QATAR UNIVERSITY

COLLEGE OF ENGINEERING

RECOGNIZING STEREOTYPED BEHAVIOR IN CHILDREN WITH AUTISM

BY

RANIM HAISAM FARAJ

A Project Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2020

COMMITTEE PAGE

The members of the Committee approve the Project of

Ranim Haisam Faraj defended on 22/04/2020.

Dr. Tarek Mohamed El-Fouly
Thesis/Dissertation Supervisor

Prof. Amr Mohamed
Committee Member

Dr. Elias Yaacoub
Committee Member

Dr.Tamer Khattab
Committee Member

ABSTRACT

FARAJ, RANIM H., Masters : June : [2020], Masters of Science in Computing

Title: Recognizing Stereotyped Behavior in Children with Autism

Supervisor of Project: Dr. Tarek M. El-Fouly.

This project works on helping in identifying and recognizing autistic children's stereotyped behaviors, which can help in diagnosing autism on children. The recognition accomplished by building a signal processing model that collects data from a smartwatch equipped with a gyroscope and accelerometer in order to produce a feature vector of 316 features. This feature vector is used to choose a predictive model with the highest accuracy, which is Ridge classifier in this project. The results show that those common stereotype behaviors could be recognized using the Ridge machine learning algorithm with overall average accuracy ranges between 98.7% to 99.5 %. For hand flapping, head banging, and running back and forth, the overall precision ranges between 98% to 100 %, overall recall ranges between 98% to 100 %, overall F1-score ranges between 98% to 100 % and overall macro, weighted and micro averages is 99 %. This Ridge classifier used to implement a real-time application developed on a smartphone (iPhone) to detect the stereotyped behaviors for autistic children who are wearing the smartwatch (Apple watch).

DEDICATION

To my beloved family members for their love, help, and support through the difficult times

To Dr. Tarek Elfouly who have encouraged me to finish this degree and for guidance and support

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RBF: Radial Basis Function

SVM: Support Vector Machine

KNN: K Nearest Neighbors

DT: Decision Trees

CHAPTER 1: INTRODUCTION

Today's smart devices, such as a smartwatch, come equipped with sensors that used for data collection for health monitoring applications [1]. The most popular sensors found in a smartwatch is Gyroscope, Accelerometer, Magnetometer, and Heart rate monitor [2]. The accelerometer is an electromechanical device that used to measure the linear acceleration using the vibrations related to movement. The gyroscope is a sensor that employs the earth's gravity to determine rotational changes and angular position with keeping orientation [3,4]. The data collected from accelerometer and gyroscope sensors data can aid in the detection of several human activities. The human activities that this project is interested in is the one related to children with autism. Autism spectrum disorder refers to a wide range of conditions, including social skill problems, stereotyped behaviors, speech, and nonverbal communication [5]. In 2018, according to the Centers for Disease Control and Prevention (CDC), autism affected about 1 in every 59 children in the US [6]. Moreover, according to the CDC, there is no medical detection for autism [6,7].

Stereotyped behavior is unusual and socially undesirable, repetitive behaviors, and it is considered a crucial diagnostic feature of people with autism [8]. This project aims to employ machine learning to recognize three common stereotype behaviors found in the children, which are hand flapping, head banging, and running back and forth through data collected from a smartwatch. The results show that those common stereotype behaviors could be recognized using the Ridge machine learning algorithm with overall average accuracy ranges between 98.7% to 99.5 %. For hand flapping, head banging, and running back and forth, the overall precision ranges between 98% to 100 %, recall ranges between 98% to 100 %, overall F1-score ranges between 98% to 100 %, and overall macro, weighted and micro averages of 99 %. This Ridge classifier

used to implement a real-time application developed on a smartphone (iPhone) to detect the stereotyped behaviors for autistic children who are wearing the smartwatch (Apple watch).

The remainder of this report is structured as follows: Section 1.1 describes the goals and objectives of this project. Chapter 2 presents related work projects related to human activity detection and background about the machine learning multiclass classification algorithms. Chapter 3 explore the methodology used to implement this project in detail with the analysis of the results. Chapter 4 explores the real-time application developed to detect the stereotyped behaviors for autistic children. Chapter 5 concludes the projects with future work and challenges.

1.1. Goals and Objectives

This project aims to develop a model that can help in detecting and recognizing autistic children's stereotyped behaviors. This project builds a signal processing model that collects data from a wearable sensor (gyroscope and accelerometer) that exists in the market (smartwatch). Then the data will be fed to a classification machine learning model for training the algorithms to produce the final predictive model with the highest accuracy that implemented in an application that helps in the classifying the children stereotyped behaviors, which aid in diagnosing the process of autism on children.

CHAPTER 2: RELATED WORK AND BACKGROUND

2.1 Related Work

Human activity recognition has a vital role in several applications. The authors in [11] used SVM, Naïve Bayes, KNN to identify temporals patterns that can aid in the activity recognition process. Random Forest algorithm applied for the features extracted from time and frequency domain for recognition of a person who is walking, and it achieved an accuracy of 96.79% [29]. The researchers in [9] used data from accelerometer sensors with various machine learning techniques such as SVM, J48, AdaBoost, and Random forest to recognize the activity of walking, jogging, running, standing, and sitting and they attained an average accuracy of 98.8283 %. MobiRAR application was developed by [10], where sensor data from the mobile device is collected to recognize ten daily human activity, and they achieved an average accuracy of 93%. The authors in [12] managed to recognize eight daily human activities using statistical learning methods, which are Naive Bayes, K-nearest neighbor, Logistic regression, Bayesian network, and Multilayer Perceptron, and they attained an accuracy of 91.55 % for Bayesian network. Children with development disabilities stereotyped movement is detected using the Weka toolkit with a recognition accuracy of 91 % [13]. The researchers in [15] used data collected from functional magnetic resonance to diagnose autism using SVM RBF kernel, and the accuracy was 59.6 %. The authors in [14] used Microsoft sensor Kinect to recognize the hand flapping movement of children with autism using Dynamic Time Wrapping with a 51% detection rate and a 76% detection rate using the eZ43-Chrono watch. The author in [16] collected accelerometer data with 102 feature vector size, and this data is fed into a machine learning algorithm DT to detect hand flapping, and they achieved an accuracy of 93%. MIT researchers worked on the detection of children with

stereotyped autism behavior, and they achieved the highest recognition accuracy of 82.3% for SVM and 77.5% for DT [44]. The raw inertial signals do not give enough knowledge, so we cannot directly feed it to the machine learning algorithms, but we can use those data to generate a new feature vector that describes the shape, distribution, and nature of the signals. Feature vector produced could be orientation-invariant such as the work of [27], who extracted features from three-dimensional acceleration signals through applying Fourier transform and the work of [28] who extracted features from accelerometer and gyroscope for gait biometrics for motion recognition.

2.2. Multiclass Classification Algorithms

A multiclass classifier also can build based on multiple binary classifiers. In order to prevent overfitting of the data, cross-validation applied to train data where data divided into several subsets for training and one subset for validation [41].

2.2.1. Decision Trees

It is a tree traversal algorithm that splits the examples into decisions in the form of nodes [18, 19]. In each node, we create a test, and if the test passes, it will be processed to the left branch and the right branch otherwise [30]. This method is inherently supporting multiclass classification [17].

2.2.2. Support Vector Machine

It is an efficient method that uses a group of mathematical functions (kernels), and it is highly effective in high dimensional spaces [31, 32]. Examples of SVM kernels are RBF, polynomial, sigmoid, and Linear SVC. RBF kernel is the most used type of kernel because there is no need for prior knowledge about the data [32]. SVM kernels support multiclass classification as One-Vs-One; however, the Linear SVM method supports multiclass classification as One-Vs-The-Rest [17].

2.2.3. Random Forest

It has based on decision trees [33]. Moreover, this method is inherently supporting multiclass classification [17].

2.2.4. Logistic Regression

It is a simple and computationally efficient probabilistic method through binomial outcomes [18]. This method supports the multiclass classification as One-Vs-The-Rest [17].

2.2.5. Ridge

This method converts the target into values between -1 and 1, and then it handles the problem as a regression problem [34]. This method is inherently supporting multiclass classification [17].

2.2.6. K-Nearest Neighbors

It is classified based on its closest neighbor, and it requires distance computation of k-nearest neighbors [18,33]. This method is inherently supporting multiclass classification [17]

2.2.7. Hyperparameter Tuning

Hyperparameters are values set by the programmer, which often can help in estimating the parameters, and it can affect the final model prediction [35]. In order to choose the appropriate hyperparameter, hyperparameter tuning algorithms developed, such as Grid Search and Random Search.

CHAPTER 3: METHODOLOGY AND VALIDATION

3.1. Data Collection

As seen in Figure 1, during each experiment three acceleration accelerometer signals (X-Y-Z) with the unit of g (gravity of earth=9.8 m/s^2) and three gyroscope angular velocity signals (X-Y-Z) with the unit of rad/s were captured using Apple's Smartwatch Series 5 worn by the children. When the smartwatch is attached to the body, the activity's motion of this body will affect the watch acceleration and angular velocity. After the smartwatch finished collecting the data for an activity (behavior), the data are transferred to an iPhone 7 and then transferred to a MacBook laptop for offline machine learning model selection. The data from the watch are collected using the SensorLog application developed by Bernd Thomas [20]. The sensor's signals are measured every 0.05 seconds (sampling frequency of 20 Hz). The sampling frequency must be selected fast enough to capture the necessary signals needed for processing, but at the same time, it should be slow enough to avoid exposure to noise [21]. Moreover, the sampling frequency should be greater or equal twice the highest sampled frequency to avoid aliasing [22]. For the human body measurements, 98% of the spectral power is below 10 Hz [23]. So, the sampling frequency of 20 Hz is selected.

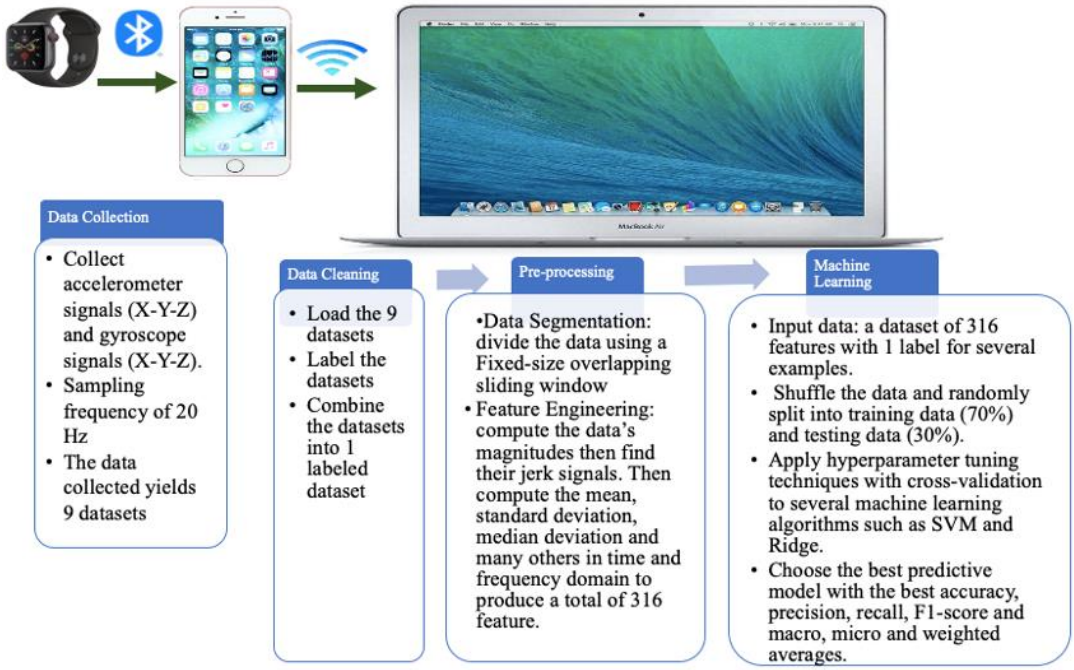

Figure 1. The process of recognizing the behaviors (hand flapping, head banging and running back and forth)

The training collected data from 3 autistic children in Qatar performing three behaviors (hand flapping – head banging- running back and forth) while wearing the Apple watch on their right hand. The data collected yields nine datasets that will be processed to produce one dataset that fed to the machine learning algorithms. The labels were manually added to the datasets, so a supervised learning algorithm used to generate the predictive model. The number of samples recorded for each behavior and every child found in Table 1.

Table 1 Number of Samples Recorded for Each Behavior

Child	Hand Flapping	Head Banging	Running Back and forth
1	8448	12109	11310
2	11141	3264	12959
3	3307	4954	7560

The choice of detecting the behavior using a smartwatch instead of a recording of a video for the kid's movement came because of the sensitive nature of children with autism that makes it better not to put them in constant surveillance. Also, this can protect the identity of the kids who had to diagnose. Moreover, the smartwatch is easy to use and looks like a regular watch. So, it blinds with their body, unlike putting a sensor on their head or their waist.

3.2. Pre-processing

3.2.1. Data Segmentation

The pre-processing computed using Python 3 language with a math library, and briefly explained in Figure 1. If the features computed for all data for each column in the original dataset, the final dataset that we will feed to the machine learning will have only three values for each feature (one for each behavior). For sure, this yields a small number of examples that are not suitable to train a machine learning algorithm. Moreover, it is not useful to compute the feature vector for each sample in the initial dataset. So, the solution will be to divide the data into small windows of fixed time before extracting the features. The method of windowing used is a Fixed size overlapping sliding window. In order the accommodate the data at the edge of the window, we used overlapping windows of 50% overlap. The data were divided into windows of size 1 seconds (1/sampling frequency = 20 sample/window), 2 seconds (2/sampling frequency = 40 sample /window) and 3 seconds (3/sampling frequency = 60 sample /window).

3.2.2. Feature Engineering

The feature engineering computed using Python 3 language with scipy, math, and statsmodels libraries, and briefly explained in Figure 1. For each window, we

will compute information that describes the signal, such as statistical features in time and frequency domain. For each example, the orientation independent jerk signal (rate of change) [25] calculated. Then the magnitude of the three signals (Z, Y, Z) for both sensors and its jerk signals is calculated using Vector L2 Norm (Euclidean norm) method [24], which found in Equation 1.

$$\text{Magnitude}(X, Y, Z) = \sqrt{X^2 + Y^2 + Z^2} \quad \text{Equation 1}$$

Several statistical methods applied to the sensor data, their magnitude, and jerk signals, a feature vector with a total number of features equal to 316 features produced. The statistical methods used are mean, standard deviation, median deviation, maximum, minimum, interquartile range, entropy, correlation, signal magnitude area, energy, Skew, and Kurtosis.

For both time and frequency domains, we calculate the mean. The mean can help in describing the distribution of a signal, and it is affected by outliers [26]. The total number of mean features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the standard deviation. Standard deviation can help to describe the distribution of a signal. It measures the spread of the distribution about the mean, and it is affected by outliers [26]. The total number of standard deviation features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the median deviation. The median deviation can help in describing the distribution of a signal, and it is not sensitive to outlier [36]. The total number of standard deviation features are (3

accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the maximum value. The total number of maximum features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the minimum value. The total number of minimum features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the interquartile range. It is a measure of statistical dispersion but is much more robust against outliers. It is where the middle fifty of our data in a dataset where most of the data lie. In other words, the interquartile range equal to the difference between the 75th data point and the 25th data point of our data [37]. The total number of interquartile Range features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For both time and frequency domains, we calculate the entropy. It measures of disorder, using probabilistic parameters [38]. The total number of entropy features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For time-domain only, we calculate the correlation. In this project, the correlation is between two axial combinations of the signals X, Y, Z. The total number of correlation features are (3 accelerometer signals + 3 gyroscope signal) * 1 domain * 2 jerk signals= 12 features.

For both time and frequency domains, we calculate the signal magnitude area. In this project, the sum of areas under each signal [39, 42]. Signal magnitude area

used for measuring a child's level of activity (distinguish between activity and inactivity) [40]. The total number of signal magnitude area features is (1 sum of accelerometer signals + 1 sum of gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =16 feature.

For both time and frequency domains, we calculate energy. Energy is Sum squared of each column for X, Y, Z [43]. The total number of energy features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 2 domain * 2 jerk signals =32 feature.

For frequency domain only, we calculate the skew. Skew measures of the lack of symmetry. In a histogram of data, the value of skew decides where the distribution is directed and by how much. In other words, it shows how much the data departed from the horizontal symmetry. The skew is computed in Equation 2 [45]. The total number of skew features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 1 domain * 2 jerk signals=16 feature.

$$skewness(g_1) = \frac{\sum \frac{(x-\bar{x})^3}{n}}{(\sum \frac{(x-\bar{x})^2}{n})^{3/2}}$$ Equation 2

Where $\bar{x}$ is the mean of the data set, and n is the sample size.

