Enabling Health Monitoring as a Service in the Cloud

Yang Li
Department of Computing
Imperial College London
London, United Kingdom
E-mail:yl4709@imperial.ac.uk

Li Guo
School of Computing, Engineering and
Physical Science, University of Central
Lancashire, Preston, United Kingdom
E-mail: lguo@uclan.ac.uk

Yike Guo*
Department of Computing
Imperial College London
London, United Kingdom
E-mail: y.guo@imperial.ac.uk

***Abstract*—The advancement of the Internet of Things and wireless sensors has paved the way for the development of new services for next-generation healthcare systems to enable superior communication between healthcare professionals. This paper presents key components of the Wiki-Health Analysis Framework, which enables an ecosystem to support scientists, developers, and professionals to publish their data analysis models as utilities in the cloud and allow users to access those services and utilise their collected sensor data without any expert knowledge. The feasibility of this approach is supported by an ECG-based health monitoring service application deployed in addition to the Wiki-Health platform.**

Keywords—wiki-health, analysis framework, health monitor, ECG analysis, big data, sensor data management

I. INTRODUCTION

Healthcare is shifting from the traditional reactive approach—treating problems at the crisis level with a proactive health management approach so that issues can be discovered and addressed at an early stage. Being able to monitor a longer term of the user's bio-signals allows us to understand their typical lifestyle and behaviours. More importantly, it allows for the tracking and discovery of any change-signals that could lead to potential health issues. With the evolution of technologies such as wearable health devices and smartphones, more and more patients are able to continuously self-track their bio-signals at any time and in any location.

The scale and richness of the sensor data being collected and analysed is rapidly growing. However, there are still many challenges to face. For example, health sensor devices such as ECG (Electrocardiography) employ a number of data channels and generate huge quantities of data. Consequently, visual inspection on such massive and rapidly growing data is extremely difficult. There is still a lack of support and tools for individual users to manage or utilise their collected data, such as monitoring and tracking significant changes in their health conditions. From the provider's perspective, such massive growth of health sensor data creates both data manageability and collaboration challenges. Traditional approaches of collecting, storing, querying, visualising, and analysing health sensor data are no longer able to cope.

In our previous work [1], a cloud-based personal health sensor data management platform named Wiki-Health was presented, which is designed to tackle some of the challenges in storing and analysing health sensor data. Within this study, we present a step along the road towards a proactive healthcare and wellbeing management approach by adapting the Wiki-Health platform for healthcare applications.

"Health monitoring as a service" is introduced as a key application for future personal health management. This is achieved through demonstration of the analysis model, design and implementation of an ECG-based personal health monitoring service application, and the illustration of experimental results. The proposed Adaptive Learning Approach (ALA) within the analysis model aims to reduce the training time while showing improved performance over existing methods.

II. BACKGROUND AND RELATED WORK

Traditional sensor network systems such as Aurora [2] and COUGAR [3] support a limited form of collaboration while operating on a fixed set of sensors. However, the drive toward the pervasive use of mobile phones and the rising adoption of sensing devices enabling people to collect data at any time or place is leading to a torrent of sensor data. Implementing cloud computing technologies appropriately can aid healthcare providers in improving the quality of medical services and the efficiency of operations, sharing information, improving collaboration, and managing expenditures. Sensor-Cloud infrastructure [4] enables the sensor management capability of cloud computing by virtualising a physical sensor. Commercial sensor network platforms such as Xively [5](formerly known as Cosm) have taken off in recent years. They provide an online scalable sensor data management platform that allows developers to connect devices and applications through a web-based application programming interface (API). However, none of these platforms yet provide sufficient support for data computation and analysis as services.

The growing global popularity of smartphones and tablets has resulted in new ways to gather information, both manually and automatically by means of an array of embedded sensors. Professional wearable biosensors can connect to smart phones and track significant physiological parameters. This has provided a more efficient and convenient way to collect personal health information like blood pressure, oxygen saturation, blood glucose level, pulse, Electrocardiogram (ECG), Electroencephalogram (EEG) and electrocardiography (EKG). ECG is one of the most widely used physiological signals in helping doctors diagnose various cardiovascular problems and identify abnormal levels of specific minerals in the blood. Traditionally, a standard ECG screening records the electrical activity of your heart for only a few minutes and is carried out in a hospital. When symptoms continue to occur without a definitive diagnosis obtained with a standard ECG, a Holter monitor will be used to record 24 to 48 hours of ECG signals, and then patients take the monitor back to their doctor. Efforts are being made to enable remote ECG monitoring

* Corresponding author

systems. For example, the AliveCor device [6, 7] is one of a growing number of health sensors that not only allow users to record and view medical grade single-channel ECG signals on a smartphone/tablet, but also to upload and share their recordings directly with their physicians or cardiologists through the internet.

