

# **Users Recognition from Accelerometer Data of Mobile Devices**

Xinran Liu, Xinxin Feng and Zhijian Huang

## **Table of Contents**

1. Project Scope
2. Data Description
3. Methods
  - I. Data Transformation
  - II. Feature Extraction
  - III. Training Methods
    - i. Plan A: Clustering of devices and training within each cluster
    - ii. Plan B: Binary classifier
4. Results
  - I. Plan A: Clustering of devices and training within each cluster
    - i. Clustering of devices
    - ii. Training within each cluster
  - II. Plan B: Binary classifier**
    - i. Training with transformed data**
    - ii. Training with extracted sequence features**
  - III. Prediction of Test Data**
5. Conclusion

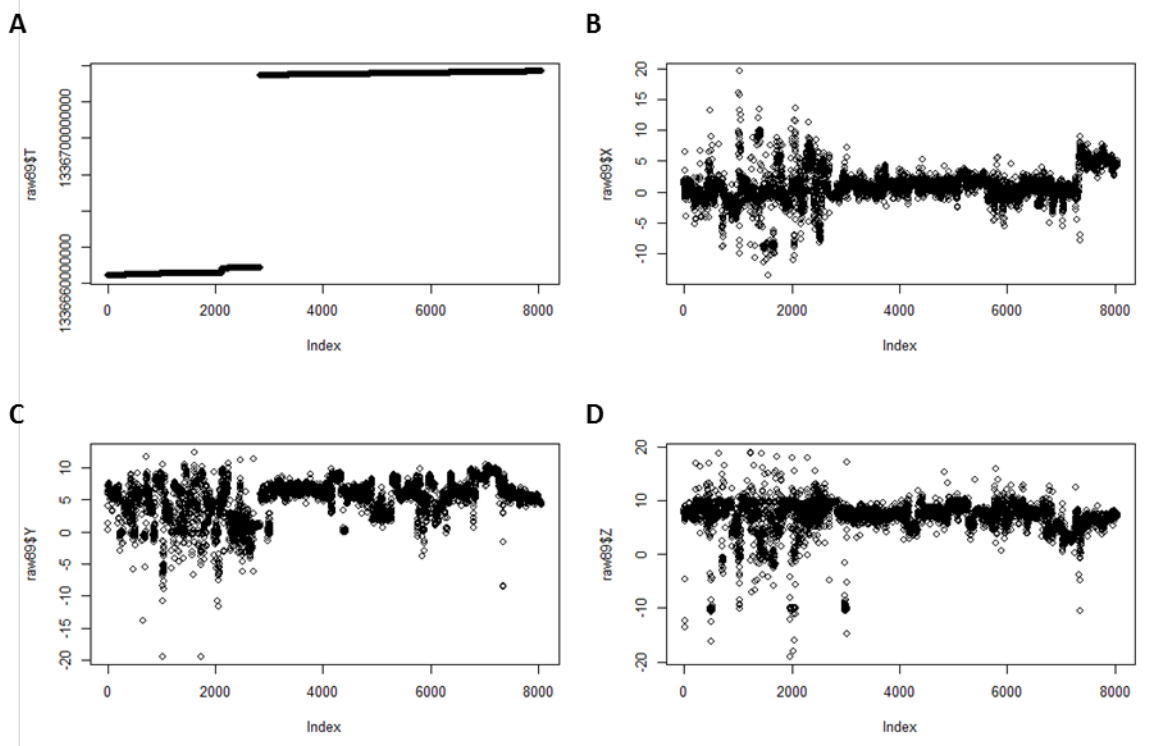
## Project Scope

The scope of this project is to recognize users of mobile accelerometers, from the accelerometer data (movement patterns) recorded for 387 different individuals (devices) over a period of several months.

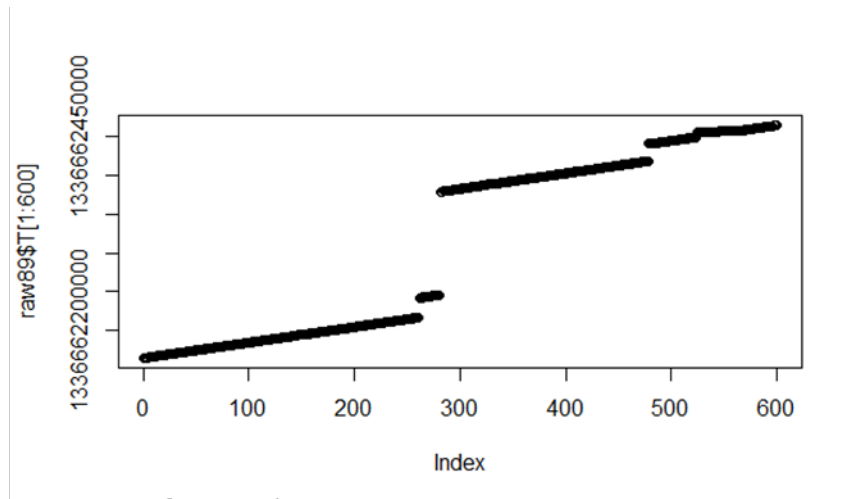
## Data Description

There are two sets of data in this project: training data and test data. In the training data (30 million observations), the accelerometer data from 387 different individuals (devices) over a period of several months are recorded. There are five variables in the training data: T (unit time (ms since 1/1/1970)), X (acceleration measured in g on X co-ordinate), Y (acceleration measured in g on Y co-ordinate), Z (acceleration measured in g on Z co-ordinate), DeviceId (unique Id of the device that generated the samples).

