Configuration Manual

MSc Research Project
Masters in Data Analytics

Abhishek Angne
Student ID: x18136923

School of Computing
National College of Ireland

Supervisor: Mr. Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Forename Surname
Student ID:	XXX
Programme:	Programme Name
Year:	2018
Module:	MSc Research Project
Supervisor:	XXX
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	1002
Page Count:	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th December 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Abhishek Angne
x18136923

1 Introduction

A configuration manual represents the entire setup in terms of software, hardware, data collection, data analysis, machine learning models that were applied.

Human Activity Recognition is a field that is garnering massive demands due to requirements in the field of medicine, surveillance monitoring, fitness and injury rehabilitation and much more. (Malik; 2017), (Sakr et al.; 2018), (Civitarese et al.; 2019).

2 System Configuration

2.1 Hardware

1. Processor: Intel(R) Core(TM) i7-7700HQ CPU @2.80 Ghz,
2. RAM: 16.0 GB,
3. OS: Windows 10 64-bit,
4. Graphics: NVIDIA GeForce GTX 1060 Ti,
5. Storage: 1TB

2.2 Software

1. Microsoft Excel 2016: Excel was for performing data loading, grabbing a quick look, saving the data in a particular format.
2. Rstudio 3.6.1: RStudio's R Markdown was used for performing Exploratory Data Analysis in R. Feature Extraction and Model Evaluation were also performed. But due to higher computational requirements, the remainder of the project was continued on Kaggle kernels in the form of a notebook.
3. Kaggle Kernels Powered by Google Cloud Engine (GPU-enabled): The entire project was run on Kaggle Kernels which were used as an IaaS(Infrastructure as a Service) in the project, wherein, a lot of pre-loaded software and hardware requirements were met.
4. Canva: Canva was used to create certain visuals like the design specification and labelled to reuse and modification logos.

3 Development Phase of the Project

The development of the project was split into multiple phases like Data Understanding, Data Analysis followed Data Transformation, Data Pre-processing, Data Normalization, and all the other steps taken in order to make the dataset apt for applying models to the data.

There were several research papers surveyed before deciding the algorithms used in the research.

3.1 Data Preparation and Pre-processing

The Extrasensory dataset was created by Yonatan Vaizman and team. (Vaizman and Ellis; 2017) with the help of smartphone and smartwatch sensors. The phones that were owned by the users in the analysis were from both Android and Apple ecosystems. [1]

13 tar.gz files were chosen and .csv versions were extracted of these files using Microsoft Excel.

These .csv files were then loaded in RStudio using the read.csv command. These .csv files were then merged using timestamps and activities were loaded. Initially, 60 users were loaded for initial analysis. However, for the machine learning analysis, 13 users were chosen at random. The loaded dataset thus had over 77000 rows and 280 features. Activities in the columns were then melted to rows and the column considered for analysis, i.e. the dependent variable for our analysis. The cleaned dataset was then split into train and test splits, eventually.

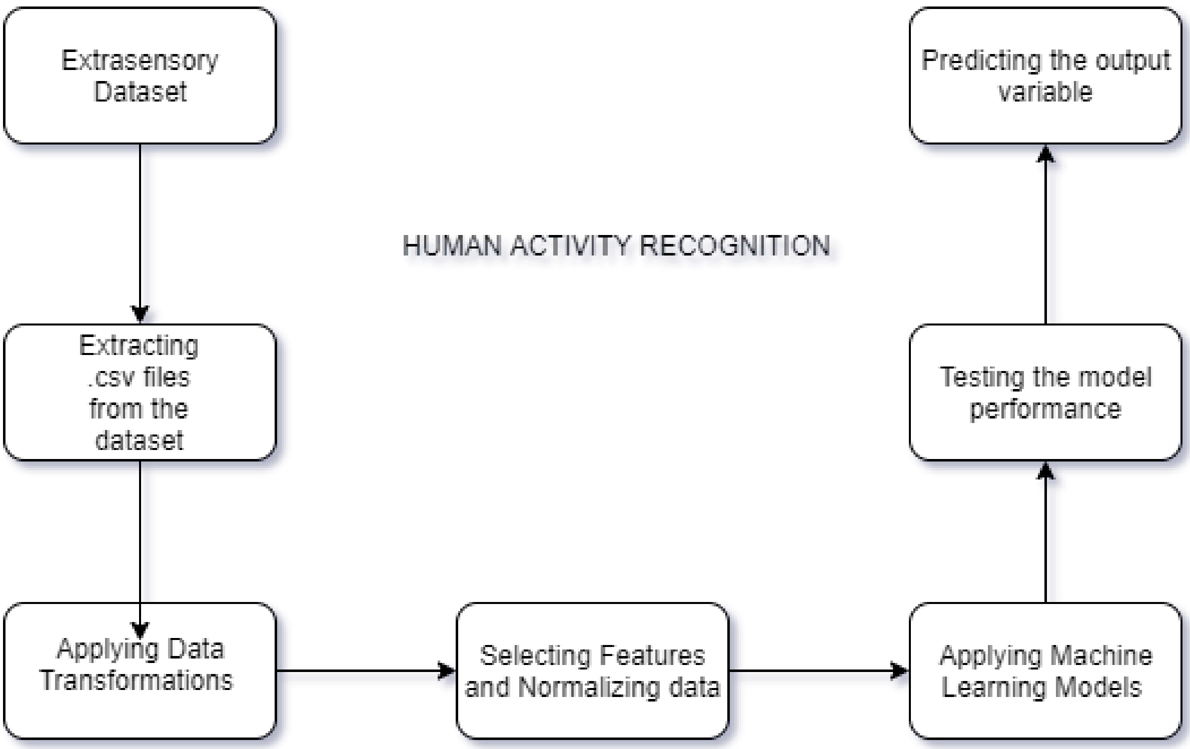

Figure 1: Workflow

[1]http://extrasensory.ucsd.edu/

Figure 1 represents the basic workflow followed in the implementation bit of this project.