Considerable research has already been undertaken on various scopes for ECG data analysis such as QRS complex detection [8, 9] and beat classification algorithms [10-15]. Many of the scopes apply supervised-based learning methods on features obtained from RR-interval and QRS complex for identifying cardiac abnormalities. They are useful at a professional level; i.e. by doctors to classify ECG beats. However, for individual users, common mistake such as improper electrode placement and variability in inter-individual morphologies of ECG signals can lead to misinterpretation of ECG examinations using the traditional supervised approach. While considerable studies [16-19] have been undertaken on adopting unsupervised/semi-supervised learning methods that aim to detect abnormal (outliers) heartbeats, there is still a lack of research on how to enable health monitoring as a service in the cloud computing and big data environment in an efficient manner, with regards to the architecture, analysis models, feedback mechanisms and performance evaluations.

III. WIKI-HEALTH

There are many forms of data that can be used to reflect a person's health, from high-resolution and large medical imaging data like X-rays and CT scans to simple numerical data of systolic and diastolic pressure measurement figures from a blood pressure device. These different formats of data can be generally classified into three types: unstructured, structured, and semi-structured. Examples of unstructured data include medical images, videos, plain text files, and PDFs. Most structured sensor data generated from healthcare devices such as blood pressure and ECG, can be represented in the form of a time-series. Being able to support user-defined information, tags and structures allows for the classifying and grouping the data in user's preferred ways. Overall, it requires the design of the system to be highly scalable, efficient, and capable of handling a huge amount of data in a variety of formats.

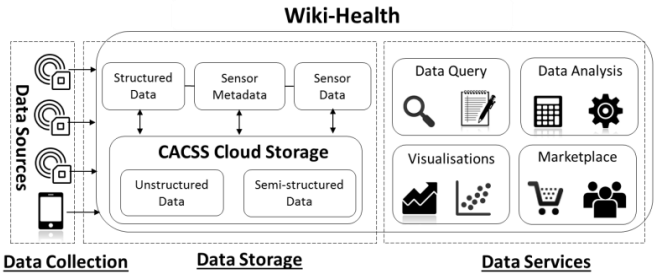

Figure 1. The architecture of Wiki-Health system.

To address the key challenges of effectively managing and making use of the rapidly growing data from health sensors, we have introduced a big data service platform, named Wiki-Health, to provide a unified solution for collecting, storing, tagging, retrieving, searching, and analysing personal health sensor data. Additionally, it also allows users to recycle and remix data, along with analysis results, and analysis models to make health-related knowledge discovery more readily available to individual users on a vast scale. The architecture of Wiki-Health is presented in Figure 1. Although we gave an overview of the entire system, in this paper only components relevant to data services will be discussed. Details of the remaining components can be found within our previous paper [20].

A. Data Storage

Wiki-Health introduces a hybrid data storage approach which aims to efficiently manage the high volume and diversity of data by storing unstructured sensor data, structured sensor data and sensor metadata separately. Sensor metadata can be thought of as descriptions of other data. It can contain background information such as type, accuracy, and location of each sensor. Sensor data refers to the actual measurements of sensors. Wiki-Health uses a relational database to store all user information, in addition to sensor metadata such as sample rate, data format, sensor type, and other Wiki-Health structured data. All sensor readings, sensor data tags, and other types of sensor data are stored in the non-relational database, such as the HBase [21, 22] and CACSS Cloud Storage System[23-25].

