Classification Analysis of Customer Churn Data

By: Pranav Vinod

Supervisor: Professor Donghui Yan

MTH 522: Advanced Mathematical Statistics

University of Massachusetts Dartmouth

05/5/2022

Contents

Abstract

Churn (loss of customers to competition) is a problem for telecom companies because it is more expensive to acquire a new customer than to keep existing one from leaving the company. Therefore, the company tries to identify in advance, customers who are likely to churn. The company then targets those customers with special programs or incentives. This approach can bring in huge loss for a company, if churn predictions are inaccurate, because then firms are wasting incentive money on customers who would have stayed anyway.

In this project, we will try to predict whether a customer will churn, i.e., if a customer will cancel their telecom subscription based on their activity data.

To do so, we will implement Logistic Regression, Random Forest and a Neural Network model to find the best one.

Data Description

COLUMN NAME	DESCRIPTION
State	State where the account is based
Account.length	Time that the account has been active
Area.code	Area code of the account
Internation.plan	Whether account has international plan or not
Voice.mail.plan	Whether account has voice plan or not
Number.vmail.messages	Number of voice mail messages
Total.day.minutes	Total day minutes used
Total.day.calls	Day calls made
Total.day.charge	Total day charge
Total.eve.minutes	Total evening minutes used

Total.eve.calls	Evening calls made
Total.eve.charge	Total evening charge
Total.night.minutes	Total night minutes used
Total.night.calls	Night calls made
Total.night.charge	Total night charge
Total.intl.minutes	Total international minutes used
Total.intl.calls	Total international calls made
Total.intl.charge	Total international charge
Customer.service.calls	Number of service calls made
Churn	Whether the customer churns, 0 if True, 1 if False

The dataset consists of 2666 observations of 20 variables. Of those 20, 4 including the target variable are categorical and 16 are numerical. The target variable in this dataset is Churn, which is has 2 levels True and False signifying if a customer will leave the telecom network or not respectively.

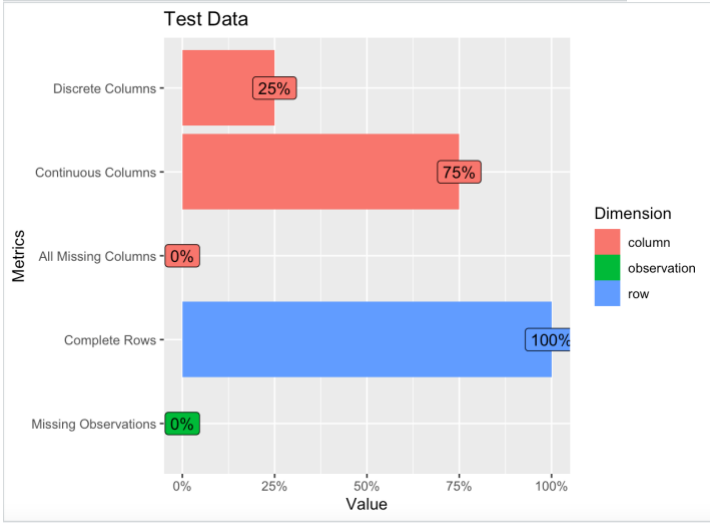
The dataset has information about the state and area code of the telecom connection. Then it has information about customer usage patterns. For example, it tells us the total minutes that are used by a customer in a day, evening and night. Accompanying that information is the charge associated with those calls. Also, the dataset has information about international calls placed by each customer and the charge and duration of those calls.
Finally, it tells us about the number of customer service calls made by the customer and if they have left the telecom provider for a competing organization.

From the outset, I have dropped the column with state names. It tells us the state where the account is based and I felt that information was being reflected in the Area code of the account column with the name Area.code. So, it did not make sense to have 2 attributes conveying the same information.