3.1.1 Libraries

```
In [2]:
#Importing the necessary libraries
library(tidyverse)
library(lubridate)
library(reshape2)
library(dplyr)
library(corrplot)
library(ggplot2)
library(anytime)
library(scales)
library(pvclust)
library(caret)
library(cluster)
#library(factoextra)
library(mlbench)
library(moments)
library(gridExtra)
library(rpart)
library(partykit)
library(e1071)
library(randomForest)
library(rattle)
library(keras)
```

Figure 2: Libraries Used

Figure 2 displays the different libraries used in R for the project.

1. The data was first loaded into a list called Templist.

   Figure 3 represents the temporary list after loading the dataset.

```
[ ]  TempList = list.files(path = "../input/templist2/", pattern = '.*csv')
```

```
[ ]  data <- (read.csv(file=paste("../input/templist2/",TempList[1], sep=""), header=TRUE, sep=","))
```

```
[ ]  id <- str_extract(TempList[1],"^[A-Z|0-9-]*")
     idColumn <-  data.frame(c(rep(id,nrow(data))))
     idUser <- idColumn
     colnames(idUser)[1] <- "idUser"

     for(i in 2:length(TempList)){
       temp <- (read.csv(file=paste("..//input/templist2//", TempList[i], sep=""), header=TRUE, sep=","))
       id <- str_extract(TempList[i],"^[A-Z|0-9-]*")
       idColumn <- data.frame(c(rep(id,nrow(temp))))
       colnames(idColumn)[1] <- "idUser"
       idUser <- rbind(idUser, idColumn)
       data <- rbind(data, temp)
     }
```

Figure 3: Templist

2. Activities for the classification were chosen. A multi-class classification required more than 2 activities. After a thorough reading of the dataset base paper, and understanding the context of data collection, four simple categories were chosen for classification. Figure 4 represents the four activities chosen.

```
[ ] #Coding the output

    ## Looking at the labels with the number of minutes/examples spend by all the users

    labels <- data

    # From the given set of activities, we will classify an individuals activity based on four prominent actions and the remaining will be classiified as a separate activity
    labels <- select(data, (c('label.SITTING', 'label.FIX_walking', 'label.FIX_running', 'label.BICYCLING', 'label.LYING_DOWN', 'label.PHONE_ON_TABLE')))

    z <- 1
    code.exit <- c()


    for(i in 1:nrow(labels)){
      next_tag <- 0
      for(j in 1:ncol(labels)){
        if (labels[i,j] == 1){
          if(next_tag == 1){
            code.exit[z] <- paste(code.exit[z] , colnames(labels[j]), sep='+');
          }
          else{
            code.exit[z] <-colnames(labels[j]);
          }

          next_tag <- 1

        }


      }
      if(next_tag == 0) code.exit[z] <- "Other activity"
      z<-z+1;
    }
```

hist(labels$code.exit = "Distribution of the selected activities", xlab = "Number of Minutes spent performing the selected activities")

Figure 4: Recoding

3. Features consisting more than 70% NA values were removed as too many features would have led to over-fitting of data.

   Figure 5 represents how NA values were treated.

```
[ ] getmode <- function(v) {
      Vuniq <- unique(v)
      Vuniq[which.max(tabulate(match(v, Vuniq)))]
    }

    columns_eliminate <- c()
    data <- data[,colSums(is.na(data))<nrow(data)]
    j <- 0
    for(i in 1:ncol(data)){
      if(sum(is.na(data[,i])) > 0){
        if(((sum(is.na(data[,i]))*100)/(sum(!is.na(data[,i]))+sum(is.na(data[,i])))) >= 70 ){
          cat("The columns removed as they exceeding the threshold of NA's",colnames(data[i]),"(>=70% NA's)\n")
          columns_eliminate[j] <- i;
          j<-j+1;
        }
      }
    }

    print(columns_eliminate)

    for(i in 1:ncol(data)){
      if(startsWith(as.character(colnames(data[i])), "label")){
        if((getmode(data[,i])) == 'NaN') data[is.na(data[,i]), i] <- 0
        else data[is.na(data[,i]), i] <- getmode(data[,i])

      }
      else{
        data[is.na(data[,i]), i] <- mean(data[,i], na.rm = TRUE)
      }
    }

    data_individual <- cbind(data, idUser)
    data <- arrange(data, data$timestamp)
```

```
The columns removed as they exceeding the threshold of NA's lf_measurements.pressure (>=70% NA's)
The columns removed as they exceeding the threshold of NA's lf_measurements.relative_humidity (>=70% NA's)
The columns removed as they exceeding the threshold of NA's lf_measurements.temperature_ambient (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.LAB_WORK (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.STROLLING (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.DOING_LAUNDRY (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.WASHING_DISHES (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.AT_A_PARTY (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.AT_A_BAR (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.SINGING (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.AT_THE_GYM (>=70% NA's)
The columns removed as they exceeding the threshold of NA's label.STAIRS_-_GOING_DOWN (>=70% NA's)
 [1] 215 218 233 249 253 254 257 258 260 267 269
```

