Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 475–484

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Using behavioral biometric sensors of mobile phones for user authentication

Nurhak Karakaya, Gülfem Işıklar Alptekin*, Özlem Durmaz İncel

Galatasaray University, Department of Computer Engineering, İstanbul, 34348, Turkey

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, logistic regression) to predict user authentication. The data used in this experiment (HMOG) are collected from 100 students. We compare the results of the algorithms and for our scenario, we show that decision forest algorithm with normalized data gives best results.

© 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of KES International.

Keywords: Continuous authentication; behavioral biometrics; mobile sensors.

1. Introduction

Statistics from 2016 [1] show that from 2 to 2.5 billion people use smartphones. The 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 doing 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

* Corresponding author. Tel.: +90 212 2274480; fax: +90 212 2595557.
E-mail address: gisiklar@gsu.edu.tr

1877-0509 © 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of KES International.
10.1016/j.procs.2019.09.202

apps on their phone. With combining those two analyses, securing mobile devices is the 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. 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]. In these experiments [2], 100 students, using the same kind of mobile phones, do specific tasks, such that they type massage, they scroll screen, they touch to screen, they press keys, etc. The experiments are done either by walking or by siting. Each student does 24 experiments. After each experiment, 11 files are created. These files keep activity and 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 ($\sqrt{X^2 + Y^2 + Z^2}$). Therefore, touch event, activity information and data from three sensors are associated to support new user authentication.

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. The authors in [2] include user data to train their models, whereas we used both user and non-user data. 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. PCA is used for field reduction. We used only one sensor 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.

After presenting related works in Section 2, in Section 3, we define project methodology together with the project motivation. In Section 4, we represent data cleaning and data preparation process. Section 5 introduces the proposed model and Section 6 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 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 behaviour [5], touch gestures [6][7][8] etc. are examined for the purpose of creating a model to decide whether user is authorized or not.

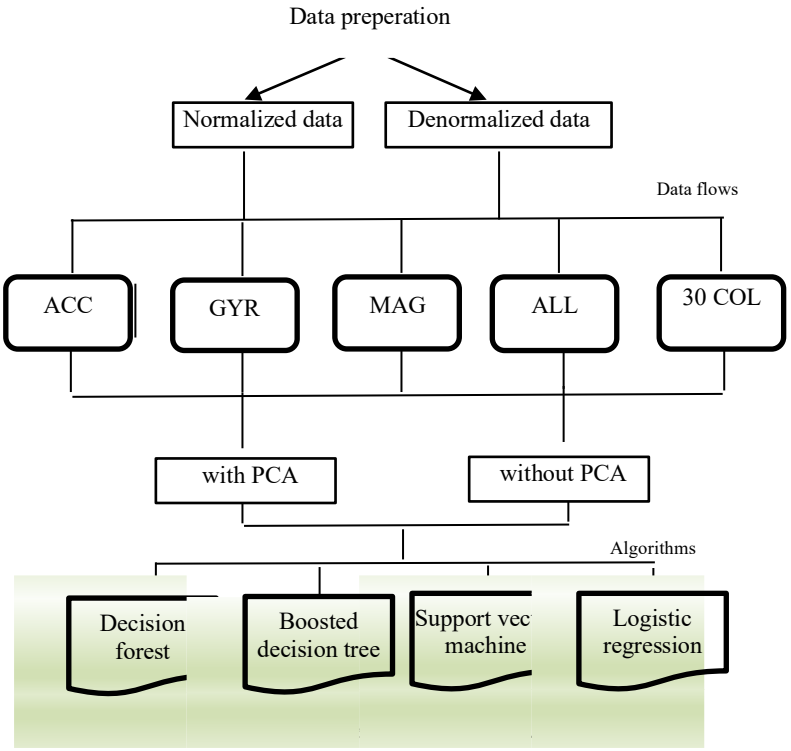
3. Proposed Methodology

In our proposed framework, we first downloaded the HMOG data [2]. For any attender, there are about 24 sessions. In each session, there are 10 files for data and 3 files for questions. These files are: *Accelerometer.csv*, *Activity.csv*, *Gyroscope.csv*, *KeyPressEvent.csv*, *Magnetometer.csv*, *OneFingerTouchEvent.csv*, *PinchEvent.csv*, *ScrollEvent.csv*, *StrokeEvent.csv*, and *TouchEvent.csv*. For our experiments, we use three sensor files (*Accelerometer.csv*, *Gyroscope.csv* and *Magnetometer.csv*), user and activity identification file (*Activity.csv*), and touch event file (*TouchEvent.csv*). 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 press a key, etc. 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. Instead of creating a highly complex data with lots of *null* values in it, we only used touch files for our models.

