**GALATASARAY UNIVERSITY**

**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

# USING BEHAVIORAL BIOMETRIC SENSORS OF MOBILE PHONES FOR USER AUTHENTICATION

**Nurhak KARAKAYA**

June 2019

# USING BEHAVIORAL BIOMETRIC SENSORS OF MOBILE PHONES FOR USER AUTHENTICATION

by

**N u r h a k   K a r a k a y a ,  B . S .**

**Thesis**

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

**MASTERS of SCIENCE**

**in**

**COMPUTER ENGINEERING**

**in the**

**GRADUATE SCHOOL OF SCIENCE AND ENGINEERING**

**of**

**GALATASARAY UNIVERSITY**

Supervisor: Assoc. Prof. Dr. Gülfem Işıklar Alptekin

June 2019

This is to certify that the thesis entitled

**USING BEHAVIORAL BIOMETRIC SENSORS OF MOBILE PHONES FOR USER AUTHENTICATION**

prepared by **Nurhak KARAKAYA** in partial fulfillment of the requirements for the degree of **Master in Computer Engineering** at the **Galatasaray University** is approved by the

**Examining Committee:**

Assoc. Prof. Dr. Gülfem Işıklar Alptekin (Supervisor)
**Department of Computer Engineering**
**Galatasaray University** -------------------------

Dr. Günce Keziban Orman
**Department of Computer Engineering**
**Galatasaray University** -------------------------

Dr. Ayşe Tosun Kühn
**Department of Computer Engineering**
**Istanbul Technical University** -------------------------

Date: -------------------------

Date: ------------------------

## ACKNOWLEDGEMENTS

I very appreciate to first of all my thesis supervisor Gülfem Işıklar Alptekin for her helps, patience and advices.

I also very appreciate for helps to Özlem Durmaz İncel. I appreciate to Galatasaray University members for their enduring to my long master years. I appreciate to my family and to my friends for their mental supports. This research has been financially supported by the Galatasaray University Research Fund, project number: 19.401.005.

May 2019

Nurhak Karakaya

## TABLE OF CONTENTS

**LIST OF SYMBOLS**

| | |
|---|---|
| **HMOG** | **:** Hand Movement Orientation and Grasp |
| **DF** | **:** Decision Forest |
| **BD** | **:** Boosted Decision Tree |
| **SVM** | **:** Support Vector Machine |
| **LR** | **:** Logistic Regression |
| **ACC** | **:** Data with only Accelerometer fields |
| **GRY** | **:** Data with only Gyroscope fields |
| **MAG** | **:** Data with only Magnetometer fields |
| **ALL** | **:** All data including three sensors and touch event fields |
| **30COLS** | **:** Data with thirty most predictive fields |
| **PCA** | **:** Principal Component Analysis |
| **UZD** | **:** Unpack Zipped Dataset |
| **SCD** | **:** Select Columns in Dataset |
| **ED** | **:** Edit Metadata |
| **CMD** | **:** Clean Missing Data |
| **ERS** | **:** Execute R Script |
| **TMH** | **:** Tune Model Hyperparameters |
| **SD** | **:** Split Data |
| **FBFS** | **:** Filter Based Feature Selection |
| **ML** | : Machine Learning |

## LIST OF FIGURES

## LIST OF TABLES

## TERMINOLOGY

**Time Variables** : The time variables that are coming from touch event table, activity table or calculated time variables from sensor system time.

**Numeric Variables** : The variables that are calculated using X,Y,Z and M columns of senssor tables.

**Binary Variables** : The variables that are calculted from categorical variables.

**USER_TABLE** : The table that is calculated by combaning three sensor tables, touch event tables and activivty tables. This table contains the new calculated numerical variables and time variables. This table contains 100 users with their own records.

**User** : An attender and all calculated fields that belongs to the same attender. A user comes from USER_TABLE. And in USER_TABLE there are 100 users.

**MODEL_TABLE** :The table that contains records which has target attribute. Target attribute contains USER labeled records whose values come from same attender and NO_USER labeled records whose valuse come from other athenders.

**Model User** : It is a combination of user and non user records. It gets all data from a user and 500.000 records from other users.

**USER / NO USER** : In MODEL_TABLE we created an attribute named TARGET. If the records come from the user that we want to authenticate then we set TAGET value as USER. If the records come from other users then we labeled those records as NO_USER in TARGET attribute.

**Experiment** : Experiment is also used for Microsoft Azure code part.

**ABSTRACT**

In this paper, we use Hand Movement Orientation and Grasp (HMOG) sensor data to authenticate smart phone users. The way a user holds, grasps a mobile phone or touches to it are all key factors for authentication. At the moment of a user makes an event on his/her smart phone, three sensors automatically collect data about magnitude, angular speed and acceleration. Moreover, touching and holding events also produce data about pressure and coordinates. In this paper, we build four types of machine learning algorithms (Decision Forest, Boosted Decision Tree, Support Vector Machine, and Logistic Regression) to predict user authentication. The data used in this experiment (HMOG) are collected from 100 attenders. We compare the results of the algorithms and for our scenario, we show that boosted decision tree algorithm with de normalized data gives the results with highest accuracy.

**ÖZET**

Bu makalede, El Hareketi Yönlendirme ve Kavrama (EHYK) algılayıcı verilerini kullanarak akıllı telefon kullanıcılarının kimliklerini doğrulamaya çalışmaktayız. Bir kullanıcının akıllı telefonunu tutma şekli, kaldırma hızı / döndürme hızı, ya da telefonunu kavraması veya ona dokunması, kimlik doğrulama için anahtar faktörlerdir. Cep telefonumuzu elimize alıp kullanmaya başladığımızda; üç algılayıcı otomatik olarak büyüklük, açısal hız ve ivme hakkında bilgi toplar. Ayrıca, telefona dokunmamız, harflere basmamız ya da ekranda elimizi oynatmamız da veri üretir. Bu makalede, telefonda yer alan algılayıcıların okuduğu bilgilerden faydalanıp çeşitli makina öğrenme algoritmaları kullanarak kimlik tanımaya çalıştık. Dört tür makine öğrenme algoritması kullandık. Bunlar: Karar Ormanı, Artırılmış Karar Ağacı, Destek Vektör Makinesi ve Lojistik Regresyon gibi algoritmalardır. Bu deneyde kullanılan veriler (EHYK) 100 mobil cihaz kullanıcısından toplanan algılayıcı verilerdir. Yaptığımız çalışmalar sonrasında, Artırılmış Karar Ağacı'nın normalize edilmemiş veri ile en yüksek kesinlik değeri verdiğini gördük.

# 1. INTRODUCTION

Statistics from 2016 [1] show that from 2 to 2.5 billion people use smartphones. Same research also shows that smartphones are not used just for calling and texting but also for looking for a job, finding a date, reading a book or making an online shopping. Online banking, mailing, playing games can also be added to this list. Same Research Center survey found that 28% of U.S. smartphone owners say they do not use a screen lock or other features to secure their phone. 14% say they never update their phone's operating system, while 10% say they do not update the apps on their phone. With combining those two analyses, securing mobile devices is a main security challenge, because it depends on human attitude or preferences to take necessary security precautions. Regarding to this behavior, researches which focus on passive security are gaining importance to answer questions about how to solve those security challenges. Hence, the main research questions that we focus on are as follows:

- Is it possible to implement a continuous authentication procedure into mobile devices to distinguish whether the original owner is using or not by analyzing behavioral biometric data?, and
- Which machine learning algorithm(s) and feature set(s) will be most accurate to distinguish true owner? Can we also use artificial neural networks to avoid manual feature extraction, or will it be expensive in manner of resource consumption?

In current mobile phone structure, there are different kinds of user authentication methods to prevent unauthorized accesses. Some of them are authentication with fingerprint, text passwords and crossing shapes. When a user use such type of authentication methods, he or she should remember the crossing shapes or the paswords to access his/her mobile phone. And also, the user should cange the passwords in some time interval so that he or she can make his/her mobile phone more secure. In some cases, these authentication methods can still be insufficient to prevent unauthorized entry. Besides, some users can keep their mobile phone in available mode for long time. Therefore, in near future, we believe that several additional techniques need to be proposed to prevent unauthorized accesses.

In this paper, we aim to differentiate mobile phone user by using Hand Movement, Orientation, and Grasp (HMOG) data, which are collected during experiments [2],[26]. In these experiments [2], some kind of event and sensor data from attenders of the experiment is recorded. The recorded data in experiment [2] contains: User information, event information, and information from three sensors: Accelerometer (measures acceleration minus Gx), Gyroscope (measures angular speed) and Magnetometer (measures ambient magnetic field). For each sensor, X, Y and Z coordinate values and time of these values are stored. For this paper, we also created a magnitude metric, which is the square root of sum of squares of X, Y and Z ($\sqrt{X^2 + Y^2 + Z^2}$).

We proposed algorithms to identify and continuously authenticate a smart phone user by analysing his/her previous data, in order to prevent unauthorized entry to his/her mobile phone. We used Microsoft Azure Machine Learning platform for building our models. Decision Forest (DF), Boosted Decision Tree (BD), Support Vector Machine (SVM) and Logistic Regression (LR) are selected as algorithms. In each of our experiments, we used two-class models, which is different from [2], where one-class models are used

After presenting related works in Section 2, in Section 3, we define project methodology together with the project motivation. In section 4, we show the data collection steps for HMOG. In Section 5, we represent data cleaning and data preparation process. Section 6 introduces the proposed model and Section 7 includes the results and related discussion.

## 2. RELATED WORK

Authentication is the process of validating the true user of a system. There are three main approaches to provide authentication. First and the most commonly used one on mobile devices is knowledge-based authentication. This technique is based on using a unique and private information which is expected to be known only by the user. This type of authentication mechanism could be a password, an id number or a secret security question. The second one is object-based authentication. The object-based authentication is based on possession of a distinguishing physical object. A security token, an id card or another trusted object can be used. The third one is biometrics. Biometrics are based on an individual's characterized physical or behavioral attributes. Common examples are fingerprints, keystroke dynamic models of the owner of the device.

There are two survey papers [3, 4] that investigate the use of biometrics for continuous authentication on smart phones. In [3], it was emphasized that sensors such as camera, microphone, etc. can be used to collect physical data, while components such as accelerometers, gyroscopes, touch screens can be used to collect behavioural biometric data such as walking, screen touch gestures, and hand gestures. In the other review paper [4], the studies in the literature were examined in terms of the type and size of data collected, classifiers used in identification, and results obtained. We should note that these papers also investigate the use of physical biometrics for authentication, but here, we specifically focus on studies using behavioural biometrics.

Touch screens are used as input medium on a great majority of smartphones. A touch screen is an electronic visual display for inputs and outputs. By applying classification algorithms to the data collected from touch- screen interactions of users such as micro movements, pressure, finger movements, etc., it is possible to recognize authorized users. There exists various research that focus on touch screen that is based authentication in the literature. In these researches, password patterns [4], tapping behavior [5], touch gestures [6][7][8] etc. are examined for the purpose of creating a model to decide whether user is authorized or not. In [8], touch screen data of 58 attenders were used and for authentication, they also used two class models like Random Forest, k-NN and Support Vector Machine. Data collection for [8] is also similar to [2], in [8] they created an application and attenders did some tasks by scrolling and touching screen. In [9], they used sensor data from some attenders. The number of attenders in [9] was 85 and they collected data during sitting, walking or standing. In [9], they claim that every user has a different locking type and different dragging type of the phone to ears. In [10], they used kernel based algorithms for sensor data. In [11], they used Accelerometer data and Wi-Fi networks for user authentication. In [12] they used just sensor data, and also they used support vector machine, random forest and k-NN as classification algorithms and used feature reduction.

## 3. PROPOSED METHODOLOGY

We divide the overall experiment into three parts: Data collection and data discovery part, modeling part and comparing the results part. The first part is the data discovery part. In that part, we apply data preparation and analysis steps. For data discovery, we worked on Sybase IQ. For modeling and presentation part we worked on Microsoft Azure ML studio.

The authors in [2] include user data to train their models. They use just the data of user that they want to authenticate, whereas we used both user and other users' data to train our models. We used two class machine learning algorisms for our experiment. We used LR, SVM, BD and DF as our machine learning algorithms for our models.

For each model, we supplied five different data flows: Data with only Accelerometer fields (ACC), data with only Gyroscope fields (GRY), data with only Magnetometer fields (MAG), all fields including three sensors and touch event (ALL), and thirty most predictive fields (30COLS). For all data flows, we used both normalized and de-normalized data. Principal Component Analysis (PCA) is used for reducing dimensionality. We used ACC, GRY and MAG models in our experiments to see if any of the three sensors will be enough to catch a user. We also used 30 attributes to create a smaller model and we want to see if these 30 attributes will be enough to catch a user.

For both normalized and de-normalized data, we use four ML algorithms: BD, DF, SVM and LR for prediction. We created separate experiments for each algorithm. Moreover, for BD, DF and LR, we used both normalized and de normalized data. For SVM, we only used normalized data. Therefore, totally we had 7 experiments (2 for BD, 2 for DF, 2 for LR and 1 for SVM) for our models. We used %70 of data as training and %30 for testing. The performances of the models are compared in terms of accuracy.

In each experiment, we had five data flows (Figure 3.1): ACC, GYR, MAG, ALL and 30COLS. For 30COLS, we used Pearson correlation to find the most correlated attributes with target. For ACC, GYR, MAG and ALL, we also used PCA. However, for 30COLS we did not use PCA. Therefore, we had in total 9 different data supplies (ACC, ACC with PCA, GYR, GYR with PCA, MAG, MAG with PCA, ALL, ALL with PCA and 30COLS). For PCA, we chose dimension count as: Attribute count / 3. For example, ALL has 168 variables, so in ALL with PCA we have 55 dimensions. We also used feature selection and correlation tests to eliminate some attributes from our models.

For BD, SVM and LR, we made parameter tuning. We used 10-fold cross validation for tuning. Then, we used these parameter values for our models. Normally, we decided to tune parameters for all users then use those parameters for models. When we tuned parameters for first six users, we saw that the parameter values for those six users are same. So, we stopped tuning and used those six users' parameters for all users. For DF we did not tune parameters. The overall methodology is illustrated in Figure 3.1.

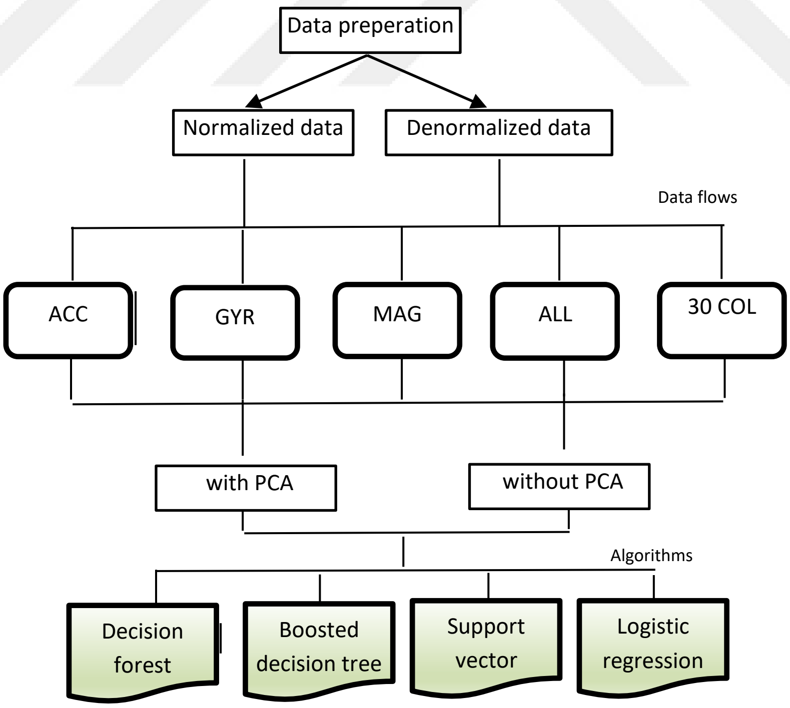


Figure 3.1: Project information flow

# 4. DATA COLLECTION

The Hand Movement, Orientation, and Grasp (HMOG) data, which are collected during experiments [2], is available at [26], and it is about 6 GB of zipped files. In HMOG data collection experiment, there are 100 attenders all of them have same kind of mobile phones (Samsung Galaxy S4). All of the attenders do some tasks in 24 sessions. Each of these sessions are done in 5 to 15 minutes.

In one session there are three tasks that an attender should do by using his/her mobile phone. The tasks are: reading documents, writing text, and navigation on Map. When doing these three tasks, the attenders do specific actions; such that they type massage, they scroll screen, they touch to screen, they press keys, etc. The attenders do their tasks on real time touches. When doing these tasks, the sensors and touch events data are recorded simultaneously with 100Hz reading speed.

The experiments are done either by walking or by siting. After each experiment, 11 data files were created. These files keep activity and user information, event information, and information from three sensors. These files are: Accelerometer.csv, Activity.csv, Gyroscope.csv, KeyPressEvent.csv, Magnetometer.csv, OneFingerTouchEvent.csv, PinchEvent.csv, ScrollEvent.csv, StrokeEvent.csv, and TouchEvent.csv. Moreover in every sessions there are three files for questions. We loaded data files but did not work on question files.

The sensor files are: Accelerometer.csv, Gyroscope.csv and Magnetometer.csv. The sensors detect any changes made in smart phone. The changes can be acceleration, orientation or magnetic field. Accelerometer measures acceleration and motions like shaking and rotating in smartphones. It detects acceleration in X, Y, Z coordinates; in other words, it detects the direction and position of the acceleration, without measuring gravitational acceleration. On the other hand, gyroscope is a sensor that measures the orientation by using Earth gravity. It helps to determine which way a phone is oriented. Magnetometer measures the magnetic field, whom changes can be critical for smart phone users. Moreover, there are events that occur when you do something on smart phones: You touch your smartphone, you scroll downward or upward on your smart phone, you pinch your smart phone, you press a key, etc.

As the next step, the data files are loaded to Sybase database. We prefer Sybase database because of the easiness of data manipulation in it. It enables creating temporary tables for data manipulations and gives good performance for aggregation functions. There are two points to consider when downloading files: firstly, some files have carriage return at the end of line, and secondly some files have new line at the end of line. We solved both problems in order not to miss any information. First of all, we downloaded all files to its own table. In total we have 10 tables: One for Activity, three for sensors and six for event tables. There are 100 attenders * each attenders has 24 sessions * in every session there are 10 files. So in total we loaded: 100 * 24 * 10 = 24000 files.

## 5. DATA DISCOVERY

HMOG data [1] includes files for 100 different users and a pdf file that explains the structure of files. The files are in separate zip files, which needs to be extracted. The corrupted files constitute a minor part of the whole data.

We created 10 tables to load data files to relational database. All of the created tables for our experiment are in Appendix A. There are two kinds of tables: tables with normal name and tables whose name ends wits "_STR". There are corruptions in some files. So we first loaded these files as a complete string to a _STR table than manually converted data of those tables to our original format. So in total we have 20 table, 10 for original data and 10 for error fixes.

The data in files are formatted as coma separated. The end of line for files are not uniquely defined; some of them ends with carriage return, whereas the others with new line. We check for carriage return and new line. For one user, there are about 24 sessions. For some users the number of sessions are less than 24. In every sessions there are 10 files (three sensor files, one activity file and six event files). We created two loops and dynamic SQL to load files into Sybase database. The first loop reads the users, and the second loop reads the sessions. Appendix B shows the load codes for all tables.

After loading all these files into the database tables, we checked data quality and data compliance issues. For data quality issues, we checked all tables one by one. For each column in a table; we checked: null count, not null count, ratios of null count, ratios of not null count, distinct count, max count and min count.

Additionally,

- We controlled most frequently encountered values of a column.
- We checked the values in files with the values in data description files. In some files there are values which are not in data description files.
- We controlled the uniqueness of any column with respect to time variables.
- We grouped column values with descending order to see where the data is cumulated.

After data quality check our first problem is the uniqueness of data. We deleted duplicate rows from our original tables. Moreover, in some tables the sequence of columns are different from the structure of data description file. So we changed the column sequence and loaded files correctly.

We did not delete null values. We find the position of null values in raw files and manually updated null values. Additionally, in data, we discovered that there are several values, which are beyond to the values in data description file. These are usually categorical values. We did not delete these rows instead we took them into account.

