RankMiner: Predicting Phone Agent Attrition

Final Project

Statistical Data Mining, ISM 6137.902S16

April, 2016

Authored by **Vishal Punjabi**

Deepen Panchal

Mohammed Ihtesham

Paridhi Varma

RANK MINER

PREDICTIVE ANALYTICS

Table of Contents

Confidentiality Agreement

This report is classified as internal and confidential under the following agreement. Circulation of this report is restricted only to internal stakeholders within the University of South Florida and RankMiner Co.

1 INTRODUCTION

1.1 MOTIVATION

The company RankMiner is a Predictive Analytics Company focused on voice based insights. Their focus area is mainly telephony based audio and the problem statement revolves around the Agent Attrition rate for a debt collection company.

1.2 PROBLEM STATEMENT

The problem focuses on the agent attrition based on the audio features of the calls made by each agent throughout the period of July – December. The prediction of agent attrition on the basis of the call features is the challenging aspect of the problem. The data provided to us is Agent details such as the commission earned per month, number

The target of the problem statement is the Phone Agents and to reduce their attrition rate, report performance of the data model, identify strength and weaknesses and to suggest further research to understand problem statement.

1.3 SCOPE

Three data sets are used in this project. Data for **298** agents with **43** attributes are provided for building this model. Information of all the calls made by each of the agents are provided in a separate data set. There are **170429** call details in the data set. Call data set have ten attributes which gives more information of the call such as duration, start time, end time, call outcome etc. In addition, a third set is provided which contains further information of each call made and captures the agents behavior on that call. A total of **62245** rows of information is provided with **178** attributes is available to analyze the characteristic of each call made by an agent.

2 DATA EXPLORTION

There are 3 datasets available from the RankMiner project – Agent Data, Call Data and Feature Dataset. This is the initial structure of the dataset that was identified.

1. Call Data

Account	Unique Customer Account Id
Audio File Name	Name Of The File Name
Skill Name	The Skill The Agent Is Currently Working In
Call Start Time	Call Start Time
Call End Time	Call End Time
Agent Id	Unique Agent Id
Call Direction	Inbound Or Outbound Call (All Calls are Outbound in your DS)
Call Duration (HMS)	Duration Of Call
Filesize(kb)	Size Of The Call Audio File
Rec Status	Exit Status For The Call

2. Agent Data

Agent_id	Unique ID Of The Agent
Payroll_id_src1,payroll_id_src2	Payroll Info Sourced From Different Systems Of The Enterprise
Hire_date_src1,hire_date_src2	Hire Date Info Sourced From Different Systems Of The Enterprise
Term_date_src1,term_date_src2	Term Date Info Sourced From Different Systems Of The Enterprise
Group_src1	Group The Agent Has Worked In Primarily
Work_shift_src1,work_shift_src2	Work Shift Info Sourced From Different Systems Of The Enterprise

Term_code	Term Type + Term Reason
Term_type	I/V
Term_reason	Reason For Termination
Jul_group	Group The Agent Worked In For The Month
Jul_hrs_worked	Hours Worked In The Month Of July
Jul_hourly_rate	Hourly Rate Of Agent That Month
Jul_revenue_generated	Revenue Of Each Agent Month Wise
Jul_commission	Commission Of Each Agent Month Wise

3. Feature Data

Audio File Name	**Name Of The Audio File**
Target_value	The Value If 1, The Agent Is Likely To Stay And If 0, More Likely To Leave
Feature_value 1-176	Features Recorded As The Numeric Values For Recording Emotional Status Of The Call Of The Account

2.1 GENERAL ASSESSMENT OF DATA QUALITY

The data in the three files was of mediocre quality. While there weren't any grave quality issues as such, there were some small to mediocre issues which needed to be addressed. Some of which are as below:

1. Missing/blank values
2. Outliers
3. Shifted Data

2.2 IDENTIFICATION OF DATA QUALITY ISSUES

1. **Agent Data set:**
   i. **Empty values in the data set:** The data for columns payroll_id, hire_date, term_date, work_shift are pulled from two different sources. There is a possible discrepancy between the two sources. For a single row, source values contain value while source 2 is empty.
   ii. Data present in the dataset are in mixed case (Upper and Lower). Though the text is same for most of data but due to mixed case they may cause issues during statistical analysis.
   iii. Dec_group columns has an outlier "102474" and is removed from the dataset.
   iv. There are significant number of Work groups fields which have value "#N/A" for agents and months.
   v. A single work group has different code and description. For example, "INB DSH" and "DSH" refer to the same DISH group. To identify groups which mean the same the values for these groups are updated to a more meaningful and uniform codes.
   vi. The data type of revenue generated for some months are factors and not numeric.
2. **Feature Data set:**
   i. Audio file name has an outlier with value "62243" and is removed from data set.
   ii. Target value of some audio files are blank.
3. **Call Data set:**
   i. A lot of fields in the data set are blanks. All these blanks are replaced by NA.
   ii. Text fields are in mixed case. Though the text is same for most of data but due to mixed case they are counted as distinct by the software. Thus, to reduce the count, all the text data is changed to upper case.
   iii. Columns in the data set are dot separated and thus pose a risk of being interpreted as a function when running R code.
   iv. The Skill name column has a number of text description which refer to the same skill group. For example – "AT and T_HCI", "AT and T B_HCI" and "TMobile_HCI" , Tmobile Tiertary A_HCI refer the same group. To identify groups which refer the same entity are updated with a more meaningful and uniform codes.

2.3 GENERAL STRATEGY FOR HANDLING DATA QUALITY ISSUES

The first step performed to handle data quality was a complete analysis of the current state of all three data sets. Information with errors, inconsistencies, duplicates or missing fields are updated with values that do not change the context of the information.

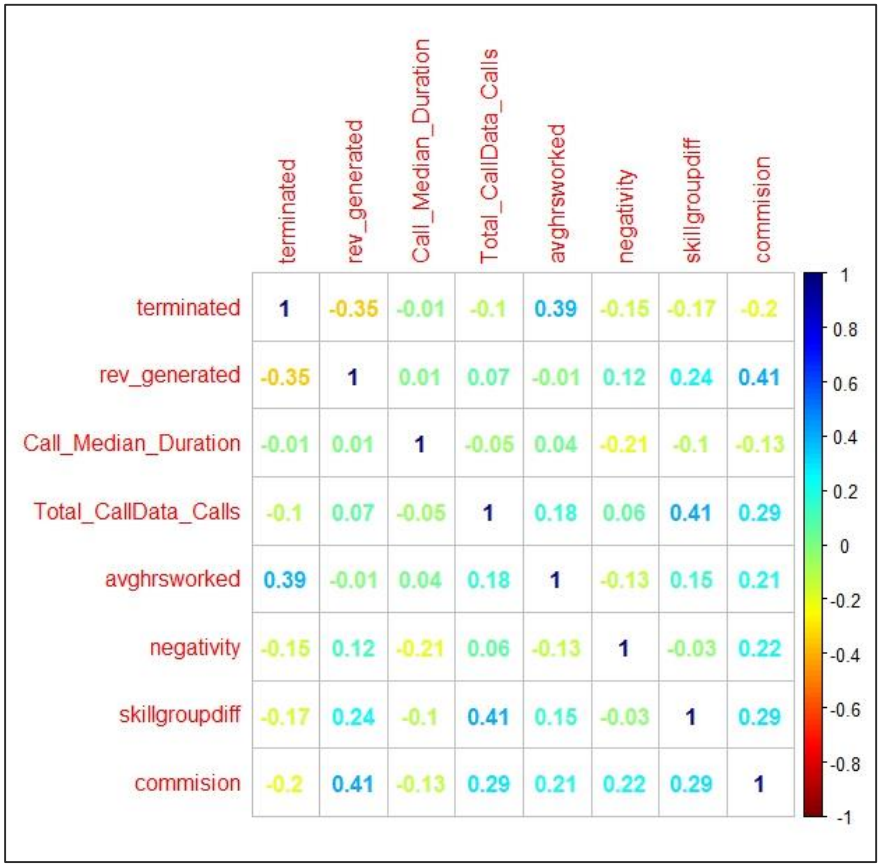
The assumptions made to tackle data issue for each data set are as follows:

i. The columns for which data is coming from two different data sources, source 1 is assumed to be authentic and wherever there is discrepancy, the values are overwritten with values from source 1.
ii. Text data present in all data sets such as skill names, skill group etc are in mixed case. Thus to reduce inconsistencies all textual data are converted to upper case.
iii. The work groups having values as "#N/A" are replaced with "OTHER" work group to handle missing values and not lost agents from analysis.
iv. Groups and sub groups under whom the agent is working has inconsistent values such as "INB DSH","DSH" – which mean DISH,"PTM","VZ" – stands for VERIZON,"INB SPR","SPR" – is same as SPRINT. Inconsistencies like this are handled by replacing it with a single value for same sub groups but different codes.
v. Dots have been removed from column names with underscore so that it does not give errors while running R code.
vi. Skill name in call data set are inconsistent and similarly as for agent data set these values can be grouped together and assigned a more meaningful name.

2.4 ADVANCED DATA EXPLORATION: VISUALIZATION OF RELATIONSHIPS

2.4.1 VISUALIZATION 1

Below is a correlation matrix, this matrix shows the relationship between every predictor identified for this project. The matrix presents a numerical relationship between them thus it can be easily identified which predictors have a direct and which one have indirect relationship among them.

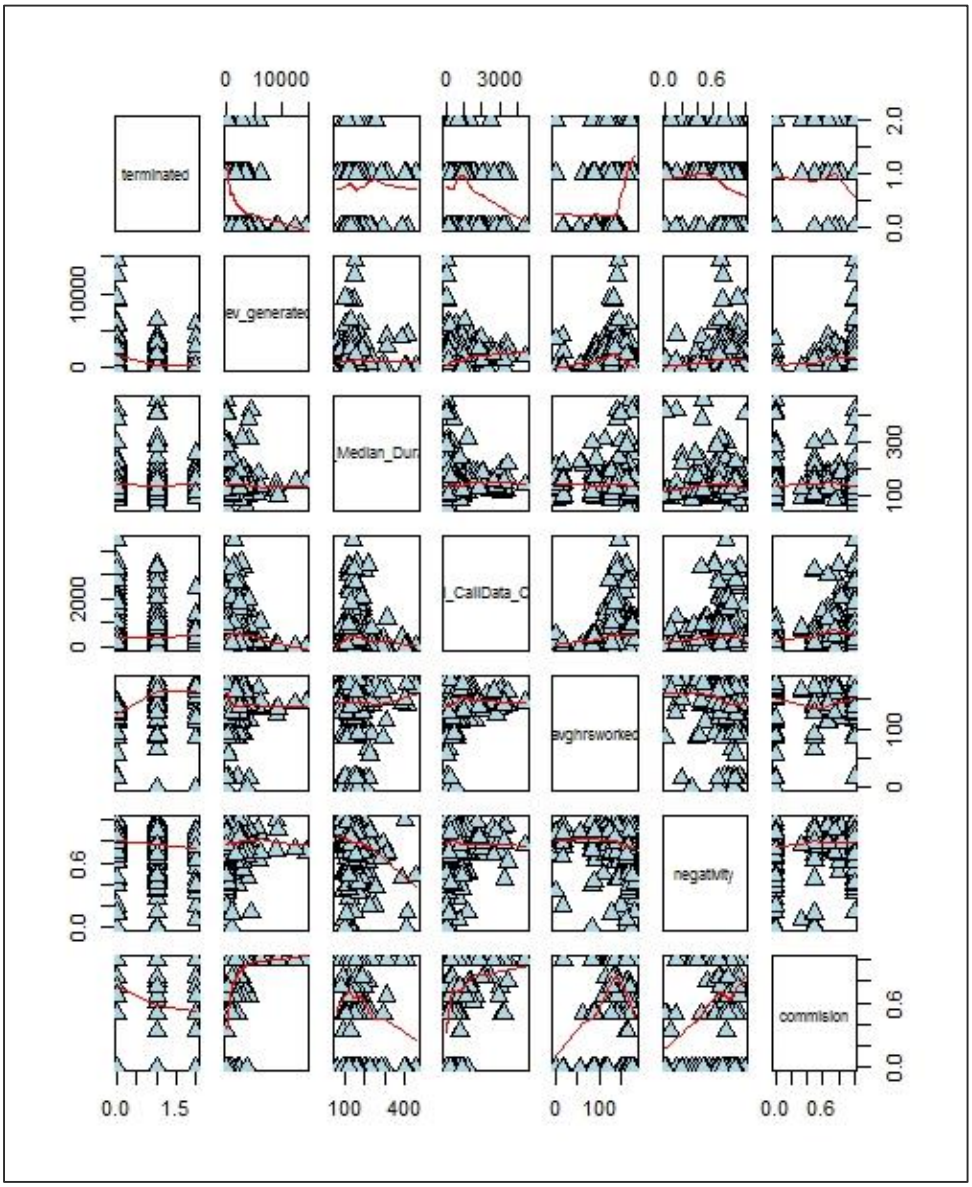


R Code:

```
#correlation matrix
cor(agent_trimmed2[,c("terminated",
"rev_generated","Call_Median_Duration","Total_CallData_Calls","avghrsworked",
"negativity","skillgroupdiff","commision")]);
##visualizing the correlation matrix
library(corrplot);
cor.agent_trimmed2 <- cor(agent_trimmed2[,c("terminated",
"rev_generated","Call_Median_Duration","Total_CallData_Calls","avghrsworked",
"negativity","skillgroupdiff","commision")]);
col1 <-colorRampPalette(c("#7F0000","red","#FF7F00","yellow","#7FFF7F",
"cyan", "#007FFF", "blue","#00007F"))
corrplot(cor.agent_trimmed2, method = "number", col = col1(100))
```

