# Medhere : A Smartwatch-based Medication Adherence Monitoring System Using Machine Learning and Distributed Computing

Jinxin Ma
jma33@dons.usfca.edu
Analytics Program
University of San Francisco

Anaelia Ovalle
aovalle@dons.usfca.edu
Data Science Program
University of San Francisco

Diane Myung-kyung Woodbridge
dwoodbridge@usfca.edu
Analytics Program
University of San Francisco

***Abstract*—Poor medication adherence threatens an individual's health and is responsible for substantial medical costs in the United States annually. In order to improve medication adherence rates and provide timely reminders, we developed a smartwatch application that collects data from embedded inertial sensors, which include an accelerometer and gyroscope, to monitor a series of actions happening during an individual's medication intake. After the collected data was delivered to a server, Apache Spark was used to distribute the data and apply machine learning algorithms in order to predict several discrete actions including medication intake. By utilizing these tools, we were able to preprocess high frequency sensor data and apply a random forest algorithm, yielding high frequency and recall of the aforementioned actions.**

***Index Terms*—medication adherence, machine learning, Apache Spark, smartwatch**

## I. Introduction

In 2010, patients in the U.S. spent $295 billion on prescription medicines, while the medication non-adherence rate was between 25% and 50%. Such low medication adherence rate costs between $100 and $300 billion annually, representing 3% to 10% of total health care costs in the US. Furthermore, the low medication adherence rate can increase hospital readmission rates and potentially cause a social and economic burden for the patient and their family. This may include higher copayments for the patient and even higher costs for employers to maintain healthcare coverage [1].

Conventionally, patients record and follow their medication intake using medication log sheets, text message reminders [2] or smartphone logging applications [3]. In order to improve the medication adherence rate, many recent studies have been developing systems utilizing low cost and wearable sensors which can remind, monitor medication intake and provide feedback. Chen's study utilized inertial sensors and a RGB-Depth camera in addition to an accelerometer and gyroscope that was attached to a patient's wrist to collect data. Dynamic time-warping was then applied to measure the similarity between time-series data with different lengths [4]. Kalantarian utilized smartwatches attached to a patient's both wrists for collecting and processing accelerometer and gyroscope data in order to detect a series of activities including 1) opening a bottle and 2) twisting a cap by using the distribution of the sensor readings [5].

Considering the ease of use, it is better to use embedded sensors in one device which is easy and light to wear. Additionally, a device that supports seamless data transfer, has a long battery life and is of durable quality improves usability. In that sense, a smartwatch provides higher usability and social acceptance along with capacities of measuring and transferring activity data. A survey with 221 people from Kalantarian's work shows that 72% of participants responded positively to wearing smartwatches [5].

Activity sensors including an accelerometer and gyroscope collects three dimensional data with frequency of up to 1000 Hz. This multidimensional high-frequency time-series data requires scalable solutions for storage, data-processing, and the application of machine-learning algorithms. Storing high-frequency data in the multi-user setting requires a cloud storage service which is scalable and accessible. Since motion data is captured per millisecond, the size of the data increases exponentially. Acknowledging these constraints, it was found that Amazon Web Service's (AWS) Simple Storage Service (S3) could provide sufficient, cost-effective storage based on these needs. With S3, data is accessible from anywhere with an option to replicate across many regions. Additionally, S3 offers a secure infrastructure through access policy options that allow only authorized users to access the data.

Distributed computing is designed for running big data on a cluster of many machines to make computation reliable, fast and inexpensive. Using distributed computing, a MapReduce job splits the data into smaller chunks followed by allocating the data to different partitions. A map task such as filtering and sorting can then process each partition in a parallel manner. The output of a map task becomes an input of a reduce operation which performs a summary operation. Apache Spark extends the MapReduce model for efficient data sharing and combines structured data-processing and machine learning all in a single framework. Spark runs programs up to 100 times faster than Hadoop MapReduce and provides an easy development environment [6].

In this study, we developed a smartphone-based medication

adherence monitoring system to collect a patient's motion data using an Android smartwatch during medication intake. The data was then stored in Amazon Web Service's (AWS) Simple Storage Service (S3) and analyzed by using Apache Spark.

## II. System Overview

### A. System Workflow

Smart-watches are good activity-monitoring devices as they contain various embedded sensors such as a three-axis accelerometer, gyroscope, near-field communication (NFC), and heart rate monitor. These seamlessly integrated sensors provide a much less obtrusive monitoring experience in comparison to smartphones or other wearable devices such as a heart rate monitor chest strap. By analyzing data collected from the sensors embedded in a smart-watch, an application can understand the context of an event and provide the right information a user might need. Additionally, information provided from a watch is more easily accessible than information provided from other devices including a laptop, tablet or smartphone [7]. In this study, we utilized an LG Watch Sport which is the first Android watch running on Android Wear OS 2.0.

