telco_churn_prediction

Manirou A. T. ILAGOUMA

21/11/2022

Introduction/overview/executive summary

In this project, you will be creating a telco churn prediction system using telco company dataset available on my github repos https://raw.githubusercontent.com/manirou-github/machine_learning_project/main/input_data.csv. First we will look the structure of the data, visualize it and then progressively build a model to predict how likely a customer will churn by analyzing its characteristics. Predicting customer churn is critical for telecommunication companies to be able to effectively retain customers.

Data Exploration

```
#Description Statistics and Data cleaning
str(input_data) #To gain structure of the data
```

```
## spec_tbl_df [550,000 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ custumer_id      : num [1:550000] 72730370 74723760 86977502 86908051 79771039 ...
##  $ tenure_day       : num [1:550000] 401 387 1508 383 673 ...
##  $ status           : num [1:550000] 0 1 1 0 1 1 1 1 1 1 ...
##  $ balance_amnt     : num [1:550000] 151.2 184.6 0.4 501 0 ...
##  $ region           : chr [1:550000] "SOUTH EAST" "NORTH" "CAPITAL CITY" "SOUTH EAST" ...
##  $ rev_o_tot_amnt   : num [1:550000] 400 2154 4100 0 1100 ...
##  $ rev_voix_pyg_amnt: num [1:550000] 100 246 0 0 0 ...
##  $ rev_sms_pyg_amnt : num [1:550000] 0 216 0 0 0 ...
##  $ rev_data_pyg_amnt: num [1:550000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ rev_mms_pyg_amnt : num [1:550000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ rev_subs_amnt    : num [1:550000] 300 1550 4100 0 1100 640 100 4600 100 8200 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   custumer_id = col_double(),
##   ..   tenure_day = col_double(),
##   ..   status = col_double(),
##   ..   balance_amnt = col_double(),
##   ..   region = col_character(),
##   ..   rev_o_tot_amnt = col_double(),
##   ..   rev_voix_pyg_amnt = col_double(),
##   ..   rev_sms_pyg_amnt = col_double(),
##   ..   rev_data_pyg_amnt = col_double(),
##   ..   rev_mms_pyg_amnt = col_double(),
##   ..   rev_subs_amnt = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

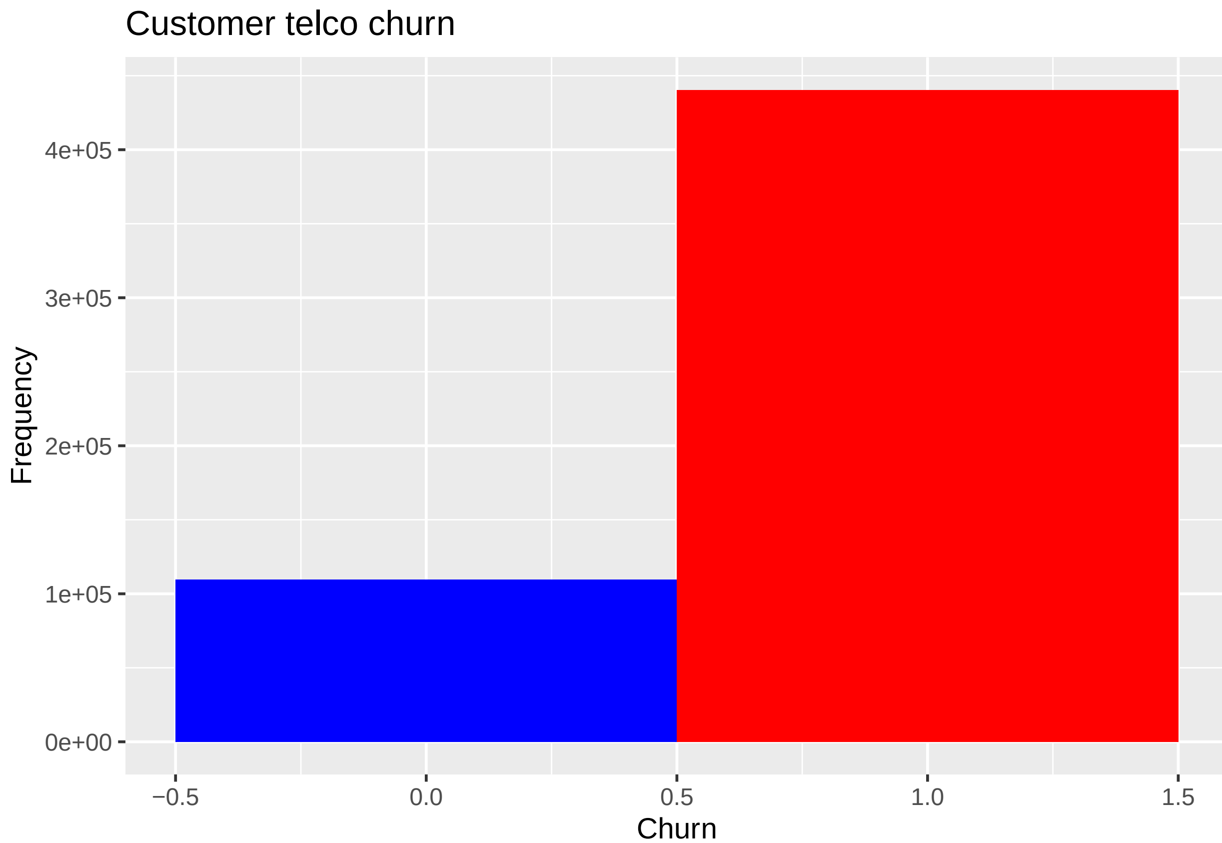
```r
summary(input_data) #To get an understanding of the data
```