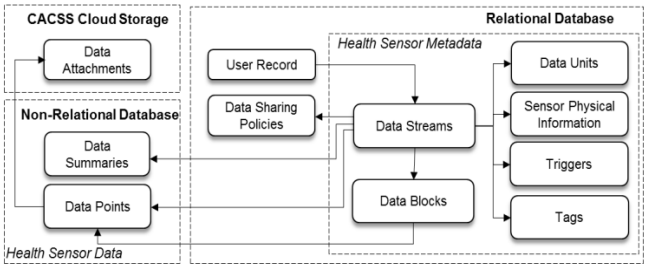

Figure 2. The hybrid data storage model.

$$\begin{bmatrix} \boldsymbol{streamid-timestampX} \\ \begin{bmatrix} v{:}\,unit\_id1 & 2.0 \\ v{:}\,unit\_id2 & 6.2 \\ v{:}\,unit\_id3 & 1.6 \end{bmatrix} \\ \begin{bmatrix} t{:}\,tag1 \\ t{:}\,tag2 \end{bmatrix} \\ \begin{bmatrix} b{:}\,blockid1 \\ b{:}\,blockid2 \end{bmatrix} \\ \begin{bmatrix} ud{:}\,userdefined1 & data \\ ud{:}\,userdefined2 & data \end{bmatrix} \\ \begin{bmatrix} \mathrm{a}{:}\,attachment1 & fileloc1 \\ a{:}\,attachment2 & fileloc2 \end{bmatrix} \end{bmatrix}$$

Figure 3. Data points storage format.

In Wiki-Health's hybrid data storage model (Figure 2), each data stream maps and holds all the information of the actual health sensor device. A single data stream can have many data units, and such units are useful for health devices that provide multiple channel readings simultaneously. For example, a data stream for a blood pressure device contains data units for systolic and diastolic pressure and heart rate measurements. Data points contain sensor reading data, data attachment index, tag mappings, block mappings, and other user-defined data. They are stored as a collection of blocks addressed by an index using the data stream ID together with a timestamp (as shown in Figure 3). Data blocks and tags are

designed and implemented for developers and users to be able to add extra dimensions to the data so that required data can be found more quickly and accurately. Data summaries can be used for storing a summary or intermediate analysis results to improve the response time on retrieving certain analysis and query results. A data stream can have multiple triggers. Triggers hold action and condition information. They are used to perform actions when a defined condition is reached. For example, such an action can be configured to alert a caregiver when a specific threshold or unusual reading is discovered.

B. Data Analysis Service

One of the goals of our system is to create an ecosystem where users can contribute their data and models. The proposed Wiki-Health Analysis Framework (WHAF) serves this purpose. WHAF supports scientists, developers and professionals to publish their data analysis models as services. Users can pay for the published models and obtain analysis results and services by applying the models to their collected data; users can sell their collected data to other parties.

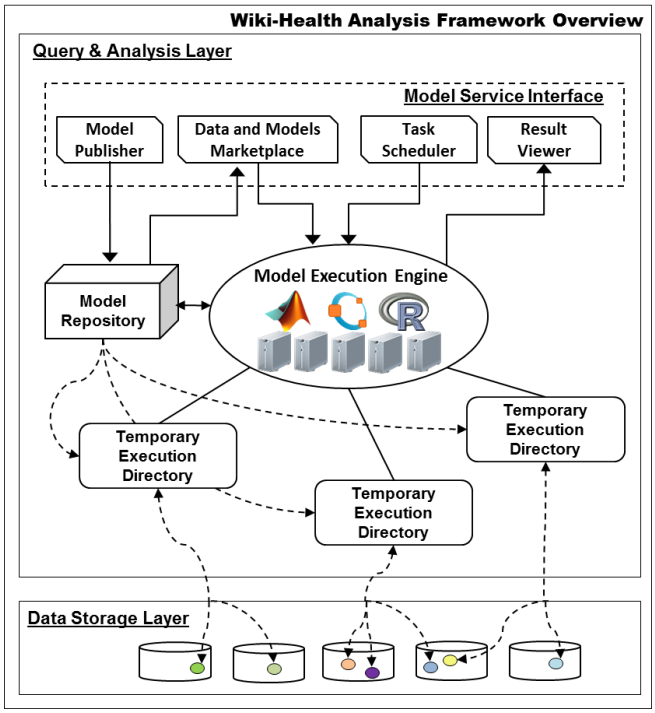

Figure 4. Architecture of Wiki-Health Analysis Framework.

