IME672A
Data Mining and Knowledge Discovery

13438-Nikhil Sahu
13370-Varun Rathi
13129-Anshul Gautam
13697-Siddharth Saurav
13381-Maneesh Kumar Meena

Project Report

Abstract

We will use different classification algorithms to identify the churn situation. In this report we will go through different-different steps to predict churn case. In our case we are creating churn model for available cell2cell company's data. We will move through certain points-

1. Introduction
2. Data
3. Data Preprocessing
4. Classification Models
5. Results and Discussion
6. References

1. Introduction

A "Churn Model" analyzes the history of customers who have churned and those that have stayed and determines the probability that a current customer with a particular profile will churn. In general these models have two main uses: (1) understanding the factors that influence a customer to churn and (2) identifying customers for whom these factors make it likely that they will churn. For example, if the model shows that customers in a certain geographic region were more likely to churn, the carrier could take steps to improve reception clarity in that region, or target an advertising campaign to the region. Churning becomes a major problem for wireless companies. Sometimes it leads to high losses to company. In order to create this model we will use certain classification methods, like Multiple/Logistic Linear Regression, Decision Tree etc. In this project, we would develop a model to identify potential churners ahead of time and target them with the appropriate incentives to entice them to stay.

Now a days wireless industries are growing rapidly. Because of a lot of choices, customers are kings for companies. Companies realize that the customer acquisition side is more limited than in the growth years of the 90's, but still use price and service deals to attract new customers, usually now at the expense of a competitor. On the development side (maximizing ARPU), they have developed and targeted new services. Some of these are attractive to customers, but the lack of a significant "killer app" has lessened the effectiveness of this strategy. Not surprisingly, attention has focused on retention, minimizing the number of customers who defect to another company. These defections are called customer "churn."

There are many reasons a customer might think of churning, and the switching cost for translating that desire into action is relatively small due to: (i) variety of companies, (ii) the similarity of their offerings and (iii) the cheap prices of handsets. In fact, the biggest switching cost, "number portability" (not allowing customers to hold on to their cell-phone numbers when they switch providers), is likely to change in the short term.

A "Churn Model" analyzes the history of customers who have churned and those that have stayed and determines the probability that a current customer with a particular profile will churn. The process of sorting customers by their probability of churning is known as "scoring". Some churn models can identify customers who are five to 10 times as likely to churn as an average customer. In general these models have two main uses: (1) understanding the factors that influence a customer to churn and (2) identifying customers for whom these factors make it likely that they will churn. For example, if the model shows that customers in a certain geographic region were more likely to churn, the carrier could take steps to improve reception clarity in that region, or target an advertising campaign to the region.

We have one purpose of this model, whereby customers with high risk of churn are identified ahead of time and targeted with appropriate marketing actions. Churn modeling offers the promise of facilitating that effort.

2. Data

We have a “calibration” database consisting of 40,000 customers and a “validation” database consisting of 31,047 customers. Each database contained (1) a “churn” variable signifying whether the customer had left the company two months after observation, and (2) a set of 75 potential predictor variables that could be used in a predictive churn model. Following usual model development procedures, the model would be estimated on the calibration data and tested on the validation data. At the time, Cell2Cell’s churn rate was about 2% per month. However, data has the calibration database so that it contained roughly 50% churners. This was to make it easier for whatever data mining tool we used to identify the factors influencing customer churn. The validation data contained 2% churners. The data are available in one data file with 71,047 rows that combines the calibration and validation customers.

Most of the attributes contain personal information of the customers and some of them contain their company record. Majority of types of attributes are numeric and binary.

3. Data Preprocessing

Data preprocessing is an important part of knowledge discovery. We will use R-codes for data preprocessing.

3.1 Handling Missing data

All the missing value (NA) in the data except CHURNDEP is replaced by the mean of that respective column using mean function.

```
data <- read.csv(“cell2cell.csv”)   # reading .csv file
print(data)                          # printing data

# now we will replace mean values column wise
for(i in 1:ncol(data)-1){
        if(i==27)
                {i=28}
        data[is.na(data[,i]), i] <- mean(data[,i], na.rm = TRUE)
}
```

3.2 Data Reduction

3.2.1. Deleting Columns- our data contain some attribute who isn't giving information for any classification model, so, we simply going to remove them.

```
data <- data(,-v(27,31,78,79))  # removing columns like ID,CALIBRAT,CSTOMER ID
```

