**PROJECT – 2**

Typical information that is available about customers’ concerns demographics, behavioral data, and revenue information. At the time of renewing contracts, some customers do and some do not: they churn. It would be extremely useful to know in advance which customers are at risk of churning, as to prevent it ‒ especially in the case of high revenue customers.

This is a typical problem to be constructed in a logistic regression format, wherein the probability of a customer renewing or not renewing is to be estimated. The coding of the data seems to be coded for renewing and not renewing. The code for renewing is 0 and that of not renewing(that is churn) is 1.

> table(pchurn$Churn)

0 1

2850 483

Primary observation of the data reveals that there are more cases of no churn (0s) than churns (1), which is quite normal with such predicting of negative events. There can be two approaches to attempt the logistic regression here. One can be to draw a training data set which has equal instances of both 0s and 1s, so that the weight of one even over the other is avoided and the model fitting can be fairer.

The other approach is to use the suggested approach of creating a training data set with 80% of the data points and leaving the rest 20% for the testing data. In this case the assumption is that the proportions of 1s and 0s would be maintained in both the 80% and 20% samples.

There seem to be no missing data in the given dataset

> sum(is.na(pchurn))

[1] 0

Examination of the raw data given in the dataset reveals the following structure and the variable type information.

> str(Churn\_1\_)

Classes ‘tbl\_df’, ‘tbl’ and 'data.frame': 3333 obs. of 21 variables:

$ Account Length: num 128 107 137 84 75 118 121 147 117 141 ...

$ VMail Message : num 25 26 0 0 0 0 24 0 0 37 ...

$ Day Mins : num 265 162 243 299 167 ...

$ Eve Mins : num 197.4 195.5 121.2 61.9 148.3 ...

$ Night Mins : num 245 254 163 197 187 ...

$ Intl Mins : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

$ CustServ Calls: num 1 1 0 2 3 0 3 0 1 0 ...

$ Churn : num 0 0 0 0 0 0 0 0 0 0 ...

$ Int'l Plan : num 0 0 0 1 1 1 0 1 0 1 ...

$ VMail Plan : num 1 1 0 0 0 0 1 0 0 1 ...

$ Day Calls : num 110 123 114 71 113 98 88 79 97 84 ...

$ Day Charge : num 45.1 27.5 41.4 50.9 28.3 ...

$ Eve Calls : num 99 103 110 88 122 101 108 94 80 111 ...

$ Eve Charge : num 16.78 16.62 10.3 5.26 12.61 ...

$ Night Calls : num 91 103 104 89 121 118 118 96 90 97 ...

$ Night Charge : num 11.01 11.45 7.32 8.86 8.41 ...

$ Intl Calls : num 3 3 5 7 3 6 7 6 4 5 ...

$ Intl Charge : num 2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...

$ State : chr "KS" "OH" "NJ" "OH" ...

$ Area Code : num 415 415 415 408 415 510 510 415 408 415 ...

$ Phone : chr "382-4657" "371-7191" "358-1921" "375-9999" ...

Since Logistic Regression works on a binominal two state variable, and in the categorical type, than a numeric type, the target variable “Churn” is converted into a factor from a numeric. We have tested estimation of logistic regression without such conversion, and the programme did not estimate and the algorithm never converged at various levels of iterations.

Further variables which are characters are also converted into factors, and those where the data points represent dummy variables, like the presence or absence of a feature, were also converted into factor, since treating them as numerical would not be logical.

pchurn$Churn <- as.factor(pchurn$Churn)

pchurn$State <- as.factor(pchurn$State)

pchurn$Phone <- as.factor(pchurn$Phone)

pchurn$`Area Code`<- as.factor(pchurn$`Area Code`)

pchurn$`VMail Plan`<- as.factor(pchurn$`VMail Plan`)

pchurn$`Intl Plan`<- as.factor(pchurn$`Intl Plan`)

While observing the nature of variables and their ability to predict based on our commonly thinking logic, the variable phone number has been excluded from the estimation process, since phone number is unique for each of the client.

