8-27-2016

# Real Time Activity Recognition of Treadmill Usage via Machine Learning

Nathan Blank
*Rose-Hulman Institute of Technology*

Matt Buckner
*Rose-Hulman Institute of Technology*, bucknemj@rose-hulman.edu

Christian Owen
*Rose-Hulman Institute of Technology*, owencb@rose-hulman.edu

Anna Scott
*Rose-Hulman Institute of Technology*, scottae@rose-hulman.edu

# Real Time Activity Recognition of Treadmill Usage via Machine Learning

Nathan Blank, Matt Buckner, Christian Owen, Anna Scott

Email: bucknemj@rose-hulman.edu, owencb@rose-hulman.edu, scottae@rose-hulman.edu

Advisors: Carlotta Berry, Valerie Galluzzi, Yosi Shibberu

Rose-Hulman Institute of Technology, Terre Haute, Indiana, USA

Spring 2016

## Abstract

Our objective is to provide real-time classification of treadmill usage patterns based on accelerometer and magnetometer measurements. We collected data from treadmills in the Rose-Hulman Student Recreation Center (SRC) using Shimmer3 sensor units. We identified useful data features and classifiers for predicting treadmill usage patterns. We also prototyped a proof of concept wireless, real-time classification system.

# Table of Contents

# Introduction

The outline of this technical report is as follows: We begin with a review of the basics of data mining and machine learning. The methods section contains a detailed description of the Shimmer3 Sensor Unit, used to generate the data for this project; how we established the ground truth data set; how features and classifiers were selected; and a description of our prototype Bluetooth-based data classification system. The Results section contain tables of the features selected and the accuracy of several classifiers over the ground truth data set. We summarize what we have learned in the course of this project and highlight areas for further research in the Conclusions section. Software tools and supporting technical information are provided in the Appendices.

## Machine Learning

Machine learning is the use of methods to categorize and cluster data. In broad terms, it is the collection of data pertaining to some event, recognizing what characteristics or values calculated from the data best describe the phenomena in question, forming a set of rules to separate the sub-events into categories based on the data, and then recognizing and categorizing new occurrences based on collected rules. For example, say we collected information about different fruits such as size, color, skin texture, weight, etc. Can we identify fruits based on these descriptive characteristics? For example, say that color tells us the most information about the identity of a fruit. We could then make a rule using color that red fruits are apples, and orange fruits are oranges. When given a new fruit, we can examine our rule to determine what fruit it may be. If it is red, we say that is an apple. As more data or fruits are collected, we can continue to refine our rule to better determine the identity of new fruits. This process is outlined in Figure 1.

### Gathering Data

In our example, gathering data is the process of collecting and observing multiple fruits. If we only gather a small subset of fruits, a small data set, then we might not have enough information if a new, unknown fruit is introduced later. For this reason we seek to gather a “Ground Truth Data Set”, or a data set where we have information on all possible expected outcomes. The goal is to possess enough information to determine what makes one outcome unique compared to others. These different expected outcomes are called “classes” and depending on intent can be labeled by hand or using software to generate labels automatically. In our fruit example, our class labels would be “Apple,” “Orange,” etc. A Ground Truth Data Set could consist of 5 of each of the desired fruit types that are free of bruises. This will give us an idea of what the “ideal” apple or orange is.

### Features

From our data set we can select features. Features are observed or calculated attributes of our classes. In the fruit example, this might include data attributes such as fruit color, weight, or size, along with computed values such as mean number of seeds, maximum weight, or standard deviation of size. Once features have been collected, we want to see which features best describe the classes. For this we use a processes called Feature Selection. Feature Selection consists of tests that determine how much a feature or set of features relate to the classes. Intuition and logic is also involved. With fruits, by intuition color would very easily distinguish between apples and oranges, but using color alone may prove problematic in recognizing the difference between an apple and a cherry. For this reason, multiple features are normally used to help define our data set.

## Classifiers

Classifiers are methods/algorithms used to find a set of rules created with selected features to categorize our data. A simple classifier might be the rule “Orange fruits are Oranges,” or a more complex sets of rules such as “Red fruits under 1 inch in diameter with a single seed are Cherries”. These rules are created automatically over a data set based on the selected features by the classifier. Many different classifiers exist. Each classifier has their own advantages and disadvantages in terms of accuracy and speed. Once a classifier has been run over a set of data, it forms what is called a “Model”, and is considered “Trained”. These rules can be further modified over time by “Retraining” the model. Classifiers are compared via a procedure called cross-validation. This process is used to prevent under-fitting and over-fitting of the model.

