

NJIT CS 636
DATA ANALYTICS WITH R

LIVERPOOL ION-SWITCHING DATASET PREDICTION

PROJECT GROUP 9
KOMALDEEP KAUR BHATIA (kb488)
CHANDNI MANDAVIYA (csm45)
SOURAV HELAPALYA ASWATHNARAYAN (sh667)
TWINKLE CHAURASIA (tc449)

Data Preprocessing

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

Here the data is having noise in the form of signal drift, so we tried to remove signal drift manually and then pass the below clean data to our models to get the accuracy.

To clean the data, we divided the data manually to several batches based on similarities between the signal classes.

And from the paper [here](#). We got to know that the data is synthesized. Also "electrophysiological" noise and drift were added. Drift is a signal bias causing the signal to no longer be a horizontal line like batches 2, 7, 8, 9, 10 above.

We tried to remove the parabolic drifts from the data in the classes 7,8,9,10 manually.

Below is the code to transform into clean data.

```
1 library(data.table)
2 library(dplyr)
3 data <- read.csv("../input/liverpool-ion-switching/train.csv",header=TRUE)
4 test_data <- read.csv("../input/liverpool-ion-switching/test.csv",header=TRUE)
5 train2 = copy(data)
6 a<-500000
7 b<-600000
8 train2[a:b,2] <- train2[a:b, 2] - (3*(train2$time[a:b] - 50)/10)
9 f <- function(x,low,high,mid){
10   return(-((-low+high)/625)*(x-mid)**2+high -low)
11 }
12
13 # CLEAN TRAIN BATCH 7
14 batch <- 7
15 a <- 500000*(batch-1)
16 b <- 500000*batch
17 train2[a:b,2] <- train2[a:b, 2] - f(data$time[a:b], -1.817,3.186,325)
18 # CLEAN TRAIN BATCH 8
19 batch <- 8
20 a <- 500000*(batch-1)
21 b <- 500000*batch
22 train2[a:b,2] <- train2[a:b, 2] - f(data$time[a:b],-0.094,4.936,375)
23 # CLEAN TRAIN BATCH 9
24 batch = 9;
25 a <- 500000*(batch-1)
26 b <- 500000*batch
27 train2[a:b,2] <- train2[a:b, 2] - f(data$time[a:b],1.715,6.689,425)
28 # CLEAN TRAIN BATCH 10
29 batch = 10;
30 a <- 500000*(batch-1)
31 b <- 500000*batch
32 train2[a:b,2] <- train2[a:b, 2] - f(data$time[a:b],3.361,8.45,475)
33 # Training batch 1 and 2(1 Slow Open Channel)
34 batch <- 1
35 a <- 500000*(batch-1)
36 b <- 500000*batch
37 batch <- 2
38 c <- 500000*(batch-1)
39 d <- 500000*batch
40 abc <- c(train2$signal[a:b],train2$signal[c:d])
41 X_train <- c()
```

19:11 | (Top Level) ⌵

Console

CLEANING THE DATA

MODELING PROCEDURE

By observing the data is divided into classes, hence it is multiclass Classification. Therefore, we tried with different algorithms such as KNN, k means, Random forest, decision tree, LSTM and Naive Bayes. But since the data has multiple classes, the best accuracy we got was in random forest after cleaning and removing the drift from data as shown above.

Also, Random Forest models are fast training models and are scalable.

The advantages of random forests include:

- The predictive performance can compete with the best supervised learning algorithms.
- They provide a reliable feature importance estimate.
- They offer efficient estimates of the test error without incurring the cost of repeated model training associated with cross-validation.

Below are the Models we tried to get better accuracy.