```
##   custumer_id         tenure_day        status        balance_amnt     
##  Min.   :70000000   Min.   :    0   Min.   :0.0000   Min.   :-24350.0  
##  1st Qu.:75727973   1st Qu.:  539   1st Qu.:1.0000   1st Qu.:     0.0  
##  Median :81480360   Median :  998   Median :1.0000   Median :     0.9  
##  Mean   :81486217   Mean   : 1351   Mean   :0.8008   Mean   :   117.8  
##  3rd Qu.:87245744   3rd Qu.: 1819   3rd Qu.:1.0000   3rd Qu.:     1.8  
##  Max.   :92999958   Max.   :19265   Max.   :1.0000   Max.   :903870.0  
##     region          rev_o_tot_amnt    rev_voix_pyg_amnt rev_sms_pyg_amnt   
##  Length:550000      Min.   :      0   Min.   :     0.0   Min.   :    0.000  
##  Class :character   1st Qu.:      0   1st Qu.:     0.0   1st Qu.:    0.000  
##  Mode  :character   Median :    200   Median :     0.0   Median :    0.000  
##                     Mean   :   1489   Mean   :   249.8   Mean   :    9.171  
##                     3rd Qu.:   1500   3rd Qu.:   120.0   3rd Qu.:    0.000  
##                     Max.   :2598650   Max.   :527576.5   Max.   :17769.000  
##  rev_data_pyg_amnt  rev_mms_pyg_amnt   rev_subs_amnt    
##  Min.   :     0.0   Min.   :     0.0   Min.   :      0  
##  1st Qu.:     0.0   1st Qu.:     0.0   1st Qu.:      0  
##  Median :     0.0   Median :     0.0   Median :    100  
##  Mean   :    10.1   Mean   :    13.5   Mean   :   1204  
##  3rd Qu.:     0.0   3rd Qu.:     0.0   3rd Qu.:   1150  
##  Max.   :917770.0   Max.   :417105.0   Max.   :2583900
```

We have 550 000 observations in our dataset with custumers characteristics:

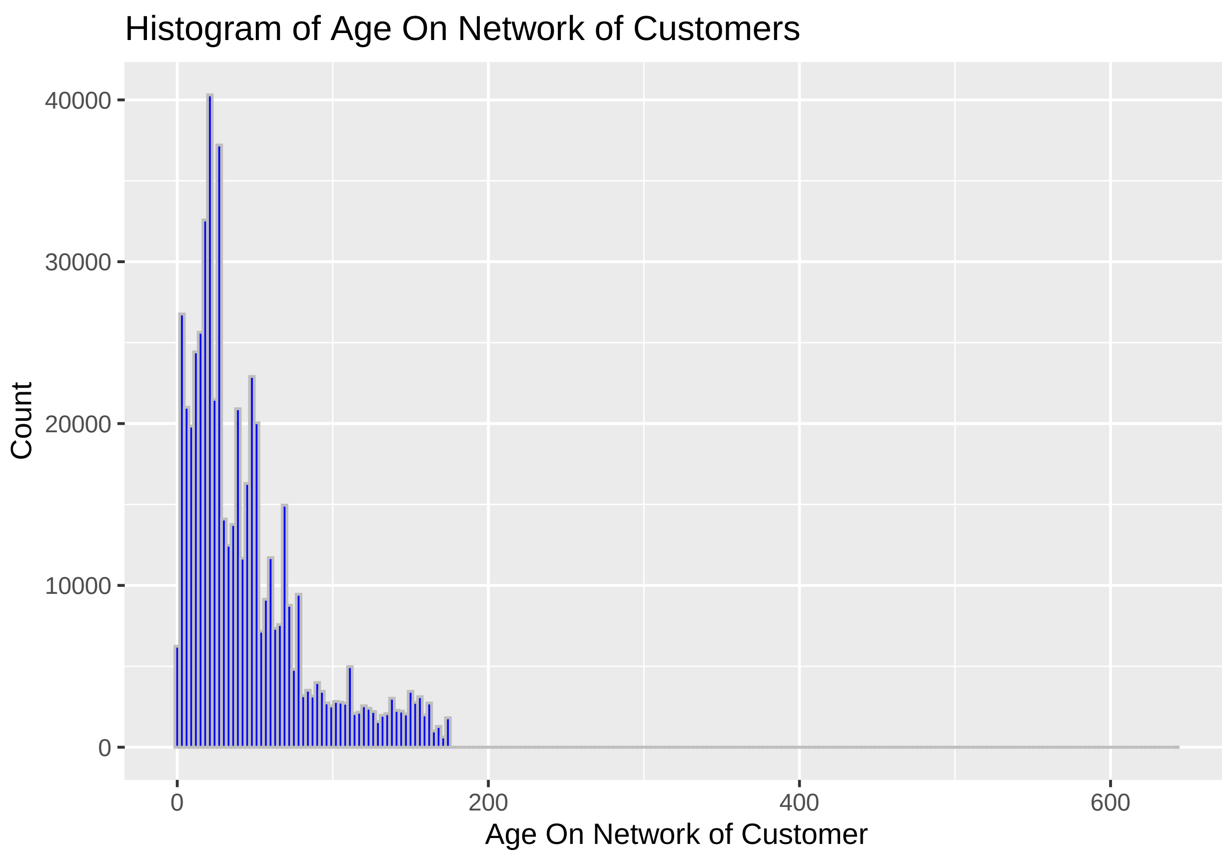
- Custumer profil information : custumer_id, tenure_month, status, balance_amnt
- rev_o_tot_amnt : custumer monthly revenue
- rev_voix_pyg_amnt : custumer monthly voice revenue
- rev_sms_pyg_amnt : custumer monthly sms revenue
- rev_data_pyg_amnt : custumer monthly data revenue
- rev_mms_pyg_amnt : custumer monthly mms revenue
- rev_subs_amnt : custumer monthly service subscription revenue
- custumer area : NORTH, EAST, WEST, SOUTH, CAPITAL CITY, BUSINESS CITY, NORTH WEST, SOUTH EAST