Some numerical data were also null because the data file contains “E”. For example, -some records contains values like that 3.0543262E-4. When we load these records to our tables, the records get null. For numerical columns we created codes to convert these types of nulls to normal data format. At Figure 5.1, we show a code part in which we convert a column into normal format.

```
select
    SYSTIME,
    EVENTTIME,
    ACTIVITY_ID,
    case
        when X LIKE '%E%' then
            case
                when substr(X,-3,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(X,RIGHT(X,3),''))
                when substr(X,-4,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(X,RIGHT(X,4),''))
            end
            *
             (CASE
                WHEN RIGHT(X,1) = '4' THEN 0.0001 WHEN RIGHT(X,1) = '5' THEN 0.00001
                WHEN RIGHT(X,1) = '9' THEN 0.000000001 WHEN RIGHT(X,1) = '0' THEN 0.0000000001
            END)
        else convert(numeric(15,10),X)
    END,
    CASE
        when Y LIKE '%E%' then
            case
                when substr(Y,-3,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(Y,RIGHT(Y,3),''))
                when substr(Y,-4,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(Y,RIGHT(Y,4),''))
            end
            *
             (CASE
                WHEN RIGHT(Y,1) = '4' THEN 0.0001 WHEN RIGHT(Y,1) = '5' THEN 0.00001
                WHEN RIGHT(Y,1) = '9' THEN 0.000000001 WHEN RIGHT(Y,1) = '0' THEN 0.0000000001
                WHEN RIGHT(Y,1) = '8' THEN 0.00000001
            END)
        else convert(numeric(15,10),Y)
    END,
    CASE
        WHEN Z LIKE '%E%' THEN
            case
                when substr(Z,-3,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(Z,RIGHT(Z,3),''))
                when substr(Z,-4,1) = 'E' then CONVERT(NUMERIC(15,10),REPLACE(Z,RIGHT(Z,4),''))
            end
            *
        (CASE
            WHEN RIGHT(Z,1) = '4' THEN 0.0001 WHEN RIGHT(Z,1) = '5' THEN 0.00001
            WHEN RIGHT(Z,1) = '9' THEN 0.000000001 WHEN RIGHT(Z,1) = '0' THEN 0.0000000001
        END)
        else convert(numeric(15,10),Z)
    END,
    PHONE_ORIENTATION
from
    TABLE_GYROSCROPE_V2_STR;
```

Figure 5.1: A part of the code for converting data in Sybase table

After data quality checks, we examined to the data compliance check.

- We controlled if the activity table is compliant to sensor tables with respect to time and activity numbers.
- We controlled if the sensor tables are also compliant to each other by time and activity id.
- We controlled if the activity table is compliant to event tables.

We represent an example of compliance test in Figure 5.2.

```
select
    sum(case when ACC.ACTIVITY_ID is not null and AC.ID is not null then 1 end) BOTH_EXIST,
    sum(case when ACC.ACTIVITY_ID is null and AC.ID is not null  then 1 end) ACTIVITY_EXIST,
    sum(case when ACC.ACTIVITY_ID is NOT null and AC.ID is  null  then 1 end) ACTIVITY_NOT_EXIST,
    100.0 * BOTH_EXIST / (BOTH_EXIST + ACTIVITY_EXIST + isnull(ACTIVITY_NOT_EXIST,0)) RATIO
from
    TABLE_ACTIVITY AC full JOIN TABLE_ACELERAMETOR ACC ON ACC.ACTIVITY_ID = AC.ID
```

Figure 5.2: Compliance tests

The results of compliance test is like that:

- There are some activates in activity table in which there are no events for that activity. It is possible, because in some activates an attender does not need to scroll, or does not need to press a key.
- There are no event in which the activity number is not in activity table. It is normal and as we expected. The activity table should cover all event tables.
- There are sensor records whose ids are not in activity table. For our expectation activity table should cover sensor tables. So we did not expect such kind of problems. We called this problem as ACTIVITY_ID_PROBLEM

The reason of ACTIVITY_ID_PROBLEM is that the activity id consist of SubjectID + Session_nember + ContentID + Runtime determined Counter value, at Table 4.1 we show all details. When we controlled the non-matching IDS, we see that they match for first 9 digits: SubjectID + Session_nember + ContentID, but the “Runtime determined Counter value” differs for some records. To solve this problem, we created another ACTIVITY_ID and called it as ACTIVITY_ID_FIRST_9 which get the first nine digit of original activity ID. The first nine digit compose of “SubjectID + Session_nember + ContentID”. But in that case when we join tables with respect to ACTIVITY_ID_FIRST_9, some duplicate records and wrong matches occurred. So we decided to eliminate such type of records from our list.

After analyzing our data, we saw that there are more touches than any other events. On average in a session, there are about 1741 one-finger touch events, 800 pinch events, 1741 scroll events, 45 stroke and 4705 touch events. Moreover there are about 50800 sensor records for one session. Instead of creating a highly complex data with lots of null values in it, we only used touch table for our models. For our experiments, we use three sensor tables (Accelerometer, Gyroscope and Magnetometer), user and activity identification table (Activity) and touch event table (TouchEvent).

Activity table has 9 attributes, given in Table 5.1 in details. We give accelerometer table in Table 4.2. Gyroscope and Magnetometer tables have the same structure as Table 5.2. We computed M as magnitude, which is equal to $\sqrt{X^2 + Y^2 + Z^2}$ , and M is not in the data file. Finally, we worked with touch event table, which has 11 attributes (Table 5.3).

The time variables in tables are absolute time stamp, or relative time stamp. We use in our project absolute time variables. A Timestamp is the number of milliseconds elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Table 5.1. Activity table attributes

| | | |
|---|---|---|
| ID | numeric(20,0) | Composed as: SubjectID + Session_nember + ContentID + Runtime determined Counter value |
| SUBJECT_ID | numeric(6,0) | 6 digits: ID of current subject |
| SESSION_NUMBER | numeric(2,0) | 1-24: Session number for current subject |
| START_TIME | numeric(20,0) | Start time of current activity, in absolute timestamps |
| END_TIME | numeric(20,0) | End time of current activity, in absolute timestamps |
| RELATIVE_START_TIME | numeric(20,0) | Start time of current activity, relative to system boot |
| RELATIVE_END_TIME | numeric(20,0) | End time of current activity, relative to system boot |
| GESTURE_SCENARIO | numeric(2,0) | 1: Sit 2: Walk |
| TASK_ID | numeric(2,0) | 1,7,13,19: Reading + Sitting<br>2,8,14,20: Reading + Walking<br>3,9,15,21: Writing + Sitting<br>6, 12, 18, 24: Map + Walking |
| CONTENT_ID | numeric(2,0) | 1: First sub-task<br>2: Second sub-task<br>3: Third sub-task |

Table 5.2: Accelerometer table attributes

| | | |
|---|---|---|
| SYSTIME | numeric(20,0) | Absolute time-stamp |
| EVENTTIME | numeric(20,0) | Sensor event relative time-stamp |
| ACTIVITY_ID | numeric(20,0) | Belonged activity |
| X | numeric(15,0) | Acceleration minus Gx on the x-axis |
| Y | numeric(15,0) | Acceleration minus Gx on the y-axis |
| Z | numeric(15,0) | Acceleration minus Gx on the z-axis |
| M | numeric(15,0) | Square root of sum of squares X,Y and Z |
| PHONE_ORIENTATION | numeric(2,0) | 0: Portrait and no rotate<br>1: Device rotated 90 degrees counter-clockwise<br>3: Device rotated 90 degrees clockwise |

Table 5.3: Touch event table attributes

| | | |
|---|---|---|
| SYSTIME | numeric(20,0) | Absolute timestamp |
| EVENTTIME | numeric(20,0) | Sensor event relative timestamp |
| ACTIVITY_ID | numeric(20,0) | Belonged activity |
| POINTER_COUNT | numeric(2,0) | 1: Single touch<br>2: Multi touch |
| POINTER_ID | numeric(2,0) | 0: Single touch, or first pointer in multi touch<br>1: Second pointer in multi touch |
| ACTION_ID | numeric(2,0) | 0 or 5: DOWN<br>1 or 6: UP<br>2: MOVE |
| X | numeric(15,0) | Touch location in X coordination |
| Y | numeric(15,0) | Touch location in Y coordination |
| PRESSURE | numeric(15,0) | Touch pressure |
| CONTACT_SIZE | numeric(15,0) | Touch contact size |
| PHONE_ORIENTATION | numeric(2,0) | 0: Portrait and no rotate<br>1: Device rotated 90 degrees counter clockwise<br>3: Device rotated 90 degrees clockwise |

The final table is built using these five tables. First, we merged activity table with touch event table. We get SUBJECT_ID, GESTURE_SCENARIO, TASK_ID and CONTENT_ID from Activity table and all columns from Touch table. In total, we obtained 14 attributes. We stored the data of new table into a temporary table called #acc_evt.

Then, we merged #acc_evt table with sensor tables on ACTIVITY_ID and SYSTIME. We first merged #acc_evt table with Accelerometer. We first found the maximum sensor system time which is the biggest sensor system time that has value of event system time – 100 ms. Then, we found minimum sensor system time which is the smallest sensor system time that has value of event system time + 100 ms.

Now for our new table, we have three time points: Touch event time, biggest sensor reading before Touch event time – 100 ms, smallest sensor reading after Touch event time + 100 ms. For example, for Accelerometer sensor; we called these three time points as SYSTIME, SEN_SYSTIME_ACC_BEFORE, SEN_SYSTIME_ACC_AFTER in Figure 5.3.

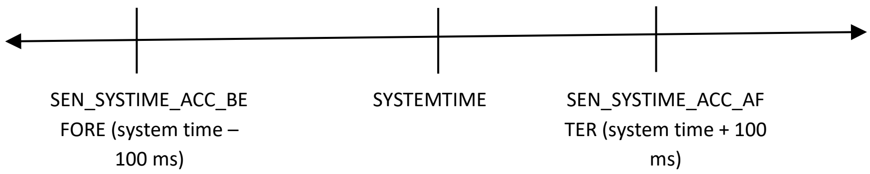


Figure 5.3: Time points

After we calculated all time variables for three sensors, we calculated new variables by using those time variables and X, Y, Z and M attributes from sensor tables. We will explain how we calculated those variables by using Accelerometer table and X column for Accelerometer. (The calculation of other variables and other sensors will follow the same way). We first calculate minimum value of X, maximum value of X, average value of X and standard deviation of X between SEN_SYSTIME_ACC_BEFORE and SYSTEMTIME. Then, we calculated; minimum value of X, maximum value of X, average value of X and standard deviation of X between SYSTEMTIME and SEN_SYSTIME_ACC_AFTER. We call all these variables as numeric variables. Then, we calculated the values of difference between before system time and after system time. For example, X_ACC_MIN_DIFF will be the difference between minimum values of X after system time and minimum values of X before system time.

X_ACC_MIN_DIFF = X_ACC_MIN_AFTER - X_ACC_MIN_BEFORE

The new variables for X column of Accelerometer table are in figure 5.4.

```
X_ACC_MEAN_BEFORE numeric(15,10),
X_ACC_MAX_BEFORE numeric(15,10),
X_ACC_MIN_BEFORE numeric(15,10),
X_ACC_STDV_BEFORE numeric(15,10),
X_ACC_MEAN_AFTER numeric(15,10),
X_ACC_MAX_AFTER numeric(15,10),
X_ACC_MIN_AFTER numeric(15,10),
X_ACC_STDV_AFTER numeric(15,10),
X_ACC_MEAN_DIFF numeric(15,10),
X_ACC_MAX_DIFF numeric(15,10),
X_ACC_MIN_DIFF numeric(15,10),
X_ACC_STDV_DIFF numeric(15,10),
```

Figure 5.4: New calculated variables

In total, we calculated 144 numerical variables: we have 4 directions (X, Y, Z, M) * 12 variables (variables in Figure 4.5) * 3 sensors = 144 new variables. For these new 144 variables, if the value of variable is null, we set it to zero.

After we calculated numerical variables, we computed binary variables. In our data set there are 7 categorical variables: GESTURE_SCENARIO, TASK_ID, POINTER_COUNT, POINTER_ID, ACTION_ID, CONTENT_ID and PHONE_ORIENTATION. If a categorical variable has n distinct values, we created n-1 distinct dummy binary variables from that categorical variable.

POINTER_COUNT has three values: 1, 2 and 3. 1 for single touch, 2 for multiple touch and 3. Normally from data definition [2], PONINTER_COUNT should have 1 and 2 as value, but in some cases it has value of 3. So, we have to consider about this new value. Hence, we created 2 new binary variables: POINTER_COUNT_S (set its value = 1, if single touch, 0 otherwise), POINTER_COUNT_M (set its value = 1 if multiple touch 0 otherwise).

TASK_ID has 24 different values. Instead of creating 24-1 different dummy variables for TASK_ID, we created 5 dummy variables, because TASK_ID can be grouped into 6 groups as in Table 5.4.

Table 5.4: TASK_ID groups

| | |
|---|---|
| 1, 7, 13, 19 | Reading + Sitting |
| 2, 8, 14, 20 | Reading + Walking |
| 3, 9, 15, 21 | Writing + Sitting |
| 4, 10, 16, 22 | Writing + Walking |
| 5, 11, 17, 23 | Map + Sitting |
| 6, 12, 18, 24 | Map + Walking |

GESTURE_SCENARIO has two values. 1 for sit and 2 for walk. We created GESTURE_SCENARIO_SIT_F as our dummy variable. And set its value to 1 if GESTURE_SCENARIO is 1, 0 otherwise.

POINTER_ID has three values. 0 for single touch and 1 for multi-touch, 2 is undefined. Normally from data definition [2], POINTER_ID should have 0 and 1 as value, but in some cases it has value of 2. So, we have to consider about this new value. Hence, we created 2 new binary variables: POINTER_ID_ST (set its value = 1, if single touch, 0 otherwise), POINTER_ID_MT (set its value = 1 if multiple touch 0 otherwise).

ACTION_ID has 5 different values. But it can be grouped into 3 groups.

- 0 or 5: DOWN
- 1 or 6: UP
- 2: MOVE

So, we created two dummy variables: ACTION_ID_DOWN (set its value = 1, if ACTION_ID = 0 or 5, 0 otherwise), ACTION_ID_UP (set its value = 1 if ACTION_ID = 1 or 6, 0 otherwise).

PHONE_ORIENTATION has three different values.

- 0: Portrait and no rotate
- 1: device rotated 90 degrees counter-clockwise
- 3: device rotated 90 degrees clockwise

We created two dummy variables for PHONE_ORIENTATION. PHONE_ORIENTATION1 (set its value = 1 if PHONE_ORIENTATION = 1 , 0 otherwise) , PHONE_ORIENTATION0 (set its value = 1 if PHONE_ORIENTATION = 0 , 0 otherwise).

CONTENT_ID has 6 different values. For that reason we created 5 dummy variables. These variables are: CONTENT_ID1, CONTENT_ID2, CONTENT_ID3, CONTENT_ID4, and CONTENT_ID5.

After creating dummy variables, we deleted original categorical variables from our list. Because categorical variables will be correlated to dummy variables. We show some part of code for creating dummy variable in Figure 5.5.

```
BEGIN
alter table TABLE_FINAL add TASK_ID_RS numeric(1);
alter table TABLE_FINAL add TASK_ID_RW numeric(1);
alter table TABLE_FINAL add TASK_ID_WS numeric(1);
alter table TABLE_FINAL add TASK_ID_WW numeric(1);
alter table TABLE_FINAL add TASK_ID_MS numeric(1);

update TABLE_FINAL set
    TASK_ID_RS = 0,
    TASK_ID_RW = 0,
    TASK_ID_WS = 0,
    TASK_ID_WW = 0,
    TASK_ID_MS = 0;

commit;

UPDATE TABLE_FINAL set TASK_ID_RS = 1 where  TASK_ID in (1, 7, 13, 19);
commit;
UPDATE TABLE_FINAL set TASK_ID_RW = 1 where  TASK_ID in (2, 8, 14, 20);
commit;
UPDATE TABLE_FINAL set TASK_ID_WS = 1 where  TASK_ID in (3, 9, 15, 21);
commit;
UPDATE TABLE_FINAL set TASK_ID_WW = 1 where  TASK_ID in (4, 10, 16, 22);
commit;
UPDATE TABLE_FINAL set TASK_ID_MS = 1 where  TASK_ID in (5, 11, 17, 23);
commit;

alter table TABLE_FINAL drop TASK_ID;
commit;
END;
```

Figure 5.5: A part of the code for creating dummy variable

After deleting categorical variables, we have in total 177 variables. These are: 10 identification and time variables, 144 calculated numerical variables and 20 dummy variables and 3 floating variables from table TOUCH. We called the final table as USER_TABLE.

In table USER_TABLE, there are 100 users. We randomly chose 20 users from those 100 users. The SubjectIDs for the randomly selected users are: 745224, 352716, 219303, 501973, 264325, 527796, 862649, 663153, 556357, 841866, 923862, 815316, 733162, 472761, 897652, 186676, 998757, 872895, 240168, and 151985. We show in Figure 5.6 random user selection code from our list of users. We first create a Sybase temporary table and add a random user to that table. If the new random id is in the table then we select another random number otherwise we add that user to the list. We continue this operation for 20 times.

```
BEGIN
declare @seed int;
declare v_rand_index numeric(10);
declare counter numeric(10);
declare V_SUBJECT_ID numeric(10);
declare v_cnt numeric(3);
set @seed=0;

set counter = 1;
while counter <= 20 loop
    set v_rand_index = ROUND(((99 - 1) * RAND() + 1), 0);

    select SUBJECT_ID into V_SUBJECT_ID from #all_ind2 where IDX =  v_rand_index;

    select count(*) into v_cnt from #rnd1 where IND = V_SUBJECT_ID;
    while v_cnt > 0 loop
        set v_rand_index = ROUND(((99 - 1) * RAND() + 1), 0);
        select SUBJECT_ID into V_SUBJECT_ID from #all_ind2 where IDX =  v_rand_index;

        select count(*) into v_cnt from #rnd1 where IND = V_SUBJECT_ID;

    end loop;

    insert into #rnd1 (IND) values(V_SUBJECT_ID);

    set counter = counter + 1;
end loop;

commit;
END;
```

Figure 5.6: Random user selection

From randomly chosen 20 users in table USER_TABLE, we create a new table called MODEL_TABLE. The model table has all columns of USER_TABLE and a new column called TARGET. We add all records of those 20 users one by one to MODEL_TABLE. The data process is like this:

- Take one of 20 chosen users from USER_TABLE and add all records of those user to MODEL_TABLE, and set TARGET value as: ”USER”. For one user in USER_TABLE there are approximately 100.000 records.
- Take randomly 500.000 records from USER_TABLE whose ids is different from the id in step one and add them to table MODEL_TABLE, and set TARGET value as: ”NO_USER”.
- Continue with next user from 20 users in table USER_TABLE.

In total, for our experiment, we had 20 model user, where each case involves about 100.000 USER and 500.000 NO_USER records. In total for one model user there are about 600.000 records. After creating 20 model users, we downloaded those model user into text files one by one. One text file takes about 1.2 GB of space at disk. So we have 1.2 * 20 = 24 GB of files. In that case it is very hard to load those files to Microsoft Azure. For that reason we take zip version of the files. In that case, one file takes about 400 MB of disk space.

# 6. DATA MODELING

In order to work in Microsoft Azure environment, we create a Microsoft account for Azure ML studio. Azure ML gives for an account 10 GB of free space for your data and experiments. One model user takes about 400 MB of disk space in zipped format. When we run our models for one model user in total it takes about 7 GB of space. For that reason we loaded one model user; run all models with respect to that model user, save its results and then finally clear all space. We do all those steps for all chosen model users which means 20 times. Note: Microsoft Azure also calls a working area as experiment, so we use experiment here as “Microsoft Azure experiment”.

Our overall project looks like as in Figure 3.1. For every steps we create an Azure ML experiment. In total we have 12 experiments: one for saving ID, 3 for tuning, one for finding attribute correlation with TARGET, 7 for models.

Our first experiment is to save ID of user to table ID_TABLE. This table is used to add ID to statistical results. We save the user ID to ID_TABLE by using “Enter Data Manually” item. Note: we need such a table because we delete identification variables from our data set. We will explain it later.