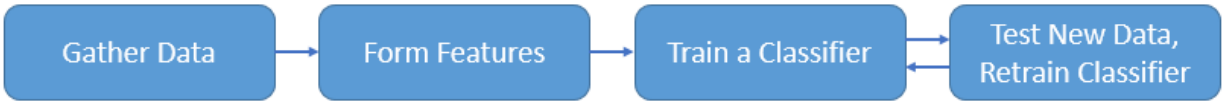



*Figure 1. The conventional machine learning pipeline*

# Methods

## Shimmer3 Sensor Unit

The Shimmer3 Sensor Unit is a small device housing multiple sensors, as listed in Table 1. It was created by Shimmer Sensing to provide a method to collect data during periods of physical action. The device is activated either upon removal from the base station or by pressing the orange button on the front of the unit, depending on settings available while connected to the base station and PC via Consensys. Once data values are recorded by the unit, the data can then be downloaded to a PC for investigation via connection to the base station and Consensys or via Bluetooth.

| Sensor name | Calibrated Units | Description |
|---|---|---|
| Timestamp | Milliseconds | Time of data point |
| Acceleration_WR: X,Y,Z | Meters/seconds$^2$ | Handles large fluctuations |
| Acceleration_LN: X,Y,Z | Meters/seconds$^2$ | Handles smaller precision values |
| Gyrometer: X,Y,Z | Degrees/second | Returns angular velocity values |
| Magnetometer: X,Y,Z | Local flux | Returns strength and direction of local magnetic field |
| Pressure | Kilopascals | Scalar ambient atmospheric pressure |
| Temperature | Degrees Celsius | Scalar ambient temperature |
| Battery Life | Millivolts | Scalar battery voltage |

*Table 1. Default active Shimmer sensors with the corresponding units and descriptions*

In our project these sensors were placed on treadmills in the Rose-Hulman SRC. We examined the data produced by running on the treadmill, walking on the treadmill, and activating the treadmill with no user on the device to form our classes of interest: Running, Walking, Not_Walking. We focused our investigations on the Acceleration_WR, Acceleration_LN, Gyrometer, and Magneometer to attempt and locate useful features in classifying treadmill usage into our three classes of interest. As shown in Figure 2, Shimmer sensor units were placed under one or both of the locations displayed – the front center and back left of the treadmill.

*Figure 2. Shimmer Sensor Units were placed under one or both locations for data collection as shown in green in the image: the front center and the back left of the machine*

## Logging Data

Initial gathering, synchronization, and examination of data was based on the physical connection of the Shimmer units to a PC via the base station and Consensys software. This method produced the majority of the data sets examined at the beginning of this project, including the Ground Truth data set which was used in the examination of different attributes, classifiers, and attempts to build a trained classifier for live usage. Following this method of logging data, we then moved to a Bluetooth connection, with the intent to categorize live streaming data. Due to differences in how the data was provided through these methods, classifiers trained on the base station synced data were incompatible with data acquired wirelessly, and new training sets had to be formed.

Information on gathering data from the Shimmer units via the base station can be found in the Shimmer user manual, and steps on connecting wirelessly are located in Appendix 4.

## Gathered Data Sets

Data was collected from Shimmers on the treadmills in the SRC. The placements of the Shimmer Sensor Units are marked in the above image by the smaller dotted squares in the front middle and back left of the treadmills as illustrated in Figure 2 and Figure 3.

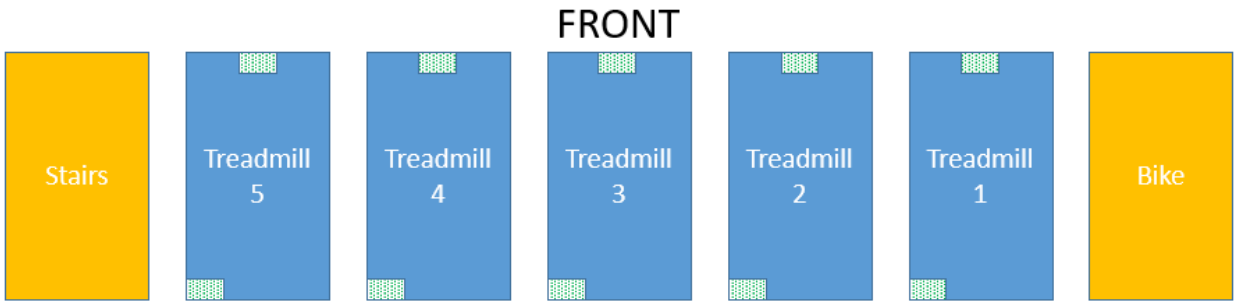



*Figure 3. A diagram illustrating the potential placement of Shimmer locations on the testing treadmills. Which location is utilized differs between data sets.*

## Ground Truth Data Set

This data set was gathered on a Saturday morning prior to the official opening of the SRC, allowing for a controlled set of actions without interference from other treadmill users. Data was collected with a single Shimmer unit placed on the front center of each treadmill. Testing was primarily focused around Treadmill 2, with certain tests including runners using other treadmills simultaneously to examine potential unwanted feedback. Testing involved different runners of various heights and weights running at set speeds for known periods of time in order to form a baseline of categorical data. A full breakdown of the activities performed can be found in the Ground Truth Journal on the server. Unfortunately, data collected on Treadmill 4 was lost.