Data were recorded every 0.2 s, with each data point defined as a 'timestamp'. On average for each DeviceId, there are  $30\text{ million}/387 = 77519$  timestamps. As an example, the total number of timestamps (variable T) of DeviceId=89 are shown in **Figure 1A** as a function of index number. As showed in **Figure 1A**, the data of DeviceId=89 were recorded at a roughly continuous fashion of the two time periods (index 1 to ~3000 and index ~3000 to ~8000). For each period, the line is not entirely linear due to the existence of rest period, as shown in **Figure 2**, which is a zoom-in of the first 600



**Figure 1.** Training data of DeviceId 89. (A) Variable T v.s. index. (B) Variable X v.s. index. (C) Variable Y v.s. index. (D) Variable Z v.s. index.



**Figure 2.** First 600 timestamps of DeviceId 89.

observations of **Figure 1A**. The four steps seen in **Figure 2** are due to the resting periods, which are defined as periods of non-movement exceeding 10 seconds.

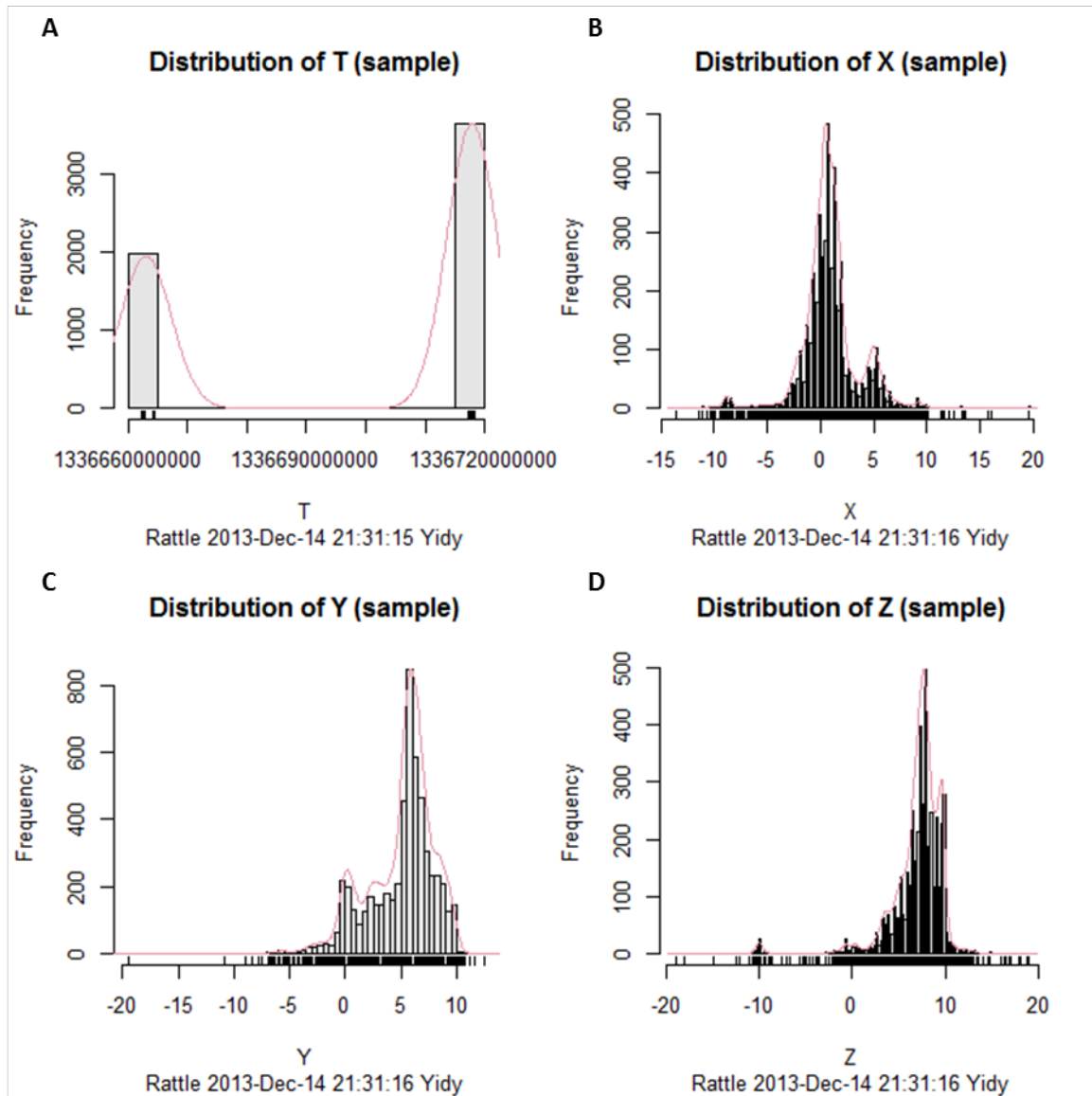
The X, Y and Z variables as a function of index number are shown in **Figures 1B to D**. For all three variables, the absolute values and variations in the first time period (index 1 to ~3000) are larger than those of the second time period (index ~3000 to ~8000), which suggests very different movement pattern between the two time periods of the same user. **Figure 3** shows the histograms of all the variables for DeviceId 89. The two time periods of this DeviceId are separated by  $(1336720000000 - 1336660000000) \times 10^{-6} = 60$ s. Acceleration along X axis is almost normally distributed with a mean of 0, whereas acceleration along Y and Z axes are mostly positive. The ranges of three coordinates X, Y and Z are about the same.

**Table 1** shows the correlations between the four variables of DeviceId 89, which suggests that there is a moderate correlation between T and Y, but there's no obvious correlation among the three acceleration directions X, Y and Z.

**Table 1.** Correlations between the four variables of DeviceId 89.

	Z	X	Y	T
Z	1.000	0.037	-0.195	-0.040
X	0.037	1.000	-0.116	0.216
Y	-0.195	-0.116	1.000	0.467
T	-0.040	0.216	0.467	1.000

In the test set, there are also five variables: T, X, Y, Z (which are the same as those in the training set), and SequenceId (unique sequence number assigned to each question). A sequence is defined as a collection of 300 timestamps, which is equivalent to  $0.2 \text{ s/timestamp} \times 300 \text{ timestamps} = 1 \text{ min}$ . There are 90,000 sequences in the test dataset, and we are going to predict the DeviceId of each sequence, based on the training data.



