Human Activity Tracking

Aadit Patel **Pranav Shah**

Sharon Richu Shaji

Abstract

This project aims to compare the performance of various machine learning models for the WISDM activity tracking dataset. Tracking human activity correctly would allow for better health monitoring systems which would lead to healthier lifestyle. The dataset comprises of information from 51 subjects who performed 18 activities for 3 minutes each which we then grouped into 3 main classes, "Non Hand Movements", "Hand Oriented General Movements" and "Hand Oriented Eating Movements". The data is collected from their phone's accelerometer and gyroscope as well as smart watch's accelerometer and gyroscope and is sampled 20 times per second (20 Hz). In this project we have used a dataset which was derived from the original dataset and has statistical features computed from the time series data. In this work, we compare models like Logistic Regression, Decision Trees, Random Forest, XGboost, AdaBoost, AutoML and Artificial Neural Networks.

1 Motivation

In today's world, smartphones and smartwatches are widely used by people. Using these smart devices, we can gather valuable information about human activity which could have great benefits in keeping track of ones physical health. The main motivation for our project is exactly this. The development of machine learning techniques coupled with access to high computation power makes it feasible to develop, train and deploy machine learning models with ease. We aim to use the accelerometer and gyroscope data of these smart devices to predict human physical activity. These predictions can then be used to understand various health concerns like blood pressure levels, heart rate and even daily water intake and food habits which would in turn be useful for remote tracking of patients.

2 Related Work

The previous work with the WISDM dataset involves two major approaches. One approach deals with using statistical tools to extract features from the time series data to transform it into a n x p matrix which is then used to make predictions. Another approach deploys deep learning models like LSTM and ConvLSTM on the dataset. [1] uses the raw time series data from the sensors and uses various statistical techniques to create 93 features. Work done in [2] focuses on comparing the performance of statistical models like KNN, decision trees and random forest. It uses the statistically extracted like in [1] to train the models. Work done by [4] also falls in the same category as [1] and [2]. They have compared more complex models like LibSVM and J-RIP with bagging models and random forests. A contrasting approach is taken by [3] where they compare deep learning models like CNN, LSTM, RNN on the raw data. Latest work on the dataset involves one done by [5]. Their work focuses on diabetic control from baseline studies of the WISDM dataset models.

3 Dataset

3.1 Introduction

In this project, we use the WISDM Smartphone and Smartwatch Activity and Biometrics Dataset provided by UCI Machine Learning Repository. The raw time series data set is divided into non-overlapping segments of 10 sec windows, per subject per activity. This 10 sec window provides 200 readings in the window, as the frequency at which the readings were registered are 20 readings/sec. Features are then generated using readings from these windows. The total number of observations (n) is 23,074, 17,281, 18,211, 16,533 for phone accelerometer data, phone gyroscope data, watch accelerometer data and watch gyroscope data respectively. The total number of features (p) for all 4 datasets is 91.

The link to the dataset can be found here.

Table 1. Dataset split-up

	Entries	Features
Phone Accelerometer Data	23,074	91
Phone Gyroscope Data	17,281	91
Watch Accelerometer Data	18,211	91
Watch Gyroscope Data	16,533	91

A short description about each feature is given below:

1. ACTIVITY : Label encoded values of the activity labels. In this dataset, there are 18 activities mentioned by the authors. These 18 categories can be grouped into 3 classes, "Non Hand Movements", "Hand Oriented General Movements" and "Hand Oriented Eating Movements". All models developed and compared in this work use these 3 grouped classes to make predictions.
2. X0-9; Y0-9, Z0-9: These 30 features collectively show the distribution of values over the x, y, and z axes. We call this a binned distribution. For each axis we determine the range of values in the 10s window (max – min value), divide this range into 10 equal-sized bins, and then record the fraction of values in each bin.
3. X,Y,ZAVG: Average sensor value over the window (per axis).
4. X,Y,ZPEAK: Time in milliseconds between the peaks in the wave associated with most activities.(per axis).
5. X,Y,ZABSOLDEV: Average absolute differ-ence between the each of the 200 readings.(per axis)
6. X,Y,ZSTANDDEV: Standard deviation of the 200 values (per axis)
7. X,Y,ZVAR: Variance of the 200 values (per axis)
8. XMFCC0-12, YMFCC0-12, ZMFCC0-12: MFCCs represent short-term power spec-trum of a wave, based on a linear cosine transform of a log power spectrum on a non-linear mel scale of frequency. There are 13 values per axis.
9. XY, XZ, YZCOS: The cosine distances between sensor values for pairs of axes (three pairs of axes).
10. XY, XZ, YZCOR: The correlation between sensor values for pairs of axes (three pairs of axes).
11. RESULTANT: Average resultant value, computed by squaring each matching x, y, and z value, summing them, taking the square root; over 200 readings.