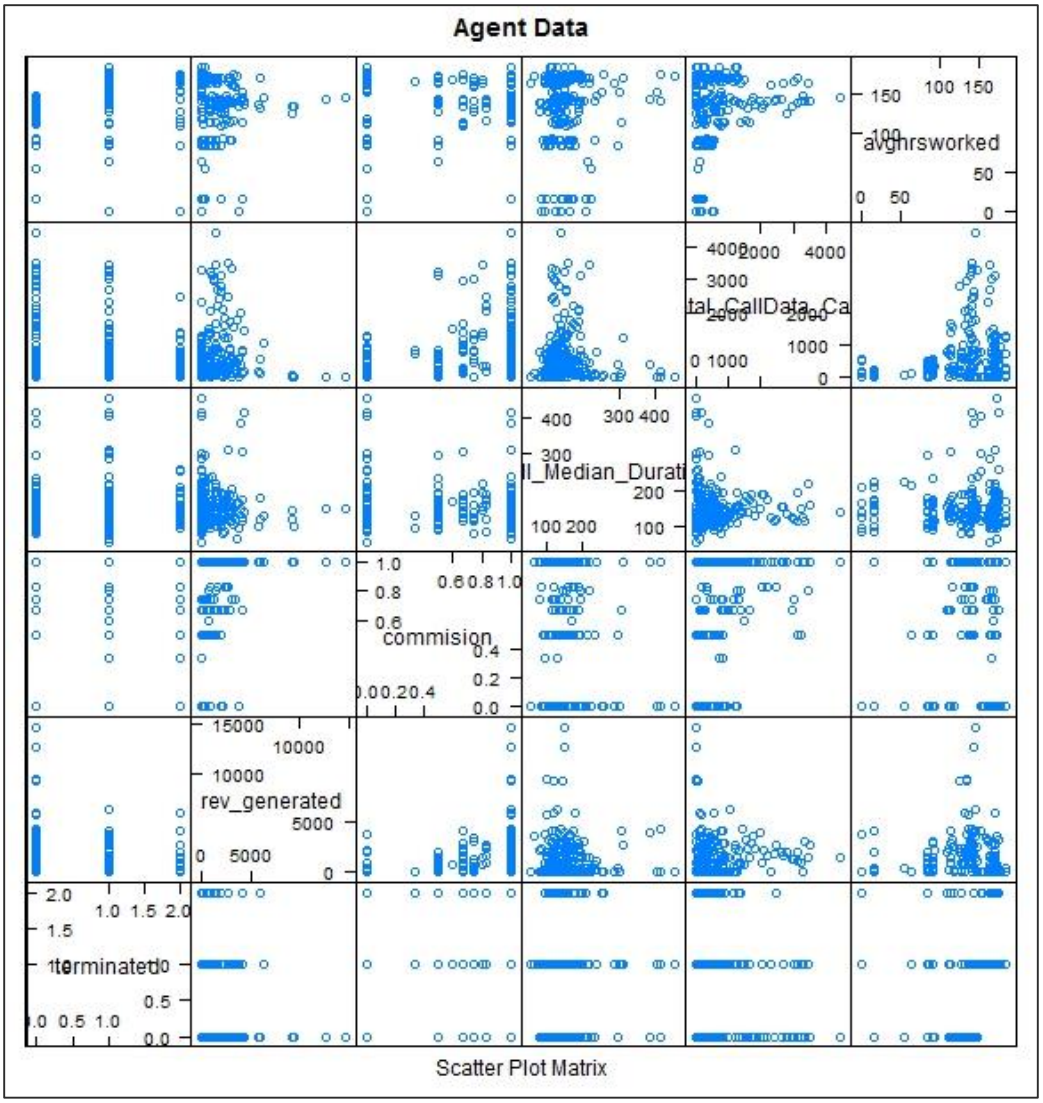
2.4.2 Visualization 2:

Another way to visualize correlation matrix is by running a scatterplot matrix.



R Code:

```
library(YaleToolkit);
pairs(agent_trimmed2[,c("terminated",
"rev_generated","Call_Median_Duration","Total_CallData_Calls","avghrsworked",
"negativity")], upper.pars=list(scatter="stats"),panel = panel.smooth, cex =
1.5, pch = 24, bg = "light blue");
```

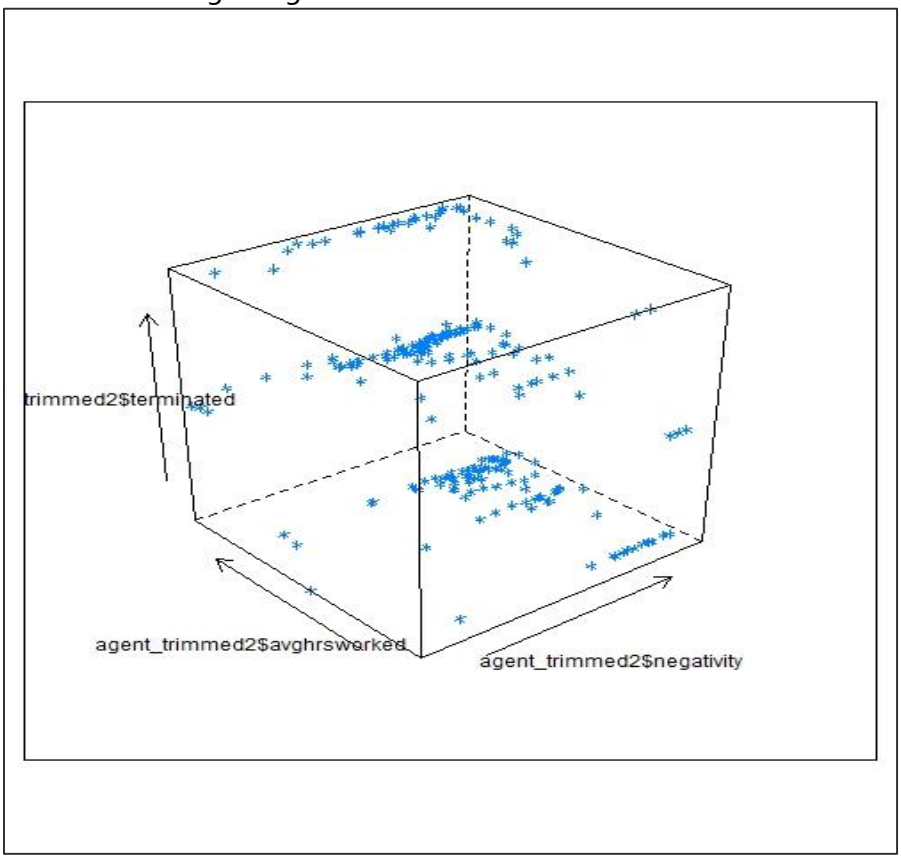


R Code:

```
splom(agent_trimmed2[,c("terminated",
"rev_generated","Call_Median_Duration","Total_CallData_Calls","avghrsworked",
"commision")], main="Agent Data")
```

2.4.3 Visualization 3:

The below plot shows that agents who work more average number of hours are more probable to left voluntarily. An important insight also from the below plot shows that agents who are working long hours are having a negative outcome from calls.

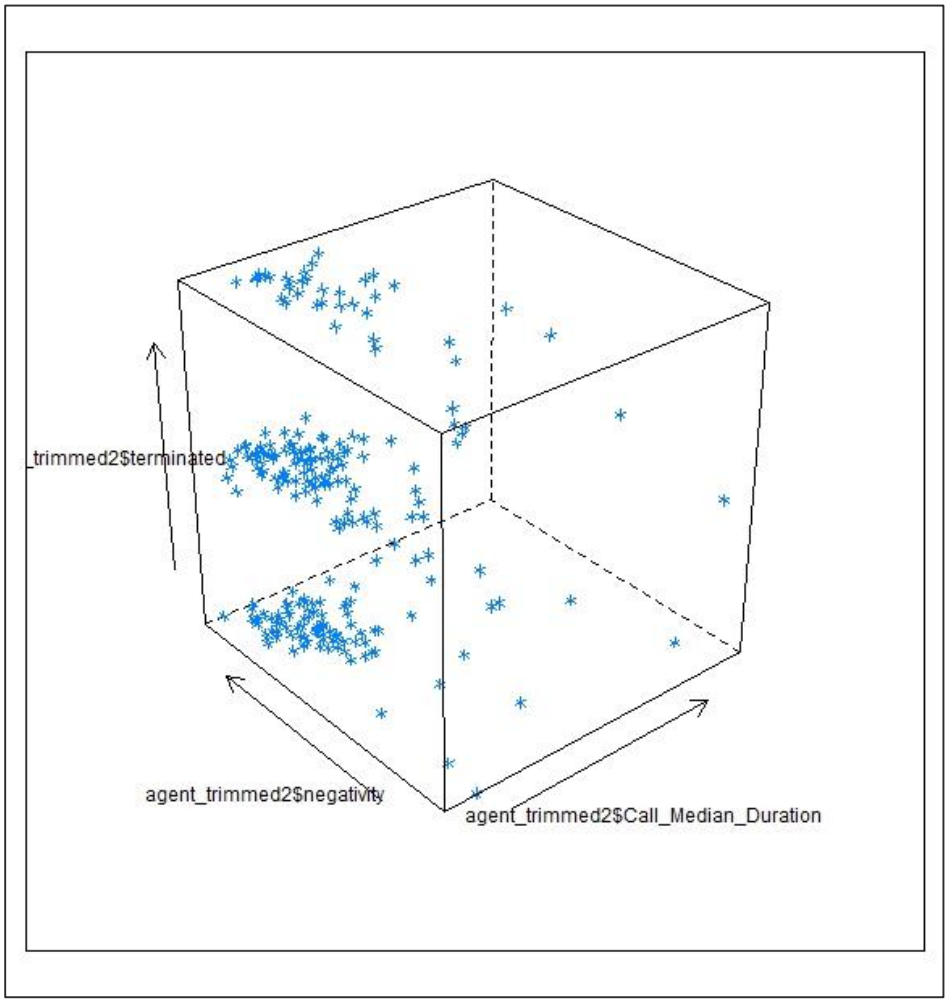


R Code:

```
library(lattice)
cloud(agent_trimmed2$terminated~agent_trimmed2$negativity+agent_trimmed2$avgh
rsworked)
cor(agent_trimmed2[,c("terminated", "negativity","avghrsworked")]);
```

2.4.4 Visualization 4:

The plot depicts that agents who exhibited negative emotions on calls were either working or left voluntarily.

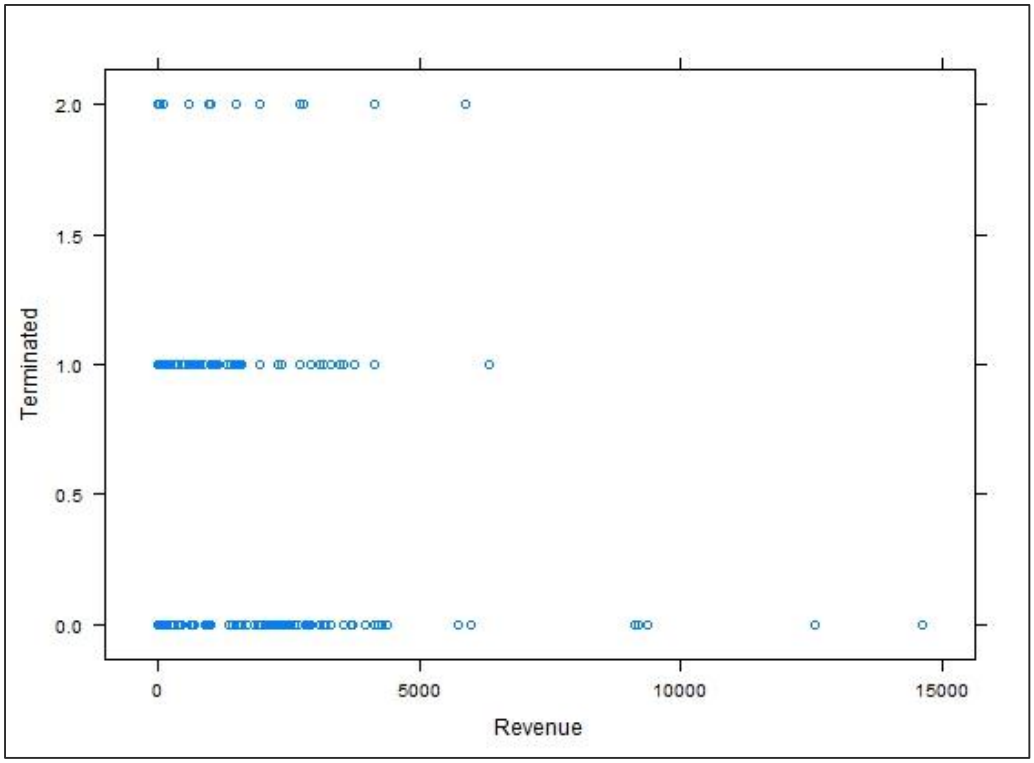


R Code:

```
cloud(agent_trimmed2$terminated~agent_trimmed2$Call_Median_Duration+agent_tri
mmed2$negativity)
```

2.4.5 Visualization 5:

This plot establishes relationship between agents who were terminated whether voluntarily or involuntarily and the revenue generated by them. From the graph, it can be confirmed that agents who left the company voluntarily were a group who were generating lesser revenue as compared to the agents who did not leave the company.