**Figure 3.** Histograms of T, X, Y and Z variables in the training data of DeviceId 89. (A) . Histograms of T. (B) . Histograms of X. (C) . Histograms of Y. (D) . Histograms of Z.

To evaluate our prediction results, we are asked to answer if a certain SequenceId corresponds to the DeviceId that is proposed in the questions.csv file (QuizDevice).

## Methods


Since the test dataset is in the form of many sequences (sessions of 300 timestamps), we need to divide the training data (**Figure 1A**) into sequences each with 300 timestamps. To do that, we randomly sample 200 or 1000 sequences with length of 300 timestamps with the *sample* function in R.

### I. Data Transformation

After the above sampling, we now have each sequence being a 300\*5 matrix, since we have 300 timestamps and 5 variables for each timestamps. Now we need to transform this 300\*5 matrix into one observation. In order to do that, we will create a new data matrix with 901 variables (DeviceId, V1, V2, V3, ..., V900), with V1 to V3 corresponding to the X, Y and Z coordinate of 1<sup>st</sup> timestamp in the sequence matrix, V4 to V6 corresponding to the X, Y and Z coordinate of 2<sup>nd</sup> timestamp in the sequence matrix, V7 to V9 corresponding to the X, Y and Z coordinate of 3<sup>rd</sup> timestamp in the sequence matrix, ..., V898 to V900 corresponding to the X, Y and Z coordinate of 300<sup>th</sup> timestamp in the sequence matrix, as illustrated in **Figure 4**. In this transformation, the information contained in T variable is retained by sequentially assembling X, Y and Z of each timestamp into the new data matrix; however, since the absolute value of T is not taken into account, the 'step' information (**Figure 2**) is discarded in the new data matrix.

Transformation of test data will be done similarly for each SequenceId.

T, ms	X	Y	Z	DeviceId
1.34E+12	3.41E-01	8.31E+00	4.14E+00	7
1.34E+12	3.81E-01	8.39E+00	4.25E+00	7
1.34E+12	2.72E-01	8.47E+00	4.02E+00	7
...				
1.34E+12	1.50E-01	8.43E+00	4.29E+00	7



DeviceId	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V898	V899	V900
7	3.41E-01	8.31E+00	4.14E+00	3.81E-01	8.39E+00	4.25E+00	2.72E-01	8.47E+00	4.02E+00	...	1.50E-01	8.43E+00	4.29E+00

**Figure 4.** Transformation of training data.

### II. Feature Extraction

Besides direct transformation of the sequence data, another way to acquire information about one sequence is to extract data features associated with the descriptive statistics and correlation among the variables for one sequence. **Table 2** lists the possible features that we can extract from one sequence, which summarizes 70 features that we plan to use in our analysis.

**Table 2.** Possible features that can be extracted from one sequence.

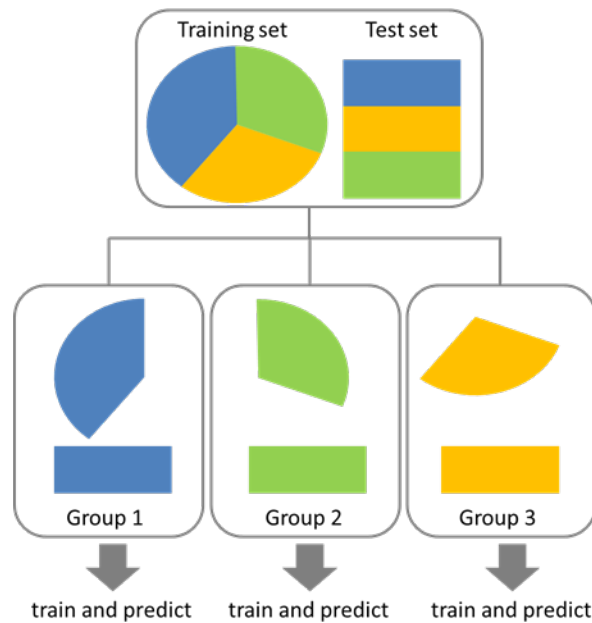
Category	Features	Numbers
Descriptive Statistics of X, Y and Z	mean, variance, minimum, 1 <sup>st</sup> quantile, median, 3 <sup>rd</sup> quantile, maximum, skewness, kurtosis, range	10 features/coordinate*3 coordinates = 30 features
Descriptive Statistics of T	Range	1
Correlation	correlation between two coordinates	3 pairs of correlation (XY, YZ and XZ)
	correlation between one coordinates and T	3 pairs of correlation (XT, YT and TZ)
	correlation between two coordinates lagged by p seconds	p=1, 2, 5, 10s: X(X+p), Y(Y+p), Z(Z+p), X(Y+p), X(Z+p), Y(Z+p) p=15, 20, 30s: X(X+p), Y(Y+p), Z(Z+p) 3+3+4*6+3*3=39 features

### III. Training Methods

#### i. Plan A: Clustering of devices and training within each cluster

##### a) Clustering of devices

To reduce the cost of computation and model building, we will attempt to divide all 387 devices into three groups, as shown in **Figure 5**. To do that, we will first mix all sequences generated from the training dataset and test dataset. Then use various clustering methods to divide all sequences into three groups, and the methods that provide the smallest misclassification rate for training sequence