Data Exploration

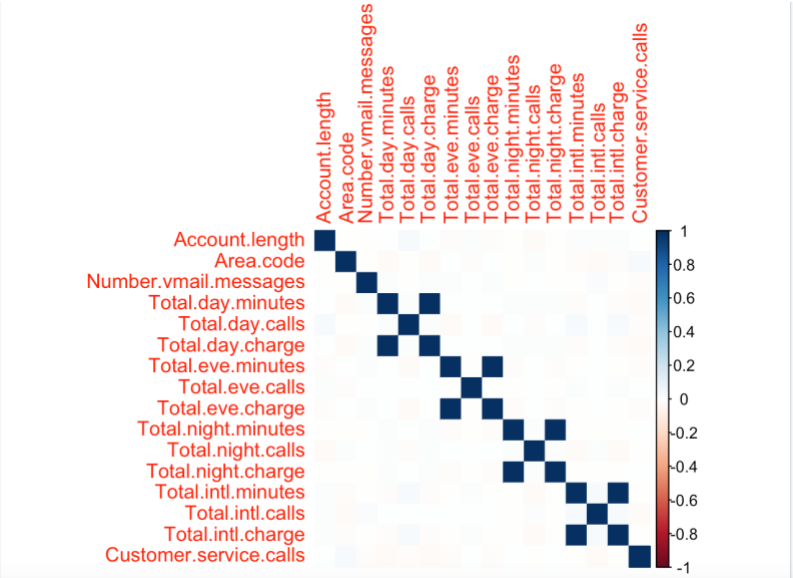
I have explored the dataset by using libraires of R to find interesting features present in the data.

To begin with I have tried to find if there are any missing values present in the training and testing sets.



There were no missing values in the dataset which certainly made further processing easier. 75% of the columns present in the dataset were numerical and the other 25 % were categorical.

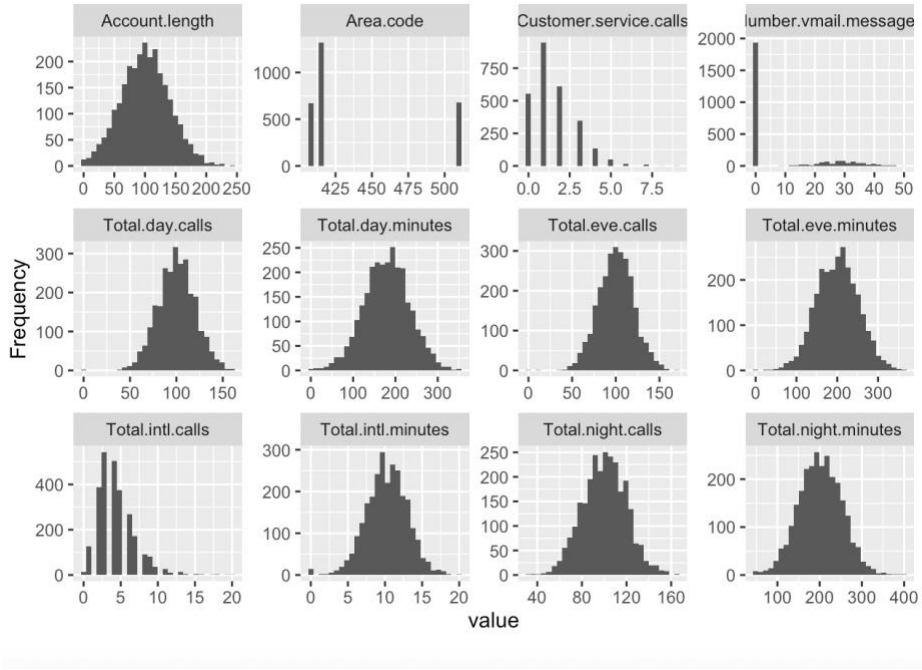
Now, I examined the numerical columns of the dataset to find if there was any collinearity present between any two pairs.



On examining the correlation plot of the numerical variables, we can see that 4 pairs of attributes are positively correlated to each other. These were Total day charge/Total day minutes. Total night charge/total night minutes. Total evening charge/total evening minutes. Total international charge/total international minutes. It is sensible that these pairs of attributes are correlated among themselves because money charged to a person would be proportional to the total minutes that they have use the network for. At the same time, it did not make sense to keep these correlated variables in the dataset. Since one of each pair conveyed the same information, we can safely drop the other one and proceed with further analysis.

So, I dropped the columns for all the charges and kept information about the total minutes in the training set.

Moving on, I plotted the frequency distribution for all numerical variables in the dataset. And I found that most of the attributes followed a normal distribution.



The area code was specific, implying that this information was taken from customers in specific areas and not throughout the country.