We continue like this: at section 6.1 we explained data normalization and Feature Selection steps, at section 6.2 we explained parameter tuning and also we explained the some details of used machine learning algorithms, at section 6.3 we explained the correlation steps. And finally, at section 6.4 we explained model structures.

### 6.1 Data Normalization and Feature Selection

Our second experiment is to create normalized and DE normalized version of our original data. We show this experiment in Figure 6.1.

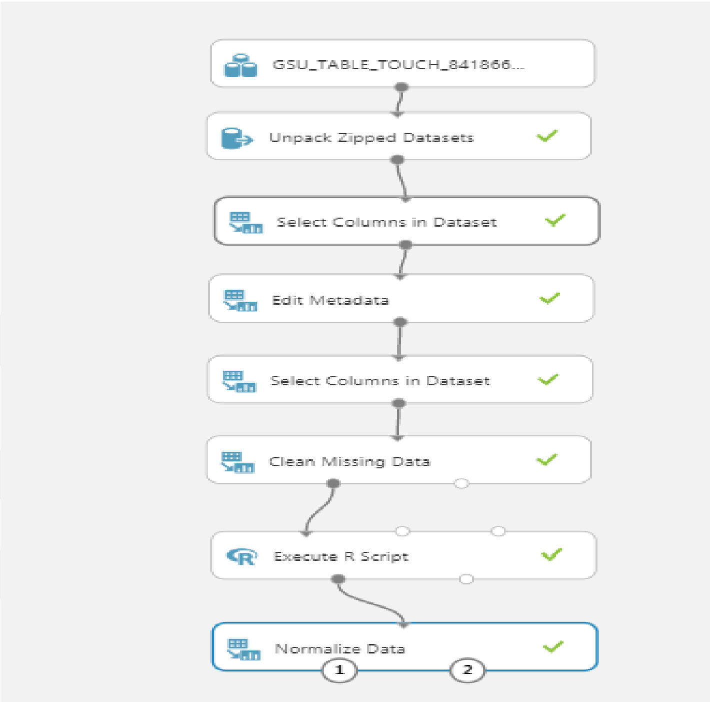


Figure 6.1: Creating normalized and de normalized data

The data for a model user is zipped because of space usage. At first step in Figure 6.1, we unzip it by using “Unpack Zipped Dataset” UZD item. Then we use a “Select Columns in Dataset” SCD item so that we eliminate last empty column. When we load a text file into Azure environment, Azure creates an empty last column so we delete it.

In Azure environment after loading data to Azure, we saw that column names in current table are in “Col1”,”Col2”,”Col3” format. So, we have to give their exact names like “TARGET”, ”SUBJECT_ID”, “SYSTIME”. To convert column names to our original format in Figure 6.1 we use an “Edit Metadata” ED item.

Before data normalization process we made attribute selections. We eliminate some attributes. For each experiment, we first excluded identification variables and time variables from our data set. The reason is that some attenders do all tasks in a specific time interval; for that reason, for those attenders time variables are highly correlated to target variable. So, we excluded all time variables and identification variables from our dataset. We add attribute selection step to our second experiment. The second SCD in Figure 6.1 is for that purpose. It eliminates time and identification variables.

There were some null variables in our list. Normally we converted null values to zero at Sybase but still there were some null values. So in Figure 6.1, we use a “Clean Missing Data” CMD item to convert nulls to zero.

Then in Figure 6.1, we use an “Execute R Script” ERS item because; normally we make dummy variables at Sybase IQ but when we check our list, we see: we do not convert PHONE_ORIENTATION and CONTENT_ID into dummy variables. At that step we convert them into dummy variables and delete the original ones. At section 5, we explained it in detail.

Finally in Figure 6.1, we normalize data by using “LogNormal” normalization. We store ERS result as “denorm” table and normalization result as “norm” table. We will use those tables in our all remaining experiments.

## 6.2 Parameter Tuning and Machine Learning Algorithms

We make three experiments for parameter tuning (for SVM, BD and LR). For all of parameter tuning, we use normalized data and then we use calculated parameters for all modeling experiments. At Figure 6.2, we show the overall structure for LR parameter tuning. The structure is same for SVM and BD also, just the classifier changes: for LR we use Two-Class Logistic Regression [13], for SVM we use Two class Support Vector Machine [18], for BD we use Two-Class Boosted Decision Tree [22]. For DF we did not tune parameters.

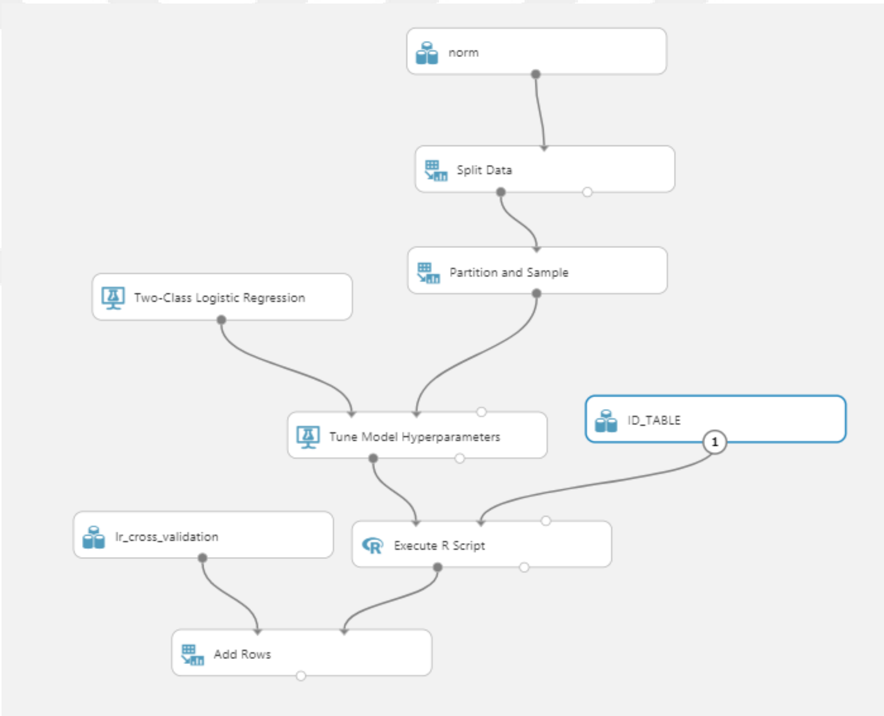


Figure 6.2: LR parameter tuning

In Figure 6.2, we use norm table, then we use "Split Data" SD item to split the data for parameter tuning. We just use %20 of all data for tuning, because the tuning takes long time. So we have to a small portion of overall data. But still, %20 of data contains about 120.000 records for one model user.

In Figure 6.2, we use Partition and Sample item to use 10 fold cross validation. This item is used when multiple parameters is used for a model. It gives 10 distinct datasets. We then use Tune Model Hyperparameters (TMH), it gets folds from Portion and Sample and parameter range from classifier and finds best tuned values.

In Figure 6.2, ID_TABLE stores the user ID for that experiment. The ERS adds ID to THH results. “lr_cross_validation” table stores the results of all LR parameters. Finally; we use an Add Row item to add last user tuning results to “lr_cross_validation” table.

We store results of LR tuning in lr_cross_validation table, SVM results in svm_cross_validation table and BD results in bd_cross_validation table.

In Two Class Logistic Regression item, we use “Create trainer mode” as Parameter Range to inform Azure this will be not a single parameter but a list of parameters. The list of parameter range for Two Class Logistic Regression item are in Figure 6.3.

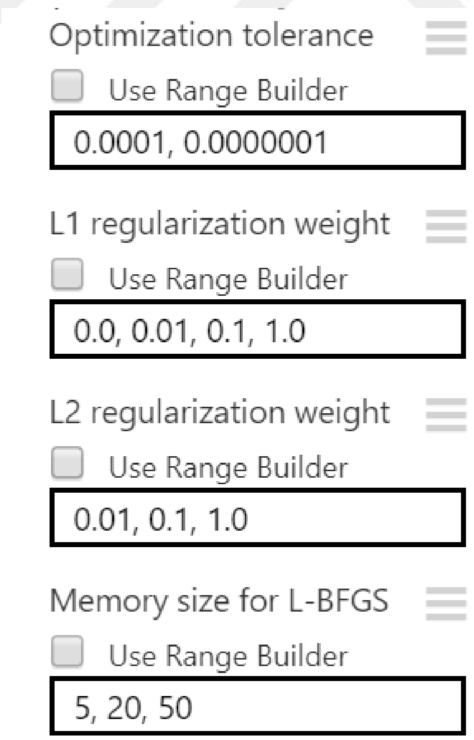


Figure 6.3: Parameter Range for LR

Two class logistic regression in Azure is a logistic regression model and is used to predict a data set which has two outcomes. It is a supervised learning method so you have to have a data set which has two results. In our experiment, we have USER and NO_USER labels. First of all, you have to inform azure which column will be predicted. In our case, it is TARGET column. Then, we set “Create trainer mode” to “Parameter range”. For parameter optimization we will use that option. But after we get best parameter values, we will use “Single Parameter” for “Create trainer mode”.

Since here we used “Parameter Range”, in our experiment after “two class logistic regression” we add a TMH and a Partition and Sample item. If we use “Single Parameter”, we have to add “Train Model” item for training. Note: after parameter tuning for all remaining experiments we add “Train Model” item.

“Optimization tolerance” sets a threshold value for optimizing the model. When the improvement between iterations falls below that value the algorithm thinks it reached an optimal value and training stops.

To create less complex models when you have lots of features in your dataset you can use regularization to prevent over fitting.  In Azure, we have L1 and L2 regularization. A model that uses L1 regularization is called Lasso regression [14], and a model that uses L2 regularization is called Ridge regression [15]. The main difference between Ridge and Lasso regression is the penalty term. They add different terms as penalty to loss functions. Ridge regression adds squared value of coefficient to loss function whereas Lasso adds absolute value of coefficient as penalty term.

Ridge regression adds a penalty term to original loss function such that all coefficients are squared. Here lambda is the penalty parameter.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}{\beta_j}^2 \tag{1}$$

If lambda is zero then Ridge regression becomes original loss function. If lambda is too much then it will add high weights and makes model so simple. In that case under fitting occurs.

Lasso regression adds a penalty term to original loss function such that all coefficients are in absolute value. Here lambda is the penalty parameter.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{2}$$

If lambda is zero then Lasso regression becomes original loss function. If lambda is too much then it will add high weights and makes model so simple. In that case under fitting occurs. In both Lasso and Ridge regressions the value of lambda is an important factor for model prediction.

Lasso shrinks less important attributes' coefficient to zero and removes all those attributes from the model. So those methods are powerful for feature selection if the attribute count is high and data is abundant. Ridge makes coefficient of less important attributes small so they have little effect on model.

Azure supports a linear combination of L1 and L2 regularization. It uses a combination of regularization for linear span. If we set L1 as x and L2 as y then ax + by = c will be a linear span of the regularization term.

"Memory size for L-BFGS" [16], [17] is a popular algorithm and used for parameter estimation. The algorithm starts with an initial value of optimal value of $X_{0,}$ then iteratively tries to find a better estimate of $X_1$, $X_2$, ..This parameter limits the amount of memory that will be used to compute next step and direction.

"Random number seed" is used for multiple runs. If you want to take in every run the same results, you have to set a seed value. For our experiment we set seed as 0. If you don’t set a seed value, next time when you run your model again you can have a slightly different results.

“Allow unknown category” is used to automatically assign values for null categorical attributes. In our experiments we don’t have such type of values.

Normally, we can do just one experiment and tune all models in that experiment. But, when we tried to do this our disk space finished and we could not finish tuning. Then, we decided to separate all tuning experiments.

For BD and SVM we have two more experiments for tuning. We saved SVM results in “svm_cross_validation” table and BD results in “bd_cross_validation” table.

SVM [18],[19],[20] is one of the popular machine learning algorithms. It is one of the earliest algorithms and still very popular. It can be used for both regression and classification. It divides training set with respect to their class labels into hyperplanes that maximizes margins between two classes. In our experiment, we used two class support vector machine. This one uses linear kernel. For other kernel types Azure have another SVM called Two-Class Locally Deep Support Vector Machine [21]. In [2] they used RBF kernel, for our experiment we used linear kernel.

Again as in LR, we user “Parameter Range” for parameter tuning and after we tuned parameters we used “Single Parameter” for modelling. Again as in LR, we used Portion and Sample for 10 fold cross validation and TMH to find best parameter values. The parameter range for SVM is in Figure 6.4.

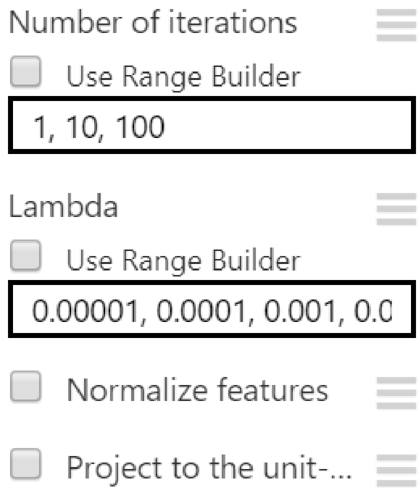


Figure 6.4: Parameter Range for SVM

“Number of iterations” is used to find how long the model will run until to find best hyperplane that divides maximum margin. In that case there will be a trade of between model speed and accuracy. Bigger numbers will result in slower models.

“Lambda” is the regularization parameter for SVM. SVM encounters optimization of two problems: maximizing the margin and minimizing the mis-classification. If lambda is larger, then more mis-classified examples are allowed in training set. If the lambda is small, then less misclassified examples are allowed. If lambda is zero this means no mis-classified examples occurs in training set. But in that case, there will be overfitting in test data. So there is a tradeoff. Smaller lambdas are usually good but too smalls can result in overfitting.

In Figure 6.4, we don’t set “Normalize features” for parameter tuning and models. For SVM we used normalized data. And before using SVM we have already normalized the data. So there is no need to normalize data again.

Two-Class Boosted Decision Tree creates a classifier which has only two results. In our case it is USER and NO_USER. The algorithm in Azure ML studio depends on boosted decision trees [23].

Boosted decision tree is an ensemble algorithm in which a new tree corrects the errors of the previous one. Here second tree corrects the errors of the first tree, and third tree corrects the errors of the second tree and so on. Finally prediction depends on the result of entire trees. The main principle in boosted decision tree: week trees come together to create a strong learner. When an input is misclassified its weight is increased so that the next tree most probably will classify it correctly. The algorithm in our experiment gives the best results, but it is a bit slower than the other algorithms.

The algorithm starts with weak learners at every step it calculates the loss function and increase the weight of misclassified inputs. For next step the new tree tries to recover the loss.

For BD models, as in SVM and LR, when we tune parameters. We used Parameter range for “Create trainer mode” and when we run models Single Parameter for “Create trainer mode”. Again as in LR and SVM, we used Portion and Sample for 10 fold cross validation and TMH to find best parameter values for BD parameter tuning. The parameter range for BD is in Figure 6.5.

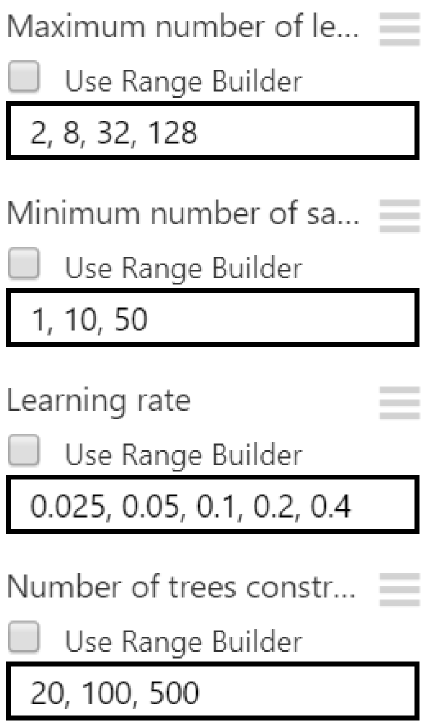


Figure 6.5: Parameter Range for BD

“Maximum number of leaves per tree” indicates the maximum number of leaf node in any tree. When you increase that value, the precision of model can increase but can also cause overfitting. Increasing this also can cause slower models.

“Minimum number of samples per leaf node” indicates the number of cases that is needed to create a node. If you increase this number, to create a new node more cases are required. If the number is just one, then any different case will create a node.

“Learning rate” it is the regularization parameter for BD. It slows down or make faster the training. Low learning rate which means more shrinkage results in more iterations to reach same accuracy. New trees are added to make correct previous trees’ errors. Adding more trees can fit the model quickly but can also cause over fitting.

“Number of trees constructed” indicates the total number of tree that will be created in ensemble. Increasing the number most probably will make better precision models but will also cause longer training times.

“Random Number seed” and “Allow unknown categorical levels” are same as in SVM and LR.

The results of tuned parameter values for BD are given as in Table 6.1, for SVM in Table 6.2 and for LR in Table 6.3. We run our models with these parameters for 20 users separately.

Table 6.1: Tuned parameter values for boosted decision tree

| **Number of leaves** | **Minimum leaf instances** | **Learning rate** | **Number of trees** |
|---|---|---|---|
| 90 | 26 | 0.375257 | 350 |

Table 6.2: Tuned parameter values for support vector machine

| **Number of iterations** | **Lambda** |
|---|---|
| 98 | 0.003046 |

Table 6.3: Tuned parameter values for logistic regression

| Optimization tolerance | L1 regularization weight | L2 regularization weight | Memory size for L-BFGS |
|---|---|---|---|
| 0.000003 | 0.381005 | 0.34971 | 48 |

For Two-Class Decision Forest [25] we did not tune parameters. We used as parameter values as default values. The parameters for DF is ate Figure 6.6.

For DF, we used bagging or bootstrap aggregating [24] as “Resampling method”. When you use a training set $T$ of size n, begging creates m number of $T_i$ new training set by randomly sampling from original training set $T$ . In that case when we create new sets from original training set some records will be unique on the other hand other records will be repeating. This type of sampling is called bootstrap sampling. Then bagging will create m models and fit m bootstrap samples. Then will vote the results and give final prediction. For our case, if we have 5 bagged decision trees and they give: USER, NO_USER, USER, USER, NO_USER. The overall votes will give results as USER.

Since we did not tune parameters for DF, we used “Single Parameter” as “Create trainer mode”. The parameter values are default Azure parameter values for bagging DF.

“Number of decision trees” specifies the number of tree that will be created. If you create more trees, then you can get better precision but models can be slower.

“Maximum depth of the decision trees” indicates that in any tree how many levels /depths can be created. Increased depths may cause overfitting and longer training times but it can also increase the precision.

“Minimum number of samples per leaf node” indicates the number of cases that is needed to create a node. If you increase this number, to create a new node more cases are required. If the number is just one, then any different case will create a node.

“Random Number seed” and “Allow unknown categorical levels” are same as in other machine learning algorithms used in our experiments.

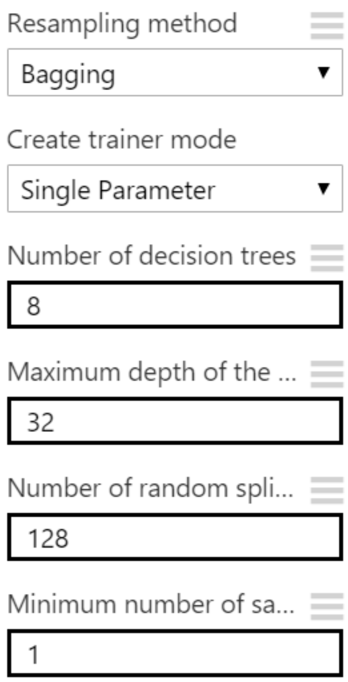


Figure 6.6: Parameters for DF

### 6.3 Finding Most Predictive Attributes

In our experiment, we try to measure a smaller model which uses a small number of attributes to recognize user. For this we create a separate experiment in which we use 30 most predictive attributes for user authentication. The overall structure to find the best 30 attributes for authentication is in Figure 6.7.

