182)

44) HOLDING\_PERIOD>=1.5 797 129 0 (0.83814304 0.16185696)

88) NO\_OF\_L\_CR\_TXNS< 18.5 710 100 0 (0.85915493 0.14084507)

176) AMT\_BR\_CSH\_WDL\_DR< 855510 615 75 0 (0.87804878 0.12195122)

352) NO\_OF\_L\_CR\_TXNS>=2.5 528 53 0 (0.89962121 0.10037879)

704) AMT\_ATM\_DR< 6650 256 12 0 (0.95312500 0.04687500) \*

705) AMT\_ATM\_DR>=6650 272 41 0 (0.84926471 0.15073529)

1410) AMT\_ATM\_DR>=12550 167 12 0 (0.92814371 0.07185629) \*

1411) AMT\_ATM\_DR< 12550 105 29 0 (0.72380952 0.27619048)

2822) AVG\_AMT\_PER\_ATM\_TXN< 12250 95 20 0 (0.78947368 0.21052632) \*

2823) AVG\_AMT\_PER\_ATM\_TXN>=12250 10 1 1 (0.10000000 0.90000000) \*

353) NO\_OF\_L\_CR\_TXNS< 2.5 87 22 0 (0.74712644 0.25287356) \*

177) AMT\_BR\_CSH\_WDL\_DR>=855510 95 25 0 (0.73684211 0.26315789) \*

89) NO\_OF\_L\_CR\_TXNS>=18.5 87 29 0 (0.66666667 0.33333333) \*

45) HOLDING\_PERIOD< 1.5 122 42 0 (0.65573770 0.34426230) \*

23) BALANCE< 50334.8 310 127 0 (0.59032258 0.40967742)

46) SCR< 350.5 115 30 0 (0.73913043 0.26086957)

92) TOT\_NO\_OF\_L\_TXNS< 17.5 64 4 0 (0.93750000 0.06250000) \*

93) TOT\_NO\_OF\_L\_TXNS>=17.5 51 25 1 (0.49019608 0.50980392) \*

47) SCR>=350.5 195 97 0 (0.50256410 0.49743590)

94) AGE\_BKT=>50,31-35,36-40 102 37 0 (0.63725490 0.36274510)

188) LEN\_OF\_RLTN\_IN\_MNTH< 212.5 90 27 0 (0.70000000 0.30000000) \*

189) LEN\_OF\_RLTN\_IN\_MNTH>=212.5 12 2 1 (0.16666667 0.83333333) \*

95) AGE\_BKT=<25,26-30,41-45,46-50 93 33 1 (0.35483871 0.64516129) \*

3) NO\_OF\_L\_CR\_TXNS>=48.5 481 164 0 (0.65904366 0.34095634)

6) AMT\_BR\_CSH\_WDL\_DR< 227270 171 30 0 (0.82456140 0.17543860)

12) NO\_OF\_L\_CR\_TXNS>=50.5 156 22 0 (0.85897436 0.14102564) \*

13) NO\_OF\_L\_CR\_TXNS< 50.5 15 7 1 (0.46666667 0.53333333) \*

7) AMT\_BR\_CSH\_WDL\_DR>=227270 310 134 0 (0.56774194 0.43225806)

14) HOLDING\_PERIOD>=19.5 87 13 0 (0.85057471 0.14942529) \*

15) HOLDING\_PERIOD< 19.5 223 102 1 (0.45739910 0.54260090)

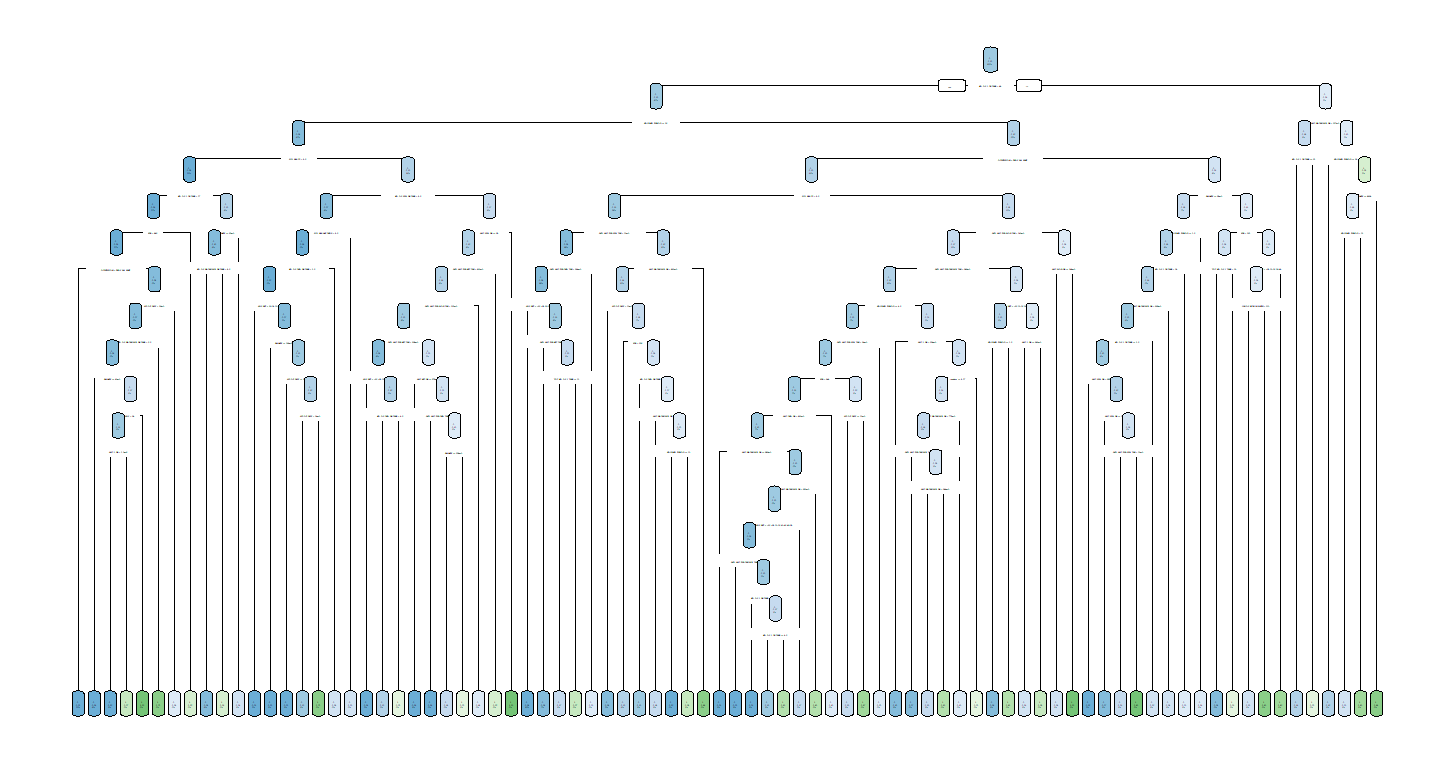
30) BALANCE>=9853.525 187 89 0 (0.52406417 0.47593583)

60) HOLDING\_PERIOD< 11.5 97 32 0 (0.67010309 0.32989691) \*

61) HOLDING\_PERIOD>=11.5 90 33 1 (0.36666667 0.63333333) \*

31) BALANCE< 9853.525 36 4 1 (0.11111111 0.88888889) \*

> rpart.plot(m\_cart)



Model has used 27 variables out of 45.

> #how the tree performs

> printcp(m\_cart)

Classification tree:

rpart(formula = TARGET ~ ., data = dev\_cart[, -1], method = "class",

control = r.ctrl)

Variables actually used in tree construction:

[1] ACC\_OP\_DATE AGE

[3] AGE\_BKT AMT\_ATM\_DR

[5] AMT\_BR\_CSH\_WDL\_DR AMT\_CHQ\_DR

[7] AMT\_L\_DR AMT\_MOB\_DR

[9] AMT\_NET\_DR AVG\_AMT\_PER\_ATM\_TXN

[11] AVG\_AMT\_PER\_CHQ\_TXN AVG\_AMT\_PER\_CSH\_WDL\_TXN

[13] AVG\_AMT\_PER\_MOB\_TXN AVG\_AMT\_PER\_NET\_TXN

[15] BALANCE FLG\_HAS\_ANY\_CHGS

[17] FLG\_HAS\_CC HOLDING\_PERIOD

[19] LEN\_OF\_RLTN\_IN\_MNTH NO\_OF\_ATM\_DR\_TXNS

[21] NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS NO\_OF\_CHQ\_DR\_TXNS

[23] NO\_OF\_L\_CR\_TXNS NO\_OF\_L\_DR\_TXNS

[25] OCCUPATION random

[27] SCR TOT\_NO\_OF\_L\_TXNS

Root node error: 1761/14080 = 0.12507

n= 14080

CP nsplit rel error xerror xstd

1 0.00361365 0 1.00000 1.00000 0.022290

2 0.00283930 14 0.93697 0.98751 0.022170

3 0.00198751 17 0.92845 0.98069 0.022104

4 0.00193072 19 0.92447 0.97501 0.022049

5 0.00189286 24 0.91482 0.97501 0.022049

6 0.00170358 27 0.90914 0.98467 0.022143

7 0.00121684 28 0.90744 0.99432 0.022235

8 0.00113572 35 0.89892 1.00625 0.022349

9 0.00068143 37 0.89665 1.01476 0.022430

10 0.00066250 42 0.89324 1.01476 0.022430

11 0.00064898 51 0.88700 1.01476 0.022430

12 0.00056786 58 0.88245 1.01476 0.022430

13 0.00037857 60 0.88132 1.01590 0.022441

14 0.00028393 73 0.87507 1.01704 0.022452

15 0.00000000 81 0.87280 1.02896 0.022563

Inflection point is at split 28.

**•Do necessary pruning**

> p\_m\_cart = prune(m\_cart, cp=.0012,"cp")

> rpart.plot(p\_m\_cart)

Warning message:

Bad 'data' field in model 'call' (expected a data.frame or a matrix).

To silence this warning:

Call rpart.plot with roundint=FALSE,

or rebuild the rpart model with model=TRUE.