4 Dataset Analysis

4.1 Summary and Visualization

As mentioned in the previous section, the 18 labels in the output were compressed into 3 label sections. The 3 label sections, in all the 4 datasets (the 4 datasets mentioned in Table 9) were found to be balanced. A visualization of the balanced distribution of labels in all 4 datasets is shown in Figure 1.

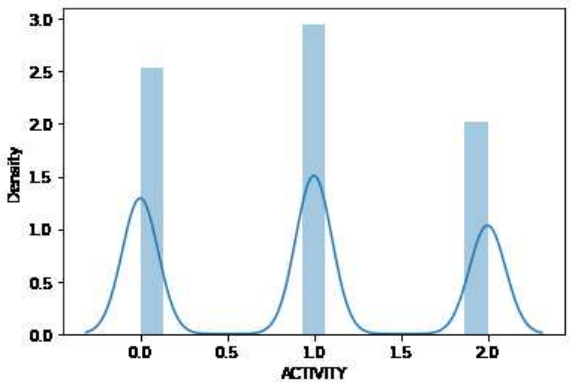

Fig. 1. Class distribution

The correlation between features can be seen in the heatmap given in Figure 2 and 3.

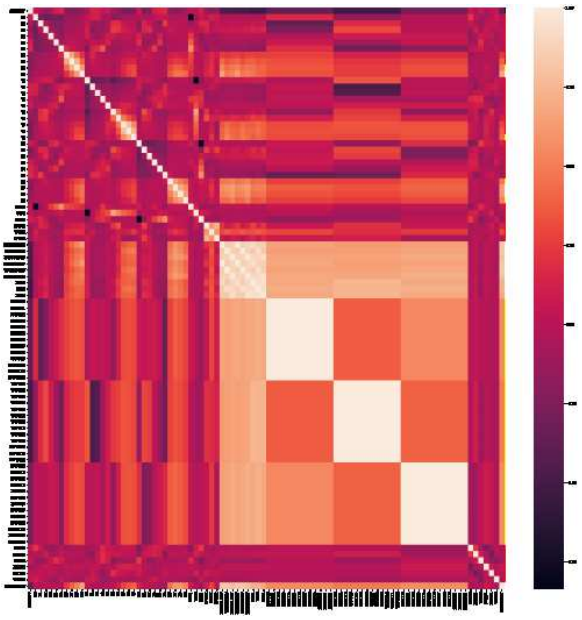

Fig. 2. Correlation Heatmap - Accelerometer data

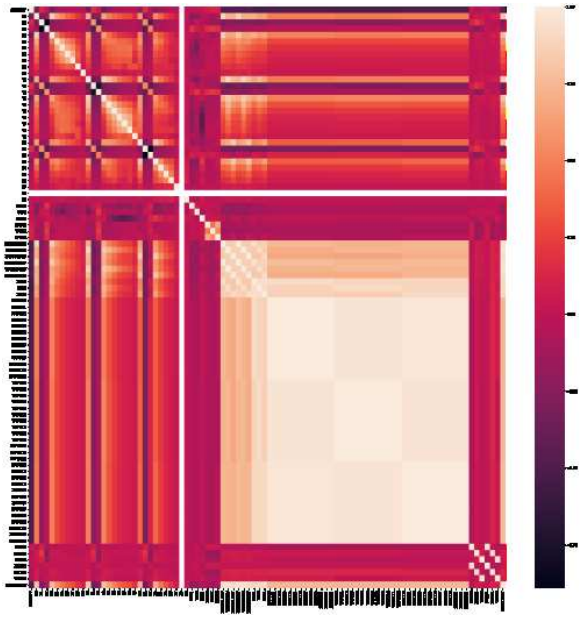

Fig. 3. Correlation Heatmap - Gyroscope data

4.2 Principle Component Analysis

We used principle component analysis to reduce the dimensions of our dataset. Since our dataset has an initial feature space of 91 features, it was trivial to check the correlations between the features. From the heatmaps of the datasets, we observed high correlation between the features. After reducing the dimensions of the dataset using PCA we found that 95% of variance in the datasets could be explained by 28 components (for accelerometer datasets) and 23 components (for gyroscope datasets). We trained models on the reduced dimensional as well as the normal high dimensional dataset. The results are summarized in Table 10 - Table 13.