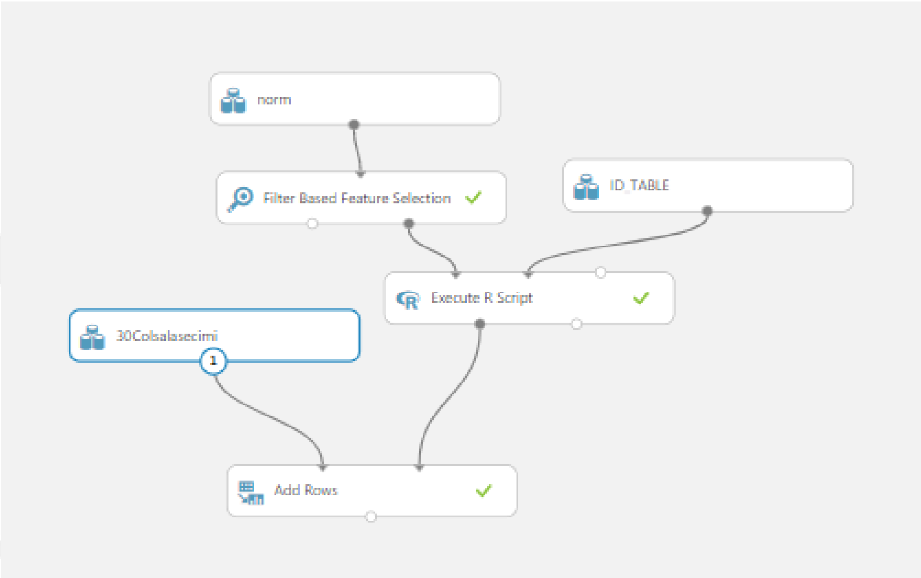


Figure 6.7: 30 Cols Attribute Selection

As in parameter tuning we use here also the normalized data. We used a “Filter Based Feature Selection” (FBFS) item to find most correlated attributes with our target. In our FBFS we use Pearson Correlation and set the number of desired features to 30. We again as in parameter tuning, use ID_TABLE to identify user.

Our ERS item connects FBFS item with ID_TABLE to add ID to FBFS results. 30Cols table stores the attribute correlation values with target. We use an Add Row item to add last user result to 30Cols table.

In 30Cols table we save all correlation results to see most correlated and less correlated attributes. In some users Magnetometer attributes are most predictive, in some users Accelerometer attributes are more predictive, and in some users Gyroscope attributes are most predictive moreover we have users in which phone orientation or touch events are more predictive. It shows that the prediction of attributes completely differs from user to user. So our smaller model which use most predictive 30 attributes completely changes from user to user.

### 6.4 Models

After calculating tuned parameters for models and 30 most predictive attributes, we make 7 different experiments for models. In our models; we use BD, DF, SVM and LR. We have 2 experiments for BD (BD with normalized data and BD with de-normalized data), 2 experiments for LR (LR with normalized data and LR with de-normalized data), 2 experiments for DF (DF with normalized data and DF with de-normalized data) and finally one experiment for SVM, for SVM we just used normalized data.

In every experiment, we create such type of data flows: data flow using Accelerometer fields (ACC), data flow using Accelerometer fields PCA taken (ACC with PCA), data flow using Gyroscope fields (GRY), data flow using Gyroscope fields PCA taken (GRY with PCA) , data flow using Magnetometer fields (MAG), data flow using Magnetometer fields PCA taken (MAG with PCA) , data flow using all fields (ALL), data flow using ALL fields PCA taken (ALL with PCA) , and thirty most predictive fields (30COLS).

So in total we have 7 experiments and in every experiment we have 9 data flows. So in total we have 7*9 = 63 model runs. In Figure 6.8, we show an experiment. This experiment is BD with normalized data. The other experiments are also similar just the classifier and the data changes. If the experiment uses moralized data we load “norm” table otherwise we load “denorm” table.

Here we will just explain BD with normalized data and we will write all about it. In Figure 6.8, first we load norm table. After that we connect three SCD items to norm table to split the data. These first three SCD items are used for sensor attribute separation. First SCD takes Accelerometer fields, second Magnetometer fields and the third SCD takes Gyroscope fields. By using these first three SCD items we are sure to create three data flows for sensors.

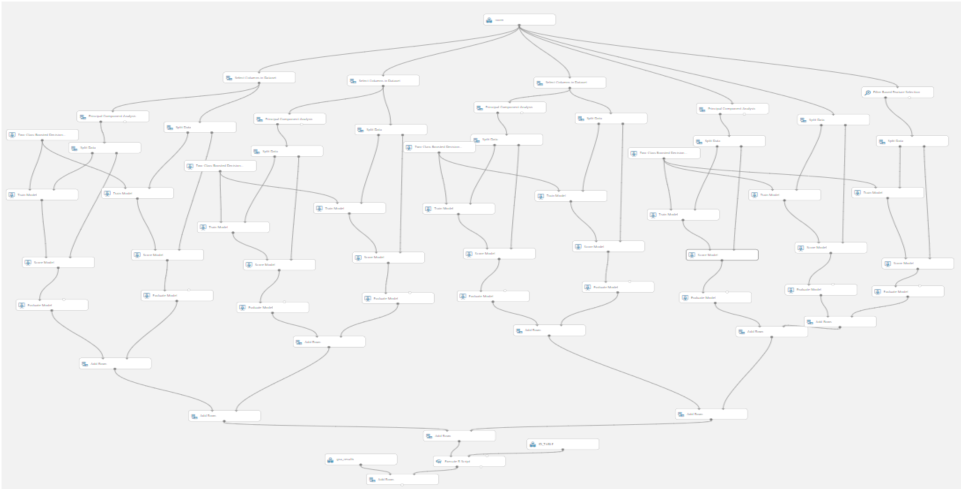

Figure 6.8: Two Class BD normalized experiment

The three data flows are similar just the used attributes changes. In Figure 6.9, we show Accelerometer attributes. In total there are 60 fields: 1 for target, 11 for dummy variables created from Activity table and 48 Accelerometer fields. In all sensor models, we use Activity table fields because the sensor values can change from type of activity. For example, sitting or walking can directly change all sensors. So, we add all activity table fields for our sensor models.

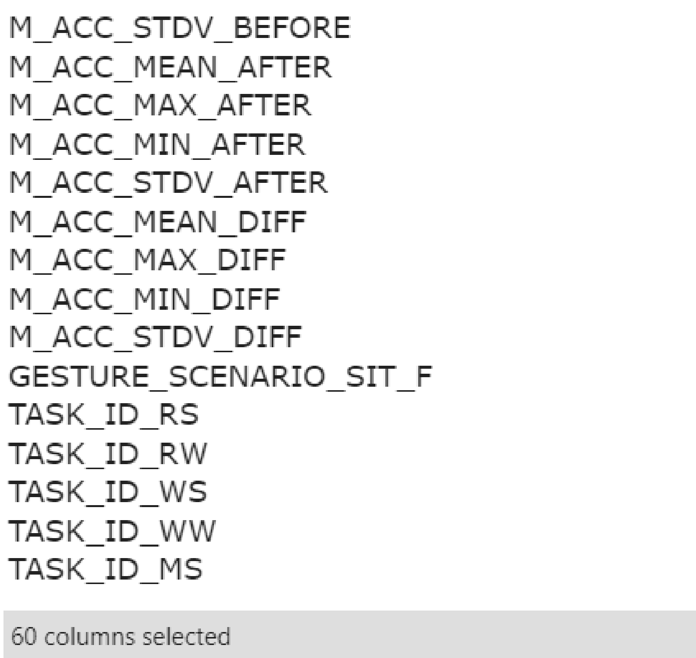


Figure 6.9: Some of Accelerometer fields

After selecting columns, we have two connections one for splitting data and the other for PCA. For all models except 30COLS, we have two flows: data without PCA and data with PCA. In Figure 6.10, we show in detail. Note: Figure 6.10 is a part of Figure 6.8.

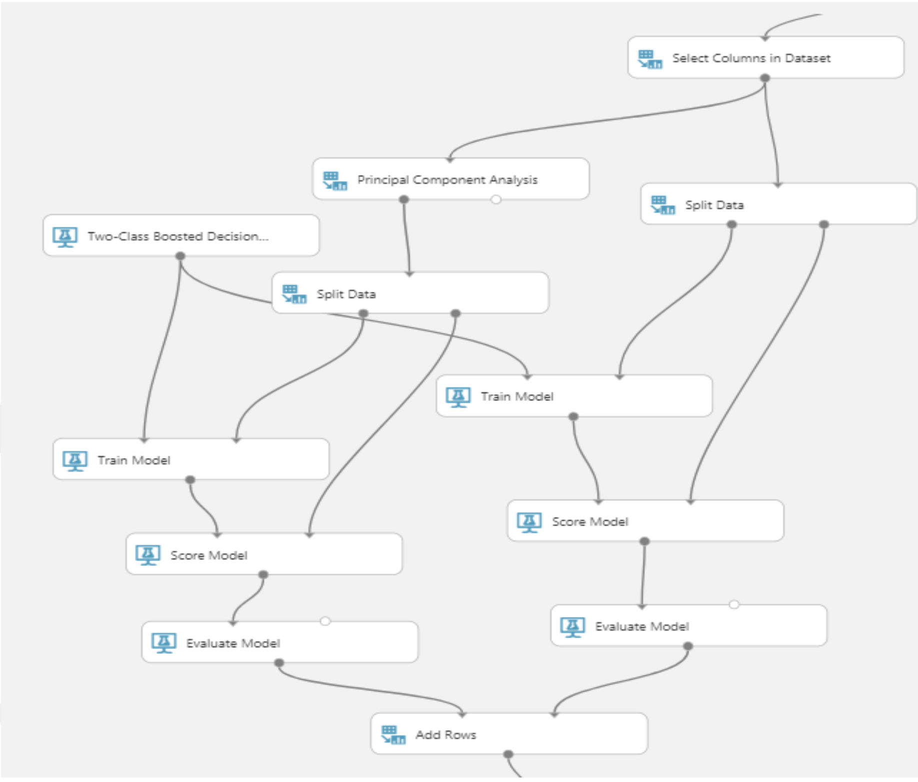


Figure 6.10: Data flow for Accelerometer model

Here for PCA, we take “Number of dimensions to reduce to” as 1/3 of attribute count. For Accelerometer, we have 60 fields. One of them is target. If we exclude target we have 59 fields, one third of 59 is: 59 / 3 = 19. So for PCA, we take “Number of dimensions to reduce to” as 19.

In Figure 6.10, we have two Split data items. One of it splitting data for PCA taken, and the other is splitting data without PCA. For our models, we use %70 or data for training and % 30 of data for testing.

For BD, we use Two Class Boosted Decision Tree classifier. And we set its parameters as the values in Table 5.1. Note: for all 9 BD data flows in this experiment we use parameters from Table 5.1.

After Classifier we have two Train model items. We train the models with %70 of data coming from Split Data item.

After training model, we use Score model to score the model. For scoring model, we use remaining %30 of data.

After that, we evaluate model to see model performances. The Evaluate model calculates; Accuracy, F1 Score, Precision, Recall, Negative Precision, Negative Recall, and Cumulative AUC for model performance. It divides all records by 10 percent probability bins. (90-100] bins keeps the last probability. It also shows the concussion matrix and ROC curve for the model.

At Figure 6.11, we show confusion matrix for BD with normalized data for user 841866. The data flow is ALL model. At Table 6.4, we show Accuracy for all Score Bins. Note: at confusion matrix we show overall accuracy and other matrices. In Figure 6.12, we show ROC curve for same run.

| True Positive | False Negative | Accuracy | Precision |
|---|---|---|---|
| 23561 | 2075 | 0.984 | 0.974 |
| **False Positive** | **True Negative** | **Recall** | **F1 Score** |
| 618 | 146067 | 0.919 | 0.946 |
| **Positive Label** | **Negative Label** | | |
| USER | NO_USER | | |

Figure 6.11: Confusion Matrix

Table 6.4: Accuracy for all Score Bins

| Score Bin | Positive Examples | Negative Examples | Fraction Above Threshold | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|---|---|---|
| (0.900,1.000] | 14812 | 30 | 0.086 | 0.937 | 0.732 | 0.998 | 0.578 |
| (0.800,0.900] | 4360 | 52 | 0.112 | 0.962 | 0.854 | 0.996 | 0.748 |
| (0.700,0.800] | 2682 | 150 | 0.128 | 0.977 | 0.916 | 0.989 | 0.852 |
| (0.600,0.700] | 1698 | 383 | 0.140 | 0.984 | 0.946 | 0.975 | 0.919 |
| (0.500,0.600] | 1009 | 829 | 0.151 | 0.985 | 0.951 | 0.944 | 0.958 |
| (0.400,0.500] | 7 | 6 | 0.151 | 0.985 | 0.951 | 0.944 | 0.958 |
| (0.300,0.400] | 591 | 1752 | 0.165 | 0.979 | 0.932 | 0.887 | 0.981 |
| (0.200,0.300] | 296 | 4167 | 0.190 | 0.956 | 0.871 | 0.775 | 0.993 |
| (0.100,0.200] | 134 | 13191 | 0.268 | 0.880 | 0.713 | 0.554 | 0.998 |
| (0.000,0.100] | 47 | 126125 | 1.000 | 0.149 | 0.259 | 0.149 | 1.000 |

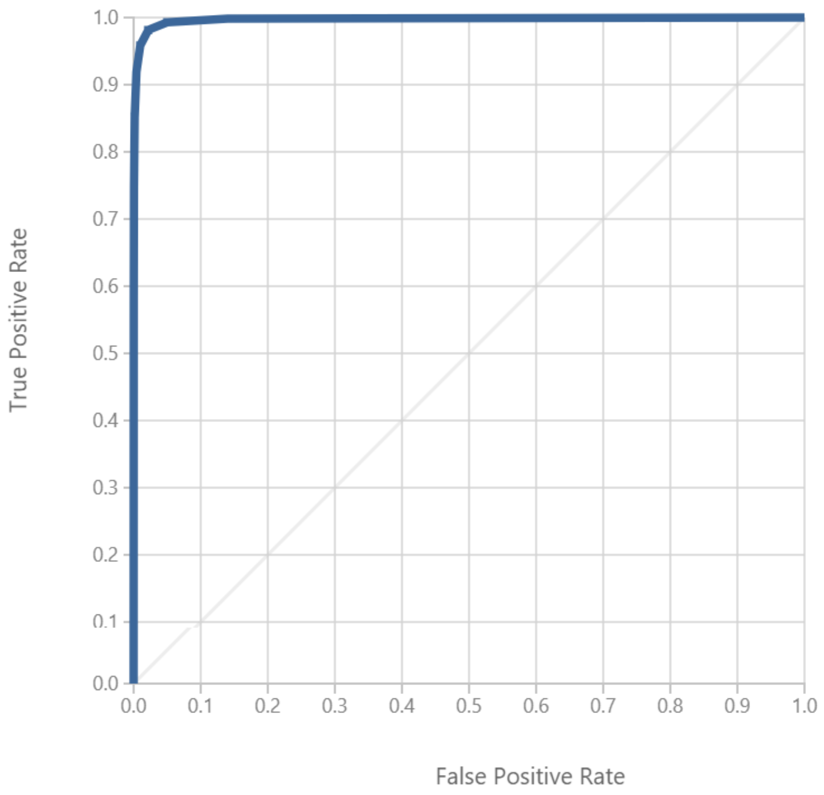


Figure 6.12: ROC curve

Finally we add ACC results and ACC with PCA results for evaluating all model performances.

In this experiment, we modeled sensor attributes alone to see if any of three sensor will be enough for user authentication. Maybe data flow from one sensor can corrupt but if the other sensors are alive, getting separate sensor models will help for user authentication in any case. Moreover, using one sensor models can be cheaper and needs less energy for authentication. For all these reasons, we create 6 sensor models (2 for ACC, 2 for GRY, and 2 for MAG) and evaluate their performances.

In Figure 6.7, we have three more connection coming from “norm” table. The two connections for ALL models (ALL and ALL with PCA) and the other is for 30COLS. For ALL again we used data as without PCA or data with PCA. For 30COLS, we did not take PCA.

In Figure 6.13, we show some part of ALL and 30COLS models. Note: Figure 6.13 is a part of Figure 6.8.

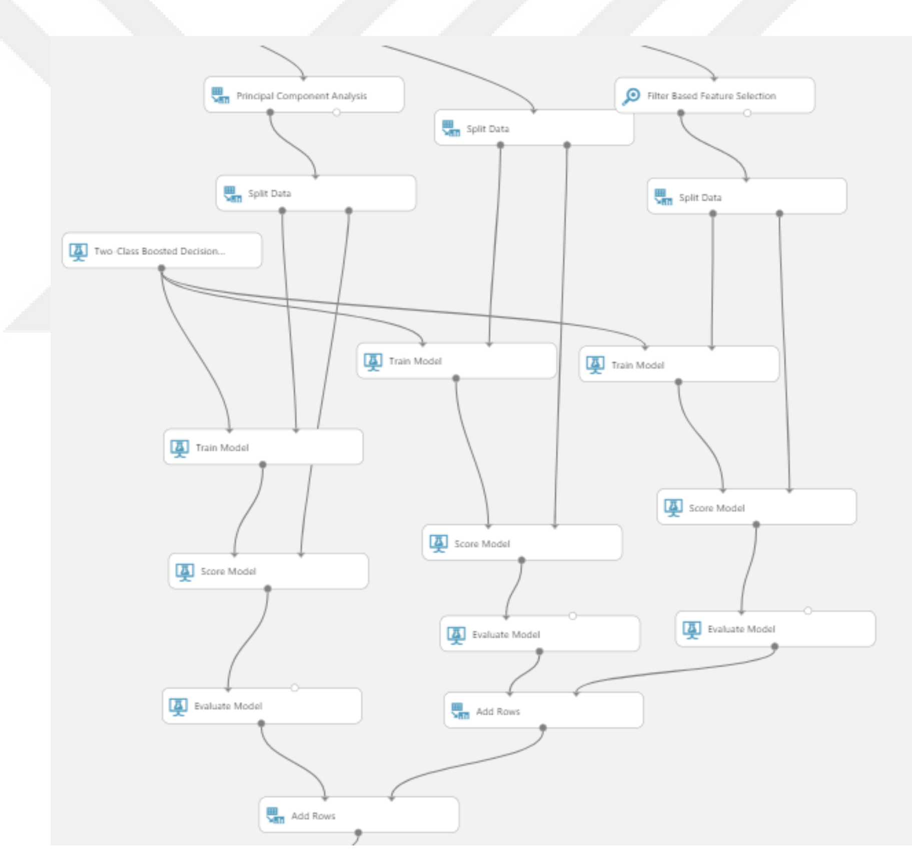


Figure 6.13: Some part of ALL and 30COLS models

Here first two connections from right of the figure 6.13 are for ALL models. Again we take PCA for ALL fields. We use 1/3 of total attributes for PCA. In ALL fields we have

168 attribute. If we exclude target we have 167 attributes. Then, one third of 167 is: 167 / 3 = 55. We take 55 dimensions for ALL with PCA.

For 30COLS, we use Filter Based Feature Selection and use Pearson Correlation for feature selection. We set “Number of Desired Feature” to 30. We are sure now, we select first 30 most correlated attributes with target.

Then for all three models (ALL, ALL with PCA and 30COLS), we have Split Data item. Here again we use %70 of data for training and %30 for testing. Again we use Two Class Boosted Decision Tree for classification. And set its parameters from Table 5. Then we score model by using remaining %30 of data. After scoring data we use Evaluate Model item to see performance of models. Finally we add all evaluation result to see overall performance.

At the end of Figure 6.7, we take all evaluation results of models together and save the results. We show this in Figure 6.14 in detail. The first two Add Rows items join rows coming from sensors and from the others respectively. Again we use ID_TABLE to store user ID. The R Script connects results coming from models and ID_TABLE. We store all results in gsu_results table. At every run, we connect gsu_result table with the final results and store the union again into gsu_results table. We do the storing process manually.