For our experiments, we use three sensor files (*Accelerometer.csv*, *Gyroscope.csv* and *Magnetometer.csv*), user and activity identification file (*Activity.csv*) and touch event file (*TouchEvent.csv*). We do not use other event files. We prefer to use touch event files, since in an experiment, there are more touch data than any other events.

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. There are two points to consider when downloading files: 1. Some files have carriage return at the end of line. 2. Some files have new line at the end of line. We solved both problems in order not to miss any information. After loading data files to database tables, we created a unique table by combining Activity, Touch Event, Accelerometer, Gyroscope and Magnetometer tables. We will explain all data manipulation events in Section 4 After merging these tables into one table, we randomly chose 20 students and downloaded their data for the proposed models in Microsoft Azure environment. For each user, there are about 100.000 records. We used these 100.000 records as USER label. Moreover, for each USER label, we randomly chose about 500.000 records from other users and labeled them as NO_USER. In total, for our experiment, we had 20 cases, where each case involves about 100.000 USER and 500.000 NO_USER records.

We then downloaded these 20 cases to Microsoft Azure environment, where we created one model for each case. Each model tries to predict its own user. For modelling algorithms, we used decision forest, boosted decision tree, support vector machine and logistic regression. We used 70% of total data for training and 30% for testing. In all models, we used five types of data flow. We aimed to measure sensor performance alone, so we selected one data flow for each sensor, which ends up three data flows. Besides, we added one data flow for all attributes and one data flow for 30 most predictive attributes with target variable. We used normalized and de-normalized data for our experiments and compared their performances. Principal Component Analysis (PCA) is exploited for reducing dimensionality. The overall methodology is illustrated in Figure 1.



4. Data Discovery

HMOG data [1] includes files for 100 different users and a pdf file that explains the structure of files. It takes about 6 GB of disk space. The files are in separate zip files, which needs to be extracted. The corrupted files constitute a minor part of the whole data. The data in files are formatted as coma separated and the end of line for files are not uniquely defined. Some of them ends with carriage return, whereas the others with new line. We created a loop and dynamic SQL to load files into Sybase database. We check for carriage return and new line. Figure 2 shows a part of this load code.

```
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;
```

Fig. 2. A part of the code for loading data into Sybase database.

For one user, there are about 24 sessions with 10 files (three sensor files, one activity file and six event files) to download for each. After loading all these files into the database tables, we checked data quality and data

compliance. For data quality issues, we checked all tables one by one. For each column in a table, we checked the following criteria: *null count*, *not null count*, *ratios of null count ant not null count*, *distinct count*, *max count* and *min count*. Additionally, we determined most frequently encountered values of a column. The uniqueness with respect to time variable with each other column is investigated. We grouped field count with descending order to see where the data is cumulated. Doing so, we controlled all columns in each table and investigated all data quality issues. The *null* problems are solved by checking files again to see the actual data in files, instead of eliminating them. Moreover, in data, we discovered that there are several values, which are beyond the values in data description file. For almost all tables, there were redundant records that we made consistent.

After data quality checks, we passed to the data compliance check. We investigated if the activity files are compliant to sensor files with respect to time and activity numbers. Besides, we analyzed if the sensor tables are also compliant to each other by time and activity id. Finally, we controlled if the activity table is compliant to event tables. We represent an example of compliance test in Figure 3. After completing data quality and data discovery part, we completed to use sensor tables, activity tables and touch event tables in our experiments.

```
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
```

Fig. 3. Compliance tests.