Figure 5: Treating NA values

4. Sensors were chosen for performing the analysis, wherein, sensor values were going to be considered for training the model as per the label and then predicting the activity on the test data. Figure 6 represents the process of sensors selection.

```r
sensors <- select(data, matches("(^raw_acc|^proc_gyro|^raw_magnet|^watch_acceleration|^watch_heading|^location|^location_quick_features|^audio_naive|^audio_properties|^discrete|^lf_measurements)
nameSensor <- "empty"
typeSensor <- "Na"
for(i in 1:length(sensors)){
  if(nameSensor == unlist(strsplit(names(sensors[i]), ".", fixed = TRUE))[[1]]){
    next;
  }

  nameSensor <- unlist(strsplit(names(sensors[i]), ".", fixed = TRUE))[[1]]
  cat(sprintf("%-40s %s\n",names(sensors[i]),typeof(sensors[[i]])))
}
```

```
raw_acc.magnitude_stats.mean                double
proc_gyro.magnitude_stats.mean              double
raw_magnet.magnitude_stats.mean             double
watch_acceleration.magnitude_stats.mean     double
watch_heading.mean_cos                      double
location.num_valid_updates                  double
location_quick_features.std_lat             double
audio_naive.mfcc0.mean                      double
audio_properties.max_abs_value              double
discrete.app_state.is_active                double
lf_measurements.light                       double
discrete.time_of_day.between0and6           double
```

As you can see the sensors (accelerometer and gyroscope), they have both positive and negative correlations between attributes. This makes us think that by applying some feature selection technique we could obtain the most representative variables.

Since we have correlations, I will execute the algorithm of simple elimination of correlation variables by threshold, the objective is to remain with fewer variables since with many variables an overfit can be produced. In order to execute it, the algorithm must not receive constant variables (standard deviation 0), nor columns with null values, which we have already discussed

Figure 6: Selecting Sensors

5. Feature selection was then applied to check which values were features were relevant and which features had the issue of multi-collinearirty. Features that were heavily correlated amidst each other. A total of 44 variables were eliminated from the overall features selected.

   Figure 7 represents the feature selection script.

```r
sensors <- select(data, matches("(^raw_acc|^proc_gyro|^raw_magnet|^watch_acceleration|^watch_heading|^location|^location_quick_features|^audio_naive|^audio_properties|^discrete|^lf_measurements)"))
nameSensor <- "empty"
typeSensor <- "Na"
for(i in 1:length(sensors)){
  if(nameSensor == unlist(strsplit(names(sensors[i]), ".", fixed = TRUE))[[1]]){
    next;
  }

  nameSensor <- unlist(strsplit(names(sensors[i]), ".", fixed = TRUE))[[1]]
  cat(sprintf("%-40s %s\n",names(sensors[i]),typeof(sensors[[i]])))
}

sensors <- select(data, matches("(^proc_gyro|^raw_acc)"))
sensors[is.na(sensors)] <- 0
oc_cor <- cor(sensors)
oc_cor[is.na(oc_cor)]  <- 0
corrplot(oc_cor, method="color",type = "full", order = "hclust", tl.col = "black", tl.cex = 0.6 )
```

```
raw_acc.magnitude_stats.mean                double
proc_gyro.magnitude_stats.mean              double
raw_magnet.magnitude_stats.mean             double
watch_acceleration.magnitude_stats.mean     double
watch_heading.mean_cos                      double
location.num_valid_updates                  double
location_quick_features.std_lat             double
audio_naive.mfcc0.mean                      double
audio_properties.max_abs_value              double
discrete.app_state.is_active                double
lf_measurements.light                       double
discrete.time_of_day.between0and6           double
```

Figure 7: Feature Selection Code

Figure 8 represents the process of looking at features with higher correlation.