This dataset was collected to provide a controlled data set to perform feature analysis, and train our initial classifier. As such, this data set has been labeled with “Running,” “Walking,” and “Not_Walking,” and video footage of all tests was collected to support our data and potentially allow for step counting in further developments of this system. This data is of particular note as it provides similar actions (running at a set of designated speeds) across multiple users of various weights and step size, along with providing examples of the major usage scenarios without additional noise. Additionally, another data set examining potential features calculated from the Ground Truth Data was formed. This compiled data set

includes the Mean, Max, and Standard Deviation over each sensor value, as well as the Mean, Max, and Standard Deviation over the Power Spectrums and Frequency Spectrums formed on the original data.

### Multi-day Data Set

A multi-day "real-world usage" data set collected prior to the start of the project by Dr. Galluzzi using two Shimmer units per treadmill. This dataset was collected to experiment with the capabilities and quirks of the shimmers, and to become more familiar with larger datasets.

### Introductory Data Set

A sub-hour "real-world usage" data set collected at the beginning of the project using two Shimmer units per treadmill on a Monday afternoon with other patrons using the SRC during data collection. This dataset was collected as an introduction for the class to see how the data looks in a "real world" situation. Since it was collected during a busy period of the gym, we were able to see how much noise we would expect, as well as how strongly signals will transfer between nearby treadmills.

### Streaming Training Set

Due to differences in configuration of the shimmers units between logging and streaming over Bluetooth, we had to collect a new ground truth training set for streaming. This dataset was collected in relative isolation at night, so it is not entirely representative of real world usage. During collection, we noted that the classifier may overfit on certain parameters, so a wide variety of situations were recorded; This included letting the treadmill operate with nobody actually on it, running next to the treadmill, and walking backwards and other strange patterns.

## Processing the Data

Data collected via the Shimmer Base Station, as in all but our Streaming Training Set, is initially downloaded as a .csv (Comma Separated Values) file from Consensys, which was examined by hand via MATLAB. Our initial efforts were to examine these raw values for any immediately obvious patterns corresponding to activity type, along with building possible features from these values by creating Power Spectrums and Frequency Spectrums, and then finding values such as Mean, Max, and Standard Deviation. Once these potential features were constructed from our data set, further data exploration was performed via Weka, which operates best on .arff (Attribute-Relation File Format) files, a proprietary format for Weka. Code for converting .csv files to .arff files is provided in Appendix 2; alternatively, online .csv to .arff converters are also available. Once the data was converted to an .arff file, examination of the data was handled via the Weka GUI (Graphic User Interface) in order to select attributes and train a classifier.

## Live Classification System

In the last few weeks of the quarter, we used wireless data streaming capabilities together with the machine learning methods that had been researched throughout the quarter to create an operational model that is able to estimate the state of a treadmill user (not moving, walking, and running). We also attempted to build a step counter to estimate the number of steps taken by a person on the treadmill.

The BtStream 0.7.0 firmware was used in conjunction with modified Python scripts from the shimmer3 repository (https://github.com/ShimmerResearch/shimmer3) to pair a PC with a shimmer3 device (via Bluetooth). Once paired, the PC could wirelessly configure the shimmer and receive streaming data from the onboard sensors. Steps for this process can be found in Appendix 4. Due to differences in how

the data is provided, classifiers trained on data gathered via connection to the Shimmer Base Station are not compatible with our current streaming system. As such, a new training set was formed via streaming data and a classifier was trained via the Random Forest classifier. We used all data provided by the Accel_WR, Accel_LN, and Mag sensors, following the results provided by Attribute Selection and Classification over our previous data.

### Description of the System

Figure 4 shows a block diagram of the current operational system. The liveClassify_v2.py script and associated modules (iotdata_simpleMostlyStatic, ShimmerBluetooth) are the heart this system.

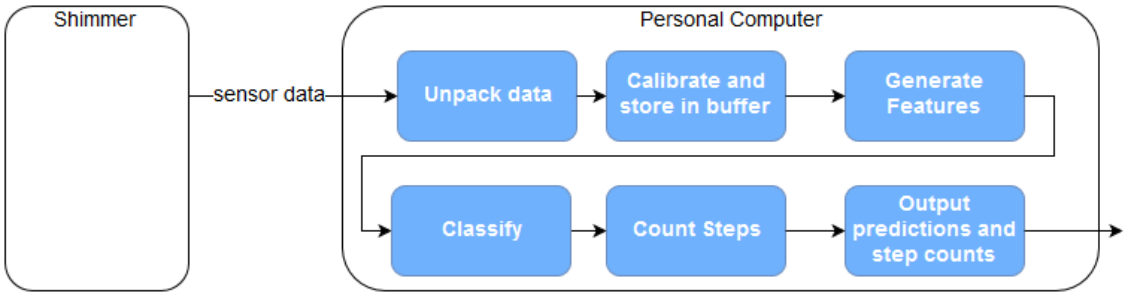



*Figure 4. Operational model system diagram. Data is unpacked in the ShimmerBluetooth class, calibration occurs in liveClassify_v2, feature generation is performed in iotdata_simpleMostlyStatic, and Classification is carried out by a random forest in liveClassify_v2. Step counting (which happens in the countSteps function in liveClassify_v2) still needs work.*