> printcp(p\_m\_cart)

Classification tree:

rpart(formula = TARGET ~ ., data = dev\_cart[, -1], method = "class",

control = r.ctrl)

Variables actually used in tree construction:

[1] ACC\_OP\_DATE AGE

[3] AGE\_BKT AMT\_ATM\_DR

[5] AMT\_BR\_CSH\_WDL\_DR AMT\_L\_DR

[7] AMT\_MOB\_DR AVG\_AMT\_PER\_ATM\_TXN

[9] AVG\_AMT\_PER\_CSH\_WDL\_TXN AVG\_AMT\_PER\_MOB\_TXN

[11] AVG\_AMT\_PER\_NET\_TXN BALANCE

[13] FLG\_HAS\_CC HOLDING\_PERIOD

[15] LEN\_OF\_RLTN\_IN\_MNTH NO\_OF\_ATM\_DR\_TXNS

[17] NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS NO\_OF\_CHQ\_DR\_TXNS

[19] NO\_OF\_L\_CR\_TXNS OCCUPATION

[21] SCR

Root node error: 1761/14080 = 0.12507

n= 14080

CP nsplit rel error xerror xstd

1 0.0036136 0 1.00000 1.00000 0.022290

2 0.0028393 14 0.93697 0.98751 0.022170

3 0.0019875 17 0.92845 0.98069 0.022104

4 0.0019307 19 0.92447 0.97501 0.022049

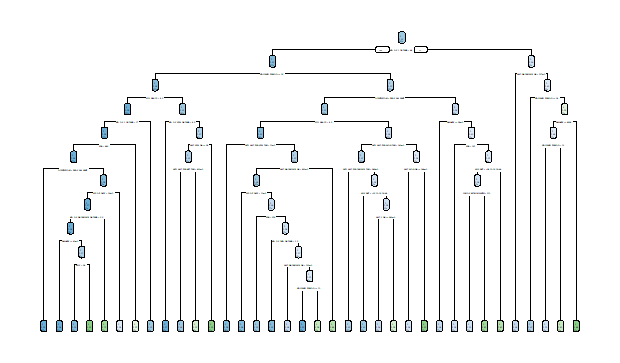
5 0.0018929 24 0.91482 0.97501 0.022049

6 0.0017036 27 0.90914 0.98467 0.022143

7 0.0012168 28 0.90744 0.99432 0.022235

8 0.0012000 35 0.89892 1.00625 0.022349

After pruning tree uses 21 variables out of 40.



**•Measure Model Performance on Development Sample**

> dev\_cart$predict.class = predict(p\_m\_cart,dev\_cart, type="class")

> dev\_cart$predict.score = predict(p\_m\_cart,dev\_cart, type="prob")

> View(dev\_cart)

> head(dev\_cart)

CUST\_ID TARGET AGE GENDER BALANCE OCCUPATION AGE\_BKT SCR

1 C7927 0 27 M 3383.75 SELF-EMP 26-30 776

3 C19922 0 40 M 18216.88 SELF-EMP 36-40 603

6 C257 0 42 F 521685.69 PROF 41-45 493

7 C8300 0 30 M 204458.60 PROF 26-30 479

9 C5692 0 42 F 13158.14 SELF-EMP 41-45 105

10 C3241 0 30 M 831150.18 PROF 26-30 170

HOLDING\_PERIOD ACC\_TYPE ACC\_OP\_DATE LEN\_OF\_RLTN\_IN\_MNTH

1 30 SA 2005-03-23 146

3 2 SA 2012-04-26 61

6 26 SA 2001-06-07 192

7 14 SA 2002-08-25 177

9 15 SA 2010-02-08 88

10 13 SA 2008-03-02 111

NO\_OF\_L\_CR\_TXNS NO\_OF\_L\_DR\_TXNS TOT\_NO\_OF\_L\_TXNS

1 7 3 10

3 10 5 15

6 5 2 7

7 6 6 12

9 18 14 32

10 14 8 22

NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS NO\_OF\_ATM\_DR\_TXNS NO\_OF\_NET\_DR\_TXNS

1 0 1 2

3 1 1 1

6 1 1 0

7 0 1 1

9 6 2 4

10 3 1 0

NO\_OF\_MOB\_DR\_TXNS NO\_OF\_CHQ\_DR\_TXNS FLG\_HAS\_CC AMT\_ATM\_DR

1 0 0 0 13100

3 0 2 0 11200

6 0 0 1 18500

7 0 4 0 6200

9 1 1 1 35400

10 0 4 0 18000

AMT\_BR\_CSH\_WDL\_DR AMT\_CHQ\_DR AMT\_NET\_DR AMT\_MOB\_DR AMT\_L\_DR

1 0 0 973557 0 986657

3 561120 49320 997570 0 1619210

6 379310 0 0 0 397810

7 0 10580 770065 0 786845

9 198430 51490 326421 170332 782073

10 869880 32610 0 0 920490

FLG\_HAS\_ANY\_CHGS AMT\_OTH\_BK\_ATM\_USG\_CHGS AMT\_MIN\_BAL\_NMC\_CHGS

1 0 0 0

3 1 0 0

6 0 0 0

7 1 0 0

9 0 0 0

10 0 0 0

NO\_OF\_IW\_CHQ\_BNC\_TXNS NO\_OF\_OW\_CHQ\_BNC\_TXNS AVG\_AMT\_PER\_ATM\_TXN

1 0 0 13100

3 0 1 11200

6 0 0 18500

7 0 0 6200

9 0 0 17700

10 0 0 18000

AVG\_AMT\_PER\_CSH\_WDL\_TXN AVG\_AMT\_PER\_CHQ\_TXN AVG\_AMT\_PER\_NET\_TXN

1 0.00 0.0 486778.50

3 561120.00 24660.0 997570.00

6 379310.00 0.0 0.00

7 0.00 2645.0 770065.00

9 33071.67 51490.0 81605.25

10 289960.00 8152.5 0.00

AVG\_AMT\_PER\_MOB\_TXN FLG\_HAS\_NOMINEE FLG\_HAS\_OLD\_LOAN random

1 0 1 1 0.000011400

3 0 1 1 0.000119954

6 0 1 1 0.000405840

7 0 0 1 0.000499109

9 170332 1 1 0.000522769

10 0 0 0 0.000570937

predict.class predict.score.0 predict.score.1

1 0 0.89230769 0.10769231

3 0 0.70000000 0.30000000

6 0 0.86079295 0.13920705

7 0 0.91372549 0.08627451

9 0 0.73913043 0.26086957

10 0 0.86898839 0.1310116

> 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(dev\_cart$predict.score)

[1] "matrix"

> ## deciling

> dev\_cart$deciles <- decile(dev\_cart$predict.score[,2])

> view(dev\_cart)

Error in view(dev\_cart) : could not find function "view"

> View(dev\_cart)

> install.packages("data.table")

Error in install.packages : Updating loaded packages

> install.packages("scales")

Error in install.packages : Updating loaded packages

> library(data.table)

> library(scales)

> tmp\_DT\_cart = data.table(dev\_cart)

> rank <- tmp\_DT\_cart[, 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)

package ‘scales’ is in use and will not be installed

**Rank Order table –** KS is 32.48%, this shows how good is model

| deciles | |  | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 |  | | 10 | | 1963 | | 668 | | 1295 | | 34.0% | | 668 | | 1295 | | 37.9% | | 10.5% | | 27.42 |
| 2 |  | | 9 | | 1709 | | 266 | | 1443 | | 15.6% | | 934 | | 2738 | | 53.0% | | 22.2% | | 30.81 |
| 3 |  | | 8 | | 1222 | | 171 | | 1051 | | 14.0% | | 1105 | | 3789 | | 62.7% | | 30.8% | | 31.99 |
| 4 |  | | 7 | | 907 | | 121 | | 786 | | 13.3% | | 1226 | | 4575 | | 69.6% | | 37.1% | | 32.48 |
| 5 |  | | 6 | | 3422 | | 329 | | 3093 | | 9.6% | | 1555 | | 7668 | | 88.3% | | 62.2% | | 26.05 |
| 6 |  | | 4 | | 1269 | | 94 | | 1175 | | 7.4% | | 1649 | | 8843 | | 93.6% | | 71.8% | | 21.86 |
| 7 |  | | 3 | | 3588 | | 112 | | 3476 | | 3.1% | | 1761 | | 12319 | | 100.0% | | 100.0% | | 0.00 |

> library(ROCR)

> library(ineq)

Warning message:

package ‘ineq’ was built under R version 3.5.2

> pred <- prediction(dev\_cart$predict.score[,2],dev\_cart$TARGET)

Error: $ operator is invalid for atomic vectors

> perf <- performance(pred, "tpr", "fpr")

> plot(perf)

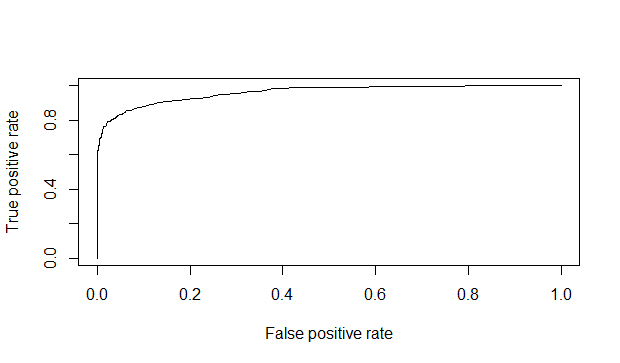
> KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])

> auc <- performance(pred,"auc");

> auc <- as.numeric(auc@y.values)

>

> gini = ineq(dev\_cart$predict.score[,2], type="Gini")



> with(dev\_cart, table(TARGET, predict.class))

predict.class

TARGET 0 1

0 12159 160

1 1423 338

**Classification error:** (160+1423)/14080

[1] 0.112429

11%

**•Test Model Performance on Hold Out Sample**

> hold\_out\_cart$predict.class <- predict(p\_m\_cart, hold\_out\_cart, type="class")

> hold\_out\_cart$predict.score <- predict(p\_m\_cart, hold\_out\_cart, type="prob")

>

>

> hold\_out\_cart$deciles <- decile(hold\_out\_cart$predict.score[,2])

>

>

> pred <- prediction(hold\_out\_cart$predict.score[,2],hold\_out\_cart$TARGET)

> perf <- performance(pred, "tpr", "fpr")

> plot(perf)

> tmp\_DT\_cart = data.table(hold\_out\_cart)

> h\_rank <- tmp\_DT\_cart[, 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)

>

> install.packages("ROCR")

Error in install.packages : Updating loaded packages

> install.packages("ineq")

Error in install.packages : Updating loaded packages

> library(ROCR)

> library(ineq)

> pred <- prediction(hold\_out\_cart$predict.score[,2],hold\_out\_cart$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(hold\_out\_cart$predict.score[,2], type="Gini")

>

> ##classification error, confusion matrix

> with(hold\_out\_cart, table(TARGET, predict.class))

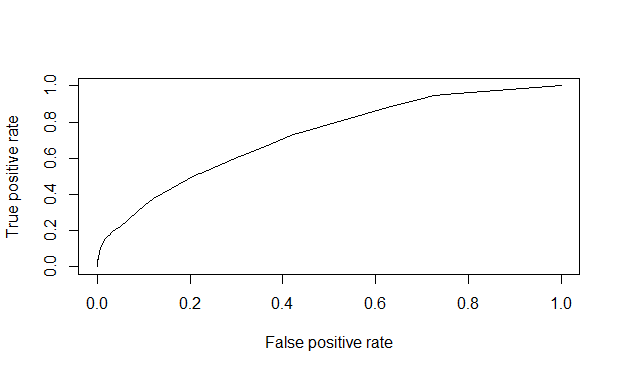
predict.class

TARGET 0 1

0 5088 81

1 640 111

| deciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 799 | | 259 | | 540 | | 32.4% | | 259 | | 540 | | 34.5% | | 10.4% | | 24.04 |
| 2 | 9 | | 696 | | 125 | | 571 | | 18.0% | | 384 | | 1111 | | 51.1% | | 21.5% | | 29.64 |
| 3 | 8 | | 546 | | 76 | | 470 | | 13.9% | | 460 | | 1581 | | 61.2% | | 30.6% | | 30.66 |
| 4 | 7 | | 356 | | 45 | | 311 | | 12.6% | | 505 | | 1892 | | 67.2% | | 36.6% | | 30.64 |
| 5 | 6 | | 1503 | | 157 | | 1346 | | 10.4% | | 662 | | 3238 | | 88.1% | | 62.6% | | 25.51 |
| 6 | 4 | | 487 | | 43 | | 444 | | 8.8% | | 705 | | 3682 | | 93.9% | | 71.2% | | 22.64 |
| 7 | 3 | | 1533 | | 46 | | 1487 | | 3.0% | | 751 | | 5169 | | 100.0% | | 100.0% | | 0.0 |



**Classification error is 12% - model is not overfitted**

(81+640)/5920

[1] 0.1217905

> auc

[1] 0.7229591

> KS

[1] 0.3073562

> gini

[1] 0.404381

**•Ensure the model is not an overfit model –**

Model is **not overfitted** as model behaves almost same for training and test data set.