R Code:

```
xyplot(agent_trimmed2$terminated~agent_trimmed2$rev_generated, xlab =
"Revenue", ylab = "Terminated");
```

3 DATA PREPARATION AND MATCHING

3.1 AGENT DATA AND CALL DATA

1. Payroll_ID is dropped from the dataset, as the agent_id identifies the agents uniquely.
2. Src1 and src2 columns are same, and are included to establish the veracity for data. The columns have been merged for the easy handling of data. If there is a contradiction, we picked src1 data.
    a. The shifts provided is considered irrelevant for this statistical analysis.
    b. Also, the number of hours worked is provided for each of the agents. Hence, the column is termed irrelevant for the final dataset.

```
agent$payroll_id_src1[is.na(agent$payroll_id_src1)] <-
agent$payroll_id_src2[is.na(agent$payroll_id_src1)]
agent$payroll_id_src2[is.na(agent$payroll_id_src2)] <-
agent$payroll_id_src1[is.na(agent$payroll_id_src2)]

agent$hire_date_src1[is.na(agent$hire_date_src1)] <-
agent$hire_date_src2[is.na(agent$hire_date_src1)]
agent$hire_date_src2[is.na(agent$hire_date_src2)] <-
agent$hire_date_src1[is.na(agent$hire_date_src2)]

agent$term_date_src1[is.na(agent$term_date_src2)] <-
agent$term_date_src1[is.na(agent$term_date_src2)]
agent$term_date_src1[is.na(agent$term_date_src1)] <-
agent$term_date_src2[is.na(agent$term_date_src1)]

agent$work_shift_src1[is.na(agent$work_shift_src1)] <-
agent$work_shift_src2[is.na(agent$work_shift_src1)]
agent$work_shift_src2[is.na(agent$work_shift_src2)] <-
agent$work_shift_src1[is.na(agent$work_shift_src2)]

# Check if src1 and src2 are different
agent$payroll_id_src1[agent$payroll_id_src1!=agent$payroll_id_src2]
agent$payroll_id_src2[agent$payroll_id_src1!=agent$payroll_id_src2]

agent$hire_date_src1[agent$hire_date_src1!=agent$hire_date_src2 ]
agent$hire_date_src2[agent$hire_date_src1!=agent$hire_date_src2]

agent$term_date_src1[agent$term_date_src1!=agent$term_date_src2
& !is.na(agent$term_date_src1) & !is.na(agent$term_date_src2)]
agent$term_date_src2[agent$term_date_src1!=agent$term_date_src2
& !is.na(agent$term_date_src1) & !is.na(agent$term_date_src2)]

agent$work_shift_src1[agent$work_shift_src1!=agent$work_shift_src2
& !is.na(agent$work_shift_src1) & !is.na(agent$work_shift_src2)]
agent$work_shift_src2[agent$work_shift_src1!=agent$work_shift_src2
& !is.na(agent$work_shift_src1) & !is.na(agent$work_shift_src2)]

#Delete irrelevant columns from src2
agent$payroll_id_src2 = NULL
```

```
agent$hire_date_src2 = NULL
agent$term_date_src2 = NULL
agent$work_shift_src2 = NULL
```

3. A predictor for the model is calculated based on the data available for each agent under termination date or termination type. If both values are blank, then that means the agent is working and is not terminated and vice-versa. The agents who are terminated are assigned 1 and those who are working are assigned 0.

```
agent$terminated <- 0
agent$terminated[!is.na(agent$term_date) | !is.na(agent$term_code)
| !is.na(agent$term_type) | !is.na(agent$term_reason)] <- 1
```

4. The columns Term_Type and Term_Reason is also termed irrelevant for the dataset as the same information is provided to the model from the column Term_code.

5. Hours_worked, hourly_rate and group for each agent every month is discarded and taken as a single value, from the mean of all the data points.

6. Revenue Generated, commission of every month is also discarded and calculated as a single dataset to generate two columns –average hours worked and negativity.
   a. Productivity - (Average hours worked/month)/total # of months
   We have considered only those months where data was available.

```
#productivity
agent[,"Dec_hrs_worked"] <- as.numeric(agent[,"Dec_hrs_worked"])
agent$avghrsworked <- NA
agent$avghrsworked <-
rowSums(agent[,c("Jul_hrs_worked","Aug_hrs_worked","Sep_hrs_worked","Oc
t_hrs_worked","Nov_hrs_worked","Dec_hrs_worked")],na.rm = TRUE)
agent$avghrsworked <- agent$avghrsworked/agent$nom

#rev_generated

agent$Oct_revenue_generated[which(agent$Oct_revenue_generated=="#N/A")]
=NA
agent$Sep_revenue_generated[which(agent$Sep_revenue_generated=="#N/A")]
=NA

agent[,"Oct_revenue_generated"] <-
as.numeric(agent[,"Oct_revenue_generated"])
agent[,"Sep_revenue_generated"] <-
as.numeric(agent[,"Sep_revenue_generated"])
agent$rev_generated <- NA
agent$rev_generated <-
rowSums(agent[,c("Jul_revenue_generated","Aug_revenue_generated","Sep_r
evenue_generated","Oct_revenue_generated","Nov_revenue_generated","Dec_
revenue_generated")],na.rm = TRUE)-
rowSums(agent[,c("Jul_commission","Aug_commission","Sep_commission","Oc
t_commission","Nov_commission","Dec_commission")],na.rm = TRUE)
agent$rev_generated <- round(agent$rev_generated/agent$nom,3)
```

```
#Calculation of commission for each month of an agent -
agent$commision <- 0

agent$commision <-
ifelse(!is.na(agent$Jul_commission),agent$commision+1,agent$commision)
agent$commision <-
ifelse(!is.na(agent$Aug_commission),agent$commision+1,agent$commision)
agent$commision <-
ifelse(!is.na(agent$Sep_commission),agent$commision+1,agent$commision)
agent$commision <-
ifelse(!is.na(agent$Oct_commission),agent$commision+1,agent$commision)
agent$commision <-
ifelse(!is.na(agent$Nov_commission),agent$commision+1,agent$commision)
agent$commision <-
ifelse(!is.na(agent$Dec_commission),agent$commision+1,agent$commision)
##0428
agent$commision <- agent$commision/agent$nom
```

The calls are segregated between positive, negative and neutral.

7. The column Call direction is omitted as all the calls are outbound.

```
####Removing Call Direction from the DATASET CALL
call$call_direction = NULL
```

8. No individual has two or more agent_ids, as in case an agent is re-hired; they will receive the same agent_id.
9. Call start time and end time have been removed from the columns as the Call Duration of the agent data calls is more important and pertinent to the conversion rate of calls.
10. Records which have term_code, term_type and term_reason as blanks and have no term_dates indicates that these are those agents which never left the company.
11. Feature_values 1-24 are measurements and statistical calculations on the ***speech of the call*** (total speaking time, etc.)
    feature_values 25-34, 45-54, 65-74, 85-94, 105-114, 125-134, and 145-154 are statistical calculations on speech that has been identified as ***expressing a negative emotion*** (e.g. anger).
    feature_values 35-44, 55-64, 75-84, 95-104, 115-124, 135-144, and 155-164 are statistical calculations on speech that has been identified ***as expressing a positive emotion*** (e.g. joy).
    feature_values 165-176 measure ***proportions of negative/positive emotions***, emotional/unemotional speech, and transitions from one to the other (positive to negative or unemotional to emotional for example).
12. The unit of Call Data is HH:MM:SS
13. Account Number in Data is the Customer Account Number.
14. The agents are assigned to a work group for a month. Since the groups are inconsistent and they refer to the same entity, thus they are made uniform by creating a new categories of groups. Also, these groups are then clubbed together into a functional group as a parent group.
    A ***custom function*** "myreplace" is written to achieve this grouping of work groups.

```
##-------------Data Grouping--------------
```

```
myreplace <- function(v,old,new){
  v= as.character(v)
  if(length(old)==length(new)){
    for(i in 1:length(old))
    {v[which(v==old[i])]= new[i]

    }
  }
  else{
    print("Check Old and New Values - they are unequal")
  }
  v
}
#Creating Vectors for new and old values to categorize the monthly group data
old_values = c("INB DSH" ,"DSH","PTM","VZ","INB
SPR","SPR","G1A","G3A","DTV","COM","INB SPAN","INB A","INB B")
new_values = c("DISH","DISH","TMOBILE","VERIZON","SPRINT","SPRINT","CREDIT
CARD","TELECOM","DIRECTV","COMMERCIAL","INBOUND","INBOUND","INBOUND")

agent[,c("Jul_group","Aug_group","Sep_group","Oct_group","Nov_group","Dec_gro
up")]  =

sapply(agent[,c("Jul_group","Aug_group","Sep_group","Oct_group","Nov_group","
Dec_group")],myreplace,old=old_values,new=new_values)

#Introducing Functional Groupings
old_functional_group =
c("TMOBILE","SPRINT","VERIZON","TELECOM","ATT","DISH","DIRECTV","LEGAL","COMM
ERCIAL","AUTO","CREDIT CARD","INBOUND")

new_functional_group = c("CELL CARRIER","CELL CARRIER","CELL CARRIER","CELL
CARRIER","CELL CARRIER","TV PROVIDER","TV
PROVIDER","LEGAL","COMMERCIAL","AUTO","CREDIT CARD","INBOUND")

agent[c("JulFunctionalGroup","AugFunctionalGroup","SepFunctionalGroup","OctFu
nctionalGroup","NovFunctionalGroup","DecFunctionalGroup")] <- NA
agent[,c("JulFunctionalGroup","AugFunctionalGroup","SepFunctionalGroup","OctF
unctionalGroup","NovFunctionalGroup","DecFunctionalGroup")] =

sapply(agent[,c("Jul_group","Aug_group","Sep_group","Oct_group","Nov_group","
Dec_group")],myreplace,old=old_functional_group,new=new_functional_group)
```

15. A list of groups for which an agent worked is created to measure the number of group changes an agent had during the last six months.

```
#Group Changes
agent$grpchange = NA
for(i in 1:nrow(agent)) {
  if(NA %in% agent$usergroup[[i]]){
    agent$grpchange[i]=length(unique(agent$usergroup[[i]]))-1
  }
  else {  agent$grpchange[i]=length(unique(agent$usergroup[[i]]))}
}
#Creating List of User Groups
for(i in 1:nrow(agent)){

agent$usergroup[i]=list(c(as.character(agent$Jul_group[i]),as.character(agent
```

```
$Aug_group[i]),as.character(agent$Sep_group[i]),as.character(agent$Oct_group[
i]),as.character(agent$Nov_group[i]),as.character(agent$Dec_group[i])))
}
```

16. The group name missing for each of the agent is calculated from the mode of the skill name of the agents.

```
#Calculating Mode of agent skills

getmode <- function(v){
  v = v[!is.na(v)]
  temp <- table(as.vector(v))
  names(temp)[temp == max(temp)]
}
agent$groupmode[log_agent] =
tapply(call$skillgroup,call$agent_id,getmode)
agent$groupmode = as.character(agent$groupmode)

####JUST CHECKING IF "new group" was a character
agent$new_group = as.character(agent$new_group)

##ALL NULL VALUES IN NEW GROUP REPLACED BY THE MODE
agent$new_group[is.na(agent$new_group)]=agent$groupmode
```

17. Calculated Median Call Duration and Total Number of calls for each agent for easy data handling and manipulation

```
#median call duration an agent talks
library(data.table)
cz <- tapply(call$CALL.DURATION.HMS.,call$agent_id, median)
cz = data.frame(names(cz),cz)
names(cz) = c("agent_id","Call_Median_Duration")
agent = (merge(x = agent,y = cz, by = "agent_id", all.x = TRUE))
#no of calls an agent made during the last six months
cy <- data.table(call)
cy <- tapply(call$account, call$agent_id, length)
cy = data.frame(names(cy),cy)
names(cy) = c("agent_id","Total_CallData_Calls")
agent = (merge(x = agent,y = cy, by = "agent_id", all.x = TRUE))
```