Successful calibration required digging through the C# source code for Shimmer Capture. This source code is provided to all shimmer customers who get an account on their website. Sampled signals from the magnetometer and accelerometer were the only sources used to generate features, as the attribute selection methods in Weka repeatedly indicated that features generated on the other signals were not helpful. This makes sense: the other signals are samples from gyroscope, ambient temperature, ambient pressure, and battery voltage.

Testing in Weka revealed that Random Forest models were accurate and quick to train; further research and input from professors revealed that this classifier is hard to overfit as relative to other classifiers (e.g. multi-layer perceptron, J48 decision tree). For these reasons, the operational system used random forests from the Scikit-Learn python library. The classifier was trained in 'scikitlearn_classifier2.ipynb' which is a Jupyter notebook containing python code.

The step counting module is unfinished. Code in MATLAB shows that the current peak detection scheme works, but when the code was ported to python for implementation in the operational system the lack of a robust peak finding method caused the module to fail. This group ran out of time, but if a robust peak detection function can be written or found online, the step counting will work.

## Results

### Selecting Attributes

Attribute evaluation was performed from the Weka GUI, the process of which is included in the Weka GUI guide in Appendix 3. Through the course of this project, multiple different evaluators were examined over the Ground Truth "Compiled" data set, the top results of which are available in Table 2. A description of each of the feature selection methods that we used and the top attributes selected by the

method it can be noted that regardless of Attribute Evaluator used, results consistently pointed to a combination of both the Magnetometer and Accelerometers

| Attribute Evaluator | Brief Description | Top Attributes |
|---|---|---|
| CorrelationAttributeEval | A component-by-component linear comparison between change in label and change in a single attribute. | Mag X Frequency Spectrum Stdev<br>Mag X Frequency Spectrum Mean<br>Mag X Mean<br>Accel WR Y Stdev<br>Accel LN Y Stdev |
| PrincipalComponents | Using each data type as a separate dimension, this method transforms the data into a new orthogonal coordinate system in order to find the dimension with the highest variance. | Mag Z Window Stdev<br>Mag X Power Spectrum Max<br>Mag Z Power Spectrum Max<br>Accel LN X Frequency Spectrum Mean<br>Accel LN Y Frequency Spectrum Mean |
| ClassifierSubsetEval | A Method to select features by running an arbitrary classifier over different subsets of features. | Accel Y Zero Crossing Rate<br>Accel Y Power Spectrum Mean<br>Mag Y Power Spectrum Max<br>Mag Y Power Spectrum Mean |
| WrapperSubsetEval | A Method to select features by running an arbitrary classifier over different subsets of features. A streamlined extension to ClassifierSubsetEval. | Accel Y Power Spectrum Mean<br>Mag Y Power Spectrum Max<br>Accel Y Zero Crossing Rate<br>Mag Y Mean |

*Table 2. A description of each of the feature selection methods that we used and the top attributes selected by the method it can be noted that regardless of Attribute Evaluator used, results consistently pointed to a combination of both the Magnetometer and Accelerometers*

Based off of these alternative evaluators, we noted that the Magnetometers and Accelerometer sensors appeared to best correlate to the classes of Running, Walking, or Not_Walking.

## Classifier Selection

Once attributes were selected, different classifiers were applied to the Ground Truth "Compiled" data set via 10-fold cross validation in order to examine the expected accuracy of each set of attributes and of each of the potential classifiers.

In Table 3. Cross validation results from testing trained J48, random forest, and naïve bayes classifiers while using the top 3, 5, and 10 features as well as the full feature set. Results showed that using a random forest classifier trained on all attributes gave the best result., different classifiers were tested over the set of attributes produced by the CorrelationAttributeEval evaluator in order to examine the effect of using more or less attributes in our classifier. Not all subsets of attributes were performed on NaiveBayes due to excessive run time.

| Selected Attributes | Classifier | True Positive Rate | False Positive Rate | Precision | Recall |
|---|---|---|---|---|---|
| **Top 3** | Decision Tree (J48) | 0.936 | 0.299 | 0.931 | 0.936 |
| | RandomForest | 0.930 | 0.295 | 0.926 | 0.930 |
| **Top 5** | Decision Tree (J48) | 0.968 | 0.130 | 0.967 | 0.968 |
| | RandomForest | 0.973 | 0.104 | 0.972 | 0.973 |
| **Top 10** | Decision Tree (J48) | 0.967 | 0.120 | 0.967 | 0.967 |
| | RandomForest | 0.977 | 0.082 | 0.977 | 0.977 |

| | | | | | |
|---|---|---|---|---|---|
| **All Attributes** | Decision Tree (J48) | 0.971 | 0.104 | 0.971 | 0.971 |
| | NaiveBayes | 0.594 | 0.034 | 0.904 | 0.594 |
| | **RandomForest** | **0.981** | **0.086** | **0.980** | **0.981** |

*Table 3. Cross validation results from testing trained J48, random forest, and naïve bayes classifiers while using the top 3, 5, and 10 features as well as the full feature set. Results showed that using a random forest classifier trained on all attributes gave the best result.*

Based on this result, we found that for our data set, training on all pertinent attributes provided a more accurate result than utilizing a smaller subset.