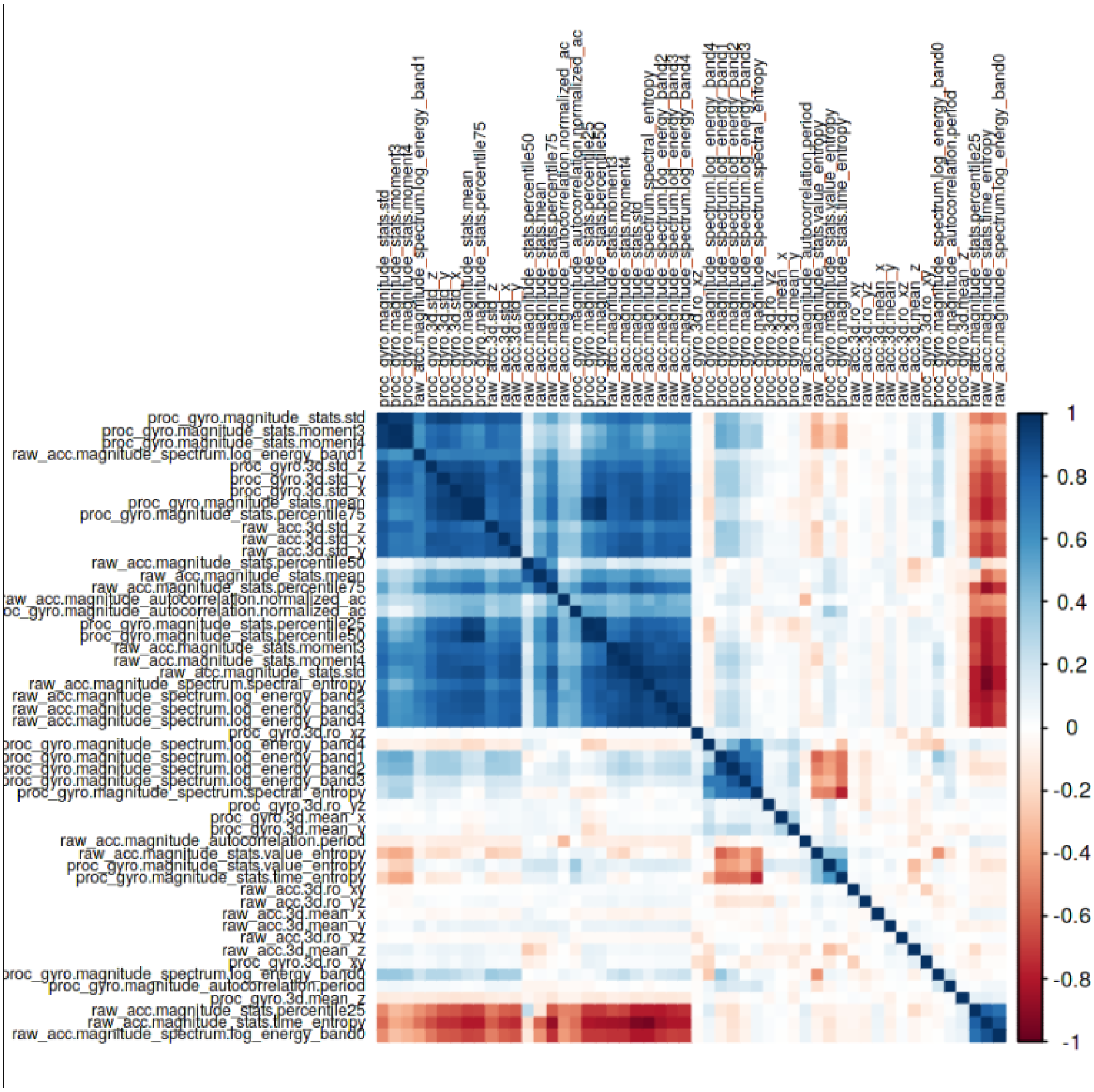

Figure 8: Feature Correlation

6. Other feature selection algorithms were also experimented with to check the actual requirement of features in terms of classifying the model.

   Figure 9 represents the Variable Importance Plot

```{r}
set.seed(123)
data <- data_all
data_all.rf <- randomForest(code.exit ~ ., data=data_all, ntree=1000, keep.forest=FALSE,
                        importance=TRUE)

varImpPlot(data_all.rf)
```

Figure 9: Variable Importance Plot Code

Figure 10 represents the Variable Importance Plot

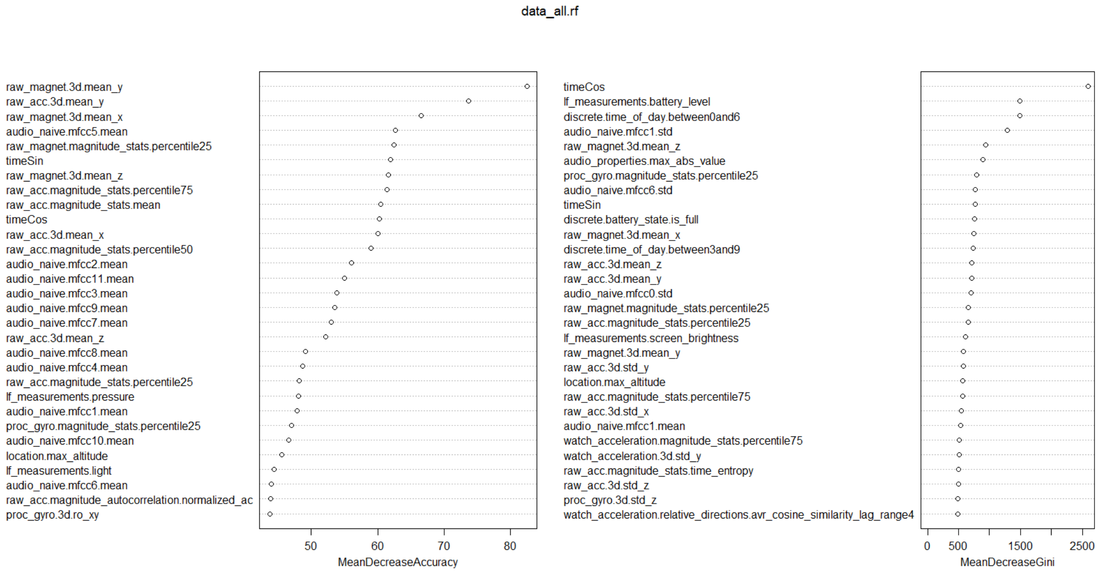

Figure 10: Feature Correlation

7. Skewness-Kurtosis calculation: In case we get a positive skewness that is greater than zero, then the tail point of the distrbution plot will point towards the right.[2]

   Figure 11 represents the distribution of sensor data and checking the distribution plot.

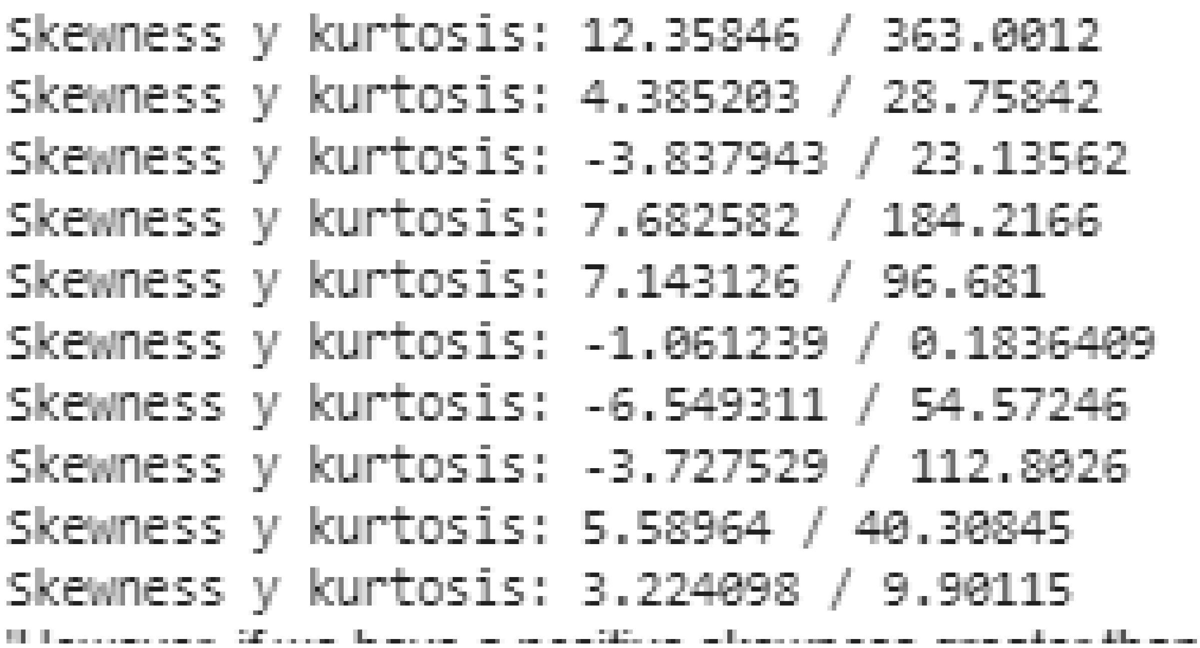
Skewness y kurtosis: 12.35846 / 363.0012
Skewness y kurtosis: 4.385203 / 28.75842
Skewness y kurtosis: -3.837943 / 23.13562
Skewness y kurtosis: 7.682582 / 184.2166
Skewness y kurtosis: 7.143126 / 96.681
Skewness y kurtosis: -1.061239 / 0.1836409
Skewness y kurtosis: -6.549311 / 54.57246
Skewness y kurtosis: -3.727529 / 112.8026
Skewness y kurtosis: 5.58964 / 40.30845
Skewness y kurtosis: 3.224098 / 9.90115