18. Rec_Status – The exit status of a call is taken from each of the call in call data

```
#CALL DATA : REC-STATUS
new_statuses = c(0,0,1,0,0,1,1,1,0,1,1,1)

call$negoutcome <- 0
#Run the below command wisely, it takes time
call$negoutcome <-
as.numeric(sapply(call$call_end_status,myreplace,old=unique(call$cal
l_end_status),new=new_statuses))
```

The exit status of a call signifies whether the agent was able to close the call successfully or not. An important predictor called "Negativity" is calculated for each agent. This parameter tells the number of calls the agents closed the call in a non-satisfactory manner.

```
#Calculating Number of Negative calls per agent.
agent$negativecalls = 0
#Agents Not in CALL DATASET - logical
log_agent = agent$agent_id %in% call$agent_id
agent$negativecalls[log_agent] = tapply(call$negoutcome,
call$agent_id, sum)
#Calculating Ratio of Negative calls/total calls
agent$negativity = 0
agent$negativity = round(agent$negativecalls / agent$Total_CallData_Calls,2)
```

3.2 FEATURE DATA

3.2.1 Categorizing feature data columns based on four categories

1) feature_values 1-24 are measurements and statistical calculations on the speech of the call (total speaking time, etc.).
2) feature_values 25-34, 45-54, 65-74, 85-94, 105-114, 125-134, and 145-154 are statistical calculations on speech that has been identified as expressing a negative emotion (e.g. anger).
3) feature_values 35-44, 55-64, 75-84, 95-104, 115-124, 135-144, and 155-164 are statistical calculations on speech that has been identified as expressing a positive emotion (e.g. joy).
4) feature_values 165-176 measure proportions of negative/positive emotions, emotional/unemotional speech, and transitions from one to the other (positive to negative or unemotional to emotional for example).

```
feature_new <- feature
feature_new$CallSpeech <- rowMeans(feature_new[,3:26])
feature_new_CallSpeech <- feature_new[,3:26]
NE1 <- rowSums(feature_new[,27:36]) #25-34
NE2 <- rowSums(feature_new[,47:56])
NE3 <- rowSums(feature_new[,67:76])
NE4 <- rowSums(feature_new[,87:96])
NE5 <- rowSums(feature_new[,107:116])
NE6 <- rowSums(feature_new[,127:136])
NE7 <- rowSums(feature_new[,147:156])
NETotal <- NE1 + NE2 + NE3 + NE4 + NE5 + NE6 + NE7
feature_new$NegativeEmotions <- NETotal/(70)
PE1 <- rowSums(feature_new[,37:46])
PE2 <- rowSums(feature_new[,57:66])
PE3 <- rowSums(feature_new[,77:86])
PE4 <- rowSums(feature_new[,97:106])
PE5 <- rowSums(feature_new[,117:126])
PE6 <- rowSums(feature_new[,137:146])
PE7 <- rowSums(feature_new[,157:166])
PETotal <- PE1 + PE2 + PE3 + PE4 + PE5 + PE6 + PE7
feature_new$PositiveEmotions <- PETotal/(70)
feature_new$PositiveNegativeProportion <- rowMeans(feature_new[, 167:178])
```

3.2.2 Synchronizing Target Values for each agent in Feature Data.

```
call$new_file_audio <- substr(call$audio_file_name,1,34)
feature_new$new_file_audio <- (substr(feature_new$AUDIO.FILE.NAME, 1,34))
tmp1 = call[which(call$new_file_audio %in%
feature_new$new_file_audio),c("new_file_audio","agent_id")]
names(call)

tmp2 = feature_new[which(feature_new$new_file_audio %in%
call$new_file_audio),]

##Merging agent id to feature
feature_new = merge(x = tmp1,y = tmp2, by.x = "new_file_audio", by.y =
"new_file_audio")

# Code to find the target values assigned to each agent
tmp3 = feature_new[which(feature_new$new_file_audio %in%
call$new_file_audio),c("new_file_audio","target_value")]
##creating new frame to store File Name | Agent ID | Target Values
target_agent = merge(x = tmp1,y = tmp3, by.x = "new_file_audio", by.y =
"new_file_audio")

# Creating a dataframe having columns "agent_id" and "target_value"
tmp4 =
data.frame(tapply(target_agent$target_value,target_agent$agent_id,unique))
tmp4 <- cbind(rownames(tmp4), tmp4)
rownames(tmp4) <- NULL
colnames(tmp4) <- c("agent_id","target_value")
names(agent)
# Adding a new column "newtarget_valuee" in agent dataset for further
computations
# Null and NA values in target_value columns have been replaced by termiated
values for respectve agents
agent = merge(x = agent, y = tmp4 , by = "agent_id", all.x = TRUE)
agent$newtarget_value[agent$target_value == "NA"]  <-
agent$terminated[agent$target_value == "NA"]
agent$newtarget_value[agent$target_value == "NULL"]  <-
agent$terminated[agent$target_value == "NULL"]
as.character(agent$target_value)
agent$newtarget_value[agent$target_value == "c(0, NA)"] <- 0
agent$newtarget_value[agent$target_value == "c(1, NA)"] <- 1
agent$newtarget_value[agent$target_value == "0"] <- 0
agent$newtarget_value[agent$target_value == "1"] <- 1
target_agent = merge(x = target_agent, y =
agent[c("agent_id","newtarget_value")], by = "agent_id", all.x = TRUE)
feature_new = cbind(feature_new,target_agent["newtarget_value"])
length(unique(feature_new$agent_id))
```

3.2.3 Aggregate Agents and Target values across feature and call data

```
feature_agent_target <- aggregate(newtarget_value ~ agent_id, data =
feature_new_1, FUN = mode)
feature_agent_target$newtarget_value <- NA
```

```
names(feature_agent_target)[names(feature_agent_target) == "newtarget_value"]
<- "target"
feature_agent_target <-
merge(feature_agent_target,agent[,c("agent_id","terminated","term_type")],by
= "agent_id", all.y = TRUE)
feature_agent_target$target[feature_agent_target$terminated == 1 &
feature_agent_target$term_type == 0] <- 0
feature_agent_target$target[feature_agent_target$terminated == 1 &
feature_agent_target$term_type == 1] <- 1
feature_agent_target$target[feature_agent_target$terminated == 0] <- 0
feature_agent_target$term_type <- NULL
feature_agent_target$newtarget_value <- NULL
table(feature_agent_target$target)
```

Assumption: The target variable is defined as follows

1 – Voluntary termination

0 – Others (Not terminated or Involuntary termination)

This creates an almost equal distribution of the target variable in the dataset of 116 agents with the following distribution

0	- 47
1	-69

3.2.4 Scaling and Aggregating Feature Data by Agent ID

Functions

```
#functions
mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

#agg_feature_mean <- function(v,w){
#  tapply(v,w,mean)
#}

scale_01 <- function(x){(x-min(x))/(max(x)-min(x))}


#remove unwanted var
feature_new$new_file_audio <- NULL
feature_new$AUDIO.FILE.NAME <- NULL
feature_new$target_value <- NULL

feature_new_1 <- feature_new[,c(1,181,178,179,180,182,2:177)]
feature_agg <- feature_new_1
```

1) **Usual Scaling of all features using in build Scale function and aggregating by agent id using mean and mode**

```
#scale function
scaled.feature_agg <- scale(feature_agg[,7:182],center = TRUE, scale = TRUE)
scaled.feature_agg<-cbind(feature_agg[,1:6],scaled.feature_agg)

scaled.feature_agg_mean = aggregate(. ~ agent_id, data = scaled.feature_agg,
FUN = mean)
scaled.feature_agg_mode = aggregate(. ~ agent_id, data = scaled.feature_agg,
FUN = mode)
```

2) **Custom Scaling function to scale between 0 and 1 of all features and aggregating by agent id using mean and mode**

```
#scale_01 function
scaled_01.feature_agg <-scale_01(feature_agg[,7:182])
scaled_01.feature_agg<-cbind(feature_agg[,1:6],scaled_01.feature_agg)

scaled_01.feature_agg_mean <- aggregate(. ~ agent_id, data =
scaled_01.feature_agg, FUN = mean)
scaled_01.feature_agg_mode <- aggregate(. ~ agent_id, data =
scaled_01.feature_agg, FUN = mode)
```

3) **0 to 1 Scaling of categories in feature data and aggregating by agent id using mean and mode**

```
#feature callspeech
feature_new_callspeech <- feature_new_1[,c(1,7:30)]

scaled_01.feature_callspeech_agg <- scale_01(feature_new_callspeech[-1])
scaled_01.feature_callspeech_agg <-
cbind(feature_new_callspeech[,"agent_id"],scaled_01.feature_callspeech_agg)
names(scaled_01.feature_callspeech_agg)[names(scaled_01.feature_callspeech_ag
g)=="feature_new_callspeech[, \"agent_id\"]"] <- "agent_id"

scaled_01.feature_callspeech_agg_mean <- aggregate(. ~ agent_id, data =
scaled_01.feature_callspeech_agg, FUN = mean)
scaled_01.feature_callspeech_agg_mode <- aggregate(. ~ agent_id, data =
scaled_01.feature_callspeech_agg, FUN = mode)


#feature negetiveemotions
feature_new_negativeemotions <-
feature_new_1[,c(1,31:40,51:60,71:80,91:100,111:120,131:140,151:160)]

scaled_01.feature_negativeemotions_agg <-
scale_01(feature_new_negativeemotions[-1])
```

```
scaled_01.feature_negativeemotions_agg <-
cbind(feature_new_negativeemotions[,"agent_id"],scaled_01.feature_negativeemo
tions_agg)
names(scaled_01.feature_negativeemotions_agg)
names(scaled_01.feature_negativeemotions_agg)[names(scaled_01.feature_negativ
eemotions_agg)=="feature_new_negativeemotions[, \"agent_id\"]"] <- "agent_id"

scaled_01.feature_negetiveemotions_agg_mean <- aggregate(. ~ agent_id, data =
scaled_01.feature_negativeemotions_agg, FUN = mean)
scaled_01.feature_negetiveemotions_agg_mode <- aggregate(. ~ agent_id, data =
scaled_01.feature_negativeemotions_agg, FUN = mode)


#feature positiveemotions
feature_new_positiveemotions <-
feature_new_1[,c(1,41:50,61:70,81:90,101:110,121:130,141:150,161:170)]

scaled_01.feature_positiveemotions_agg <-
scale_01(feature_new_positiveemotions[-1])
scaled_01.feature_positiveemotions_agg <-
cbind(feature_new_positiveemotions[,"agent_id"],scaled_01.feature_positiveemo
tions_agg)
names(scaled_01.feature_positiveemotions_agg)
names(scaled_01.feature_positiveemotions_agg)[names(scaled_01.feature_positiv
eemotions_agg)=="feature_new_positiveemotions[, \"agent_id\"]"] <- "agent_id"

scaled_01.feature_positiveemotions_agg_mean <- aggregate(. ~ agent_id, data =
scaled_01.feature_positiveemotions_agg, FUN = mean)
scaled_01.feature_positiveemotions_agg_mode <- aggregate(. ~ agent_id, data =
scaled_01.feature_positiveemotions_agg, FUN = mode)


#feature positivenegetiveproportions
feature_new_positivenegativeproportion <- feature_new_1[,c(1,171:182)]

scaled_01.feature_positivenegativeproportion_agg <-
scale_01(feature_new_positivenegativeproportion[-1])
scaled_01.feature_positivenegativeproportion_agg <-
cbind(feature_new_positivenegativeproportion[,"agent_id"],scaled_01.feature_p
ositivenegativeproportion_agg)
names(scaled_01.feature_positivenegativeproportion_agg)
names(scaled_01.feature_positivenegativeproportion_agg)[names(scaled_01.featu
re_positivenegativeproportion_agg)=="feature_new_positivenegativeproportion[,
\"agent_id\"]"] <- "agent_id"

scaled_01.feature_positivenegativeproportion_agg_mean <- aggregate(. ~
agent_id, data = scaled_01.feature_positivenegativeproportion_agg, FUN =
mean)
scaled_01.feature_positivenegativeproportion_agg_mode <- aggregate(. ~
agent_id, data = scaled_01.feature_positivenegativeproportion_agg, FUN =
mode)
```

3.2.5 Principle Component Analysis on the above scaled and aggregated data

1) **Usual Scaling of all features using in build Scale function and aggregating by agent id using mean and mode**

```
#pca scaled data
pca_scaled_feature_mean <- prcomp(scaled.feature_agg_mean[,7:182])
summary(pca_scaled_feature_mean)
pca_scaled_feature_mean_ds <- pca_scaled_feature_mean$x[,1:23] #99%
```