For frequency domain only, we calculate the kurtosis. Kurtosis measures the sharpness of the data peaks when graphed as a histogram. As the Kurtosis value increases, the histogram will have a sharper peak. The skew is computed in Equation 3 [45]. The total number of kurtosis features are (3 accelerometer signals + 3 gyroscope signal + 2 magnitudes) * 1 domain * 2 jerk signals=16 feature.

$$\text{Kurtosis}(a_4) = \frac{\sum\frac{(x-\bar{x})^4}{n}}{(\sum\frac{(x-\bar{x})^2}{n})^2} \text{ Equation 3}$$

Where $\bar{x}$ is the mean of the data set, and n is the sample size.

3.3. Machine Learning

The machine learning computed using Python 3 language with Scikit-learn, and briefly explained in Figure 1. The final dataset was shuffled and randomly split into training data (70%) and testing data (30%). Then hyperparameter tuning techniques Random and Grid search were used with Stratified K Fold cross-validation for DT, Linear SVM, SVM, Random Forest, Logistic Regression, Ridge and KNN algorithms. The initial hyperparameter fed to the hyperparameter tuning techniques found in Table 2. Leave on out cross-validation method, and the Gradient Boosted Decision Trees machine learning algorithm was tested, but it took a half-day with no output, so it was terminated, and another methods are chosen.

Table 2. Hyperparameter and their Values

Algorithm	Hyperparameter	Values
DT	Maximum Depth	3 to 10 with steps 2
Linear SVM	C	0.125, 0.5, 1, 2, 8, 16
SVM	Kernel	Polynomial, RBF, Sigmoid
SVM	Gamma	0.0078125, 0.125, 2
SVM	C	100, 10, 1.0, 0.1, 0.001
Random Forest	N estimators	10 to 1000 with steps 100
Random Forest	Maximum Depth	3 to 15 with steps 2
Logistic Regression	Penalty	l2 and l1
Logistic Regression	C	0.01, 0.1, 1, 10, 100
Ridge	Alpha	0.1 to 1.0 with steps 0.1
KNN	Weights	Uniform, Distance
KNN	N neighbors	1 to 21 with steps 2

3.4. Validation

3.4.1. Best Hyperparameter from Hyperparameter Tuning

After feeding the initial hyperparameter found in table 2 to the hyperparameter tuning techniques, the best hyperparameter value chosen which gave the best results on the hold out data by the techniques found in Table 3 for Random Search and Table 4 for Grid Search. The Table 4 shows the quality of the best hyperparameters chosen. The bigger value of the average Cross Validate Scores of Best Estimator, the better.

Table 3. Best Hyperparameter Yielded from Hyperparameter Tuning for Random Search

Algorithm	Hyperparameter	1 sec	2 sec	3 sec
DT	Maximum Depth	9	7	7
Linear SVM	C	0.125	0.5	1
SVM	Kernel	Polynomial	Polynomial	Polynomial
SVM	Gamma	2	0.0078125	0.125
SVM	C	10	0.001	0.001
Random Forest	N estimators	910	110	410
Random Forest	Maximum Depth	11	13	13
Logistic Regression	Penalty	I1	I1	I2
Logistic Regression	C	1	100	10
Ridge	Alpha	0.7	0.2	0.1
KNN	Weights	Distance	Distance	Uniform
KNN	N neighbors	7	3	3

Table 4. Best Hyperparameter Yielded from Hyperparameter Tuning for Grid Search

Algorithm	Hyperparameter	1 sec	2 sec	3 sec
DT	Maximum Depth	7	7	5
Linear SVM	C	16	1	16
SVM	Kernel	Polynomial	Polynomial	Polynomial
SVM	Gamma	0.0078125	0.0078125	0.0078125
SVM	C	100	100	100
Random Forest	N estimators	110	110	210
Random Forest	Maximum Depth	13	13	11
Logistic Regression	Penalty	I2	I1	I2
Logistic Regression	C	100	10	10
Ridge	Alpha	0.2	0.6	0.3
KNN	Weights	Distance	Distance	Distance
KNN	N neighbors	5	3	7

Table 5. Average Cross Validate Scores of Best Estimator

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.96	0.97	0.97	0.96	0.96
Linear SVM	0.91	0.91	0.90	0.92	0.86	0.91
SVM	0.95	0.95	0.95	0.96	0.95	0.96
Random Forest	0.98	0.98	0.98	0.99	0.99	0.98
Logistic Regression	0.98	0.99	0.94	0.92	0.99	0.94
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.90	0.91	0.92	0.91	0.92	0.92

3.4.2. Average Accuracy

Average accuracy shows the effectiveness of the used machine learning classifier by showing the degree of closeness to the true values of the labels. It computed using Equation 4 and the results for all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) shown in table 6.

$$\text{Average Accuracy (model)} = \frac{\sum_{i=1}^{k} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i}}{\text{k}} \qquad \text{Equation 4}$$

For k=total number of classes (3 classes), tp= true positive, tn= true negative, fp=false positive, fn=false negative.

Table 6. Average Accuracy for All Behaviors

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.974	0.975	0.972	0.972	0.975	0.972
Linear SVM	0.935	0.798	0.755	0.825	0.942	0.797
SVM	0.961	0.952	0.961	0.963	0.964	0.942
Random Forest	0.988	0.985	0.991	0.985	0.986	0.981
Logistic Regression	0.992	0.997	0.945	0.913	0.992	0.948
Ridge	0.991	0.988	0.992	0.987	0.995	0.992
KNN	0.910	0.927	0.909	0.904	0.919	0.937

The results show that those common stereotype behaviors could be recognized using the Ridge machine learning algorithm with overall average accuracy ranges between 98.7% to 99.5 %. So Ridge classifier is the most recommended model to classify those stereotype behaviors. The accuracy of DT is not profoundly affected by changing the window size and the hyperparameter tuning method. For Linear SVM low accuracies, it is either the data that does not work well with this classifier or the hyperparameter tuning method was not successful in finding the best hyperparameter. For SVM, the Polynomial kernel to use on the data and the accuracy of DT is not profoundly affected by changing the window size and the hyperparameter tuning method. The accuracy of KNN is lowest compared to the other classifiers.

3.4.3. Precision

Precision is the capability of the classifier not to label a negative example as

positive. It computed using Equation 5 and the results for all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) shown in Table 7, Table 8 and Table 9 for the three stereotyped behavior for children with autism.

$$\text{Precision (model)} = \frac{\sum_{i=1}^{k-1} tp_i}{\sum_{i=1}^{k-1}(tp_i + fp_i)} \quad \text{Equation 5}$$

For k=total number of classes (3 classes), tp= true positive, fp=false positive.

Table 7. Precision for Hand-Flapping

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.95	0.97	0.97	0.97	0.97
Linear SVM	0.91	0.62	0.89	0.70	0.91	0.97
SVM	0.94	0.93	0.94	0.94	0.94	0.91
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.98	0.92	0.89	0.99	0.92
Ridge	1.00	0.98	0.99	0.99	0.99	1.00
KNN	0.88	0.90	0.86	0.87	0.91	0.90

Table 8. Precision for Head-banging

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.98	0.97	0.99	0.97	0.97
Linear SVM	0.91	0.93	0.59	0.84	0.95	0.73
SVM	0.95	0.95	0.97	0.97	0.94	0.92
Random Forest	0.99	0.99	1.00	1.00	0.99	0.97
Logistic Regression	0.99	0.98	0.92	0.88	0.99	0.92
Ridge	0.98	0.98	1.00	0.99	0.99	0.98
KNN	0.93	0.94	0.96	0.93	0.92	0.97

Table 9. Precision for Running Back and Forth

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.99	0.97	0.96	0.98	0.97
Linear SVM	0.97	0.99	0.97	0.99	0.96	0.81
SVM	0.98	0.97	0.97	0.98	0.99	0.98
Random Forest	0.98	0.98	0.98	0.98	0.98	0.99
Logistic Regression	0.99	0.99	0.98	0.96	1.00	0.98
Ridge	0.99	1.00	0.99	0.98	1.00	0.99
KNN	0.92	0.94	0.91	0.92	0.93	0.95

The results show that those common stereotype behaviors could be recognized with the highest precision using the Ridge algorithm. The overall precision ranges between 98% to 100 % for hand flapping, head banging, and running back and forth.

3.4.4. Recall

The recall is the ability of the classifier to find all the positive samples. It computed using Equation 6 and the results for all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) shown Table 10, Table 11, Table 12 for the three stereotyped behavior for children with autism.

$$\text{Recall (model)} = \frac{\sum_{i=1}^{k-1} tp_i}{\sum_{i=1}^{k-1}(tp_i + fn_i)} \quad \text{Equation 6}$$

For k=total number of classes (3 classes), tp= true positive, fn=false negative.

Table 10. Recall for Hand-Flapping

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.96	0.97	0.96	0.96	0.96	0.95
Linear SVM	0.88	0.95	0.80	0.93	0.92	0.35
SVM	0.95	0.94	0.95	0.97	0.94	0.91
Random Forest	0.98	0.99	0.99	0.98	0.98	0.96
Logistic Regression	0.99	0.99	0.91	0.85	0.99	0.90
Ridge	0.98	0.99	1.00	0.98	0.99	0.98
KNN	0.83	0.89	0.85	0.84	0.85	0.91

Table 11. Recall for Head-banging

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.97	0.97	0.96	0.98	0.97
Linear SVM	0.97	0.91	0.96	0.95	0.93	0.98
SVM	0.96	0.94	0.95	0.95	0.96	0.96
Random Forest	0.98	0.97	0.99	0.97	0.98	0.99
Logistic Regression	0.99	0.97	0.94	0.91	0.99	0.96
Ridge	0.99	0.98	0.99	0.98	0.99	1.00
KNN	0.92	0.91	0.90	0.90	0.92	0.90

Table 12. Recall for Running Back and Forth

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.98	0.98	0.98	0.98	0.99
Linear SVM	0.95	0.61	0.57	0.67	0.96	0.98
SVM	0.97	0.97	0.98	0.96	0.98	0.96
Random Forest	1.00	0.99	0.99	0.99	1.00	0.99
Logistic Regression	0.99	0.99	0.97	0.96	0.99	0.97
Ridge	1.00	1.00	0.99	0.99	1.00	1.00
KNN	0.96	0.97	0.96	0.96	0.97	0.98

The results show that those common stereotype behaviors could be recognized with the highest recall using the Ridge algorithm. The overall recall ranges between 98% to 100 % for hand flapping, head banging, and running back and forth.

3.4.5. F1-score

F1-score is the harmonic mean of precision and recall. Using Equation 7 and the results for all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) shown in Table 13, Table 14, Table 15 for the three stereotyped behavior for children with autism.

$$\mathrm{F1-score\ (model)} = \frac{2*Precision*Recall}{Precision+\ Recall} \quad \text{Equation 7}$$

Table 13. F1-score for Hand-Flapping

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.96	0.96	0.97	0.97	0.96
Linear SVM	0.90	0.75	0.84	0.80	0.92	0.51
SVM	0.94	0.93	0.94	0.95	0.94	0.91
Random Forest	0.98	0.98	0.99	0.98	0.98	0.97
Logistic Regression	0.99	0.99	0.92	0.87	0.99	0.91
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.86	0.89	0.85	0.85	0.88	0.90

Table 14. F1-score for Head-banging

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.97	0.97	0.98	0.97	0.97
Linear SVM	0.94	0.92	0.73	0.89	0.94	0.84
SVM	0.96	0.94	0.96	0.96	0.95	0.94
Random Forest	0.99	0.98	0.99	0.98	0.98	0.98
Logistic Regression	0.99	0.98	0.93	0.89	0.99	0.94
Ridge	0.99	0.98	0.99	0.98	0.99	0.99
KNN	0.92	0.92	0.93	0.91	0.92	0.93

Table 15. F1-score for Running Back and Forth

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.98	0.98	0.98	0.97	0.98	0.98
Linear SVM	0.96	0.75	0.72	0.80	0.96	0.89
SVM	0.97	0.97	0.97	0.97	0.99	0.97
Random Forest	0.99	0.99	0.99	0.99	0.99	0.99
Logistic Regression	0.99	0.99	0.98	0.96	0.99	0.98
Ridge	0.99	1.00	0.99	0.99	1.00	1.00
KNN	0.94	0.95	0.93	0.93	0.95	0.96

The results show that those common stereotype behaviors could be recognized with the highest F1-score using the Ridge algorithm. The overall F1-score ranges between 98% to 100 % for hand flapping, head banging, and running back and forth.

3.4.6. Support

Support is the number of samples of the true label that lie in that class. It is computed for all classifiers through two hyperparameter tuning algorithms (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) is shown in Table 16, Table 17 and Table 18 for the three stereotyped behavior for children with autism.

Table 16. Support for Hand-Flapping

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	688	351	207	685	344	216
Linear SVM	688	351	207	685	344	216
SVM	688	351	207	685	344	216
Random Forest	688	351	207	685	344	216
Logistic Regression	688	351	207	685	344	216
Ridge	688	351	207	685	344	216
KNN	688	351	207	685	344	216

Table 17. Support for Head-banging

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	598	305	231	603	289	218
Linear SVM	598	305	231	603	289	218
SVM	598	305	231	603	289	218
Random Forest	598	305	231	603	289	218
Logistic Regression	598	305	231	603	289	218
Ridge	598	305	231	603	289	218
KNN	598	305	231	603	289	218

Table 18. Support for Running Back and Forth

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	961	466	309	959	489	313
Linear SVM	961	466	309	959	489	313
SVM	961	466	309	959	489	313
Random Forest	961	466	309	959	489	313
Logistic Regression	961	466	309	959	489	313
Ridge	961	466	309	959	489	313
KNN	961	466	309	959	489	313

3.4.7. Micro Average

Micro Average aggregates the contributions of all classes to compute the average metric. It counts the total true positives, false negatives, and false positives. It is computed for all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 second, 3 seconds) is shown in Table 19, Table 20, Table 21 and Table 22 for the precision, recall, F1-score, and support.

Table 19. Micro Average for Precision

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	-	-	0.98	-
Linear SVM	0.93	0.80	-	-	0.94	-
SVM	0.96	0.95	-	-	0.96	-
Random Forest	0.99	0.98	-	-	0.99	-
Logistic Regression	0.99	0.99	-	-	0.99	-
Ridge	0.99	0.99	-	-	0.99	-
KNN	0.91	0.93	-	-	0.92	-

Table 20. Micro Average for Recall

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	-	-	0.98	-
Linear SVM	0.93	0.80	-	-	0.94	-
SVM	0.96	0.95	-	-	0.96	-
Random Forest	0.99	0.98	-	-	0.99	-
Logistic Regression	0.99	0.99	-	-	0.99	-
Ridge	0.99	0.99	-	-	0.99	-
KNN	0.91	0.93	-	-	0.92	-

Table 21. Micro Average for F1-score

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	0.97	0.97	0.98	0.97
Linear SVM	0.93	0.80	0.76	0.83	0.94	0.80
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.99	0.95	0.91	0.99	0.95
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.93	0.91	0.90	0.92	0.94

Table 22. Micro Average for Support

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	2247	1122	747	2247	1122	747
Linear SVM	2247	1122	747	2247	1122	747
SVM	2247	1122	747	2247	1122	747
Random Forest	2247	1122	747	2247	1122	747
Logistic Regression	2247	1122	747	2247	1122	747
Ridge	2247	1122	747	2247	1122	747
KNN	2247	1122	747	2247	1122	747

The results show that those common stereotype behaviors recognized with the highest micro average using the Ridge algorithm. The overall macro average is 99 % for hand flapping, head banging, and running back and forth.

3.4.8. Macro Average

The macro average calculates metrics independently for each class and finds the unweighted mean. For all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 second, 3 seconds) is shown in Table 23, Table 24, Table 25 and Table 26 for the precision, recall, F1-score, and support.

Table 23. Macro Average for Precision

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.97	0.97	0.97	0.97	0.97
Linear SVM	0.93	0.85	0.81	0.84	0.94	0.84
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.99	0.99	0.99	0.99	0.98
Logistic Regression	0.99	0.99	0.94	0.91	0.99	0.94
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.93	0.91	0.90	0.92	0.94

Table 24. Macro Average for Recall

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.97	0.97	0.97	0.97	0.97
Linear SVM	0.93	0.82	0.78	0.85	0.94	0.77
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.98	0.98
Logistic Regression	0.99	0.99	0.94	0.91	0.99	0.94
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.90	0.92	0.90	0.90	0.91	0.93

Table 25. Macro Average for F1-score

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.97	0.97	0.97	0.97	0.97
Linear SVM	0.93	0.81	0.76	0.83	0.94	0.75
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.99	0.94	0.91	0.99	0.94
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.92	0.90	0.90	0.91	0.93

Table 26. Macro Average for Support

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	2247	1122	747	2247	1122	747
Linear SVM	2247	1122	747	2247	1122	747
SVM	2247	1122	747	2247	1122	747
Random Forest	2247	1122	747	2247	1122	747
Logistic Regression	2247	1122	747	2247	1122	747
Ridge	2247	1122	747	2247	1122	747
KNN	2247	1122	747	2247	1122	747

The results show that those common stereotype behaviors recognized with the highest macro average using the Ridge machine learning algorithm. The overall macro average is 99 % for hand flapping, head banging, and running back and forth.

