Project Report
On
University of Liverpool - Ion Switching

Data Analytics in R - Final Project
Spring 2020

Submitted by : Group 7
Group Link : https://www.kaggle.com/ashishkumar69

Darshan Shah (ds962)
Jasneek Singh Chugh (jc2433)
Rohit Phadke (rp879)
Gagan Shrivastava (gs562)
Ashish Kumar (ak2633)

INTRODUCTION

A single ion channel is a pore in the cell membrane that can assume an "open" state in which ions (such as potassium or sodium) can pass through. Ion channels are "gated", i.e. they open in response to a specific stimulus, such as a change in membrane potential (voltage-gated ion channels) or the binding of a neurotransmitter (ligand-gated ion channels). Ion channels are present in the membranes of all excitable cells. Their functions include establishing a resting membrane potential, shaping action potentials and other electrical signals by gating the flow of ions across the cell membrane, controlling the flow of ions across secretory and epithelial cells, and regulating cell volume.

When ion channels open, they pass electric currents. Existing methods of detecting these state changes are slow and laborious. Humans must supervise the analysis, which imparts considerable bias, in addition to being tedious. These difficulties limit the volume of ion channel current analysis that can be used in research.

The task is to create a model to predict the number of open channels based on electrophysiological signal data, at each timestamp.

Overview of the data:

The data was recorded in the batches of 50 seconds. Therefore, each 500,000 rows are in one batch. The training data has 10 batches, total 5,000,000 and the test data has 4 batches, total 2,000,000 records.

We have used the cleaned data, i.e. data without the drift.