```r
#Customer churn overview
ggplot(input_data,aes(x=status))+
  geom_histogram(binwidth = 1, fill = c("Blue","Red"))+
  labs(title="Customer telco churn",x="Churn",y="Frequency")
```



We have 19,91% of churners

```
#Customer AON(Age On Network) overview
ggplot(input_data,aes(x=tenure_day/30))+
  geom_histogram(color="Gray", binwidth = 3, fill = "Blue")+
  labs(x="Age On Network of Customer",y="Count",title="Histogram of Age On Network of Customers")
```

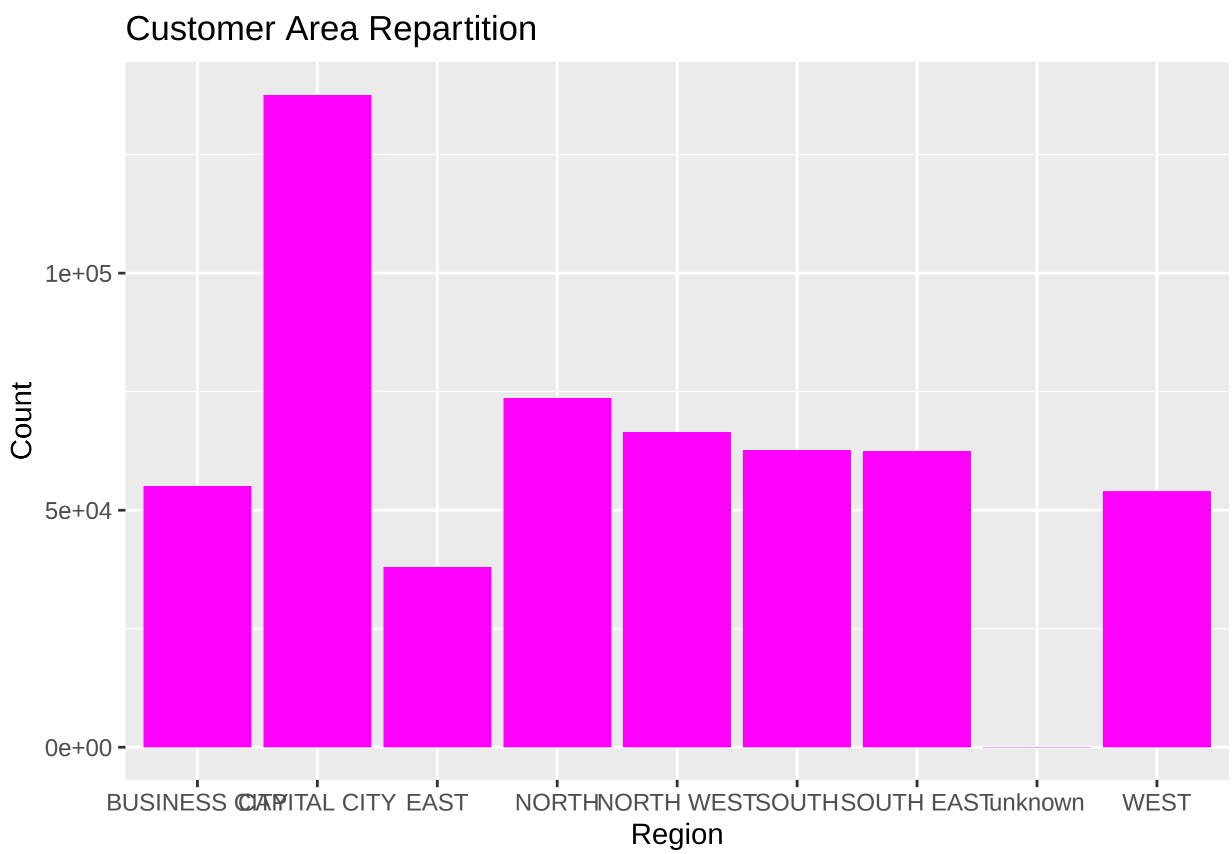



```
mean(input_data$tenure_day/30) # 45 Months
```

```
## [1] 45.04211
```

The average age on network above 45 months, less than 4 years

```
#Custumer Geographic Repartition
ggplot(input_data,aes(x=region))+
  geom_bar(stat="count", fill = "Magenta")+
  labs(x="Region",y="Count",title="Customer Area Repartition")
```



The capital city has the highest number of churn

Methods/Analysis

```
#data cleaning
# transform the variable region into numeric
region_longer <- input_data %>%
       mutate(flag=1) %>%
       replace_na(list(region="unknown")) %>%
       select(custumer_id,region,flag) %>%
       pivot_wider(names_from = region, values_from = flag, values_fill = 0)

tmp <- input_data %>%
       inner_join(region_longer,by=c("custumer_id"="custumer_id")) %>%
       mutate(tenure_month=tenure_day/30) %>%
       select(custumer_id,tenure_month,status,balance_amnt,rev_o_tot_amnt,
              rev_voix_pyg_amnt,rev_sms_pyg_amnt,rev_data_pyg_amnt,
              rev_mms_pyg_amnt,rev_subs_amnt,NORTH,EAST,WEST,SOUTH,`CAPITAL CITY`,`BUSINESS CITY`,`NORTH


input_data <- tmp

rm(tmp,region_longer)
```

```r
#Data partition 80/20 split

input_data$status <- factor(input_data$status)