3.4.9. Weighted Average

Weighted average calculates the metrics for each label, and find the average weighted by support. For all classifiers through two hyperparameter tuning algorithm (Random and Grid search) for the three window sizes of data (1 second, 2 second, 3 seconds) is shown in Table 27, Table 28, Table 29 and Table 30 for the precision, recall, F1-score, and support.

Table 27. Weighted Average for Precision

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	0.97	0.97	0.98	0.97
Linear SVM	0.94	0.86	0.83	0.86	0.94	0.83
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.99	0.95	0.91	0.99	0.95
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.93	0.91	0.90	0.92	0.94

Table 28. Weighted Average for Recall

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	0.97	0.97	0.98	0.97
Linear SVM	0.93	0.80	0.76	0.83	0.94	0.80
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.99	0.95	0.91	0.99	0.95
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.93	0.91	0.90	0.92	0.94

Table 29. Weighted Average for F1-score

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	0.97	0.98	0.97	0.97	0.98	0.97
Linear SVM	0.93	0.80	0.76	0.82	0.94	0.76
SVM	0.96	0.95	0.96	0.96	0.96	0.94
Random Forest	0.99	0.98	0.99	0.98	0.99	0.98
Logistic Regression	0.99	0.99	0.95	0.91	0.99	0.95
Ridge	0.99	0.99	0.99	0.99	0.99	0.99
KNN	0.91	0.93	0.91	0.90	0.92	0.94

Table 30. Weighted Average for Support

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	2247	1122	747	2247	1122	747
Linear SVM	2247	1122	747	2247	1122	747
SVM	2247	1122	747	2247	1122	747
Random Forest	2247	1122	747	2247	1122	747
Logistic Regression	2247	1122	747	2247	1122	747
Ridge	2247	1122	747	2247	1122	747
KNN	2247	1122	747	2247	1122	747

The results show that those common stereotype behaviors recognized with the highest weighted average using the Ridge algorithm. The overall weighted average is 99 % for hand flapping, head banging, and running back and forth.

3.4.10. Computation Time

The computing time for all classifiers through two hyperparameter tuning algorithms (Random and Grid search) for the three window sizes of data (1 second, 2 seconds, 3 seconds) is shown in Table 31 for training and Table 32 for testing.

Table 31. Computation Time for Training (Minute: second.mili-second)

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	18.31	7.081	09.576	13.340	6.55598	14.956
Linear SVM	01:47.1	31.698	32.883	02:27.1	31.9804	41.775
SVM	54:12.5	07:10.3	01:35.0	3:35:04	15:36.39	00.004
Random Forest	04:31.7	55.849	01:01.6	33:36.6	08:57.35	12:08
Logistic Regression	01:06.1	27.235	06.936	17.308	29.33735	09.154
Ridge	01.642	00.739	01.156	03.171	00.87426	01.531
KNN	16.195	05.524	03.843	30.227	11.23441	09.841

Table 32. Computation Time for Testing (Minute:second.mili-second)

	Random Search			Grid Search		
Algorithm	1 sec	2 sec	3 sec	1 sec	2 sec	3 sec
DT	00.018	00.004	0.0001	00.016	00.00416	00.016
Linear SVM	00.014	00.003	0.0001	0.0001	00.00224	00.004
SVM	00.262	00.066	00.031	00.359	00.06559	00.059
Random Forest	00.358	00.029	00.062	00.031	00.02232	00.109
Logistic Regression	00.009	00.003	00.016	00.078	00.00280	0.0001
Ridge	00.005	00.002	0.0001	00.016	00.00267	0.0001
KNN	00.491	00.148	00.094	00.437	00.15917	00.125

Ridge seems to complete the training and testing in a reasonable short time. Random Forest seems to be a computationally exhaustive for this type of data.

3.4.11. Confusion Matrix

The confusion matrix is a table that allows us the visualize and reports the number of false positives, false negatives, true positives, and true negatives. Figure 2 and Figure 3 show that Ridge classifier managed to match the predicated label and true label with a range between 98% to 100% for the three stereotyped behavior for children with autism. The best confusion matrix chosen is for Ridge classifier, which managed mostly to match the predicated label, and the true label is the one for grid search hyperparameter tuning for a 2-second window of data. The value of alpha, according to Table 4, is 0.6. The rest of the confusion matrices found in the Appendix: confusion matrices section.

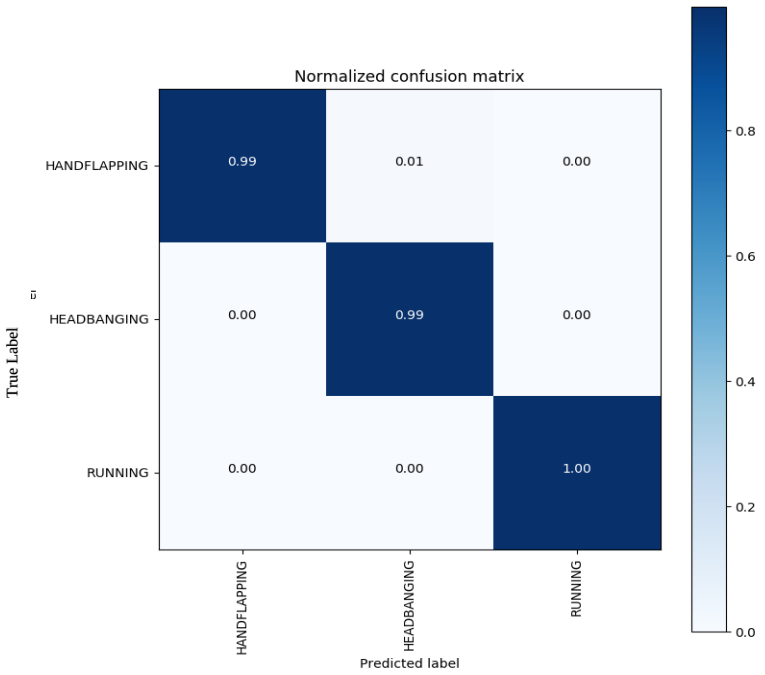

Figure 2. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the ridge algorithm

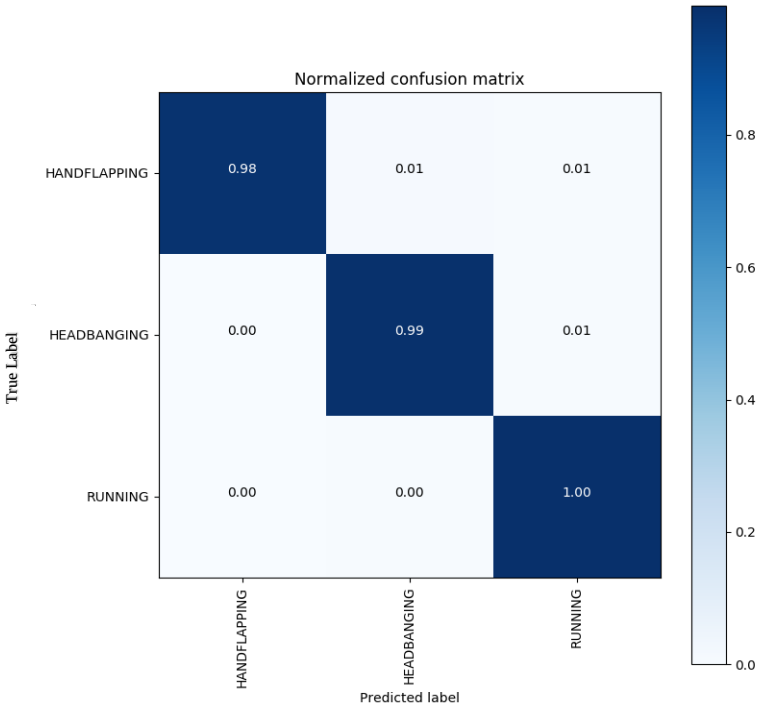

Figure 3. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the ridge algorithm

CHAPTER 4: APPLICATION

4.1. System Overview

Figure 4 shows a real-time application developed on a smartphone (iPhone) to detect the stereotyped behaviors for children who are wearing the smartwatch (Apple watch). The application uses the Swift Core Motion framework and CMMotionManager, the watch we can collect the accelerometer and gyroscope data. The application uses the Swift WatchConnectivity framework to send 160 samples of data to the iPhone from the watch. The iPhone is running a Python Flask webserver hosted by pythonanywhere, which it has the trained classification model and the prediction code. Pythonanywhere website allows a python code to hosted on it with the ability to install the needed libraries or packages. The iPhone receives the data from the watch. Then the iPhone sends the data to the server in the form of an HTTP post. The Flask server receives the data from the iPhone and imports a pickled model (classification trained model) and deserializes it to predict the classification label. It chooses the most frequent label of all predications. Then the label sent from the server as an HTTP response to the iPhone. The iPhone sends the label to the watch, and it displayed on the watch. The pickle operation used to serialize our trained machine learning algorithms and save the serialized format to a model utilized by the prediction code. This operation can eliminate the time spent in retraining the classification model. The classification model chosen is Ridge classifier with a window of 2 seconds and alpha 0.6.

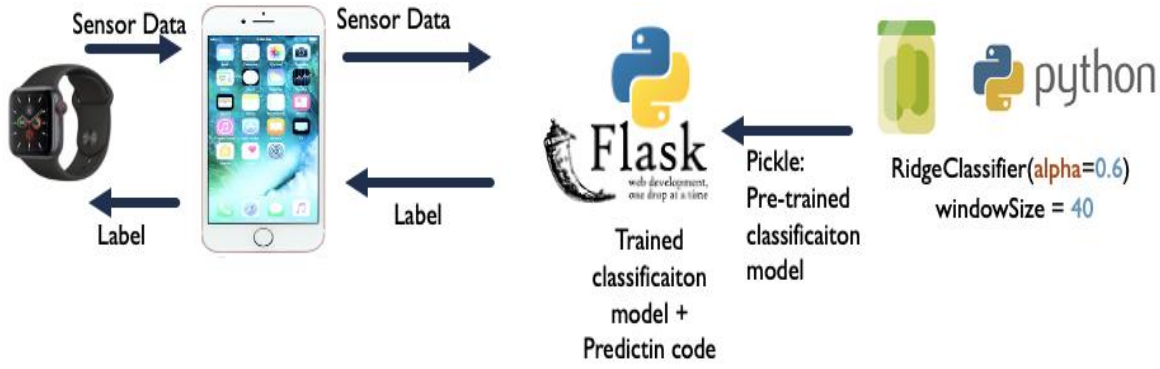

Figure 4. Recognizing stereotyped behavior in children with autism system overview

4.2. System Limitation

The application is limited to be developed as an IOS application since the smartwatch can only communicate with an iPhone. Also, this application is limited to the watch's framework sensors and methods. Also, the watch must be near the iPhone so it can send the data to the iPhone. Moreover, hosting on pythonanywhere as Free service limited the application by service CPU allowance is 100 seconds, 512MB storage, and low bandwidth. Also, since we are using a web server, an internet connection is needed for this application to work.

4.3. System Testing

The user press on the "Start Recording" button on the watch, as shown in Figure 5. The watch collects the sensor data, which printed on the watch console in Figure 6. The watch sends the sensors data to the iPhone, which printed on the iPhone console in Figure 7. The iPhone sends the data as an HTTP post to the Flask server, which uses the data to predict the label. The sensor data received by the server, and the predicted label shown in the server log in Figure 8. The label 2 indicates the second behavior, which is Head Banging. This label sent as an HTTP response to the iPhone, which displayed in Figure 9. The iPhone sends this label to the watch, which printed on the watch console in Figure 10. The watch displays the label "Head Banging" in

Figure 11. The system tested on a child who was not part of the training of the data for the three behaviuors. The demonstration video of the system's test found in [46].

Figure 5. The watch user interface before recording the sensor data

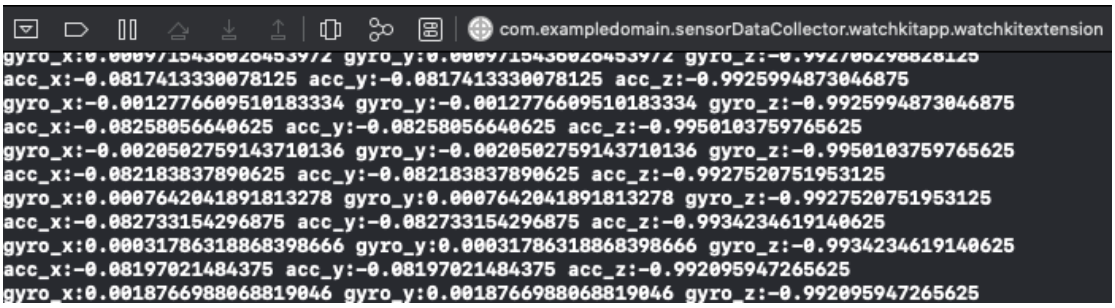

Figure 6. The watch console while recording the sensor data

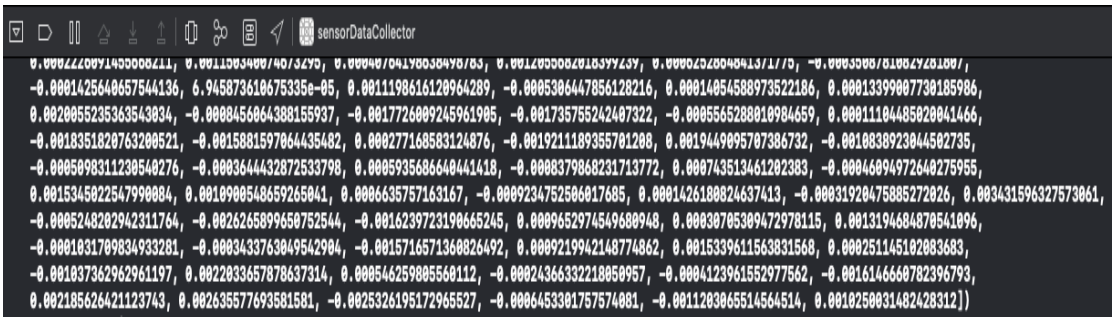

Figure 7. The iPhone console after receiving the sensor data

```
0-04-28 22:25:38 -0.006378173828125 -0.082916259765625 -0.9925384521484375 0.0002668581437319517 0.0002981983125209808 3 0.0022033657878637314
0-04-28 22:25:38 -0.00555419921875 -0.0839996337890625 -0.992889404296875 -0.0020964241120964 29 -0.0009360415861010551 0.000546259805560112
0-04-28 22:25:38 -0.005401611328125 -0.0837860107421875 -0.993499755859375 0.0010160997044295073 0.0013455050066113472 -0.0002436633221805095 7
0-04-28 22:25:38 -0.0052490234375 -0.0831451416015625 -0.9929046630859375 -0.0002543579321354627 6 0.0022009657695889473 -0.0004123961552977562
0-04-28 22:25:38 -0.006072998046875 -0.0833892822265625 -0.9946441650390625 -0.001238482422195375 -0.0008311020210385323 -0.0016146660782396793
0-04-28 22:25:38 -0.0052032470703125 -0.0837860107421875 -0.992706298828125 -2.8128037229180336e-05 0.0009715436026453972 0.002185626421123743
0-04-28 22:25:38 -0.0045166015625 -0.0817413330078125 -0.9925994873046875 -0.0011963089928030968 -0.0012776609510183334 0.002635577693581581
0-04-28 22:25:38 -0.0044708251953125 -0.08258056640625 -0.9950103759765625 0.002755514346063137 -0.0020502759143710136 -0.0025326195172965527
0-04-28 22:25:38 -0.0047760009765625 -0.082183837890625 -0.9927520751953125 0.0013547784183174372 0.0007642041891813278 -0.0006453301757574081
0-04-28 22:25:38 -0.0045776367187 5 -0.082733154296875 -0.9934234619140625 0.0008851997554302216 0.0003178631886839866 6 -0.0011203065514564514
0-04-28 22:25:38 -0.0051727294921875 -0.08197021484375 -0.992095947265625 -0.0007075334433466196 0.0018766988068819046 0.0010250031482428312
0-04-28 22:25:39 ......................
0-04-28 22:25:39 .........Final Label..............
0-04-28 22:25:39 2.0
0-04-28 22:25:39 ......................
```