Table 1. Activity table attributes.

ID	numeric(20,0)	Composed as: SubjectID + Session_number + 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, 2,8,14,20: Reading + Walking, 3,9,15,21: Writing + Sitting 6, 12, 18, 24: Map + Walking
CONTENT_ID	numeric(2,0)	1: First sub-task, 2: Second sub-task, 3: Third sub-task

Table 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, 1: Device rotated 90 degrees counter-clockwise, 3: Device rotated 90 degrees clockwise

Table 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, 2: Multi touch
POINTER_ID	numeric(2,0)	0: Single touch, or first pointer in multi touch, 1: Second pointer in multi touch
ACTION_ID	numeric(2,0)	0 or 5: DOWN, 1 or 6: UP, 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, 1: Device rotated 90 degrees counter clockwise, 3: Device rotated 90 degrees clockwise

Activity table has 9 attributes, given in Table 1 in details. Sensor tables have the same structure, therefore as an example we give the accelerometer table in Table 2. Gyroscope and Magnetometer tables have the same structure as Table 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 3). The final table is built using these five tables. First, we merged activity table with touch 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. Then this table is merged with sensor tables to determine *maximum sensor system time*, which is equal to *event system time* – 100 ms, and *minimum sensor system time*, which is equal to *event system time* + 100 ms.

Now for each touch event in Touch 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. We called these three time points as SYSTEMTIME, SEN_SYSTIME_ACC_BEFORE, SEN_SYSTIME_ACC_AFTER in Figure 4.

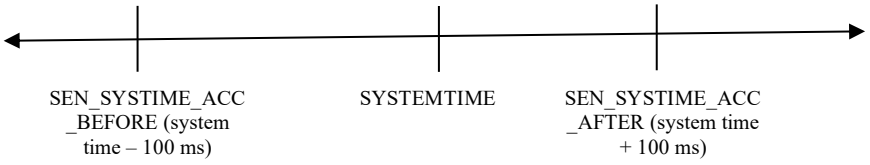


Fig. 4. Time points

Following the calculation of time points, we calculated new variables by using Sensor directions and time variables. Let us explain how to calculate new variables for the Accelerometer and X variable of Accelerometer. (The calculation of other variables and other sensors will follow the same way.)

We first calculate minimum, maximum, average and standard deviation of X between SEN_SYSTIME_ACC_BEFORE and SYSTEMTIME, and minimum, maximum, average and standard deviation of X between SYSTEMTIME and SEN_SYSTIME_ACC_AFTER. Then, we calculate the difference between the values before 100 ms and after 100 ms. The new calculated variables are given in Figure 5.

```
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),
```

Fig. 5. New calculated variables.

In total, we have 4 directions (X, Y, Z, M) * 12 variables (variables in Figure 5) * 3 sensors = 144 new variables. For these new 144 variables, if the value of variable is *null*, we set it to zero. As the following step, we computed binary variables.

Table 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

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

value, we create *n*-1 distinct dummy binary variables. For example; POINTER_COUNT has three values: 1, 2 and 3. 1 for single touch, 2 for multiple touch and 3 has no meaning with respect to data definition, however in some cases, POINTER_COUNT has value of 3. 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 group as in Table 4. After creating dummy variables, we deleted original categorical variables from our list.

5. Data Modeling

For modelling, we randomly chose 20 users from the list. 100.000 records of each user are associated with USER label. For each user, we randomly chose about 500.000 records from other users and labeled these records as NO_USER. Hence, in one data set, there are about 600.000 records; 100.000 for USER and 500.000 for NO_USER. We called the label as "target". Then, we loaded these 20 data sets to Microsoft Azure. Separate models are built for those 20 users' data sets. We used four machine learning algorithms: BD, DF, SVM and LR for predicting. We created separate codes for each algorithm. Moreover, for BD, DF and LR, we used both normalized and denormalized 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). In this part, model and experiment are used interchangeably, because Microsoft Azure calls code parts as Experiment. For each 20 data sets, we run those 7 experiments separately. We used %70 of data as training and %30 for testing. The performances of the models are compared in terms of accuracy.