3.2.2. Merging- In order to merge two or more similar columns that might fall under same category of attribute and the rows which contains all zero values were replaced by maximum value of respective column. We will certain attributes which is giving the same information but one at a time.

```
# for credit rating
cred <- c("highest","high","good","medium","low","verylow","lowest")
for(i in 35:41){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,80]=cred[c(i-34)] } } }
```

```
# for prizm
prz <- c("rural","suburban","town")
for(i in 42:44){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,81]=cred[c(i-41)] } }
```

```
# for occupation
occup <- c("professional", "clerical", "crafts", "student", "homemaker", "retired", "self-
employed")
for(i in 49:55){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,82]= occup[c(i-48)] } } }
```

```
# for marital status
mar <- c("unknown", "married", "unmarried")
for(i in 57:59){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,83]= mar[c(i-56)] } } }
```

```
mail <- c("mailord", "mailres", "mailres")
# for mail response
for(i in 60:62){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,84]= mail[c(i-59)]   }  }    }
```

```
# for user type
user <- c("newcell", "notnewcell")
for(i in 68:69){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,85]= user[c(i-67)]   }     }       }
```

```
# for income
income <- c("incmiss", "income")
for(i in 71:72){
        for(j in 1:nrow(data)){
                        if(data[j,i]==1){
                                data[j,86]= income[c(i-70)]  } }  }
```

3.2.3 Correlation-   If more than one attributes are highly correlated then they will give same information so we remove these attributes. Threshold value for correlation is .75.

```
# correlation matrix
for(i in 1:48){
         for(j in (i+1):49){
                 if (cor(data[,i],data[,j])>.75){
                         count=count+1 print(i) print(j) print("end") }}}
```

```
# reduced columns
data<-data[,-c(15,18,19,20,18,19,17,26,28,48)]
```

3.3 Handing Noisy Data

All the outliers are replaced by the quartile values (either upper quartile or lower quartile) of the respective column except the columns which contains binary input and columns which are merged together as they are shifted in the last to understand the data better.

```
for(i in 1:49) {
        x <- median(data[,i], na.rm = TRUE)
        qnt<-quantile(data[,i],c(.25,.75),na.rm=TRUE)
        h=1.5*IQR(data[,i],na.rm=TRUE)
        lwr=qnt[1]-h
        upr=qnt[2]+h
        count=0
         for (j in 1:nrow(data)){
                if (!(is.na(data[j,i]))){
                        if(data[j,i]<lwr||data[j,i]>upr){
                                data[j,i]<- x
                                count=count+1}}}print (count)  }
```

3.4 Rough Set for Data Reduction

In order to reduce the attributes the Rough set algorithm is applied over z- score normalized data.

```
# z-score normalization
for(i in 1:ncol(data)){
        x <- data[,i]
        mean <- mean(x)
        sd <- sd(x)
        for(j in 2:nrow(data)){
                data[j,i] <- (data[j,i]-mean)/sd
        }
 }
```

The fundamental concept behind Rough Set Theory is the approximation of lower and upper spaces of a set, the approximation of spaces being the formal classification of knowledge regarding the interest domain. The subset generated by lower approximations is characterized by objects that will definitely form part of an interest subset, whereas the upper approximation is characterized by objects that will possibly form part of an interest subset. Every subset defined through upper and lower approximation is known as Rough Set.

```
# rough set
>data_before_discretization <- read.csv(file.choose(),header=TRUE)
>data_before_discretization <- data_before_discretization[,-1]
>decisiontable.1<- SF.asDecisionTable(dataset = data_before_discretization, decision.attr =45)
>cut.values <- D.discretization.RST(decisiontable.1, type.method = "global.discernibility")
>d.tra <- SF.applyDecTable(decisiontable.1, cut.values)
>red.rst <- FS.greedy.heuristic.reduct.RST(d.tra,,21,qualityF = X.gini,epsilon=0)
>fs.tra <- SF.applyDecTable(d.tra,red.rst)
```

Attributes that are deleted after applying rough sets to the data-

CALIBRAT,MCYCLE,REFER,RETACCPT,RETCALLS,CREDITCD,TRUCK,RV,OWNRENT,TRAVEL,CREDITAD,REFURB,AGE1,PHONES,CALLFWDV,OUTCALLS,occupation,mail,CHURN

- NA in the test data i.e. data[1:31046,29] in CHURNDEP is replaced by the respective values of the CHURN column .