split_set <- createDataPartition(y=input_data$status, p=.80, list = FALSE)
train_set <- input_data[split_set,]
test_set <- input_data[-split_set,]
```

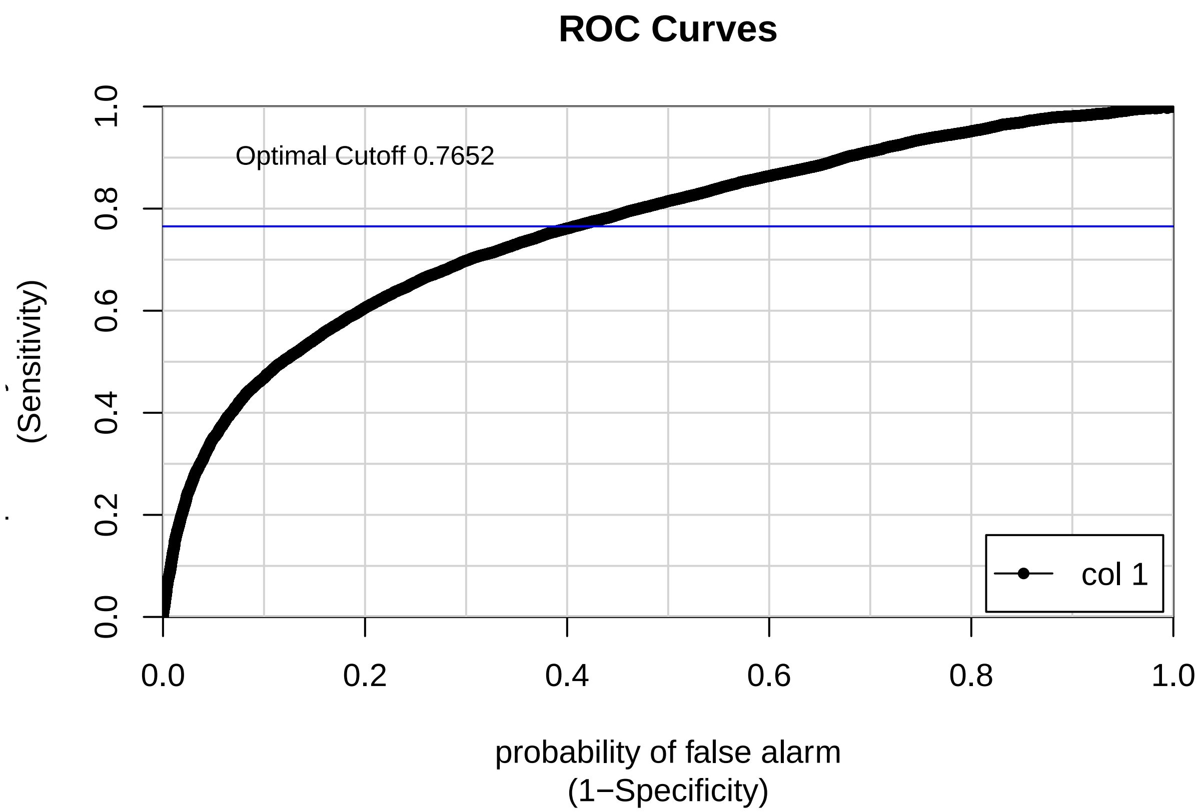
The first model we try will be logistic regression. It involves regressing predictor variables on a binary outcome using a binomial link function. Let's fit the model using the base general linear modeling function in R, glm.

```r
#Model 1 : Logistic Regression
lr_model <- glm(status~tenure_month+balance_amnt+rev_o_tot_amnt+rev_voix_pyg_amnt+
                  rev_sms_pyg_amnt+rev_data_pyg_amnt+rev_mms_pyg_amnt+rev_subs_amnt
                    +NORTH+EAST+WEST+SOUTH+`CAPITAL CITY`+`BUSINESS CITY`+`NORTH WEST`+`SOUTH EAST`, fa
summary(lr_model)
```

```
##
## Call:
## glm(formula = status ~ tenure_month + balance_amnt + rev_o_tot_amnt +
##     rev_voix_pyg_amnt + rev_sms_pyg_amnt + rev_data_pyg_amnt +
##     rev_mms_pyg_amnt + rev_subs_amnt + NORTH + EAST + WEST +
##     SOUTH + `CAPITAL CITY` + `BUSINESS CITY` + `NORTH WEST` +
##     `SOUTH EAST`, family = "binomial", data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.5337   0.0487   0.4828   0.7602   1.6281
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.565e+00  8.239e-01  -1.900  0.05744 .
## tenure_month       1.542e-02  1.453e-04 106.114  < 2e-16 ***
## balance_amnt       2.371e-05  8.231e-06   2.881  0.00397 **
## rev_o_tot_amnt     1.906e-02  1.264e-03  15.080  < 2e-16 ***
## rev_voix_pyg_amnt -1.890e-02  1.264e-03 -14.948  < 2e-16 ***
## rev_sms_pyg_amnt  -1.758e-02  1.287e-03 -13.665  < 2e-16 ***
## rev_data_pyg_amnt -1.906e-02  1.264e-03 -15.079  < 2e-16 ***
## rev_mms_pyg_amnt  -1.905e-02  1.264e-03 -15.075  < 2e-16 ***
## rev_subs_amnt     -1.817e-02  1.265e-03 -14.367  < 2e-16 ***
## NORTH              2.130e+00  8.239e-01   2.586  0.00972 **
## EAST               2.086e+00  8.240e-01   2.531  0.01136 *
## WEST               1.998e+00  8.239e-01   2.425  0.01529 *
## SOUTH              2.195e+00  8.239e-01   2.664  0.00773 **
## `CAPITAL CITY`     1.489e+00  8.239e-01   1.807  0.07073 .
## `BUSINESS CITY`    1.857e+00  8.239e-01   2.253  0.02423 *
## `NORTH WEST`       1.981e+00  8.239e-01   2.404  0.01622 *
## `SOUTH EAST`       2.068e+00  8.239e-01   2.510  0.01207 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 439374  on 440000  degrees of freedom
## Residual deviance: 380909  on 439984  degrees of freedom
## AIC: 380943
##
## Number of Fisher Scoring iterations: 8
```

```
#Model prediction
lr_prediction <- predict(lr_model, test_set, type = "response")

#Generate ROC curve
model_AUC <- colAUC(lr_prediction,test_set$status,plotROC = T)
abline(h=model_AUC, col = "Blue")
text(.2,.9,cex = .8, labels = paste("Optimal Cutoff", round(model_AUC,4)))
```