Figure 8. The server log after predicting the label

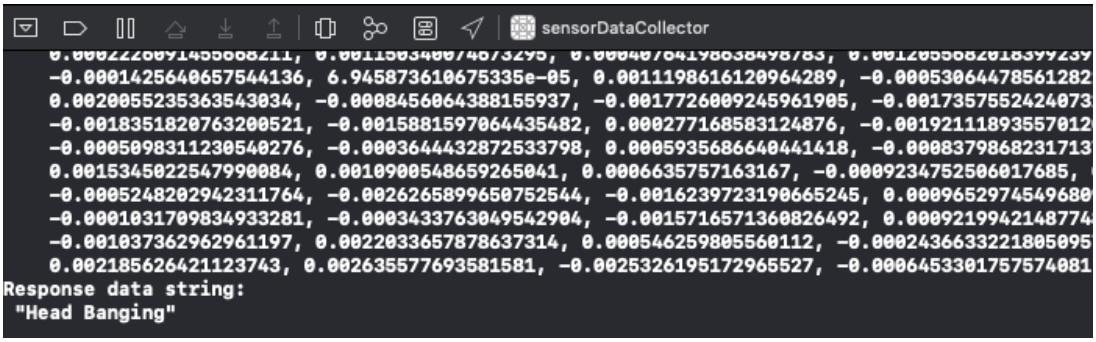

Figure 9. The iPhone console after receiving the label

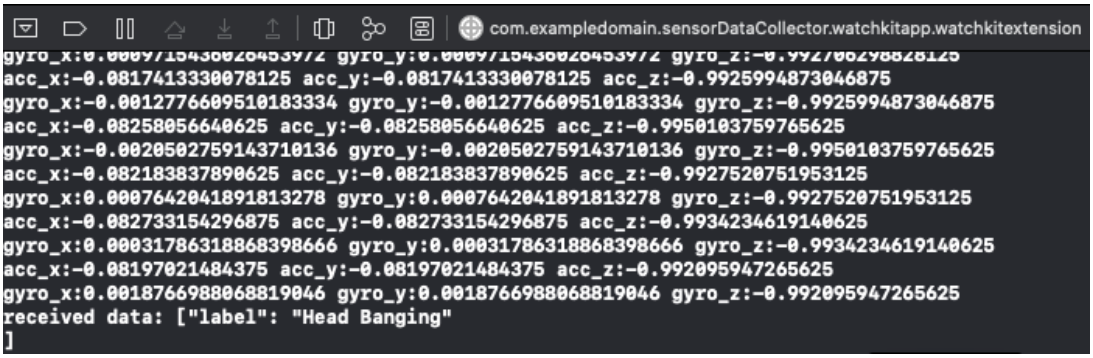

Figure 10. The watch console after receiving the label

Figure 11. The watch user interface after receiving the label

4.4. Implementation Attempts

Several implementation trails tried out before the implementation discussed in sections 4.1, 4.2, and 4.3. The first implementation we tried is to use Apple's Core ML tools to convert our Python classifier into a classifier that could be understood by the Swift language. This implementation did not work because of the feature engineering process, where the number of features increased from 6 inputted features (sensor data) to 316 features. The principle of custom scikit-learn pipelines used to produce the feature and finding the label inside a pipeline, and it worked in producing the label, but Core ML conversion tools did not support it. The second implementation method is using Kivy-IOS in which everything in Python, and it converts it into a format that the IOS understands it. However, when the code is exported and tested on IOS, it showed that Kivy-IOS does not support needed python packages such as Pandas and Scipy. The third implementation method is using the BeeWare project, which had a similar issue to the Kivy-IOS.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1. Conclusion

This project works on helping in recognizing autistic children's stereotyped behaviors. The recognition achieved by building a signal processing model that collects data from a smartwatch equipped with a gyroscope and accelerometer. Then those data were processed and a feature vector of 316 features produced. Then a new feature vector with its labels was fed to a classification machine learning models for training the algorithms to produce the final predictive model. The best predictive model is the Ridge classifier. The results show that those common stereotype behaviors could be recognized using the Ridge machine learning algorithm with overall average accuracy ranges between 98.7% to 99.5 %. For hand flapping, head banging, and running back and forth, the overall precision ranges between 98% to 100 %, recall ranges between 98% to 100 %, overall F1-score ranges between 98% to 100 %, and overall macro, weighted and micro averages of 99 %. This Ridge classifier used to implement a real-time application developed on a smartphone (iPhone) to detect the stereotyped behaviors for autistic children who are wearing the smartwatch (Apple watch). This application used to build an application to aid in diagnosing the process of autism on children.

5.2. Future Work Directions

The data were collected from accelerometer and gyroscope sensors only, so it possible to combine another type of sensors. Also, collecting data for more stereotyped behaviors of autistic children could be added to this project. Another possible future work for this project would be to collect more data from more volunteers and apply neural networks. Also, it is possible to use feature ranking methods such as information gain and dimension reduction methods such as principal

component analysis. These methods help in choosing which features are essential to the machine learning process and which features discarded.

5.3. Challenges

When we started with this project, there was no obvious way to relate the data collected from the sensor and the feature extracted from them to a specific body movement. So, feature engineering was a challenging task, and it took many papers readings and experimentation in order to produce a proper dataset that can apply machine learning algorithms on it. Acquiring an adequate number of participants in this experiment was challenging, and it took more time than expected because many parents refused to allow their children to participate. Even though it is safe to use the smartwatch and their instructor supervises the process of collecting the data. Also, we had to wait until all the children performed the three stereotyped behavior to start producing the final dataset after preprocessing the data.

REFERENCES

[1] Masoud, M., Jaradat, Y., Manasrah, A. and Jannoud, I.

Sensors of Smart Devices in the Internet of Everything (IoE) Era: Big Opportunities and Massive Doubts

[2] Saha, J. (2019). Do Smartwatches Have Built-In Sensors? [Complete Guide 2019]. Retrieved 11 March 2020, from https://smartwatchzone.in/do-smartwatches-have-built-in-sensors/

[3] Accelerometer vs Gyroscope sensor, and IMU, how to pick one?. (2020). Retrieved 11 March 2020, from https://www.seeedstudio.com/blog/2019/12/24/what-is-accelerometer-gyroscope-and-how-to-pick-one/

[4] Goodrich, R. (2020). Accelerometer vs. Gyroscope: What's the Difference?. Retrieved 11 March 2020, from https://www.livescience.com/40103-accelerometer-vs-gyroscope.html

[5] What Is Autism? | Autism Speaks. (2020). Retrieved 11 March 2020, from https://www.autismspeaks.org/what-autism

[6] Autism Statistics and Facts | Autism Speaks. (2020). Retrieved 11 March 2020, from https://www.autismspeaks.org/autism-facts-and-figures

[7] Hospital, M. (2020). 30 Facts to Know about Autism Spectrum Disorder. Retrieved 5 March 2020, from https://www.massgeneral.org/children/autism/lurie-center/30-facts-to-know-about-autism-spectrum-disorder

[8] Rojahn, J., & Sisson, L. (1990). Stereotyped Behavior. In J. Rojahn & L. Sisson, *Handbook of Behavior Modification with the Mentally Retarde* (pp. 181-182). Boston, MA: Springer, Boston, MA.

[9] Gupta, S., & Kumar, A. (2015). Human Activity Recognition through

Smartphone's Tri-Axial Accelerometer using Time Domain Wave Analysis and Machine Learning. *International Journal Of Computer Applications*, *127*(18), 22-26. doi: 10.5120/ijca2015906733

[10] Pham, C. (2015). MobiRAR: Real-Time Human Activity Recognition Using Mobile Devices. *2015 Seventh International Conference On Knowledge And Systems Engineering (KSE)*. doi: 10.1109/kse.2015.43

[11] Liu, Y., Nie, L., Liu, L., & Rosenblum, D. (2016). From action to activity: Sensor-based activity recognition. *Neurocomputing*, *181*, 108-115. doi: 10.1016/j.neucom.2015.08.096

[12] Balli, S., & Sağbas, E. (2017). The Usage of Statistical Learning Methods on Wearable Devices and a Case Study: Activity Recognition on Smartwatches. *Advances In Statistical Methodologies And Their Application To Real Problems*. doi: 10.5772/66213

[13] Lee, Y., & Song, M. (2017). Using a Smartwatch to Detect Stereotyped Movements in Children With Developmental Disabilities. *IEEE Access*, *5*, 5506-5514. doi: 10.1109/access.2017.2689067

[14] Goncalves, N., Rodrigues, J., Costa, S., & Soares, F. (2012). Automatic detection of stereotyped hand flapping movements: Two different approaches. *2012 IEEE RO-MAN: The 21St IEEE International Symposium On Robot And Human Interactive Communication*. doi: 10.1109/roman.2012.6343784

[15] Ghiassian, S., Greiner, R., Jin, P., & Brown, M. (2016). Using Functional or Structural Magnetic Resonance Images and Personal Characteristic Data to Identify ADHD and Autism. *PLOS ONE*, *11*(12), e0166934. doi: 10.1371/journal.pone.0166934

[16] Amiri, A., Peltier, N., Goldberg, C., Sun, Y., Nathan, A., Hiremath, S., &

Mankodiya, K. (2017). WearSense: Detecting Autism Stereotypic Behaviors through Smartwatches. *Healthcare*, *5*(1), 11. doi: 10.3390/healthcare5010011

[17] 1.12. Multiclass and multilabel algorithms — scikit-learn 0.22.2 documentation. (2020). Retrieved 1 March 2020, from https://scikit-learn.org/stable/modules/multiclass.html

[18] Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference On Machine Learning, Big Data, Cloud And Parallel Computing (Comitcon)*. doi: 10.1109/comitcon.2019.8862451

[19] Gupta, P. (2017). Decision Trees in Machine Learning. Retrieved 4 March 2020, from https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[20] Thomas, B. SensorLog. Retrieved 11 March 2020, from https://apps.apple.com/us/app/sensorlog/id388014573

[21] Irvine, T. (2012). *AN INTRODUCTION TO THE SHOCK RESPONSE SPECTRUM* [Ebook]. Retrieved from http://www.vibrationdata.com/tutorials2/srs_intr.pdf

[22] Texas Instruments. (2004). *AN-236 An Introduction to the Sampling Theorem* [Ebook]. Retrieved from http://www.ti.com/lit/an/snaa079c/snaa079c.pdf

[23] Jagos, H., Oberzaucher, J., Reichel, M., Zagler, W., & Hlauschek, W. (2010). A multimodal approach for insole motion measurement and analysis. *Procedia Engineering*, *2*(2), 3103-3108. doi: 10.1016/j.proeng.2010.04.118

[24] Brownlee, J. (2018). Gentle Introduction to Vector Norms in Machine Learning. Retrieved 6 March 2020, from https://machinelearningmastery.com/vector-norms-machine-learning/

[25] Hamalainen, W., Jarvinen, M., Martiskainen, P., & Mononen, J. (2011). Jerk-

based feature extraction for robust activity recognition from acceleration data. *2011 11Th International Conference On Intelligent Systems Design And Applications*. doi: 10.1109/isda.2011.6121760

[26] The Effects of Outliers. Retrieved 3 March 2020, from http://www.statisticslectures.com/topics/outliereffects/

[27] Kobayashi, T., Hasida, K., & Otsu, N. (2011). Rotation invariant feature extraction from 3-D acceleration signals. *2011 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP)*. doi: 10.1109/icassp.2011.5947150

[28] Zhong, Y., & Deng, Y. (2014). Sensor orientation invariant mobile gait biometrics. *IEEE International Joint Conference On Biometrics*. doi: 10.1109/btas.2014.6996246

[29] Singha, T.B., Nath, R.K., & Narsimhadhan, A.V. (2017). Person Recognition using Smartphones' Accelerometer Data. *ArXiv, abs/1711.04689.*

[30] Quinian, R. (1993). *C4.5 programs for machine learning*. [Place of publication not identified]: Morgan Kaufmann.

[31] S 1.4. Support Vector Machines — scikit-learn 0.22.2 documentation. Retrieved 2 March 2020, from https://scikit-learn.org/stable/modules/svm.html

[32] Team, D. Kernel Functions-Introduction to SVM Kernel & Examples - DataFlair. Retrieved 6 March 2020, from https://data-flair.training/blogs/svm-kernel-functions/

[33] Stamp, M. (2018). A Survey of Machine Learning Algorithms and Their Application in Information Security. *Computer Communications And Networks*, 33-55. doi: 10.1007/978-3-319-92624-7_2

[34] 1.1. Linear Models — scikit-learn 0.22.2 documentation. Retrieved 2 March 2020, from https://scikit-learn.org/stable/modules/linear_model.html#ridge-

regression-and-classification

[35] Brownlee, J. (2017). What is the Difference Between a Parameter and a Hyperparameter?. Retrieved 5 March 2020, from https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/

[36] Statistical Language - Measures of Central Tendency. Retrieved 27 February 2020, from https://www.abs.gov.au/websitedbs/a3121120.nsf/home/statistical+language+-+measures+of+central+tendency

[37] Interquartile Range: Definition. Retrieved 27 February 2020, from https://stattrek.com/statistics/dictionary.aspx?definition=interquartile%20range

[38] Entropy in machine learning - From physics to data analytics. (2019). Retrieved 22 February 2020, from https://amethix.com/entropy-in-machine-learning/

[39] Calculate Signal Magnitude Area of an accelerometer in Matlab - MATLAB Answers - MATLAB Central. (2014). Retrieved 23 February 2020, from https://www.mathworks.com/matlabcentral/answers/142765-calculate-signal-magnitude-area-of-an-accelerometer-in-matlab

[40] Khusainov, R., Azzi, D., Achumba, I., & Bersch, S. (2013). Real-Time Human Ambulation, Activity, and Physiological Monitoring: Taxonomy of Issues, Techniques, Applications, Challenges and Limitations. *Sensors*, *13*(10), 12852-12902. doi: 10.3390/s131012852

[41] Bronshtein, A. (2017). Train/Test Split and Cross Validation in Python. Retrieved 25 February 2020, from https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

[42] Farkas, I.I., & Doran, E. (2011). ACTIVITY RECOGNITION FROM

ACCELERATION DATA COLLECTED WITH A TRI-AXIAL ACCELEROMETER.

[43] Cheng, L., Leung, A., & Ozawa, S. (2018). *Neural information processing* (p. 309). Springer.

[44] Goodwin, M., Haghighi, M., Tang, Q., Akcakaya, M., Erdogmus, D., & Intille, S. (2014). Moving towards a real-time system for automatically recognizing stereotypical motor movements in individuals on the autism spectrum using wireless accelerometry. *Proceedings Of The 2014 ACM International Joint Conference On Pervasive And Ubiquitous Computing - Ubicomp '14 Adjunct*. doi: 10.1145/2632048.2632096

[45] Brown, S. (2016). Measures of Shape: Skewness and Kurtosis. Retrieved 13 March 2020, from https://brownmath.com/stat/shape.htm

[46] https://www.youtube.com/watch?v=4fzF5SkyQro&feature=youtu.be

APPENDIX A: CONFUSION MATRICES

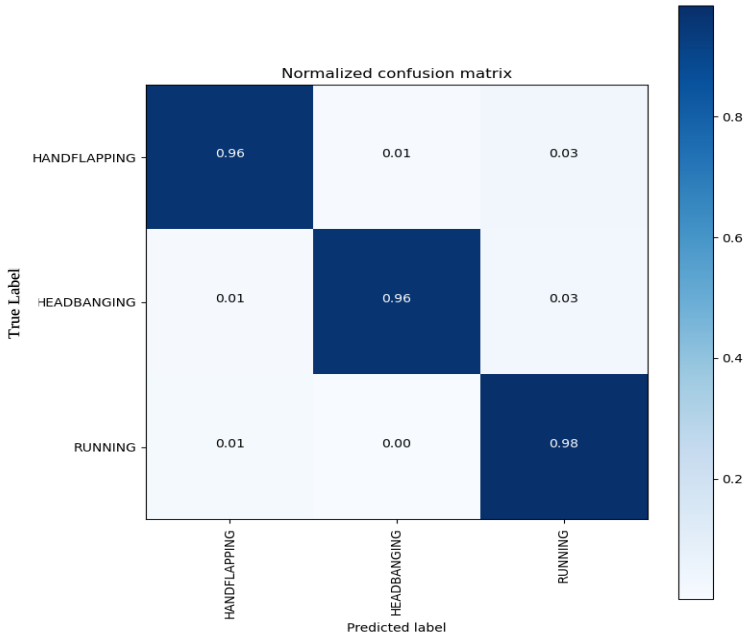

Figure 12. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the DT algorithm

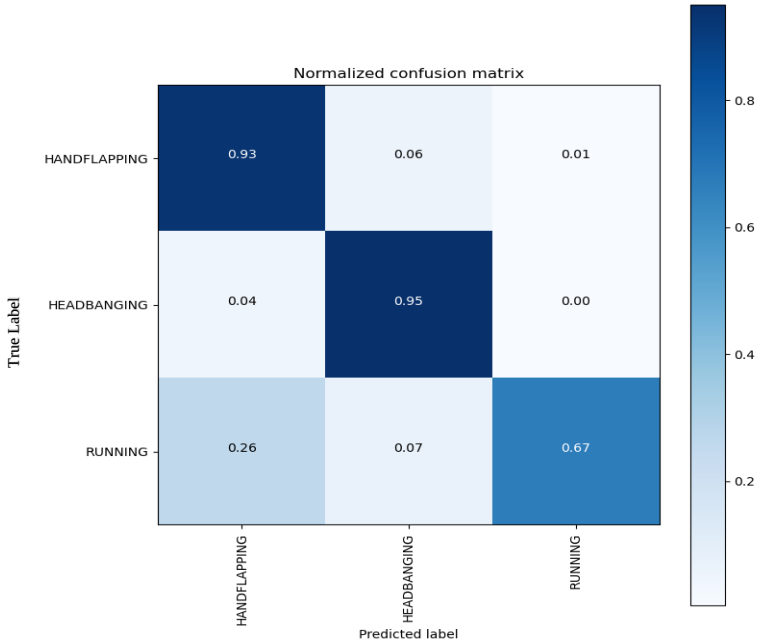

Figure 13. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the Linear SVM algorithm