99% of the variance is explained by the first 23 principle components

```
pca_scaled_feature_mode <- prcomp(scaled.feature_agg_mode[,7:182])
summary(pca_scaled_feature_mode)
pca_scaled_feature_mode_ds <- as.data.frame(pca_scaled_feature_mode$x[,1:21])
#99%
```

99% of the variance is explained by the first 21 principle components

2) **Custom Scaling function to scale between 0 and 1 of all features and aggregating by agent id using mean and mode**

```
#pca scaled_01 data
pca_scaled_01_feature_mean <- prcomp(scaled_01.feature_agg_mean[,7:182])
summary(pca_scaled_01_feature_mean)
pca_scaled_01_feature_mean_ds <- pca_scaled_01_feature_mean$x[,1:4] #98.5%
pca_scaled_01_feature_mean_ds <-
cbind(as.character(scaled_01.feature_agg_mean$agent_id),pca_scaled_01_feature
_mean_ds)
names(pca_scaled_01_feature_mean_ds)[names(pca_scaled_01_feature_mean_ds)=="V
1"] <- "agent_id"
```

98.5% of the variance is explained by the first 4 principle components

```
pca_scaled_01_feature_mode <- prcomp(scaled_01.feature_agg_mode[,7:182])
summary(pca_scaled_01_feature_mode)
pca_scaled_01_feature_mode_ds <-
as.data.frame(pca_scaled_01_feature_mode$x[,1:6]) #99.2%
pca_scaled_01_feature_mode_ds <-
cbind(scaled_01.feature_agg_mode[,"agent_id"],pca_scaled_01_feature_mode_ds)
names(pca_scaled_01_feature_mode_ds)[names(pca_scaled_01_feature_mode_ds)=="s
caled_01.feature_agg_mode[, \"agent_id\"]"] <- "agent_id"
```

99.2 % of the variance is explained by the first 6 principle components

3) **0 to 1 Scaling of categories in feature data and aggregating by agent id using mean and mode**

```
#feature callspeech
```

```
pca_scaled_01_feature_callspeech_mean <-
prcomp(scaled_01.feature_callspeech_agg_mean[-1])
summary(pca_scaled_01_feature_callspeech_mean)
pca_scaled_01_feature_callspeech_mean_ds <-
as.data.frame(pca_scaled_01_feature_callspeech_mean$x[,1:6]) #80.75..99.5%
```

99.5 % of the variance is explained by the first 6 principle components

```
#feature negetiveemotions

pca_scaled_01_feature_negetiveemotions_mean <-
prcomp(scaled_01.feature_negetiveemotions_agg_mean[-1])
summary(pca_scaled_01_feature_negetiveemotions_mean)
pca_scaled_01_feature_negetiveemotions_mean_ds <-
as.data.frame(pca_scaled_01_feature_negetiveemotions_mean$x[,1:4])
#92.8..99.3%
```

99.3 % of the variance is explained by the first 4 principle components

```
#feature positiveemotions

pca_scaled_01_feature_positiveemotions_mean <-
prcomp(scaled_01.feature_positiveemotions_agg_mean[-1])
summary(pca_scaled_01_feature_positiveemotions_mean)
pca_scaled_01_feature_positiveemotions_mean_ds <-
as.data.frame(pca_scaled_01_feature_positiveemotions_mean$x[,1:8]) #85..99.2%
```

99.2 % of the variance is explained by the first 8 principle components

```
#feature positivenegetiveproportions

pca_scaled_01_feature_positivenegativeproportion_mean <-
prcomp(scaled_01.feature_positivenegativeproportion_agg_mean[-1])
summary(pca_scaled_01_feature_positivenegativeproportion_mean)
pca_scaled_01_feature_positivenegativeproportion_mean_ds <-
as.data.frame(pca_scaled_01_feature_positivenegativeproportion_mean$x[,1:3])
#94.5..99.2%
```

99.2 % of the variance is explained by the first 3 principle components

3.2.6 Dataset Creation

1) Mean Aggregation with 0-1 Scaling

```
ds_feature_scaled01_mean <-
merge(feature_agent_target,pca_scaled_01_feature_mean_ds,by.x =
"agent_id",by.y = "V1", all.y = TRUE)
str(ds_feature_scaled01_mean)
ds_feature_scaled01_mean$PC1 <-
as.numeric(as.character(ds_feature_scaled01_mean$PC1))
ds_feature_scaled01_mean$PC2 <-
as.numeric(as.character(ds_feature_scaled01_mean$PC2))
```

```
ds_feature_scaled01_mean$PC3 <-
as.numeric(as.character(ds_feature_scaled01_mean$PC3))
ds_feature_scaled01_mean$PC4 <-
as.numeric(as.character(ds_feature_scaled01_mean$PC4))
```

1) **Mode Aggregation with 0-1 Scaling**

```
ds_feature_scaled01_mode <-
merge(feature_agent_target,pca_scaled_01_feature_mode_ds,by.x =
"agent_id",by.y = "agent_id", all.y = TRUE)
str(ds_feature_scaled01_mode)
```

2) **Mean Aggregation of Feature categories with 0-1 Scaling**

```
ds_feature_scaled01_emotions_mean <-
cbind(pca_scaled_01_feature_callspeech_mean_ds$PC1,pca_scaled_01_feature_nege
tiveemotions_mean_ds$PC1,pca_scaled_01_feature_positiveemotions_mean_ds$PC1,p
ca_scaled_01_feature_positivenegativeproportion_mean_ds$PC1)
ds_feature_scaled01_emotions_mean <-
cbind(as.data.frame(scaled_01.feature_positivenegativeproportion_agg_mean$age
nt_id),ds_feature_scaled01_emotions_mean)
names(ds_feature_scaled01_emotions_mean)
names(ds_feature_scaled01_emotions_mean)[names(ds_feature_scaled01_emotions_m
ean)=="scaled_01.feature_positivenegativeproportion_agg_mean$agent_id"] <-
"agent_id"
ds_feature_scaled01_emotions_mean <-
merge(feature_agent_target,ds_feature_scaled01_emotions_mean,by.x =
"agent_id",by.y = "agent_id", all.y = TRUE)
```

4 STATISTICAL ANALYSIS AND MODELING

4.1 AGENT AND CALL DATA

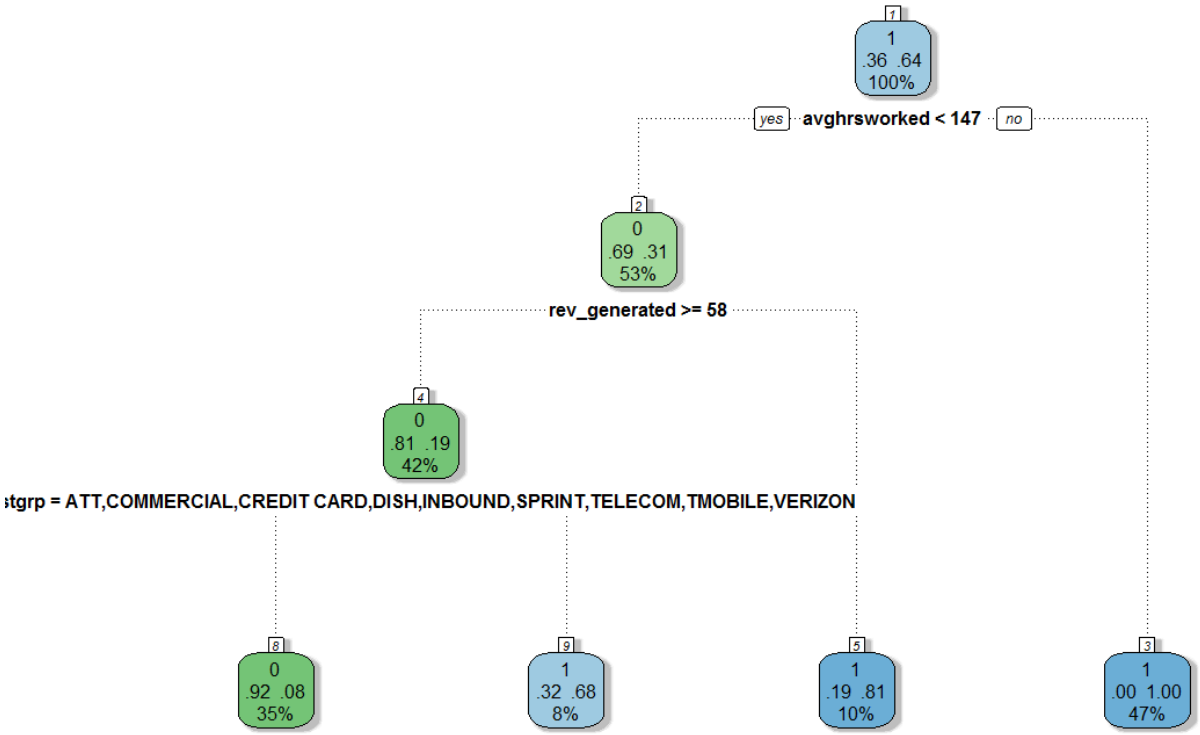
We decided to first build a few trees without using the Rank Miner feature vectors. The interpretation of doing so was to analyze if such data could be combined with the RankMiner data in order to build better models. The statistical language that we have used to build our models is 'R'. So our modelling process is:

1. **Build models on the master data statistics of the agent as obtained from the agent dataset and their call dataset**
2. **Build models on to evaluate the usefulness of the Feature Vectors that are generated by Rank Miner**
3. **Integrate the above two best datasets obtained and try to find a statistically significant model**

4.1.1 MODEL 1: Decision trees

Code:

```
tree = rpart(terminated~.,data=agent_trimmed,control =
rpart.control(cp=0.02),method="class")

library(rpart.plot)
fancyRpartPlot(tree)
```




```
##Recall - Confusion Matrix
confusion.advisory_tree <-
table(agent_trimmed[,"terminated"],predict(tree,agent_trimmed,type =
"response"))
confusion.advisory_tree ##Accuracy == 92.7%

#Printing the Tree:
> print(tree)
n= 249

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 249 90 1 (0.36144578 0.63855422)
  2) avghrsworked< 147.1417 131 41 0 (0.68702290 0.31297710)
    4) rev_generated>=58.1275 105 20 0 (0.80952381 0.19047619)
      8) lastgrp=ATT,COMMERCIAL,CREDIT CARD,DISH,INBOUND,SPRINT,TELECOM,TMOBILE,VERIZON 86  7 0 (
0.91860465 0.08139535) *
      9) lastgrp=DIRECTV,OTHER 19  6 1 (0.31578947 0.68421053) *
    5) rev_generated< 58.1275 26  5 1 (0.19230769 0.80769231) *
  3) avghrsworked>=147.1417 118  0 1 (0.00000000 1.00000000) *
```

The above snapshot clearly shows the number of agents in each of the splits and the accuracies of the splits. We can derive our predictions based on the above representation of the tree.

```
> confusion.advisory_tree

    0   1
0  79  11
```

```
  1    7 152

##Accuracy == 92.7%
```

Analysis and Interpretation (Similar for all tree):

The tree has given us a very good accuracy on the Recall Data.

It has output major splits as visible in the tree above.

Significant Predictors: 1. Average Hours Worked

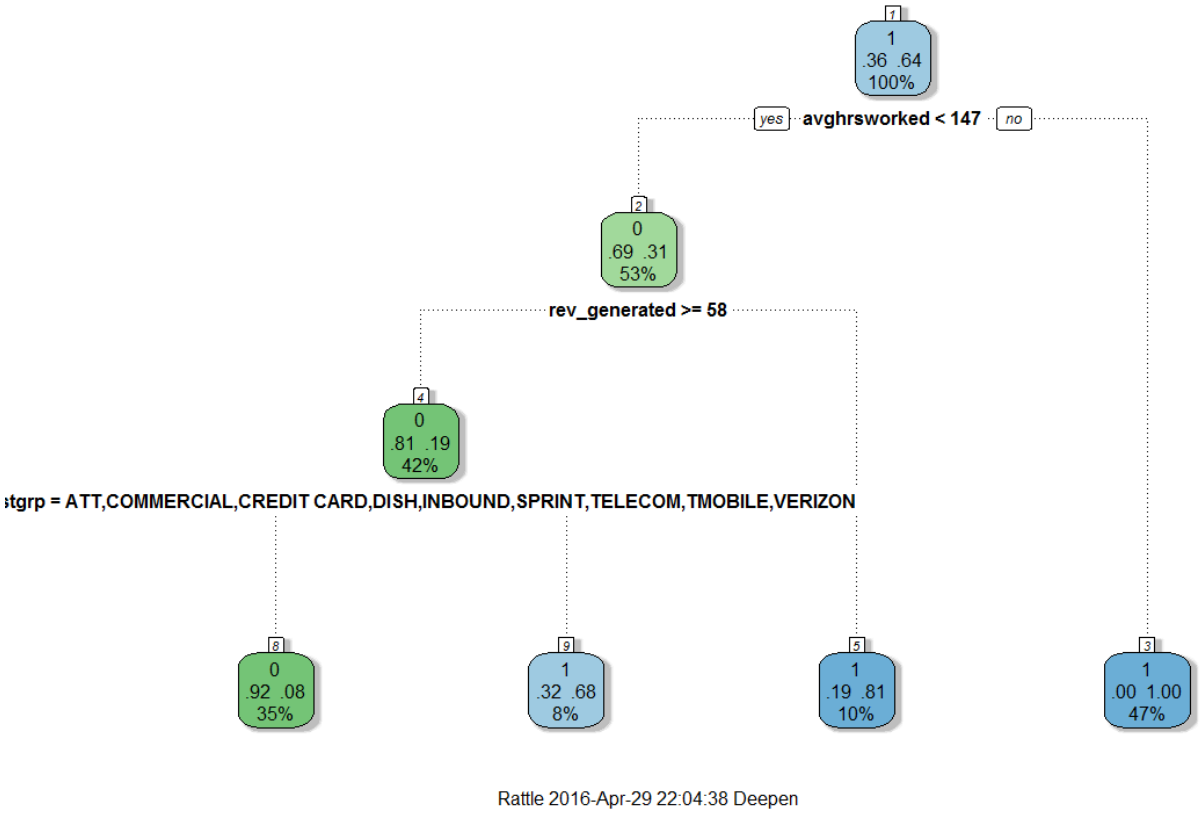
2. Revenue Generated

3. The last group the agent worked for

These results were also very upfront in the visualizations at the beginning of this report.