> pchurn <- pchurn[,-c(21)]

The approach of creating a 50-50 balanced sample was implemented but did not give any result better than the 80-20 on the test data, since the training data set had two few observations when trying to matching the number of ones and 0s. So the entire project is attempted with 80-20 sampling.

smp\_size <- floor(0.8 \* nrow(pchurn))

train\_ind <- sample(seq\_len(nrow(pchurn)), size = smp\_size)

churntrain <- pchurn[train\_ind, ]

churntrain <- pchurn[train\_ind, ]

churntest <- pchurn[-train\_ind, ]

Further the training model is estimated in the ordinary model and also in the step wise process with backward selection method.

fit1 <- glm(Churn~.,data = churntrain, family = binomial(link = 'logit'),control = list (maxit=50))

step\_fit1 <- stepAIC(fit1,method='backward')

The summaries of each of them are presented below.

summary(fit1)

summary(step\_fit1)

> summary(fit1)

> summary(fit1)

Call:

glm(formula = Churn ~ ., family = binomial(link = "logit"), data = churntrain,

control = list(maxit = 50))

Deviance Residuals:

Min 1Q Median 3Q Max

-2.1752 -0.4937 -0.3044 -0.1570 3.0290

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -10.637992 1.333934 -7.975 1.53e-15 \*\*\*

`Account Length` 0.001089 0.001624 0.671 0.50252

`VMail Message` 0.041342 0.020622 2.005 0.04499 \*

`Day Mins` -0.514276 3.858038 -0.133 0.89396

`Eve Mins` 1.900539 1.927695 0.986 0.32418

`Night Mins` -0.175437 1.021404 -0.172 0.86363

`Intl Mins` -4.816912 6.201407 -0.777 0.43731

`CustServ Calls` 0.512333 0.046042 11.128 < 2e-16 \*\*\*

IntlPlan1 2.405929 0.173916 13.834 < 2e-16 \*\*\*

`VMail Plan`1 -2.129032 0.661026 -3.221 0.00128 \*\*

`Day Calls` 0.002953 0.003246 0.910 0.36300

`Day Charge` 3.101984 22.694538 0.137 0.89128

`Eve Calls` 0.005004 0.003304 1.515 0.12983

`Eve Charge` -22.261306 22.678418 -0.982 0.32629

`Night Calls` 0.001037 0.003313 0.313 0.75425

`Night Charge` 3.978975 22.696533 0.175 0.86083

`Intl Calls` -0.066890 0.028132 -2.378 0.01742 \*

`Intl Charge` 18.127198 22.966905 0.789 0.42995

StateAL 0.446038 1.197215 0.373 0.70947

StateAR 1.616385 1.125119 1.437 0.15082

StateAZ 0.400545 1.229647 0.326 0.74462

StateCA 2.219920 1.165796 1.904 0.05688 .

StateCO 1.394813 1.134980 1.229 0.21910

StateCT 1.292593 1.123748 1.150 0.25004

StateDC 1.154440 1.183623 0.975 0.32939

StateDE 1.488692 1.122247 1.327 0.18466

StateFL 1.331437 1.143654 1.164 0.24434

StateGA 0.930152 1.164866 0.799 0.42458

StateHI 0.028714 1.291415 0.022 0.98226

StateIA -0.124390 1.473817 -0.084 0.93274

StateID 1.363085 1.141362 1.194 0.23238

StateIL 0.245674 1.186079 0.207 0.83591

StateIN 0.939946 1.136138 0.827 0.40806

StateKS 1.702774 1.121314 1.519 0.12888

StateKY 1.410586 1.141837 1.235 0.21669

StateLA 1.445078 1.178845 1.226 0.22026

StateMA 1.767330 1.115271 1.585 0.11304

StateMD 0.985829 1.129847 0.873 0.38292

StateME 1.595178 1.124551 1.419 0.15604

StateMI 2.330063 1.100633 2.117 0.03426 \*

StateMN 1.773945 1.099216 1.614 0.10657

StateMO 1.227267 1.152785 1.065 0.28705

StateMS 2.177051 1.111244 1.959 0.05010 .