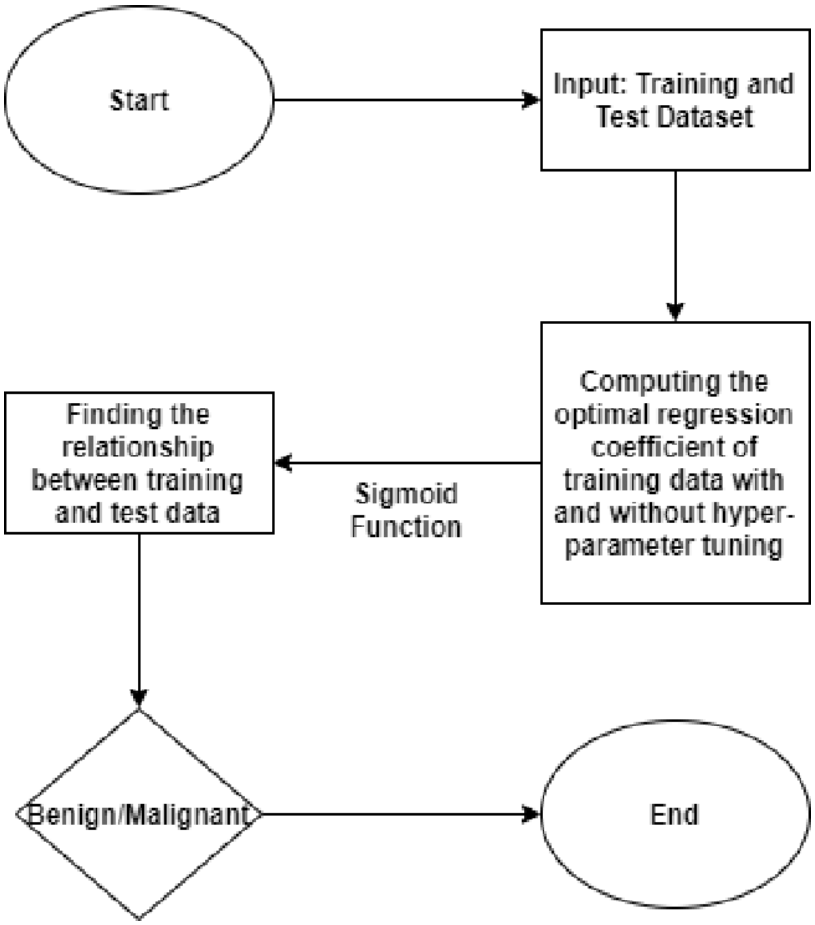
Lastly, I tried to find that distribution of the target variable churn. I found that most of the cases were for people who did not churn. There were 2278 cases for people who did not churn whereas there were only 388 observations for people who left the network for their competition. The ratio was consistent throughout the train and the test set at approx. 0.170 for true to false observations.

```
> table(train$Churn)   # 0.172, more cases of false than true

False  True
 2278   388
> table(test$Churn)    # 0.166  more cases of false than true

False  True
  572    95
```

Logistic regression

Logistic regression is a machine learning model used for classification of categorical variables. Logistic regression makes use of odds ratio and the logit function to create a generalized linear model

The algorithm for a logistic regression model can be understood through the following flowchart:



The steps involved in the algorithm are as follows:

1.We start off with the training and testing set
2. We compute the optimal regression coefficients with the training data
3. Then using the sigmoid/logit function, we find the relationship between the probabilities and the regression output.
4. Finally, we interpret those output in the context of our dataset and evaluate our model.

Now I implemented the logistic regression model in R using the *glm* function

```
Call:
glm(formula = Churn ~ ., family = binomial(link = "logit"), data = train_final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2287   0.1982   0.3410   0.5128   2.1255

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           7.7395755  1.0361961   7.469 8.07e-14 ***
Account.length       -0.0008571  0.0015685  -0.546  0.58476
Area.code             0.0006087  0.0014690   0.414  0.67861
International.planYes -2.0977591  0.1595518 -13.148  < 2e-16 ***
Voice.mail.planYes     2.0315470  0.6553951   3.100  0.00194 **
Number.vmail.messages -0.0374140  0.0206236  -1.814  0.06966 .
Total.day.minutes     -0.0125970  0.0012187 -10.337  < 2e-16 ***
Total.day.calls       -0.0028906  0.0030978  -0.933  0.35075
Total.eve.minutes     -0.0056747  0.0012705  -4.467 7.95e-06 ***
Total.eve.calls        0.0007921  0.0030824   0.257  0.79720
Total.night.minutes   -0.0028341  0.0012418  -2.282  0.02247 *
Total.night.calls     -0.0020081  0.0031666  -0.634  0.52598
Total.intl.minutes    -0.1000652  0.0227692  -4.395 1.11e-05 ***
Total.intl.calls       0.1201464  0.0287918   4.173 3.01e-05 ***
Customer.service.calls -0.5077384  0.0440787 -11.519  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2212.2  on 2665  degrees of freedom
Residual deviance: 1730.6  on 2651  degrees of freedom
AIC: 1760.6

Number of Fisher Scoring iterations: 6
```

$$\log(odds(P)) = 7.73 - 0.012 * Total.day.minutes - 0.005 * Total.eve.minutes - 0.0007 * Total.night.minutes - 0.100 * Total.intl.minutes + 0.120 * Total.intl.calls - 0.507 * Customer.service.calls$$