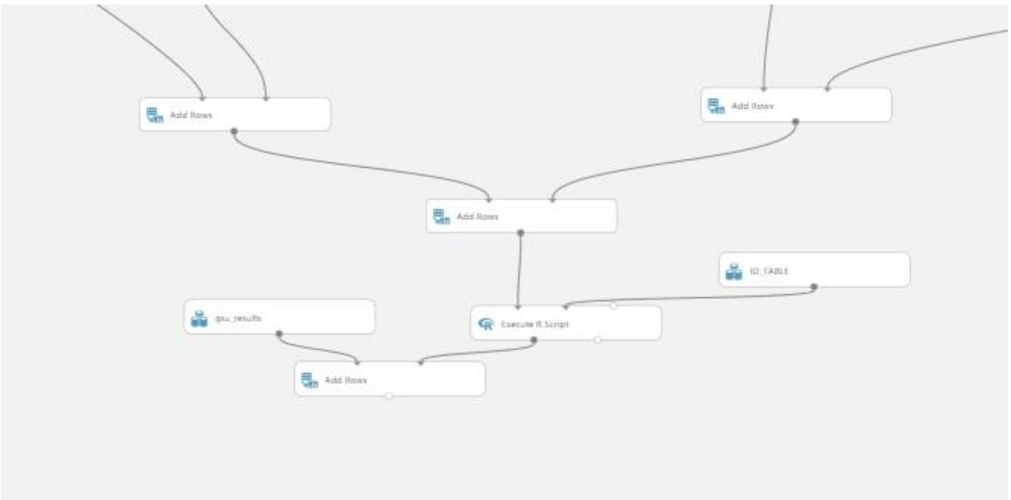


Figure 6.14: Saving Results

After running all experiments for every user we are ready to compare results of models. For one user we run 7 experiments and in total we have 20 * 7 = 140 runs.

## 7. RESULTS

The first results are obtained by taking the average of all 20 users' results. Although various metrics are given, the models are compared by the accuracy perspective. We also store precision and recall results. Our first results about the parameter tuning. For BD, LR and SVM we tuned parameters. For DF we did not tune parameters. We use DF parameters as Azure initial values.

We first tuned SVM parameters. The results of SVM parameters are at Table 7.1: we used Number of iterations as 98, Lambda as 0.003046. These values give the best Accuracy.

Table 7.1. SVM Parameter Values

| Number of iterations | Lambda | Accuracy | Precision | Recall |
|---|---|---|---|---|
| 98 | 0.003046 | 0.828229 | 0.708693 | 0.52781 |
| 86 | 0.099535 | 0.794518 | 0.604628 | 0.507398 |
| 29 | 0.046737 | 0.794302 | 0.603957 | 0.507584 |
| 68 | 0.031466 | 0.794209 | 0.603346 | 0.50898 |
| 63 | 0.046956 | 0.794186 | 0.60363 | 0.507553 |

Then we tuned LR parameters. The results of LR parameters are at Table 7.2: we used Optimization Tolerance as 0.00003, L1 weight as 0.381005, L2 weight as 0.34971 and Memory Size as 48. These values give the best Accuracy.

Table 7.2. LR Parameter Values

| OptimizationTolerance | L1Weight | L2Weight | MemorySize | Accuracy | Precision | Recall |
|---|---|---|---|---|---|---|
| 0.000003 | 0.381005 | 0.34971 | 48 | 0.884355 | 0.804311 | 0.708379 |
| 0.000055 | 0.08111 | 0.195253 | 25 | 0.881447 | 0.799972 | 0.699197 |
| 0.00003 | 0.988544 | 0.64627 | 39 | 0.87745 | 0.799459 | 0.678537 |
| 0.00007 | 0.526284 | 0.934678 | 35 | 0.87462 | 0.79484 | 0.669882 |
| 0.000082 | 0.848052 | 0.991983 | 6 | 0.860216 | 0.765888 | 0.632534 |

Then we tuned BD parameters. The results of BD parameters are at Table 7.3: we used Number of Leaves as 90, Minimum Leaf instances as 26, and Learning rate as 0.375257 and Number of trees as 350. These values give the best Accuracy.

Table 7.3 : BD Parameter Values

| Number of leaves | Minimum leaf instances | Learning rate | Number of trees | Accuracy | Precision | Recall |
|---|---|---|---|---|---|---|
| 90 | 26 | 0.375257 | 350 | 0.993196 | 0.987077 | 0.975555 |
| 39 | 49 | 0.266012 | 386 | 0.99266 | 0.983 | 0.976745 |
| 70 | 4 | 0.095172 | 237 | 0.98912 | 0.979687 | 0.960403 |
| 104 | 42 | 0.396963 | 35 | 0.988199 | 0.973712 | 0.96141 |
| 5 | 19 | 0.153678 | 479 | 0.981704 | 0.955555 | 0.943832 |

We compared sensor performances. While for some users ACC models performs better, for the others MAG model performs better. The reason is that: When we use Pearson correlation to find the correlation of target with attributes, we saw that sometimes it is the magnetometer attributes, which gets higher results than accelerometer attributes, and sometimes it is the opposite case. In Table 7.4, we show top five most correlated attributes with TARGET for all users.

Table 7.4: Most correlated attributes with TARGET

| ID | CORRELATED 1 | CORRELATED 2 | CORRELATED 3 | CORRELATED 4 | CORRELATED 5 |
|---|---|---|---|---|---|
| 745224 | Y_ACC_MAX_BEFORE | Y_ACC_MAX_AFTER | Z_ACC_MEAN_BEFORE | Y_ACC_MEAN_AFTER | Y_ACC_MEAN_BEFORE |
| | 0.526101477917448 | 0.526027094774484 | 0.522512065817992 | 0.517671665718106 | 0.517449449850825 |
| 352716 | Z_ACC_MAX_BEFORE | Y_ACC_MIN_AFTER | Y_ACC_MIN_BEFORE | Z_ACC_MAX_AFTER | Z_ACC_MEAN_BEFORE |
| | 0.31494038435346 | 0.289254903725818 | 0.288017204636527 | 0.28704661486678 | 0.273874370108157 |
| 219303 | Y_MAG_MIN_AFTER | Y_MAG_MEAN_AFTER | Y_MAG_MAX_AFTER | Y_MAG_MIN_BEFORE | Y_MAG_MEAN_BEFORE |
| | 0.275500852736378 | 0.275243046883244 | 0.27496804219146 | 0.272805645166704 | 0.272492784722381 |
| 501973 | CONTACT_SIZE | M_GYR_MAX_AFTER | M_ACC_STDV_AFTER | M_GYR_MEAN_AFTER | M_ACC_MAX_BEFORE |
| | 0.301796412920593 | 0.238582206332669 | 0.229281328856425 | 0.225511080336985 | 0.225108059171782 |
| 264325 | X_MAG_MAX_BEFORE | X_MAG_MEAN_BEFORE | X_MAG_MIN_BEFORE | X_MAG_MAX_AFTER | X_MAG_MEAN_AFTER |
| | 0.268759433468619 | 0.268461737089369 | 0.26822403387547 | 0.267922212908424 | 0.267660345187204 |
| 527796 | X_GYR_STDV_AFTER | Z_ACC_STDV_AFTER | M_ACC_STDV_AFTER | X_GYR_STDV_BEFORE | Z_ACC_STDV_BEFORE |
| | 0.336968464687707 | 0.333874416733462 | 0.326149157965839 | 0.319573415872206 | 0.295362838411421 |
| 862649 | X_ACC_MIN_BEFORE | Y_MAG_MAX_BEFORE | Y_MAG_MAX_AFTER | Y_MAG_MEAN_BEFORE | Y_MAG_MEAN_AFTER |
| | 0.152423855213381 | 0.151665152754996 | 0.151243843796255 | 0.151060324524325 | 0.150699822174548 |
| 663153 | PHONE_ORIENTATION1 | PHONE_ORIENTATION0 | X_ACC_MAX_BEFORE | X_ACC_MEAN_BEFORE | X_ACC_MEAN_AFTER |
| | 0.653784286723699 | 0.589865039787968 | 0.528486460795191 | 0.528277768166644 | 0.527682453389752 |
| 556357 | Z_MAG_MIN_BEFORE | Z_MAG_MEAN_BEFORE | Z_MAG_MAX_BEFORE | Z_MAG_MIN_AFTER | Z_MAG_MEAN_AFTER |
| | 0.267229361060606 | 0.267008026163402 | 0.266757442851155 | 0.254351616657512 | 0.25426544228856 |
| 923862 | M_ACC_MEAN_AFTER | M_ACC_MEAN_BEFORE | M_ACC_MIN_AFTER | M_ACC_MAX_AFTER | M_ACC_MAX_BEFORE |
| | 0.286352911307566 | 0.280610217036483 | 0.258239428924014 | 0.247442679474636 | 0.244429037975945 |
| 815316 | PHONE_ORIENTATION0 | Y_ACC_MAX_BEFORE | Y_ACC_MAX_AFTER | Y_ACC_MEAN_BEFORE | Y_ACC_MEAN_AFTER |
| | 0.896122822699534 | 0.784132912462087 | 0.771949335739134 | 0.743498858762641 | 0.734805086511852 |
| 733162 | Z_ACC_MIN_BEFORE | Z_ACC_MIN_AFTER | Z_ACC_MEAN_BEFORE | Z_ACC_MEAN_AFTER | Z_ACC_MAX_BEFORE |
| | 0.391518650967635 | 0.387015884185687 | 0.374153409295053 | 0.367525068939596 | 0.353763303540903 |
| 472761 | Z_ACC_MAX_BEFORE | Z_ACC_MAX_AFTER | M_ACC_MAX_BEFORE | Z_ACC_MEAN_AFTER | Z_ACC_MEAN_BEFORE |
| | 0.294705192453602 | 0.291713758896863 | 0.278452885709171 | 0.27467140204743 | 0.274397884308291 |
| 897652 | Y_MAG_MIN_AFTER | Y_MAG_MEAN_AFTER | Y_MAG_MIN_BEFORE | Y_MAG_MAX_AFTER | Y_MAG_MEAN_BEFORE |
| | 0.349747792237405 | 0.349375898770985 | 0.349203661091282 | 0.348967851406809 | 0.348799815368137 |
| 186676 | M_ACC_MIN_AFTER | M_ACC_MIN_BEFORE | M_ACC_MEAN_BEFORE | M_ACC_MEAN_AFTER | M_MAG_MAX_AFTER |
| | 0.355140113982211 | 0.346975516833136 | 0.322640215508059 | 0.311971895598156 | 0.266452236290741 |
| 998757 | Y_ACC_MEAN_BEFORE | Y_ACC_MIN_BEFORE | Y_ACC_MAX_BEFORE | Y_ACC_MIN_AFTER | Y_ACC_MEAN_AFTER |
| | 0.463621596093162 | 0.463141254211321 | 0.45931382519527 | 0.452386156859775 | 0.451366805284635 |
| 872895 | X_MAG_MIN_AFTER | X_MAG_MIN_BEFORE | X_MAG_MEAN_AFTER | X_MAG_MAX_AFTER | X_MAG_MEAN_BEFORE |
| | 0.333847952479681 | 0.33296365440044 | 0.332794896893683 | 0.331772985523584 | 0.331558306075711 |
| 240168 | M_ACC_MAX_BEFORE | M_ACC_MAX_AFTER | M_ACC_MEAN_BEFORE | M_ACC_MEAN_AFTER | M_ACC_MIN_AFTER |

| | | | | | |
|---|---|---|---|---|---|
| | 0.444019869519928 | 0.438917418637273 | 0.416556642510241 | 0.411703916002569 | 0.30807696781216 |
| 151985 | M_MAG_MAX_BEFORE | M_MAG_MEAN_BEFORE | M_MAG_MIN_BEFORE | M_MAG_MAX_AFTER | M_MAG_MEAN_AFTER |
| | 0.751201252427723 | 0.750532188464783 | 0.749848916132387 | 0.704952367349935 | 0.704333207327102 |
| 841866 | CONTACT_SIZE | Y_ACC_MAX_AFTER | Y_ACC_MAX_BEFORE | Y_ACC_MEAN_AFTER | Y_ACC_MEAN_BEFORE |
| | 0.28135342230165 | 0.253835484000719 | 0.252479429249677 | 0.233841627542155 | 0.231490271890425 |

For correlation test we also measure the how many times an attribute comes in first five location. At Table 7.5 we show how many times an attribute comes as a most correlated attribute, as a second most correlated attribute, as a third  most correlated attribute etc.

Table 7.5. How many times an attribute is one of most correlated

| CORRELATED 1 | # | CORRELATED 2 | # | CORRELATED 3 | # | CORRELATED 4 | # | CORRELATED 5 | # |
|---|---|---|---|---|---|---|---|---|---|
| CONTACT_SIZE | 2 | Y_ACC_MAX_AFTER | 2 | M_ACC_MEAN_BEFORE | 2 | M_ACC_MEAN_AFTER | 2 | M_ACC_MAX_BEFORE | 2 |
| Y_MAG_MIN_AFTER | 2 | Y_MAG_MEAN_AFTER | 2 | M_ACC_STDV_AFTER | 2 | X_MAG_MAX_AFTER | 2 | Y_ACC_MEAN_AFTER | 2 |
| Z_ACC_MAX_BEFORE | 2 | M_ACC_MAX_AFTER | 1 | Y_ACC_MAX_BEFORE | 2 | Y_ACC_MEAN_AFTER | 2 | Y_ACC_MEAN_BEFORE | 2 |
| M_ACC_MAX_BEFORE | 1 | M_ACC_MEAN_BEFORE | 1 | Y_MAG_MAX_AFTER | 2 | Z_ACC_MEAN_AFTER | 2 | Y_MAG_MEAN_BEFORE | 2 |
| M_ACC_MEAN_AFTER | 1 | M_ACC_MIN_BEFORE | 1 | Z_ACC_MEAN_BEFORE | 2 | M_ACC_MAX_AFTER | 1 | Z_ACC_MEAN_BEFORE | 2 |
| M_ACC_MIN_AFTER | 1 | M_GYR_MAX_AFTER | 1 | M_ACC_MAX_BEFORE | 1 | M_GYR_MEAN_AFTER | 1 | M_ACC_MIN_AFTER | 1 |
| M_MAG_MAX_BEFORE | 1 | M_MAG_MEAN_BEFORE | 1 | M_ACC_MIN_AFTER | 1 | M_MAG_MAX_AFTER | 1 | M_MAG_MAX_AFTER | 1 |
| PHONE_ORIENTATION0 | 1 | PHONE_ORIENTATION0 | 1 | M_MAG_MIN_BEFORE | 1 | X_ACC_MEAN_BEFORE | 1 | M_MAG_MEAN_AFTER | 1 |
| PHONE_ORIENTATION1 | 1 | X_MAG_MEAN_BEFORE | 1 | X_ACC_MAX_BEFORE | 1 | X_GYR_STDV_BEFORE | 1 | X_ACC_MEAN_AFTER | 1 |
| X_ACC_MIN_BEFORE | 1 | X_MAG_MIN_BEFORE | 1 | X_MAG_MEAN_AFTER | 1 | Y_ACC_MEAN_BEFORE | 1 | X_MAG_MEAN_AFTER | 1 |
| X_GYR_STDV_AFTER | 1 | Y_ACC_MAX_BEFORE | 1 | X_MAG_MIN_BEFORE | 1 | Y_ACC_MIN_AFTER | 1 | X_MAG_MEAN_BEFORE | 1 |
| X_MAG_MAX_BEFORE | 1 | Y_ACC_MIN_AFTER | 1 | Y_ACC_MAX_AFTER | 1 | Y_MAG_MAX_AFTER | 1 | Y_MAG_MEAN_AFTER | 1 |
| X_MAG_MIN_AFTER | 1 | Y_ACC_MIN_BEFORE | 1 | Y_ACC_MIN_BEFORE | 1 | Y_MAG_MEAN_BEFORE | 1 | Z_ACC_MAX_BEFORE | 1 |
| Y_ACC_MAX_BEFORE | 1 | Y_MAG_MAX_BEFORE | 1 | Y_MAG_MIN_BEFORE | 1 | Y_MAG_MIN_BEFORE | 1 | Z_ACC_STDV_BEFORE | 1 |
| Y_ACC_MEAN_BEFORE | 1 | Z_ACC_MAX_AFTER | 1 | Z_MAG_MAX_BEFORE | 1 | Z_ACC_MAX_AFTER | 1 | Z_MAG_MEAN_AFTER | 1 |
| Z_ACC_MIN_BEFORE | 1 | Z_ACC_MIN_AFTER | 1 | | | Z_MAG_MIN_AFTER | 1 | | |
| Z_MAG_MIN_BEFORE | 1 | Z_ACC_STDV_AFTER | 1 | | | | | | |
| | | Z_MAG_MEAN_BEFORE | 1 | | | | | | |

It shows us that the parameters completely changes from user to user. In some users sensor data gives best results for some users touch event data gives best results.

When we compare sensor results with respect to models we see that Magnetometer performs better in DF and BD but Accelerometer performs better in LR and SVM. But the values of Magnetometer and Accelerometer are so close. So we can not say that Magnetometer gives better perfomace.

In Table 7.6, we show DF results for sensor, in Table 7.7 we show BD results for sensors, in Table 7.8 we show LR results for sensor and in Table 7.9 we show SVM results for sensors. In all tables there are 18 results because we use both normalized and de normalized data for BD,LR and DF. But for SVM we have 9 results because for SVM we just used normalized data. Note : the results are avarage of all 20 model users.

Table 7.6: DF Results. Normalized and De Normalized data

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | Normalized | 0.992055 | 0.981225 | 0.967784 |
| MAG | Normalized | 0.984081 | 0.953795 | 0.946984 |
| 30COLS | Normalized | 0.968247 | 0.923459 | 0.86851 |
| ACC | Normalized | 0.964171 | 0.917219 | 0.859335 |
| ALL with PCA | Normalized | 0.961614 | 0.942086 | 0.80115 |
| MAG with PCA | Normalized | 0.960821 | 0.909617 | 0.839957 |
| ACC with PCA | Normalized | 0.950292 | 0.888859 | 0.797156 |
| GRY | Normalized | 0.916732 | 0.875118 | 0.614414 |
| GRY with PCA | Normalized | 0.902699 | 0.853346 | 0.541459 |

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | De-Normalized | 0.994229 | 0.987369 | 0.976155 |
| MAG | De-Normalized | 0.993593 | 0.978965 | 0.981824 |
| ALL with PCA | De-Normalized | 0.989785 | 0.982921 | 0.951127 |
| MAG with PCA | De-Normalized | 0.988962 | 0.973025 | 0.959094 |
| 30COLS | De-Normalized | 0.97816 | 0.95089 | 0.916837 |
| ACC | De-Normalized | 0.967054 | 0.92807 | 0.874002 |
| ACC with PCA | De-Normalized | 0.963162 | 0.922011 | 0.856201 |
| GRY | De-Normalized | 0.918862 | 0.884842 | 0.630053 |
| GRY with PCA | De-Normalized | 0.912008 | 0.85397 | 0.614659 |

Table 7.7: BD Results. Normalized and De Normalized data

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | Normalized | 0.996111 | 0.982467 | 0.991248 |
| ALL with PCA | Normalized | 0.983862 | 0.953004 | 0.947445 |
| MAG | Normalized | 0.983041 | 0.943442 | 0.954194 |
| 30COLS | Normalized | 0.972161 | 0.910419 | 0.920293 |
| ACC | Normalized | 0.963021 | 0.886518 | 0.895616 |
| MAG with PCA | Normalized | 0.960447 | 0.87903 | 0.881583 |
| ACC with PCA | Normalized | 0.948162 | 0.844382 | 0.846266 |
| GRY | Normalized | 0.905658 | 0.745916 | 0.697098 |
| GRY with PCA | Normalized | 0.8837 | 0.682679 | 0.612755 |

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | De-Normalized | 0.997317 | 0.987706 | 0.994475 |
| ALL with PCA | De-Normalized | 0.994193 | 0.978985 | 0.983608 |
| MAG | De-Normalized | 0.992348 | 0.970046 | 0.984742 |
| MAG with PCA | De-Normalized | 0.986981 | 0.955986 | 0.96781 |
| 30COLS | De-Normalized | 0.983635 | 0.946876 | 0.957458 |
| ACC | De-Normalized | 0.969493 | 0.907356 | 0.918888 |
| ACC with PCA | De-Normalized | 0.963485 | 0.890274 | 0.897898 |
| GRY | De-Normalized | 0.914418 | 0.774275 | 0.740073 |
| GRY with PCA | De-Normalized | 0.895709 | 0.722844 | 0.675208 |

Table 7.8: LR Results. Normalized and De Normalized data

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | Normalized | 0.924224 | 0.789046 | 0.686001 |
| ALL with PCA | Normalized | 0.905942 | 0.73715 | 0.588172 |
| 30COLS | Normalized | 0.892781 | 0.709761 | 0.520253 |
| ACC | Normalized | 0.87534 | 0.654405 | 0.431529 |
| MAG | Normalized | 0.862697 | 0.606739 | 0.272891 |
| ACC with PCA | Normalized | 0.862243 | 0.600669 | 0.345911 |
| MAG with PCA | Normalized | 0.856722 | 0.548401 | 0.233163 |
| GRY | Normalized | 0.835687 | 0.499719 | 0.154874 |
| GRY with PCA | Normalized | 0.828695 | 0.387521 | 0.078386 |

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | De-Normalized | 0.918873 | 0.790926 | 0.652578 |
| ALL with PCA | De-Normalized | 0.905617 | 0.767943 | 0.569884 |
| 30COLS | De-Normalized | 0.889927 | 0.723636 | 0.479914 |
| MAG | De-Normalized | 0.870602 | 0.653397 | 0.348817 |
| ACC | De-Normalized | 0.86594 | 0.653944 | 0.374243 |
| MAG with PCA | De-Normalized | 0.861988 | 0.580987 | 0.279547 |
| ACC with PCA | De-Normalized | 0.858432 | 0.612202 | 0.326387 |
| GRY | De-Normalized | 0.829698 | 0.45738 | 0.129285 |
| GRY with PCA | De-Normalized | 0.825401 | 0.457932 | 0.102514 |

Table 7.9: SVM Results

| Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| ALL | Normalized | 0.913508 | 0.786037 | 0.618775 |
| ALL with PCA | Normalized | 0.897431 | 0.766317 | 0.531923 |
| 30COLS | Normalized | 0.890139 | 0.743792 | 0.502651 |
| ACC | Normalized | 0.869704 | 0.678231 | 0.409189 |
| ACC with PCA | Normalized | 0.860143 | 0.622211 | 0.347434 |
| MAG | Normalized | 0.859586 | 0.637857 | 0.25921 |
| MAG with PCA | Normalized | 0.855579 | 0.625727 | 0.234166 |
| GRY | Normalized | 0.830493 | 0.420945 | 0.131047 |
| GRY with PCA | Normalized | 0.825175 | 0.322237 | 0.084511 |

Since the most correlated attributes changes from user to user, the 30COLS models usually use different attributes. PCA used model performances are lower compared to models without PCA. But again the results are so close, so we cannot say taking PCA lowers the Accuracy.