```
#Convert probabilities to class
churn_class <- ifelse(lr_prediction>0.76,1,0)

churn_class <- factor(churn_class)

#Confusion Matrix
confusionMatrix(churn_class,test_set$status)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction     0     1
##          0 16290 29733
##          1  5621 58355
##
##                Accuracy : 0.6786
##                  95% CI : (0.6758, 0.6814)
##     No Information Rate : 0.8008
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2872
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7435
##             Specificity : 0.6625
##          Pos Pred Value : 0.3540
##          Neg Pred Value : 0.9121
##              Prevalence : 0.1992
##          Detection Rate : 0.1481
##    Detection Prevalence : 0.4184
##       Balanced Accuracy : 0.7030
##
##        'Positive' Class : 0
##
```

The precision of this model is 67%. We will neural network model to improve the accuracy.

The second model is Neural Network. It's a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. W'll use nnet package to process

```
#Model 2 : Neural Network

nn_model <- multinom(status~tenure_month+balance_amnt+rev_o_tot_amnt+rev_voix_pyg_amnt+
                       rev_sms_pyg_amnt+rev_data_pyg_amnt+rev_mms_pyg_amnt+rev_subs_amnt
                     +NORTH+EAST+WEST+SOUTH+`CAPITAL CITY`+`BUSINESS CITY`+`NORTH WEST`+`SOUTH EAST`, da
```

```
## # weights:  18 (17 variable)
## initial  value 304985.452595
## iter  10 value 217030.037923
## iter  20 value 196309.354339
## iter  30 value 193184.723831
## iter  30 value 193184.722518
## iter  30 value 193184.721994
## final  value 193184.721994
## converged
```

```
summary(nn_model) #summary of the model
```

```
## Call:
## multinom(formula = status ~ tenure_month + balance_amnt + rev_o_tot_amnt +
```

```
##     rev_voix_pyg_amnt + rev_sms_pyg_amnt + rev_data_pyg_amnt +
##     rev_mms_pyg_amnt + rev_subs_amnt + NORTH + EAST + WEST +
##     SOUTH + `CAPITAL CITY` + `BUSINESS CITY` + `NORTH WEST` +
##     `SOUTH EAST`, data = train_set)
##
## Coefficients:
##                          Values    Std. Err.
## (Intercept)       -7.734021e-01 5.670249e-03
## tenure_month       1.488083e-02 1.401643e-04
## balance_amnt      -9.684995e-07 2.767633e-06
## rev_o_tot_amnt     1.180523e-02 9.292505e-04
## rev_voix_pyg_amnt -1.168969e-02 9.295566e-04
## rev_sms_pyg_amnt  -1.078786e-02 9.504822e-04
## rev_data_pyg_amnt -1.058063e-02 9.344343e-04
## rev_mms_pyg_amnt  -1.172549e-02 9.309179e-04
## rev_subs_amnt     -1.122977e-02 9.301272e-04
## NORTH              1.431228e+00 1.123608e-02
## EAST               1.344051e+00 1.340747e-02
## WEST               1.320224e+00 1.168775e-02
## SOUTH              1.487757e+00 1.133498e-02
## `CAPITAL CITY`     7.974993e-01 8.007253e-03
## `BUSINESS CITY`    1.158689e+00 1.072261e-02
## `NORTH WEST`       1.239789e+00 1.038939e-02
## `SOUTH EAST`       1.297534e+00 1.066968e-02
##
## Residual Deviance: 386369.4
## AIC: 386403.4
```

```r
#Model prediction
nn_prediction <- predict(nn_model,test_set) #Prediction using Neural model in conjunction with the test
prediction_table <- table(nn_prediction, test_set$status) #Put information into confusion matrix
prediction_table #print confusion matrix
```

```
##
## nn_prediction     0     1
##             0     2     2
##             1 21909 88086
```

```r
#correct classification
sum(diag(prediction_table))/sum(prediction_table)
```

```
## [1] 0.8008073
```

```r
#Missclassification Rate
1-sum(diag(prediction_table))/sum(prediction_table)
```

```
## [1] 0.1991927
```

The precision of this model is 80%. We will try decision tree model to improve the accuracy.

The Next model we try is Decision tree. it is a classification method that uses tree-like models of decisions and their possible outcomes. This method is one of the most commonly used tools in machine learning analysis. We will use the rpart library in order to use recursive partitioning methods for decision trees. This exploratory method will identify the most important variables related to churn in a hierarchical format.

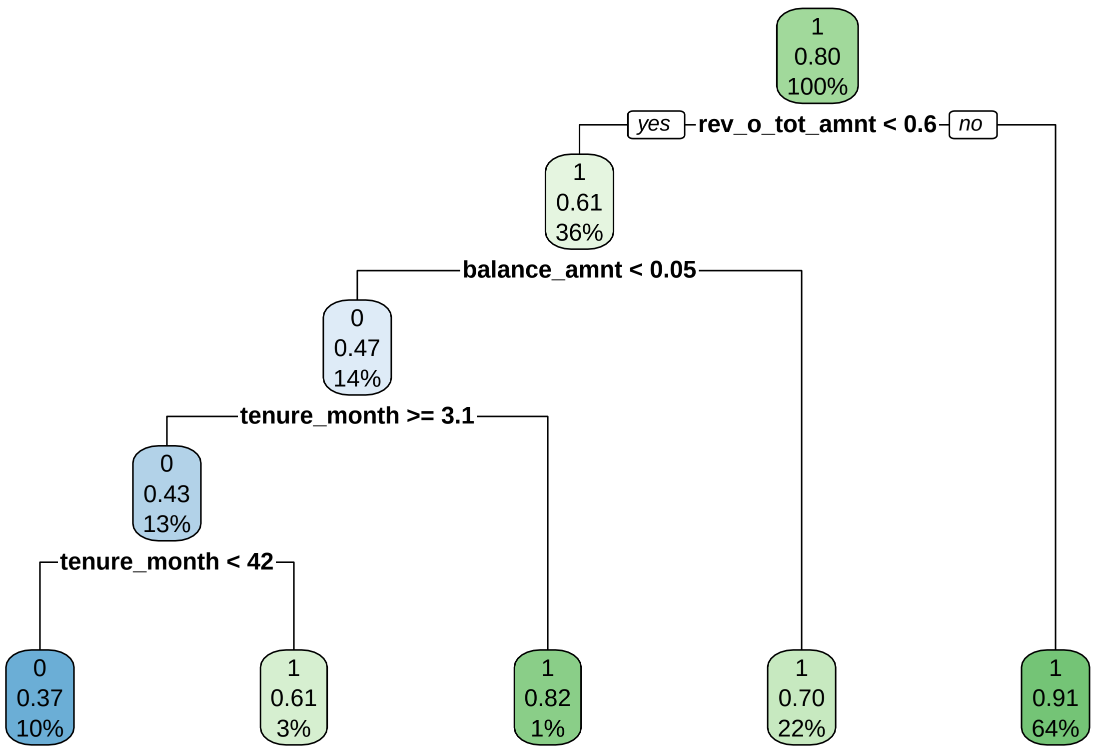
```r
###Model 3 : Decision Trees ###