### Live Classification System Results

Unfortunately, the calibration settings and general setup of the shimmers was not documented when collecting the ground truth dataset (Ground Truth Data Set: 03_27_16) which prompted the collection of a new dataset. The main problem with the ground truth set is the fact that we did not record which accelerometer was sampled. An important discovery that the team made in the process of setting the shimmers up for streaming is that there are multiple accelerometer sensors available on a single sensor. Future groups should note which accelerometer they sampled prior to data collection. Temporarily, a small ad-hoc dataset (Streaming Training Set: 05_12_16 – 05_13_16) was collected and used to train the classifier. Cross validation of the model showed 97% accuracy in classification, but formal quantitative results were not collected on the system. To formally test the system, a validation dataset would need to be collected, but (after observing the system work in real time) it can be said that the system looks promising overall.

## Conclusions

We found most of our time was spent in data exploration and feature selection. The Accelerometer_WR, Accelerometer_LR, and Magnetometer sensors provided the highest accuracy for classification. Of the classifiers examined, Random Forest provided the most accurate classification results for the selected features. One classifier that we did not fully explore is Neural Networks. These classifiers have the potential to reduce the amount of work required in feature selection. We plan on further exploring Neural Networks in future works.

The collection of a good ground truth data set is essential for the proper evaluation of classification methods. We found that as our system matured, especially with the introduction of Bluetooth data syncing, our initial ground truth data set became incompatible with the new data collected. As such, in order to provide proper evaluation of classifiers we note the importance of collecting a new ground truth data set that matches the final system layout.

We observed battery life of the Shimmer3 units proved to be a shortcoming in long term deployment. As such, a data transfer method that reduces power consumption or a classifier that reduces the amount of data that needs to be transferred is an area worthy of further consideration.

One subject not covered within the scope of this project is data security. For a full implementation of a similar system, steps would need to be taken to preserve the security of user data.

## Appendix 1: Useful Applications

### Consensys

Download Link: http://www.shimmersensing.com/shop/consensys

The official Windows software provided by Shimmer Sensing, allowing for the syncing of data from the Shimmer sensor units via the provided base station. Can be downloaded at the above link.

### Weka

Download Link: http://www.cs.waikato.ac.nz/ml/weka/downloading.html

Command Line Guide: http://www.uky.edu/~nyu222/tutorials/Weka.htm

Software produced by The University of Waikato that allows for easy classification and investigation of large data sets. Furthermore, models (used for classification) can be constructed based off existing data sets and new data can be classified based off these pre-created models. Many different classifiers and clustering methods are provided within the Weka GUI (Graphic User Interface), and further tools can be accessed via the Weka Command Line.

### MATLAB

Available via Rose Intranet

Software that allows for further data exploration. Initial data examination for our project group began in MATLAB, but extended to Weka for its ease and integration with Python. MATLAB can be of use if more complex customized data manipulations is required.

### Python

Download Link: https://www.python.org/downloads/

NOTE: 2.7 was needed for the Python Weka Wrapper at time of writing (May 2016)

Used as our primary programming language to collect, manipulate, and classify our data via the Python Weka Wrapper (See Below).

### Python Weka Wrapper

Download Link and Guide: http://pythonhosted.org/python-weka-wrapper/install.html

Usage Guide: http://pythonhosted.org/python-weka-wrapper/index.html

Used to integrate Weka and its constructed models within Python. Installation can be complex for Windows machines, but all required files and steps are included in the first link. The second link connects to multiple examples for both command line and Python script usage.

### SSH Client

Windows SSH Client PuTTY: http://www.putty.org/

Linux command: ssh <username>@shimmer.csse.rose-hulman.edu

Used to access the provided server containing collected data and journals.

### FTP Client

SecureFX should be available on current Rose Laptops

Alternative Windows application: https://filezilla-project.org/

Linux command: ftp shimmer.csse.rose-hulman.edu

Used to easily download and upload the potentially large data sets. This is suggested to be done on campus instead of over the VPN.