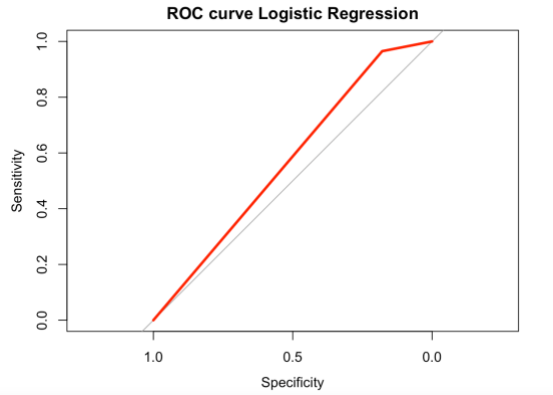
This model created tells us that total international calls, total international minutes, customer service calls, total evening minutes, and if they had an international plan are the significant factors when finding out the churn of a customer.

Further, we desire a small residual deviance as compared to the null deviance. But we can see that the difference between residual and null deviance is not that high. Also, the model has a relatively high AIC value of 1760.6

After getting this model, it is important to evaluate this model.

To do so, I created a confusion table for the logistic regression model and got the following result:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  17  20
         1  78 552

               Accuracy : 0.8531
                 95% CI : (0.8239, 0.8791)
    No Information Rate : 0.8576
    P-Value [Acc > NIR] : 0.655

                  Kappa : 0.1932

 Mcnemar's Test P-Value : 8.518e-09

            Sensitivity : 0.17895
            Specificity : 0.96503
         Pos Pred Value : 0.45946
         Neg Pred Value : 0.87619
              Precision : 0.45946
                 Recall : 0.17895
                     F1 : 0.25758
             Prevalence : 0.14243
         Detection Rate : 0.02549
   Detection Prevalence : 0.05547
      Balanced Accuracy : 0.57199