Figure 11: Skewness$_{K}urtosis$

[2] http://www.sthda.com/english/wiki/beautiful-dendrogram-visualizations-in-r-5-must-known-methods-unsupervised-machine-learning

8. The Kurtosis gives us an insight of how weeping the data structure is. In simpler words, if it is on a positive side, the higher the value of the kurtosis, the more are weights are the data. Skewness and Kurtosis are taken from the moments library in R.

9. Using ggplot to visualize the distribution.

```
ggplot(features, aes(x=features$raw_acc.magnitude_stats.moment3)) + geom_density()
ggplot(features, aes(x=features$raw_acc.magnitude_stats.moment3)) + geom_histogram(aes(y=..density.., fill=..count..))
+ stat_function(fun=dnorm,color="red",args=list(mean=mean(features$raw_acc.magnitude_stats.moment3),sd=sd(features$raw_acc.magnitude_stats.moment3)))
qqnorm(features$raw_acc.magnitude_stats.moment3)
```

Figure 12: ggplot

10. Hierarchical clustering has been performed using the agglomerative method (AGNES). Clustering is a method utilized to message similar points to identify groups. These activities are generally those wherein user defined labels need to be referred and we are working on feature labels which were application defined.

```
#We can see groupings that we have detected before visualizing the data.
set.seed(123)
oc <- select(data, starts_with('label'))
oc <- select(oc, -starts_with('label_source'))
oc[is.na(oc)] <- 0
oc <- oc[, colSums(oc) > 1]
#oc <- t(oc)
oc.pre <- preProcess(oc,method="scale")
oc.scaled <- predict(oc.pre, oc)
oc.diana <- agnes(t(oc.scaled), metric="euclidean")
pltree(oc.diana,cex=.6)

#Here I set K equal to 8, to point out the clusters (It's a random value)

set.seed(123)
oc <- select(data, starts_with('label'))
oc <- select(oc, -starts_with('label_source'))
oc[is.na(oc)] <- 0
oc <- oc[, colSums(oc) > 1] #I avoid columns without more than one example
#oc <- t(oc)
oc.pre <- preProcess(oc,method="scale")
oc.scaled <- predict(oc.pre, oc)
oc.agnes <- agnes(t(oc.scaled), metric="euclidean")
pltree(oc.agnes, hang=-1, cex = 0.6)
```