KS value and classification error are almost same for both the data set.

**Part 2 | Random Forest**

**•Split data into Development (70%) and Hold-out (30%) Sample**

> loan\_ranForest=read.table("PL\_XSELL.csv",sep = ",",header = T)

> View(loan\_ranForest)

> #converting acc\_op\_date to date format

> loan\_ranForest$ACC\_OP\_DATE = as.Date(loan\_ranForest$ACC\_OP\_DATE, format="%m/%d/%Y")

>

>

> #split data in test and train data set

>

> ind = sample(2,nrow(loan\_ranForest),replace = TRUE ,prob = c(.7,.3))

> dev\_ranForest = loan\_ranForest[ind==1,]

> test\_ranForest = loan\_ranForest[ind==2,]

> c(nrow(dev\_ranForest),nrow(test\_ranForest))

[1] 14052 5948

> str(loan\_ranForest)

'data.frame': 20000 obs. of 40 variables:

$ CUST\_ID : Factor w/ 20000 levels "C1","C10","C100",..: 17699 16532 11027 17984 2363 11747 18115 15556 15216 12494 ...

$ TARGET : int 0 0 0 0 0 0 0 0 0 0 ...

$ AGE : int 27 47 40 53 36 42 30 53 42 30 ...

$ GENDER : Factor w/ 3 levels "F","M","O": 2 2 2 2 2 1 2 1 1 2 ...

$ BALANCE : num 3384 287489 18217 71720 1671623 ...

$ OCCUPATION : Factor w/ 4 levels "PROF","SAL","SELF-EMP",..: 3 2 3 2 1 1 1 2 3 1 ...

$ AGE\_BKT : Factor w/ 7 levels "<25",">50","26-30",..: 3 7 5 2 5 6 3 2 6 3 ...

$ SCR : int 776 324 603 196 167 493 479 562 105 170 ...

$ HOLDING\_PERIOD : int 30 28 2 13 24 26 14 25 15 13 ...

$ ACC\_TYPE : Factor w/ 2 levels "CA","SA": 2 2 2 1 2 2 2 1 2 2 ...

$ ACC\_OP\_DATE : Date, format: "2005-03-23" ...

$ LEN\_OF\_RLTN\_IN\_MNTH : int 146 104 61 107 185 192 177 99 88 111 ...

$ NO\_OF\_L\_CR\_TXNS : int 7 8 10 36 20 5 6 14 18 14 ...

$ NO\_OF\_L\_DR\_TXNS : int 3 2 5 14 1 2 6 3 14 8 ...

$ TOT\_NO\_OF\_L\_TXNS : int 10 10 15 50 21 7 12 17 32 22 ...

$ NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS: int 0 0 1 4 1 1 0 3 6 3 ...

$ NO\_OF\_ATM\_DR\_TXNS : int 1 1 1 2 0 1 1 0 2 1 ...

$ NO\_OF\_NET\_DR\_TXNS : int 2 1 1 3 0 0 1 0 4 0 ...

$ NO\_OF\_MOB\_DR\_TXNS : int 0 0 0 1 0 0 0 0 1 0 ...

$ NO\_OF\_CHQ\_DR\_TXNS : int 0 0 2 4 0 0 4 0 1 4 ...

$ FLG\_HAS\_CC : int 0 0 0 0 0 1 0 0 1 0 ...

$ AMT\_ATM\_DR : int 13100 6600 11200 26100 0 18500 6200 0 35400 18000 ...

$ AMT\_BR\_CSH\_WDL\_DR : int 0 0 561120 673590 808480 379310 0 945160 198430 869880 ...

$ AMT\_CHQ\_DR : int 0 0 49320 60780 0 0 10580 0 51490 32610 ...

$ AMT\_NET\_DR : num 973557 799813 997570 741506 0 ...

$ AMT\_MOB\_DR : int 0 0 0 71388 0 0 0 0 170332 0 ...

$ AMT\_L\_DR : num 986657 806413 1619210 1573364 808480 ...

$ FLG\_HAS\_ANY\_CHGS : int 0 1 1 0 0 0 1 0 0 0 ...

$ AMT\_OTH\_BK\_ATM\_USG\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ AMT\_MIN\_BAL\_NMC\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_IW\_CHQ\_BNC\_TXNS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_OW\_CHQ\_BNC\_TXNS : int 0 0 1 0 0 0 0 0 0 0 ...

$ AVG\_AMT\_PER\_ATM\_TXN : num 13100 6600 11200 13050 0 ...

$ AVG\_AMT\_PER\_CSH\_WDL\_TXN : num 0 0 561120 168398 808480 ...

$ AVG\_AMT\_PER\_CHQ\_TXN : num 0 0 24660 15195 0 ...

$ AVG\_AMT\_PER\_NET\_TXN : num 486779 799813 997570 247169 0 ...

$ AVG\_AMT\_PER\_MOB\_TXN : num 0 0 0 71388 0 ...

$ FLG\_HAS\_NOMINEE : int 1 1 1 1 1 1 0 1 1 0 ...

$ FLG\_HAS\_OLD\_LOAN : int 1 0 1 0 0 1 1 1 1 0 ...

$ random : num 1.14e-05 1.11e-04 1.20e-04 1.37e-04 1.74e-04 ...

>

> #table(loan\_ranForest$OCCUPATION)

>

>

>

>

**•Build Model using Random Forest technique**

library(randomForest)

>

> ## Calling syntax to build the Random Forest

> RF <- randomForest(as.factor(TARGET) ~ ., data = dev\_ranForest[,-1],

+ ntree=501, mtry = 10, nodesize = 25,

+ importance=TRUE)

> print(RF)

Call:

randomForest(formula = as.factor(TARGET) ~ ., data = dev\_ranForest[, -1], ntree = 501, mtry = 10, nodesize = 25, importance = TRUE)

Type of random forest: classification

Number of trees: 501

No. of variables tried at each split: 10

OOB estimate of error rate: 8.76%

Confusion matrix:

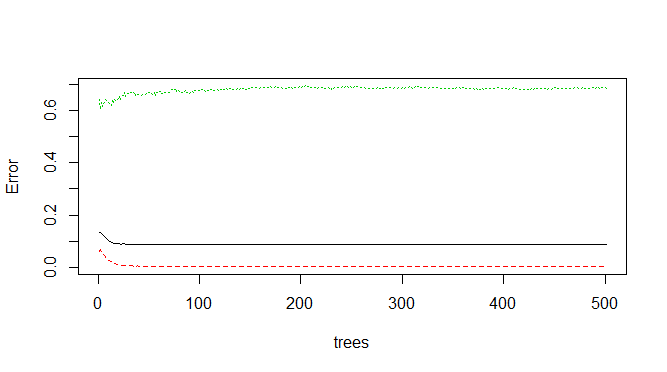
0 1 class.error

0 12269 21 0.001708706

1 1210 552 0.686719637

**Optimum value of ntree :**

> plot(RF, main="")



RF$err.rate##OOB stablizes around 40-70 tree

OOB 0 1

[1,] 0.13291998 0.061242538 0.6410658

[2,] 0.13372024 0.069225589 0.6041257

[3,] 0.13043065 0.060875217 0.6293706

[4,] 0.12584459 0.056498265 0.6158038

[5,] 0.12128183 0.049619013 0.6203411

[6,] 0.11912011 0.045320111 0.6358100

[7,] 0.11508496 0.039375424 0.6420555

[8,] 0.11049482 0.035037958 0.6379009

[9,] 0.10758671 0.031964979 0.6358915

[10,] 0.10301941 0.027366864 0.6314581

[11,] 0.09956989 0.024094411 0.6264302

[12,] 0.09865599 0.022482014 0.6292711

[13,] 0.09609074 0.020719471 0.6213758

[14,] 0.09577425 0.017682529 0.6399773

[15,] 0.09215212 0.014575360 0.6331630

[16,] 0.09267564 0.013998535 0.6413167

[17,] 0.09153025 0.012776693 0.6407491

[18,] 0.08996441 0.011881510 0.6345062

[19,] 0.09060498 0.011555990 0.6418842

[20,] 0.09024269 0.010334445 0.6475596

[21,] 0.09059209 0.009926770 0.6532350

[22,] 0.08817250 0.008787632 0.6418842

[23,] 0.08909764 0.008380797 0.6520999

[24,] 0.08931113 0.007648495 0.6589103

[25,] 0.08966695 0.008136697 0.6583428

[26,] 0.09023627 0.007241660 0.6691260

[27,] 0.08845716 0.007078926 0.6560726

[28,] 0.08881298 0.006753458 0.6611805

[29,] 0.08867065 0.006021155 0.6651532

[30,] 0.08881298 0.005939788 0.6668558

[31,] 0.08909764 0.005777055 0.6702611

[32,] 0.08902647 0.005777055 0.6696935

[33,] 0.08852832 0.005370220 0.6685585

[34,] 0.08810134 0.005207486 0.6662883

[35,] 0.08831483 0.004719284 0.6713961

[36,] 0.08774552 0.004963385 0.6651532

[37,] 0.08724737 0.005207486 0.6594779

[38,] 0.08738970 0.004637917 0.6645857

[39,] 0.08738970 0.005044752 0.6617480

[40,] 0.08710504 0.004719284 0.6617480

[41,] 0.08696271 0.004556550 0.6617480

[42,] 0.08660689 0.004231082 0.6611805

[43,] 0.08568175 0.003580146 0.6583428

[44,] 0.08646456 0.003986981 0.6617480

[45,] 0.08653572 0.003986981 0.6623156

[46,] 0.08646456 0.003986981 0.6617480

[47,] 0.08617990 0.003742880 0.6611805

[48,] 0.08632223 0.003498779 0.6640182

[49,] 0.08710504 0.003661513 0.6691260

[50,] 0.08724737 0.003661513 0.6702611

[51,] 0.08710504 0.003498779 0.6702611

[52,] 0.08653572 0.003580146 0.6651532

[53,] 0.08646456 0.003580146 0.6645857

[54,] 0.08582408 0.003498779 0.6600454

[55,] 0.08660689 0.003417413 0.6668558

[56,] 0.08682038 0.003336046 0.6691260

[57,] 0.08582408 0.003580146 0.6594779

[58,] 0.08717620 0.003580146 0.6702611

[59,] 0.08703387 0.003417413 0.6702611

[60,] 0.08746086 0.003254679 0.6748014

[61,] 0.08710504 0.003091945 0.6730988

[62,] 0.08689155 0.003010578 0.6719637

[63,] 0.08589525 0.002847844 0.6651532

[64,] 0.08610874 0.002847844 0.6668558

[65,] 0.08582408 0.002685110 0.6657208

[66,] 0.08667805 0.002847844 0.6713961

[67,] 0.08603757 0.002603743 0.6679909

[68,] 0.08603757 0.002359642 0.6696935

[69,] 0.08653572 0.002685110 0.6713961

[70,] 0.08639340 0.002685110 0.6702611

[71,] 0.08710504 0.002929211 0.6742338

[72,] 0.08760319 0.002359642 0.6821793

[73,] 0.08753202 0.002441009 0.6810443

[74,] 0.08795901 0.002603743 0.6833144

[75,] 0.08774552 0.002766477 0.6804767

[76,] 0.08746086 0.002603743 0.6793417

[77,] 0.08653572 0.002441009 0.67309

Optimal number of trees are around ~55

To get the important variable: Most important variables are Balance and SCR.