During the experiments, we explored that in all cases, BD usually gives higher accuracy. Following BD, DF has the second rank among accuracy comparisons. In some cases DF gives better results. Note: we do not take parameter tuning in DF. Maybe it will give better results if we tune it. From Table 7.10 to Table 7.18, we show machine learning algorithm results for different data flow. These are avarage of 20 model users' results.

Table 7.10: 30COLS Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | Normalized | 0.972161 | 0.910419 | 0.920293 |
| DF | Normalized | 0.968247 | 0.923459 | 0.86851 |
| LR | Normalized | 0.892781 | 0.709761 | 0.520253 |
| SVM | Normalized | 0.890139 | 0.743792 | 0.502651 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | De-Normalized | 0.983635 | 0.946876 | 0.957458 |
| DF | De-Normalized | 0.97816 | 0.95089 | 0.916837 |
| LR | De-Normalized | 0.889927 | 0.723636 | 0.479914 |

Table 7.11: ACC Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.964171 | 0.917219 | 0.859335 |
| BD | Normalized | 0.963021 | 0.886518 | 0.895616 |
| LR | Normalized | 0.87534 | 0.654405 | 0.431529 |
| SVM | Normalized | 0.869704 | 0.678231 | 0.409189 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | De-Normalized | 0.969493 | 0.907356 | 0.918888 |
| DF | De-Normalized | 0.967054 | 0.92807 | 0.874002 |
| LR | De-Normalized | 0.86594 | 0.653944 | 0.374243 |

Table 7.12: ACC with PCA Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.950292 | 0.888859 | 0.797156 |
| BD | Normalized | 0.948162 | 0.844382 | 0.846266 |
| LR | Normalized | 0.862243 | 0.600669 | 0.345911 |
| SVM | Normalized | 0.860143 | 0.622211 | 0.347434 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | De-Normalized | 0.963485 | 0.890274 | 0.897898 |
| DF | De-Normalized | 0.963162 | 0.922011 | 0.856201 |
| LR | De-Normalized | 0.858432 | 0.612202 | 0.326387 |

Table 7.13: ALL Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | Normalized | 0.996111 | 0.982467 | 0.991248 |
| DF | Normalized | 0.992055 | 0.981225 | 0.967784 |
| LR | Normalized | 0.924224 | 0.789046 | 0.686001 |
| SVM | Normalized | 0.913508 | 0.786037 | 0.618775 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | De-Normalized | 0.997317 | 0.987706 | 0.994475 |
| DF | De-Normalized | 0.994229 | 0.987369 | 0.976155 |
| LR | De-Normalized | 0.918873 | 0.790926 | 0.652578 |

Table 7.14: ALL with PCA Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | Normalized | 0.983862 | 0.953004 | 0.947445 |
| DF | Normalized | 0.961614 | 0.942086 | 0.80115 |
| LR | Normalized | 0.905942 | 0.73715 | 0.588172 |
| SVM | Normalized | 0.897431 | 0.766317 | 0.531923 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| BD | De-Normalized | 0.994193 | 0.978985 | 0.983608 |
| DF | De-Normalized | 0.989785 | 0.982921 | 0.951127 |
| LR | De-Normalized | 0.905617 | 0.767943 | 0.569884 |

Table 7.15: GRY Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.916732 | 0.875118 | 0.614414 |
| BD | Normalized | 0.905658 | 0.745916 | 0.697098 |
| LR | Normalized | 0.835687 | 0.499719 | 0.154874 |
| SVM | Normalized | 0.830493 | 0.420945 | 0.131047 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | De-Normalized | 0.918862 | 0.884842 | 0.630053 |
| BD | De-Normalized | 0.914418 | 0.774275 | 0.740073 |
| LR | De-Normalized | 0.829698 | 0.45738 | 0.129285 |

Table 7.16: GRY with PCA Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.902699 | 0.853346 | 0.541459 |
| BD | Normalized | 0.8837 | 0.682679 | 0.612755 |
| LR | Normalized | 0.828695 | 0.387521 | 0.078386 |
| SVM | Normalized | 0.825175 | 0.322237 | 0.084511 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | De-Normalized | 0.912008 | 0.85397 | 0.614659 |
| BD | De-Normalized | 0.895709 | 0.722844 | 0.675208 |
| LR | De-Normalized | 0.825401 | 0.457932 | 0.102514 |

Table 7.17: MAG Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.984081 | 0.953795 | 0.946984 |
| BD | Normalized | 0.983041 | 0.943442 | 0.954194 |
| LR | Normalized | 0.862697 | 0.606739 | 0.272891 |
| SVM | Normalized | 0.859586 | 0.637857 | 0.25921 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | De-Normalized | 0.993593 | 0.978965 | 0.981824 |
| BD | De-Normalized | 0.992348 | 0.970046 | 0.984742 |
| LR | De-Normalized | 0.870602 | 0.653397 | 0.348817 |

Table 7.18: MAG with PCA Results. Normalized and De Normalized data

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | Normalized | 0.960821 | 0.909617 | 0.839957 |
| BD | Normalized | 0.960447 | 0.87903 | 0.881583 |
| LR | Normalized | 0.856722 | 0.548401 | 0.233163 |
| SVM | Normalized | 0.855579 | 0.625727 | 0.234166 |

| Model | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|
| DF | De-Normalized | 0.988962 | 0.973025 | 0.959094 |
| BD | De-Normalized | 0.986981 | 0.955986 | 0.96781 |
| LR | De-Normalized | 0.861988 | 0.580987 | 0.279547 |

The accuracy values of each algorithm for each data flow are presented in Figure 7.1-Figure 7.5. We were expecting normalizing data will give better results but after our results come, we see that in nearly all cases de normalized data gave better results. But again, the value are so close so we cannot easily say that data normalization or de normalization helps for predictions.

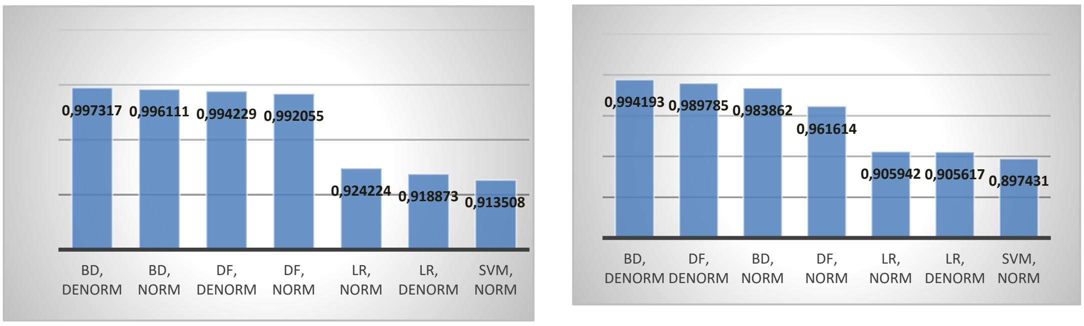


Figure 7.1: (a) Accuracy values for 'ALL' data flow; (b) Accuracy values for 'ALL with PCA' data flow.

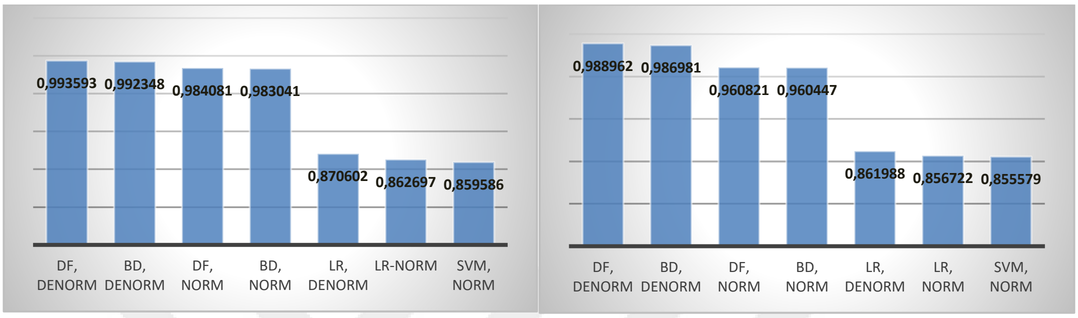


Figure 7.2: (a) Accuracy values for 'MAG' data flow; (b) Accuracy values for 'MAG with PCA' data flow.

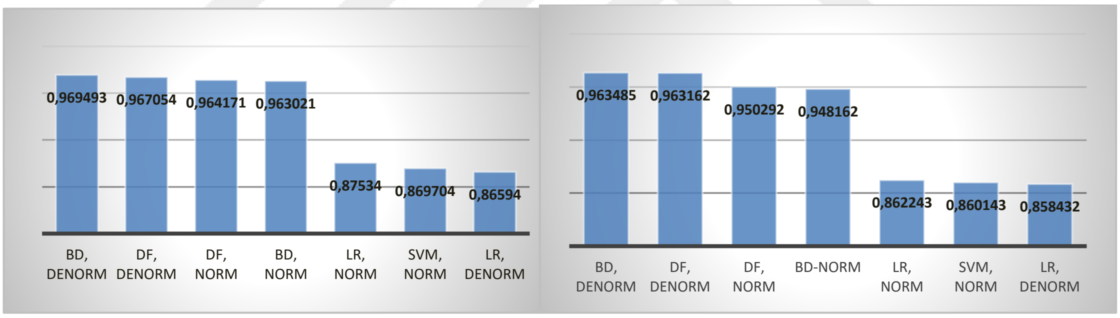


Figure 7.3: (a) Accuracy values for 'ACC' data flow; (b) Accuracy values for 'ACC with PCA' data flow.

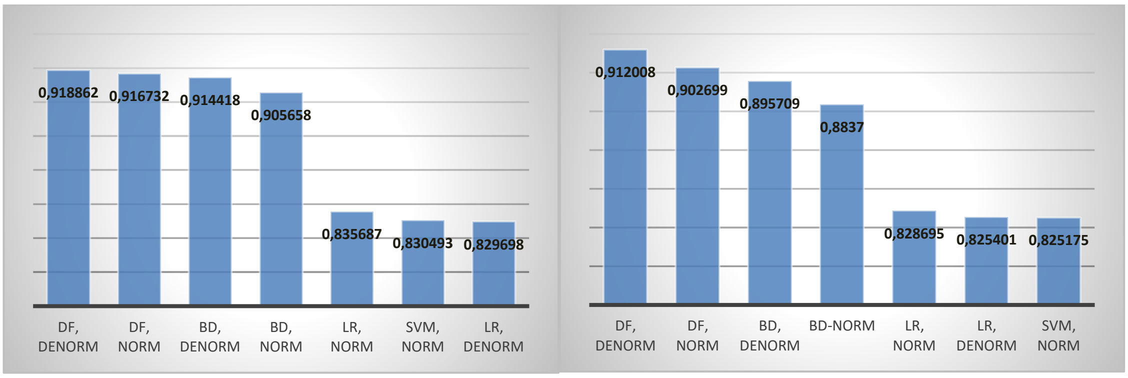


Figure 7.4: (a) Accuracy values for 'GYR' data flow; (b) Accuracy values for 'GYR with PCA' data flow.

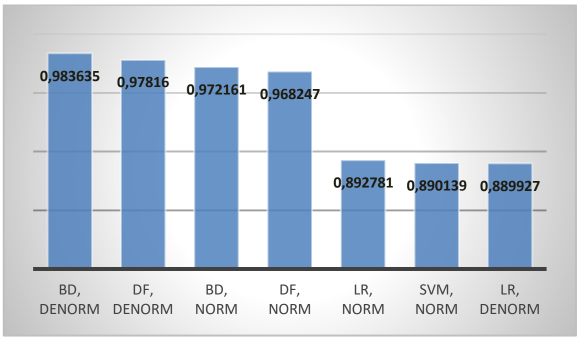


Figure 7.5: Accuracy values for '30COLS' data flow.

At table 7.19, we show for all model users the most predictive model and its Accuracy, Precision and Recall values.

Table 7.19 Most predictive model for all user

| ID | Model | Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|---|---|
| 151985 | BD | ALL | De-Normalized | 0.99946 | 0.99883 | 1 |
| 186676 | BD | ALL | De-Normalized | 0.998534 | 0.993622 | 0.998129 |
| 219303 | BD | ALL with PCA | De-Normalized | 0.995971 | 0.976074 | 0.99034 |
| 240168 | BD | ALL | De-Normalized | 0.999348 | 0.997093 | 0.998694 |
| 264325 | BD | ALL | De-Normalized | 0.996835 | 0.989106 | 0.998692 |
| 352716 | BD | ALL | De-Normalized | 0.998839 | 0.995469 | 0.998243 |
| 472761 | BD | ALL | De-Normalized | 0.997507 | 0.988085 | 0.994633 |
| 501973 | BD | ALL | De-Normalized | 0.997879 | 0.987144 | 0.992931 |
| 527796 | BD | ALL | De-Normalized | 0.996583 | 0.989336 | 0.997303 |
| 556357 | BD | ALL | De-Normalized | 0.998304 | 0.992823 | 0.989831 |
| 663153 | BD | ALL | De-Normalized | 0.998783 | 0.99532 | 0.99579 |
| 733162 | BD | ALL | De-Normalized | 0.999023 | 0.996741 | 0.997265 |
| 745224 | BD | ALL | De-Normalized | 0.997706 | 0.993516 | 0.997285 |
| 815316 | BD | ALL | Normalized | 0.999643 | 0.998656 | 0.999388 |
| 841866 | BD | ALL | Normalized | 0.996727 | 0.987251 | 0.990794 |
| 862649 | BD | ALL | De-Normalized | 0.99816 | 0.990044 | 0.99809 |
| 872895 | BD | ALL | De-Normalized | 0.999048 | 0.994367 | 0.996566 |
| 897652 | BD | ALL | De-Normalized | 0.99947 | 0.997567 | 0.999684 |
| 923862 | BD | ALL | De-Normalized | 0.997682 | 0.991213 | 0.99695 |
| 998757 | BD | ALL | De-Normalized | 0.99737 | 0.988199 | 0.997415 |

At table 7.20, we show for all model users the most predictive sensor model and its Accuracy, Precision and Recall values.

Table 7.20 Most predictive Sensor model for all users

| ID | Model | Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|---|---|
| 151985 | DF | MAG | De-Normalized | 0.998633 | 0.997197 | 0.999844 |
| 186676 | DF | MAG | De-Normalized | 0.996185 | 0.986931 | 0.991595 |
| 219303 | BD | MAG | De-Normalized | 0.994998 | 0.97564 | 0.9824 |
| 240168 | BD | MAG | De-Normalized | 0.997416 | 0.985944 | 0.997499 |
| 264325 | BD | MAG | De-Normalized | 0.988493 | 0.966474 | 0.989575 |
| 352716 | BD | MAG | De-Normalized | 0.996658 | 0.986258 | 0.995729 |
| 472761 | BD | MAG | De-Normalized | 0.992713 | 0.9694 | 0.980199 |
| 501973 | BD | MAG | De-Normalized | 0.993339 | 0.961979 | 0.975746 |
| 527796 | DF | MAG | De-Normalized | 0.990525 | 0.972644 | 0.990579 |
| 556357 | DF | MAG | De-Normalized | 0.997462 | 0.989403 | 0.984621 |
| 663153 | DF | MAG | De-Normalized | 0.995501 | 0.981069 | 0.986168 |
| 733162 | BD | MAG | De-Normalized | 0.992866 | 0.96927 | 0.987519 |
| 745224 | DF | MAG | De-Normalized | 0.991516 | 0.979207 | 0.986838 |
| 815316 | DF | ACC | De-Normalized | 0.997381 | 0.989046 | 0.996697 |
| 841866 | BD | MAG | Normalized | 0.9901 | 0.96358 | 0.97012 |
| 862649 | DF | MAG | De-Normalized | 0.994157 | 0.973599 | 0.988877 |
| 872895 | DF | MAG | De-Normalized | 0.996448 | 0.980545 | 0.985682 |
| 897652 | BD | MAG | De-Normalized | 0.997085 | 0.986842 | 0.998164 |
| 923862 | BD | MAG | De-Normalized | 0.992026 | 0.973724 | 0.985712 |
| 998757 | DF | MAG | De-Normalized | 0.995247 | 0.983171 | 0.990767 |

At Table 7,21 we show the sensors with highest scored model, and the same model with diffrent sensors. For example, for user 745224 DF with De-normalized data with MAG fields gets hishest accurcy value. So we show DF with De-normalized data for other two sensors.

Table 7.21 Accuracy values for All sensors

| ID | Model | Data | Flow | Accuracy | Flow (2) | Accuracy (2) | Flow (3) | Accuracy (3) |
|---|---|---|---|---|---|---|---|---|
| 745224 | DF | De-Normalized | MAG | 0.991516 | ACC | 0.969815 | GRY | 0.906342 |
| 352716 | BD | De-Normalized | MAG | 0.996658 | ACC | 0.978078 | GRY | 0.917494 |
| 219303 | BD | De-Normalized | MAG | 0.994998 | ACC | 0.976587 | GRY | 0.937014 |
| 501973 | BD | De-Normalized | MAG | 0.993339 | ACC | 0.969966 | GRY | 0.945009 |
| 264325 | BD | De-Normalized | MAG | 0.988493 | ACC | 0.946867 | GRY | 0.86838 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 527796 | DF | De-Normalized | MAG | 0.990525 | ACC | 0.956157 | GRY | 0.892178 |
| 862649 | DF | De-Normalized | MAG | 0.994157 | ACC | 0.958806 | GRY | 0.927815 |
| 663153 | DF | De-Normalized | MAG | 0.995501 | ACC | 0.984793 | GRY | 0.94117 |
| 556357 | DF | De-Normalized | MAG | 0.997462 | ACC | 0.972422 | GRY | 0.958968 |
| 841866 | BD | Normalized | MAG | 0.9901 | ACC | 0.967816 | GRY | 0.914201 |
| 815316 | DF | De-Normalized | ACC | 0.997381 | MAG | 0.996074 | GRY | 0.949063 |
| 472761 | BD | De-Normalized | MAG | 0.992713 | ACC | 0.963858 | GRY | 0.914449 |
| 897652 | BD | De-Normalized | MAG | 0.997085 | ACC | 0.977373 | GRY | 0.911441 |
| 186676 | DF | De-Normalized | MAG | 0.996185 | ACC | 0.965079 | GRY | 0.915062 |
| 998757 | DF | De-Normalized | MAG | 0.995247 | ACC | 0.968441 | GRY | 0.916677 |
| 872895 | DF | De-Normalized | MAG | 0.996448 | ACC | 0.975998 | GRY | 0.951928 |
| 240168 | BD | De-Normalized | MAG | 0.997416 | ACC | 0.985992 | GRY | 0.941615 |
| 151985 | DF | De-Normalized | MAG | 0.998633 | ACC | 0.942081 | GRY | 0.826891 |
| 923862 | BD | De-Normalized | MAG | 0.992026 | ACC | 0.947643 | GRY | 0.903371 |
| 733162 | BD | De-Normalized | MAG | 0.992866 | ACC | 0.981226 | GRY | 0.906829 |

At table 7.22, we show average values of Accuracy, Precision and Recall of all users for all models.