**Figure 5.** Proposed clustering and training steps in Plan A.

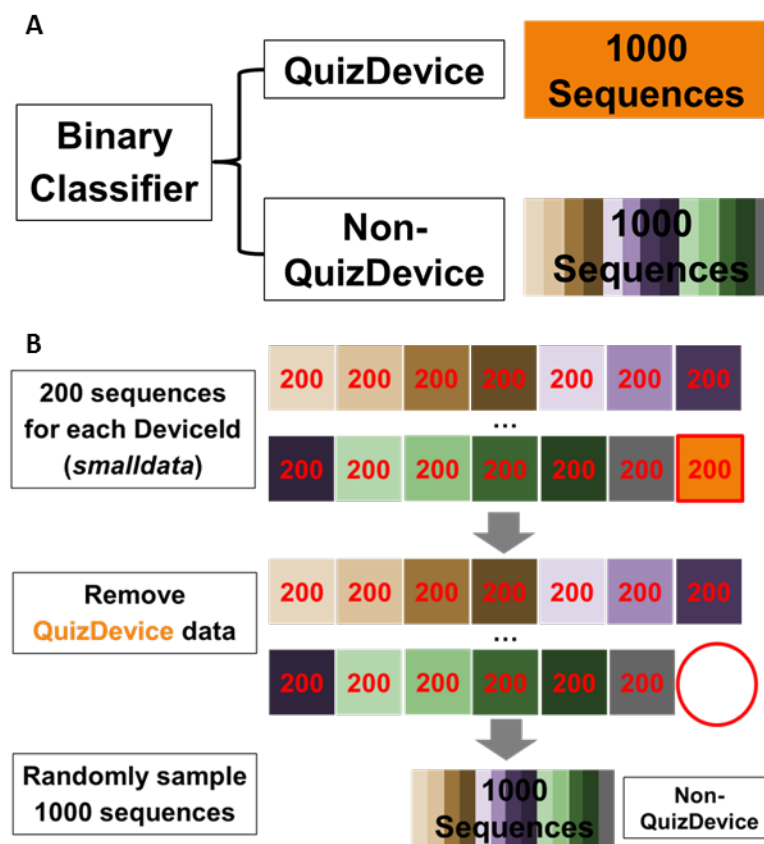
will be selected. For each device, misclassification rate of training sequences is defined as the ratio of the number of sequences that falls outside the designated group and the total number of sequences, in which the designated group is the one containing the largest number of sequences from that device.

b) Data Training within cluster

If the above clustering methods afford a reasonable misclassification rate of the training set (say  $<0.05$ ), we will proceed to training the three groups of data individually with various statistical methods, such as discriminant analysis, SVM, tree method, random forest, etc. Performances of different training methods will be evaluated on the basis of training errors and other criteria.

## ii. Plan B: Binary classifier

The purpose of this binary classifier is to take advantage of the QuizDevice that is provided by the question.csv file. As shown in **Figure 6A**, in this approach, we use the QuizDevice and a collection of all the remaining 386 DeviceId (non-QuizDevice) to generate a model with binary outcomes (QuizDevice or non-QuizDevice). In order to obtain the 1000 sequences from non-QuizDevice, we first generated a smaller subset of training sequences from the training dataset that is composed of 200 sequences of each of the 387 DeviceId (that is,  $200 \text{ sequences/DeviceId} * 387 \text{ DeviceId} = 77400$



**Figure 6.** Proposed binary classification method. (A) Binary classifier using 1000 sequences from QuizDevice and 1000 sequences from Non-QuizDevice; (B) Method used to generate the 1000 sequences from Non-QuizDevice.

sequences, which is referred to as *smallsample*), as shown in **Figure 6B**. Next, when predicting a sequence with a QuizDevice of 89, for example, we use the 1000 sequences from QuizDevice 89 and 1000 sequences from non-QuizDevice 89 to build the training models. The 1000 sequences for non-QuizDevice 89 are randomly sampled from the above-mentioned *smallsample* excluding all the sequences of DeviceId 89 (**Figure 6B**). Prediction of this QuizDevice sequences (QuizDevice 89 or non-QuizDevice 89) is done based on our generated models with smaller error rates, which are used to vote for the final prediction for this sequence. When building the models, both data transformation method and feature extraction method can be used.



## Results

### I. Plan A: Clustering of devices and training within each cluster

#### i. Clustering of devices

We first used training sequences of DeviceId 7, 8 and 9 to test the misclassification rates based on different clustering methods, and found that the misclassification rates are very high with both kmeans and Ward hierarchical clustering methods, as shown in **Table 3**. We next tried to group 12 DeviceIds into 3 clusters; however, the misclassification rates are still rather high, as shown in **Table 4**. According to these results, we might conclude that the clustering method is not suitable for this project, and we should turn to other statistical methods.

**Table 3.** Summary of cluster methods for DeviceId 7, 8 and 9.

kmeans	DeviceId 7	DeviceId 8	DeviceId 9
Group 1	588	159	181
Group 2	134	625	525
Group 3	78	16	94
misclassification rate	0.27	0.22	0.34
Ward Hierarchical Clustering	DeviceId 7	DeviceId 8	DeviceId 9
Group 1	452	56	89
Group 2	284	727	623
Group 3	64	17	88
misclassification rate	0.44	0.09	0.22

**Table 4.** Summary of cluster methods for 12 DeviceIds.

kmeans	Devi celd 7	Devi celd 8	Devi celd 9	Devi celd 12	Devi celd 23	Devi celd 25	Devi celd 26	Devi celd 27	Devi celd 33	Devi celd 37	Devi celd 39	Devi celd 45
Group 1	548	738	654	490	740	722	176	730	275	861	527	370
Group 2	221	35	117	439	160	124	774	162	150	99	294	451
Group 3	31	27	29	71	100	154	50	108	575	40	179	179
misclassification rate	0.32	0.08	0.18	0.51	0.26	0.28	0.23	0.27	0.43	0.14	0.47	0.55
Ward Hierarchical Clustering	Devi celd 7	Devi celd 8	Devi celd 9	Devi celd 12	Devi celd 23	Devi celd 25	Devi celd 26	Devi celd 27	Devi celd 33	Devi celd 37	Devi celd 39	Devi celd 45
Group 1	687	756	687	577	784	805	252	806	406	890	598	483

Group 2	83	25	100	414	150	75	716	138	111	78	289	408
Group 3	30	19	13	9	66	120	32	56	483	32	113	109
misclassification rate	0.14	0.06	0.14	0.42	0.22	0.20	0.28	0.19	0.52	0.11	0.40	0.52

## ii. Training within each cluster

We next tried to use tree and RandomForest methods to check the test error rates of models with different number of categories. We randomly partitioned the 1000 sequences from each DeviceId into two datasets, 80% for training dataset and the 20% remaining for test dataset. For both methods, we tried two, three and five mixture DeviceIds, and various parameters, and found that the test rates almost increase linearly along the number of DeviceIds, as shown in **Table 5**. Therefore, we can speculate that if using more than 10 DeviceIds (categories) in the training model, the test error rate will be unacceptably large.