Once data is transmitted to the server, the data-processing pipeline was built with four steps: 1) RDD creation, 2) DataFrame creation, 3) DataFrame processing, and 4) Activity classifications using machine learning algorithms.
Spark Core is a foundation of the system and provides basic Spark components including resilient distributed datasets (RDDs) and functions. An RDD is an abstraction of a distributed collection of data with operations applicable to the data and provides networking, security, scheduling and data shuffling functions.
In this work, we created RDDs from the data in the S3 bucket and used it for DataFrame conversion. Spark SQL provides functions for manipulating large sets of distributed and structured data. This structured data takes the form of a DataFrame which includes columns and values, similar to a table in a relational database management system (RDBMS). Spark translates them to operations on RDDs and executes them as Spark jobs [6][8]. Spark SQL uses SQL-like functionalities and query languages. In our system, each of the RDDs is piped into a DataFrame using SparkSQL and DataFrame schema definitions.

### B. Algorithm

*1) Data Preprocessing:* In order to save storage and computing resources, the data is only collected from the smartwatch application when there is a new sensor event triggered by an accelerometer or a gyroscope. Therefore, for discretizing the data and calculating the statistics of data, missing data imputation was necessary. Additionally, as this work applies classification algorithms to different lengths of timeseries data from the 3-axis accelerometer and gyroscope, data discretizion was applied. The psuedo-code for missing data imputation is listed in Algorithm 1 .

**Algorithm 1** Missing data imputation psuedocode

**if** $first\_reading[timestamp] > min\_timestamp$ **then**:
  $time = min\_timestamp$
**else**
  $time = first\_reading[timestamp]$
**end if**
**while** $first\_reading[timestamp] > time$ **do**:
  $reading = first\_reading[sensor\_data]$
  $time+ = time\_increment$
**end while**
**for** $data\ in\ sensor\_readings$ **do**:
  **while** $time < data[timestamp]$ **do**:
    $reading = prev\_reading$
    $time+ = time\_increment$
  **end while**
  $reading = data[sensor\_data]$
  $time+ = time\_increment$
**end for**
**if** $time <= max\_timestamp$ **then**:
  $reading = prev\_reading$
  $time+ = time\_increment$
**end if**

Once missing data is imputed, we discretized high-frequency data which was collected by the millisecond. As the time duration of each data varies, we reduced the time series data length of $n$ to the length of $f$ ($f \leq n$) and calculated statistics for the entire data and over a sliding window. When the original time series after imputing missing data is $C = c_1, ..., c_n$, the mean over the sliding window ($\overline{C}$) is calculated by (1). In addition to the mean in (1), we also calculated other statistical values including minimum, maximum, median and standard deviation accordingly.

$$\overline{\mu}_i = \frac{f}{n} \sum_{j=\frac{n}{f}(i-1)+1}^{\frac{n}{f}i} c_j \tag{1}$$

*2) Training and Test Sets Split:* We applied a stratified shuffle-split to convert the original data set into a training set that had 80% of samples and a test set that had the rest 20% of samples. A stratified shuffle-split divides the raw dataset into homogeneous subgroups. In our research, each subgroup is an activity. Afterwards, the correct number of samples is drawn from each subgroup to ensure that the training and test sets are representative of the overall population. A stratified shuffle-split can reduce sampling bias where a random split may fail.

We then trained a machine learning model on the training set and made predictions on the test set. This allowed us to see if the model generalized well on unseen data.

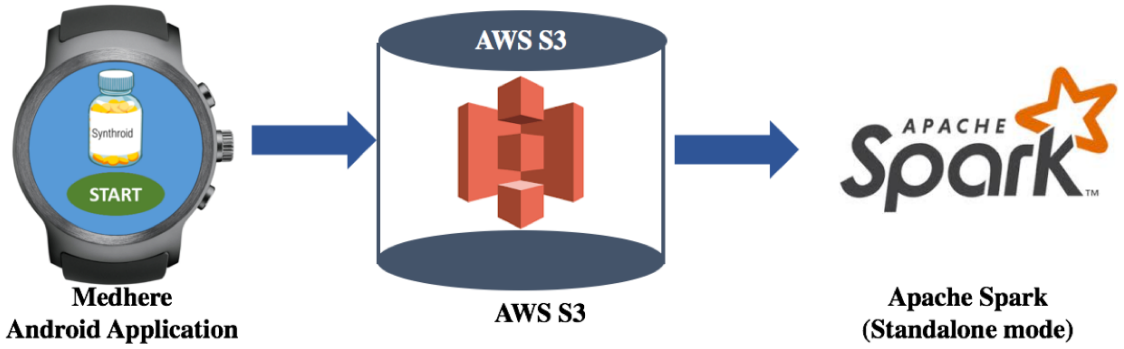



Fig. 1: Android application, AWS S3 and Apache Spark for data collection, storage and processing accordingly.