The data analysis service consists of the model execution engine, model service interface and model repository. Model service interface contains model publisher, data and models marketplace, task scheduler, and result viewer. Figure 4 illustrates the overview of the Wiki-Health Analysis Framework. To follow, the components are defined at length in the following paragraph.

Model Service Interface

- Model Publisher – provides both web page and API access for model developers and experts to publish models as services. To publish a model, developers need to describe what the model service will do; specify required permissions for accessing user's data; terms and conditions of the model service; upload all the function files for the model to execute; define the formats of all the input and output parameters for the Main Function; set the price and how it is charged to the user's usage. Currently, we have implemented pay-per-use and one-off pricing models for the proof of concept.
- Data and Models Marketplace (DMM) – serves two main purposes: the first is data commodity exchange. It enables end users as data sellers to share their collected data at a price they set and get paid by selling it. Data buyers who are interested in making use of the data, such as pharmaceutical companies, healthcare providers, insurance companies and research institutions can see a random sample of the data before it is purchased. The other purpose is models as services trading. Developers and experts, as sellers who create and publish their developed analysis models and algorithms in the marketplace. End users, as buyers who can easily connect their collected data to ready-made models. In such a way, users can obtain professional data analysis services rapidly and without any programming or data analysis knowledge.
- Task Scheduler – is a component that provides the ability to schedule the launch of model execution tasks to the Model Execution Engine. It allows developers and users to schedule tasks to run periodically on a given schedule.
- Result Viewer – provides an interface to allow users and developers to visualise and obtain analysis results and logs.

Model Execution Engine (MEE) – delivers services for executing data analysis models in cloud environments and preserving execution codes within existing problem solving environments such as MATLAB [26], Octave [27] and R [28]. For each model execution task, it first creates a Temporary Execution Directory (TED). Then, it retrieves the required model files from Model Repository and data through the Data Storage Layer, placing them all in the TED. Next, it sets TED as the working directory and start executing the Main Function that is used as the entry point for each model. The user declares the filename, inputs, outputs, and actionable information of the Main Function during the model publishing process. Within the Main Function, external functions and subroutines can be called and embedded for performing computations. Users and developers are able to configure the integration and connections between data sources to the model, such as the mappings of different data streams to the inputs and outputs. The Main Function provides the ability for developers to easily migrate their existing local data analysis functions and subroutines to the Wiki-Health Analysis Framework. All the

output and intermediate files are kept in the TED. Finally, MEE conducts the corresponding actions such as persisting output files and data through Data Storage Layer to the underlying databases and trigger message alerts to the user.

Model Repository – provides an environment to store analysis models and associated files. Model files are uploaded through Model Publisher into the Model Repository, storing all user and system defined functions, models and scripts for reuse of data and knowledge.

C. *Service and Business Model*

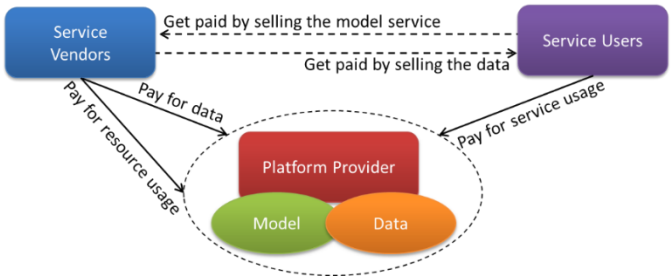

Figure 5: Service and Business Model.

Extensive research has been performed to investigate business and pricing models for different levels of cloud services. Figure 5 shows the service delivery model of Wiki-Health. Three parties are involved in the service model: service vendors, service users and platform provider. Service vendors develop analysis models and publish them as services to be deployed by the platform provider. The platform provider hosts the infrastructure for storing all of the data, offering computational resources for executing data analysis services, metering all the resource and service usage, managing data sharing and permission controls, and providing the marketplace for trading services and data. Service vendors generate revenue by selling the analysis services to the service users. They are billed by the platform provider for the computational resource usage of published data analysis services, such as the virtual machine running time and network usage. Service users are able to sell their collected data to the service vendors who are interested in making use of the data.