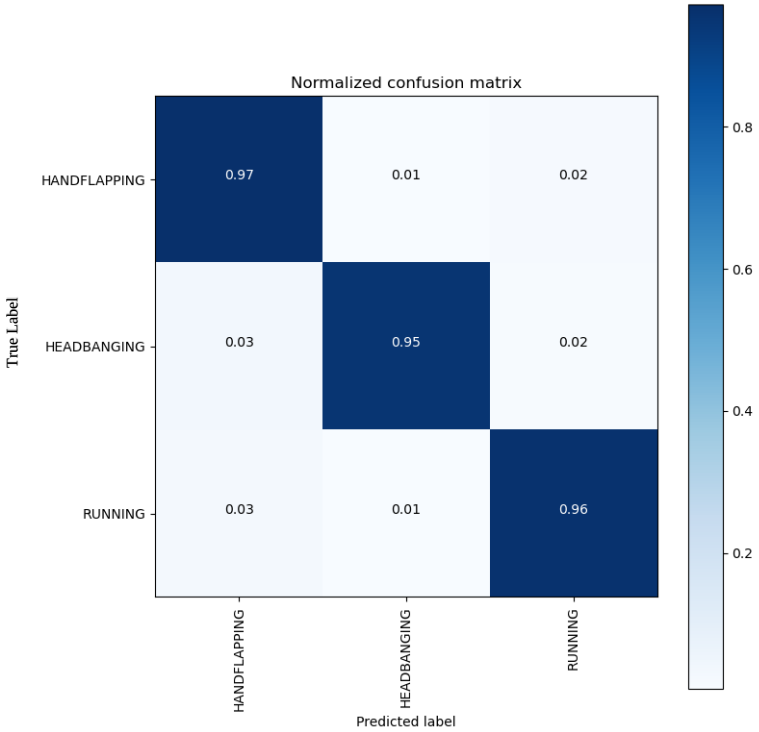

Figure 14. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the SVM algorithm

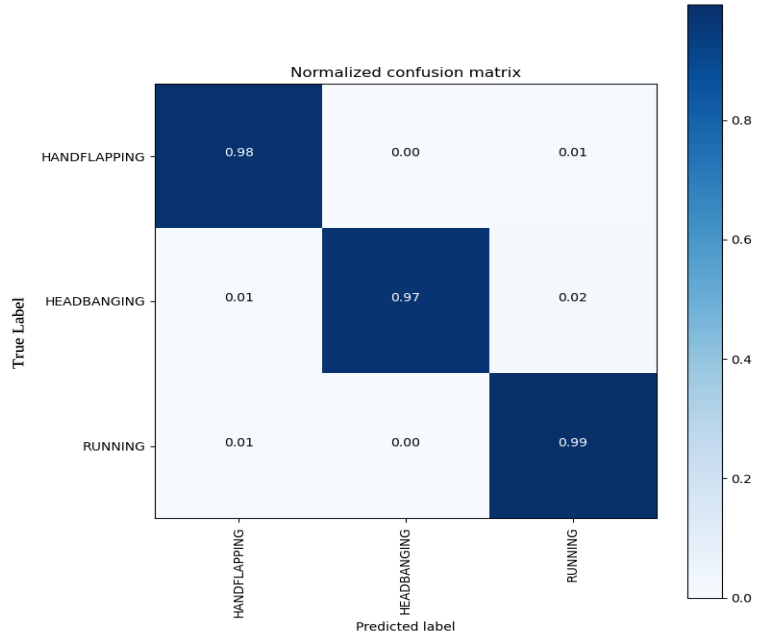

Figure 15. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the Random Forest algorithm

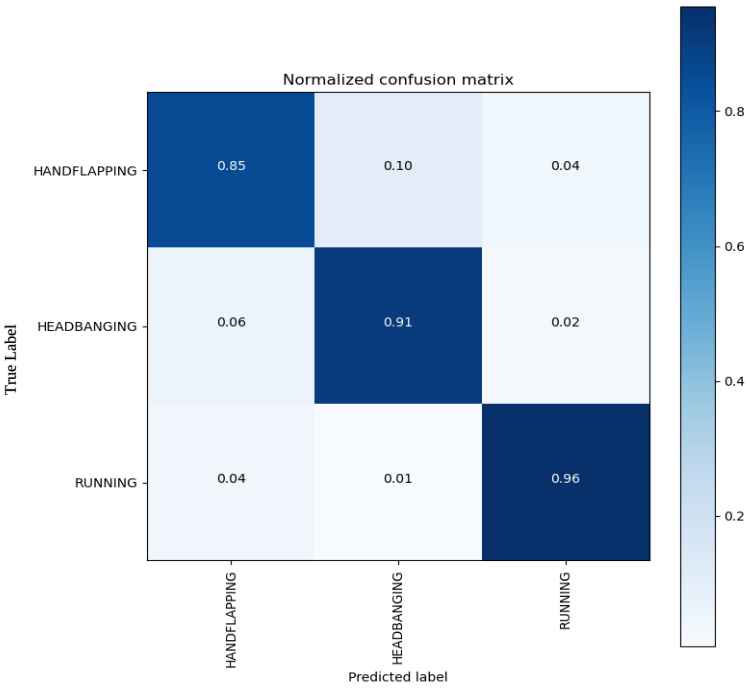

Figure 16. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the Logistic Regression algorithm

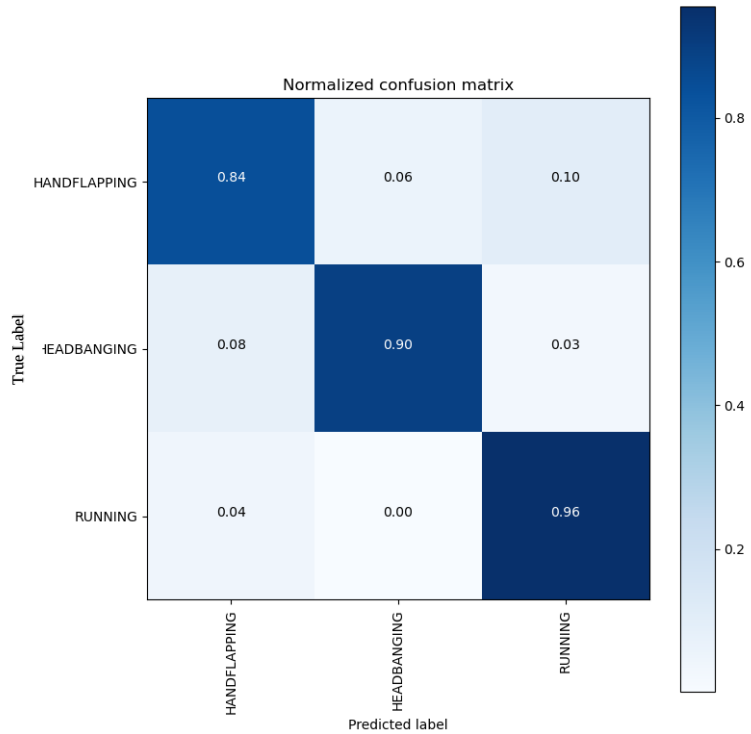

Figure 17. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the KNN algorithm

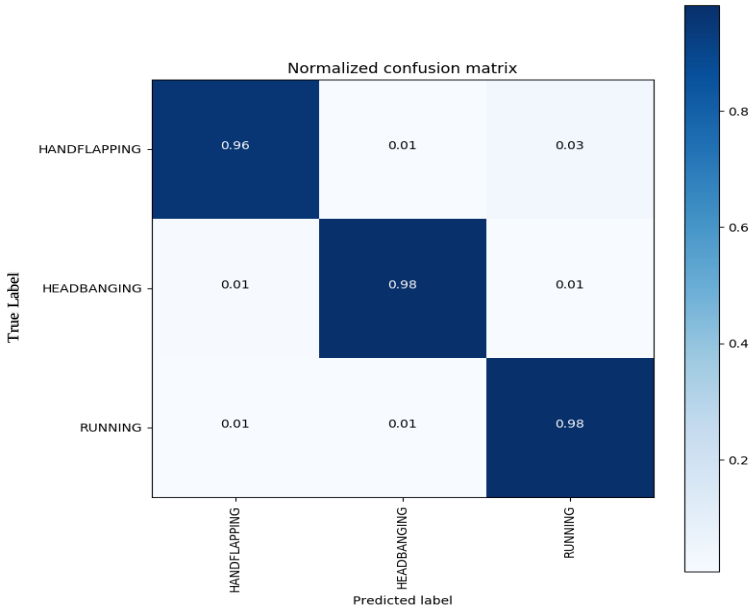

Figure 18. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the DT algorithm

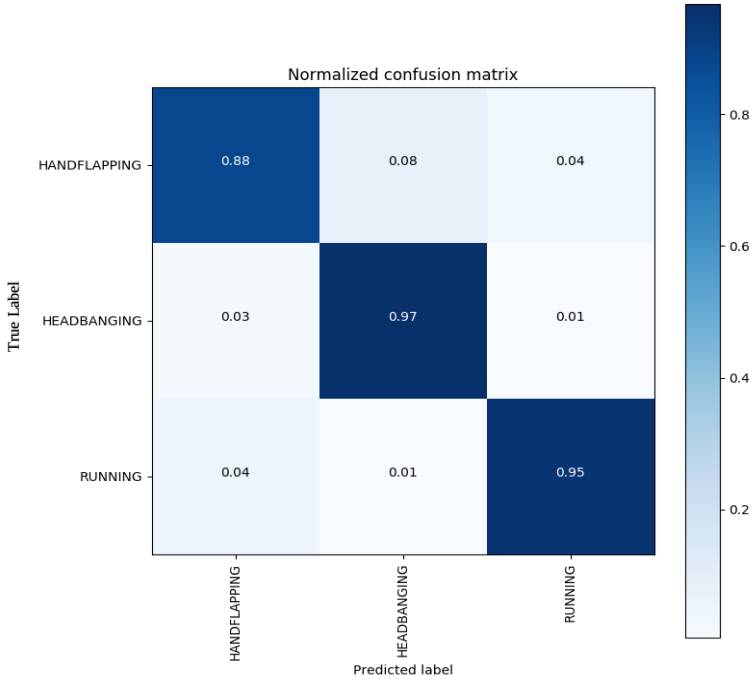

Figure 19. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the Linear SVM algorithm

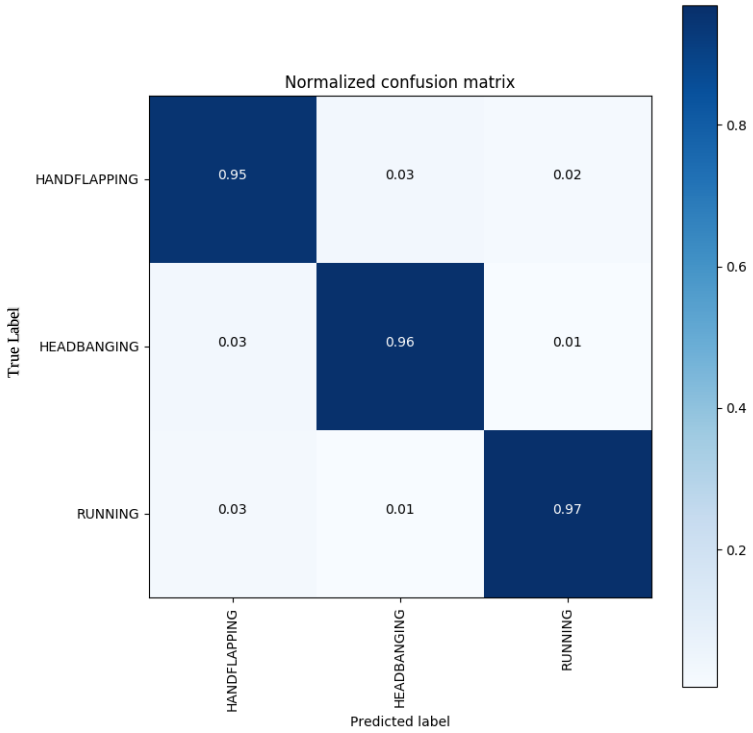

Figure 20. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the SVM algorithm

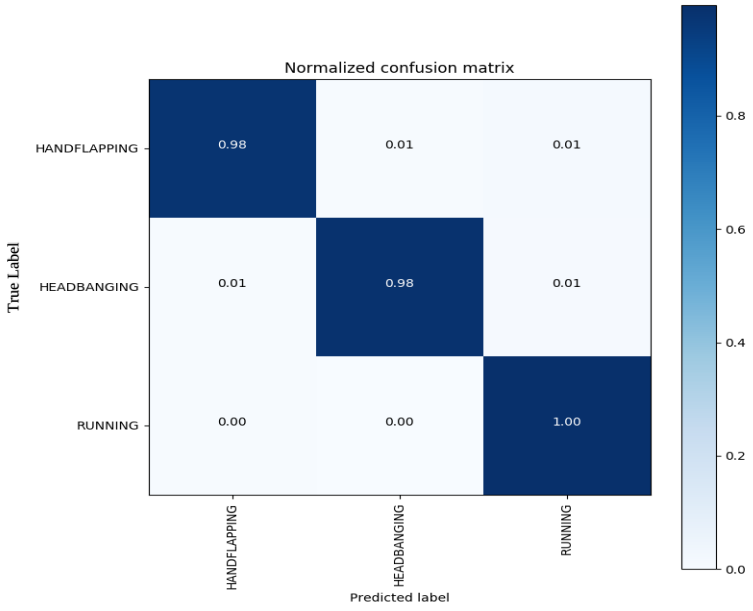

Figure 21. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the Random Forest algorithm

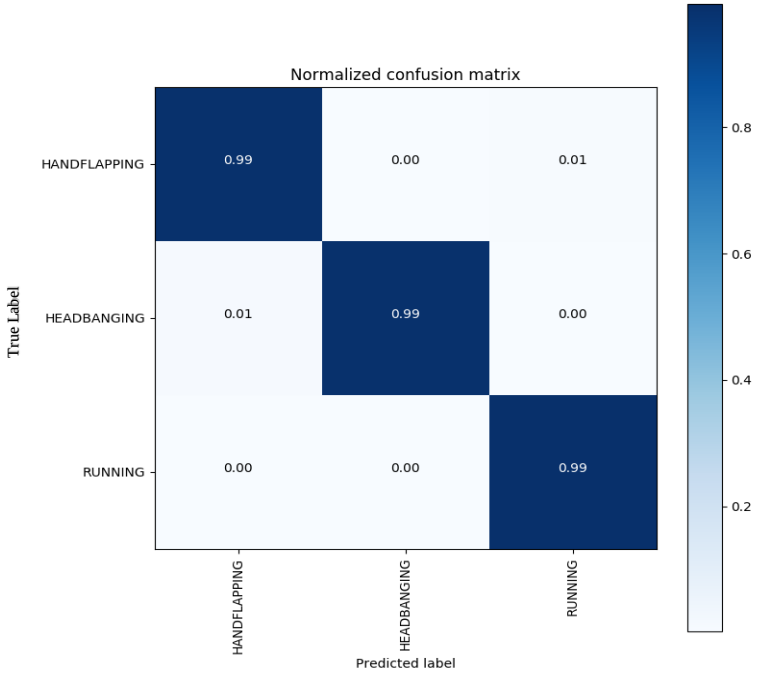

Figure 22. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the Logistic Regression algorithm