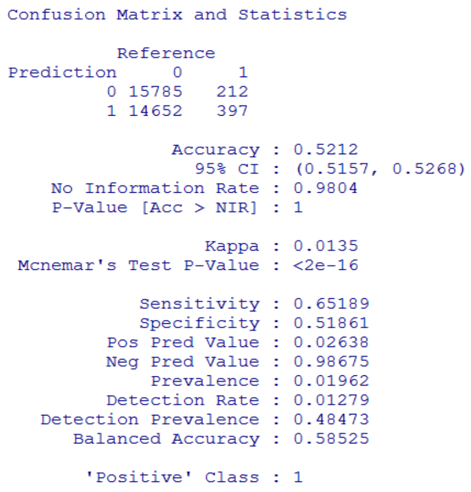
4. Classification

4.1 Decision Tree

Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

The basic syntax for creating a decision tree in R is − *ctree (formula,data)*

```
> library(party)
> test<- data[1:31046,]
> train <-data[31047:71046,]
> modeldt <- ctree(CHURNDEP ~. , data = train)
> predictions = predict(modeldt,test)
> pred <- ifelse(predictions>0.5,1,0)
>library(caret)
>
confusionMatrix(data=factor(pred),reference=factor(test$CHURNDEP),positive='1')
```



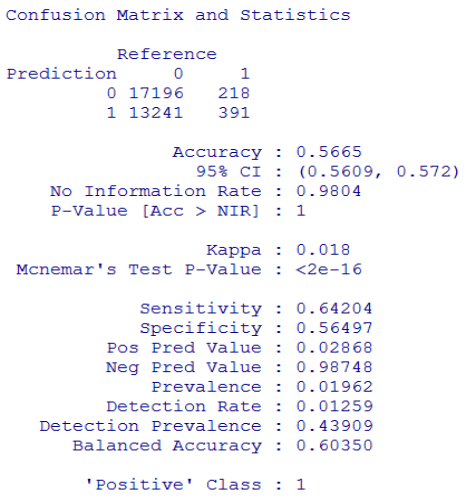
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 15785   212
         1 14652   397

               Accuracy : 0.5212
                 95% CI : (0.5157, 0.5268)
    No Information Rate : 0.9804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0135
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.65189
            Specificity : 0.51861
         Pos Pred Value : 0.02638
         Neg Pred Value : 0.98675
             Prevalence : 0.01962
         Detection Rate : 0.01279
   Detection Prevalence : 0.48473
      Balanced Accuracy : 0.58525

       'Positive' Class : 1
```

4.2 SVM (Support Vector Machine)

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. In our case we have used regression SVM.

The basic syntax for **svm()** function in logistic regression is – *svm (formula,data)*

```
> library(e1071)
> test<- data[1:31046,]
> train <-data[31047:71046,]
> modelsvm <- svm(CHURNDEP ~. , data = train)
> predictions = predict(modelsvm,test)
> pred <- ifelse(predictions>0.5,1,0)
>library(caret)
>
confusionMatrix(data=factor(pred),reference=factor(test$CHURNDEP),positive='1')
```

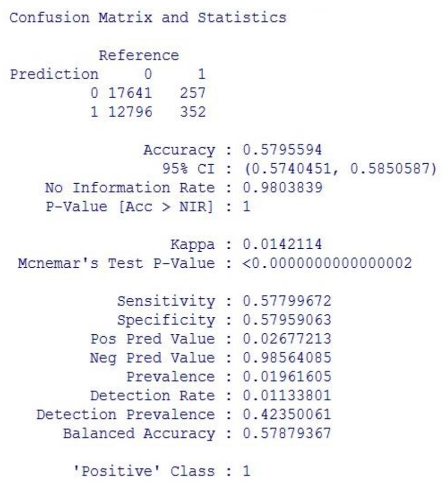


```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 17196   218
         1 13241   391

               Accuracy : 0.5665
                 95% CI : (0.5609, 0.572)
    No Information Rate : 0.9804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.018
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.64204
            Specificity : 0.56497
         Pos Pred Value : 0.02868
         Neg Pred Value : 0.98748
             Prevalence : 0.01962
         Detection Rate : 0.01259
   Detection Prevalence : 0.43909
      Balanced Accuracy : 0.60350

       'Positive' Class : 1
```

4.3 Neural Network

Neural Network (NN) has neurons and layers in its architecture. 3 layers in Neural Network are *Input Layer* , *Hidden Layer* and *Output Layer*. Each of these layers have units or neurons in it. Based on the sample data, the weights have to be estimated. There are various algorithms to training the network. The estimate labels are compared against target or actual labels. The gap is called Error and the error is back propagated to improve the weights and reduce the error.