**Table 5.** Test errors for tree and RandomForest methods with different number of categories.

Methods	Parameters	DeviceId used	testing error
Tree	mindev=0.0005	7, 8	0.19
	mindev=0.005	7, 8, 9	0.36
	mindev=0.0005	7, 8, 9, 12, 23	0.42
Random Forest	ntree=700,sample=1000	7, 8	0.03
	ntree=50,sample=500	7, 8	0.12
	ntree=500,sample=500	7, 8	0.12
	ntree=400,sample=1000	7, 8, 9	0.2
	ntree=700,sample=1000	7, 8, 9, 12, 23	0.22

In summary, for both sequences clustering and training with multiple categories, the error rates are very high, which in turn suggests that Plan A is not feasible with this data set. Therefore, we next turned to an alternatively promising method, in which only two categories with binary outcome are considered in each training model (Plan B). This approach has the advantage of taking into account the given QuizDevice.

## II. Plan B: Binary Classifier

### i. Training with transformed data

#### a. Test errors of different methods

As discussed above, we are now focused on building models with two outcomes: the QuizDevice and non-QuizDevice. To predict the DeviceId of a certain sequence, we first assembled 1000 sequences from the QuizDevice and another 1000 sequences from *smallsample* (non-QuizDevice

sequences). Again we randomly partitioned the 2000 sequences (samples) into training dataset (80%) and test dataset (20%). The test errors and parameters for these four devices (7, 8, 12 and 23) with various methods are shown in **Table 6**.

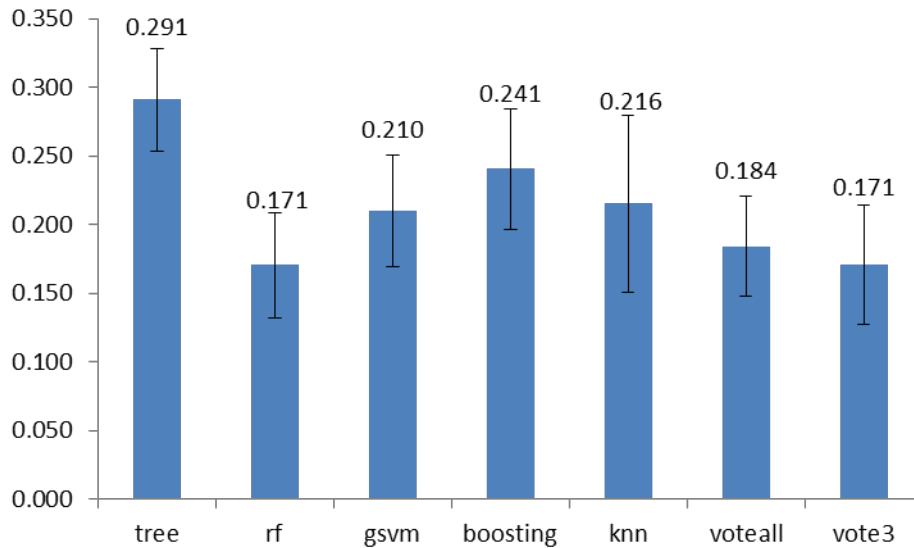
**Table 6.** Test errors for four devices with multiple methods.

Methods	Type	Test Error				Parameters
		Device 7	Device 8	Device 12	Device 23	
kNN		0.10	0.21	0.09	0.13	K=1/3/5/7/.../151
randomforest		0.15	0.13	0.10	0.17	Mtry=30 ntree=50 n.tree: 1000
gbm		0.29	0.19	0.16	0.39	shrinkage:0.05 bag.fraction: 0.5
Discriminant Analysis	LDA	0.44	0.40	0.36	0.41	
Discriminant Analysis	RDA	0.50	0.50	0.50	0.50	
Discriminant Analysis	NB	0.36	0.32	0.30	0.51	
Tree Model		0.26	0.20	0.22	0.37	Mindev: 0.005 minsize: 2
Support Vector Machine	quadratic kernel		0.41	0.42		
Support Vector Machine	linear kernel		0.43	0.39		
Support Vector Machine	Gaussian kernel	0.17	0.18	0.11	0.26	Automatical

From **Table 6**, we can see that kNN, tree, randomForest, gbm and SVM with Gaussian kernel are much reliable, and provide smaller test errors. In particular, both kNN and RandomForest generate much promising and consistent results. Thus, these statistical methods will be chosen for our final runs. We will vote for the final predictions based on the smallest error rate among these methods.

b. Test errors of the 20 DeviceIds

**Figure 7** summarizes average test error results for 20 DeviceIds. RandomForest and vote3 (voting based on RandomForest, SVM with Gaussian kernel and KNN) have the lowest error rate, 17.1%.



**Figure 7.** Average prediction results for 20 DeviceId. (rf: randomForest; gsvm: SVM with Gaussian kernel; boosting: gbm; voteall: voting among tree, rf, gsvm, boosting and knn; vote3: voting among rf, gsvm,knn)

Given the relatively high error rate and high computation cost using the above transformed training data, we decided to build our models based on the extracted features shown in **Table 2**.