> impVar <- round(randomForest::importance(RF), 2)

> impVar[order(impVar[,3], decreasing=TRUE),]

0 1 MeanDecreaseAccuracy

OCCUPATION 44.58 51.89 55.21

BALANCE 39.52 57.64 55.04

AGE\_BKT 36.60 52.06 53.39

SCR 39.76 51.73 51.85

HOLDING\_PERIOD 33.41 40.54 43.51

FLG\_HAS\_CC 34.02 41.81 41.29

AMT\_L\_DR 37.19 17.33 40.81

NO\_OF\_L\_CR\_TXNS 32.28 17.01 36.63

AGE 26.32 33.05 34.99

TOT\_NO\_OF\_L\_TXNS 30.17 7.86 33.84

AVG\_AMT\_PER\_CSH\_WDL\_TXN 26.76 26.33 31.39

ACC\_OP\_DATE 25.42 24.36 30.39

AMT\_BR\_CSH\_WDL\_DR 25.57 29.23 30.04

LEN\_OF\_RLTN\_IN\_MNTH 24.80 23.24 28.76

AVG\_AMT\_PER\_CHQ\_TXN 22.54 9.92 24.59

AMT\_CHQ\_DR 21.93 15.34 23.98

GENDER 21.92 18.56 23.64

AVG\_AMT\_PER\_NET\_TXN 19.21 20.88 22.91

NO\_OF\_L\_DR\_TXNS 21.55 10.07 22.17

AMT\_ATM\_DR 19.16 11.75 21.85

AMT\_NET\_DR 16.83 16.38 19.19

AMT\_MOB\_DR 14.64 11.71 18.11

AVG\_AMT\_PER\_ATM\_TXN 15.35 8.34 17.66

AVG\_AMT\_PER\_MOB\_TXN 13.33 16.26 17.19

NO\_OF\_CHQ\_DR\_TXNS 16.01 12.59 16.74

FLG\_HAS\_ANY\_CHGS 13.25 14.61 16.12

NO\_OF\_ATM\_DR\_TXNS 15.94 -9.42 16.04

NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS 13.26 16.69 15.44

FLG\_HAS\_OLD\_LOAN 9.68 14.95 14.29

NO\_OF\_IW\_CHQ\_BNC\_TXNS 7.14 14.41 14.26

ACC\_TYPE 10.67 7.16 12.23

NO\_OF\_OW\_CHQ\_BNC\_TXNS 6.10 11.98 10.90

FLG\_HAS\_NOMINEE 6.73 10.20 10.54

NO\_OF\_NET\_DR\_TXNS 6.58 6.76 7.42

AMT\_MIN\_BAL\_NMC\_CHGS 2.66 6.05 5.45

NO\_OF\_MOB\_DR\_TXNS 4.22 -0.33 4.49

AMT\_OTH\_BK\_ATM\_USG\_CHGS -0.49 3.05 0.94

random 0.33 0.49 0.52

MeanDecreaseGini

OCCUPATION 57.28

BALANCE 127.90

AGE\_BKT 82.73

SCR 119.72

HOLDING\_PERIOD 89.13

FLG\_HAS\_CC 28.85

AMT\_L\_DR 85.80

NO\_OF\_L\_CR\_TXNS 90.42

AGE 55.00

TOT\_NO\_OF\_L\_TXNS 90.38

AVG\_AMT\_PER\_CSH\_WDL\_TXN 67.80

ACC\_OP\_DATE 84.24

AMT\_BR\_CSH\_WDL\_DR 72.91

LEN\_OF\_RLTN\_IN\_MNTH 68.86

AVG\_AMT\_PER\_CHQ\_TXN 53.24

AMT\_CHQ\_DR 55.27

GENDER 20.77

AVG\_AMT\_PER\_NET\_TXN 48.30

NO\_OF\_L\_DR\_TXNS 45.49

AMT\_ATM\_DR 60.00

AMT\_NET\_DR 46.36

AMT\_MOB\_DR 27.99

AVG\_AMT\_PER\_ATM\_TXN 62.03

AVG\_AMT\_PER\_MOB\_TXN 30.17

NO\_OF\_CHQ\_DR\_TXNS 24.10

FLG\_HAS\_ANY\_CHGS 9.04

NO\_OF\_ATM\_DR\_TXNS 21.11

NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS 26.33

FLG\_HAS\_OLD\_LOAN 6.37

NO\_OF\_IW\_CHQ\_BNC\_TXNS 8.01

ACC\_TYPE 7.64

NO\_OF\_OW\_CHQ\_BNC\_TXNS 6.54

FLG\_HAS\_NOMINEE 5.77

NO\_OF\_NET\_DR\_TXNS 10.11

AMT\_MIN\_BAL\_NMC\_CHGS 1.25

NO\_OF\_MOB\_DR\_TXNS 3.20

AMT\_OTH\_BK\_ATM\_USG\_CHGS 0.48

random 40.69

**Optimum value of mtry: 28**

tRF <- tuneRF(x = dev\_ranForest[,-c(1,2)],

+ y=as.factor(dev\_ranForest$TARGET),

+ mtryStart = 6, ##sqrt of predictor variables

+ ntreeTry=55, ##OOB Stablizes (should be odd number)

+ stepFactor = 1.5,

+ improve = 0.0001,

+ trace=TRUE,

+ plot = TRUE,

+ doBest = TRUE,

+ nodesize = 10, ###need to change for overfitting prob

+ importance=TRUE

+ )

mtry = 6 OOB error = 6.26%

Searching left ...

mtry = 4 OOB error = 7.02%

-0.1217292 1e-04

Searching right ...

mtry = 9 OOB error = 6.03%

0.03526735 1e-04

mtry = 13 OOB error = 5.73%

0.05070755 1e-04

mtry = 19 OOB error = 5.71%

0.002484472 1e-04

mtry = 28 OOB error = 5.46%

0.04483188 1e-04

mtry = 38 OOB error = 5.52%

-0.01173403 1e-04

**Tuned RF model:**

> tRF$importance

0 1 MeanDecreaseAccuracy

AGE 5.382967e-03 0.0338760625 8.959923e-03

GENDER 2.721520e-03 0.0118315106 3.865665e-03

BALANCE 1.245187e-02 0.1105077644 2.475787e-02

OCCUPATION 1.369393e-02 0.1106226943 2.586057e-02

AGE\_BKT 1.068447e-02 0.0843537452 1.992991e-02

SCR 1.332695e-02 0.1060083683 2.496365e-02

HOLDING\_PERIOD 3.528354e-02 0.1221766414 4.619281e-02

ACC\_TYPE 4.291756e-03 0.0059103001 4.497163e-03

ACC\_OP\_DATE 1.789038e-02 0.0376485068 2.036467e-02

LEN\_OF\_RLTN\_IN\_MNTH 1.586967e-02 0.0237929209 1.685748e-02

NO\_OF\_L\_CR\_TXNS 5.071279e-02 0.0478461678 5.034706e-02

NO\_OF\_L\_DR\_TXNS 8.630635e-02 0.0461092253 8.127226e-02

TOT\_NO\_OF\_L\_TXNS 6.049033e-02 0.0328075417 5.701676e-02

NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS 5.100091e-03 0.0130865842 6.102211e-03