Link for the cleaned data: https://www.kaggle.com/cdeotte/data-without-drift

```
#Loading required packages and Libraries
library(data.table)
library(dplyr)
library(ggplot2)
library('scales')
library('grid')
library('gridExtra')
library('RColorBrewer')
library('corrplot')
library('ggridges')
library(caTools)
library(randomForest)
library(glmnet)
library(tidyverse)
library(caret)
library(rpart)

> setwd("/Users/jasneekchugh/Desktop/DataScience/R-Programming/Ion-Switching")

> #reading the required files
> trainData <- fread('data/train_clean.csv', sep = ",", header=T)
> testData<- fread('data/test_clean.csv', sep = ",", header=T)
> sampleSubmission<- fread('data/sample_submission.csv', sep = ",", header=T)

#summary of the data
> #The data was recorded in batches of 50 seconds. Therefore, there are 500,000 rows per batch.
> str(trainData)
Classes ‘data.table’ and 'data.frame':5000000 obs. of  3 variables:
 $ time        : num  1e-04 2e-04 3e-04 4e-04 5e-04 6e-04 7e-04 8e-04 9e-04 1e-03 ...
 $ signal      : num  -2.76 -2.85 -2.42 -3.13 -3.14 ...
 $ open_channels: int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

> head(trainData)#Train data has 10 batches

	time	signal	open_channels
1	1e-04	-2.7600	0
2	2e-04	-2.8557	0
3	3e-04	-2.4074	0
4	4e-04	-3.1404	0
5	5e-04	-3.1525	0
6	6e-04	-2.6418	0

> dim(trainData)
[1] 5000000 3

> table(trainData$open_channels) #0-10, 11 levels. Prediction will 11 possible values.