StateMT 2.427435 1.097603 2.212 0.02700 \*

StateNC 1.116763 1.136444 0.983 0.32576

StateND 0.634509 1.154978 0.549 0.58275

StateNE 0.894908 1.178670 0.759 0.44770

StateNH 1.481558 1.154348 1.283 0.19933

StateNJ 2.061961 1.097628 1.879 0.06030 .

StateNM 0.847872 1.160722 0.730 0.46510

StateNV 1.793789 1.110745 1.615 0.10632

StateNY 1.575861 1.105773 1.425 0.15412

StateOH 1.104356 1.124224 0.982 0.32594

StateOK 1.576622 1.119743 1.408 0.15912

StateOR 1.453913 1.115354 1.304 0.19239

StatePA 1.318305 1.162798 1.134 0.25691

StateRI 0.428543 1.189329 0.360 0.71861

StateSC 2.189639 1.127466 1.942 0.05213 .

StateSD 1.547295 1.122915 1.378 0.16823

StateTN 0.354486 1.233796 0.287 0.77387

StateTX 1.897881 1.107449 1.714 0.08658 .

StateUT 1.667361 1.118436 1.491 0.13601

StateVA -0.192135 1.232503 -0.156 0.87612

StateVT 0.958567 1.142152 0.839 0.40132

StateWA 1.897597 1.108268 1.712 0.08686 .

StateWI 1.071777 1.158856 0.925 0.35504

StateWV 0.836724 1.132888 0.739 0.46016

StateWY 0.821463 1.142816 0.719 0.47226

`Area Code`415 -0.108192 0.158872 -0.681 0.49587

`Area Code`510 -0.156209 0.183401 -0.852 0.39436

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Null deviance: 2240.3 on 2665 degrees of freedom

Residual deviance: 1640.8 on 2596 degrees of freedom

AIC: 1780.8

Number of Fisher Scoring iterations: 6

> summary(step\_fit1)

Call:

glm(formula = Churn ~ `VMail Message` + `Eve Mins` + `CustServ Calls` +

IntlPlan + `VMail Plan` + `Day Charge` + `Eve Calls` + `Night Charge` +

`Intl Calls` + `Intl Charge`, family = binomial(link = "logit"),

data = churntrain, control = list(maxit = 50))

Deviance Residuals:

Min 1Q Median 3Q Max

-2.1684 -0.5094 -0.3390 -0.1988 3.2401

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -8.580482 0.654653 -13.107 < 2e-16 \*\*\*

`VMail Message` 0.037210 0.019649 1.894 0.058269 .

`Eve Mins` 0.007612 0.001300 5.857 4.72e-09 \*\*\*

`CustServ Calls` 0.495483 0.043761 11.322 < 2e-16 \*\*\*

IntlPlan1 2.232850 0.161537 13.823 < 2e-16 \*\*\*

`VMail Plan`1 -1.983117 0.630005 -3.148 0.001645 \*\*

`Day Charge` 0.076962 0.007100 10.840 < 2e-16 \*\*\*

`Eve Calls` 0.004505 0.003144 1.433 0.151961

`Night Charge` 0.077898 0.027614 2.821 0.004788 \*\*

`Intl Calls` -0.075247 0.027161 -2.770 0.005599 \*\*

`Intl Charge` 0.300728 0.084524 3.558 0.000374 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Null deviance: 2240.3 on 2665 degrees of freedom

Residual deviance: 1728.1 on 2655 degrees of freedom

AIC: 1750.1

Number of Fisher Scoring iterations: 6

The summary of both the estimations shown above show that step wise estimation is relatively more efficient and is able to converge better to a lesser residual deviance. There seems to be a strong Intercept, which is not explained by any of the variables given in the dataset. Of the variables given International Plan, International Charges, Customer Service Calls, are strong positive predictors, and International Calls, Vmail plan seems to be a strong negative predictor of the log odds ratio of the Churn. When we observe the difference between the null.deviance and the residual deviance using the chisquare test, all of the results are significant at levels below 0.1%.