```
> library(neuralnet)
> test<- data[1:31046,]
> train <-data[31047:71046,]
modelnn <- neuralnet(CHURNDEP ~. , data = train,
hidden=0,threshold=0.01,act.fct="tanh")
> predictions = predict(modelnn,test)
> pred <- ifelse(predictions$net.result>0.5,1,0)
>library(caret)
>
confusionMatrix(data=factor(pred),reference=factor(test$CHURNDEP),positive='1')
```



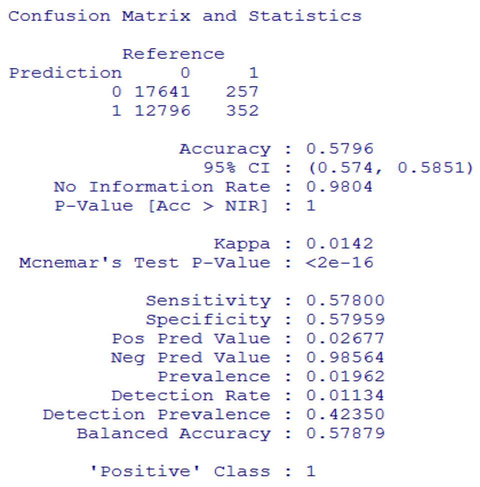
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 17641   257
         1 12796   352

               Accuracy : 0.5795594
                 95% CI : (0.5740451, 0.5850587)
    No Information Rate : 0.9803839
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0142114
 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.57799672
            Specificity : 0.57959063
         Pos Pred Value : 0.02677213
         Neg Pred Value : 0.98564085
             Prevalence : 0.01961605
         Detection Rate : 0.01133801
   Detection Prevalence : 0.42350061
      Balanced Accuracy : 0.57879367

       'Positive' Class : 1
```

4.4 Multiple Linear Regression

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.
The basic syntax for **lm()** function in multiple regression is –

lm(y ~ x1+x2+………+xn, data) y= response variable , x = predictor variables.

```
> library(party)
> test<- data[1:31046,]
> train <-data[31047:71046,]
> modellm <- lm(CHURNDEP ~. , data = train)
> predictions = predict.lm(modellm,test)
> pred <- ifelse(predictions>0.5,1,0)
> library(caret)
>
confusionMatrix(data=factor(pred),reference=factor(test$CHURNDEP),positive='1')
```



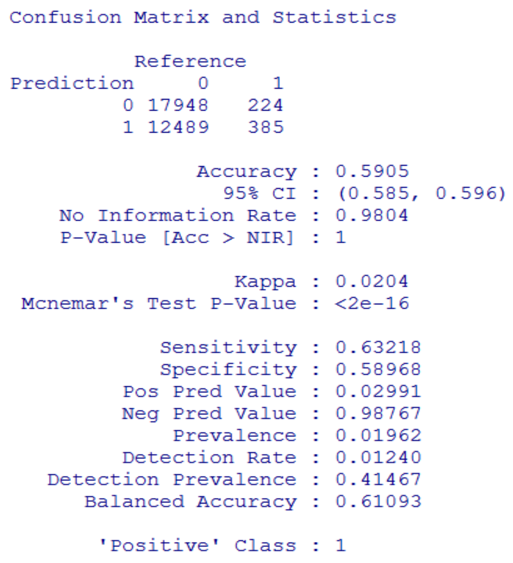
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 17641   257
         1 12796   352

               Accuracy : 0.5796
                 95% CI : (0.574, 0.5851)
    No Information Rate : 0.9804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0142
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.57800
            Specificity : 0.57959
         Pos Pred Value : 0.02677
         Neg Pred Value : 0.98564
             Prevalence : 0.01962
         Detection Rate : 0.01134
   Detection Prevalence : 0.42350
      Balanced Accuracy : 0.57879

       'Positive' Class : 1
```

4.5 Random Forest

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

The basic syntax for creating a random forest in R is – *RandomForest(formula,data)*