5 Problem Formulation

The chosen task is a supervised multiclass classification problem where the inputs are the features extracted from the the time series data and the outputs are the activities performed. The dataset contains data that is prelabeled into distict classes, and thus classifies as a supervised classification problem. Since the dataset is numerical, there is a scope to explore multiple models and compare their performace. Once the features are extracted from the raw data, 80% of the dataset is used as the training dataset, which will be used for model training, and 20% of the dataset is taken as testing dataset, which will be used to test the efficiency and performance of the models. The choice of loss function will depend on the model used for classification. The details about the models are given in Section 6. Since our datasets were balanced we chose accuracy as our preferred metric for performance analysis. We tested the models with and without hyperparameter tuning as well as with and without feature selection using PCA. The results are summarised in Table 10, Table 11, Table 12 and Table 13.

6 Methodologies

Below are the different models we trained and tested our data against. We performed PCA for feature selection and compared the results obtained by training and testing the different models with the original dataset as well with the dataset after feature selection. We also performed hyperparameter tuning to obtain the optimal parameters for the different models and datasets. The major python packages used are scikit learn, xgboost, autosklearn, pandas, numpy, hyperopt and matplotlib.

6.1 Logistic Regression

Logistic Regression is chosen as our baseline model. It is one of the most simplest and commonly used model for classification problems. This method is usually used for binary classification but can also be used for multiclass classification such as ours by transforming it into a multiple binary classification problem. Since logistic regression tends to perform well on most linear datasets which have classification task, we choose it to be our baseline.

6.2 Decision Trees

Decision Trees are one of the most easiest model to implement and understand. It can be used for both regression as well as classification problems and hence suits the problem statement we're tackling. In decision trees, the data is split into homogenous groups based on decision rules inferred from the training data that has the most information gain.

6.3 AdaBoost

AdaBoost is a boosting technique, and thus combines various weak learners to create a strong model. The idea is that each weak learner can overfit on some feature and then we average out the predictions from various such weak learners to get an more accurate prediction. The weak learners are usually decision trees. Each sample is assigned a equal weight at the starting of the training process which gets updated depending on the prediction of that sample made by the model.

6.4 Random Forest

Random Forest is supervised machine learning algorithm. The random forest algorithm is an extension of the

bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees. The feature randomness, also called as "random subspace method", generates a random subsets of features which ensures low correlation among decision trees. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.

6.5 XGBoost

XGBoost stands for Extreme Gradient Boosting. XGBoost is an implementation of Gradient Boosting Machines (GBM) and is used for supervised learning. XGBoost was built to push the limit of computational resources for boosted trees. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set. XGBoost was built to push the limit of computational resources for boosted trees. XGBoost is an implementation of GBM, with major improvements. GBM's build trees sequentially, but XGBoost is parallelized. This makes XGBoost faster. Feature importance was extracted for the four datasets (Figure 4 - Figure 7).