       'Positive' Class : 0
```



The model had an accuracy of 85% which is good, but it had a ROC curve which does not have the ideal shape. Further, its AUC is low at 0.572. It also has a low F1 score of 25.7%.

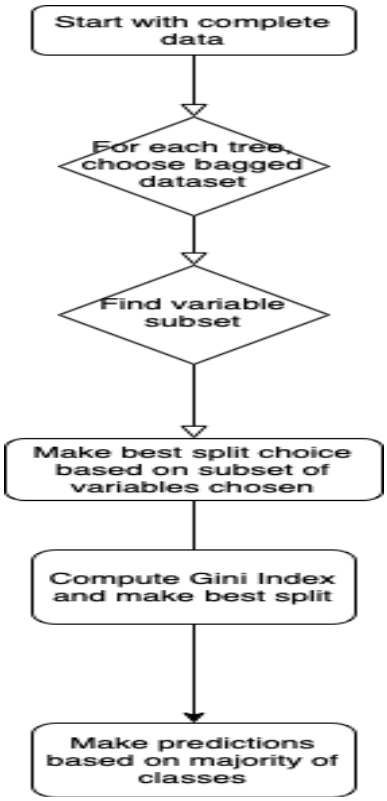
In our case, it is more important to look at the false positive rate. We do not want any customers who would churn to be classified as customers who won't churn. That would lead to a loss of that customer. Because of that, throughout this project, we will focus more on F1 value than Accuracy.

Due to the low AUC and problems with the deviance values, I implemented a random forest on the same dataset.

Random Forest

Random Forest is a type of aggregate decision tree model. Random Forest improves upon a single decision tree by creating a large number of trees and averaging over the value of each tree to give the final result. In case of a classification problem, it chooses the majority output from all trees as the outcome.

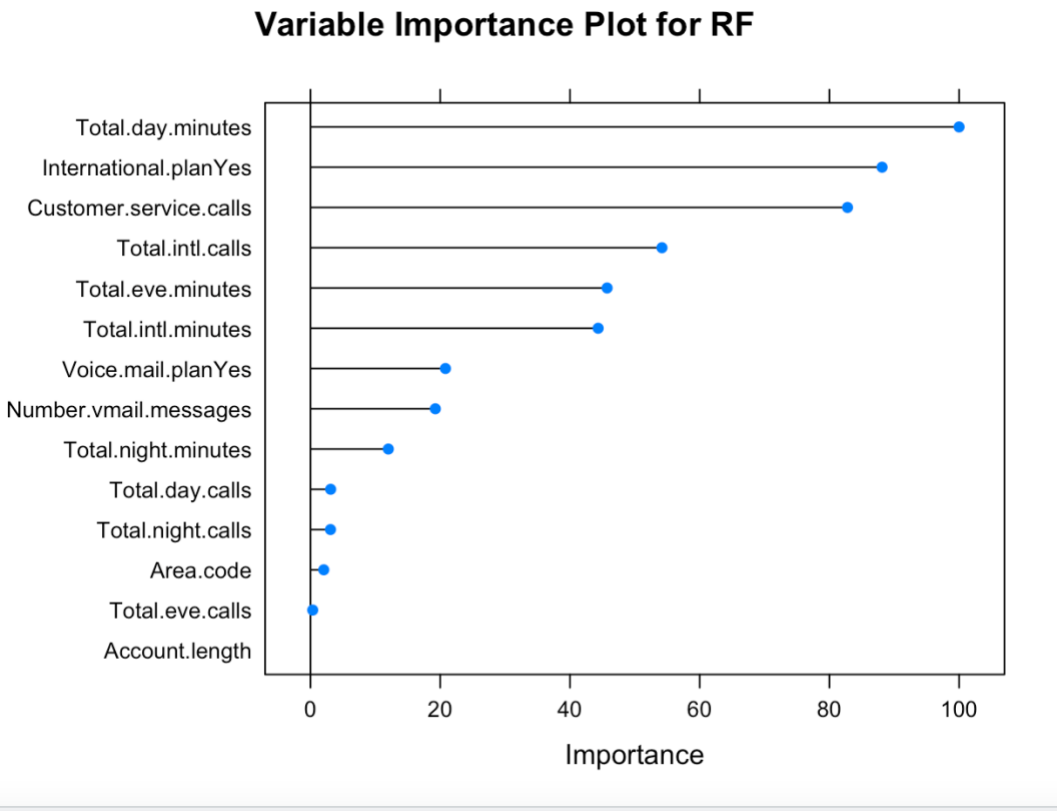
The algorithm for a random forest model is as follows:



The steps are:

1. We start with the complete dataset
2. Then instead of using the train set to start building our tree, we will use a boosted dataset, which is obtained by randomly selecting observations from the train set with replacement. For each tree in our forest, we will choose a unique boosted dataset and start building our tree.
3. Now at each split, we have a choice of variables to choose from. We will randomly select a subset of variables to choose from and constrain ourselves to that subset at each split.
4. To find the best split, we will compute the Gini index for each possible split and choose the one with the smallest Gini index.
5. Finally, on our forest, we will make predictions based on the output from the majority of the trees for a classification problem.

Having implemented the model, the following results were obtained:

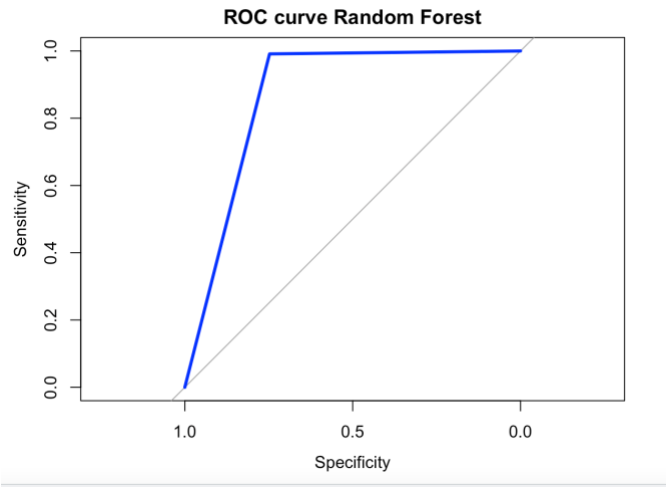


This plot tells which the important variables are while predicting churn of each customer. We find that Total.day.minutes is the most important factor in prediction and the time that they had the account for was the least important variable.

The forest created consisted of 500 trees and at each split 8 of the parameters were chosen at random.

To evaluate this model, I created its confusion table and ROC curve

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  71   5
         1  24 567

               Accuracy : 0.9565
                 95% CI : (0.9382, 0.9707)
    No Information Rate : 0.8576
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8058

 Mcnemar's Test P-Value : 0.0008302

            Sensitivity : 0.7474
            Specificity : 0.9913
         Pos Pred Value : 0.9342
         Neg Pred Value : 0.9594
              Precision : 0.9342
                 Recall : 0.7474
                     F1 : 0.8304
             Prevalence : 0.1424
         Detection Rate : 0.1064
   Detection Prevalence : 0.1139
      Balanced Accuracy : 0.8693

       'Positive' Class : 0
```