- Decision Tree
- Random Forest.
- K-Means
- Naïve Bayes
- LSTM

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help

Source
Console Terminal
~/Sim2/R/Finial Project/

Warning message:
package 'rpart.plot' was built under R version 3.6.3
> data <- read.csv("train.csv",header=TRUE)
> test_data <- read.csv("test.csv",header=TRUE,colClasses = c("character", "character"))
>
> shuffle_index <- sample(1:nrow(data))
> data <- data[shuffle_index, ]
> head(data)
      time  signal open_channels
4336957 433.6957  6.5649          4
4522250 452.2250  3.9794          7
3319066 331.9066  3.2020          1
2551536 255.1536  3.5222          5
1067596 106.7596 -2.3703          0
1190921 119.0921 -2.6357          0
> summary(data)
      time      signal  open_channels
Min.   : 0.0001   Min.   :-5.796   Min.    : 0.000
1st Qu.:125.0001  1st Qu.: -1.595   1st Qu.: 1.000
Median :250.0000  Median : 1.124   Median : 2.000
Mean   :250.0000  Mean   : 1.386   Mean   : 2.726
3rd Qu.:375.0000  3rd Qu.: 3.690   3rd Qu.: 4.000
Max.   :500.0000  Max.   :13.244   Max.   :10.000
> data <- transform(data, open_channels =as.factor(open_channels))
> sapply(data, class)
      time      signal  open_channels
"numeric"  "numeric"  "factor"
> data[ data == "?" ] <- NA
> colSums(is.na(data))
      time      signal  open_channels
      0          0          0
> data <- data[!(data$open_channels %in% c(NA)),]
> colSums(is.na(data))
      time      signal  open_channels
      0          0          0
> sample = sample.split(data$open_channels, SplitRatio = 0.85)
> train = subset(data, sample == TRUE)
> test = subset(data, sample == FALSE)
> dim(train)
[1] 4249999      3
> dim(test)
[1] 750001      3
> fit <- rpart(open_channels~., data = test, method = 'class')
> predictions_train = predict(fit, newdata=train[-3])
> predictions_test = predict(fit, newdata=test[-3])
> accuracy_train <- mean((colnames(predictions_train)[apply(predictions_train,1,which.max)]) == train$open_channels)
[1] 0.7520724
> accuracy_test <- mean((colnames(predictions_test)[apply(predictions_test,1,which.max)]) == test$open_channels)
[1] 0.752443
> test_data1 <- read.csv("test.csv",header=TRUE)
> predictions_test_data <- predict(fit, newdata=test_data1)
> test_data$open_channels <- (colnames(predictions_test_data)[apply(predictions_test_data,1,which.max)])
> str(test_data)
'data.frame':   2000000 obs. of  3 variables:
 $ time      : chr  "500.0001" "500.0002" "500.0003" "500.0004" ...
 $ signal    : chr  "-2.6496" "-2.6494" "-2.8600" "-2.4350" ...
 $ open_channels: chr  "0" "0" "0" "0" ...
> test_data$signal <- NULL
> write.table(test_data, file = 'dt_predicted_test_final.csv', sep=",",row.names=FALSE)
>

```

DECISION TREE IMPLEMENTATION

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source on Save
1 library(randomForest)
2 require(caTools)
3 data <- read.csv("train_clean.csv",header=TRUE)
4 test_data <- read.csv("test.csv",header=TRUE,colClasses = c("character", "character"))
5 shuffle_index <- sample(1:nrow(data))
6 data <- data[shuffle_index, ]
7 head(data)
8 summary(data)
9 data <- transform(data, open_channels =as.factor(open_channels))
10 sapply(data, class)
11 data[ data == "?" ] <- NA
12 colSums(is.na(data))
13 data <- data[!(data$open_channels %in% c(NA)),]
14 colSums(is.na(data))
15 sample = sample.split(data$open_channels, SplitRatio = .75)
16 train = subset(data, sample == TRUE)
17 test = subset(data, sample == FALSE)
18 dim(train)
19 dim(test)
15:58 (Top Level)
Console Terminal
~/Sem2/R/Final Project/
> rf <- randomForest(open_channels ~., data=train)
> predictions_train = predict(rf, newdata=train[-3])
> predictions_test = predict(rf, newdata=test[-3])
> accuracy_train <- mean(predictions_train == train$open_channels)
> accuracy_train
[1] 1
> accuracy_test <- mean(predictions_test == test$open_channels)
> accuracy_test
[1] 0.9572156
> predictions_test_data <- predict(rf, newdata=test_data)
> test_data$open_channels <- predictions_test_data
> str(test_data)
'data.frame': 2000000 obs. of 3 variables:
 $ time : chr "500.0001" "500.0002" "500.0003" "500.0004" ...
 $ signal : chr "-2.6498" "-2.8494" "-2.8600" "-2.4350" ...
 $ open_channels: Factor w/ 11 levels "0","1","2","3",...: 3 3 3 3 3 3 3 3 3 3 ...
> test_data$signal <- NULL
> write.table(test_data, file = 'rf_predicted_test_final.csv', sep="," ,row.names=FALSE)
> |
```

RANDOM FOREST IMPLEMENTATION

```
#####K-Means#####
#loading the require library
library(VIM)
library(tidyverse)
library(factoextra)
#loading the dataset
train <- read.csv('train.csv/train.csv')
test<-read.csv('test.csv/test.csv')

#to check if the data has any missing values or not
#aggr(train)

#distance <- get_dist(train)
k2 <- kmeans(train, centers = 5, nstart = 25)
str(k2)
# fviz_cluster(k2, data = train)

#to find optimal k
set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(train, k, nstart = 10 )$tot.withinss
}

#k=4 is the elbow point
set.seed(123)
model <- kmeans(train, 4, nstart = 25)

#prediction
fitted(model)
```