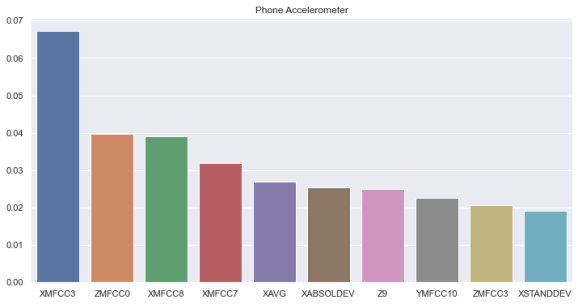

Fig. 4. Top 10 important features of Phone Accelerometer

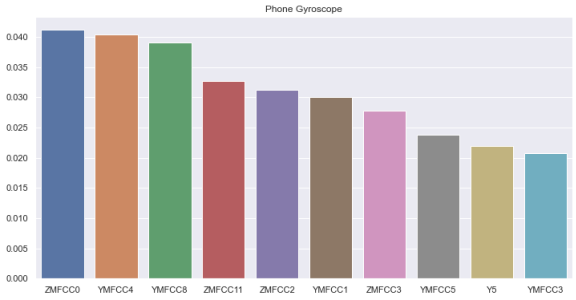

Fig. 5. Top 10 important features of Phone Gyroscope

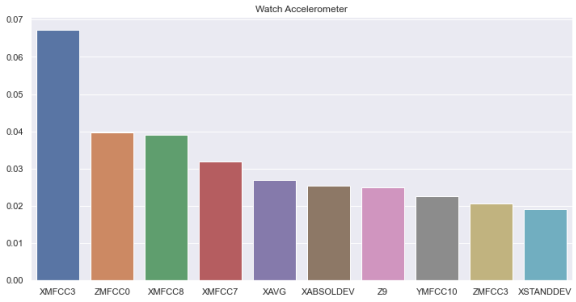

Fig. 6. Top 10 important features of Watch Accelerometer

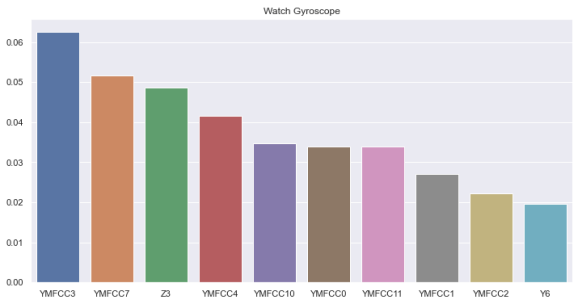

Fig. 7. Top 10 important features of Watch Gyrposcope

6.6 ANN

The network used in this project comprised of an input layer which has 91 nodes, owing to the 91 features from our dataset and the output layer comprises of 3 nodes due to our 3 classes. The architecture for our network is mentioned in the table. In order to capture the non linearity in the dataset, we made the model with deep layers and higher nodes per layer. It was evident from our experimentation that simpler models had high bias and were thus not able to capture the variation of the dataset. For the activation function, we choose Relu due to its high efficiency with non-linear input data and no *zero* gradient problem. The final layer had an activation function of softmax because we are using a cross entropy loss function and thus require that the output from our model be a probability distribution of the classes.

6.7 AutoML

Automated Machine Learning (AutoML) eliminates the need for machine learning experts to build and train models. The success of machine learning heavily relies on experts to preprocess the data, construct appropriate features, select the right model, optimize hyperparameters etc. With AutoML, machine learning methods can now be implemented without this expert knowledge. It takes in labeled training data as input and outputs an optimized model.

7 Experiments and Results

7.1 Hyperparameter tuning

7.1.1 Hyperopt

Hyperopt uses a form of Bayesian optimization for parameter tuning that allows you to get the best parameters for a given model. It can optimize a model with hundreds of parameters on a large scale. In the 'hyperopt' library, we have specifically used the 'Tree Parzen Estimator' method or TPE to optimize the parameters. TPE uses a cheap surrogate model to estimate the performance of the expensive objective function on a set of parameters. For hyperparameter tuning, we can provide a considerable range for each hyperparameter as the library will not try the model on all combinations of the hyperparameters. It will use the history of score and combination of hyperparameters to choose the next set of hyperparameters to test while minimizing some score. Table 6 refers to the hypermarameters tuned for Random Forest Model using hyperopt. Table 7 refers to the hyperparameters tuned for the XGBoost model using hyperopt and Table 2, Table 3, Table 4 and Table 5 refer to the hyperparameters tuned for decision trees.

1. Decision Tree

Table 2. Phone Accelerometer

	Without PCA	**With PCA**
max_depth	33	86
splitter	'best'	'best'
criterion	"entropy"	"entropy"
min_samples_split	2	2
max_features	None	None

Table 3. Phone Gyroscope

	Without PCA	**With PCA**
max_depth	42	34
splitter	'best'	'best'
criterion	"gini"	"entropy"
min_samples_split	2	2
max_features	None	None

Table 4. Watch Accelerometer

	Without PCA	**With PCA**
max_depth	41	23
splitter	'best'	'best'
criterion	"entropy"	"entropy"
min_samples_split	2	4
max_features	None	None

Table 5. Watch Gyroscope

	Without PCA	**With PCA**
max_depth	15	14
splitter	'best'	'best'
criterion	"gini"	"gini"
min_samples_split	3	3
max_features	None	None

2. Random Forest
3. XGBoost

Table 6. Hyperparameters for all datasets (Random Forest)

Parameter	Value
max_depth	15
criterion	”entropy”
min_samples_split	4
max_features	’auto’
n_estimators	450
n_jobs	-1

Table 7. Hyperparameters for all datasets (XGBoost)