# Appendix 2: Links to existing code

## MATLAB

https://github.com/cowen314/IoT_Work/

## Python

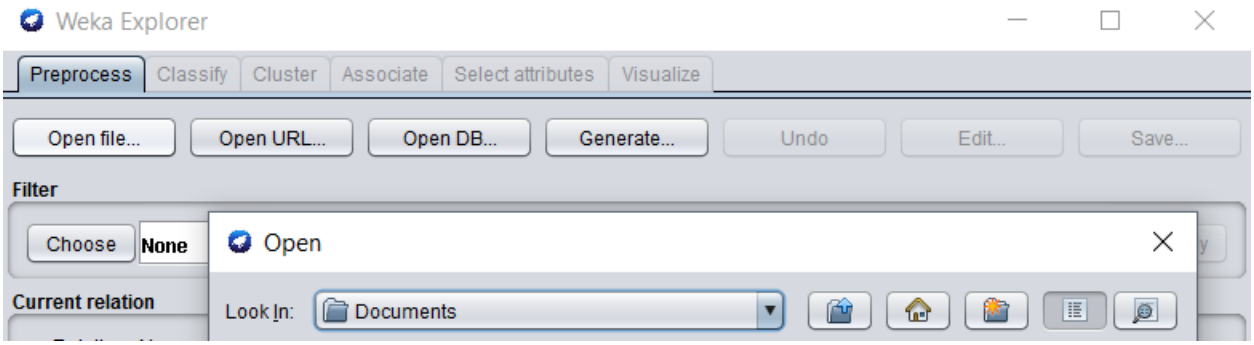
https://github.com/Healbadbad/InternetOfThingsClass

# Appendix 3: Weka Tutorials

The following is a short tutorial on how to quickly find attributes and classify data from within the Weka GUI.

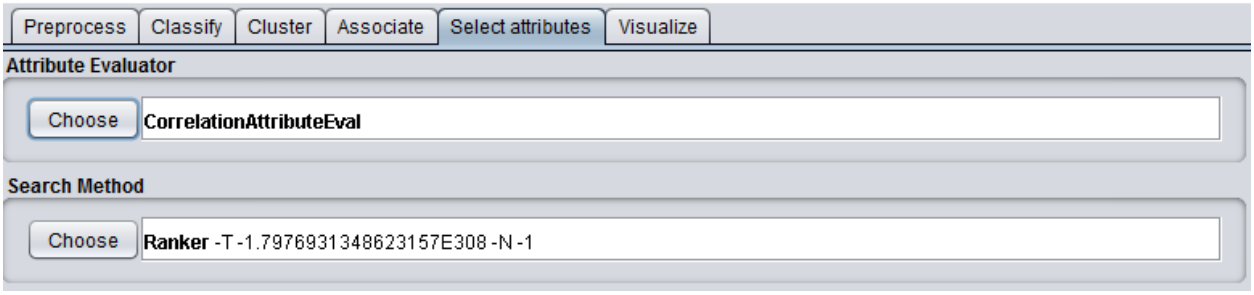
Open Weka on your computer and select “Explorer” from the GUI Chooser page that appears.

## Preprocessing



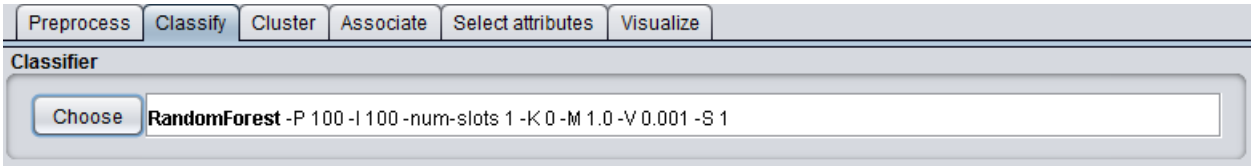

Open the file from the Weka Explorer window via the “Open File” button, and then utilize the Attributes window in the lower left to remove unneeded data types such as Time Stamp, Temperature, Pressure, and Battery by selected all undesired data types and pressing “Remove” in the lower left. The Select Attribute Tab can then be used to evaluate attributes of the file.

## Attribute Selection




The CorrelationAttributeEval option in the Attribute Evaluator will provide rapid ordered results of the most significant attributes in a linear comparison of each attribute with the “class” selected in the drop down box under Attribute Selection Mode (if available). It is also highly suggested to use Cross-validation whenever data is being examined within Weka to avoid overfitting.

## Classification




Classification within the Weka GUI can be reached from the "Classify" tab. For the purposes of this project, a majority of our Classification was performed with the Trees > RandomForest classifier, with 10 Fold Cross-validation. Although other classifiers were examined, as noted in "Classifiers", RandomForest was used as it produced the best results in testing. Further experiments can be performed by removing additional low-correlation fields via the "Preprocess" tab. Once a model has been trained, right click on it in the Result List on the lower left corner to save the model for future classification.