#Time- 50 seconds long 10khz sample(500,000 rows per batch) meaning we have 10 batches.
#The data is continuous within the batches, but discontinuous between the batches.

	Var1	Freq
1	0	1240152
2	1	985865
3	2	553924
4	3	668609
5	4	403410
6	5	277877
7	6	188112
8	7	265015
9	8	245183
10	9	136120
11	10	35733

> head(testData) #Test data has 4 batches

	time	signal
1	500.0001	-2.6513
2	500.0002	-2.8466
3	500.0003	-2.8538
4	500.0004	-2.4438
5	500.0005	-2.6125
6	500.0006	-2.5692

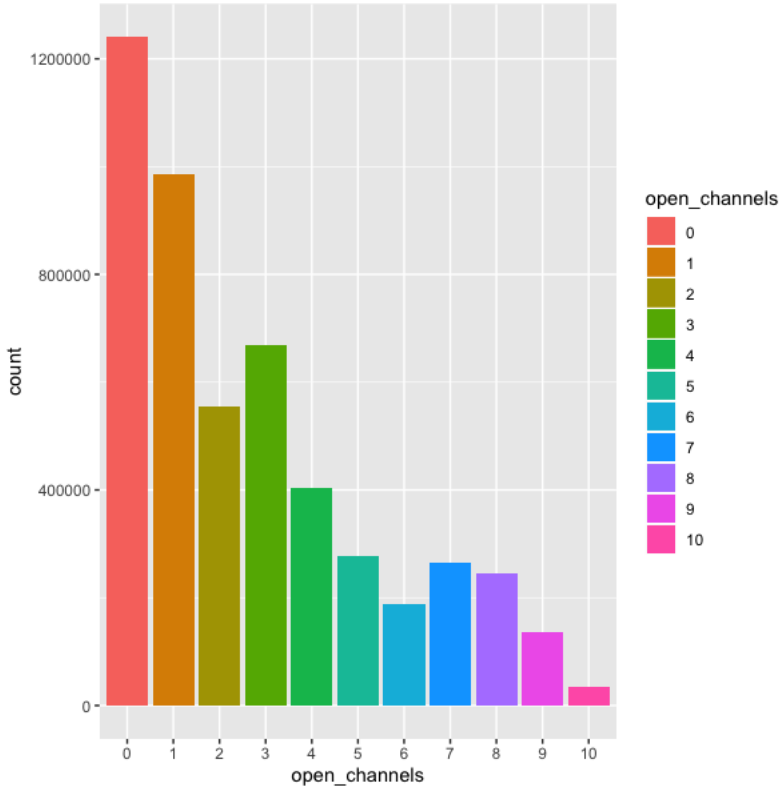
> dim(testData)#2000000, 2
[1] 2000000 2

> #reformatting
> trainData <- trainData %>%

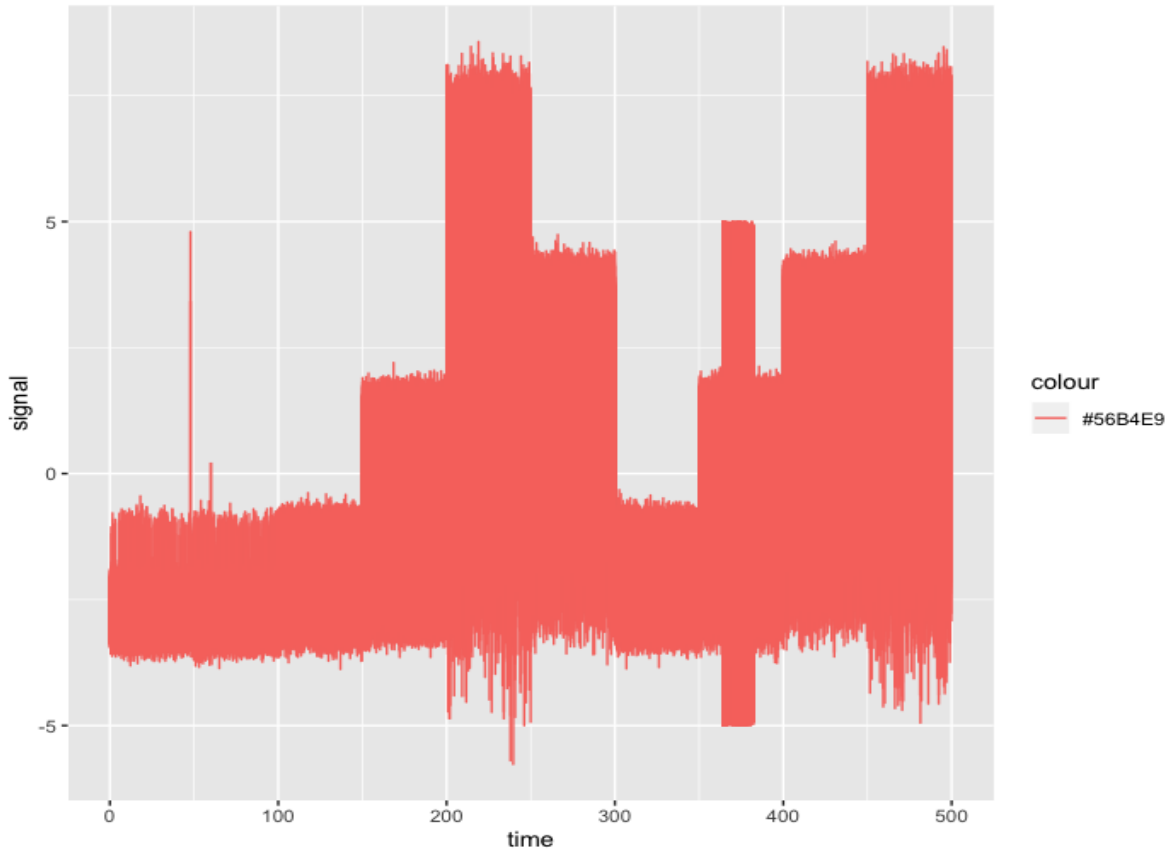
+ mutate(open_channels = factor(open_channels)) %>%