Parameter	Value
eta	0.4
colsample_bytree	0.409
gamma	1.43
max_depth	18
min_child_weight	4
n_estimators	400
reg_alpha	3.08
reg_lambda	0.368
colsample_bylevel	0.483
max_delta_step	1
scale_pos_weight	0.7
subsample	0.89
objective	’multi:softmax’

7.1.2 Grid Search Cross Validation

Grid Search Cross Validation is another hyperparameter tuning method commonly used. It takes a user given parameter space and outputs an optimal combination of parameters based on the scoring metric given. We have used the GridSearchCV function from scikit-learn's model_selection package for some of the models due to the high execution time of the above mentioned hyperopt tuning method. The following optimal parameters were obtained:

2. AdaBoost
1. Logistic Regression

Table 8. Hyperparameters for all datasets

Parameter	Value
penalty	’elasticnet’
solver	’saga’
multi_class	multinomial’
C	0.1
max_iter	4
n_jobs	-1
l1_ratio	0

Table 9. Hyperparameters for all datasets

Parameter	Value
algorithm	’SAMME’
learning_rate	0.5
n_estimators	100

7.2 Metrics

For evaluating the performance of the models, we used accuracy, precision, recall and F1 score. We have used accuracy as the main evaluation metric for all GridSearch and hyperopt parameter tuning because the dataset is balanced. Since our task is a multiclass classification task, making accuracy as the primary metric makes it easier to make sense of the model and draw conclusions.

We have also calculated precision, recall and F1 score for all the models. This gave us idea on how the model is performing to every class and whether it is favouring one class over the other. Since the dataset was balanced, the precision and recall scores for all models were very close to one another.

8 Conclusions and Discussions

In this work, we compared the performance of various machine learning models based on their accuracy, precision, recall and F1 score. Our approach on evaluating the models was based on the analysis of our dataset. We found that the dataset is balanced which drove our methodology to focus on improving accuracy of our models. In this work, we used two methods to find the optimal hyperparameters, hyperopt and GridSearch with cross validation. We observed that GridSearchCV proved to be practical

when the search space was less due to its high computational power. For higher hyperparameter search space models, we used hyperopt. From our experiments, we found that XGBoost, Random Forest, AutoML and ANN work the best for all 4 datasets. The average accuracy across all 4 datasets of XGBoost was found to be 88.11% and that of Random forest was found to be 86.23%, which are the top 2 performing models. We also observed that the performance of models on the accelerometer data is better than that of the gyroscope data. This stems from the fact that the gyroscope data is more correlated than that of accelerometer data, thus affecting its predictive performance. We concluded that PCA proved impractical for the given dataset since it was non linear in nature. In this work, we tried using PCA to compress the data and use the components for training, but that did not prove to give us higher accuracy. A potential future work for the project can include reducing the dimensions of the given dataset using autoencoders. This is because autoencoders would be able to express the non-linear nature of the dataset which PCA cannot. Another approach to solve the problem could be from the raw time series dataset. This would involve using deeplearning models like LSTM and RNN.

9 References

[1] Weiss, Gary M. "Wisdm smartphone and smartwatch activity and biometrics dataset." UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set (2019).

[2] Weiss, Gary M., Kenichi Yoneda, and Thaier Hayajneh. "Smartphone and smartwatch-based biometrics using activities of daily living." IEEE Access 7 (2019): 133190-133202.

[3] Peppas, Konstantinos, et al. "Real-time physical activity recognition on smart mobile devices using convolutional neural networks." Applied Sciences 10.23 (2020): 8482.

[4] Walse, K. H., R. V. Dharaskar, and V. M. Thakare. "Performance evaluation of classifiers on WISDM dataset for human activity recognition." Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies. 2016.

[5] Carlson, Anders L., et al. "Hypoglycemia and glycemic control in older adults with type 1 diabetes: baseline results from the WISDM study." Journal of diabetes science and technology 15.3 (2021): 582-592.

10 Acknowledgements

We would like to thank Prof. Lyle Ungar for giving us the opportunity to do this project. We would also like to share our sincere gratitude to our TA, Yifei Li, for his valuable guidance and helpful suggestions throughout this project.