```
> library(randomForest)
> test<- data[1:31046,]
> train <-data[31047:71046,]
> modelrf <- randomForest(CHURNDEP ~. , data =
train)
> predictions = predict(modelrf,test)
> pred <- ifelse(predictions>0.5,1,0)
>library(caret)
>
confusionMatrix(data=factor(pred),reference=fact
or(test$CHURNDEP),positive='1')
```



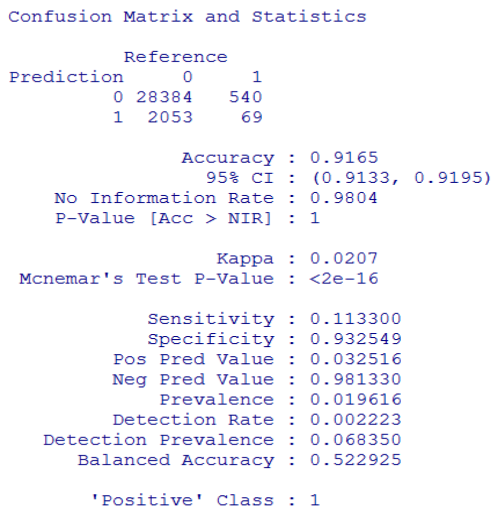
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 17948   224
         1 12489   385

               Accuracy : 0.5905
                 95% CI : (0.585, 0.596)
    No Information Rate : 0.9804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0204
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.63218
            Specificity : 0.58968
         Pos Pred Value : 0.02991
         Neg Pred Value : 0.98767
             Prevalence : 0.01962
         Detection Rate : 0.01240
   Detection Prevalence : 0.41467
      Balanced Accuracy : 0.61093

       'Positive' Class : 1
```

4.6 Logistic Regression

The is a regression model in which the response variable has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The basic syntax for **glm()** function in logistic regression is – *glm(formula,data,family)*

```
> library(party)
> test<- data[1:31046,]
> train <-data[31047:71046,]
> modelglm <- glm(CHURNDEP ~. , data = train
,family='binomial')
> predictions = predict.lm(modelglm,test)
> pred <- ifelse(predictions>0.5,1,0)
> library(caret)
>
confusionMatrix(data=factor(pred),reference=fact
or(test$CHURNDEP),positive='1')
```



```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 28384   540
         1  2053    69

               Accuracy : 0.9165
                 95% CI : (0.9133, 0.9195)
    No Information Rate : 0.9804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0207
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.113300
            Specificity : 0.932549
         Pos Pred Value : 0.032516
         Neg Pred Value : 0.981330
             Prevalence : 0.019616
         Detection Rate : 0.002223
   Detection Prevalence : 0.068350
      Balanced Accuracy : 0.522925

       'Positive' Class : 1
```