+ mutate(signal= as.numeric(signal))

Data Visualizations

> ggplot(data=trainData)+geom_bar(aes(x=open_channels, fill= open_channels))

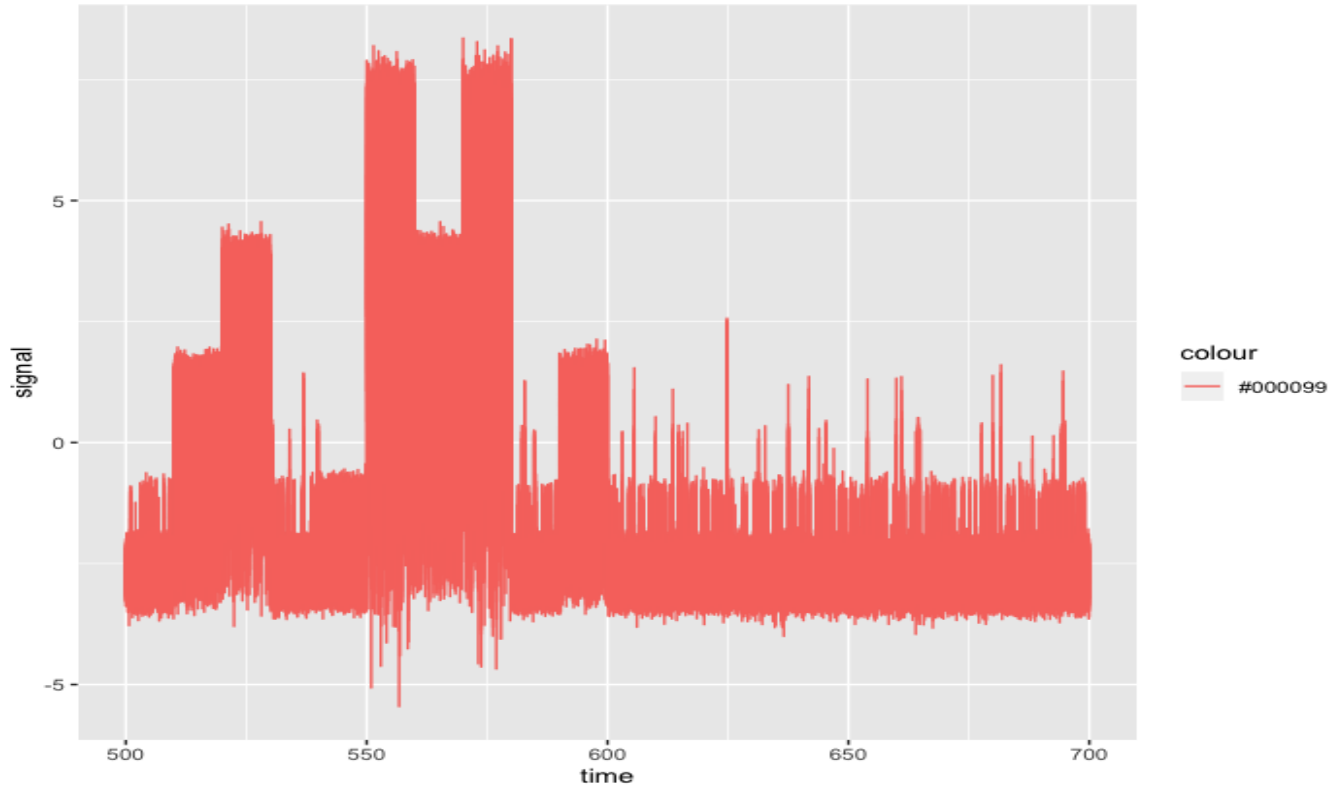


#Time VS Signal
#Train Data
> ggplot(data=trainData)+geom_line(aes(x=time,y=signal, color="#56B4E9"))

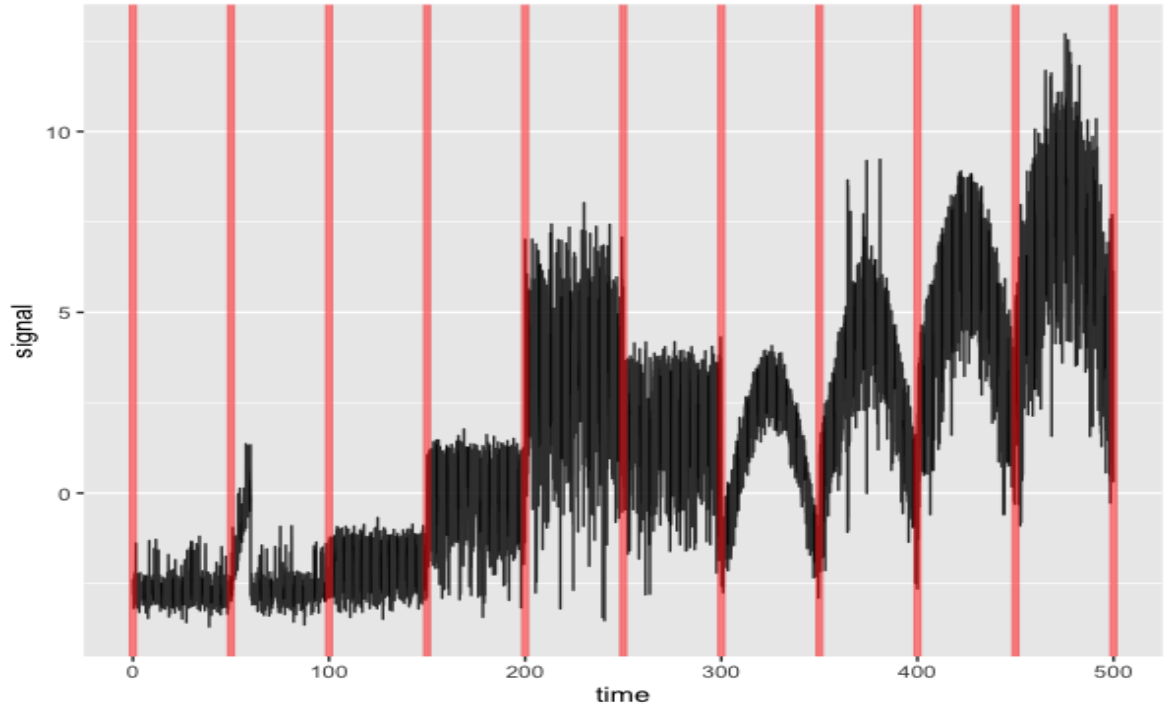


#Test Data

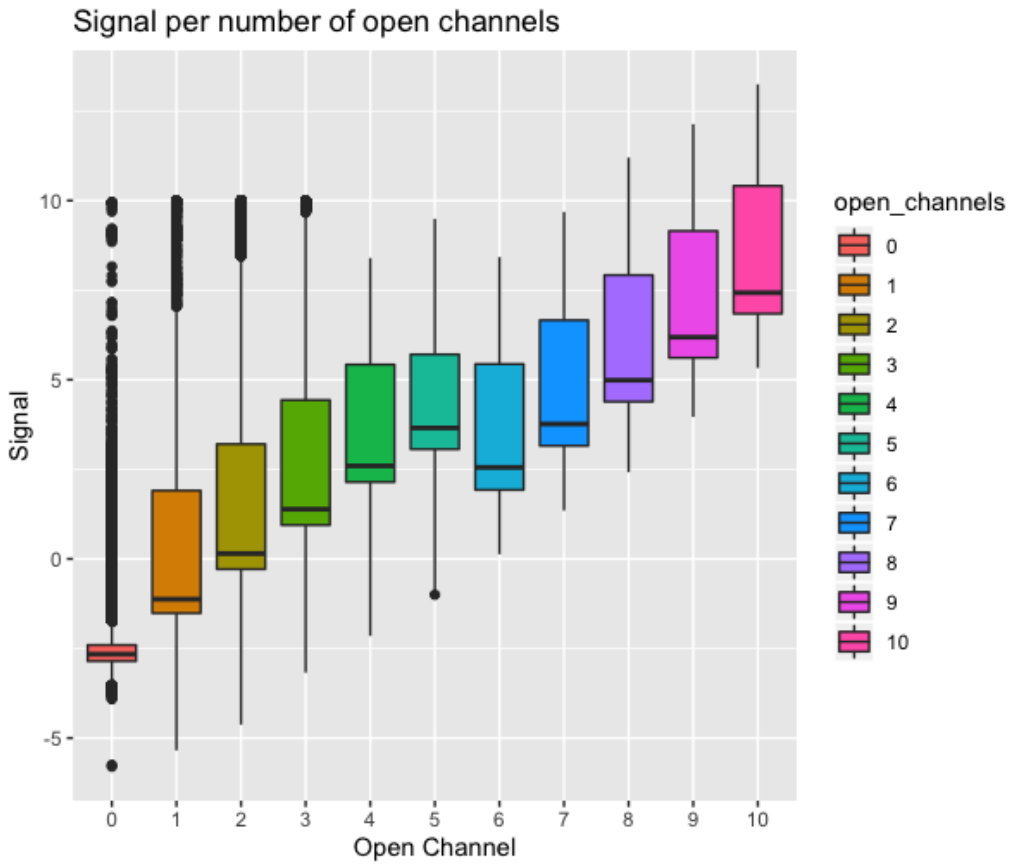
> ggplot(data=testData)+geom_line(aes(x=time,y=signal, color="#000099"))



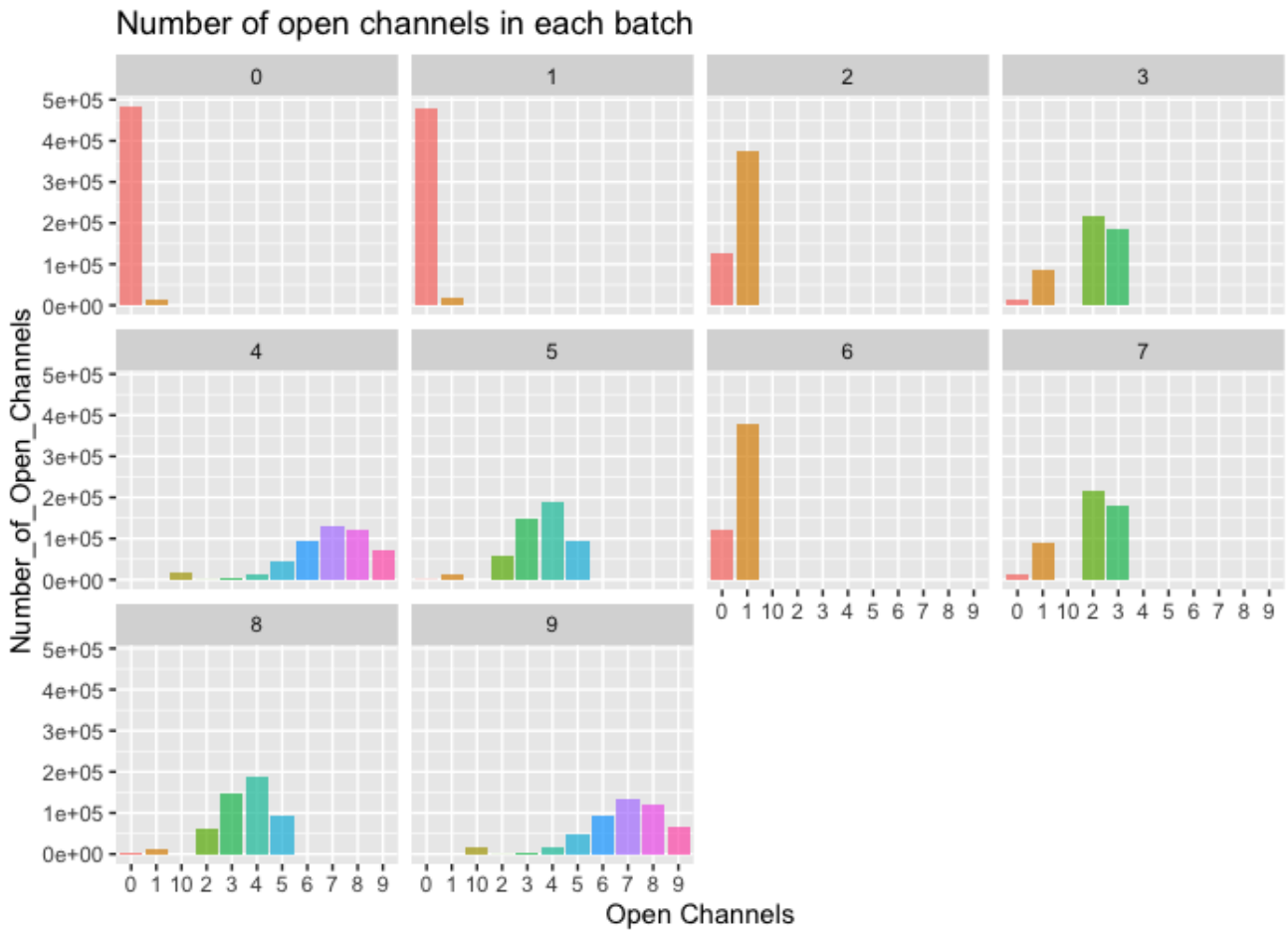
```
> trainData %>% sample_n(10000) %>%
+   ggplot(aes(x = time, y = signal))+
+   geom_line(color = "black", alpha = 0.7)+
+   geom_vline(xintercept = seq(0, 500, 50),size = 1.5, alpha = 0.7, color = "red")
```