K-MEANS IMPLEMENTATION

```
In [27]: model <- keras_model_sequential()
```

```
In [39]: model %>%  
  layer_dense(units = 1, activation = 'softmax',  
  input_shape = 2)
```

```
In [48]: summary(model)
```

```
Model: "sequential_1"  
-----  
Layer (type)                Output Shape          Param #  
-----  
dense_2 (Dense)              (None, 1)              3  
-----  
dense_3 (Dense)              (None, 1)              2  
-----  
dense_4 (Dense)              (None, 1)              2  
-----  
Total params: 7  
Trainable params: 7  
Non-trainable params: 0  
-----
```

```
In [41]: sgd <- optimizer_sgd(lr = 0.01)
```

```
In [46]: model %>% compile(  
  loss = 'binary_crossentropy',  
  optimizer = sgd,  
  metrics = 'accuracy'  
)
```

```
In [ ]: ## model %>% fit(  
  x = traindata,  
  y = trainingtarget,  
  epochs = 1,  
  batch_size = 3,  
  validation_split = 0.2,  
  verbose = 0  
)
```

LSTM IMPLEMENTATION


```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source on Save
1 library(e1071)
2 library(caret)
3 library(caTools)
4 data<-read.csv(file="train.csv",header = TRUE)
5 test_data <- read.csv("test.csv",header=TRUE,colClasses = c("character", "character"))
6 shuffle_index <- sample(1:nrow(data))
7 data <- data[shuffle_index, ]
8 data<-transform(data,open_channels=as.factor(open_channels))
9 sapply(data,class)
10 data[data=="?"]<-NA
11 colSums(is.na(data))
12 split_values<-sample.split(data$open_channels,SplitRatio=0.85)
13 train<-subset(data,split_values==TRUE)
14 test<-subset(data,split_values==FALSE)
15 nb<-naiveBayes(open_channels~.,train)
16 predictions_train = predict(nb, newdata=train[-3])
17 predictions_test = predict(nb, newdata=test[-3])
18 accuracy_train <- mean(predictions_train == train$open_channels)
19 accuracy_train
20 accuracy_test <- mean(predictions_test == test$open_channels)
21 accuracy_test
22 predictions_test_data <- predict(nb, newdata=test_data)
23 test_data$open_channels <- predictions_test_data
24 str(test_data)
25 test_data$signal <- NULL
26 write.table(test_data, file = 'nb_predicted_test_final.csv', sep="," ,row.names=FALSE)
27
28
29
```

NAÏVE BAYES IMPLEMENTATION

KAGGLE LEADERBOARD SCORE.

Link to my account: <https://www.kaggle.com/komaldeepkaurbhatia>

Team Name on Kaggle: **CS636_Final_Project9**

≡

kaggle

🏠 Home

🏆 Compete

📊 Data


📓 Notebooks


💬 Discuss


🎓 Courses


⋮ More


Recently Viewed

 cleandata

 kernel4747f9f52

 Final project

 kernel10e1a97467

 University of Liverpool ...

🔍 Search

🔔

👤

Overview

Data

Notebooks

Discussion

Leaderboard

Rules


Team

My Submissions

Submit Predictions

1680

Olivia




0.927

1

7d

1681

Hao Chen




0.927

1

6d

1682

hiro22




0.927

1

5d

1683

Vasi




0.927

1

3d

1684

Daiki Katsuragawa




0.927

1

2d

1685

Rajaram Parab




0.927

1

2d

1686

Himanshu Chandra




0.927

3

2d

1687

Abhinav Kumar Sintoo







0.927

1

2d

1688

CS636_Final_Project9




0.927

6

1d





Your Best Entry

Your submission scored 0.927, which is an improvement of your previous score of 0.316. Great job!

 Tweet this!

1689

OPIR vvb




0.927

3

7h

1690

Naoto Kai




0.927

1

2mo

1691

Combinatorial Cat



0.927

2

2mo

KAGGLE LEADERBOARD SCORE

INDIVIDUAL CONTRIBUTIONS

KOMALDEEP KAUR BHATIA (kb488)

Participated in analyzing data and cleaning data for the model. Tried to get the best accuracy using Random Forest algorithm, Decision Tree modelling and XG Boost algorithm. Worked on understanding the distortion of the data and signals. Worked in making the Final report.

CHANDNI MANDAVIYA (csm45)

Tried making model of LSTM and K-Means. Participated in analyzing data and cleaning data for the model. Research on which models will be most fitted for this type of data. Worked in making the Final report.

SOURAV HELAPALYA ASWATHNARAYAN (sh667)

Tired making a model in Decision Trees and k-folds with logistics regression. Helped is developing the code and logic. Tried different methods to find the best accuracy. Participated in analyzing data and cleaning data for the model.

TWINKLE CHAURASIA (tc449)

Tried making the model through the KNN algorithm. Participated in analyzing data and cleaning data for the model. Worked on the Presentation documentation.