For each experiment, we first explored identification variables and time variables from our data set. The reason is that some users do all tasks in specific time interval; for that reason, time variables are highly correlated to target variable. Then, we differentiated models by normalizing the data. In each experiment, we had five data flows (Figure 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, because we had already done feature selection. 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 164 variables, so in ALL with PCA we have 54 dimensions.

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. For parameter tuning, we used data of two users (USER and NO_USER). Then, we tuned parameters for two users and used those parameters for every other user. We did not tune parameters for all users, because of its time consumption. The result of tuned parameter values for BD are given as in Table 5, for SVM in Table 6 and for LR in Table 7. We run our models with these parameters for 20 data sets separately.
tables should be numbered with Arabic numerals. Every table should have a caption. Headings should be placed above tables, left justified. Only horizontal lines should be used within a table, to distinguish the column headings from the body of the table, and immediately above and below the table. Tables must be embedded into the text and not supplied separately. Below is an example which the authors may find useful.

Table 5. 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. Tuned parameter values for support vector machine.

Number of iterations	Lambda
98	0.003046

Table 7. 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

6. Results

The 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 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 8, we show top five most correlated attributes with TARGET for randomly chosen two users; whereas Table 9 shows the five attributes that have lowest correlation with TARGET. (The users are the same two in Table 8 and Table 9.)

Table 8. Most correlated attributes with TARGET.

1st user attributes	1st user values	2nd user attributes	2nd user values
Y_ACC_MAX_BEFORE	0.526101	Y_ACC_MEAN_BEFORE	0.420160
Y_ACC_MAX_AFTER	0.526027	Y_ACC_MAX_BEFORE	0.418556
Z_ACC_MEAN_BEFORE	0.522512	Y_ACC_MIN_BEFORE	0.417846
Y_ACC_MEAN_AFTER	0.517672	Y_ACC_MEAN_AFTER	0.408958
Y_ACC_MEAN_BEFORE	0.517449	Y_ACC_MIN_AFTER	0.408290

From our correlation test, we saw that all of the attributes in our data set have correlation with TARGET. Therefore, using all of the attributes is required as they all contribute to prediction. In our experiment, ALL data flow always get better results than any other data flows in each model. In Table 9, we show 5 attributes that have lowest correlations with TARGET, when the users are same as in Table 8.

Table 9. Least correlated attributes with TARGET.

1st user attributes	1st user values	2nd user attributes	2nd user values
M_MAG_MAX_DIFF	0.027014	Y_ACC_MAX_DIFF	0.018830
M_ACC_MIN_AFTER	0.026047	M_GYR_MEAN_DIFF	0.018744
M_ACC_MEAN_AFTER	0.025893	M_GYR_MAX_DIFF	0.017962
Z_GYR_MIN_AFTER	0.025417	M_GYR_MIN_AFTER	0.017787
M_MAG_STDV_DIFF	0.025038	Y_GYR_STDV_AFTER	0.017341

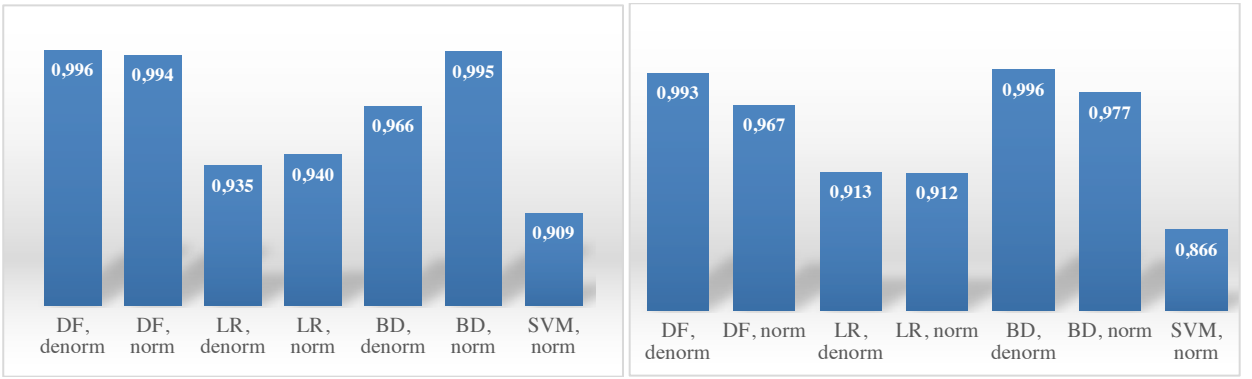


Fig. 5. (a) Accuracy values for 'ALL' data flow; (b) Accuracy values for 'ALL with PCA' data flow.