#Data partition

split_set <- sample(2,nrow(input_data),replace=TRUE,prob = c(0.80,0.20))

train_set <- input_data[split_set==1,]

test_set <- input_data[split_set==2,]

dtree_model <- rpart(status~tenure_month+balance_amnt+rev_o_tot_amnt+rev_voix_pyg_amnt+
                       rev_sms_pyg_amnt+rev_data_pyg_amnt+rev_mms_pyg_amnt+rev_subs_amnt
                     +NORTH+EAST+WEST+SOUTH+`CAPITAL CITY`+`BUSINESS CITY`+`NORTH WEST`+`SOUTH EAST`,da
summary(dtree_model)
```

```
## Call:
## rpart(formula = status ~ tenure_month + balance_amnt + rev_o_tot_amnt +
##     rev_voix_pyg_amnt + rev_sms_pyg_amnt + rev_data_pyg_amnt +
##     rev_mms_pyg_amnt + rev_subs_amnt + NORTH + EAST + WEST +
##     SOUTH + `CAPITAL CITY` + `BUSINESS CITY` + `NORTH WEST` +
##     `SOUTH EAST`, data = train_set)
##   n= 439658
##
##           CP nsplit rel error   xerror        xstd
## 1 0.02444579      0 1.0000000 1.000000 0.003023371
## 2 0.01000000      4 0.8728225 0.872891 0.002869017
##
## Variable importance
##    rev_o_tot_amnt     rev_subs_amnt rev_voix_pyg_amnt      balance_amnt
##                37                30                13                11
##      tenure_month
##                 9
##
## Node number 1: 439658 observations,    complexity param=0.02444579
##   predicted class=1  expected loss=0.1992503  P(node) =1
##     class counts: 87602 352056
##    probabilities: 0.199 0.801
##   left son=2 (158173 obs) right son=3 (281485 obs)
##   Primary splits:
##       rev_o_tot_amnt    < 0.6      to the left,  improve=18919.350, (0 missing)
##       rev_subs_amnt     < 10.5     to the left,  improve=17828.270, (0 missing)
##       balance_amnt      < 0.05     to the left,  improve= 9480.248, (0 missing)
##       rev_voix_pyg_amnt < 0.05     to the left,  improve= 9381.984, (0 missing)
##       tenure_month      < 41.35    to the left,  improve= 4668.182, (0 missing)
##   Surrogate splits:
##       rev_subs_amnt     < 10.5     to the left,  agree=0.926, adj=0.795, (0 split)
##       rev_voix_pyg_amnt < 0.05     to the left,  agree=0.768, adj=0.356, (0 split)
##       balance_amnt      < 0.05     to the left,  agree=0.669, adj=0.080, (0 split)
##       tenure_month      < 3.116667 to the left,  agree=0.664, adj=0.067, (0 split)
##       BUSINESS CITY     < 0.5      to the right, agree=0.641, adj=0.002, (0 split)
##
## Node number 2: 158173 observations,    complexity param=0.02444579
##   predicted class=1  expected loss=0.3949283  P(node) =0.3597637
```

```
##     class counts: 62467 95706
##    probabilities: 0.395 0.605
##   left son=4 (63101 obs) right son=5 (95072 obs)
##   Primary splits:
##       balance_amnt < 0.05     to the left,  improve=4057.2690, (0 missing)
##       tenure_month < 41.71667 to the left,  improve=2134.9390, (0 missing)
##       CAPITAL CITY < 0.5      to the right, improve=1401.1150, (0 missing)
##       NORTH        < 0.5      to the left,  improve= 444.2087, (0 missing)
##       SOUTH        < 0.5      to the left,  improve= 426.2730, (0 missing)
##   Surrogate splits:
##       tenure_month < 7.05     to the left,  agree=0.643, adj=0.105, (0 split)
##
## Node number 3: 281485 observations
##   predicted class=1  expected loss=0.08929428  P(node) =0.6402363
##     class counts: 25135 256350
##    probabilities: 0.089 0.911
##
## Node number 4: 63101 observations,    complexity param=0.02444579
##   predicted class=0  expected loss=0.4660623  P(node) =0.1435229
##     class counts: 33692 29409
##    probabilities: 0.534 0.466
##   left son=8 (56828 obs) right son=9 (6273 obs)
##   Primary splits:
##       tenure_month  < 3.083333 to the right, improve=1715.62700, (0 missing)
##       CAPITAL CITY  < 0.5      to the right, improve=1190.48800, (0 missing)
##       SOUTH         < 0.5      to the left,  improve=1055.25600, (0 missing)
##       NORTH         < 0.5      to the left,  improve= 366.27350, (0 missing)
##       BUSINESS CITY < 0.5      to the right, improve=  19.22411, (0 missing)
##
## Node number 5: 95072 observations
##   predicted class=1  expected loss=0.3026653  P(node) =0.2162408
##     class counts: 28775 66297
##    probabilities: 0.303 0.697
##
## Node number 8: 56828 observations,    complexity param=0.02444579
##   predicted class=0  expected loss=0.4273246  P(node) =0.129255
##     class counts: 32544 24284
##    probabilities: 0.573 0.427
##   left son=16 (44069 obs) right son=17 (12759 obs)
##   Primary splits:
##       tenure_month < 41.68333 to the left,  improve=1133.23500, (0 missing)
##       CAPITAL CITY < 0.5      to the right, improve= 875.94030, (0 missing)
##       SOUTH        < 0.5      to the left,  improve= 508.79060, (0 missing)
##       NORTH        < 0.5      to the left,  improve= 155.30510, (0 missing)
##       WEST         < 0.5      to the left,  improve=  28.77837, (0 missing)
##   Surrogate splits:
##       balance_amnt < -21.9    to the right, agree=0.775, adj=0, (0 split)
##
## Node number 9: 6273 observations
##   predicted class=1  expected loss=0.1830065  P(node) =0.01426791
##     class counts:  1148  5125
##    probabilities: 0.183 0.817
##
## Node number 16: 44069 observations
```