NO\_OF\_ATM\_DR\_TXNS 3.221723e-02 -0.0014909674 2.798035e-02

NO\_OF\_NET\_DR\_TXNS 1.791653e-03 0.0015246632 1.758651e-03

NO\_OF\_MOB\_DR\_TXNS 4.644891e-04 0.0004597811 4.632000e-04

NO\_OF\_CHQ\_DR\_TXNS 1.118673e-02 0.0118149733 1.126763e-02

FLG\_HAS\_CC 1.256740e-02 0.1106195770 2.488390e-02

AMT\_ATM\_DR 2.676160e-02 0.0277953134 2.688980e-02

AMT\_BR\_CSH\_WDL\_DR 2.133041e-02 0.0334098087 2.284831e-02

AMT\_CHQ\_DR 1.967222e-02 0.0225531185 2.002928e-02

AMT\_NET\_DR 1.042795e-02 0.0161917504 1.115015e-02

AMT\_MOB\_DR 6.834710e-03 0.0098615509 7.215586e-03

AMT\_L\_DR 3.438836e-02 0.0289207692 3.370129e-02

FLG\_HAS\_ANY\_CHGS 1.133219e-03 0.0063884772 1.792410e-03

AMT\_OTH\_BK\_ATM\_USG\_CHGS -7.539173e-06 0.0001306875 1.004507e-05

AMT\_MIN\_BAL\_NMC\_CHGS 1.813155e-05 0.0001923660 3.987118e-05

NO\_OF\_IW\_CHQ\_BNC\_TXNS 3.986035e-04 0.0035767104 7.973922e-04

NO\_OF\_OW\_CHQ\_BNC\_TXNS 3.828884e-04 0.0033700196 7.572610e-04

AVG\_AMT\_PER\_ATM\_TXN 1.975044e-02 0.0263603438 2.057732e-02

AVG\_AMT\_PER\_CSH\_WDL\_TXN 1.751287e-02 0.0334176404 1.950951e-02

AVG\_AMT\_PER\_CHQ\_TXN 1.763577e-02 0.0173047031 1.759084e-02

AVG\_AMT\_PER\_NET\_TXN 1.140668e-02 0.0209881882 1.260862e-02

AVG\_AMT\_PER\_MOB\_TXN 7.463806e-03 0.0179320207 8.778058e-03

FLG\_HAS\_NOMINEE 4.101796e-04 0.0023500542 6.543794e-04

FLG\_HAS\_OLD\_LOAN 6.378672e-04 0.0052484114 1.215700e-03

random -2.483661e-05 0.0005035168 4.354671e-05

MeanDecreaseGini

AGE 90.0543433

GENDER 30.3764568

BALANCE 204.0722602

OCCUPATION 79.1863895

AGE\_BKT 135.8580814

SCR 203.1984601

HOLDING\_PERIOD 130.9780622

ACC\_TYPE 10.8064920

ACC\_OP\_DATE 123.1094640

LEN\_OF\_RLTN\_IN\_MNTH 86.8635286

NO\_OF\_L\_CR\_TXNS 119.0933793

NO\_OF\_L\_DR\_TXNS 59.8087744

TOT\_NO\_OF\_L\_TXNS 109.2642406

NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS 38.3721880

NO\_OF\_ATM\_DR\_TXNS 22.7331325

NO\_OF\_NET\_DR\_TXNS 8.3569120

NO\_OF\_MOB\_DR\_TXNS 2.7607955

NO\_OF\_CHQ\_DR\_TXNS 31.3711604

FLG\_HAS\_CC 33.3445152

AMT\_ATM\_DR 81.7625687

AMT\_BR\_CSH\_WDL\_DR 109.3282510

AMT\_CHQ\_DR 76.2436523

AMT\_NET\_DR 62.9634517

AMT\_MOB\_DR 35.5560409

AMT\_L\_DR 130.2968560

FLG\_HAS\_ANY\_CHGS 13.8615375

AMT\_OTH\_BK\_ATM\_USG\_CHGS 0.6659140

AMT\_MIN\_BAL\_NMC\_CHGS 0.7727809

NO\_OF\_IW\_CHQ\_BNC\_TXNS 11.7283866

NO\_OF\_OW\_CHQ\_BNC\_TXNS 10.3216445

AVG\_AMT\_PER\_ATM\_TXN 85.6749103

AVG\_AMT\_PER\_CSH\_WDL\_TXN 103.5769798

AVG\_AMT\_PER\_CHQ\_TXN 75.3661138

AVG\_AMT\_PER\_NET\_TXN 68.6901821

AVG\_AMT\_PER\_MOB\_TXN 43.9574806

FLG\_HAS\_NOMINEE 10.2047427

FLG\_HAS\_OLD\_LOAN 12.5347237

random 70.2032496

**•Measure Model Performance on Development Sample**

> dev\_ranForest$predict.class = predict(tRF,dev\_ranForest, type="class")

> dev\_ranForest$predict.score = predict(tRF,dev\_ranForest, type="prob")

## deciling

> ## 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

+ ))))))))))

+ }

>

>

> dev\_ranForest$deciles <- decile(dev\_ranForest$predict.score[,2])

>

>

> library(data.table)

> tmp\_DT = data.table(dev\_ranForest)

> 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)

KS value is too high – model can be overfitted

| eciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 1408 | | 1408 | | 0 | | 100% | | 1408 | | 0 | | 80.0% | | 0% | | 0.80 |
| 2 | 9 | | 1431 | | 354 | | 1077 | | 25% | | 1762 | | 1077 | | 100.0% | | 9% | | 0.91 |
| 3 | 8 | | 1487 | | 0 | | 1487 | | 0% | | 1762 | | 2564 | | 100.0% | | 21% | | 0.79 |
| 4 | 7 | | 1357 | | 0 | | 1357 | | 0% | | 1762 | | 3921 | | 100.0% | | 32% | | 0.68 |
| 5 | 6 | | 1581 | | 0 | | 1581 | | 0% | | 1762 | | 5502 | | 100.0% | | 45% | | 0.55 |
| 6 | 5 | | 1225 | | 0 | | 1225 | | 0% | | 1762 | | 6727 | | 100.0% | | 55% | | 0.45 |
| 7 | 4 | | 1483 | | 0 | | 1483 | | 0% | | 1762 | | 8210 | | 100.0% | | 67% | | 0.33 |
| 8 | 3 | | 1399 | | 0 | | 1399 | | 0% | | 1762 | | 9609 | | 100.0% | | 78% | | 0.22 |
| 9 | 2 | | 1907 | | 0 | | 1907 | | 0% | | 1762 | | 11516 | | 100.0% | | 94% | | 0.06 |
| 10 | 1 | | 774 | | 0 | | 774 | | 0% | | 1762 | | 12290 | | 100.0% | | 100% | | 0.0 |

> with(dev\_ranForest, table(TARGET, predict.class))

predict.class

TARGET 0 1

0 12290 0

1 172 1590

> (172)/14052

[1] 0.01224025

**Classification error is 1.2%**

**•Test Model Performance on Hold Out Sample**

> test\_ranForest$predict.class <- predict(tRF, test\_ranForest, type="class")

> test\_ranForest$predict.score <- predict(tRF, test\_ranForest, type="prob")

>

> test\_ranForest$deciles <- decile(test\_ranForest$predict.score[,2])

>

> tmp\_DT = data.table(test\_ranForest)

> 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)

| eciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 598 | | 517 | | 81 | | 86.0% | | 517 | | 81 | | 69.0% | | 2.0% | | 0.67 |
| 2 | 9 | | 600 | | 108 | | 492 | | 18.0% | | 625 | | 573 | | 83.0% | | 11.0% | | 0.72 |
| 3 | 8 | | 607 | | 66 | | 541 | | 11.0% | | 691 | | 1114 | | 92.0% | | 21.0% | | 0.71 |
| 4 | 7 | | 606 | | 13 | | 593 | | 2.0% | | 704 | | 1707 | | 94.0% | | 33.0% | | 0.61 |
| 5 | 6 | | 605 | | 22 | | 583 | | 4.0% | | 726 | | 2290 | | 97.0% | | 44.0% | | 0.53 |
| 6 | 5 | | 613 | | 12 | | 601 | | 2.0% | | 738 | | 2891 | | 98.0% | | 56.0% | | 0.42 |
| 7 | 4 | | 567 | | 7 | | 560 | | 1.0% | | 745 | | 3451 | | 99.0% | | 66.0% | | 0.33 |
| 8 | 3 | | 674 | | 3 | | 671 | | 0.0% | | 748 | | 4122 | | 100.0% | | 79.0% | | 0.21 |
| 9 | 2 | | 552 | | 0 | | 552 | | 0.0% | | 748 | | 4674 | | 100.0% | | 90.0% | | 0.10 |
| 10 | 1 | | 526 | | 2 | | 524 | | 0.0% | | 750 | | 5198 | | 100.0% | | 100.0% | | 0.00 |

There is more then 5% difference in dev and test sample , hence model is overfitted

#classification error

> (15+333)/5920

[1] 0.05878378

**•Ensure the model is not an overfit model**

Model is **overfitted** , need to check while changing parameter .

Lets check with reduce # of trees and and we see the OOB error reduce to 3%

> RF <- randomForest(as.factor(TARGET) ~ ., data = dev\_ranForest[,-1],

+ ntree=101, mtry = 28,replace=FALSE, nodesize = 10,

+ importance=TRUE)

> print(RF)

Call:

randomForest(formula = as.factor(TARGET) ~ ., data = dev\_ranForest[, -1], ntree = 101, mtry = 28, replace = FALSE, nodesize = 10, importance = TRUE)

Type of random forest: classification

Number of trees: 101

No. of variables tried at each split: 28

OOB estimate of error rate: 5.73%

Confusion matrix:

0 1 class.error

0 12089 53 0.004365014

1 743 999 0.426521240

**Part 3 | Neural Network**

**•Split data into Development (70%) and Hold-out (30%) Sample**

loan\_NN =read.table("PL\_XSELL.csv",sep = ",",header = T)

> table(loan\_NN$TARGET)

0 1

17488 2512

> ind = sample(2,nrow(loan\_NN),replace = TRUE ,prob = c(.7,.3))

> dev\_NN = loan\_NN[ind==1,]

> hold\_out\_NN = loan\_NN[ind==2,]

**Convert the categorical variables**

occ.matrix <- model.matrix(~ OCCUPATION - 1, data = dev\_NN)

> dev\_NN <- data.frame(dev\_NN, occ.matrix)

>

> cc.matrix <- model.matrix(~ AGE\_BKT - 1, data = dev\_NN)

> dev\_NN <- data.frame(dev\_NN, occ.matrix)

>

> cc.matrix <- model.matrix(~ ACC\_TYPE - 1, data = dev\_NN)

> dev\_NN <- data.frame(dev\_NN, occ.matrix)

> str(dev\_NN)

'data.frame': 14008 obs. of 70 variables:

$ CUST\_ID : Factor w/ 20000 levels "C1","C10","C100",..: 17699 11027 17984 2363 11747 15556 15216 11149 12489 5037 ...

$ TARGET : int 0 0 0 0 0 0 0 0 0 0 ...

$ AGE : int 27 40 53 36 42 53 42 43 53 43 ...

$ GENDER : Factor w/ 3 levels "F","M","O": 2 2 2 2 1 1 1 2 2 1 ...

$ BALANCE : num 3384 18217 71720 1671623 521686 ...

$ OCCUPATION : Factor w/ 4 levels "PROF","SAL","SELF-EMP",..: 3 3 2 1 1 2 3 1 4 4 ...

$ AGE\_BKT : Factor w/ 7 levels "<25",">50","26-30",..: 3 5 2 5 6 2 6 6 2 6 ...

$ SCR : int 776 603 196 167 493 562 105 164 239 912 ...

$ HOLDING\_PERIOD : int 30 2 13 24 26 25 15 24 20 3 ...

$ ACC\_TYPE : Factor w/ 2 levels "CA","SA": 2 2 1 2 2 1 2 1 2 2 ...

$ ACC\_OP\_DATE : Factor w/ 4869 levels "1/01/2000","1/01/2001",..: 2318 2780 3710 1606 3339 2208 1745 2096 4136 4855 ...

$ LEN\_OF\_RLTN\_IN\_MNTH : int 146 61 107 185 192 99 88 183 142 200 ...

$ NO\_OF\_L\_CR\_TXNS : int 7 10 36 20 5 14 18 4 5 4 ...

$ NO\_OF\_L\_DR\_TXNS : int 3 5 14 1 2 3 14 4 4 3 ...

$ TOT\_NO\_OF\_L\_TXNS : int 10 15 50 21 7 17 32 8 9 7 ...

$ NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS: int 0 1 4 1 1 3 6 2 1 2 ...

$ NO\_OF\_ATM\_DR\_TXNS : int 1 1 2 0 1 0 2 1 0 0 ...

$ NO\_OF\_NET\_DR\_TXNS : int 2 1 3 0 0 0 4 0 0 0 ...

$ NO\_OF\_MOB\_DR\_TXNS : int 0 0 1 0 0 0 1 1 0 0 ...

$ NO\_OF\_CHQ\_DR\_TXNS : int 0 2 4 0 0 0 1 0 3 1 ...

$ FLG\_HAS\_CC : int 0 0 0 0 1 0 1 1 1 1 ...

$ AMT\_ATM\_DR : int 13100 11200 26100 0 18500 0 35400 8900 0 0 ...

$ AMT\_BR\_CSH\_WDL\_DR : int 0 561120 673590 808480 379310 945160 198430 362780 270160 660120 ...

$ AMT\_CHQ\_DR : int 0 49320 60780 0 0 0 51490 0 18290 0 ...