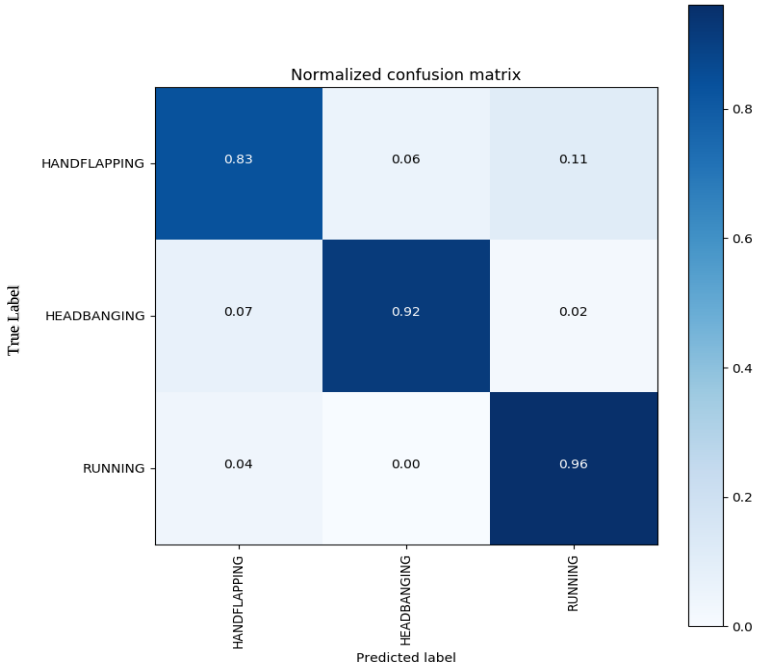

Figure 23. Confusion matrix for random search hyperparameter tuning for a window of 1-second data computed in the KNN algorithm

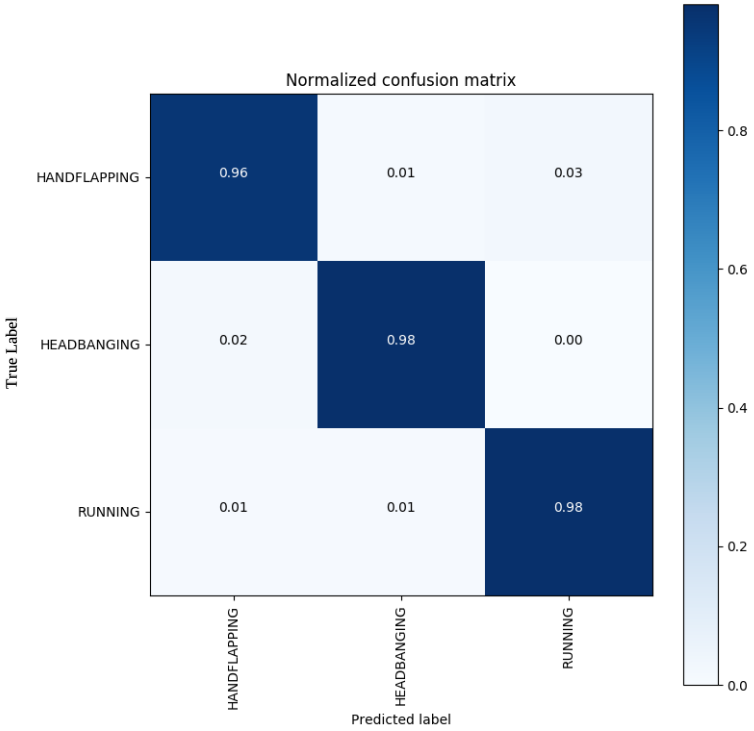

Figure 24. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the DT algorithm

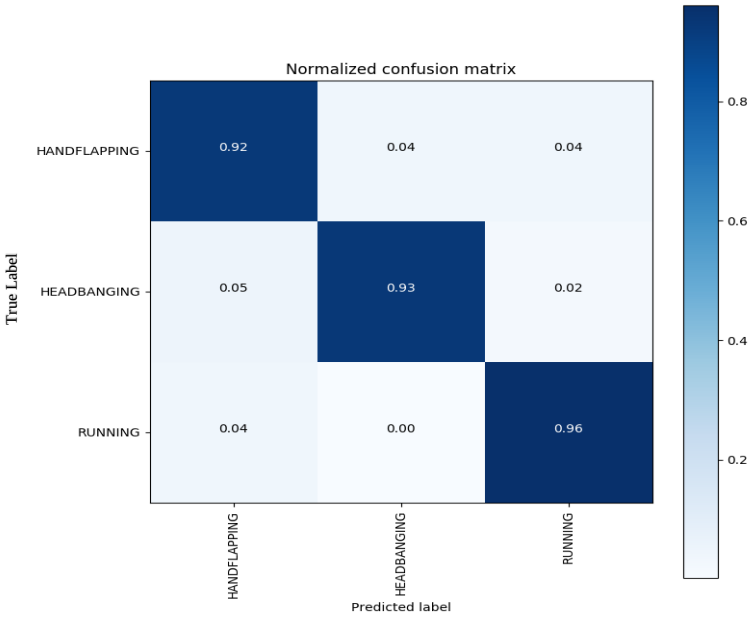

Figure 25. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the Linear SVM algorithm

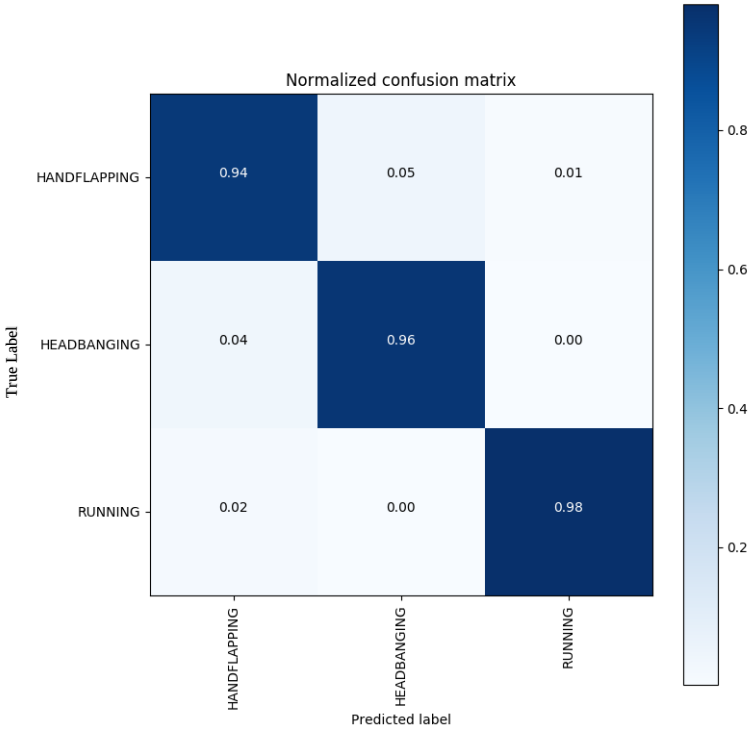

Figure 26. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the SVM algorithm

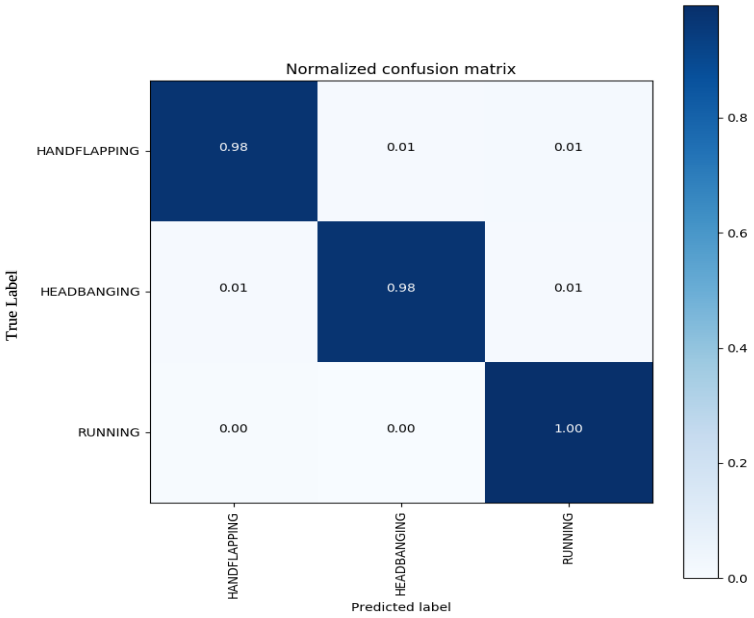

Figure 27. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the Random Forest algorithm

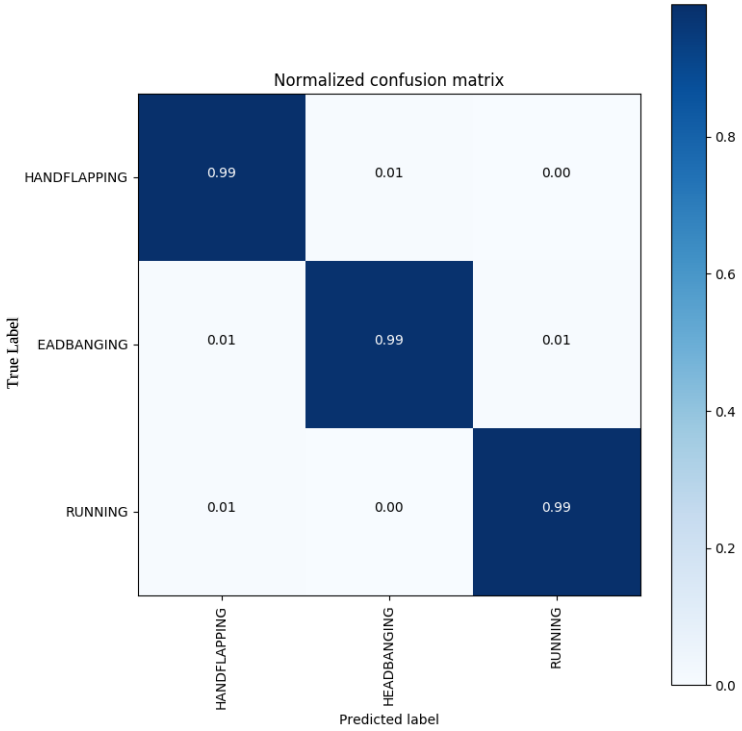

Figure 28. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the Logistic Regression algorithm

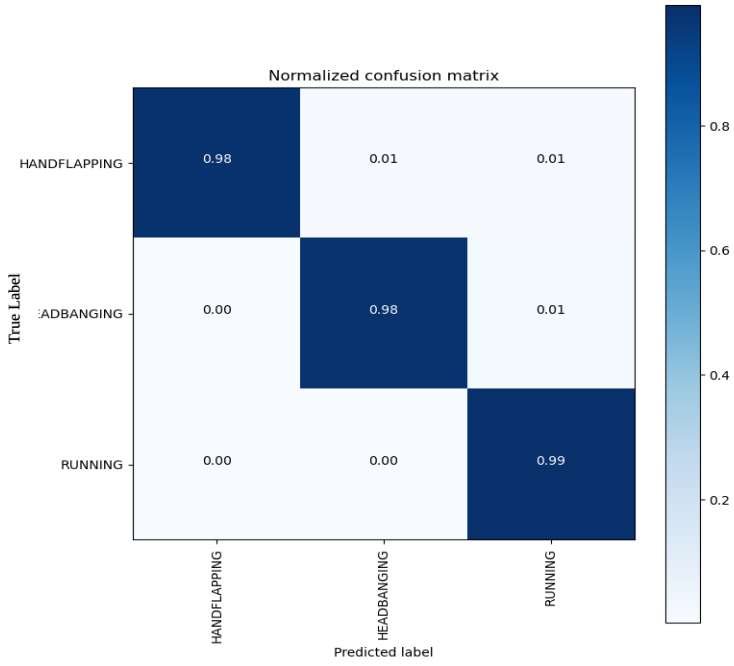

Figure 29. Confusion matrix for grid search hyperparameter tuning for a window of 1-second data computed in the Ridge algorithm

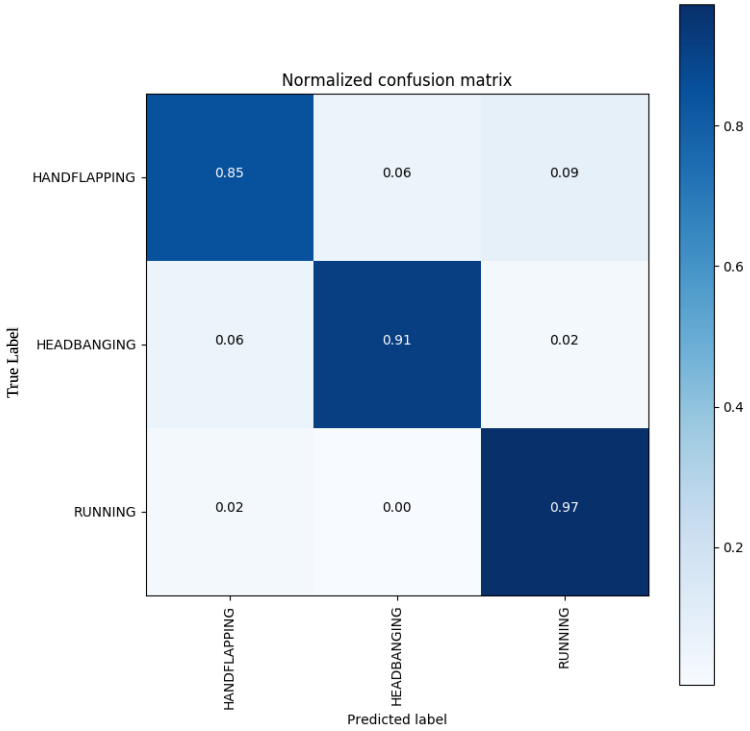

Figure 30. Confusion matrix for grid search hyperparameter tuning for a window of 2-second data computed in the KNN algorithm

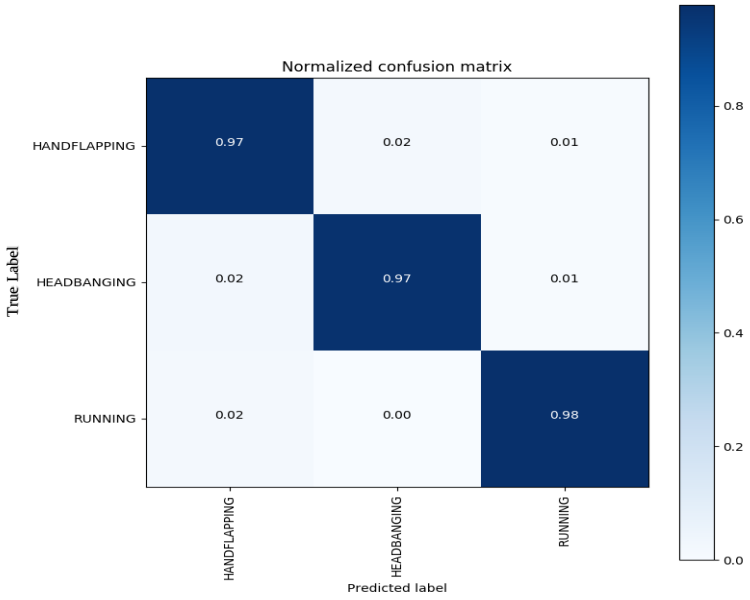

Figure 31. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the DT algorithm

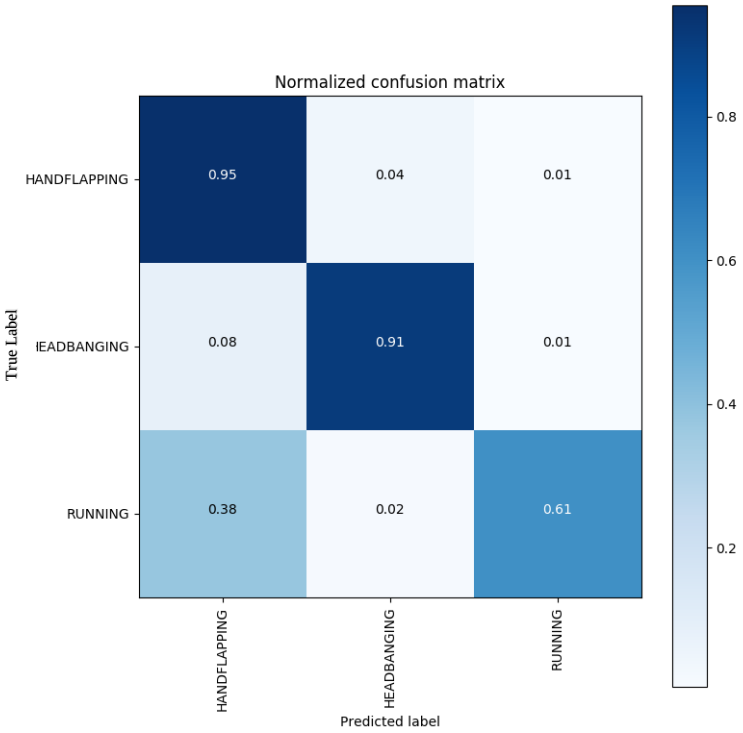

Figure 32. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the Linear SVM algorithm