```
##   predicted class=0  expected loss=0.373596  P(node) =0.1002347
##     class counts: 27605 16464
##    probabilities: 0.626 0.374
##
## Node number 17: 12759 observations
##   predicted class=1  expected loss=0.3870993  P(node) =0.02902028
##     class counts:  4939  7820
##    probabilities: 0.387 0.613
```

```
#To plot the decision tree
rpart.plot(dtree_model)
```




```
#churn prediction
dtree_prediction <- predict(dtree_model,train_set,type="class")

#confusion matrix
confusionMatrix(dtree_prediction,train_set$status)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0  27605  16464
##          1  59997 335592
##
```

```
##                Accuracy : 0.8261
##                  95% CI : (0.825, 0.8272)
##     No Information Rate : 0.8007
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3299
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.31512
##             Specificity : 0.95323
##          Pos Pred Value : 0.62640
##          Neg Pred Value : 0.84834
##              Prevalence : 0.19925
##          Detection Rate : 0.06279
##    Detection Prevalence : 0.10023
##       Balanced Accuracy : 0.63418
##
##        'Positive' Class : 0
##
```

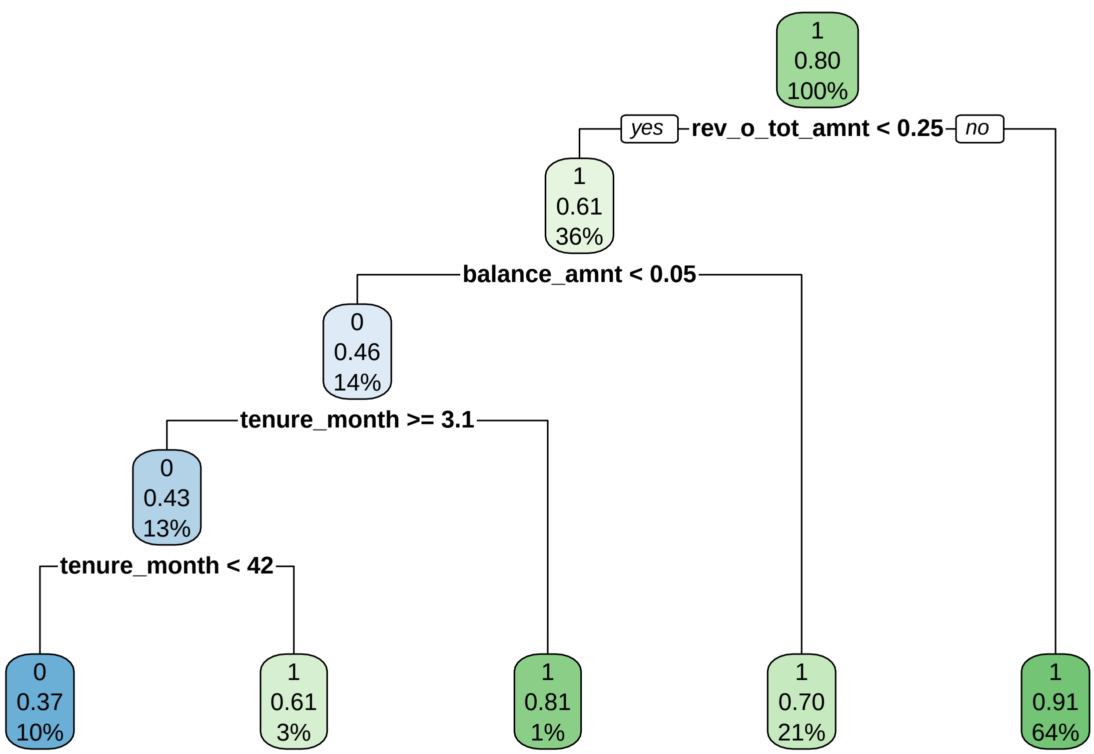
```
### test_set

dtree_model <- rpart(status~.,data = test_set)
summary(dtree_model)
```

```
## Call:
## rpart(formula = status ~ ., data = test_set)
##   n= 110342
##
##           CP nsplit rel error  xerror        xstd
## 1 0.02568774      0 1.0000000 1.00000 0.006040108
## 2 0.01000000      4 0.8720168 0.87229 0.005730032
##
## Variable importance
##    rev_o_tot_amnt     rev_subs_amnt rev_voix_pyg_amnt      balance_amnt
##                37                30                13                11
##      tenure_month
##                 9
##
## Node number 1: 110342 observations,    complexity param=0.02568774
##   predicted class=1  expected loss=0.1989813  P(node) =1
##     class counts: 21956 88386
##    probabilities: 0.199 0.801
##   left son=2 (39535 obs) right son=3 (70807 obs)
##   Primary splits:
##       rev_o_tot_amnt    < 0.25     to the left,  improve=4665.897, (0 missing)
##       rev_subs_amnt     < 10.5     to the left,  improve=4365.475, (0 missing)
##       balance_amnt      < 0.05     to the left,  improve=2408.497, (0 missing)
##       rev_voix_pyg_amnt < 0.05     to the left,  improve=2281.020, (0 missing)
##       tenure_month      < 41.65    to the left,  improve=1153.426, (0 missing)
##   Surrogate splits:
##       rev_subs_amnt     < 10.5     to the left,  agree=0.926, adj=0.795, (0 split)
##       rev_voix_pyg_amnt < 0.05     to the left,  agree=0.766, adj=0.348, (0 split)
```