$ AMT\_NET\_DR : num 973557 997570 741506 0 0 ...

$ AMT\_MOB\_DR : int 0 0 71388 0 0 0 170332 171334 0 0 ...

$ AMT\_L\_DR : num 986657 1619210 1573364 808480 397810 ...

$ FLG\_HAS\_ANY\_CHGS : int 0 1 0 0 0 0 0 0 0 0 ...

$ AMT\_OTH\_BK\_ATM\_USG\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ AMT\_MIN\_BAL\_NMC\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_IW\_CHQ\_BNC\_TXNS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_OW\_CHQ\_BNC\_TXNS : int 0 1 0 0 0 0 0 0 1 0 ...

$ AVG\_AMT\_PER\_ATM\_TXN : num 13100 11200 13050 0 18500 ...

$ AVG\_AMT\_PER\_CSH\_WDL\_TXN : num 0 561120 168398 808480 379310 ...

$ AVG\_AMT\_PER\_CHQ\_TXN : num 0 24660 15195 0 0 ...

$ AVG\_AMT\_PER\_NET\_TXN : num 486779 997570 247169 0 0 ...

$ AVG\_AMT\_PER\_MOB\_TXN : num 0 0 71388 0 0 ...

$ FLG\_HAS\_NOMINEE : int 1 1 1 1 1 1 1 1 1 1 ...

$ FLG\_HAS\_OLD\_LOAN : int 1 1 0 0 1 1 1 0 1 1 ...

$ random : num 1.14e-05 1.20e-04 1.37e-04 1.74e-04 4.06e-04 ...

$ GENDERF : num 0 0 0 0 1 1 1 0 0 1 ...

$ GENDERM : num 1 1 1 1 0 0 0 1 1 0 ...

$ GENDERO : num 0 0 0 0 0 0 0 0 0 0 ...

$ OCCUPATIONPROF : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP : num 0 0 0 0 0 0 0 0 1 1 ...

$ OCCUPATIONPROF.1 : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL.1 : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP.1 : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP.1 : num 0 0 0 0 0 0 0 0 1 1 ...

$ OCCUPATIONPROF.2 : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL.2 : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP.2 : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP.2 : num 0 0 0 0 0 0 0 0 1 1 ...

$ GENDERF.1 : num 0 0 0 0 1 1 1 0 0 1 ...

$ GENDERM.1 : num 1 1 1 1 0 0 0 1 1 0 ...

$ GENDERO.1 : num 0 0 0 0 0 0 0 0 0 0 ...

$ OCCUPATIONPROF.3 : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL.3 : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP.3 : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP.3 : num 0 0 0 0 0 0 0 0 1 1 ...

$ OCCUPATIONPROF.4 : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL.4 : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP.4 : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP.4 : num 0 0 0 0 0 0 0 0 1 1 ...

$ OCCUPATIONPROF.5 : num 0 0 0 1 1 0 0 1 0 0 ...

$ OCCUPATIONSAL.5 : num 0 0 1 0 0 1 0 0 0 0 ...

$ OCCUPATIONSELF.EMP.5 : num 1 1 0 0 0 0 1 0 0 0 ...

$ OCCUPATIONSENP.5 : num 0 0 0 0 0 0 0 0 1 1 ...

allvars = colnames(dev\_NN\_new)

> predictorVars = allvars[!allvars%in%"TARGET"]

> predictorVars = paste(predictorVars,collapse = "+")

> form=as.formula(paste("TARGET~",predictorVars,collapse = "+"))

> form

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 + OCCUPATIONPROF.1 + OCCUPATIONSAL.1 + OCCUPATIONSELF.EMP.1 +

OCCUPATIONSENP.1 + OCCUPATIONPROF.2 + OCCUPATIONSAL.2 + OCCUPATIONSELF.EMP.2 +

OCCUPATIONSENP.2 + GENDERF.1 + GENDERM.1 + GENDERO.1 + OCCUPATIONPROF.3 +

OCCUPATIONSAL.3 + OCCUPATIONSELF.EMP.3 + OCCUPATIONSENP.3 +

OCCUPATIONPROF.4 + OCCUPATIONSAL.4 + OCCUPATIONSELF.EMP.4 +

OCCUPATIONSENP.4 + OCCUPATIONPROF.5 + OCCUPATIONSAL.5 + OCCUPATIONSELF.EMP.5 +

OCCUPATIONSENP.5

**•Build Model using Neural Network technique**

> NN\_Model=neuralnet(formula=form,

+ data=dev\_NN\_new,

+ hidden = 2,

+ err.fct = "sse",

+ linear.output = FALSE,

+ lifesign = "full",

+ lifesign.step=10,

+ threshold = .01,

+ stepmax = 2000)

hidden: 2 thresh: 0.01 rep: 1/1 steps: 10 min thresh: 141.469940142666

20 min thresh: 3.96644876639281

30 min thresh: 2.49859987625657

40 min thresh: 1.38047313127015

50 min thresh: 1.20865695367495

60 min thresh: 0.728688255350948

70 min thresh: 0.470585767491499

80 min thresh: 0.470585767491499

90 min thresh: 0.447786217123128

100 min thresh: 0.447786217123128

110 min thresh: 0.447786217123128

120 min thresh: 0.447786217123128

130 min thresh: 0.447786217123128

140 min thresh: 0.447786217123128

150 min thresh: 0.447786217123128

160 min thresh: 0.447786217123128

170 min thresh: 0.447786217123128

180 min thresh: 0.447786217123128

190 min thresh: 0.0887226017629949

200 min thresh: 0.0887226017629949

210 min thresh: 0.0543098882518153

220 min thresh: 0.0453431597312622

230 min thresh: 0.0453431597312622

240 min thresh: 0.0453431597312622

250 min thresh: 0.0453431597312622

260 min thresh: 0.0453431597312622

270 min thresh: 0.0453431597312622

280 min thresh: 0.0453431597312622

290 min thresh: 0.0453431597312622

300 min thresh: 0.0453431597312622

310 min thresh: 0.0453431597312622

320 min thresh: 0.0453431597312622

330 min thresh: 0.0453431597312622

340 min thresh: 0.0453431597312622

350 min thresh: 0.0453431597312622

360 min thresh: 0.0453431597312622

370 min thresh: 0.0453431597312622

380 min thresh: 0.0453431597312622

390 min thresh: 0.0398706995559851

400 min thresh: 0.0398706995559851

410 min thresh: 0.0398706995559851

420 min thresh: 0.0398706995559851

430 min thresh: 0.0398706995559851

440 min thresh: 0.0398706995559851

450 min thresh: 0.0398706995559851

460 min thresh: 0.0398706995559851

470 min thresh: 0.0398706995559851

480 min thresh: 0.0398706995559851

490 min thresh: 0.0398706995559851

500 min thresh: 0.0371224077657652

510 min thresh: 0.0371224077657652

520 min thresh: 0.0371224077657652

530 min thresh: 0.0371224077657652

540 min thresh: 0.0371224077657652

550 min thresh: 0.0371224077657652

560 min thresh: 0.0371224077657652

570 min thresh: 0.0371224077657652

580 min thresh: 0.0371224077657652

590 min thresh: 0.0371224077657652

600 min thresh: 0.0342058141525678

610 min thresh: 0.0291302038592725

620 min thresh: 0.0291302038592725

630 min thresh: 0.0291302038592725

640 min thresh: 0.0291302038592725

650 min thresh: 0.0291302038592725

660 min thresh: 0.0291302038592725

670 min thresh: 0.0291302038592725

680 min thresh: 0.0291302038592725

690 min thresh: 0.0291302038592725

700 min thresh: 0.0291302038592725

710 min thresh: 0.0280157969397722

720 min thresh: 0.0280157969397722

730 min thresh: 0.0272688479001726

740 min thresh: 0.0272688479001726

750 min thresh: 0.0272688479001726

760 min thresh: 0.0265496483309847

770 min thresh: 0.0252182742481352

780 min thresh: 0.0252182742481352

790 min thresh: 0.0252182742481352

800 min thresh: 0.0245753273476944

810 min thresh: 0.0245753273476944

820 min thresh: 0.0239570228648457

830 min thresh: 0.0239570228648457

840 min thresh: 0.0228174257192961

850 min thresh: 0.0228174257192961

860 min thresh: 0.0226504966985968

870 min thresh: 0.0226504966985968

880 min thresh: 0.0226504966985968

890 min thresh: 0.0226504966985968

900 min thresh: 0.0226504966985968

910 min thresh: 0.0226504966985968

920 min thresh: 0.0196597443476527

930 min thresh: 0.0177201806502168

940 min thresh: 0.0175587553203274

950 min thresh: 0.0175587553203274

960 min thresh: 0.0175587553203274

970 min thresh: 0.0175587553203274

980 min thresh: 0.0175587553203274

990 min thresh: 0.0175587553203274

1000 min thresh: 0.0175587553203274

1010 min thresh: 0.0175587553203274

1020 min thresh: 0.0175587553203274

1030 min thresh: 0.0175587553203274

1040 min thresh: 0.017344774163237

1050 min thresh: 0.017344774163237

1060 min thresh: 0.016888746212474

1070 min thresh: 0.016888746212474

1080 min thresh: 0.0160440260247394

1090 min thresh: 0.0160440260247394

1100 min thresh: 0.0156361395273364

1110 min thresh: 0.0156361395273364

1120 min thresh: 0.0152434850127779

1130 min thresh: 0.0152434850127779

1140 min thresh: 0.0152434850127779

1150 min thresh: 0.0148661936554492

1160 min thresh: 0.0141726983141714

1170 min thresh: 0.0141726983141714

1180 min thresh: 0.0141726983141714

1190 min thresh: 0.0141726983141714

1200 min thresh: 0.0141726983141714

1210 min thresh: 0.0141726983141714

1220 min thresh: 0.011454392363353

1230 min thresh: 0.011454392363353

1240 min thresh: 0.0111096284551699

1250 min thresh: 0.0111096284551699

1260 min thresh: 0.0106057428744334

1270 min thresh: 0.0106057428744334

1280 min thresh: 0.0106057428744334

1290 min thresh: 0.0106057428744334

1296 error: 773.99927 time: 11.74 secs

plot(NN\_Model

****

**•Measure Model Performance on Development Sample**

> dev\_NN\_new$prob = NN\_Model$net.result[[1]]

> View(dev\_NN\_new)

> ## The distribution of the estimated probabilities

> quantile(dev\_NN\_new$prob, c(0,1,5,10,25,50,75,90,95,98,99,100)/100)

0% 1% 5% 10% 25% 50%

0.1260591 0.1260591 0.1260591 0.1260591 0.1260591 0.1260591

75% 90% 95% 98% 99% 100%

0.1260591 0.1260591 0.1260591 0.1260591 0.1427812 0.5660946

> hist(dev\_NN\_new$prob)

> hist(dev\_NN\_new$prob)

> ## 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

+ ))))))))))

+ }

>

> ## deciling

> dev\_NN\_new$deciles <- decile(dev\_NN\_new$prob)

> ## Ranking code

> ##install.packages("data.table")