```
> ggplot(trainData,aes(x=open_channels,y=signal,fill=open_channels))+ geom_boxplot()+
labs(title="Signal per number of open channels", x="Open Channel", y="Signal",
colour="open_channels") + scale_x_discrete(limits=as.character(0:10))
```




```
#Dividing the Train data into batches
trainData$batch<- as.factor((trainData$time - 0.0001)%/%50)
trainData$time_batch <- ((trainData$time - 0.0001)%%50)+0.0001

#Visualizing Number of Open Channels in Each Batch
trainData %>% count(batch, open_channels) %>% rename(Number_of_Open_Channels = n)
%>%  ggplot(aes(x = as.character(open_channels), y = Number_of_Open_Channels,
  fill = as.character(open_channels)))+
  geom_bar(stat = "identity", alpha = 0.7)+
  facet_wrap(~batch)+theme(legend.position = "none")+
  labs(title = "Number of open channels in each batch", x = "Open Channels")
```



MODEL IMPLEMENTATION:

We have used Random Forest model to predict the Open Channels. We have used it with 10% bagging of data. We choose to use Random Forest model because the data set was very heavy and it was unbalanced. The Random Forest model has fast training speed and quick prediction capability and can handle unbalanced data as it tries to minimize the overall error rate.

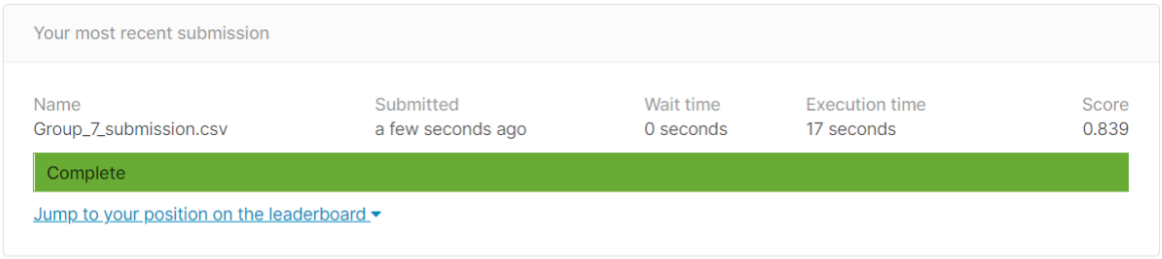
As model creation and prediction was done separately please refer to the attached R file "Final.R" for code.