## ii. Training with extracted sequence features

DeviceId 89 was used as an example to illustrate the features of sequences and the training methods. **Figure 8** shows the histograms of selected feature variables. We then used various methods to construct models with binary classifiers on 80% of the training data, and test these models with the other 20% of data. Test errors, model parameters and required time are listed in **Table 7**.

As shown in **Table 7**, for DeviceId 89, both RandomForest and boosting perform best with test error of ~2%. Tree method is also competitive with error rate of ~5%. Lasso and KNN have comparable performance with ~8% error. Among these methods, SVM with Gaussian kernel performs worst with more than 10% error rate. For each method, the optimal parameters were determined and then applied for all the DeviceIDs. For example, **Figure 9** shows that 872 iterations are required for boosting to obtain optimal results, and for KNN, it is better using five nearest neighbors to implement a majority vote.

Both tree method and random forest find important variables to design a good prediction model. **Figure 10** shows important variables selected by tree method (**Figure 10A**) and RandomForest method (**Figures 10B and C**). MaxY, MeanZ, quan75\_X and Timrange were found to be important variables. Comparison of the four important variables between DeviceId 89 and DeviceId 7 based on their cumulative probability (**Figure 11**) shows that the distribution patterns of these four variables are very different between the two DeviceIDs.

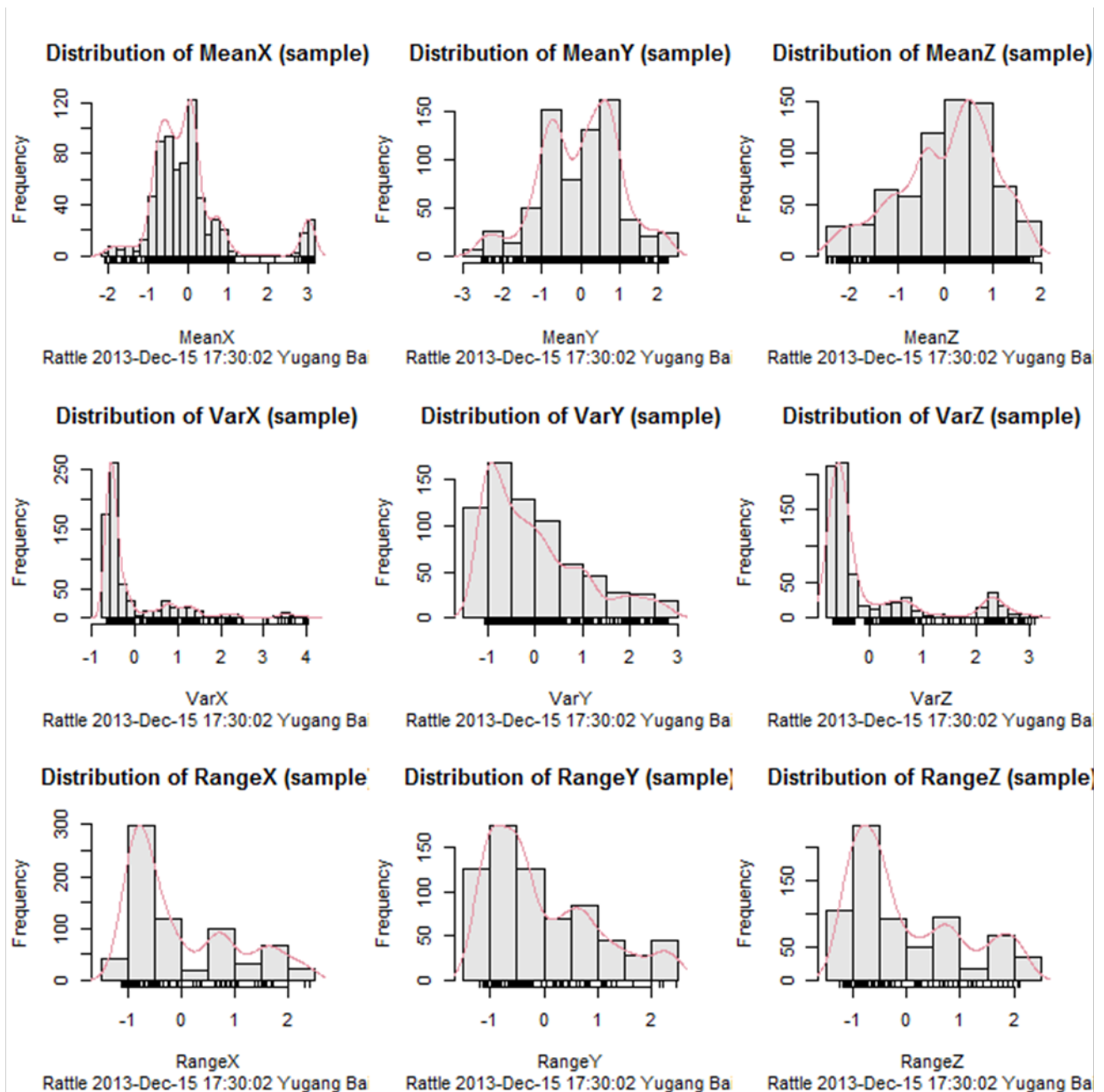
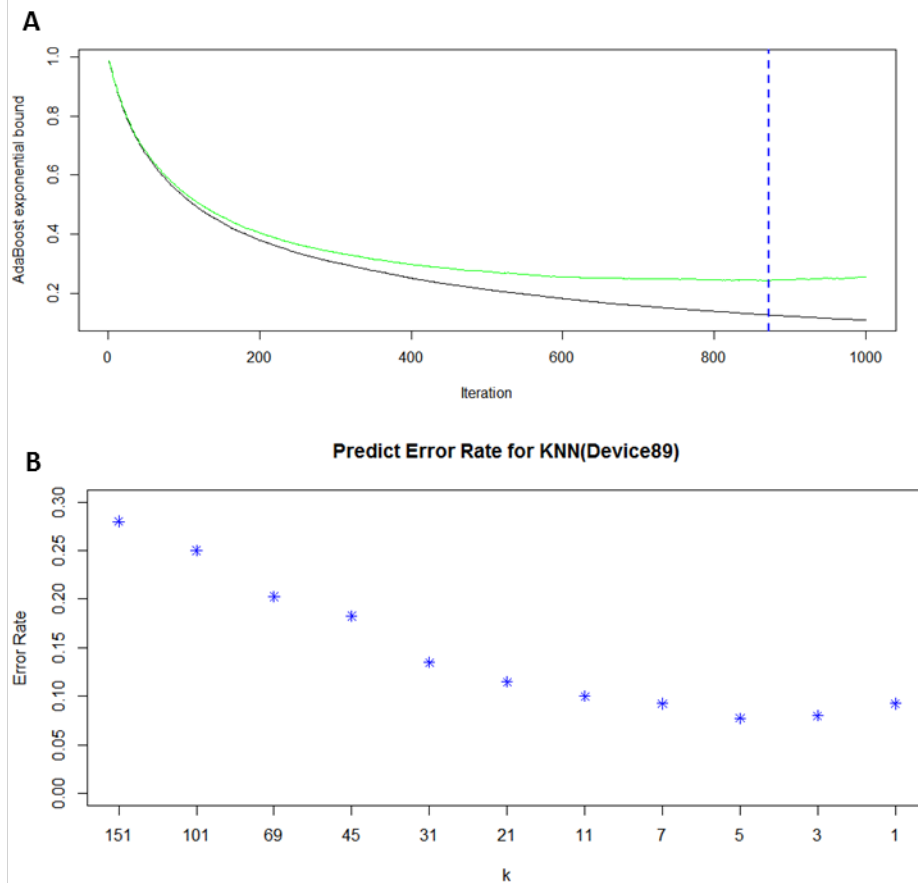
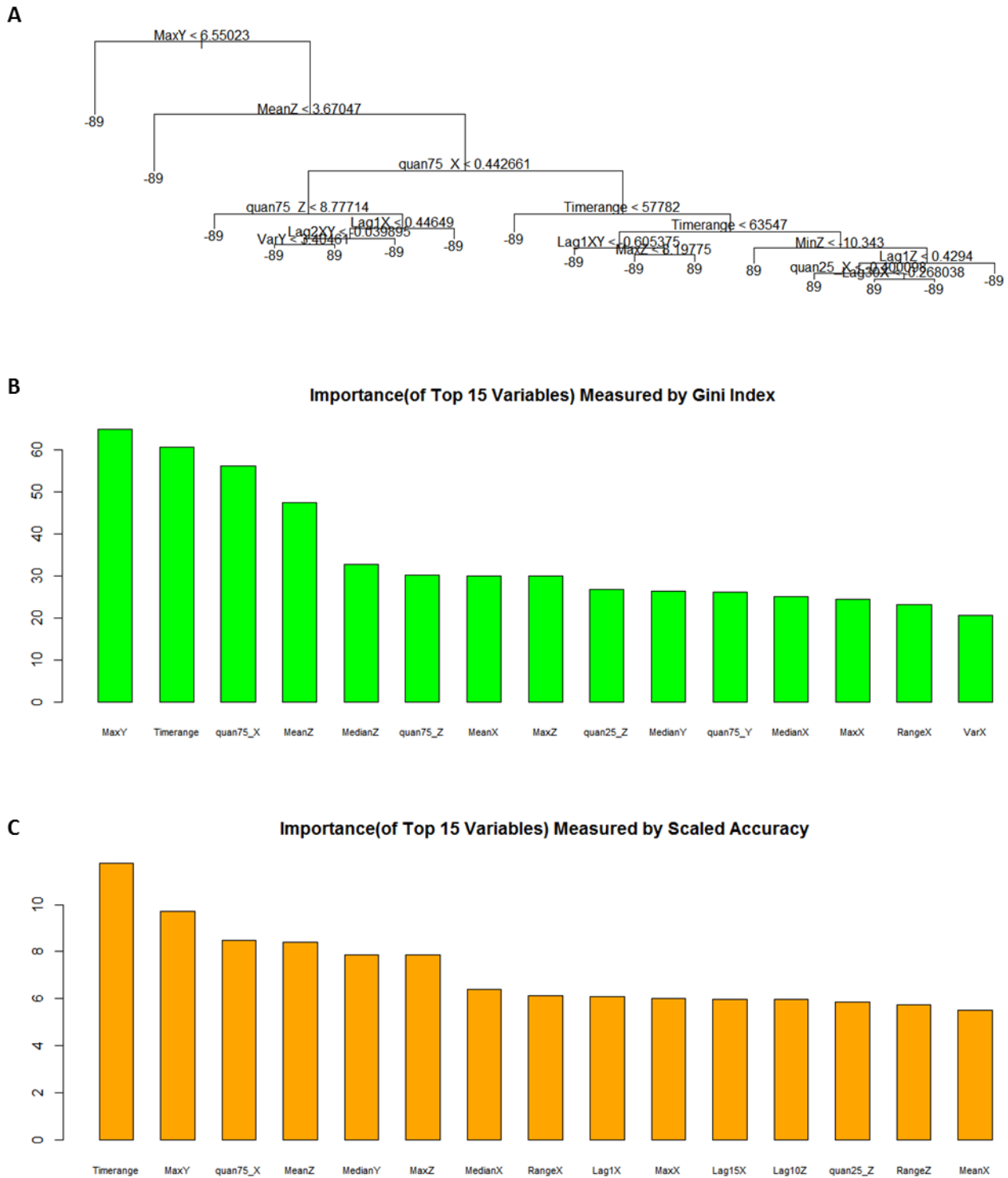


Figure 8. Histograms of selected feature variables for Devideld 89.