Figure 13: Hierarchial Clustering

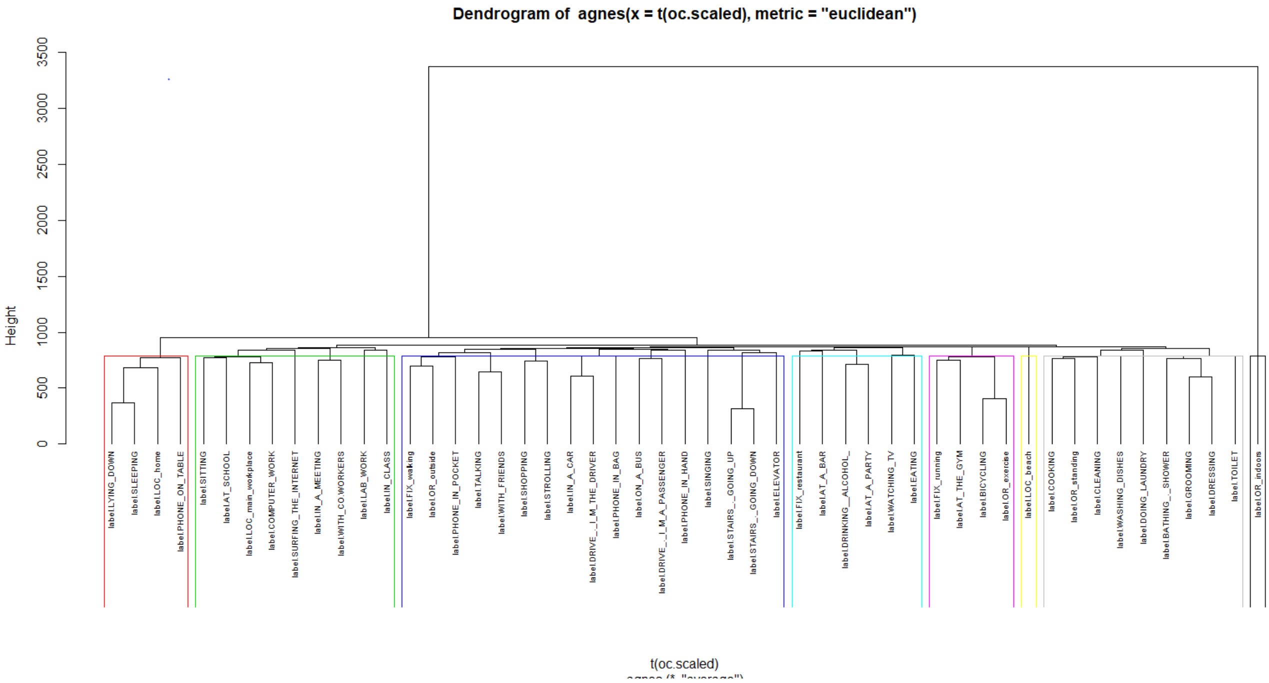

Figure 14: Hierarchial Clustering

11. Transforming the time variable, that is, the timestamp column, to a format wherein the continuity of time would only be maintained with a combination of a cosine and sine time component. Individual cosine and sine plots were not able to complete the cyclic requirement of time.

12. Recoding the output:

```
set.seed(123)
seconds_in_day = 24*60*60
time1 <- as.data.frame(sin(2*pi*data["timestamp"]/seconds_in_day))
time2 <- as.data.frame(cos(2*pi*data["timestamp"]/seconds_in_day))
colnames(time1) <- "timeSin"
colnames(time2) <- "timeCos"
cyclic_time <- cbind(time1, time2)
data<-cbind(cyclic_time, data)
ggplot(data, aes(timeSin))+geom_density()
ggplot(data, aes(timeCos))+geom_density()
ggplot(data, aes(timeSin, timeCos))+geom_point()
hours <- anytime(data$timestamp, tz="PST8PDT")
hours <- format(as.POSIXct(hours, "%Y-%m-%d %H:%M:%S", tz = ""), format = "%Y-%m-%d %H:%M")
hours <- format(as.POSIXct(hours, "%Y-%m-%d %H:%M", tz = ""), format = "%H")

set.seed(123)
timeSin <- select(data, starts_with('timeSin'))
timeCos <- select(data, starts_with('timeCos'))
labels <- select(labels, -starts_with("label_source"))
```