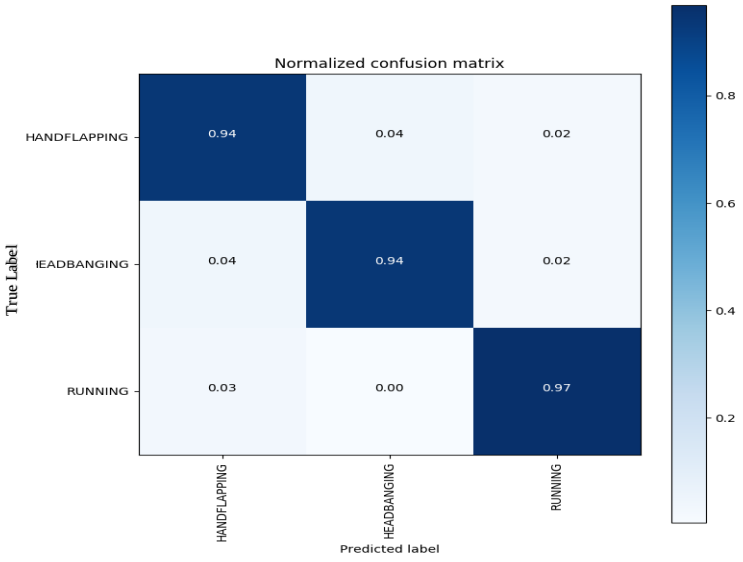

Figure 33. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the SVM algorithm

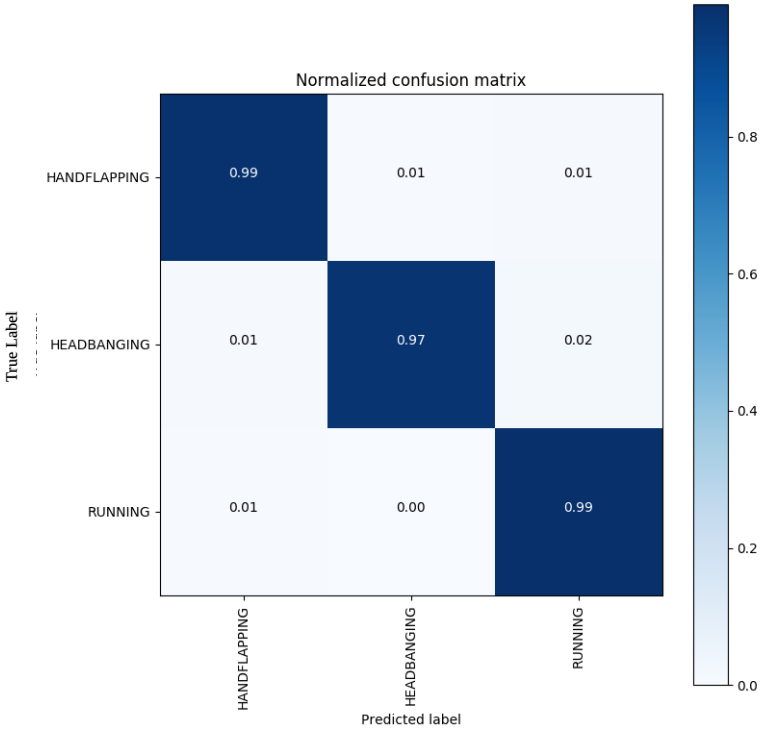

Figure 34. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the Random Forest algorithm

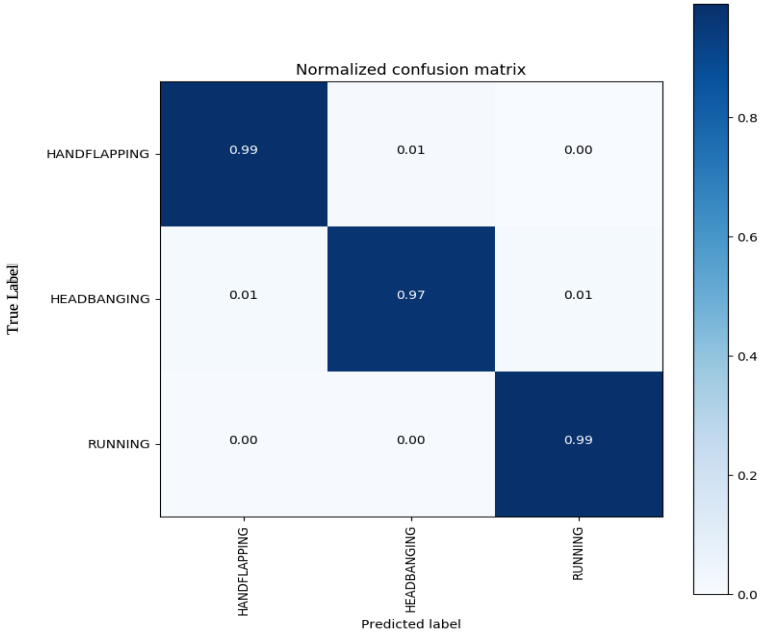

Figure 35. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the Logistic Regression algorithm

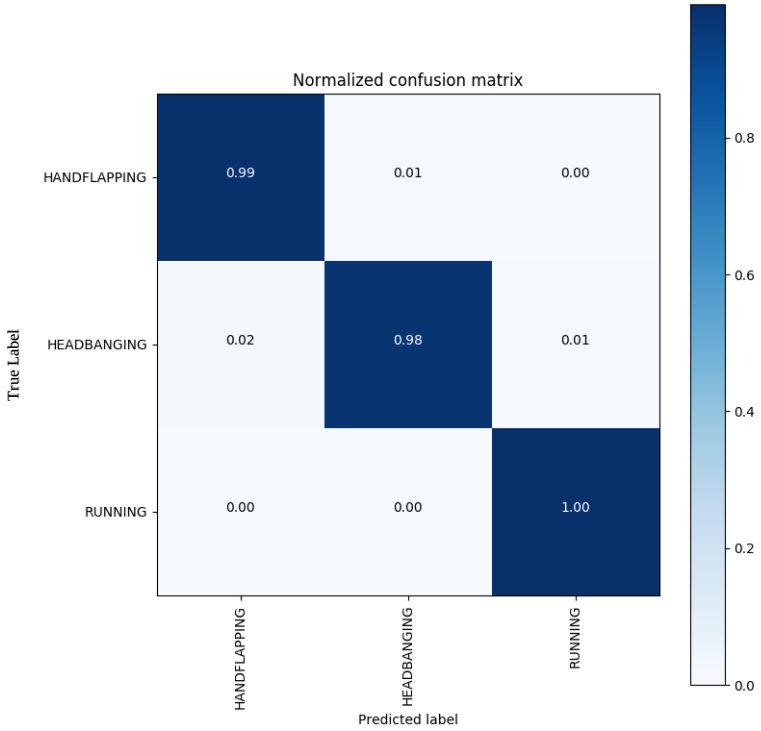

Figure 36. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the Ridge algorithm

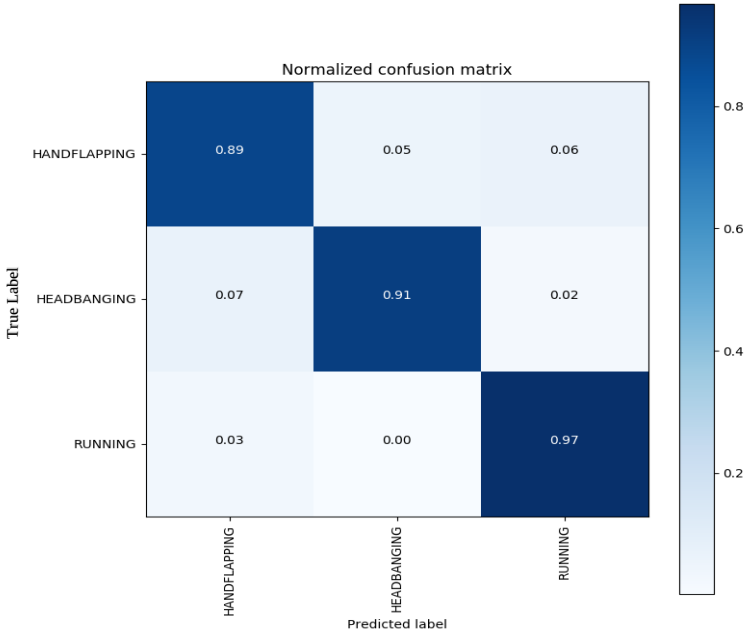

Figure 37. Confusion matrix for random search hyperparameter tuning for a window of 2-second data computed in the KNN algorithm

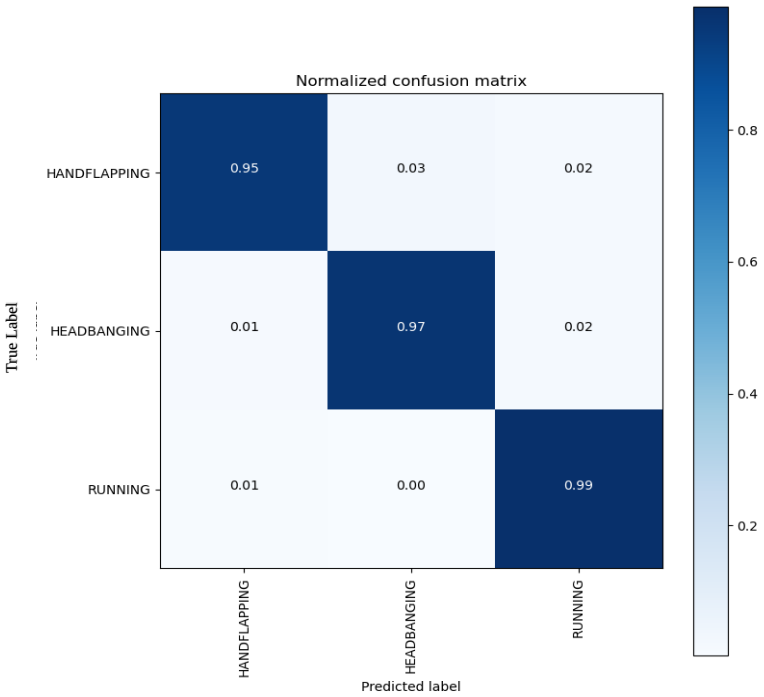

Figure 38. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the DT algorithm

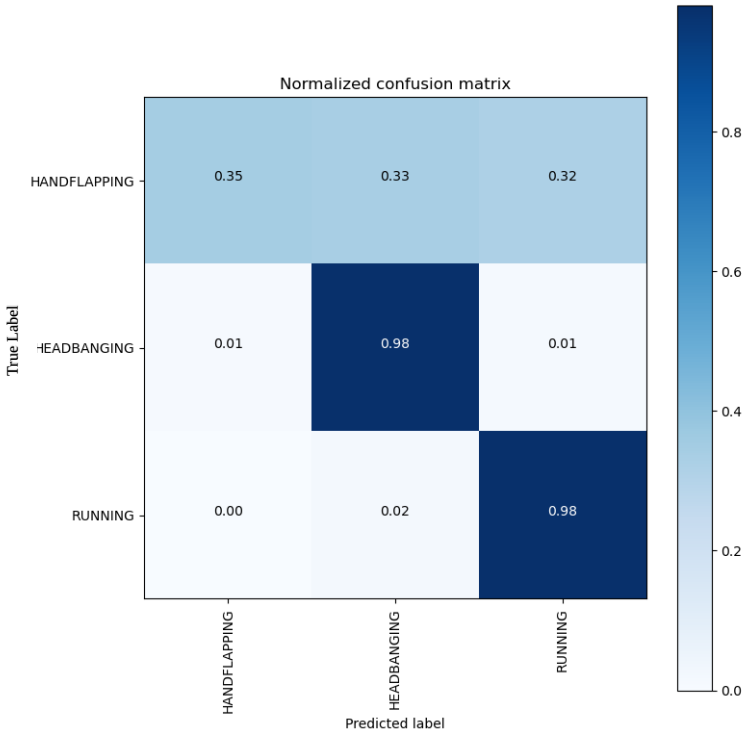

Figure 39. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the Linear SVM algorithm

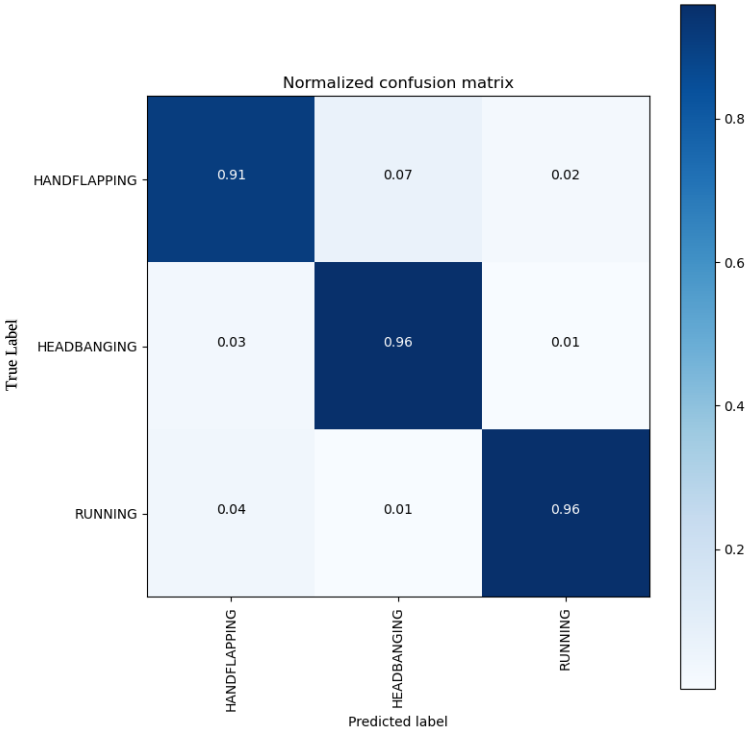

Figure 40. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the SVM algorithm

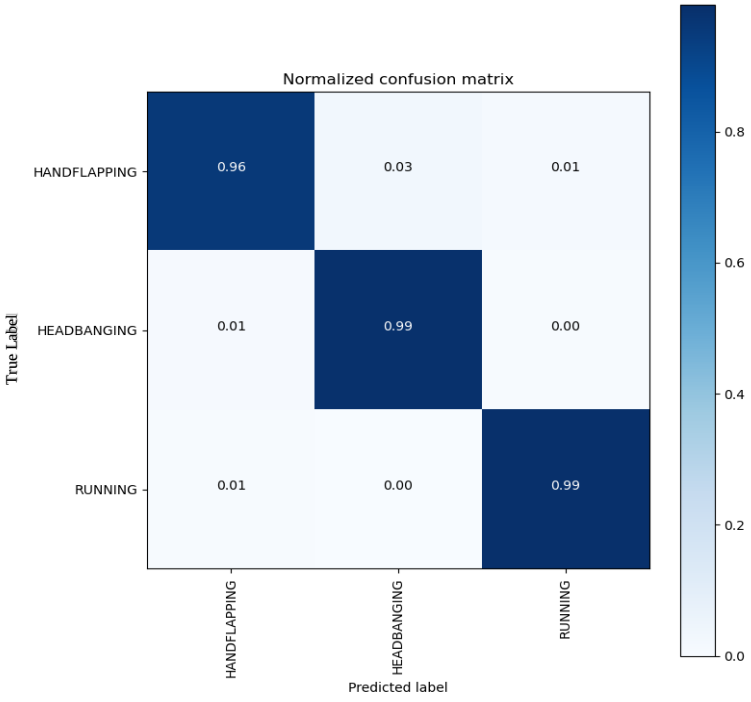

Figure 41. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the Random Forest algorithm