```
##       balance_amnt      < 0.05     to the left,  agree=0.670, adj=0.079, (0 split)
##       tenure_month      < 3.116667 to the left,  agree=0.666, adj=0.068, (0 split)
##       custumer_id       < 92996980 to the right, agree=0.642, adj=0.000, (0 split)
##
## Node number 2: 39535 observations,    complexity param=0.02568774
##   predicted class=1  expected loss=0.3935753  P(node) =0.3582951
##     class counts: 15560 23975
##    probabilities: 0.394 0.606
##   left son=4 (15854 obs) right son=5 (23681 obs)
##   Primary splits:
##       balance_amnt < 0.05     to the left,  improve=1067.39000, (0 missing)
##       tenure_month < 41.01667 to the left,  improve= 513.49540, (0 missing)
##       CAPITAL CITY < 0.5      to the right, improve= 369.79610, (0 missing)
##       NORTH        < 0.5      to the left,  improve= 122.50130, (0 missing)
##       SOUTH        < 0.5      to the left,  improve=  96.20291, (0 missing)
##   Surrogate splits:
##       tenure_month < 7.05     to the left,  agree=0.639, adj=0.099, (0 split)
##       custumer_id  < 70003860 to the left,  agree=0.599, adj=0.000, (0 split)
##
## Node number 3: 70807 observations
##   predicted class=1  expected loss=0.09033005  P(node) =0.6417049
##     class counts:  6396 64411
##    probabilities: 0.090 0.910
##
## Node number 4: 15854 observations,    complexity param=0.02568774
##   predicted class=0  expected loss=0.4644254  P(node) =0.1436806
##     class counts:  8491  7363
##    probabilities: 0.536 0.464
##   left son=8 (14272 obs) right son=9 (1582 obs)
##   Primary splits:
##       tenure_month  < 3.083333 to the right, improve=419.089300, (0 missing)
##       CAPITAL CITY  < 0.5      to the right, improve=321.415800, (0 missing)
##       SOUTH         < 0.5      to the left,  improve=247.533800, (0 missing)
##       NORTH         < 0.5      to the left,  improve=117.949900, (0 missing)
##       BUSINESS CITY < 0.5      to the right, improve=  4.971212, (0 missing)
##
## Node number 5: 23681 observations
##   predicted class=1  expected loss=0.2985094  P(node) =0.2146146
##     class counts:  7069 16612
##    probabilities: 0.299 0.701
##
## Node number 8: 14272 observations,    complexity param=0.02568774
##   predicted class=0  expected loss=0.4261491  P(node) =0.1293433
##     class counts:  8190  6082
##    probabilities: 0.574 0.426
##   left son=16 (11128 obs) right son=17 (3144 obs)
##   Primary splits:
##       tenure_month < 41.7     to the left,  improve=277.479700, (0 missing)
##       CAPITAL CITY < 0.5      to the right, improve=245.337100, (0 missing)
##       SOUTH        < 0.5      to the left,  improve=108.150600, (0 missing)
##       NORTH        < 0.5      to the left,  improve= 64.085610, (0 missing)
##       WEST         < 0.5      to the left,  improve=  7.700228, (0 missing)
##   Surrogate splits:
##       custumer_id  < 92996160 to the left,  agree=0.78, adj=0.001, (0 split)
```

```
##       balance_amnt < -5.25    to the right, agree=0.78, adj=0.001, (0 split)
##
## Node number 9: 1582 observations
##   predicted class=1  expected loss=0.1902655  P(node) =0.01433724
##     class counts:   301  1281
##    probabilities: 0.190 0.810
##
## Node number 16: 11128 observations
##   predicted class=0  expected loss=0.3737419  P(node) =0.1008501
##     class counts:  6969  4159
##    probabilities: 0.626 0.374
##
## Node number 17: 3144 observations
##   predicted class=1  expected loss=0.3883588  P(node) =0.02849323
##     class counts:  1221  1923
##    probabilities: 0.388 0.612
```

```r
#To plot the decision tree
rpart.plot(dtree_model)
```




```r
#churn prediction
dtree_prediction <- predict(dtree_model,test_set,type="class")


#Missclassification Rate
confusionMatrix(dtree_prediction,test_set$status)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0  6969  4159
##          1 14987 84227
##
##                Accuracy : 0.8265
##                  95% CI : (0.8242, 0.8287)
##     No Information Rate : 0.801
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3319
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.31741
##             Specificity : 0.95295
##          Pos Pred Value : 0.62626
##          Neg Pred Value : 0.84894
##              Prevalence : 0.19898
##          Detection Rate : 0.06316
##    Detection Prevalence : 0.10085
##       Balanced Accuracy : 0.63518
##
##        'Positive' Class : 0
##
```

The precision of this model is 82%. we improve the accuracy.

Results

In this project, We try different models :

- logistic regression with precision of 67%
- Neural network with 80% accuracy
- Decision tree with 82% accuracy

Conclusion

Decision tree seems to be the best model for churn prediction. Although it is possible to improve the model by enriching the data with the characteristics of the customers such as gender, age of the customer… All this information is missing in the data repository. It will be our next challenge