## Reading Weka Classifier Output

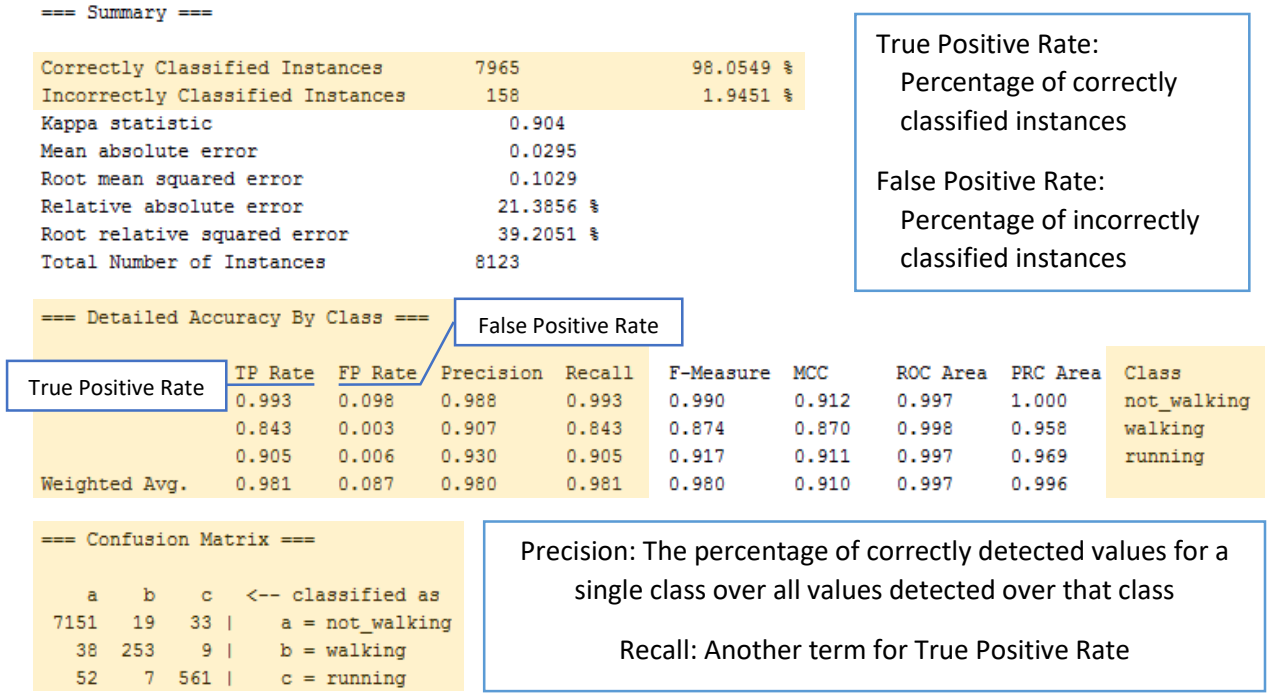



*Figure 5. Example Weka output including accuracy & classification statistics by class, and a confusion matrix. Relevant fields are highlighted, displaying the accuracy of this particular classifier.*

Figure 5. Example Weka output including accuracy & classification statistics by class, and a confusion matrix.shows an example of the output produced from Weka when a classifier is run. The initial values at the top provide a quick overview of the accuracy of the chosen classifier, providing quantitative results on how many instances were correctly classified. The middle of the image displays True Positive Rate, False Positive Rate, Precision, and Recall as defined in the same figure. Finally, a confusion matrix appears at the bottom of the figure, displaying a numeric breakdown of exactly how many values were classified as a given label via the columns, and which label they actually belonged to in each row. Via this matrix, items on the upper left to lower right diagonal are correctly classified. Items off of this diagonal were incorrectly identified.

# Appendix 4: Bluetooth Connection

Follow these steps to establish a link:

I. Load the latest version of the BtStream firmware onto a shimmer of choice
II. Pair the shimmer with a computer. For pairing with a Windows 7 machine, see the sub-instructions below. Be sure that the shimmer is turned on prior to attempting to pair.
    - In control panel, navigate to the 'Add a Device' panel. The device should show up here as 'Shimmer3-EXXX'. Double click on the device. If the pairing code has not been set manually, it will be '1 2 3 4' by default.