We now try changing the complexity parameter (cp) to reduce pruning:

```
####Changing complexity parameter to 0.01 -- reducing pruning
tree2 = rpart(terminated~.,data=agent_trimmed,control =
rpart.control(cp=0.01),method="class")
rpart.plot(tree2,extra=101,type=4) ##Does not change , i.e. best splits
obtained
```



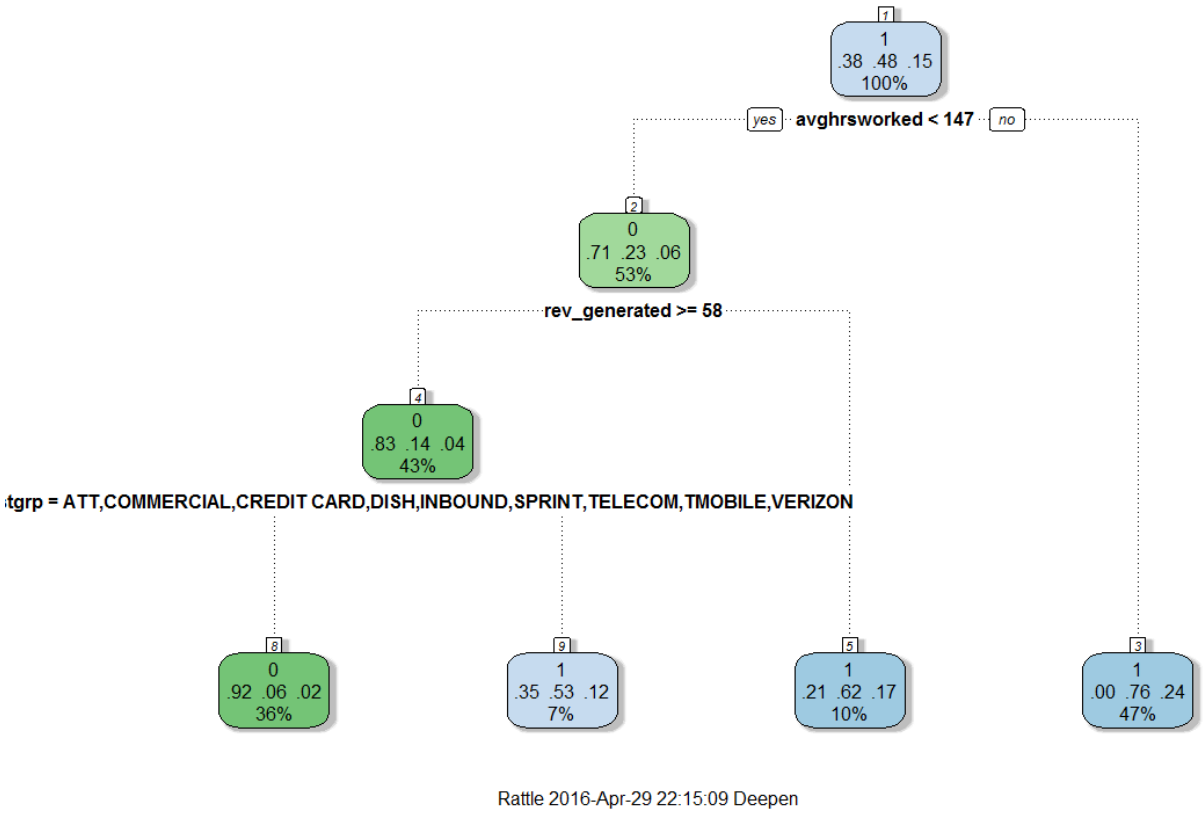
No significant difference is obtained by doing so, thus we would try making some different tweaks to our data

Here, we have separated the "Voluntary" and "Involuntary" people leaving.
The output of our decision tree will be classified into:
Voluntary = 1, Involuntary = 2 and Did Not Leave = 0.

```
########TREE with 2 ---------------------------------

agent_trimmed2 =
agent[,c("terminated","term_type","lastgrp","rev_generated","commision","Call
_Median_Duration","Total_CallData_Calls",
"negativity","skillgroupdiff","avghrsworked")]


##Deleting 49 rows as done for agent_trimmed
agent_trimmed2= agent_trimmed2[!is.na(agent_trimmed2$lastgrp),]

##Deleting rows that have their term type unknown - 16 agents
agent_trimmed2 = agent_trimmed2[!(is.na(agent_trimmed2$term_type) &
agent_trimmed2$terminated==1),]
```

```
###Changing the value of INVOLUNTARILY left agents to 2
agent_trimmed2$terminated[which(agent_trimmed2$term_type==0 &
agent_trimmed2$terminated==1)]=2


###Changing the value of INVOLUNTARILY left agents to 2
agent_trimmed2$terminated[which(agent_trimmed2$term_type==0 &
agent_trimmed2$terminated==1)]=2


###Excluding Term Type
agent_trimmed2$term_type = NULL

#########TREE----------------------------
tree3 = rpart(terminated~.,data=agent_trimmed2,control =
rpart.control(cp=0.02),method="class")

fancyRpartPlot(tree3)
```

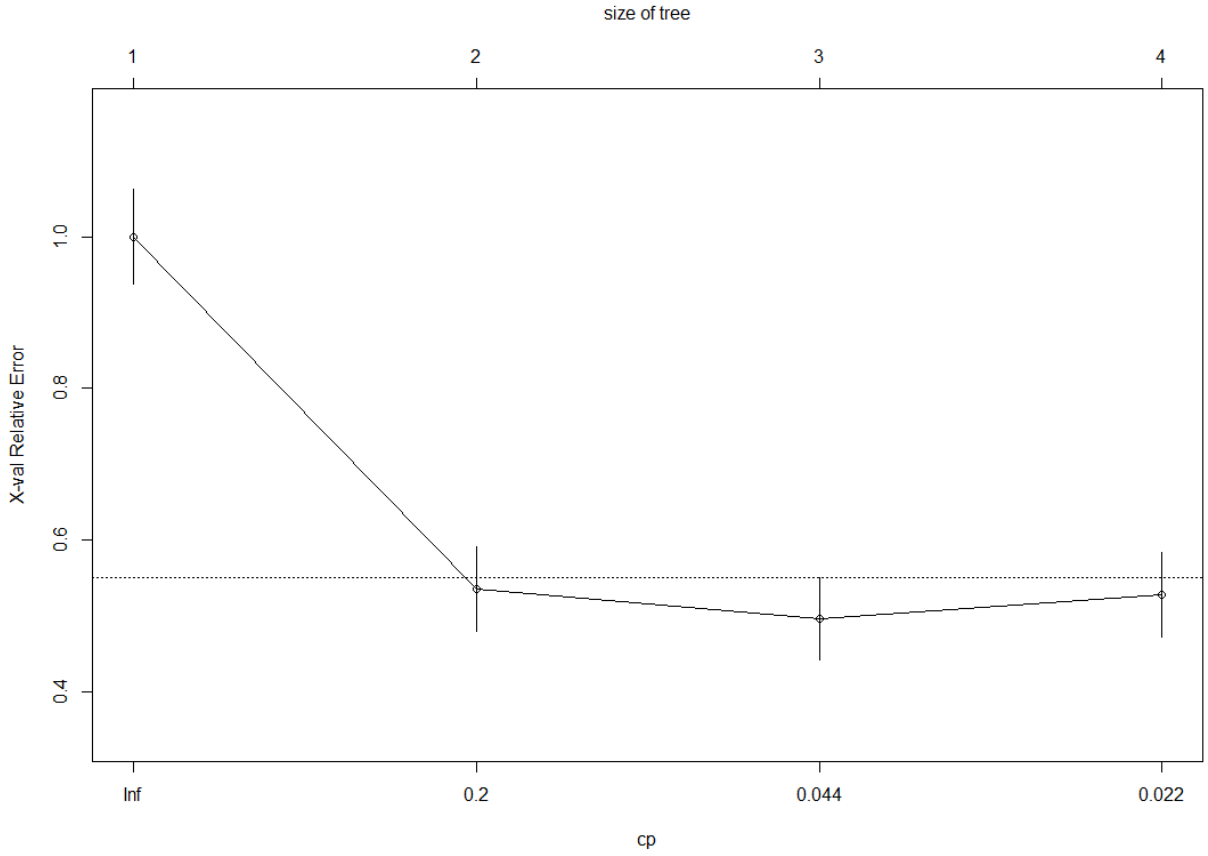
Plot:



Analysis of the tree:

```
> print(tree3)
n= 240

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 240 125 1 (0.37500000 0.47916667 0.14583333)
  2) avghrsworked< 147.1417 127  37 0 (0.70866142 0.22834646 0.06299213)
    4) rev_generated>=58.1275 103  18 0 (0.82524272 0.13592233 0.03883495)
      8) lastgrp=ATT,COMMERCIAL,CREDIT CARD,DISH,INBOUND,SPRINT,TELECOM,TMOBILE,VERIZON 86   7 0
(0.91860465 0.05813953 0.02325581) *
      9) lastgrp=DIRECTV,OTHER 17   8 1 (0.35294118 0.52941176 0.11764706) *
    5) rev_generated< 58.1275 24   9 1 (0.20833333 0.62500000 0.16666667) *
  3) avghrsworked>=147.1417 113  27 1 (0.00000000 0.76106195 0.23893805) *
```

Plotting the Complexity Parameter to identify if pruning is required and until what number of split



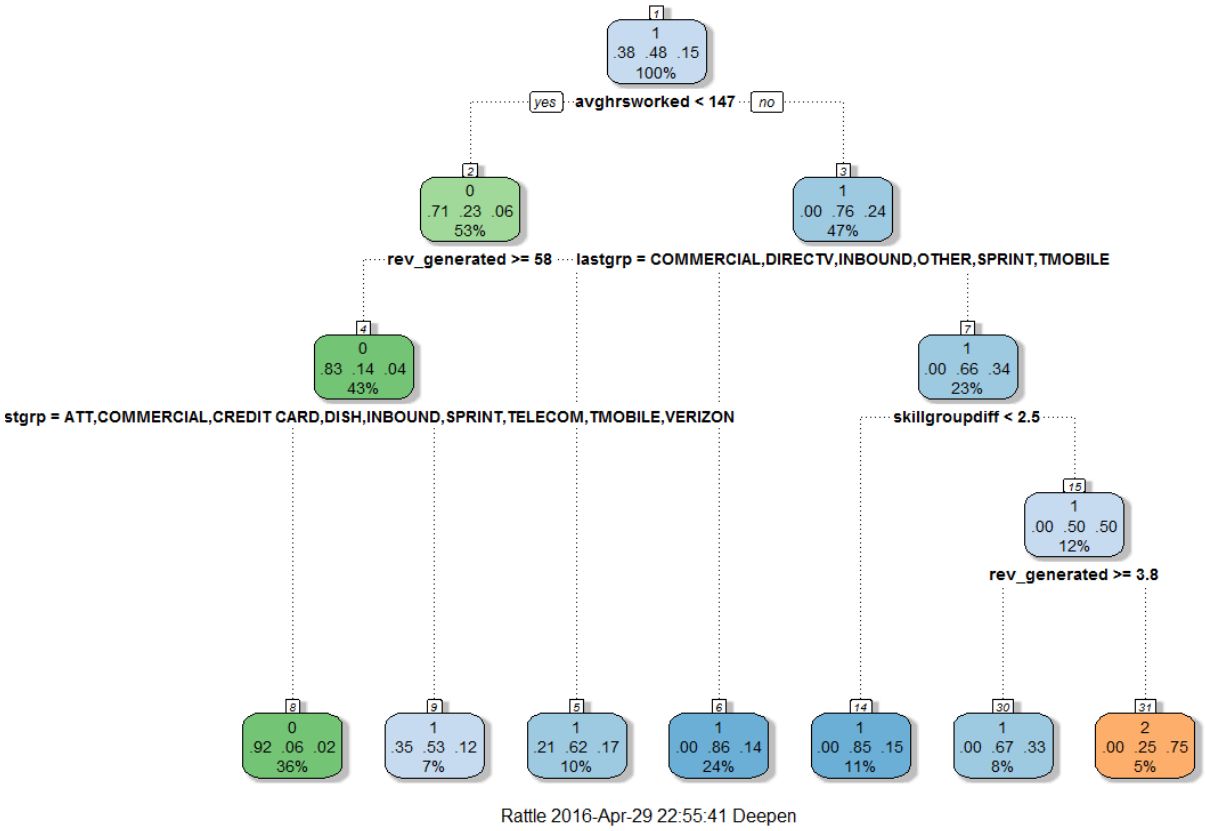

```
##Recall - Confusion Matrix
confusion.tree3 <-
table(agent_trimmed2[,"terminated"],predict(tree3,agent_trimmed2,type =
"response"))
confusion.tree3    ##Accuracy == 92.7%
```

```
> confusion.tree3

     0   1   2
0   79  11   0
1    5 110   0
2    2  33   0
```

```
##Accuracy == 92.7%
```