> library(data.table)

data.table 1.12.0 Latest news: r-datatable.com

Warning message:

package ‘data.table’ was built under R version 3.5.3

> library(scales)

Warning message:

package ‘scales’ was built under R version 3.5.3

>

> tmp\_DT = data.table(dev\_NN\_new)

> 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);

> 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)

Gave only 1 decile hence as per model there is no difference in responders and non-responders .

| deciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 14008 | | 1776 | | 12232 | | 13.0% | | 1776 | | 12232 | | 100% | | 100% | | 0 |

> #scale the data

>

>

> loan\_NN1 =read.table("PL\_XSELL.csv",sep = ",",header = T)

> ind = sample(2,nrow(loan\_NN1),replace = TRUE ,prob = c(.7,.3))

> dev\_NN1 = loan\_NN1[ind==1,]

> hold\_out\_NN1 = loan\_NN1[ind==2,]

>

> View(dev\_NN1)

>

> occ.matrix <- model.matrix(~ GENDER - 1, data = dev\_NN1)

> dev\_NN1 <- data.frame(dev\_NN1, occ.matrix)

>

>

> occ.matrix <- model.matrix(~ OCCUPATION - 1, data = dev\_NN1)

> dev\_NN1 <- data.frame(dev\_NN1, occ.matrix)

>

> cc.matrix <- model.matrix(~ AGE\_BKT - 1, data = dev\_NN1)

> dev\_NN1 <- data.frame(dev\_NN1, occ.matrix)

>

> cc.matrix <- model.matrix(~ ACC\_TYPE - 1, data = dev\_NN1)

> dev\_NN1 <- data.frame(dev\_NN1, occ.matrix)

>

>

>

> dev\_NN1\_new=within(dev\_NN1,rm(TARGET,CUST\_ID,ACC\_OP\_DATE,GENDER,OCCUPATION,AGE\_BKT,ACC\_TYPE))

> str(dev\_NN1\_new)

'data.frame': 13953 obs. of 48 variables:

$ AGE : int 27 40 53 36 30 42 30 53 30 46 ...

$ BALANCE : num 3384 18217 71720 1671623 204459 ...

$ SCR : int 776 603 196 167 479 105 170 239 499 525 ...

$ HOLDING\_PERIOD : int 30 2 13 24 14 15 13 20 9 9 ...

$ LEN\_OF\_RLTN\_IN\_MNTH : int 146 61 107 185 177 88 111 142 74 198 ...

$ NO\_OF\_L\_CR\_TXNS : int 7 10 36 20 6 18 14 5 1 14 ...

$ NO\_OF\_L\_DR\_TXNS : int 3 5 14 1 6 14 8 4 1 6 ...

$ TOT\_NO\_OF\_L\_TXNS : int 10 15 50 21 12 32 22 9 2 20 ...

$ NO\_OF\_BR\_CSH\_WDL\_DR\_TXNS: int 0 1 4 1 0 6 3 1 0 2 ...

$ NO\_OF\_ATM\_DR\_TXNS : int 1 1 2 0 1 2 1 0 0 1 ...

$ NO\_OF\_NET\_DR\_TXNS : int 2 1 3 0 1 4 0 0 0 1 ...

$ NO\_OF\_MOB\_DR\_TXNS : int 0 0 1 0 0 1 0 0 0 0 ...

$ NO\_OF\_CHQ\_DR\_TXNS : int 0 2 4 0 4 1 4 3 1 2 ...

$ FLG\_HAS\_CC : int 0 0 0 0 0 1 0 1 0 1 ...

$ AMT\_ATM\_DR : int 13100 11200 26100 0 6200 35400 18000 0 0 5500 ...

$ AMT\_BR\_CSH\_WDL\_DR : int 0 561120 673590 808480 0 198430 869880 270160 0 546350 ...

$ AMT\_CHQ\_DR : int 0 49320 60780 0 10580 51490 32610 18290 77670 17390 ...

$ AMT\_NET\_DR : num 973557 997570 741506 0 770065 ...

$ AMT\_MOB\_DR : int 0 0 71388 0 0 170332 0 0 0 0 ...

$ AMT\_L\_DR : num 986657 1619210 1573364 808480 786845 ...

$ FLG\_HAS\_ANY\_CHGS : int 0 1 0 0 1 0 0 0 0 0 ...

$ AMT\_OTH\_BK\_ATM\_USG\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ AMT\_MIN\_BAL\_NMC\_CHGS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_IW\_CHQ\_BNC\_TXNS : int 0 0 0 0 0 0 0 0 0 0 ...

$ NO\_OF\_OW\_CHQ\_BNC\_TXNS : int 0 1 0 0 0 0 0 1 0 0 ...

$ AVG\_AMT\_PER\_ATM\_TXN : num 13100 11200 13050 0 6200 ...

$ AVG\_AMT\_PER\_CSH\_WDL\_TXN : num 0 561120 168398 808480 0 ...

$ AVG\_AMT\_PER\_CHQ\_TXN : num 0 24660 15195 0 2645 ...

$ AVG\_AMT\_PER\_NET\_TXN : num 486779 997570 247169 0 770065 ...

$ AVG\_AMT\_PER\_MOB\_TXN : num 0 0 71388 0 0 ...

$ FLG\_HAS\_NOMINEE : int 1 1 1 1 0 1 0 1 1 1 ...

$ FLG\_HAS\_OLD\_LOAN : int 1 1 0 0 1 1 0 1 0 1 ...

$ random : num 1.14e-05 1.20e-04 1.37e-04 1.74e-04 4.99e-04 ...

$ GENDERF : num 0 0 0 0 0 1 0 0 0 0 ...

$ GENDERM : num 1 1 1 1 1 0 1 1 1 1 ...

$ GENDERO : num 0 0 0 0 0 0 0 0 0 0 ...

$ OCCUPATIONPROF : num 0 0 0 1 1 0 1 0 0 0 ...

$ OCCUPATIONSAL : num 0 0 1 0 0 0 0 0 0 1 ...

$ OCCUPATIONSELF.EMP : num 1 1 0 0 0 1 0 0 1 0 ...

$ OCCUPATIONSENP : num 0 0 0 0 0 0 0 1 0 0 ...

$ OCCUPATIONPROF.1 : num 0 0 0 1 1 0 1 0 0 0 ...

$ OCCUPATIONSAL.1 : num 0 0 1 0 0 0 0 0 0 1 ...

$ OCCUPATIONSELF.EMP.1 : num 1 1 0 0 0 1 0 0 1 0 ...

$ OCCUPATIONSENP.1 : num 0 0 0 0 0 0 0 1 0 0 ...

$ OCCUPATIONPROF.2 : num 0 0 0 1 1 0 1 0 0 0 ...

$ OCCUPATIONSAL.2 : num 0 0 1 0 0 0 0 0 0 1 ...

$ OCCUPATIONSELF.EMP.2 : num 1 1 0 0 0 1 0 0 1 0 ...

$ OCCUPATIONSENP.2 : num 0 0 0 0 0 0 0 1 0 0 ...

> View(dev\_NN1\_new)

>

>

> allvars = colnames(dev\_NN1\_new)

> predictorVars = allvars[!allvars%in%"TARGET"]

> predictorVars = paste(predictorVars,collapse = "+")

> form1=as.formula(paste("TARGET~",predictorVars,collapse = "+"))

>

> form1

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 + OCCUPATIONPROF.1 + OCCUPATIONSAL.1 + OCCUPATIONSELF.EMP.1 +

OCCUPATIONSENP.1 + OCCUPATIONPROF.2 + OCCUPATIONSAL.2 + OCCUPATIONSELF.EMP.2 +

OCCUPATIONSENP.2

>

> dev\_NN\_Scaled = scale(dev\_NN1\_new)

>

> dev\_NN\_Scaled <- cbind(dev\_NN1[2], dev\_NN\_Scaled)

> View(dev\_NN\_Scaled)

Scaled NN\_Model\_ =neuralnet(formula=form1,

+ data=dev\_NN\_Scaled,

+ hidden = 2,

+ err.fct = "sse",

+ linear.output = FALSE,

+ lifesign = "full",

+ lifesign.step=1,

+ threshold = .1,

+ stepmax = 2000)

hidden: 2 thresh: 0.1 rep: 1/1 steps: 1 min thresh: 1231.24249764033

2 min thresh: 1199.18790638122

3 min thresh: 1112.29562171063

4 min thresh: 977.891993924895

5 min thresh: 850.188685164604

6 min thresh: 712.873635858722

7 min thresh: 561.64656502241

8 min thresh: 414.724466062172

9 min thresh: 272.316331185081

10 min thresh: 160.845680182675

11 min thresh: 82.898700504022

12 min thresh: 46.5556401574375

13 min thresh: 9.08631895665958

14 min thresh: 6.39082204551175

15 min thresh: 6.39082204551175

16 min thresh: 6.39082204551175

17 min thresh: 6.39082204551175

18 min thresh: 6.39082204551175

19 min thresh: 6.39082204551175

20 min thresh: 4.02526180604452

21 min thresh: 4.02526180604452

22 min thresh: 4.02526180604452

23 min thresh: 4.02526180604452

24 min thresh: 4.02526180604452

25 min thresh: 4.02526180604452

26 min thresh: 4.02526180604452

27 min thresh: 4.02526180604452

28 min thresh: 4.02526180604452

29 min thresh: 4.02526180604452

30 min thresh: 4.02526180604452

31 min thresh: 4.02526180604452

32 min thresh: 4.02526180604452

33 min thresh: 3.90369981699195

34 min thresh: 3.90369981699195

35 min thresh: 3.90369981699195

36 min thresh: 3.90369981699195

37 min thresh: 3.55469538059962

38 min thresh: 3.55469538059962

39 min thresh: 3.55469538059962

40 min thresh: 3.55469538059962

41 min thresh: 3.55469538059962

42 min thresh: 3.55469538059962

43 min thresh: 3.55469538059962

44 min thresh: 3.55469538059962

45 min thresh: 3.55469538059962

46 min thresh: 3.55469538059962

47 min thresh: 3.55469538059962

48 min thresh: 3.33564836932702

49 min thresh: 3.33564836932702

50 min thresh: 3.33564836932702

51 min thresh: 3.33564836932702

52 min thresh: 2.34622554392298

53 min thresh: 2.34622554392298

54 min thresh: 2.34622554392298

55 min thresh: 2.34622554392298

56 min thresh: 2.34622554392298

57 min thresh: 2.34622554392298

58 min thresh: 2.34622554392298

59 min thresh: 2.34622554392298

60 min thresh: 2.34622554392298

61 min thresh: 2.34622554392298

62 min thresh: 2.34622554392298

63 min thresh: 2.34622554392298

64 min thresh: 2.20293348797954

65 min thresh: 2.03752764303205

66 min thresh: 1.88380697540929

67 min thresh: 1.88380697540929

68 min thresh: 1.88380697540929

69 min thresh: 1.88380697540929

70 min thresh: 1.03554881171336

71 min thresh: 1.03554881171336

72 min thresh: 1.03554881171336

73 min thresh: 1.03554881171336

74 min thresh: 1.03554881171336

279 min thresh: 0.405000778489964

280 min thresh: 0.405000778489964

281 min thresh: 0.405000778489964

282 min thresh: 0.405000778489964

283 min thresh: 0.405000778489964

284 min thresh: 0.405000778489964

285 min thresh: 0.405000778489964

286 min thresh: 0.405000778489964

287 min thresh: 0.405000778489964

288 min thresh: 0.405000778489964

289 min thresh: 0.405000778489964

290 min thresh: 0.405000778489964

291 min thresh: 0.405000778489964

292 min thresh: 0.405000778489964

293 min thresh: 0.405000778489964

294 min thresh: 0.405000778489964

295 min thresh: 0.405000778489964

296 min thresh: 0.405000778489964

297 min thresh: 0.405000778489964

298 min thresh: 0.405000778489964

299 min thresh: 0.405000778489964

300 min thresh: 0.405000778489964

301 min thresh: 0.405000778489964

302 min thresh: 0.405000778489964

303 min thresh: 0.405000778489964

304 min thresh: 0.405000778489964

305 min thresh: 0.405000778489964

306 min thresh: 0.405000778489964

307 min thresh: 0.405000778489964

308 min thresh: 0.405000778489964

309 min thresh: 0.405000778489964

310 min thresh: 0.405000778489964

311 min thresh: 0.405000778489964

312 min thresh: 0.405000778489964

991 min thresh: 0.107011337830506

992 min thresh: 0.107011337830506

993 min thresh: 0.107011337830506

994 min thresh: 0.107011337830506

995 min thresh: 0.107011337830506

996 min thresh: 0.107011337830506

997 error: 702.54288 time: 7.66 secs

> plot(NN\_Model\_Scaled)