Since the most correlated attributes changes from user to user, the 30COLS models usually use different attributes. The fact that decreasing the number of attribute decreases model performances using PCA, is also observed in our experiment. In our case, we chose the dimension count as 1/3 of the attribute count. In all models using PCA, model performances are lower compared to models without PCA. In our paper, when using normalized data, we normalized most of the attributes. Since the skewness of some attributes are high, normalizing data gives better results compared to denormalized data. During the experiments, we explored that in all cases, DF gives higher accuracy. Following DF, BD has the second rank among accuracy comparisons. The accuracy values of each algorithm for each data flow are presented in Figure 5-Figure 9. Normalized data gives better results than

denormalized data. Since the skewness of some attributes are high normalizing data gives better results than denormalized data. Results show that, in all cases DF gives higher accuracy followed by BD and LR. In terms of sensor results, we saw that manometer gives best results.

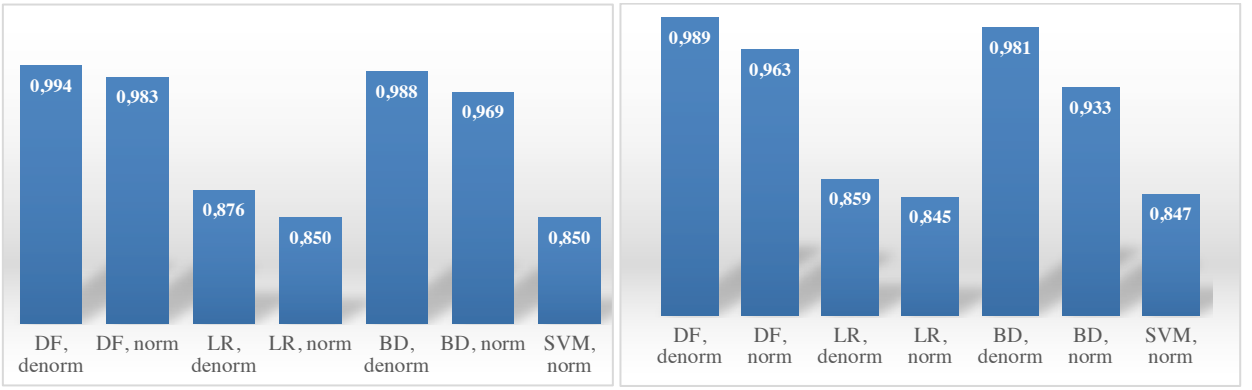


Fig. 6. (a) Accuracy values for 'MAG' data flow; (b) Accuracy values for 'MAG with PCA' data flow.