Again changing the complexity parameter to see any possible extensions to the tree:

```
tree4 = rpart(terminated~.,data=agent_trimmed2,control =
rpart.control(cp=0.01),method="class")
rpart.plot(tree4,extra=101,type=4) ##Does not change , i.e. best splits
obtained
```



It does appear that the right split gives different results

```
confusion.tree4 <-
table(agent_trimmed2[,"terminated"],predict(tree4,agent_trimmed2,type =
"response"))
confusion.tree4 ##Accuracy == 81.25%
```

Difference in the models:

The above 4 models have been interpreted by differing the Dependent Variable from (0 and 1) to (0 and [1,2]). Thus, it would not be ideal to compare 1 & 2 with 3 & 4.

Thus, based on business decisions and implications, we can choose the model that suits best.

By now, we are confident that our important predictors are Average Worked hours, the Revenue generated and the Last group in which the agent worked before he left.

We will now run a regression model to identify the importance of the variables to support our analysis.

4.1.2 MODEL 2: Logistic Binomial Regression

MODEL 2.1

We use the same dataset as we used for MODEL 1 viz. agent_trimmed that has the Dependent variables as 0 and 1.

```
lr = glm(terminated ~ .,data=agent_trimmed,family=binomial)
summary(lr)
> summary(lr)

Call:
glm(formula = terminated ~ ., family = binomial, data = agent_trimmed)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1701  -0.4627   0.1165   0.4701   3.3188

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.615e+00  1.641e+00  -0.984  0.32502
lastgrpCOMMERCIAL      3.593e-03  1.530e+00   0.002  0.99813
lastgrpCREDIT CARD     9.833e-01  1.187e+00   0.828  0.40756
lastgrpDIRECTV         1.636e+00  1.161e+00   1.409  0.15889
lastgrpDISH           -9.544e-01  9.614e-01  -0.993  0.32085
lastgrpG 13 LEGAL      1.277e+01  1.455e+03   0.009  0.99300
lastgrpINBOUND        -7.250e-01  9.925e-01  -0.731  0.46508
lastgrpOTHER           3.759e+00  1.230e+00   3.057  0.00224 **
lastgrpSPRINT         -1.033e+00  9.922e-01  -1.041  0.29787
lastgrpTELECOM         2.153e+00  1.183e+00   1.820  0.06881 .
lastgrpTMOBILE         8.683e-01  9.816e-01   0.885  0.37640
lastgrpVERIZON         8.402e-01  1.275e+00   0.659  0.50975
rev_generated         -3.853e-04  1.767e-04  -2.180  0.02923 *
commision             -1.497e+00  8.592e-01  -1.743  0.08139 .
Call_Median_Duration  -3.798e-03  4.355e-03  -0.872  0.38318
Total_CallData_Calls  -4.133e-04  2.944e-04  -1.404  0.16036
negativity            -1.147e+00  1.146e+00  -1.001  0.31685
skillgroupdiff        -3.704e-01  1.401e-01  -2.644  0.00819 **
avghrsworked           4.878e-02  7.752e-03   6.293 3.11e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 318.35  on 241  degrees of freedom
Residual deviance: 158.63  on 223  degrees of freedom
  (7 observations deleted due to missingness)
AIC: 196.63
```

Look at the AIC value above, we will use this to compare any modelling changes.

MODEL 2.2

```
###It seems that the Lastgroup with "OTHER" is significant, so we create a
dummy for last group = other or not.
```

```
agent_trimmed3 = agent_trimmed
agent_trimmed3$lastother <- 0

agent_trimmed3$lastgrp[which(agent_trimmed3$lastgrp=="OTHER")] = 1
agent_trimmed3$lastgrp <- NULL
```

Running a new Model with the above changes:

```
lr1 = glm(terminated ~ .,data=agent_trimmed3,family=binomial)
summary(lr1)
```

```
> summary(lr1)

Call:
glm(formula = terminated ~ ., family = binomial, data = agent_trimmed3)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0779  -0.7300   0.2936   0.5849   2.9006

Coefficients: (1 not defined because of singularities)
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        0.0139171  1.1001005   0.013  0.98991
rev_generated     -0.0005237  0.0001434  -3.652  0.00026 ***
commision         -1.1391877  0.6316768  -1.803  0.07132 .
Call_Median_Duration -0.0036163  0.0035895  -1.007  0.31372
Total_CallData_Calls -0.0002895  0.0002146  -1.349  0.17743
negativity        -0.7348621  0.9871869  -0.744  0.45663
skillgroupdiff    -0.2151289  0.0960168  -2.241  0.02506 *
avghrsworked       0.0310459  0.0047743   6.503 7.89e-11 ***
lastother                 NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 318.35  on 241  degrees of freedom
Residual deviance: 205.19  on 234  degrees of freedom
  (7 observations deleted due to missingness)
AIC: 221.19
```

We obtained a higher AIC value compared to the previous one. It seems that the model was overfitted. Now lets try to analyze our Null Hypothesis that adding the dependent variables is a good predictor of the model i.e. the reduction in the Null Deviation to the Residual Deviance. We have used the chi-square characteristic to determine our p-value.

```
#Null Deviance ; P- Value =
```

```r
1-pchisq(318.35,df=241)
##0.0006

#Statistical reducation in Null Deviance by adding the independent variables
1-pchisq(205.19,df=234)
##0.91

##
1-pchisq(318.35-205.19,df=241-234)
#0
```

```
#This gives the p value of the model. it is approximating 0 which means that
the null hypothesis can be rejected.
#Null Hypothesis being that there is no significant difference between the
predictors for the ones who left and ones who did not leave.


#AIC = 221.19


glm_predicted = predict(lr1,agent_trimmed3,type = "response")
glm_predicted[glm_predicted>=0.5]=1
glm_predicted[glm_predicted<0.5]=0

##Accuracy on Recall
confusion.lr1 = table(agent_trimmed3[,"terminated"],glm_predicted)
```

```
> confusion.lr1
   glm_predicted
      0   1
  0  65  24
  1  22 131
```

```
(65+131)/249
#Accuracy = 78.7%
```

4.1.3 MODEL ANALYSIS AND RESULTS

Since Model 2.1 has a better AIC then Model 2.2, we used it for calculating the odds of the dependency on the dependent variable.

4.2 FEATURE DATA

4.2.1 Model Selection and Development:

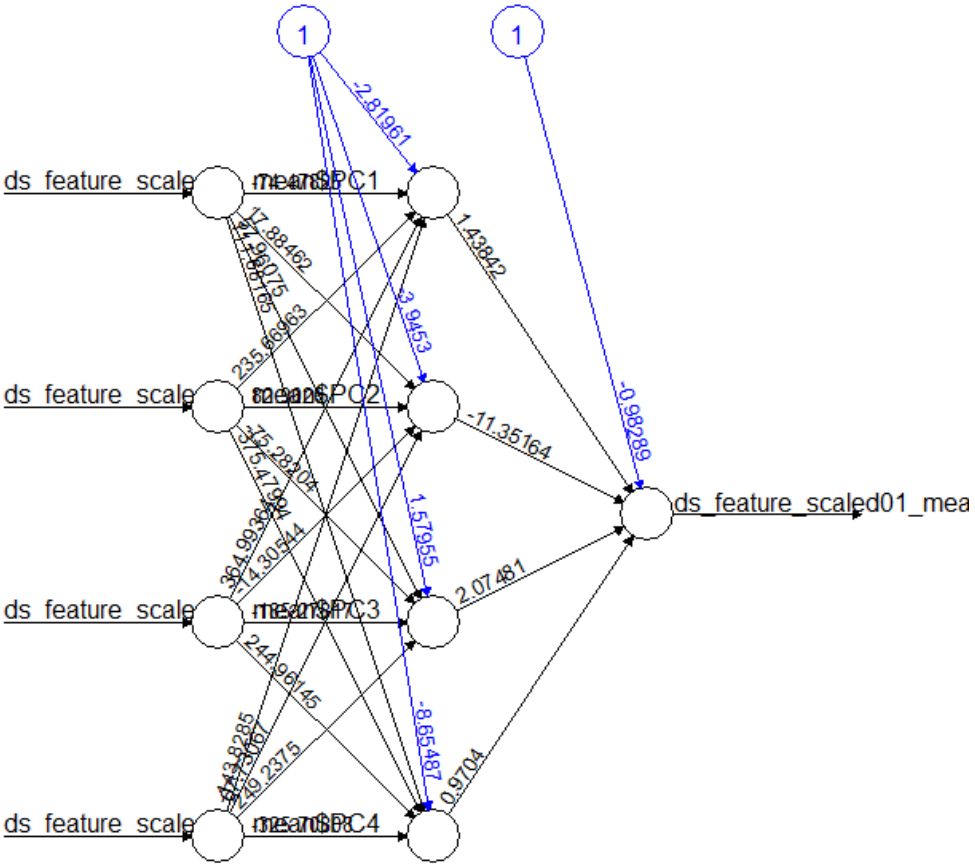
Three models were run on three different feature dataset

1) **Top 4 PCA components (98.5% var) aggregated by mean for 117 agents**
2) **Top 6 PCA components (99.2% var) aggregated by mode for 117 agents**
3) **Top 1 PCA component from each of the 4 categories in feature dataset i.e. 4 PCAs (each > 85% var)**

Detailed:

1) **Top 4 PCA components (98.5% var) aggregated by mean for 117 agents**

```
#scaled01_mean
net_feature_scaled01_mean.sqrt <-
neuralnet(ds_feature_scaled01_mean$target~ds_feature_scaled01_mean$PC1+ds_fea
ture_scaled01_mean$PC2+ds_feature_scaled01_mean$PC3+ds_feature_scaled01_mean$
PC4,ds_feature_scaled01_mean, hidden = 4)
plot(net_feature_scaled01_mean.sqrt)
ds_feature_scaled01_mean_recall <- ds_feature_scaled01_mean[,4:7]
net_feature_scaled01_mean.results <- compute(net_feature_scaled01_mean.sqrt,
ds_feature_scaled01_mean_recall) #Run them through the neural network
ls(net_feature_scaled01_mean.results)
ds_feature_scaled01_mean_predicted <- ds_feature_scaled01_mean
ds_feature_scaled01_mean_predicted$predicted <- NA
ds_feature_scaled01_mean_predicted$predicted <-
net_feature_scaled01_mean.results$net.result
summary(ds_feature_scaled01_mean_predicted$predicted)
ds_feature_scaled01_mean_predicted$predicted[net_feature_scaled01_mean.result
s$net.result >= 0.5] <- 1
ds_feature_scaled01_mean_predicted$predicted[net_feature_scaled01_mean.result
s$net.result < 0.5] <- 0
length(ds_feature_scaled01_mean_predicted$agent_id[ds_feature_scaled01_mean_p
redicted$target == ds_feature_scaled01_mean_predicted$predicted &
ds_feature_scaled01_mean_predicted$predicted == 1])
#53/116*100 = 45.68
length(ds_feature_scaled01_mean_predicted$agent_id[ds_feature_scaled01_mean_p
redicted$target == 1])
#69/116*100 = 59.48
#confusion matrix
table(ds_feature_scaled01_mean_predicted$target,ds_feature_scaled01_mean_pred
icted$predicted)
net_feature_scaled01_mean.acc = (32+57)/116
```



Error: 8.363399 Steps: 4928

Model Validation:

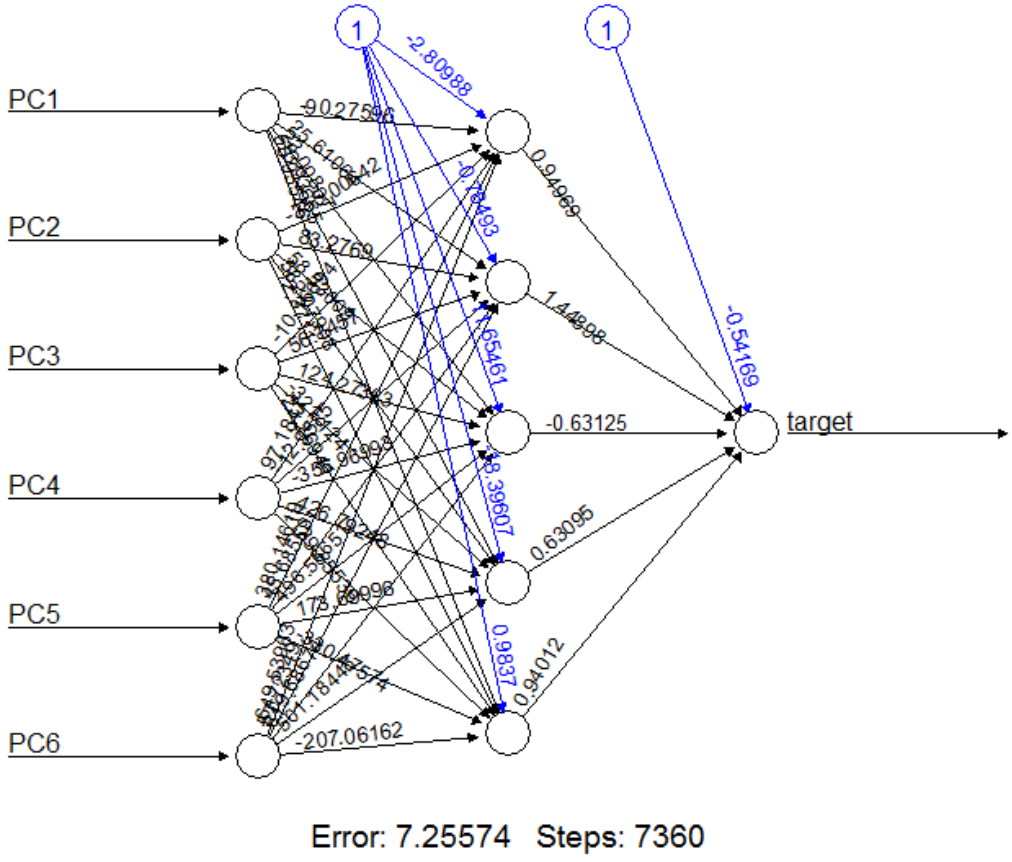
Net Recall Accuracy - 76.72%
Confusion Matrix –

	0	1
0	32	15
1	12	57

2) **Top 6 PCA components (99.2% var) aggregated by mode for 117 agents**

```
#scaled01_mode
net_feature_scaled01_mode.sqrt <-
neuralnet(target~PC1+PC2+PC3+PC4,ds_feature_scaled01_mode, hidden = 4)
plot(net_feature_scaled01_mean.sqrt)
ds_feature_scaled01_mode_recall <- ds_feature_scaled01_mode[,4:7]
net_feature_scaled01_mode.results <- compute(net_feature_scaled01_mode.sqrt,
ds_feature_scaled01_mode_recall) #Run them through the neural network
ds_feature_scaled01_mode_predicted <- ds_feature_scaled01_mode
```

```
ds_feature_scaled01_mode_predicted$predicted <- NA
ds_feature_scaled01_mode_predicted$predicted <-
net_feature_scaled01_mode.results$net.result
summary(ds_feature_scaled01_mode_predicted$predicted)
ds_feature_scaled01_mode_predicted$predicted[net_feature_scaled01_mode.result
s$net.result >= 0.5] <- 1
ds_feature_scaled01_mode_predicted$predicted[net_feature_scaled01_mode.result
s$net.result < 0.5] <- 0
length(ds_feature_scaled01_mode_predicted$agent_id[ds_feature_scaled01_mode_p
redicted$target == ds_feature_scaled01_mode_predicted$predicted &
ds_feature_scaled01_mode_predicted$predicted == 1])
66/116*100
length(ds_feature_scaled01_mode_predicted$agent_id[ds_feature_scaled01_mode_p
redicted$target == 1])
#69/116*100 = 59.48
#confusion matrix
table(ds_feature_scaled01_mode_predicted$target,ds_feature_scaled01_mode_pred
icted$predicted)
net_feature_scaled01_mode.acc <- (26+55)/116
```



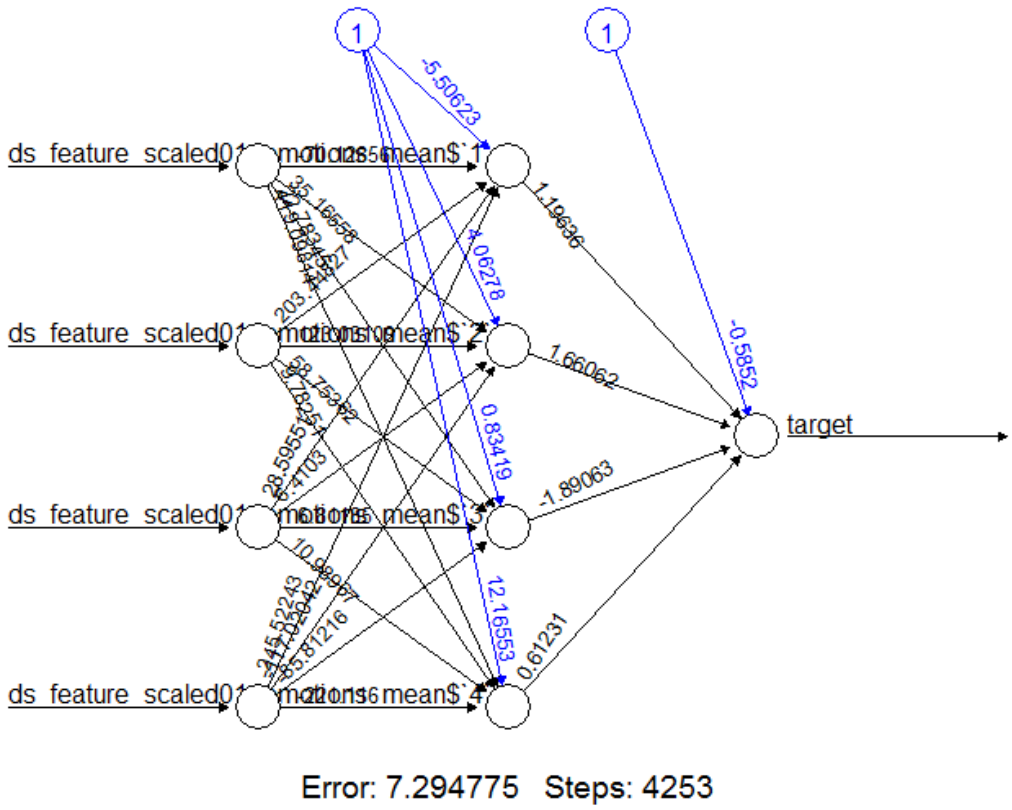
Model Validation:

Net Recall Accuracy – 81.9 %
Confusion Matrix –

```
   0  1
0 38  9
1 12 57
```

3) Top PCA components from each of the 4 categories in feature dataset (each > 85% var)

```
#scaled01_feature_emotions
net_feature_scaled01_emotions.sqrt <-
neuralnet(target~ds_feature_scaled01_emotions_mean$`1`+ds_feature_scaled01_em
otions_mean$`2`+ds_feature_scaled01_emotions_mean$`3`+ds_feature_scaled01_emo
tions_mean$`4`,ds_feature_scaled01_emotions_mean, hidden = 4)
plot(net_feature_scaled01_emotions.sqrt)
ds_feature_scaled01_emotions_mean_recall <-
ds_feature_scaled01_emotions_mean[,4:7]
net_feature_scaled01_emotions.results <-
compute(net_feature_scaled01_emotions.sqrt,
ds_feature_scaled01_emotions_mean_recall) #Run them through the neural
network
ds_feature_scaled01_emotions_mean_predicted <-
ds_feature_scaled01_emotions_mean
ds_feature_scaled01_emotions_mean_predicted$predicted <- NA
ds_feature_scaled01_emotions_mean_predicted$predicted <-
net_feature_scaled01_emotions.results$net.result
summary(ds_feature_scaled01_emotions_mean_predicted$predicted)
ds_feature_scaled01_emotions_mean_predicted$predicted[net_feature_scaled01_em
otions.results$net.result >= 0.5] <- 1
ds_feature_scaled01_emotions_mean_predicted$predicted[net_feature_scaled01_em
otions.results$net.result < 0.5] <- 0
length(ds_feature_scaled01_emotions_mean_predicted$agent_id[ds_feature_scaled
01_emotions_mean_predicted$target ==
ds_feature_scaled01_emotions_mean_predicted$predicted &
ds_feature_scaled01_emotions_mean_predicted$predicted == 1])
66/116*100
length(ds_feature_scaled01_emotions_mean_predicted$agent_id[ds_feature_scaled
01_emotions_mean_predicted$target == 1])
#69/116*100 = 59.48
#confusion matrix
table(ds_feature_scaled01_emotions_mean_predicted$target,ds_feature_scaled01_
emotions_mean_predicted$predicted)
net_feature_scaled01_emotions.acc = (37+60)/116
```



Error: 7.294775 Steps: 4253

Model Validation:

Net Recall Accuracy – 83.62 %
Confusion Matrix –

	0	1
0	37	10
1	9	60

4.2.2 Model Analysis and Results

The baseline accuracy for predicting a voluntary termination i.e. a 1 for 116 agents is 69/116 i.e. **59.48%.**

So any model with accuracy more than this is an acceptable model.

Summary of model accuracies-

Model 1	Model 2	Model 3
76.72%	81.90%	83.62%

Hence a neural network model on Top PCA components extracted from each category i.e. Call Speech, Negative Emotions, Positive Emotions, Positive Negative Proportions is the best model with least FP and FN so far and to predict recall with 83.62 % accuracy.

4.2.3 Modeling Extensions

1) Aggregating PCA components of each category based on 95% or mode instead of mean and remodeling.
2) Building models on the in-build scaled data as well
3) Using more combinations of PCA components and parameters for modeling.
4) Changing Neural Network parameters like Threshold, Hidden nodes, epochs etc.

5 CONCLUSION

5.1 MANAGARIAL IMPLECATIONS OF RESULTS

It seems that agents who are working for greater number of hours are leaving the company ~ 76% Voluntarily and 24% Involuntarily.

As a manager, this could probably imply different meanings to me for both the different types of attritions:

1. Agents who leave voluntarily are probably over burdened with their work. They are unhappy about the extra hours that they are putting. Some questions I would ask myself:

    - Are these agents being supported by additional incentives for the higher number of hours they are putting in?
    - Are they happy with their work?
    - Are they actually productive enough for the amount of hours they put?

I would closely monitor these agents.

2. Agents who have left Involuntarily or fired might be the cause of some unproductive hours being added to the timesheet over time. This costs the company additional resources and thus management decision might have influenced their termination.
3. Agents who have generated less revenue for the company has ~60% employees who have left voluntarily. For the same reasons discussed above, their productivity is being reduced as they might not be enjoying their work or planning to switch their company.
4. A large proportion of agents who have generated higher revenue are staying back with the company.
    - I might have to check if the company is focusing or paying attention to agents who have only generated revenue? What about those who are also putting in more hours and might be trying hard, but not being successful.
    - Is my business model too biased on incentives to agents who generate sufficient revenue?
5. It also seems that a certain "Groups" have higher attrition rates. I might monitor these groups closely and try to find a detail analysis on them. It is highly likely that the agents are facing a huge amount of problems with their customers in these groups that are forcing them to move away from work.

Using Call Audio Feature Data, analysis on the audio categories i.e. the call speech quality, positive emotions, negative emotions , positive negative emotion proportions affect agent attrition rate and can be predicted with more than 75% accuracy.

6 FUTURE WORK

1) We have only used Principle component analysis for dimension reduction, we can try using Lasso as well for variable selection to run better models.

2) Multivariate Analysis and more Logistic regressions on feature data can be considered on feature data subsets

3) Bayesian Probabilistic models could be applied to the feature, agent and call data together to understand how call audio data is influenced by the agent behavior and get probability estimations on the termination.

7 SOURCE CODE

rankminer_Final.R

8 ACKNOWLEDGEMENT

It is my proud privilege to epitomize my deepest sense of gratitude and indebtedness to the Professor **Daniel Zantedeschi ,** an Eminent Scholar for the Information Systems Decision Sciences Department of the University of South Florida and **Eric** from RankMiner for his valuable guidance, keen and sustained interest, intuitive ideas and persistent endeavor. His inspiring assistance, laconic reciprocation and affectionate care enabled me to complete my work smoothly and successfully.