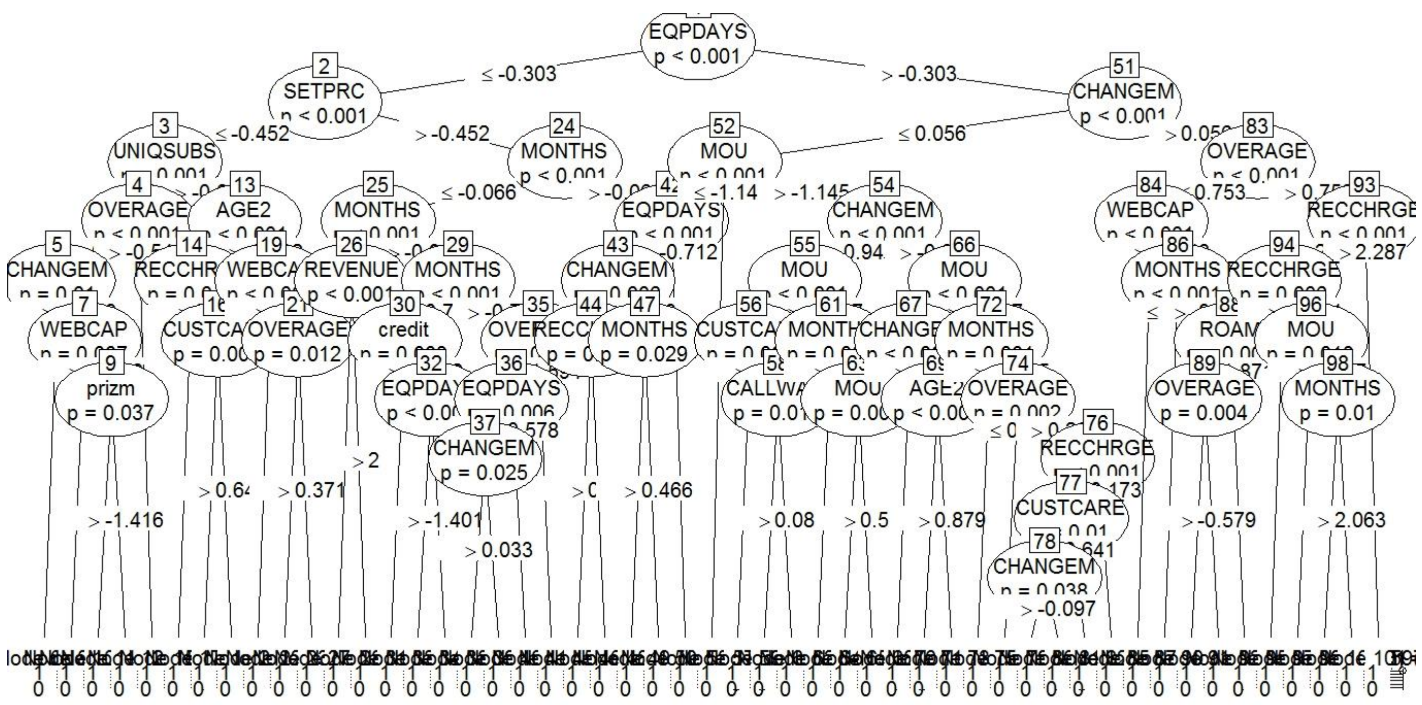
5. Result & Discussion-

	Accuracy	Sensitivity	Specificity
Decision Tree	0.5212	0.6519	0.5186
SVM	0.5665	0.6420	0.5650
Neural Network	0.58	0.5780	0.5796
Multilinear Regression	0.5796	0.5780	0.5796
Random Forest	0.5905	0.6322	0.5897
Logistic Regression	0.9165	0.133	0.9325

As can seen from the table, the performance of Logistic regression is best among the different technique used with respect to the accuracy of the model

(i) From the comparing of different model, we can say that our logistic regression has the highest accuracy, but, if we look at the sensitivity, it is relatively low. So, if we consider accuracy and cost (look at discussion no. 4) of the future churn management plan than we will choose logistic regression. To create this model, we have used all attributes from final reduced data.

(ii).



- EQPDAYS: Number of days of the current equipment
- MONTHS: Months in service
- CAHNGEM: % Change in minutes of usage
- MOU: Mean monthly minutes of use.
- RECCHRGE: Mean Total recurring charge
- UNIQSUBS: Unique Subscription by Customer

Variables like months, MOU, UNIQSUBS, and Change in Revenues are major attributes in cell2cell data. It indicates longer the time a customer remains with a Cell2Cell lesser will be its churn. Also number of calls made to the retention team has direct effect on the churn rate. Variable like MOU, UNIQSUBS, RECCHARGE are actionable. EQPDAYS, MONTHS and RECCHARGE can be improved by various promoting program by the provider.

(iii) Based on above derived important variables, the following incentives plan can be offered to the customers to reduce the possible churn

- From the model we got EQPDAYS as one of the important factors for churn prediction. From decision tree we can see that after 303 days customers change their handset.
- Company can offer new cell phones without changing their connection or upgrading their present connection with better offers.
- By offering various discount plans to the customers in order to increase the usage minutes by customer saving old cell phones.
- The most common effort to manage churn today is the Reactive Retention Program, where company representatives attempt to convince customer to stay at the exact moment they are calling to leave the company.
- Some other methods for customer acquisition are devising adequate service plans, setting up the pricing structure, choosing the handsets to be launched and laying out the advertising and promotion plan.

(iv) Profitabilit-

		Decision (from Predictive Modeling)	
		Churn	**Not Churn**
Actual	**Churn**	True Positive *(Profit = LCV – Incentive Cost)*	False Negative
	Not Churn	False Positive *(Loss = Incentive Cost)*	True Negative

For Churn-

True Positive- These are the customers which is predicted by our model and he accepted company's offer. So, the profit for the company-

Profit= Revenue from Customer- Cost of offer;

False Positive- These are the customers which is predicted falsely and he accepted company's offer. But, he don't have any incentive to churn.

Loss = cost of offer.

If our Profit is more than the cost then our model is a success.

6. References-

http://dni-institute.in/blogs/neural-network-tutorial/

http://www.tutorialspoint.com/r/index.htm

http://datascienceplus.com/fitting-neural-network-in-r/

http://www.cyclismo.org/tutorial/R/plotting.html

https://stat.ethz.ch/pipermail/r-help/2004-December/062499.html

http://www.svm-tutorial.com/2014/10/support-vector-regression-r/