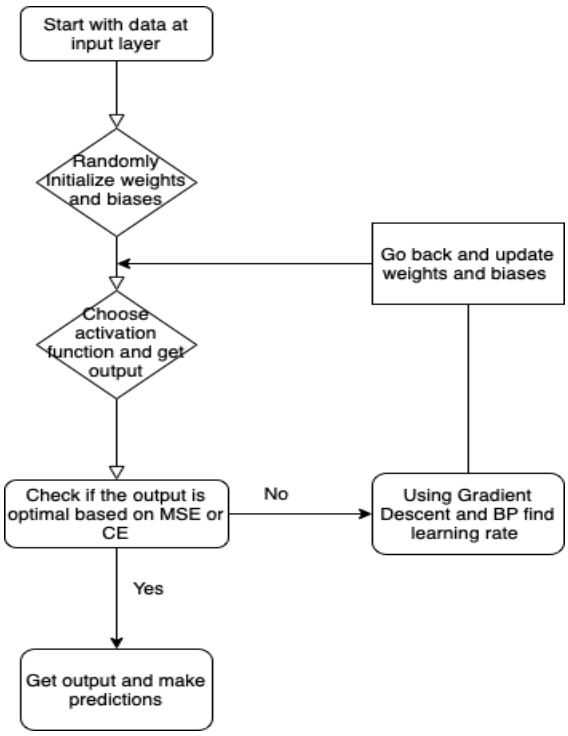
We can see that the random forest has better performance metrics as compared to logistic regression.

It has a high accuracy at 95.6% and an area under curve of 0.868. The shape of the ROC curve is also close to the ideal ROC curve which hugs the top left corner.
It also has an F1 score of 83.04. On all three comparison metrics, random forest performs better.

Having achieved a high accuracy, I tried to better it by implementing a neural network on this dataset.

Neural Network

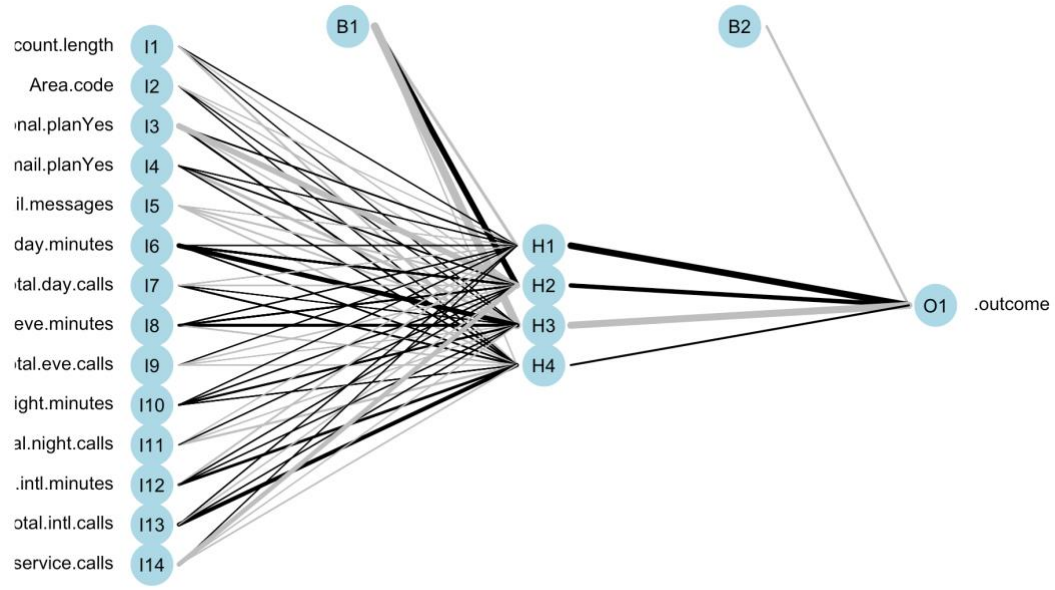
Neural Network is an advanced machine learning model used for regression and classification problems.