```
start_time <- Sys.time()
suppressMessages(library(randomForest))
suppressMessages(library(glmnet))
suppressMessages(library(tidyverse))
suppressMessages(library(caret))
suppressMessages(library(dplyr))
suppressMessages(library(rpart))
library(e1071)
#here we are reading the data to train
train<-read.csv("train (1).csv")
```

```
#here we are reading the data to test
test<-read.csv("test (1).csv")
test<-test[,c(1,2)]

#we are taking the samples fomr the total training data.
x<-train[sample(nrow(train))[c(1:200000)],c(2,3)]
x_train<-x[sample(0.8*nrow(x)),c(1,2)]
x_train[,2]<-as.factor(x_train[,2])
x_val<-x[-sample(0.8*nrow(x)),c(1,2)]
x_val[,2]<-as.factor(x_val[,2])
t<-as.data.frame(x_val[,-2])
colnames(t)<-'signal'

#we are building the model using the random forest
model<-randomForest(open_channels~.,data =x_train,importance = TRUE)
pred<-predict(model,newdata=t,type="response")

#creating the confusion matrix and storing it in the result
result<-confusionMatrix(pred,x_val[,2])[4]
mean(result$byClass[,11])#val_accuracy:
#rm(list=ls())
preds = data.frame(c(11:2000010))
n= 40

#we are training the data with 40 samples.
for (i in 1:n){

  x<-train[sample(nrow(train))[c(1:100000)],c(2,3)]
  x_train<-x[sample(0.8*nrow(x)),c(1,2)]
  x_train[,2]<-as.factor(x_train[,2])
  model<-randomForest(open_channels~.,data =x_train,importance = TRUE)
  pred<-predict(model,newdata=subset(test),type="response")
  preds <-cbind(preds,pred)

}

colnames(preds)<-c(0:n)
preds<-preds[,c(2:n)]
head(preds)

#creating a CSV from the prediction data.
write.csv(preds,'Group_7_submission.csv',row.names = FALSE,quote = FALSE)
test<-test[,1]
y_final = read.csv("Group_7_submission.csv")
y_final<-apply(y_final, 1,median)
df<-data.frame(cbind(test,y_final))
colnames(df)<-c("time","open_channels")
write.csv(df,'Group_7_submission.csv',row.names = FALSE,quote = FALSE)
#hist(df$open_channels)
end_time <- Sys.time()
```

end_time - start_time
rm(list=ls())

KAGGLE SCREENSHOT:



Member Contribution :

1.Model Creation : Rohit ,Gagan and Darshan.
2.Data Visualization : Jasneek and Ashish.