> with(fit1, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 1.373453e-85

> with(step\_fit1, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 1.061981e-103

Analysis of Variance also show the same results

> anova(fit1,test='Chisq')

Df Deviance Resid. Df Resid. Dev Pr(>Chi)

NULL 2665 2240.3

`Account Length` 1 1.199 2664 2239.1 0.273572

`VMail Message` 1 20.777 2663 2218.3 5.159e-06 \*\*\*

`Day Mins` 1 123.764 2662 2094.6 < 2.2e-16 \*\*\*

`Eve Mins` 1 30.044 2661 2064.5 4.223e-08 \*\*\*

`Night Mins` 1 4.220 2660 2060.3 0.039959 \*

`Intl Mins` 1 16.019 2659 2044.3 6.270e-05 \*\*\*

`CustServ Calls` 1 109.975 2658 1934.3 < 2.2e-16 \*\*\*

IntlPlan 1 186.170 2657 1748.1 < 2.2e-16 \*\*\*

`VMail Plan` 1 10.594 2656 1737.5 0.001134 \*\*

`Day Calls` 1 0.402 2655 1737.2 0.526308

`Day Charge` 1 0.001 2654 1737.1 0.975055

`Eve Calls` 1 1.819 2653 1735.3 0.177489

`Eve Charge` 1 0.579 2652 1734.8 0.446764

`Night Calls` 1 0.412 2651 1734.3 0.520850

`Night Charge` 1 0.000 2650 1734.3 0.991350

`Intl Calls` 1 8.182 2649 1726.2 0.004231 \*\*

`Intl Charge` 1 0.762 2648 1725.4 0.382557

State 50 83.818 2598 1641.6 0.001928 \*\*

`Area Code` 2 0.774 2596 1640.8 0.679225

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(step\_fit1, test = 'Chisq')

Df Deviance Resid. Df Resid. Dev Pr(>Chi)

NULL 2665 2240.3

`VMail Message` 1 21.022 2664 2219.3 4.540e-06 \*\*\*

`Eve Mins` 1 30.404 2663 2188.9 3.508e-08 \*\*\*

`CustServ Calls` 1 99.098 2662 2089.8 < 2.2e-16 \*\*\*

IntlPlan 1 194.737 2661 1895.0 < 2.2e-16 \*\*\*

`VMail Plan` 1 10.390 2660 1884.7 0.001267 \*\*

`Day Charge` 1 126.581 2659 1758.1 < 2.2e-16 \*\*\*

`Eve Calls` 1 2.051 2658 1756.0 0.152128

`Night Charge` 1 7.909 2657 1748.1 0.004918 \*\*

`Intl Calls` 1 7.151 2656 1741.0 0.007491 \*\*

`Intl Charge` 1 12.909 2655 1728.1 0.000327 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Further the VIF analysis of the step wise estimates for training shows the following. There seems be multicollinearity between Vmail Plan and Vmail Message. VIFs greater than 10, is not usually advisable in logistic estimation. They essentially convey that either of them could be used as a predictor since they seem to have a linear relationship. In such cases the ability to marginally predict the target variable decreases. Hence the VIF should be as low as possible, so that each of the predictor variables can be recognised as unique features with the ability to add to the log of odds ratio of the outcome.

> vif(fit1)

GVIF Df GVIF^(1/(2\*Df))

`Account Length` 1.035552e+00 1 1.017621

`VMail Message` 1.635785e+01 1 4.044484

`Day Mins` 1.042641e+07 1 3228.995292

`Eve Mins` 2.144777e+06 1 1464.505833

`Night Mins` 6.395655e+05 1 799.728379

`Intl Mins` 7.083465e+04 1 266.147798

`CustServ Calls` 1.139769e+00 1 1.067600

IntlPlan 1.189838e+00 1 1.090797

`VMail Plan` 1.638972e+01 1 4.048422

`Day Calls` 1.041045e+00 1 1.020316

`Day Charge` 1.042645e+07 1 3229.002120

`Eve Calls` 1.041942e+00 1 1.020756

`Eve Charge` 2.144759e+06 1 1464.499563

`Night Calls` 1.048207e+00 1 1.023820

`Night Charge` 6.395594e+05 1 799.724592

`Intl Calls` 1.041928e+00 1 1.020749

`Intl Charge` 7.083380e+04 1 266.146193

State 1.875389e+00 50 1.006308

`Area Code` 1.082760e+00 2 1.020077

> vif(step\_fit1)