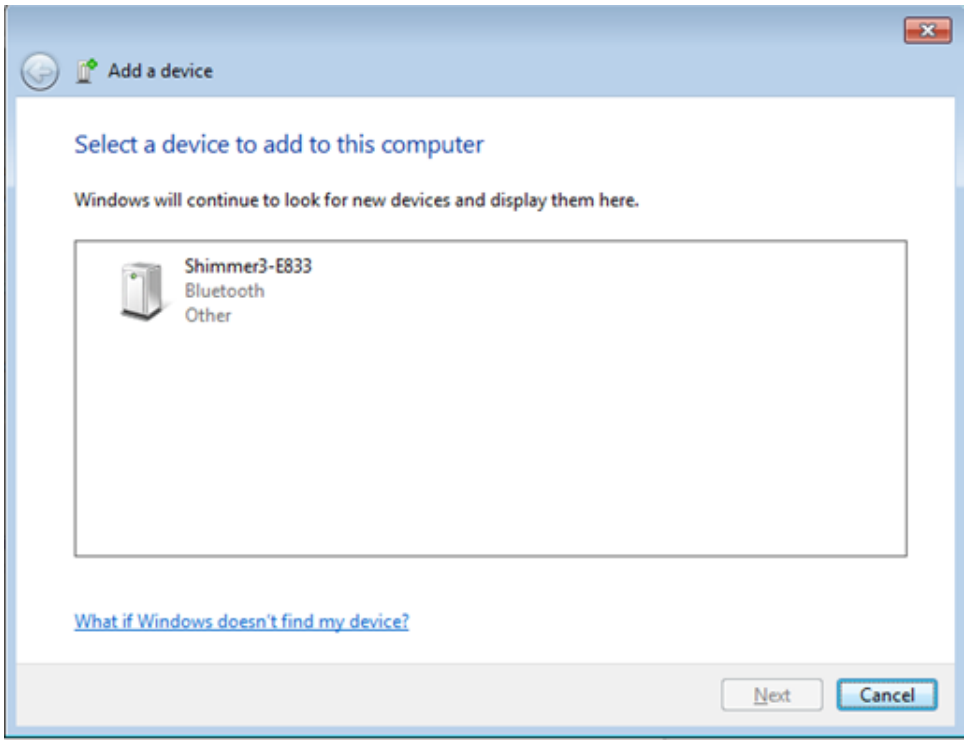



*Figure 6: The "Add a device" window*

    - If everything above works correctly, driver software will begin to install. Once the installations have finished, press the Start button and search for 'Bluetooth'. Click on 'Change Bluetooth Settings'. Click on the 'COM Ports' tab and remember the outgoing port number of the connected shimmer device.

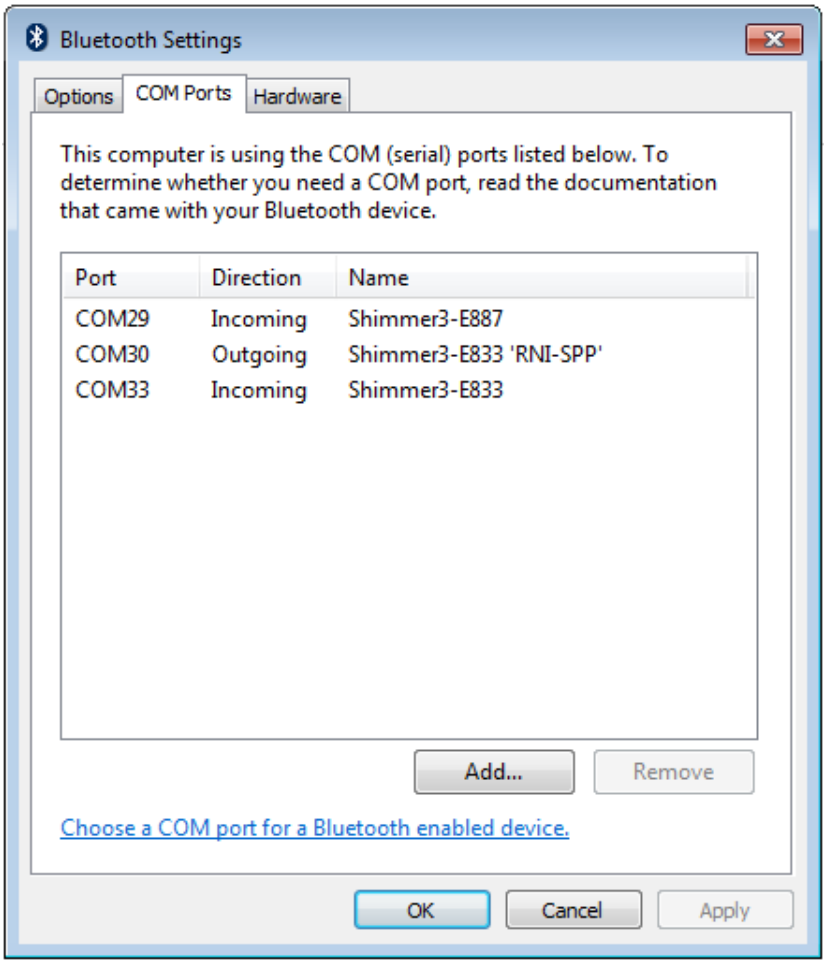



*Figure 7: The "Bluetooth Settings" window, which shows that COM30 is the port to be passed to the python script when communicating with the device from the PC.*

III. With the pairing complete, the connection between shimmer and computer can be checked by executing the following steps:
   - Download the python_scripts folder from the shimmer3 repository. From the command line set the current path to this folder, then execute the line 'python aAccelGsrGyro51.2Hz.py ComXX' where XX is the outgoing COM port number of the connected shimmer.
   - If the shimmer is properly connected raw accelerometer, GSR, and gyroscope values will continuously print to the command window.