Model	Accuracy	Accuracy*	Precision	Precision*	Recall	Recall*	F1-Score	F1-Score*
Logistic Regression	0.6565	0.6370	0.6612	0.6434	0.6543	0.6358	0.6561	0.6375
Logistic Regression (tuned)	0.6494	0.6364	0.6546	0.6428	0.6474	0.6350	0.6350	0.6396
Decision Tree	0.8777	0.8279	0.8780	0.8268	0.8787	0.8279	0.8783	0.8273
Decision Tree (tuned)	0.8758	0.8290	0.8763	0.8281	0.8761	0.8283	0.8762	0.8281
AdaBoost (tuned)	0.626	0.613	0.642	0.634	0.635	0.599	0.631	0.607
Random Forest (tuned)	0.925	0.9109	0.9249	0.9120	0.9260	0.9093	0.9254	0.9106
XGBoost (tuned)	0.9328	0.9126	0.9329	0.9123	0.9325	0.9122	0.9327	0.9122
AutoML	0.9326	-	0.9332	-	0.9320	-	0.9325	-
ANN	0.889	-	0.8896	-	0.889	-	0.8891	-

Table 10. Results of Phone Accelerometer Dataset. (*) stands for results when PCA is applied on the dataset

Model	Accuracy	Accuracy*	Precision	Precision*	Recall	Recall*	F1-Score	F1-Score*
Logistic Regression	0.6204	0.6135	0.6372	0.6272	0.6228	0.6170	0.6240	0.6172
Logistic Regression (tuned)	0.6063	0.6051	0.6263	0.6248	0.6071	0.6057	0.6102	0.6090
Decision Tree	0.6598	0.6456	0.6575	0.6410	0.6556	0.6406	0.6564	0.6451
Decision Tree (tuned)	0.6502	0.6450	0.6467	0.6402	0.6449	0.6394	0.6457	0.6397
AdaBoost (tuned)	0.592	0.562	0.632	0.586	0.620	0.563	0.595	0.568
Random Forest	0.7324	0.7411	0.7384	0.7479	0.7325	0.7417	0.7319	0.7407
XGBoost	0.7671	0.7477	0.7663	0.7490	0.7657	0.7477	0.7648	0.7466
AutoML	0.7732	-	0.7768	-	0.7728	-	0.7734	-
ANN	0.725	-	0.73	-	0.725	-	0.726	-

Table 11. Results of Phone Gyroscope Dataset. (*) stands for results when PCA is applied on the dataset

Model	Accuracy	Accuracy*	Precision	Precision*	Recall	Recall*	F1-Score	F1-Score*
Logistic Regression	0.7952	0.7699	0.7978	0.7702	0.7958	0.7710	0.7958	0.7700
Logistic Regression (tuned)	0.7669	0.7672	0.7693	0.7679	0.7659	0.7680	0.7670	0.7673
Decision Tree	0.8915	0.8493	0.8920	0.8488	0.8910	0.8491	0.8915	0.8490
Decision Tree (tuned)	0.8888	0.8438	0.8898	0.8432	0.8884	0.8435	0.8890	0.8433
AdaBoost (tuned)	0.712	0.737	0.728	0.757	0.725	0.732	0.726	0.738
Random Forest	0.9341	0.9283	0.9351	0.9297	0.9331	0.9269	0.9337	0.9279
XGBoost	0.9467	0.9346	0.9468	0.9347	0.9466	0.9342	0.9466	0.9344
AutoML	0.9492	-	0.9491	-	0.9488	-	0.9487	-
ANN	0.922	-	0.922	-	0.922	-	0.922	-

Table 12. Results of Watch Accelerometer Dataset. (*) stands for results when PCA is applied on the dataset

Model	Accuracy	Accuracy*	Precision	Precision*	Recall	Recall*	F1-Score	F1-Score*
Logistic Regression	0.7127	0.6797	0.7108	0.6783	0.7156	0.6845	0.7109	0.6787
Logistic Regression (tuned)	0.6368	0.6380	0.6401	0.6416	0.6444	0.6450	0.6358	0.6368
Decision Tree	0.7901	0.7719	0.7878	0.7689	0.7861	0.7692	0.7869	0.7690
Decision Tree (tuned)	0.7973	0.7792	0.7938	0.7780	0.7948	0.7786	0.7943	0.7782
AdaBoost (tuned)	0.577	0.553	0.585	0.540	0.598	0.565	0.573	0.534
Random Forest	0.8575	0.8693	0.8561	0.8680	0.8558	0.8664	0.8551	0.8667
XGBoost	0.8778	0.8693	0.8766	0.8684	0.8757	0.8666	0.8760	0.8674
AutoML	0.8799	-	0.8783	-	0.8802	-	0.8782	-
ANN	0.8412	-	0.842	-	.8412	-	0.8415	-

Table 13. Results of Watch Gyroscope Dataset. (*) stands for results when PCA is applied on the dataset