IV. Data Analysis Method

Figure 6 illustrates the analysis workflow of proposed the ECG-based health monitoring service, containing three mechanisms: personal profiling, global training and personal abnormal detection. The personal profiling mechanism takes the user's previous ECG signals and builds a Personal ECG Model (PEM) for each user. Global training mechanism will take only the data containing expert diagnosed abnormal ECG signals and build a Global Model of Abnormalities (GMA). Through monitoring newly measured ECG data, the personal abnormal detection mechanism first compared new data with pre-trained PEM to detect if there are any suspicious signals that are significant different from prior ECG readings, i.e. outliers. If so, it extracts these suspicious ECG signals and passes them into the GMA to try to classify the closest cardiac abnormality. The raw data of suspicious ECG signals together with classification are then sent to health professionals for visual inspection and feedbacks. Feedbacks include classifying the suspicious ECG signals as usual signals and update the PEM for false positive alerts or call in the patient for further tests to diagnose if suspicious signals can be signs of issues.

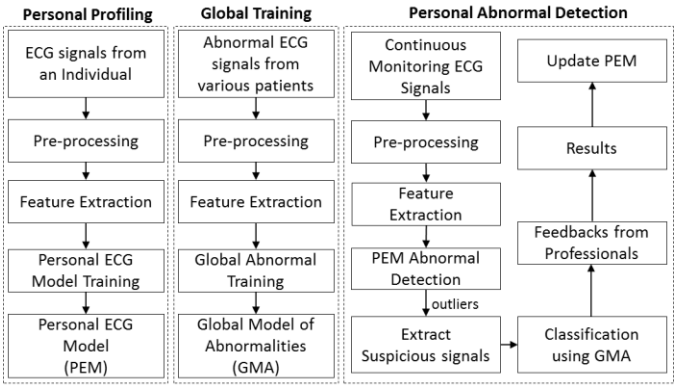

Figure 6. The workflow of the proposed health monitoring service

A. *Data Pre-Processing*

All the data is pre-processed by moving average filter and applying wavelet transformation to remove noise and eliminate baseline drift (as shown in Figure 7).

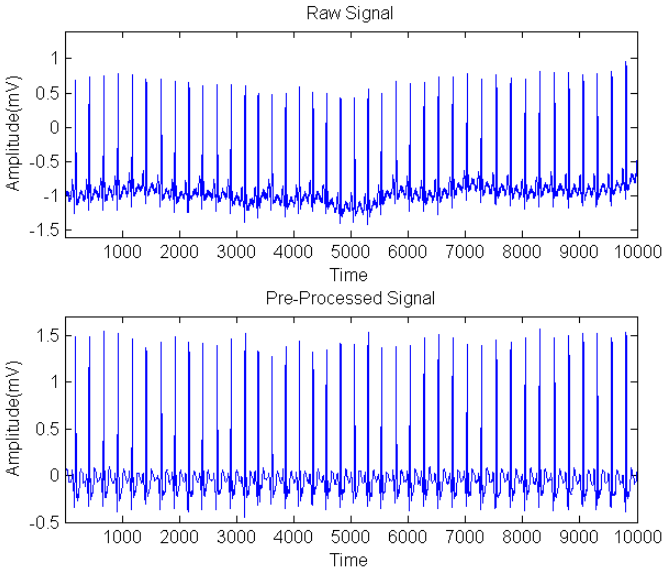

Figure 7. Sample beat waveforms of raw ECG signals and pre-processed signals.

B. *Beat blocks*

First, R-peaks are detected from the long ECG signals using the implementation of the ECG QRS detection algorithm by Pan and Tompkins[8]. Then, we adapt a beat block window to split the samples into beat blocks with R-peak centred in each block. The duration, amplitude, and shape of the QRS complex are vital in diagnosing disease conditions. A significant change in QRS complex can indicate something has happened that affects the beating of the heart. The size of the beat block window is set as 0.6 seconds, which should capture most of the information from a single heartbeat cycle (as shown in Figure 8).