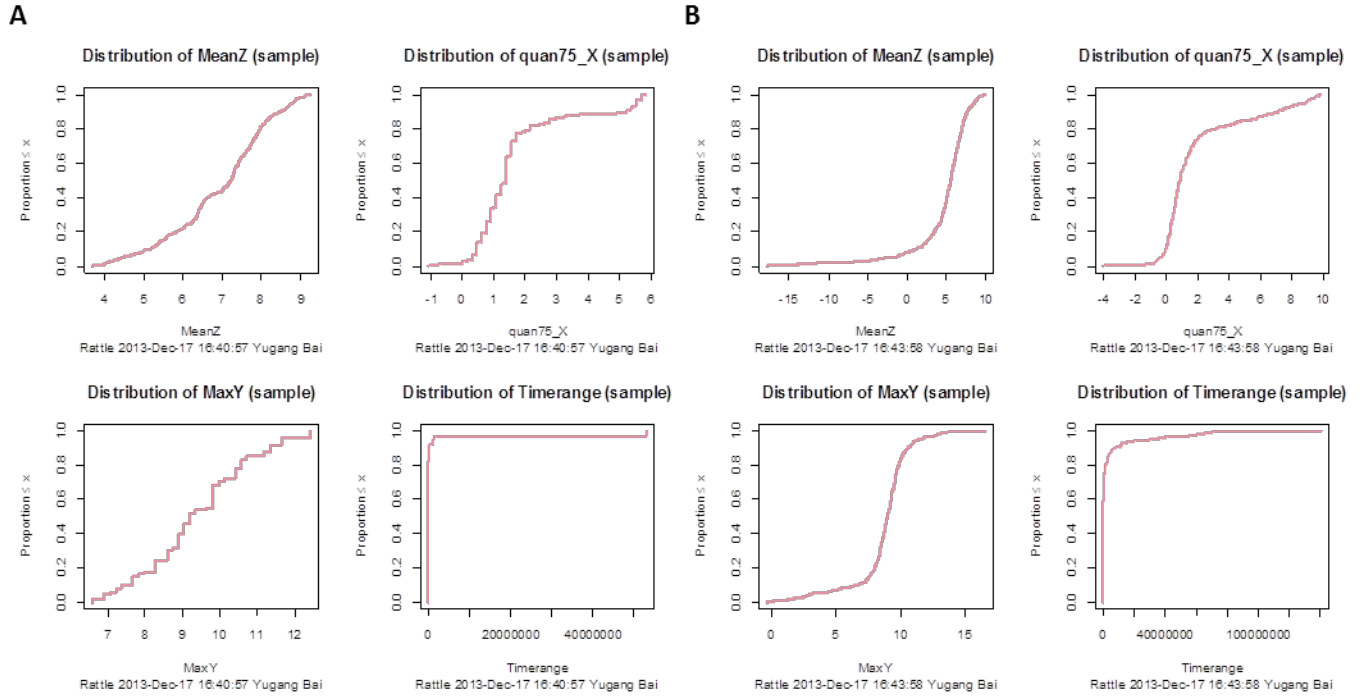
**Table 7.** Train models of DeviceId 89 and their parameter selection.

Method	Parameter	test error
RandomForest	mtry=m, ntree=50	0.02
Tree	Grow a big tree with mindev=0.005, minsize=2 Apply 5-fold cv to pick the optimal tree with size=15 Prune big tree, method="misclass"	0.0475
Boosting	distribution="adaboost", n.tree=1000 shrinkage=0.05, bag.fraction=0.5, cv.folds= 5 iteration=872 ( <b>Figure 9A</b> )	0.0225
Gaussian Kernel SVM	sigma = 9.09002200424809e-06 Number of Support Vectors : 857	0.1125
Lasso	lambda.min=0.0004429133	0.0875
KNN	k=5 ( <b>Figure 9B</b> )	0.0775

**Figure 9.** Parameter optimization in boosting and KNN methods. (A) Iteration optimization in boosting method. (B) Error rates with different k values in KNN method.



**Figure 10.** Variable selection in tree and RandomForest methods. (A) Variables selected by tree methods. (B) Variable importance plot measured by Gini Index in RandomForest method. (C) Variable importance plot measured by scaled accuracy in RandomForest method.



**Figure 11.** Comparison of the four importance variables between (A) DeviceId 89 and (B) DeviceId 7 with cumulative probability plot.

### III. Prediction of Test Data

Now we proceeded to train 387 binary classifiers using the above-mentioned method and parameters and predict the DeviceId of the sequences in the real test data. **Table 8** summarizes the accuracies with different methods calculated from kaggle. According to table 8, tree, random forest and boosting have similar performance in this project, and make ~76% of prediction accuracy. The prediction accuracy of KNN and SVM is only about 60%.

**Table 8.** Summary of accuracies of test sequence prediction with different models.

Model	Accuracy
Tree	74.632%
RandomForest	76.546%
Boosting	76.357%
KNN (k=5)	64.507%
Gaussian Kernel SVM	61.119%
Voting among tree, RandomForest and boosting	77.313%



## Conclusion

In this project, we used different methods to transform the training and test datasets and applied statistical models on the different transformed datasets to predict users of mobile accelerometers, from the accelerometer data (movement patterns) recorded for 387 different individuals (devices) over a period of several months. Clustering methods seem not to work for this project, due to the high error rate and computation cost. Based on the original data of X, Y, and Z-coordinates, those statistical methods, i.e., tree, random forest, boosting, KNN and SVM, generated comparable high test error rate. Fortunately, binary classifier method with extracted sequence features was proven to be the most effective way to predict the DeviceId of the sequences in our study. Among these extracted sequence features, Timesrange, MeanZ, MaxY and quan75\_X were found to be the most important variables/features to make prediction. Using tree method, RandomForest and boosting, we achieved about 76% prediction accuracy of test sequence, and furthermore the accuracy was increased to 77% when tree, RandomForest and boosting were used to vote for the final prediction.