The steps involved in the neural network model are as follows:

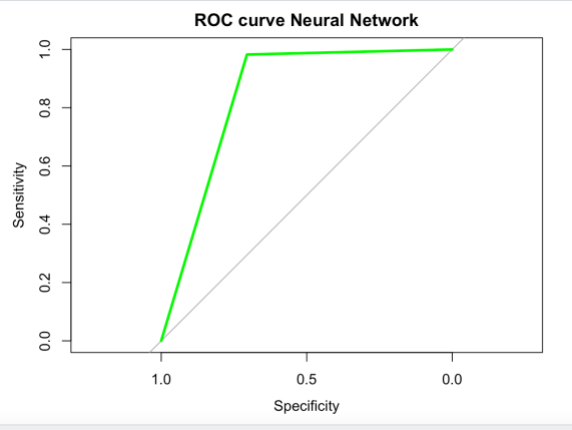
1. It involved randomly initializing weights and biases in the network and then getting an output based on those parameters and the chosen activation function.
2. Having the output as a function of all the parameters in the network, we implement gradient descent to find the optimal value of correction and find the learning rate for those parameters.
3. Finally, using backpropagation, we go back and update these weights and biases until we get an output that does not change beyond a point.
4. The function to minimize in this case is the Cross Entropy in the network for classification problems.

Upon implementing the neural network algorithm, it has 1 hidden layer and 4 nodes within it.



To evaluate this model, I constructed its confusion matrix to compare predictions with the test set.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  67  28
         1  10 562

               Accuracy : 0.943
                 95% CI : (0.9226, 0.9594)
    No Information Rate : 0.8846
    P-Value [Acc > NIR] : 1.793e-07

                  Kappa : 0.7468

 Mcnemar's Test P-Value : 0.00582

            Sensitivity : 0.8701
            Specificity : 0.9525
         Pos Pred Value : 0.7053
         Neg Pred Value : 0.9825
              Precision : 0.7053
                 Recall : 0.8701
                     F1 : 0.7791
             Prevalence : 0.1154
         Detection Rate : 0.1004
   Detection Prevalence : 0.1424
      Balanced Accuracy : 0.9113

       'Positive' Class : 0
```



The results from the neural network were as follows:

It had a high accuracy of 94.3% and an AUC of 0.844.It also has an F1 score of 77.9% Based on these results, we can finally compare all the three models.

Results

	Logistic Regression	Random Forest	Neural Network
Accuracy	0.853	0.956	0.943
AUC	0.572	0.868	0.844
F1 Score	0.257	0.830	0.779

On comparing the three models based on Accuracy and AUC of the model, we can see that logistic regression performs the worst among the three. Among the other two, random forest performs marginally better than neural network, but the difference is very small. Both have high accuracy approximately equal to 95% and an area under the ROC curve of 0.86 and 0.84 respectively.

Further when we look at the F1 score, random forest performs better than neural network significantly.

Lastly, since I got this dataset from Kaggle.com and it was openly available. Some people have previously worked on it and achieved an accuracy of 93% using random forest model.

Conclusion

Therefore, I would recommend using the random forest to predict whether customers will churn or not based on their activity data. The company can target those customers with specialized programs to keep them.

References

1. https://www.researchgate.net/figure/Algorithmic-flow-chart-of-Logistic-Regression-Lei-et-al-2016-42-Support-vector_fig3_343472002
2. https://www.crowdanalytix.com/contests/why-customer-churn
3. Dataset
4. https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Appendix

```
library(dplyr)
library(ggplot2)
library(corrplot)
library(mlbench)
library(caret)
library(pROC)
library(nnet)
library(DataExplorer)
library(NeuralNetTools)

# Read train and test data
train <- read.csv(file.choose(), header = T)
test <- read.csv(file.choose(), header = T)

# Summary and structure
summary(train)
str(train)

# Missing Values
plot_intro(train, title = 'Train Data')
plot_intro(test, title = 'Test Data')
# Distribution of target variable in train and test
table(train$Churn)   # 0.172, more cases of false than true
table(test$Churn)    # 0.166  more cases of false than true

# Histogram of numeric variables
train_num <- train[,-c(1,4,5,20)]
train_num[] <- sapply(train_num, as.numeric)
str(train_num)

# Correlation plot of numeric variables
corr <- cor(train_num)
corrplot(corr, method = 'color')

# Based on correlation plot, we can see that some variables
are strongly correlated to each other
```

```
# This has the potential to skew our analysis, so we drop
one of the correlated variables
# One from each of the 4 pairs