`VMail Message` `Eve Mins` `CustServ Calls` IntlPlan `VMail Plan`

15.530018 1.024770 1.081355 1.076219 15.562698

`Day Charge` `Eve Calls` `Night Charge` `Intl Calls` `Intl Charge`

1.042390 1.003917 1.012379 1.009835 1.014915

Then we move on to find out the Area Under the Curve and the ROC curve plotting to find out the strength of a model to predict the target outcome.

> tg1 – Based on Normal Logistic Estimation on training data

Area under the curve: 0.8483

> tg2 - Based on Step Wise Estimation on training data

Area under the curve: 0.8278

> g1 - Based on Normal Logistic Estimation on testing data

Area under the curve: 0.7766

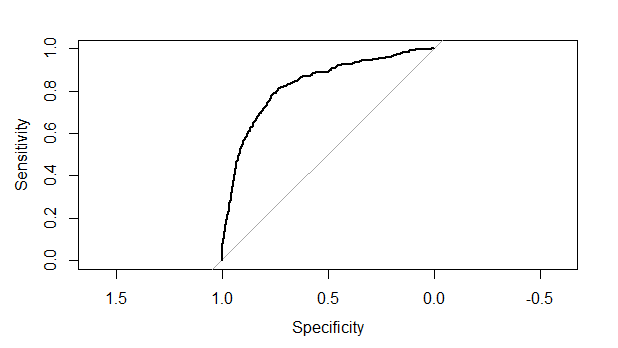
> g2 - Based on Step Wise Estimation on testing data

Area under the curve: 0.8024

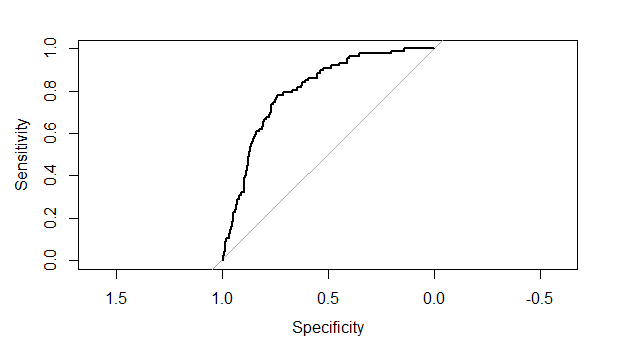
The results show that the model performs better on the training data, but weakens on test data. But it is still in the acceptable range of 80% and marginally above it.

The ROC for both the Step Wise estimations on Training Model and Test Data is as follows

ROC based on Training Model



ROC based on Test Data



We move on to find out the confusion matrix scores using the various thresholds. Since the Step-Wise estimation has been showing better AIC and deviance, the threshold analysis is undertaken initially for the training data on the step-wise estimations predictions and later on the test data.

> churntrain$Pred\_Churn1 <- ifelse(trainpred\_step1 <0.5,0,1)

> table(churntrain$Churn)

0 1

2270 396

> table(churntrain$Pred\_Churn1)

0 1

2508 158

The 50% threshold can be seen to have classified more 0s than 1s. It is predicting more non-churns than churns. This is a problem with the unbalanced datasets. The following is the confusion matrix related to the same.