*3) Machine Learning Model:* Random forest models were used throughout the modeling process. A random forest is a ensemble method that trains several decision tree models separately and aggregates predictions from each decision tree to make final predictions [9].
Compare to a single decision tree, random forest has the following differences:

- Decision tree uses all samples in the training set while each tree in random forest is constructed from a proportion of samples (usually 30%) randomly drawn with replacement from the training set.
- When splitting a node during the construction of decision tree, the best split is chosed among all features. In random forest, the split of a tree is picked among a random subset of features.

As a result of the randomness, random forests usually yield a model that generalizes well on unseen data. In addition, random forests are implemented in Apache Spark which allows our solution to be scalable.

## III. Experiment Output

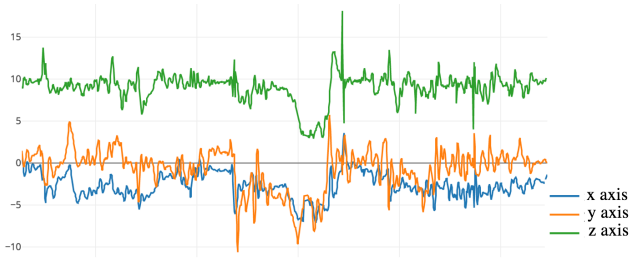
For the experiment, we utilized the LG Watch Sport running on Android Wear 2.0 with API version 26 and Apache Spark version 2.2.

### A. Data

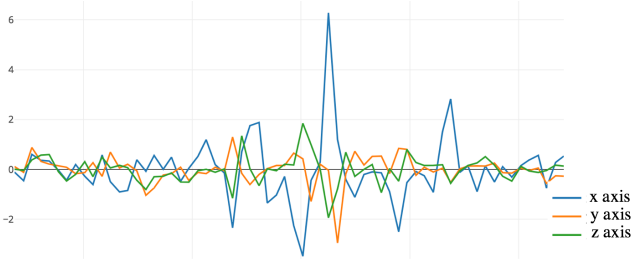
We collected data of six different activities including 1) medication intake while wearing a watch on one's non-dominant wrist, 2) medication intake while wearing a watch on one's dominant wrist, 3) walking, 4) texting, 5) writing with a pen, and 6) drinking bottled water. For the activities 3) - 6), a study subject could wear the watch on their preferred wrist.
We collected the aforementioned data from six different subjects listed in Table I. The duration of each activity varied from 2.85 to 16.13 seconds. Figure 2 visualizes an example of a sensor data stream from one of the subjects as they took a pill, wearing the watch on one's dominant wrist. This activity included opening a medication bottle, placing a pill into the mouth, closing the medication bottle.

TABLE I: Information of subjects who participated in the experiment.

| | Sex | Age | Dominant wrist | Wrist wearing a watch on |
|---|---|---|---|---|
| Subject 1 | F | 30-34 | Right | Left |
| Subject 2 | M | 25-29 | Right | Right |
| Subject 3 | F | 25-29 | Right | Right |
| Subject 4 | M | 30-34 | Right | Left |
| Subject 5 | M | 20-24 | Right | Right |
| Subject 6 | M | 35-39 | Right | Right |




(a) Accelerometer data




(b) Gyroscope data

Fig. 2: Collected timeseries high frequency data stream during medication intake.

After data preprocessing, the input data includes $30 \cdot (f + 1)$ features. Since we wanted to correctly distinguish medication intake regardless of what wrist the study subject wore in other activities, we combined the labels of activities 1) and 2) into a single class "medication intake".

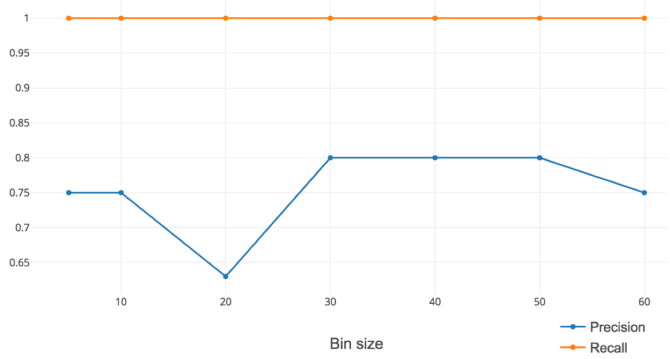



Fig. 3: Precision and recall of medication intake with different bin sizes ($f$).