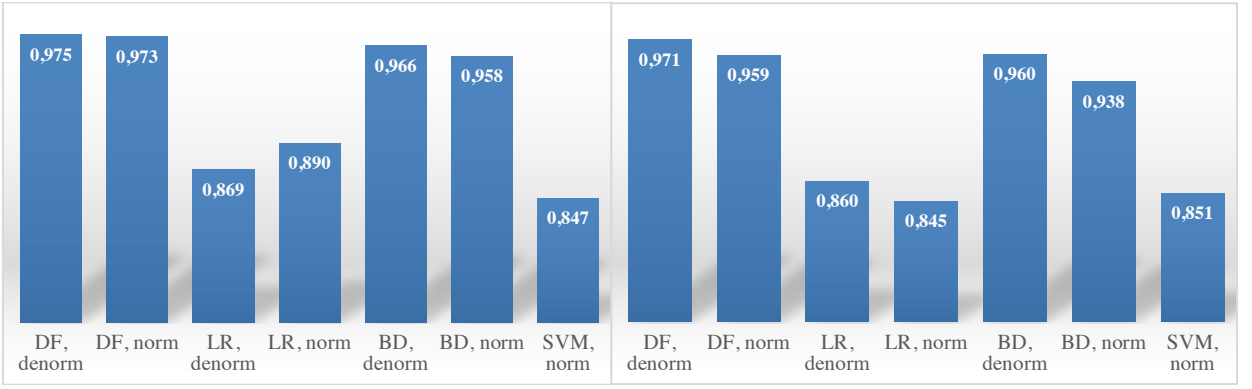


Fig. 7. (a) Accuracy values for 'ACC' data flow; (b) Accuracy values for 'ACC with PCA' data flow.

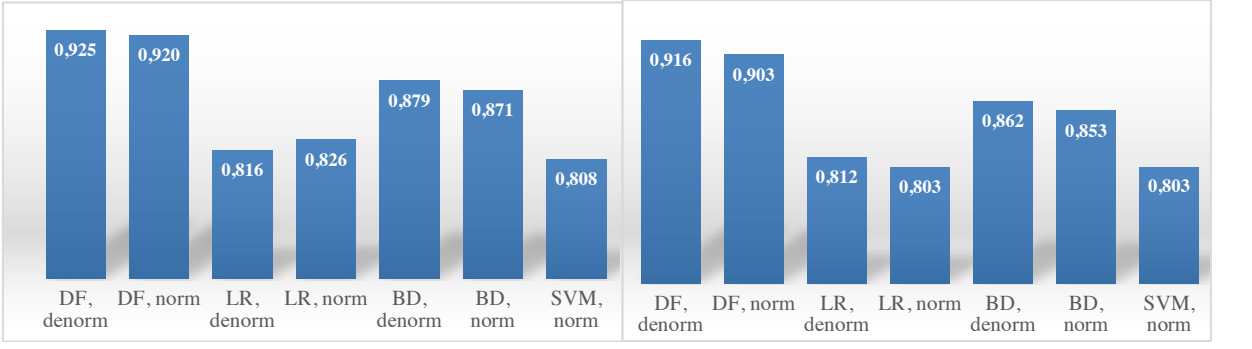


Fig. 8. (a) Accuracy values for 'GYR' data flow; (b) Accuracy values for 'GYR with PCA' data flow.

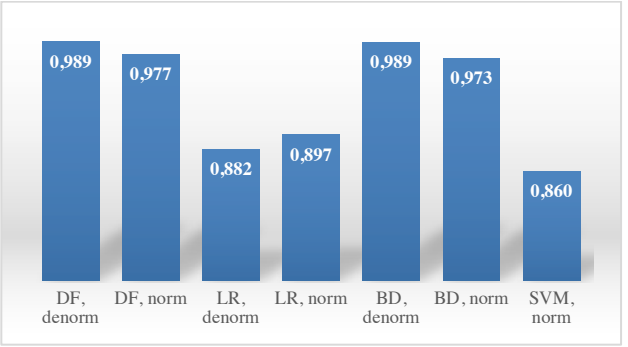


Fig. 9. Accuracy values for '30COLS' data flow.

7. Conclusion

In this paper, the data set of HMOG, which is a set of behavioral 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 run with different data flows are presented. The results demonstrate the efficiency of the usage of sensors for continuous user authentication.

Acknowledgements

This research has been financially supported by the Galatasaray University Research Fund, project number: 19.401.004.

References

[1] Portal, T.S. (2016) "Number of smartphone users world-wide from 2014 to 2020", online: 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] Zebik Korytkowski, M., Angryk, R. Scherer, R. (2017) "Convolutional Neural Networks for Time Series Classification", *16th International Conference on Artificial Intelligence and Soft Computing (ICAISC '17),* in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J. Zurada (Eds.), *Artificial Intelligence and Soft Computing*, Series: *Lecture Notes in Computer Science (LNCS)*, *Vol. 10246.*, Springer-Verlag, ISBN: 978-3-319-07172-5, pp. 635-642.