> churntrain$Pred\_Churn1 <- ifelse(trainpred\_step1 <0.45,0,1)

> table(churntrain$Pred\_Churn1)

0 1

2465 201

> churntrain$Pred\_Churn1 <- ifelse(trainpred\_step1 <0.40,0,1)

> table(churntrain$Pred\_Churn1)

0 1

2420 246

> churntrain$Pred\_Churn1 <- ifelse(trainpred\_step1 <0.35,0,1)

> table(churntrain$Pred\_Churn1)

0 1

2366 300

> churntrain$Pred\_Churn1 <- ifelse(trainpred\_step1 <0.30,0,1)

> table(churntrain$Pred\_Churn1)

0 1

2273 393

> table(churntrain$Churn)

0 1

2270 396

> confusionMatrix(churntrain$Churn, as.factor(churntrain$Pred\_Churn1))#50%

Reference

Prediction 0 1

0 2201 69

1 307 89

Accuracy : 0.859

95% CI : (0.8452, 0.872)

No Information Rate : 0.9407

P-Value [Acc > NIR] : 1

Kappa : 0.2585

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

Sensitivity : 0.8776

Specificity : 0.5633

Pos Pred Value : 0.9696

Neg Pred Value : 0.2247

Prevalence : 0.9407

Detection Rate : 0.8256

Detection Prevalence : 0.8515

Balanced Accuracy : 0.7204

> confusionMatrix(churntrain$Churn, as.factor(churntrain$Pred\_Churn1))#30%

Reference

Prediction 0 1

0 2078 192

1 195 201

Accuracy : 0.8548

95% CI : (0.8409, 0.868)

No Information Rate : 0.8526

P-Value [Acc > NIR] : 0.3841

Kappa : 0.4243

Mcnemar's Test P-Value : 0.9190

Sensitivity : 0.9142

Specificity : 0.5115

Pos Pred Value : 0.9154

Neg Pred Value : 0.5076

Prevalence : 0.8526

Detection Rate : 0.7794

Detection Prevalence : 0.8515

Balanced Accuracy : 0.7128

Applying on the Test Data we start with the traditional 50% threshold limit, then move on to the optimum threshold achieved through iterations on training data. Then further we find out that 28% is quite well suited for the testing data, which is just 2 percentage points down than the training model’s threshold , which almost nearly predicts the actual 0s and 1s, as shown in the following table of actual churns in the churntest data set.

> table(churntest$Churn)

0 1

580 87

> churntest$Pred\_Churn1 <- ifelse(pred\_step1 <0.50,0,1)

> table(churntest$Pred\_Churn1)

0 1

631 36

> churntest$Pred\_Churn1 <- ifelse(pred\_step1 < 0.30,0,1)

> table(churntest$Pred\_Churn1)

0 1

593 74

> churntest$Pred\_Churn1 <- ifelse(pred\_step1 < 0.28,0,1)

> table(churntest$Pred\_Churn1)

0 1

579 88

> confusionMatrix(churntest$Churn, as.factor(churntest$Pred\_Churn1))#50%

Reference

Prediction 0 1

0 557 23

1 74 13

Accuracy : 0.8546

95% CI : (0.8255, 0.8805)

No Information Rate : 0.946

P-Value [Acc > NIR] : 1

Kappa : 0.1462

Mcnemar's Test P-Value : 3.84e-07

Sensitivity : 0.8827

Specificity : 0.3611

Pos Pred Value : 0.9603

Neg Pred Value : 0.1494

Prevalence : 0.9460

Detection Rate : 0.8351

Detection Prevalence : 0.8696

Balanced Accuracy : 0.6219

> confusionMatrix(churntest$Churn, as.factor(churntest$Pred\_Churn1))#28%

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 520 60

1 59 28

Accuracy : 0.8216

95% CI : (0.7904, 0.8499)

No Information Rate : 0.8681

P-Value [Acc > NIR] : 0.9997

Kappa : 0.2173

Mcnemar's Test P-Value : 1.0000

Sensitivity : 0.8981

Specificity : 0.3182

Pos Pred Value : 0.8966

Neg Pred Value : 0.3218

Prevalence : 0.8681

Detection Rate : 0.7796

Detection Prevalence : 0.8696

Balanced Accuracy : 0.6081

'Positive' Class : 0