# Dropping those columns from train and test sets
train <- train[,-c(9,12,15,18)]
test <- test[,-c(9,12,15,18)]

# Correlation plot after dropping variables
train_num <- train[,-c(1,4,5,16)]
train_num[] <- sapply(train_num, as.numeric)
corr <- cor(train_num)
corrplot(corr, method = 'color')

# Histogram of numerical variables
plot_histogram(train_num)

# Converting Churn into factor with 0 and 1 level
train <- train %>% mutate(Churn=
                            case_when(Churn=='True'~0,
                                      Churn=='False'~1))
train$Churn <- as.factor(train$Churn)

test <- test %>% mutate(Churn=
                            case_when(Churn=='True'~0,
                                      Churn=='False'~1))
test$Churn <- as.factor(test$Churn)

# Converting International Plan to factor
train <- train %>% mutate(International.plan=
                            case_when(International.plan=='
No'~0,
                                      International.plan=='
Yes'~1))
train$International.plan <-
as.factor(train$International.plan)

test <- test %>% mutate(International.plan=
```

```
                            case_when(International.plan=='No
'~0,
                                      International.plan=='Ye
s'~1))
test$International.plan <-
as.factor(test$International.plan)

# Converting Voice Mail plan to factor

train <- train %>% mutate(Voice.mail.plan=
                              case_when(Voice.mail.plan=='No'
~0,
                                        Voice.mail.plan=='Yes
'~1))
train$Voice.mail.plan <- as.factor(train$Voice.mail.plan)

test <- test %>% mutate(Voice.mail.plan=
                            case_when(Voice.mail.plan=='No'~0
,
                                      Voice.mail.plan=='Yes'~
1))
test$Voice.mail.plan <- as.factor(test$Voice.mail.plan)


# Dropping the state column from data
train_final[] <- train[,-c(1)]
test_final[] <- test[,-c(1)]
str(test_final)

##################################################################
##############################

# Logistic Regression
set.seed(123)
log_reg <- glm(Churn~., data=train_final, family =
binomial(link = 'logit'))
summary(log_reg)
```

```r
logit <- predict(log_reg, test_final, type = 'response')
prob <- as.factor(ifelse(logit>0.5,1,0))

# Confusion Matrix
confusionMatrix(prob, test_final$Churn, mode =
'everything')  # 0.853


library(pROC)
roc_qda=roc(response=test_final$Churn, predictor=
factor(prob,
                                                        ord
ered = TRUE), plot=TRUE)
plot(roc_qda, col="red", lwd=3, main="ROC curve Logistic
Regression")
auc_qda<-auc(roc_qda)

o##################################################################
##############################

# Random Forest
set.seed(123)
cvcontrol <- trainControl(method="repeatedcv",
                          number = 5,
                          repeats = 5,
                          allowParallel=TRUE,
                          savePredictions = T)
set.seed(123)
forest <- train(Churn ~ . ,
                data=train_final,
                method="rf",
                trControl=cvcontrol,
                importance=TRUE)
plot(varImp(forest),
    main = "Variable Importance Plot for RF")

# Testing confusion matrix
p <- predict(forest, test_final, type = 'raw')
```

```
confusionMatrix(p, test_final$Churn, mode =
'everything')  # 0.9565

# ROC Curve
roc_qda=roc(response=test_final$Churn, predictor= factor(p,
                                                         ord
ered = TRUE), plot=TRUE)
plot(roc_qda, col="blue", lwd=3, main="ROC curve Random
Forest")
auc_qda<-auc(roc_qda)   # AUC = 0.868

################################################################
################################
# Neural Network

set.seed(123)
g <- expand.grid(size = seq(from=3, to=9, by=1) ,
                 decay = seq(from=0.1, to=0.7, by=0.05))

Model.nn <- train(Churn ~., data=train_final,
                  method="nnet",
                  trControl=cvcontrol,
                  preProcess=c("center","scale"),
                  tuneGrid = g,
                  maxit = 100,
                  metric="Accuracy")

plotnet(Model.nn$finalModel)

testpred.nn <- predict(Model.nn, test_final)
confusionMatrix(test_final$Churn, testpred.nn,
mode='everything')   #0.943

roc_qda=roc(response=test_final$Churn, predictor=
factor(testpred.nn,
                                                         ord
ered = TRUE), plot=TRUE)
plot(roc_qda, col="green", lwd=3, main="ROC curve Neural
```

```
Network")
auc_qda<-auc(roc_qda)    # AUC = 0.844
```