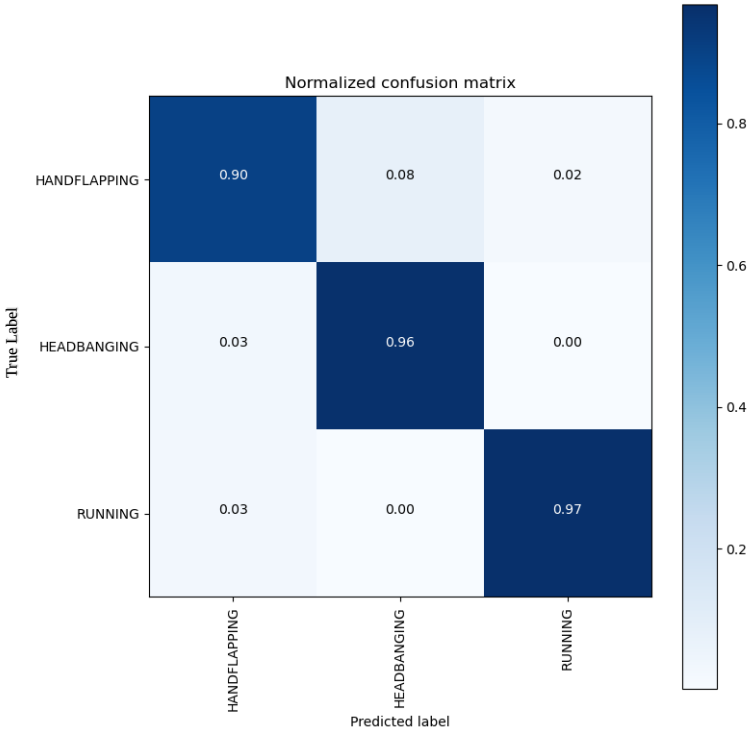

Figure 42. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the Logistic Regression algorithm

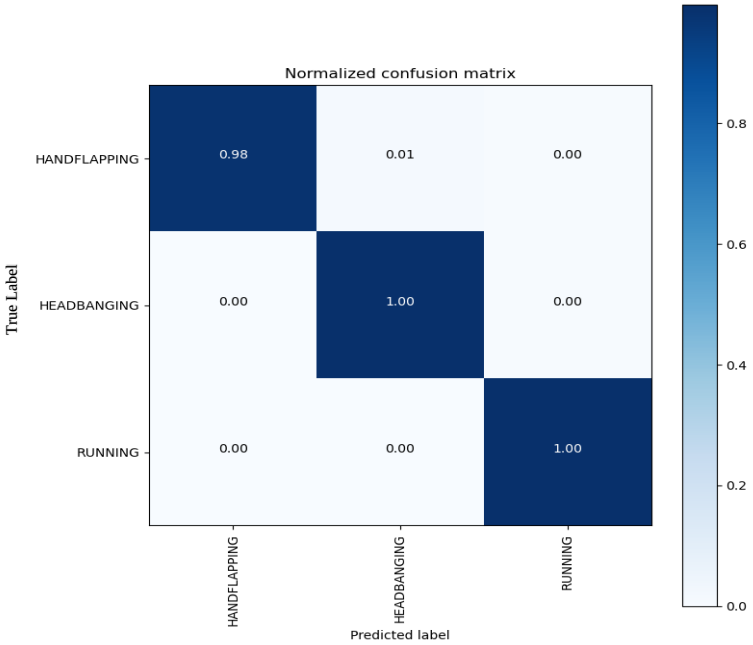

Figure 43. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the Ridge algorithm

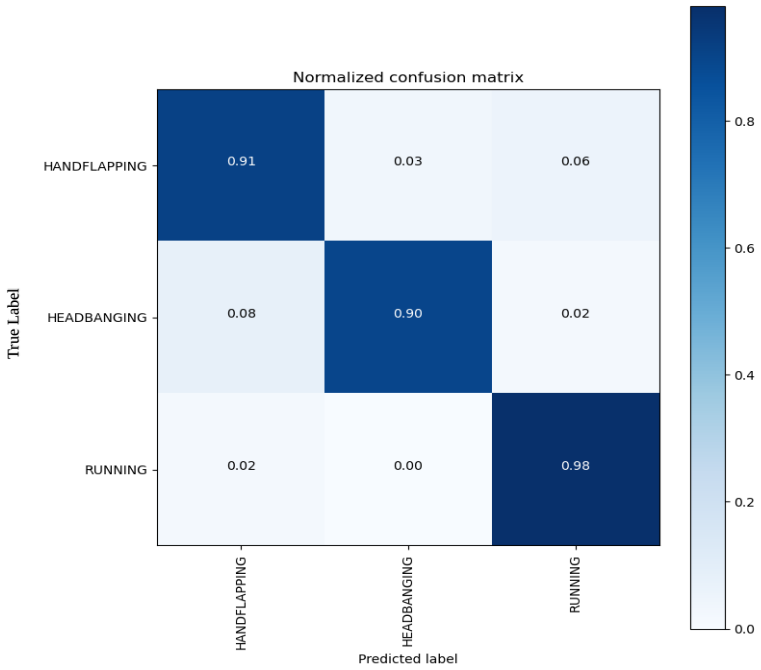

Figure 44. Confusion matrix for grid search hyperparameter tuning for a window of 3-second data computed in the KNN algorithm

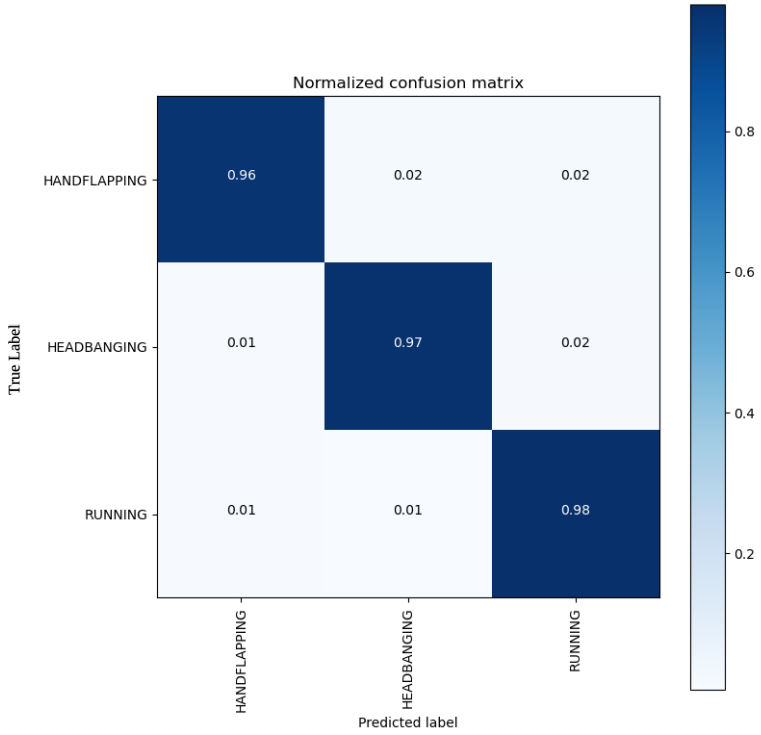

Figure 45. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the DT algorithm

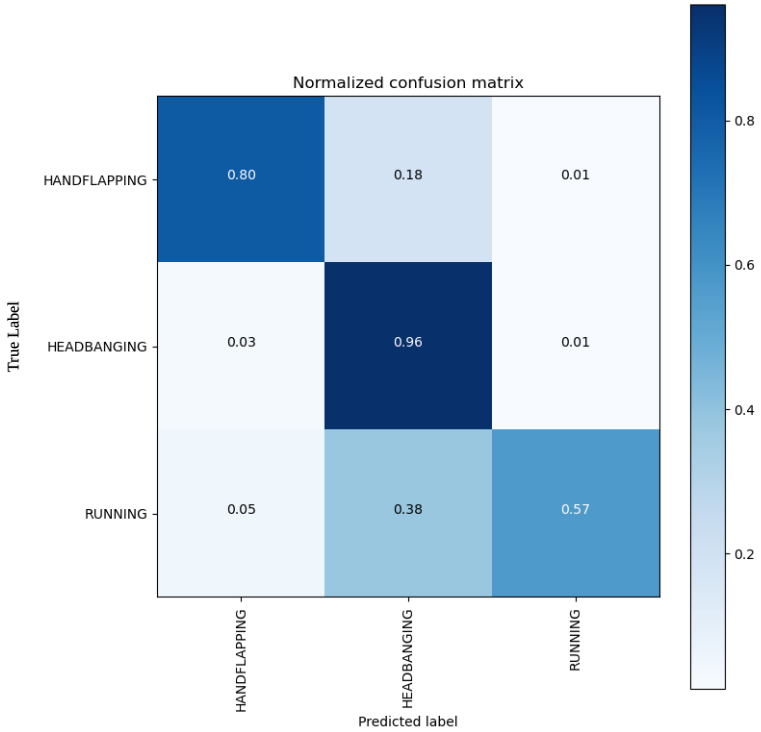

Figure 46. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the Linear SVM algorithm

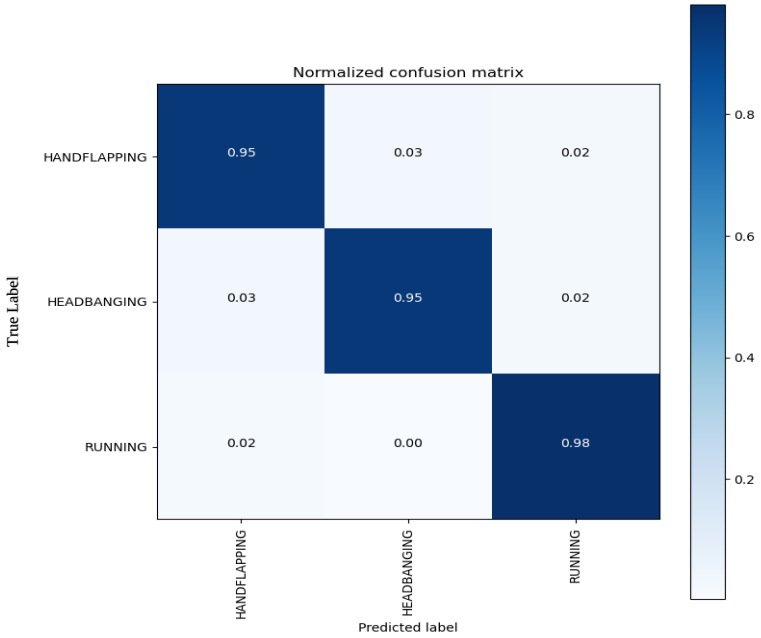

Figure 47. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the SVM algorithm

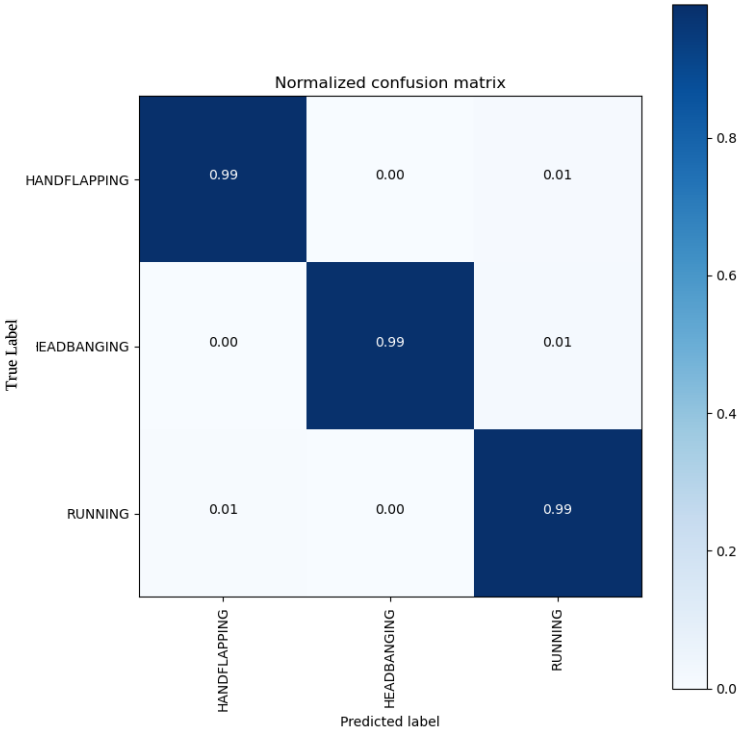

Figure 48. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the Random Forest algorithm

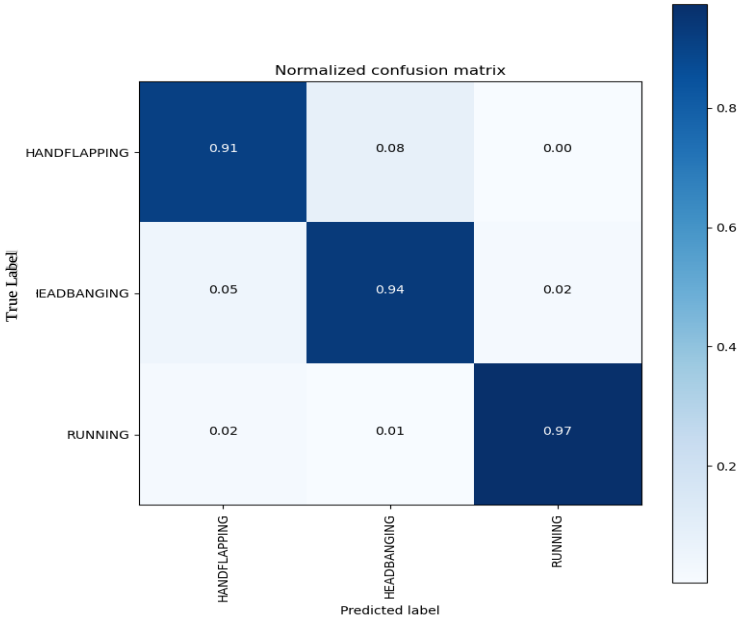

Figure 49. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the Logistic Regression algorithm

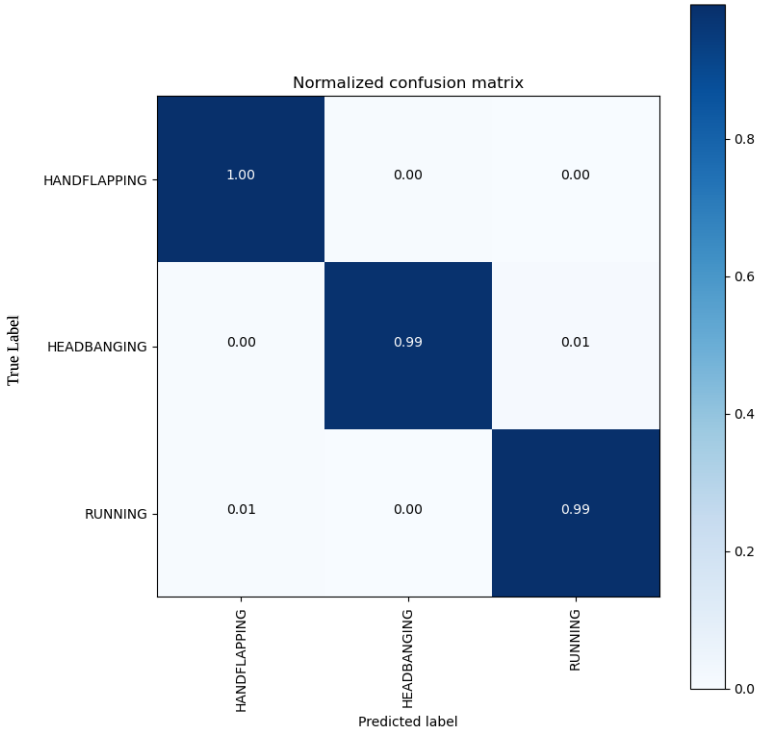

Figure 50. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the Ridge algorithm

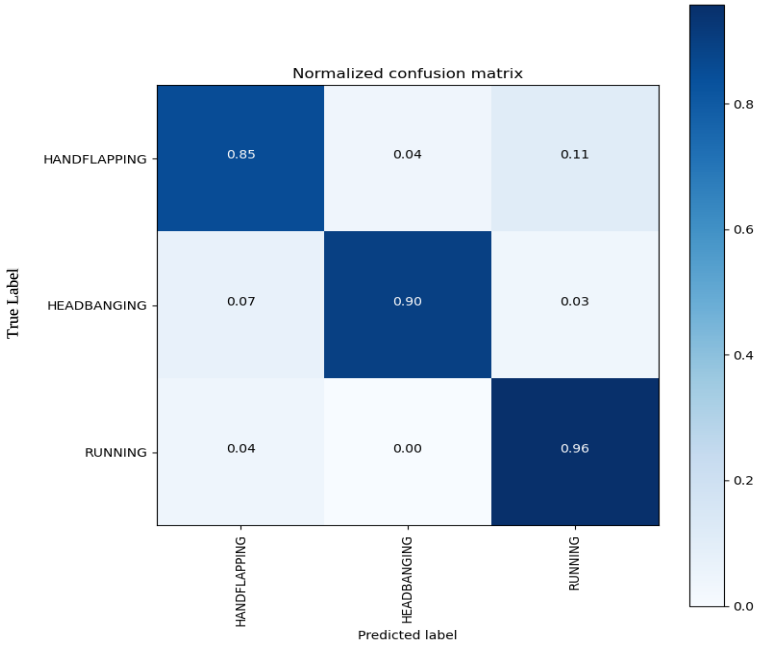

Figure 51. Confusion matrix for random search hyperparameter tuning for a window of 3-second data computed in the KNN algorithm