Table 7.22: All Results together.

| Model | Flow | Data | Accuracy | Precision | Recall |
|---|---|---|---|---|---|
| BD | ALL | De-Normalized | 0.997317 | 0.987706 | 0.994475 |
| BD | ALL | Normalized | 0.996111 | 0.982467 | 0.991248 |
| DF | ALL | De-Normalized | 0.994229 | 0.987369 | 0.976155 |
| BD | ALL with PCA | De-Normalized | 0.994193 | 0.978985 | 0.983608 |
| DF | MAG | De-Normalized | 0.993593 | 0.978965 | 0.981824 |
| BD | MAG | De-Normalized | 0.992348 | 0.970046 | 0.984742 |
| DF | ALL | Normalized | 0.992055 | 0.981225 | 0.967784 |
| DF | ALL with PCA | De-Normalized | 0.989785 | 0.982921 | 0.951127 |
| DF | MAG with PCA | De-Normalized | 0.988962 | 0.973025 | 0.959094 |
| BD | MAG with PCA | De-Normalized | 0.986981 | 0.955986 | 0.96781 |
| DF | MAG | Normalized | 0.984081 | 0.953795 | 0.946984 |
| BD | ALL with PCA | Normalized | 0.983862 | 0.953004 | 0.947445 |
| BD | 30COLS | De-Normalized | 0.983635 | 0.946876 | 0.957458 |
| BD | MAG | Normalized | 0.983041 | 0.943442 | 0.954194 |
| DF | 30COLS | De-Normalized | 0.97816 | 0.95089 | 0.916837 |
| BD | 30COLS | Normalized | 0.972161 | 0.910419 | 0.920293 |
| BD | ACC | De-Normalized | 0.969493 | 0.907356 | 0.918888 |
| DF | 30COLS | Normalized | 0.968247 | 0.923459 | 0.86851 |
| DF | ACC | De-Normalized | 0.967054 | 0.92807 | 0.874002 |
| DF | ACC | Normalized | 0.964171 | 0.917219 | 0.859335 |
| BD | ACC with PCA | De-Normalized | 0.963485 | 0.890274 | 0.897898 |
| DF | ACC with PCA | De-Normalized | 0.963162 | 0.922011 | 0.856201 |

| | | | | | |
|---|---|---|---|---|---|
| BD | ACC | Normalized | 0.963021 | 0.886518 | 0.895616 |
| DF | ALL with PCA | Normalized | 0.961614 | 0.942086 | 0.80115 |
| DF | MAG with PCA | Normalized | 0.960821 | 0.909617 | 0.839957 |
| BD | MAG with PCA | Normalized | 0.960447 | 0.87903 | 0.881583 |
| DF | ACC with PCA | Normalized | 0.950292 | 0.888859 | 0.797156 |
| BD | ACC with PCA | Normalized | 0.948162 | 0.844382 | 0.846266 |
| LR | ALL | Normalized | 0.924224 | 0.789046 | 0.686001 |
| LR | ALL | De-Normalized | 0.918873 | 0.790926 | 0.652578 |
| DF | GRY | De-Normalized | 0.918862 | 0.884842 | 0.630053 |
| DF | GRY | Normalized | 0.916732 | 0.875118 | 0.614414 |
| BD | GRY | De-Normalized | 0.914418 | 0.774275 | 0.740073 |
| SVM | ALL | Normalized | 0.913508 | 0.786037 | 0.618775 |
| DF | GRY with PCA | De-Normalized | 0.912008 | 0.85397 | 0.614659 |
| LR | ALL with PCA | Normalized | 0.905942 | 0.73715 | 0.588172 |
| BD | GRY | Normalized | 0.905658 | 0.745916 | 0.697098 |
| LR | ALL with PCA | De-Normalized | 0.905617 | 0.767943 | 0.569884 |
| DF | GRY with PCA | Normalized | 0.902699 | 0.853346 | 0.541459 |
| SVM | ALL with PCA | Normalized | 0.897431 | 0.766317 | 0.531923 |
| BD | GRY with PCA | De-Normalized | 0.895709 | 0.722844 | 0.675208 |
| LR | 30COLS | Normalized | 0.892781 | 0.709761 | 0.520253 |
| SVM | 30COLS | Normalized | 0.890139 | 0.743792 | 0.502651 |
| LR | 30COLS | De-Normalized | 0.889927 | 0.723636 | 0.479914 |
| BD | GRY with PCA | Normalized | 0.8837 | 0.682679 | 0.612755 |
| LR | ACC | Normalized | 0.87534 | 0.654405 | 0.431529 |
| LR | MAG | De-Normalized | 0.870602 | 0.653397 | 0.348817 |
| SVM | ACC | Normalized | 0.869704 | 0.678231 | 0.409189 |
| LR | ACC | De-Normalized | 0.86594 | 0.653944 | 0.374243 |
| LR | MAG | Normalized | 0.862697 | 0.606739 | 0.272891 |
| LR | ACC with PCA | Normalized | 0.862243 | 0.600669 | 0.345911 |
| LR | MAG with PCA | De-Normalized | 0.861988 | 0.580987 | 0.279547 |
| SVM | ACC with PCA | Normalized | 0.860143 | 0.622211 | 0.347434 |
| SVM | MAG | Normalized | 0.859586 | 0.637857 | 0.25921 |
| LR | ACC with PCA | De-Normalized | 0.858432 | 0.612202 | 0.326387 |
| LR | MAG with PCA | Normalized | 0.856722 | 0.548401 | 0.233163 |
| SVM | MAG with PCA | Normalized | 0.855579 | 0.625727 | 0.234166 |
| LR | GRY | Normalized | 0.835687 | 0.499719 | 0.154874 |
| SVM | GRY | Normalized | 0.830493 | 0.420945 | 0.131047 |
| LR | GRY | De-Normalized | 0.829698 | 0.45738 | 0.129285 |
| LR | GRY with PCA | Normalized | 0.828695 | 0.387521 | 0.078386 |
| LR | GRY with PCA | De-Normalized | 0.825401 | 0.457932 | 0.102514 |
| SVM | GRY with PCA | Normalized | 0.825175 | 0.322237 | 0.084511 |

# 8. CONCLUSION

In this paper; the data set of HMOG, which is a set of behavioural biometric features for continuous authentication of smartphone users, is used. The changes in the sensors in terms of acceleration, orientation and magnetic field are detected using different machine learning algorithms. Each data model aims to differentiate the real user of the mobile phone from non-users. The accuracy values of different algorithms with different data flows are presented. The results demonstrate the efficiency of the usage of sensors for continuous user authentication.

From three sensor Magnetometer in terms of accuracy gives best results but values for Accelerometer in terms of Accuracy are so close to Magnetometer Accuracy values. So we cannot easily say that which sensor performs better. But Gyroscope Accuracy values are lower than the Accuracy values of these two sensors.

Taking normalization a bit made Accuracy values lower. But here again the values of Accuracy for normalized and de normalized data are so close for same type of data flow. So we cannot easily say that normalization or de normalization of data affects Accuracy.

Algorithms strongly affects accuracy. BD and DF gives good result compared to SVM and LR. Here boosting and bagging algorisms archives better accuracy. We used linear kernel for SVM and lasso and ridge for logistic regression but two of them gave worse results compared to DF and BD. So we can easily say ensemble algorithms gives better performance.

Taking PCA a bit made Accuracy values lower. But here again the values of Accuracy for data with PCA and data without PCA are so close for same type of data flow. So we cannot easily say that taking PCA of data affects Accuracy.

# 9. FURTHER RESEARCH

In this experiment we examined different kind of machine learning algorithms for mobile phone user authentication. We used PCA and normalization for data manipulation moreover we used different data flows to see different behaviours. In our experiment, we did not measure how quickly our models detect a mobile phone users. We have very high accuracies for ensemble algorithms but we don't know how quickly these algorithms will detect a mobile phone users. So an android application can be done to detect how quickly these algorism performs. If the time for authentication is too slow these type of authentication will be useless.

Moreover, in real time we did not measure how much energy is required for training a model. We used 7 different kind of models and all of them has 9 data flows. But we don't know in real live how much energy will consume those 7 *9 = 63 models. And we don't know if the energy consumption will be enough for model training in real life. Additionally after model training, how much energy will be consumed for implementation phrase. If the energy consumption is too high for model training, these type of authentication will be useless. And also, after model training if the energy consumption is still too high for authentication again these type of attentions will be useless.

Additionally, to train a model we need about 1.2 GB of data. And when we run all models a model user about 7 GB of space is used. This is a bit high for authentication in mobile phones. So how much of data will be required for authentication in mobile phones. If the data is so abundant then there won't be enough disk space in mobile phones. Here both the space for data collection and space for training data should be considered. After collection of data there should be available space for training and for implementation phrase.

The final question is about required CPU and memory usage. To train such kind of models in mobile phone, there should be enough memory and CPU for training. When we try to implement such kind of models, current mobile phones may need extra memory or CPU space.

## REFERENCES

[1] Portal, T.S. (2016) "Number of smartphone users world-wide from 2014 to 2020", online: **URL:** *https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.*

[2] Sitova, Z., Sedenka, J., Yang, Q., Peng, P., Zhou, G., Gasti, P., Balagani, K.S. (2016) "HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users." *IEEE Transactions on Information Forensics and Security* **11(5)**: 877-892.

[3] Patel, V.M, Chellappa, R., Chandra, D., Barbello, B. (2016) "Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges." *IEEE Signal Processing Magazine* **33(4)**, 49-61, doi:10.1109/MSP.2016.2555335.

[4] Alzubaidi, A., Kalita, J. (2016) "Authentication of Smartphone Users Using Behavioral Biometrics." *IEEE Communications Surveys & Tutorials* **18(3)**: 1998-2026.

[5] De Luca, A., Brudy, F., Linder, C., Hang, A., Hussman, H. (2012) "Touch me ones and I know it's you!: Implicit Authentication based on touch screen patterns" *ACM Annual Conf. Hum. Factors Computer System*: 987–996.

[6] Zeng, N, Huang, H., Bai, K., Wang, H. (2014) "You are how you touch: User verification on smartphones via tapping behaviors." *IEEE Int. Conference Networks and Protocols (ICNP)*: 221–232.

[7] Zhao, X, Feng, T., Shi, W. (2013) "Continuous mobile authentication using a novel graphic touch gesture feature." *IEEE 6th International Conference Biom: Theory Appl. Syst.(BTAS)*: 1–6.

[8] Antal, M., Szabo, L.,Z. (2015) "Biometric authentication based on touchscreen swipe patterns." *9th International Conference Interdisciplinarity in Engineering*: 862-869.

[9] Buriro, A., Crispo, B., Conti, M. (2018) "A bimodal behavioral biometric-based user authentication scheme for smartphones." *Journal of Information Security and Applications:* 89-103.

[10] Li, Y., Hu, H., Zhao, G. (2018) “Sensor-Based Continuous Authentication Using Cost-Effective Kernel Ridge Regression.” *Digital Object Identifier 10.1109/ACCESS.2018.2841347:*

[11] Li, G., Bours, P. (2018) “Studying WiFi and Accelerometer Data Based Authentication Method on Mobile Phones.” *ICBEA '18, May 16-18, 2018, Amsterdam, Netherlands. https://doi.org/10.1145/3230820.3230824:*

[12] Yuksel, A., S., Senel, F., A., Cankaya, I., A. (2018) “Classification of Soft Keyboard Typing Behaviors Using Mobile Device Sensors with Machine Learning.” *Arabian Journal for Science and Engineering https://doi.org/10.1007/s13369-018-03703-8:*

[13] Two-Class Logistic Regression **URL:** https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-logistic-regression

[14] James G.,Witten D., Hastie T., Tibshirani R. “An Introduction to Statistical Learning” 219-227 (2013)

[15] James G.,Witten D., Hastie T., Tibshirani R. “An Introduction to Statistical Learning” 215-219 (2013)

[16] Broyden Fletcher Goldfarb Shanno algorithm URL: https://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm

[17] Xiao, Y., Wei, Z., Wang, Z. (2008) “A limited memory BFGS-type method for large-scale unconstrained optimization.” *Computers and Mathematics with Applications 56:* 1001-1009

[18] Two-Class Support Vector Machine **URL:** https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-support-vector-machine

[19] James G.,Witten D., Hastie T., Tibshirani R. “An Introduction to Statistical Learning” 337-268 (2013)

[20] Alpaydın, E. “Introduction to Machine Learning” 218-225 (2004)

[21] Two-Class Locally Deep Support Vector Machine **URL:** https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-locally-deep-support-vector-machine

[22] Two-Class Boosted Decision Tree **URL:** https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-boosted-decision-tree

[23] Gradient boosting. **URL:** https://en.wikipedia.org/wiki/Gradient_boosting

[24] Bagging https://en.wikipedia.org/wiki/Bootstrap_aggregating

[25] Two-Class Decision Forest **URL:** https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-forest

[26] HMOG data **URL:** http://www.cs.wm.edu/~qyang/hmog.html

[27] Compute absolute time stamp **URL**: https://www.timestampconvert.com/

# APPENDICES

## Appendix A

```
TABLE_ACCELEROMETER:
        SYSTIME       numeric(20,0) NULL          ,
        EVENTTIME numeric(20,0) NULL          ,
        ACTIVITY_ID              numeric(20,0) NULL          ,
        X             numeric(15,10)              NULL          ,
        Y             numeric(15,10)              NULL          ,
        Z             numeric(15,10)              NULL          ,
        M             numeric(15,10)              NULL          ,
        PHONE_ORIENTATION        numeric(2,0)  NULL

TABLE_ACCELEROMETER_STR:
        STR_VAL varchar(2000)

TABLE_ACTIVITY:
        ID            numeric(20,0) NULL          ,
        SUBJECT_ID numeric(6,0)   NULL          ,
        SESSION_NUMBER           numeric(2,0)  NULL          ,
        START_TIME               numeric(20,0) NULL          ,
        END_TIME      numeric(20,0) NULL          ,
        RELATIVE_START_TIME  numeric(20,0) NULL          ,
        RELATIVE_END_TIME        numeric(20,0) NULL          ,
        GESTURE_SCENARIO         numeric(2,0)  NULL          ,
        TASK_ID       numeric(2,0)  NULL          ,
        CONTENT_ID               numeric(2,0)  NULL          ,
        USER_ID       numeric(6,0)  NULL          ,
        SESSION_ID numeric(2,0)   NULL

TABLE_ACTIVITY_STR:
        STR_VAL varchar(2000)

TABLE_GYROSCROPE:
        SYSTIME       numeric(20,0) NULL          ,
        EVENTTIME numeric(20,0) NULL          ,
        ACTIVITY_ID              numeric(20,0) NULL          ,
        X             numeric(15,10)              NULL          ,
        Y             numeric(15,10)              NULL          ,
        Z             numeric(15,10)              NULL          ,
        M             numeric(15,10)              NULL          ,
        PHONE_ORIENTATION        numeric(2,0)  NULL

TABLE_GYROSCROPE_STR:
        STR_VAL varchar(2000)

TABLE_KEYPRESS:
        SYSTIME       numeric(20,0) NULL          ,
        PRESSTIME  numeric(20,0) NULL          ,
        PRESSTYPE  numeric(2,0)  NULL          ,
        ACTIVITY_ID              numeric(20,0) NULL          ,
        KEY_ID        numeric(4,0)  NULL          ,
        PHONE_ORIENTATION        numeric(2,0)  NULL

TABLE_KEYPRESS_STR:
        STR_VAL varchar(2000
```

TABLE_MAGNETOMETER:

SYSTIME numeric(20,0) NULL ,
EVENTTIME numeric(20,0) NULL ,
ACTIVITY_ID numeric(20,0) NULL ,
X numeric(15,10) NULL ,
Y numeric(15,10) NULL ,
Z numeric(15,10) NULL ,
M numeric(15,10) NULL ,
PHONE_ORIENTATION numeric(2,0) NULL

TABLE_MAGNETOMETER_STR

STR_VAL varchar(2000)

TABLE_ONEFINGERTOUCH

SYSTIME numeric(20,0) NULL ,
PRESSTIME numeric(20,0) NULL ,
ACTIVITY_ID numeric(20,0) NULL ,
TAP_ID numeric(10,0) NULL ,
TAP_TYPE numeric(2,0) NULL ,
ACTION_TYPE numeric(2,0) NULL ,
X numeric(15,10) NULL ,
Y numeric(15,10) NULL ,
PRESSURE numeric(15,10) NULL ,
CONTACT_SIZE numeric(15,10) NULL ,
PHONE_ORIENTATION numeric(2,0) NULL

TABLE_ONEFINGERTOUCH_STR

STR_VAL varchar(2000)

TABLE_PINCH

SYSTIME numeric(20,0) NULL ,
PRESSTIME numeric(20,0) NULL ,
ACTIVITY_ID numeric(20,0) NULL ,
EVENT_TYPE numeric(2,0) NULL ,
PINCH_ID numeric(20,0) NULL ,
TIME_DELTA numeric(20,0) NULL ,
FOCUS_X numeric(15,10) NULL ,
FOCUS_Y numeric(15,10) NULL ,
SPAN numeric(15,10) NULL ,
SPAN_X numeric(15,10) NULL ,
SPAN_Y numeric(15,10) NULL ,
SCALE_FACTOR numeric(15,10) NULL ,
PHONE_ORIENTATION numeric(2,0) NULL

TABLE_PINCH_STR

STR_VAL varchar(2000)

TABLE_SCROLL

SYSTIME numeric(20,0) NULL ,
BEGINTIME numeric(20,0) NULL ,
CURRENTTIME numeric(20,0) NULL ,
ACTIVITY_ID numeric(20,0) NULL ,
SCROLL_ID numeric(20,0) NULL ,
START_ACTION_TYPE numeric(2,0) NULL ,
START_X numeric(15,10) NULL ,
START_Y numeric(15,10) NULL ,
START_PRESSURE numeric(15,10) NULL ,
START_SIZE numeric(15,10) NULL ,
CURRENT_ACTION_TYPEnumeric(2,0) NULL ,
CURRENT_Xnumeric(15,10) NULL ,

```
        CURRENT_Y numeric(15,10)        NULL    ,
        CURRENT_PRESSURE        numeric(15,10)        NULL    ,
        CURRENT_SIZE            numeric(15,10)        NULL    ,
        DISTANCE_X              numeric(15,10)        NULL    ,
        DICTANCE_Y              numeric(15,10)        NULL    ,
        PHONE_ORIENTATION       numeric(2,0)   NULL
TABLE_SCROLL_STR
        (STR_VAL varchar(2000)

TABLE_STROKE
        SYSTIME     numeric(20,0) NULL    ,
        BEGINTIME   numeric(20,0) NULL    ,
        ENDTIME     numeric(20,0) NULL    ,
        ACTIVITY_ID             numeric(20,0) NULL    ,
        SCROLL_ID   numeric(20,0) NULL    ,
        START_ACTION_TYPE       numeric(2,0)   NULL    ,
        START_X     numeric(15,10)        NULL    ,
        START_Y     numeric(15,10)        NULL    ,
        START_PRESSURE          numeric(15,10)        NULL    ,
        START_SIZE numeric(15,10)         NULL    ,
        END_ACTION_TYPE         numeric(2,0)   NULL    ,
        END_X       numeric(15,10)        NULL    ,
        END_Y       numeric(15,10)        NULL    ,
        END_PRESSURE            numeric(15,10)        NULL    ,
        END_SIZE    numeric(15,10)        NULL    ,
        SPEED_X     numeric(15,10)        NULL    ,
        SPEED_Y     numeric(15,10)        NULL    ,
        PHONE_ORIENTATION       numeric(2,0)   NULL

TABLE_STROKE_STR
        STR_VAL varchar(2000)

TABLE_TOUCH
        SYSTIME     numeric(20,0) NULL    ,
        EVENTTIME numeric(20,0) NULL    ,
        ACTIVITY_ID             numeric(20,0) NULL    ,
        POINTER_COUNT           numeric(2,0)   NULL    ,
        POINTER_ID numeric(2,0)   NULL    ,
        ACTION_ID   numeric(2,0)   NULL    ,
        X           numeric(15,10)        NULL    ,
        Y           numeric(15,10)        NULL    ,
        PRESSURE    numeric(15,10)        NULL    ,
        CONTACT_SIZE            numeric(15,10)        NULL    ,
        PHONE_ORIENTATION       numeric(2,0)   NULL

TABLE_TOUCH_STR
        STR_VAL varchar(2000)
```