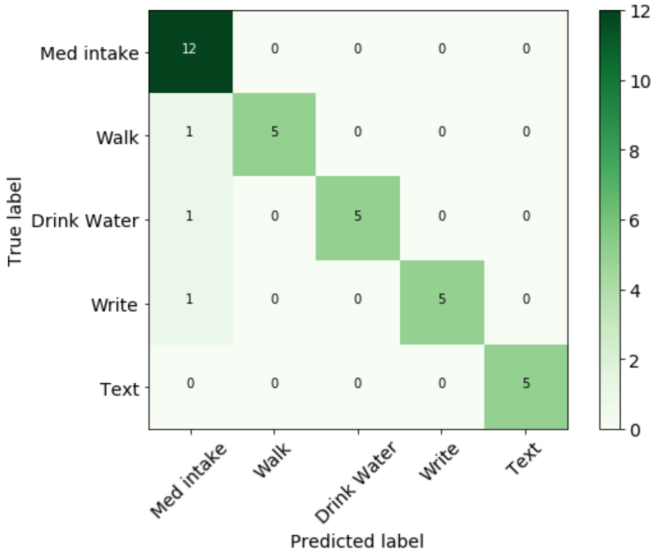



Fig. 4: Confusion matrix, where $f = 40$.

### B. Results

We applied the developed algorithm with different bin sizes ($f$). Figure 3 visualizes different bin sizes and its corresponding recall and precision.

$$Precision = \frac{t_p}{t_p + f_p}, \quad Recall = \frac{t_p}{t_p + f_n} \tag{2}$$

At the bin size of 40, we achieved a recall of 1.00 and precision of 0.80 for the medication intake class (Figure 3) within 161.15 seconds (Figure 5.) The average precision and recall across all activities were 0.91 and 0.93 respectively.
Figure 4 shows the confusion matrix on a test data set. The experiment results indicate that most activities are well classified including drinking bottled water which includes similar sequences of activities such as opening a cap, drinking water out of a bottle and closing the cap.

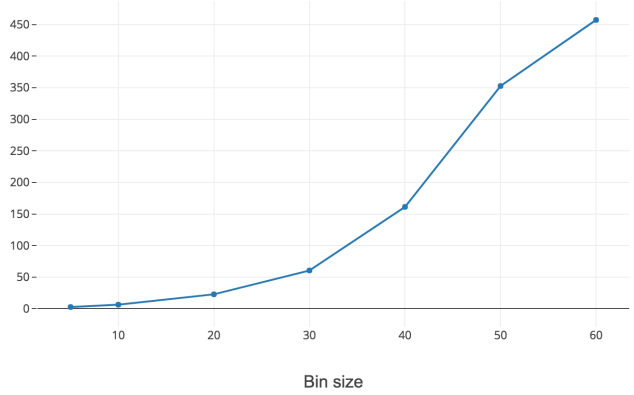



Fig. 5: Run-time (seconds) to build a model with different bin size ($f$) .

## IV. Conclusion

In this work, we developed a medication adherence monitoring system using a smartwatch, stored data in AWS S3, preprocessed the data and applied a random forest algorithm using Apache Spark. The experiment results showed a recall of 1.00 and a precision of 0.80 with a bin size ($f$) of 40 using 1,230 features. While few activities with a higher variation in sensor readings were misclassified, texting, an activity with a low variation, was not misclassified.
In addition to the biosensors, the smartwatch is equipped with an NFC which establishes communication and exchanges data between two electronic devices within close proximity (about 10 cm). While our study results show that the applied algorithm could sometimes misclassify data, perhaps applying NFC sensors could enhance study results.

## References

[1] A. O. Iuga and M. J. McGuire, "Adherence and health care costs," *Risk management and healthcare policy*, vol. 7, p. 35, 2014.
[2] M.-k. Suh, T. Moin, J. Woodbridge, M. Lan, H. Ghasemzadeh, A. Bui, S. Ahmadi, and M. Sarrafzadeh, "Dynamic self-adaptive remote health monitoring system for diabetics," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. IEEE, 2012, pp. 2223–2226.
[3] K. Dorman, M. Yahyanejad, A. Nahapetian, M.-k. Suh, M. Sarrafzadeh, W. McCarthy, and W. Kaiser, "Nutrition monitor: a food purchase and consumption monitoring mobile system," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2009, pp. 1–11.
[4] C. Chen, N. Kehtarnavaz, and R. Jafari, "A medication adherence monitoring system for pill bottles based on a wearable inertial sensor," in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 2014, pp. 4983–4986.
[5] H. Kalantarian, N. Alshurafa, and M. Sarrafzadeh, "Detection of gestures associated with medication adherence using smartwatch-based inertial sensors," *IEEE Sensors Journal*, vol. 16, no. 4, pp. 1054–1061, 2016.
[6] A. Spark, "Apache spark: Lightning-fast cluster computing," 2016. [Online]. Available: http://spark.apache.org
[7] A. Ho, *Step-by-step Android Wear Application Development*. Amazon Digital Services, 2015.
[8] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1383–1394.
[9] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.