Figure 15: Timestamp Transformation

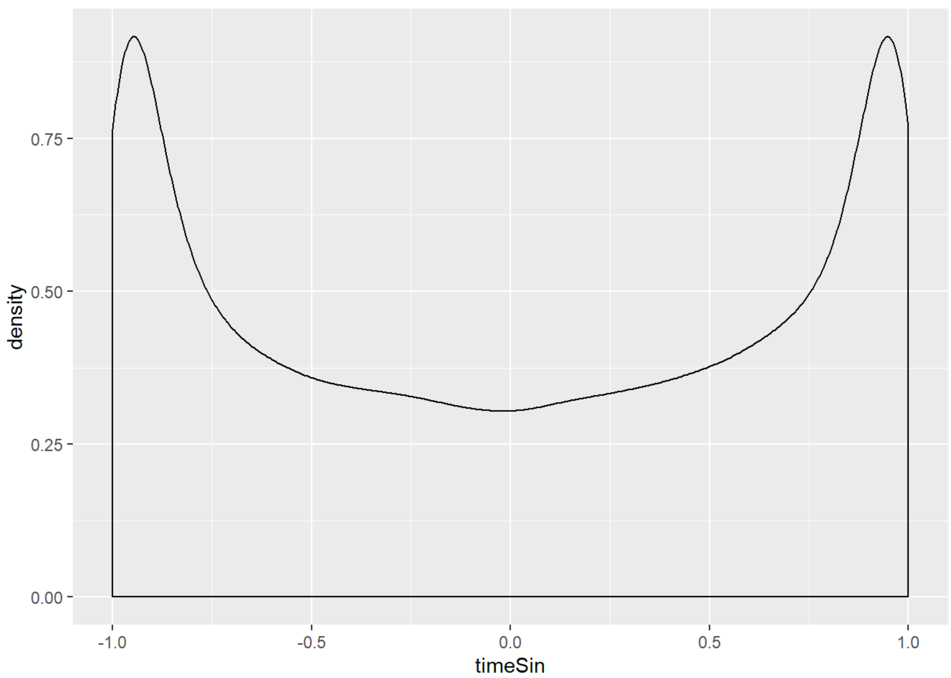

Figure 16: Timestamp Sin Transformation

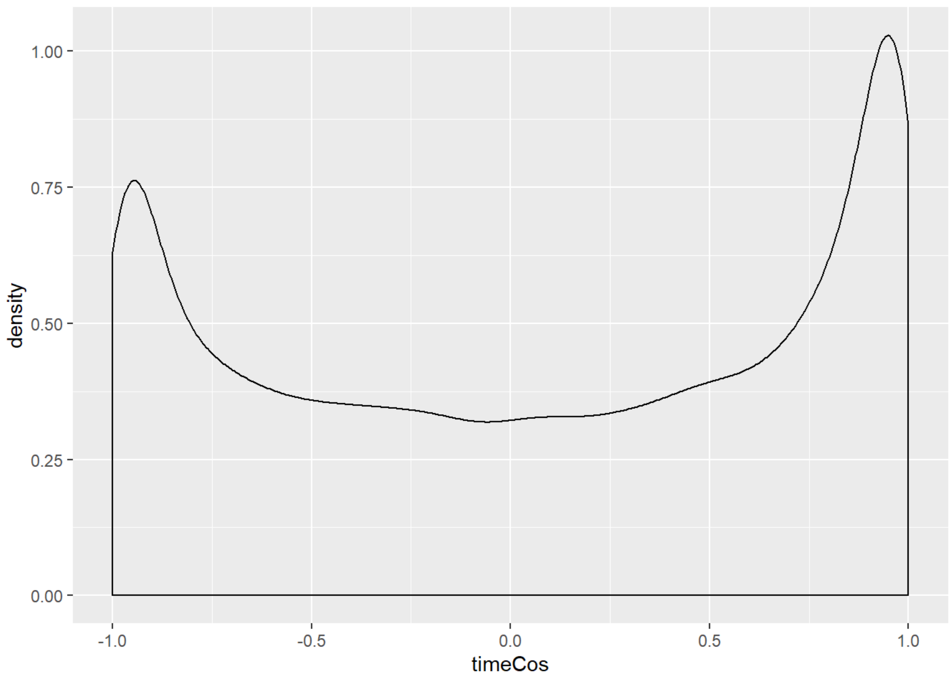

Figure 17: Timestamp Cos Transformation

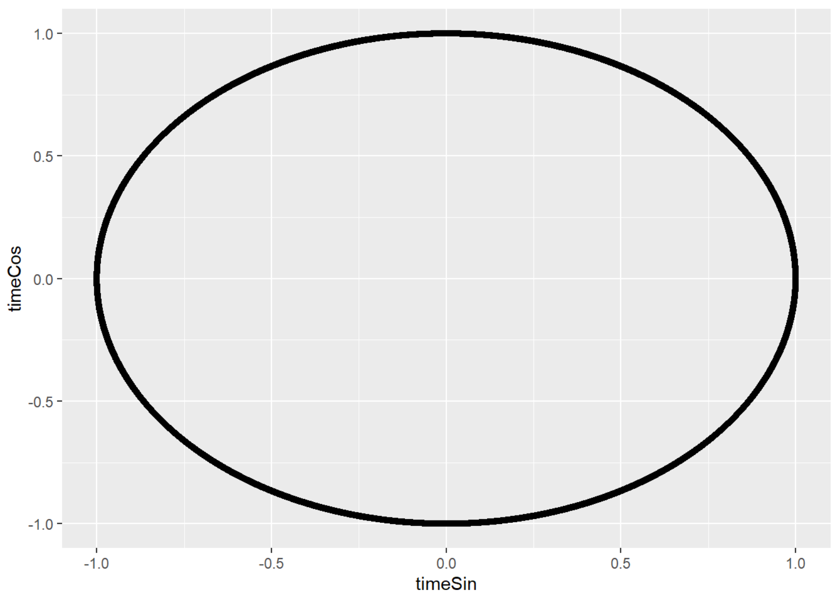

Figure 18: Timestamp SinCos Transformation

The activities are then split between 4 classes namely,'label.SITTING', 'label.FIX_walking', 'label.LYING_DOWN', and 'Other activity'.

13. Applying Normalization and Applying Random Forest and visualizing the output(Library used: randomForest) (Gao et al.; 2019), (Xu et al.; 2019), (Zhang et al.; 2019), (Can S; 2019), and (Hülsmann et al.; 2018)

Recoding the output

set.seed(123) data <- data_all data_all.rf <- randomForest(code.exit ~ ., data=data_all, ntree=1000, keep.forest=FALSE, importance=TRUE)

varImpPlot(data_all.rf)**

```
preProcMod<-preProcess(data_all[colnames(features)],method=c("center","scale"))
# Comment
data_all.Transf<-predict(preProcMod,data_all)
Vars_Enter <- colnames(select(data_all.Transf, -starts_with('code.exit')))
```

```
library(caret)
set.seed(123)
train.index <- createDataPartition(data_all.Transf$code.exit, p = .7, list = FALSE)
train <- data_all.Transf[ train.index,]
test  <- data_all.Transf[-train.index,]
#dfte<-data_all.Transf[1:10000,]
#randomForest <- randomForest( x= data_all.Transf[Vars_Enter], y = as.factor(data_all.Transf$code.exit),n_tree=3)
rf <- randomForest(x=train[1:178],y=as.factor(train[,179]))
cat("Random forest information: ")
print(rf)
saveRDS(rf, "rf.rds")
```

```
ypred<-predict(rf,test[1:178])

tstab<-table(test[,179],ypred)

confusionMatrix(tstab)
```

Figure 19: Random Forest Code

```
Confusion Matrix and Statistics

                   ypred
                    label.FIX_walking label.LYING_DOWN label.SITTING
  label.FIX_walking              1054                1           324
  label.LYING_DOWN                  0             6397           163
  label.SITTING                   116               25          7757
  Other activity                  158               56           712
                   ypred
                    Other activity
  label.FIX_walking            299
  label.LYING_DOWN             135
  label.SITTING                520
  Other activity              5515

Overall Statistics

               Accuracy : 0.892
                 95% CI : (0.8879, 0.896)
    No Information Rate : 0.3855
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8454

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: label.FIX_walking Class: label.LYING_DOWN
Sensitivity                           0.79367                  0.9873
Specificity                           0.97151                  0.9822
Pos Pred Value                        0.62813                  0.9555
Neg Pred Value                        0.98729                  0.9950
Prevalence                            0.05716                  0.2789
Detection Rate                        0.04537                  0.2754
Detection Prevalence                  0.07223                  0.2882
Balanced Accuracy                     0.88259                  0.9848
                     Class: label.SITTING Class: Other activity
Sensitivity                        0.8661                0.8525
Specificity                        0.9537                0.9448
Pos Pred Value                     0.9215                0.8562
Neg Pred Value                     0.9191                0.9432
Prevalence                         0.3855                0.2785
Detection Rate                     0.3339                0.2374
Detection Prevalence               0.3623                0.2772
Balanced Accuracy                  0.9099                0.8986
```