# Appendix B

```
begin
        declare v_root varchar(200);
        declare v_text varchar(200);
        declare v_file varchar(200);
        declare v_user_count numeric(10);
        declare user_counter numeric(10);
        declare session_counter numeric(2);
        declare v_usr_id numeric(6);

        declare SQL_STR varchar(2000);

        set v_root = '/data/hmog_dataset/public_dataset/';


        CREATE TABLE #user_ids
        (usr_id NUMERIC(6));


        insert into #user_ids (usr_id) values(100669);
        insert into #user_ids (usr_id) values(151985);
        -- ...
        -- insert all user ids
        --- ...
        commit;

        select count() into v_user_count from #user_ids;


        select ROWID(usr1) as IDX,usr_id into #user_ids2 from #user_ids as usr1;

        commit;
        set user_counter = 1;
        user_loop: loop

                if user_counter > v_user_count then leave user_loop
                end if;

                select usr_id into v_usr_id from #user_ids2 where IDX = user_counter;
                set session_counter = 1;
                session_loop: loop

                if session_counter > 24 then leave session_loop
                end if;

                set v_text =
                        v_root || convert(varchar(10),v_usr_id) + '/' || convert(varchar(10),v_usr_id) || '/' ||
                                convert(varchar(10),v_usr_id) ||'_session_' || convert(varchar(2),session_counter) || '/';

                if session_counter = 1 then
                        message v_text type info to client;
                end if;


                begin
                        set v_file = v_text || 'Activity.csv';
                        SET SQL_STR = 'LOAD table TABLE_ACTIVITY '
                        ||'( '
                        ||'             ID   ",",'
                        ||'             SUBJECT_ID   ",",'
                        ||'             SESSION_NUMBER   ",",'
                        ||'             START_TIME   ",",'
                        ||'             END_TIME   ",",'
                        ||'             RELATIVE_START_TIME   ",",'
                        ||'             RELATIVE_END_TIME   ",",'
                        ||'             GESTURE_SCENARIO  ",",'
                        ||'             TASK_ID  ",",'
                        ||'             CONTENT_ID "\\x0a") '
                        ||' FROM '''  || v_file ||''''
                        ||' QUOTES OFF '
                        ||' ESCAPES OFF; ';

                        execute immediate SQL_STR;
                        COMMIT;

                        COMMIT;
                        exception
                          when others then
                                        set v_file = v_text || 'Activity.csv';
                                        SET SQL_STR = 'LOAD table TABLE_ACTIVITY '
                                        ||'( '
                                        ||'             ID   ",",'
                                        ||'             SUBJECT_ID   ",",'
                                        ||'             SESSION_NUMBER   ",",'
                                        ||'             START_TIME   ",",'
                                        ||'             END_TIME   ",",'
                                        ||'             RELATIVE_START_TIME   ",",'
                                        ||'             RELATIVE_END_TIME   ",",'
                                        ||'             GESTURE_SCENARIO  ",",'
                                        ||'             TASK_ID  ",",'
                                        ||'             CONTENT_ID "\\x0d") '
                                        ||' FROM '''  || v_file ||''''
                                        ||' QUOTES OFF '
                                        ||' ESCAPES OFF; ';

                                        execute immediate SQL_STR;
                                        COMMIT;
```

```
end;

begin
        set v_file = v_text || 'Accelerometer.csv';
        SET SQL_STR = 'load table TABLE_ACCELEROMETER '
        ||'( '
        ||'             SYSTIME   '','','
        ||'             EVENTTIME   '','','
        ||'             ACTIVITY_ID   '','','
        ||'             X   '','','
        ||'             Y   '','','
        ||'             Z   '','','
        ||'             PHONE_ORIENTATION  ''\\x0a'') '
        ||' FROM '''  || v_file ||''''
        ||' QUOTES OFF '
        ||' ESCAPES OFF; ';

        execute immediate SQL_STR;
        COMMIT;
        exception
          when others then
                        set v_file = v_text || 'Accelerometer.csv';
                        SET SQL_STR = 'load table TABLE_ACCELEROMETER '
                        ||'( '
                        ||'             SYSTIME   '','','
                        ||'             EVENTTIME   '','','
                        ||'             ACTIVITY_ID   '','','
                        ||'             X   '','','
                        ||'             Y   '','','
                        ||'             Z   '','','
                        ||'             PHONE_ORIENTATION  ''\\x0d'') '
                        ||' FROM '''  || v_file ||''''
                        ||' QUOTES OFF '
                        ||' ESCAPES OFF; ';

                        execute immediate SQL_STR;
                        COMMIT;
end;
begin

        set v_file = v_text || 'Gyroscope.csv';
        SET SQL_STR = 'load table TABLE_GYROSCROPE '
        ||'( '
        ||'             SYSTIME   '','','
        ||'             EVENTTIME   '','','
        ||'             ACTIVITY_ID   '','','
        ||'             X   '','','
        ||'             Y   '','','
        ||'             Z   '','','
        ||'             PHONE_ORIENTATION  ''\\x0a'') '
        ||' FROM '''  || v_file ||''''
        ||' QUOTES OFF '
        ||' ESCAPES OFF; ';

        execute immediate SQL_STR;
        COMMIT;
        exception
          when others then

                        set v_file = v_text || 'Gyroscope.csv';
                        SET SQL_STR = 'load table TABLE_GYROSCROPE '
                        ||'( '
                        ||'             SYSTIME   '','','
                        ||'             EVENTTIME   '','','
                        ||'             ACTIVITY_ID   '','','
                        ||'             X   '','','
                        ||'             Y   '','','
                        ||'             Z   '','','
                        ||'             PHONE_ORIENTATION  ''\\x0d'') '
                        ||' FROM '''  || v_file ||''''
                        ||' QUOTES OFF '
                        ||' ESCAPES OFF; ';

                        execute immediate SQL_STR;
                        COMMIT;
end;
begin
        set v_file = v_text || 'Magnetometer.csv';
        SET SQL_STR = 'load table TABLE_MAGNETOMETER '
        ||'( '
        ||'             SYSTIME   '','','
        ||'             EVENTTIME   '','','
        ||'             ACTIVITY_ID   '','','
        ||'             X   '','','
        ||'             Y   '','','
        ||'             Z   '','','
        ||'             PHONE_ORIENTATION  ''\\x0a'') '
        ||' FROM '''  || v_file ||''''
        ||' QUOTES OFF '
        ||' ESCAPES OFF; ';

        execute immediate SQL_STR;
        COMMIT;
        exception
          when others then
                        set v_file = v_text || 'Magnetometer.csv';
                        SET SQL_STR = 'load table TABLE_MAGNETOMETER '
```

```
                                        ||'( '
                                        ||'             SYSTIME   ",",'
                                        ||'             EVENTTIME   ",",'
                                        ||'             ACTIVITY_ID   ",",'
                                        ||'             X   ",",'
                                        ||'             Y   ",",'
                                        ||'             Z   ",",'
                                        ||'             PHONE_ORIENTATION   "\\x0d") '
                                        ||' FROM '''  || v_file ||''''
                                        ||' QUOTES OFF '
                                        ||' ESCAPES OFF; ';

                                        execute immediate SQL_STR;
                                        COMMIT;
end;


begin

            set v_file = v_text || 'TouchEvent.csv';
            SET SQL_STR = 'load table TABLE_TOUCH '
            ||'( '
            ||'             SYSTIME   ",",'
            ||'             EVENTTIME   ",",'
            ||'             ACTIVITY_ID   ",",'
            ||'             POINTER_COUNT   ",",'
            ||'             POINTER_ID   ",",'
            ||'             ACTION_ID   ",",'
            ||'             X  ",",'
            ||'             Y  ",",'
            ||'             PRESSURE  ",",'
            ||'             CONTACT_SIZE  ",",'
            ||'             PHONE_ORIENTATION   "\\x0a") '
            ||' FROM '''  || v_file ||''''
            ||' QUOTES OFF '
            ||' ESCAPES OFF; ';

            execute immediate SQL_STR;
            COMMIT;
            exception
              when others then
                                        set v_file = v_text || 'TouchEvent.csv';
                                        SET SQL_STR = 'load table TABLE_TOUCH '
                                        ||'( '
                                        ||'             SYSTIME   ",",'
                                        ||'             EVENTTIME   ",",'
                                        ||'             ACTIVITY_ID   ",",'
                                        ||'             POINTER_COUNT   ",",'
                                        ||'             POINTER_ID   ",",'
                                        ||'             ACTION_ID   ",",'
                                        ||'             X  ",",'
                                        ||'             Y  ",",'
                                        ||'             PRESSURE  ",",'
                                        ||'             CONTACT_SIZE  ",",'
                                        ||'             PHONE_ORIENTATION   "\\x0d") '
                                        ||' FROM '''  || v_file ||''''
                                        ||' QUOTES OFF '
                                        ||' ESCAPES OFF; ';

                                        execute immediate SQL_STR;
                                        COMMIT;
end;

begin

            set v_file = v_text || 'KeyPressEvent.csv';
            SET SQL_STR = 'load table TABLE_KEYPRESS '
            ||'( '
            ||'             SYSTIME   ",",'
            ||'             PRESSTIME   ",",'
            ||'             ACTIVITY_ID   ",",'
            ||'             PRESSTYPE   ",",'
            ||'             KEY_ID ",",'
            ||'             PHONE_ORIENTATION   "\\x0a") '
            ||' FROM '''  || v_file ||''''
            ||' QUOTES OFF '
            ||' ESCAPES OFF; ';

            execute immediate SQL_STR;
            COMMIT;
            exception
              when others then

                                        set v_file = v_text || 'KeyPressEvent.csv';
                                        SET SQL_STR = 'load table TABLE_KEYPRESS '
                                        ||'( '
                                        ||'             SYSTIME   ",",'
                                        ||'             PRESSTIME   ",",'
                                        ||'             ACTIVITY_ID   ",",'
                                        ||'             PRESSTYPE   ",",'
                                        ||'             KEY_ID ",",'
                                        ||'             PHONE_ORIENTATION   "\\x0d") '
                                        ||' FROM '''  || v_file ||''''
                                        ||' QUOTES OFF '
                                        ||' ESCAPES OFF; ';

                                        execute immediate SQL_STR;
                                        COMMIT;
end;
```

```
begin
        set v_file = v_text || 'OneFingerTouchEvent.csv';
        SET SQL_STR = 'load table TABLE_ONEFINGERTOUCH '
        ||'( '
        ||'         SYSTIME   '','','
        ||'         PRESSTIME   '','','
        ||'         ACTIVITY_ID   '','','
        ||'         TAP_ID   '','','
        ||'         TAP_TYPE   '','','
        ||'         ACTION_TYPE   '','','
        ||'         X  '','','
        ||'         Y  '','','
        ||'         PRESSURE  '','','
        ||'         CONTACT_SIZE  '','','
        ||'         PHONE_ORIENTATION   ''\\x0a'') '
        ||' FROM '''  || v_file ||''''
        ||' QUOTES OFF '
        ||' ESCAPES OFF; ';

        execute immediate SQL_STR;
        COMMIT;
        exception
          when others then
                        set v_file = v_text || 'OneFingerTouchEvent.csv';
                        SET SQL_STR = 'load table TABLE_ONEFINGERTOUCH '
                        ||'( '
                        ||'         SYSTIME   '','','
                        ||'         PRESSTIME   '','','
                        ||'         ACTIVITY_ID   '','','
                        ||'         TAP_ID   '','','
                        ||'         TAP_TYPE   '','','
                        ||'         ACTION_TYPE   '','','
                        ||'         X  '','','
                        ||'         Y  '','','
                        ||'         PRESSURE  '','','
                        ||'         CONTACT_SIZE  '','','
                        ||'         PHONE_ORIENTATION   ''\\x0d'') '
                        ||' FROM '''  || v_file ||''''
                        ||' QUOTES OFF '
                        ||' ESCAPES OFF; ';

                        execute immediate SQL_STR;
                        COMMIT;
end;

begin
        set v_file = v_text || 'PinchEvent.csv';
        SET SQL_STR = 'load table TABLE_PINCH '
        ||'( '
        ||'         SYSTIME   '','','
        ||'         PRESSTIME   '','','
        ||'         ACTIVITY_ID   '','','
        ||'         EVENT_TYPE   '','','
        ||'         PINCH_ID   '','','
        ||'         TIME_DELTA   '','','
        ||'         FOCUS_X  '','','
        ||'         FOCUS_Y  '','','
        ||'         SPAN  '','','
        ||'         SPAN_X  '','','
        ||'         SPAN_Y  '','','
        ||'         SCALE_FACTOR  '','','
        ||'         PHONE_ORIENTATION   ''\\x0a'') '
        ||' FROM '''  || v_file ||''''
        ||' QUOTES OFF '
        ||' ESCAPES OFF; ';

        execute immediate SQL_STR;
        COMMIT;
        exception
          when others then
                        set v_file = v_text || 'PinchEvent.csv';
                        SET SQL_STR = 'load table TABLE_PINCH '
                        ||'( '
                        ||'         SYSTIME   '','','
                        ||'         PRESSTIME   '','','
                        ||'         ACTIVITY_ID   '','','
                        ||'         EVENT_TYPE   '','','
                        ||'         PINCH_ID   '','','
                        ||'         TIME_DELTA   '','','
                        ||'         FOCUS_X  '','','
                        ||'         FOCUS_Y  '','','
                        ||'         SPAN  '','','
                        ||'         SPAN_X  '','','
                        ||'         SPAN_Y  '','','
                        ||'         SCALE_FACTOR  '','','
                        ||'         PHONE_ORIENTATION   ''\\x0d'') '
                        ||' FROM '''  || v_file ||''''
                        ||' QUOTES OFF '
                        ||' ESCAPES OFF; ';

                        execute immediate SQL_STR;
                        COMMIT;
end;


begin
```

```
		set v_file = v_text || 'ScrollEvent.csv';
		SET SQL_STR = 'load table TABLE_SCROLL '
		||'( '
		||'		SYSTIME   '','' '
		||'		BEGINTIME   '','' '
		||'		CURRENTTIME   '','' '
		||'		ACTIVITY_ID   '','' '
		||'		SCROLL_ID   '','' '
		||'		START_ACTION_TYPE   '','' '
		||'		START_X  '','' '
		||'		START_Y  '','' '
		||'		START_PRESSURE  '','' '
		||'		START_SIZE  '','' '
		||'		CURRENT_ACTION_TYPE   '','' '
		||'		CURRENT_X  '','' '
		||'		CURRENT_Y  '','' '
		||'		CURRENT_PRESSURE  '','' '
		||'		CURRENT_SIZE  '','' '
		||'		DISTANCE_X  '','' '
		||'		DICTANCE_Y  '','' '
		||'		PHONE_ORIENTATION   ''\\x0a'') '
		||' FROM '''  || v_file ||''''
		||' QUOTES OFF '
		||' ESCAPES OFF; ';

		execute immediate SQL_STR;
		COMMIT;
		exception
		  when others then
				set v_file = v_text || 'ScrollEvent.csv';
				SET SQL_STR = 'load table TABLE_SCROLL '
				||'( '
				||'		SYSTIME   '','' '
				||'		BEGINTIME   '','' '
				||'		CURRENTTIME   '','' '
				||'		ACTIVITY_ID   '','' '
				||'		SCROLL_ID   '','' '
				||'		START_ACTION_TYPE   '','' '
				||'		START_X  '','' '
				||'		START_Y  '','' '
				||'		START_PRESSURE  '','' '
				||'		START_SIZE  '','' '
				||'		CURRENT_ACTION_TYPE   '','' '
				||'		CURRENT_X  '','' '
				||'		CURRENT_Y  '','' '
				||'		CURRENT_PRESSURE  '','' '
				||'		CURRENT_SIZE  '','' '
				||'		DISTANCE_X  '','' '
				||'		DICTANCE_Y  '','' '
				||'		PHONE_ORIENTATION   ''\\x0d'') '
				||' FROM '''  || v_file ||''''
				||' QUOTES OFF '
				||' ESCAPES OFF; ';

				execute immediate SQL_STR;
				COMMIT;
end;


begin
		set v_file = v_text || 'StrokeEvent.csv';
		SET SQL_STR = 'load table TABLE_STROKE  '
		||'( '
		||'		SYSTIME   '','' '
		||'		BEGINTIME   '','' '
		||'		ENDTIME   '','' '
		||'		ACTIVITY_ID   '','' '
		||'		START_ACTION_TYPE   '','' '
		||'		START_X  '','' '
		||'		START_Y  '','' '
		||'		START_PRESSURE  '','' '
		||'		START_SIZE  '','' '
		||'		END_ACTION_TYPE   '','' '
		||'		END_X  '','' '
		||'		END_Y  '','' '
		||'		END_PRESSURE  '','' '
		||'		END_SIZE  '','' '
		||'		SPEED_X  '','' '
		||'		SPEED_Y  '','' '
		||'		PHONE_ORIENTATION   ''\\x0a'') '
		||' FROM '''  || v_file ||''''
		||' QUOTES OFF '
		||' ESCAPES OFF; ';

		execute immediate SQL_STR;
		COMMIT;
		exception
		  when others then
				set v_file = v_text || 'StrokeEvent.csv';
				SET SQL_STR = 'load table TABLE_STROKE  '
				||'( '
				||'		SYSTIME   '','' '
				||'		BEGINTIME   '','' '
				||'		ENDTIME   '','' '
				||'		ACTIVITY_ID   '','' '
				||'		START_ACTION_TYPE   '','' '
				||'		START_X  '','' '
				||'		START_Y  '','' '
```

```
||'                    START_PRESSURE  ",",'
||'                    START_SIZE  ",",'
||'                    END_ACTION_TYPE   ",",'
||'                    END_X  ",",'
||'                    END_Y  ",",'
||'                    END_PRESSURE  ",",'
||'                    END_SIZE  ",",'
||'                    SPEED_X  ",",'
||'                    SPEED_Y  ",",'
||'                    PHONE_ORIENTATION   "\\x0d") '
||' FROM '''  || v_file ||''''
||' QUOTES OFF '
||' ESCAPES OFF; ';

execute immediate SQL_STR;
COMMIT;
                                        end;



        set session_counter = session_counter+1
      end loop session_loop;
      set user_counter = user_counter+1;
    end loop user_loop;


exception
  when others then
                                        message SQL_STR type info to client;
    set sp_sqlstate = sqlstate;
    set sp_sqlcode = sqlcode;
    rollback work;
    signal sp_exception;
    return 1
end;
```

# BIOGRAPHICAL SKETCH

Nurhak Karakaya was born in Kars 1982. He graduated from Milliyet Anadolu High School at 2001. Same year he started Computer Engineering (B.S) at Bogazici University. He graduated from Bogazici University at 2006. At 2017 he started MEF University on Big Data Analytics (M.S.), his first thesis was about "Carbon Price Forecasting". He graduated M.S degree from MEF University at 2018. Same year, he started his second thesis in Galatasaray University. He is working as a computer engineer in banking sector.