****

> View(dev\_NN1\_new)

> ## The distribution of the estimated probabilities

> quantile(dev\_NN1\_new$prob, c(0,1,5,10,25,50,75,90,95,98,99,100)/100)

0% 1% 5% 10% 25% 50%

0.02854043 0.02854729 0.02864516 0.02901703 0.03688872 0.10624121

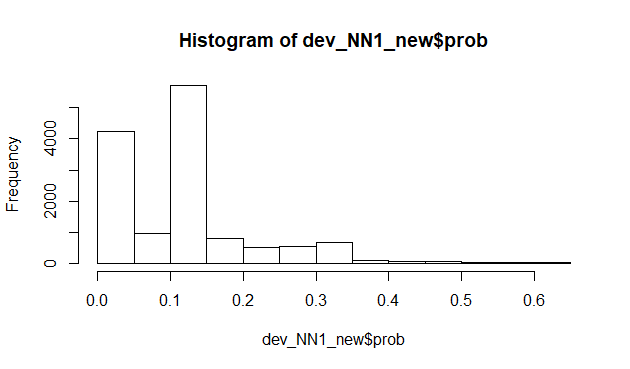
75% 90% 95% 98% 99% 100%

0.13526068 0.27810999 0.31668605 0.42758264 0.51287631 0.64953312

> hist(dev\_NN1\_new)

Error in hist.default(dev\_NN1\_new) : 'x' must be numeric

> hist(dev\_NN1\_new$prob)



## deciling

> dev\_NN\_Scaled$deciles = decile(dev\_NN\_Scaled$prob)

> tmp\_DT = data.table(dev\_NN\_Scaled)

> 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);

> 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

| deciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 1396 | | 529 | | 867 | | 38.0% | | 529 | | 867 | | 30.0% | | 7.0% | | 0.23 |
| 2 | 9 | | 1395 | | 258 | | 1137 | | 18.0% | | 787 | | 2004 | | 45.0% | | 16.0% | | 0.29 |
| 3 | 8 | | 1395 | | 209 | | 1186 | | 15.0% | | 996 | | 3190 | | 57.0% | | 26.0% | | 0.31 |
| 4 | 7 | | 1395 | | 143 | | 1252 | | 10.0% | | 1139 | | 4442 | | 65.0% | | 36.0% | | 0.29 |
| 5 | 6 | | 1396 | | 167 | | 1229 | | 12.0% | | 1306 | | 5671 | | 74.0% | | 47.0% | | 0.27 |
| 6 | 5 | | 1395 | | 102 | | 1293 | | 7.0% | | 1408 | | 6964 | | 80.0% | | 57.0% | | 0.23 |
| 7 | 4 | | 1395 | | 130 | | 1265 | | 9.0% | | 1538 | | 8229 | | 87.0% | | 67.0% | | 0.20 |
| 8 | 3 | | 1395 | | 95 | | 1300 | | 7.0% | | 1633 | | 9529 | | 93.0% | | 78.0% | | 0.15 |
| 9 | 2 | | 1395 | | 79 | | 1316 | | 6.0% | | 1712 | | 10845 | | 97.0% | | 89.0% | | 0.08 |
| 10 | 1 | | 1396 | | 49 | | 1347 | | 4.0% | | 1761 | | 12192 | | 100.0% | | 100.0% | | 0.0 |

Top decile has response rate of 38%

Ks is .31 – which is not that good as at-least it should be around .4 for marketing hence there are chances of improvement .

Confusion matrix and classification error

dev\_NN\_Scaled$Class = ifelse(dev\_NN\_Scaled$prob>0.40,1,0)

> with( dev\_NN\_Scaled, table(TARGET, as.factor(Class) ))

TARGET 0 1

0 12046 146

1 1575 186

> (146+1575)/13853

[1] 0.124233

**Classification error is 12%**

pred <- prediction(dev\_NN\_Scaled$prob, dev\_NN\_Scaled$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)

>

> library(ineq)

Warning message:

package ‘ineq’ was built under R version 3.5.2

> gini = ineq(dev\_NN\_Scaled$prob, type="Gini")

>

>

> auc

[1] 0.7075124

> KS

[1] 0.3079595

> gini

[1] 0.408166

**•Test Model Performance on Hold Out Sample**

occ.matrix <- model.matrix(~ GENDER - 1, data = hold\_out\_NN)

> hold\_out\_NN <- data.frame(hold\_out\_NN, occ.matrix)

> occ.matrix <- model.matrix(~ OCCUPATION - 1, data = hold\_out\_NN)

> hold\_out\_NN <- data.frame(hold\_out\_NN, occ.matrix)

>

> cc.matrix <- model.matrix(~ AGE\_BKT - 1, data = hold\_out\_NN)

> hold\_out\_NN <- data.frame(hold\_out\_NN, occ.matrix)

>

> cc.matrix <- model.matrix(~ ACC\_TYPE - 1, data = hold\_out\_NN)

> hold\_out\_NN <- data.frame(hold\_out\_NN, occ.matrix)

> View(hold\_out\_NN)

> hold\_out\_NN\_new=within(hold\_out\_NN,rm(TARGET,CUST\_ID,ACC\_OP\_DATE,GENDER,OCCUPATION,AGE\_BKT,ACC\_TYPE))

> allvars = colnames(hold\_out\_NN\_new)

> predictorVars = allvars[!allvars%in%"TARGET"]

> predictorVars = paste(predictorVars,collapse = "+")

> form1=as.formula(paste("TARGET~",predictorVars,collapse = "+"))

> allvars = colnames(hold\_out\_NN\_new)

> predictorVars = allvars[!allvars%in%"TARGET"]

> predictorVars = paste(predictorVars,collapse = "+")

> form1=as.formula(paste("TARGET~",predictorVars,collapse = "+"))

> hold\_out\_Scaled = scale(hold\_out\_NN\_new)

> hold\_out\_Scaled <- cbind(hold\_out\_NN[2], hold\_out\_Scaled)

> View(dev\_NN\_Scaled)

> View(hold\_out\_Scaled)

> NN\_Model\_Scaled\_test =neuralnet(formula=form1,

+ data=hold\_out\_Scaled,

+ hidden = 2,

+ err.fct = "sse",

+ linear.output = FALSE,

+ lifesign = "full",

+ lifesign.step=1,

+ threshold = .1,

+ stepmax = 2000)

286757629537

623 min thresh: 0.100286757629537

624 min thresh: 0.100286757629537

625 min thresh: 0.100286757629537

626 min thresh: 0.100286757629537

627 error: 277.52653 time: 2.42 secs

****

> hold\_out\_NN\_new$Predict.score = compute.output$net.result

> ## deciling

> hold\_out\_NN\_new$deciles = decile(hold\_out\_NN\_new$Predict.score)

> tmp\_DT = data.table(hold\_out\_NN)

> h\_rank <- tmp\_DT[, list(

+ cnt = length(TARGET),

+ cnt\_resp = sum(TARGET),

+ cnt\_non\_resp = sum(TARGET == 0)) ,

+ by=deciles][order(-deciles)]

> tmp\_DT = data.table(hold\_out\_NN\_new)

> 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)

| deciles | | cnt | | cnt\_resp | | cnt\_non\_resp | | rrate | | cum\_resp | | cum\_non\_resp | | cum\_rel\_resp | | cum\_rel\_non\_resp | | ks | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  |
| 1 | 10 | | 601 | | 210 | | 391 | | 35.0% | | 210 | | 391 | | 29.0% | | 7.0% | | 0.22 |
| 2 | 9 | | 601 | | 99 | | 502 | | 16.0% | | 309 | | 893 | | 43.0% | | 17.0% | | 0.26 |
| 3 | 8 | | 601 | | 97 | | 504 | | 16.0% | | 406 | | 1397 | | 57.0% | | 26.0% | | 0.31 |
| 4 | 7 | | 601 | | 66 | | 535 | | 11.0% | | 472 | | 1932 | | 66.0% | | 37.0% | | 0.29 |
| 5 | 6 | | 601 | | 60 | | 541 | | 10.0% | | 532 | | 2473 | | 74.0% | | 47.0% | | 0.27 |
| 6 | 5 | | 601 | | 34 | | 567 | | 6.0% | | 566 | | 3040 | | 79.0% | | 57.0% | | 0.22 |
| 7 | 4 | | 601 | | 62 | | 539 | | 10.0% | | 628 | | 3579 | | 88.0% | | 68.0% | | 0.20 |
| 8 | 3 | | 601 | | 36 | | 565 | | 6.0% | | 664 | | 4144 | | 93.0% | | 78.0% | | 0.15 |
| 9 | 2 | | 601 | | 37 | | 564 | | 6.0% | | 701 | | 4708 | | 98.0% | | 89.0% | | 0.09 |
| 10 | 1 | | 601 | | 16 | | 585 | | 3.0% | | 717 | | 5293 | | 100.0% | | 100.0% | | 0.0 |

**•Ensure the model is not an overfit model**

LMost of the statistics of train and test sample are similar , ks values is exactly same in both the data sample .Hence we can say that model is not overfitted

**Part 4 | Model Comparison**

**•Compare the 3 Model’s Performance**

**–CART :** KS is 32.48% and 1st decile cover 34% of population. Classification error is around 11%

**–Random Forest** : KS is too high and model is overfitted . Classification error is around 5%

**–Neural Network :** Ks is 31% and 1st decile cover 38% of population . Classification error is around 12%

**•Ensemble Model – Create Ensemble Model based on the output of the above 3 models •Compare the Ensemble Model performance with individual models**