Figure 20: Random Forest Classification Matrix

14. Applying XGBoost and visualizing the output (Library used: xgboost) (Li et al.; 2019) ()

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1 1035    2  119  184
         2    3 6449   34  104
         3  324  124 7704  811
         4  316  120  561 5342

Overall Statistics

               Accuracy : 0.8837
                 95% CI : (0.8795, 0.8878)
    No Information Rate : 0.3623
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8335

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.61681   0.9633   0.9152   0.8294
Specificity           0.98585   0.9915   0.9150   0.9406
Pos Pred Value        0.77239   0.9786   0.8595   0.8427
Neg Pred Value        0.97063   0.9852   0.9500   0.9349
Precision             0.77239   0.9786   0.8595   0.8427
Recall                0.61681   0.9633   0.9152   0.8294
F1                    0.68588   0.9709   0.8865   0.8360
Prevalence            0.07223   0.2882   0.3623   0.2772
Detection Rate        0.04455   0.2776   0.3316   0.2299
Detection Prevalence  0.05768   0.2837   0.3858   0.2729
Balanced Accuracy     0.80133   0.9774   0.9151   0.8850
```

Figure 21: XGB$_C LM$

15. Applying Multilayer Perceptron and visualizing the output (Library used: keras)

```
model %>%
layer_dense(units = 128, activation = layer_activation_leaky_relu(), input_shape = (178)) %>%
layer_dropout(rate=0.2) %>%
layer_dense(units = 256)  %>%
layer_dropout(rate=0.2) %>%
layer_dense(units = 4, activation = 'softmax')
summary(model)
model %>% compile(
loss = 'sparse_categorical_crossentropy',
optimizer = optimizer_adam(),
metrics = c('accuracy'))
#'sparse_categorical_crossentropy'
```

```
Model: "sequential"
________________________________________________________________________________
Layer (type)                     Output Shape                      Param #
================================================================================
dense (Dense)                    (None, 128)                       22912
________________________________________________________________________________
dropout (Dropout)                (None, 128)                       0
________________________________________________________________________________
dense_1 (Dense)                  (None, 256)                       33024
________________________________________________________________________________
dropout_1 (Dropout)              (None, 256)                       0
________________________________________________________________________________
dense_2 (Dense)                  (None, 4)                         1028
================================================================================
Total params: 56,964
Trainable params: 56,964
Non-trainable params: 0
```

Figure 22: MLP

16. Epochs vs Loss Plot

```
[ ]
    vymax = max(c(history$metrics$loss,history$metrics$val_loss))
    plot(history$metrics$loss,main="Training/Validation errors for Extrasensory",col="blue",
         type="l",xlab="Epochs",ylab="Loss",ylim=c(0,vymax))
    lines(history$metrics$val_loss,col="red")
```

Figure 23: Epochs vs Loss Plot code

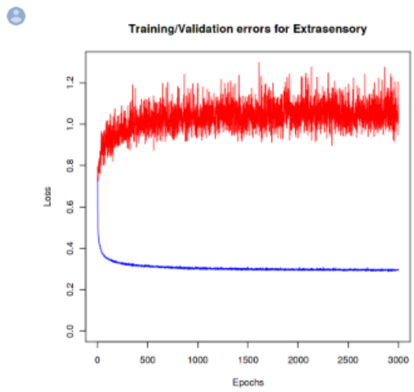

Figure 24: Epochs vs Loss Plot

17. At 3000 Epochs, the improvement in MLP performance.

```
history
```

```
Trained on 18,585 samples (batch_size=128, epochs=3,000)
Final epoch (plot to see history):
     acc: 0.9155
    loss: 0.2213
 val_acc: 0.6662
val_loss: 1.247
```

Figure 25: Epochs vs Loss Plot

References

Can S, E. (2019). Continuous Stress Detection Using Wearable Sensors in Real Life: Algorithmic Programming Contest Case Study.

Civitarese, G., Bettini, C., Sztyler, T. and Riboni, D. (2019). newNECTAR : Collaborative active learning for knowledge-based probabilistic activity recognition , *Pervasive and Mobile Computing* **56**: 88–105.
URL: *https://doi.org/10.1016/j.pmcj.2019.04.006*

Gao, X., Luo, H., Wang, Q., Zhao, F., Ye, L. and Zhang, Y. (2019). A human activity recognition algorithm based on stacking denoising autoencoder and lightGBM, *Sensors (Switzerland)* **19**(4).

Hülsmann, F., Göpfert, J. P., Hammer, B., Kopp, S. and Botsch, M. (2018). Classification of motor errors to provide real-time feedback for sports coaching in virtual reality — A case study in squats and Tai Chi pushes, *Computers and Graphics (Pergamon)* **76**: 47–59.

Human Action Recognition using Image Processing and Artificial Neural Networks (2013). **80**(9): 31–34.

Li, H., Pu, B., Kang, Y. and Lu, C. Y. (2019). Research on massive ECG data in XGBoost, *Journal of Intelligent and Fuzzy Systems* **36**(2): 1161–1169.

Malik, O. A. (2017). IMECE2012-87809, pp. 1–10.

Sakr, N. A., Abu-Elkheir, M., Atwan, A. and Soliman, H. H. (2018). Current trends in complex human activity recognition, *Journal of Theoretical and Applied Information Technology* **96**(14): 4564–4583.

Vaizman, Y. and Ellis, K. (2017). Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches.

Xu, S., Tang, Q., Jin, L. and Pan, Z. (2019). A Cascade Ensemble Learning Model for Human Activity Recognition with Smartphones, pp. 1–18.

Zhang, H. B., Zhang, Y. X., Zhong, B., Lei, Q., Yang, L., Du, J. X. and Chen, D. S. (2019). A comprehensive survey of vision-based human action recognition methods, *Sensors (Switzerland)* **19**(5): 1–21.