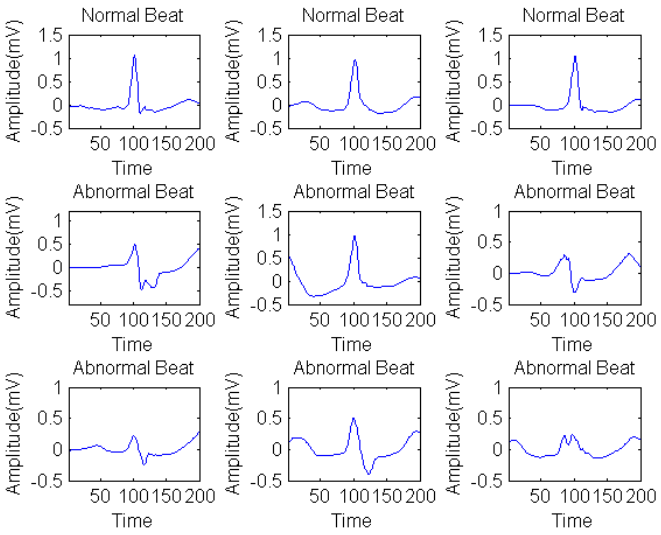

Figure 8. Samples of normal and abnormal beat blocks.

C. Features

In this work, it has been determined that we explore the combination of signal and spectral features extracted from the beat blocks for classification. For each beat block, signal features are calculated by neglecting the time t coordinate entirely and consider the normalized signal $x(t)$ as a distribution over x. The signal data from the first channel of the MIT-BIH database from Physiobank [29, 30] is used for designing and testing the analysis model. It was obtained by placing the electrodes on the chest, i.e. modified limb lead II (MLII). For consistency of comparison, signal bin boundaries are set between -0.3 and 0.5 which should be able to capture most amplitude distribution of QRS complex for MLII signals. We current use 60 linearly spaced bins for signal features. For other lead types, different sample rates, or changes to the window used for features, a different range might be required.

In each ECG beat block, every peak produces a significant response in frequency domain. Spectral features are calculated by applying the one-sided discrete Fourier transform (ODFS) of each beat block.

$$\mathrm{DFT(f)} = \hat{\mathrm{F}}(\omega_j) = \sum_{t_i=t_{min}}^{t_{max}} \mathrm{f}(t_i) e^{-2\pi i t_i \omega_j} \qquad (1)$$

Where f is the original signal, t_i are the time points at which the ECG signal is sampled, ω_j is the frequency in HZ. Compared to the commonly used fast Fourier transform (FFT), ODFS offers the ability to adjust frequency resolution and frequency bands, independently on the resolution of the signal. If the signal is real, ODFS will omit the redundant parts of negative frequency spectrum.

We apply the one-sided DFT with a pass band of [0,20] Hz. Typical signals contain relatively few peaks, so the high-frequency information is mostly noise. The computed spectrum of $\hat{\mathrm{F}}$ has 40 complex coefficients. We use both their real and imaginary parts as features, giving us 80 spectral features.

D. One-Class Support Vector Machine

One-Class Support Vector Machine (OCSVM) [31] is a novelty detection method that attempts to find a hyperplane in the feature space corresponding to a kernel function that separates data points from the origin and maximises the distance from this hyperplane to the origin. Different from classical SVM, it uses two classes for training. OCSVM only requires data from one class in the training stage and is able to classify if newly encountered data belong to the same class or not. Therefore, it is used for detecting personal abnormal or unusual beats. Consider a single class training set $x_i \epsilon \Re^m, i = 1, \dots, n$, each x_i is associated to a class label $y_i \epsilon \{+1\}$, n is the number of training instances, the objective function of the OCSVM model can be written as follows:

$$\min \quad \left(\frac{1}{2} w^T w - \rho + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \right) \qquad (2)$$

$$s.t. \quad (w \cdot \phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \text{ for all } i = 1, \dots, n$$

ϕ is the mapping function to map the sample vectors to a higher dimensional feature space. ν is the trade-off parameter to specify the fraction of outlier allowed. By setting the parameter $\nu \epsilon (0,1)$, it gives us a way to adjust the trade-off between over fitting and generalization. ξ_i are introduced slack variables relax the optimality constraints. w and ρ are hyperplane parameters which we want to compute to give the minimization of the objective function (2). Then, the decision function $f(x) = sgn(w \cdot \phi(x) - \rho)$ will be positive for most training points x_i.

E. Adaptive Learning Approach (ALA) for PEM

Suppose if we are only using one classifier, when there is new training data arriving, the traditional way is to add new data to the existing training data and to re-train the model using all of the data. The training time often increases linearly as the size of training samples grows. Thus, this is not feasible for near real-time analysis or large-scale data. Therefore, we propose a simple and effective Adaptive Learning Approach (ALA) for PEM. ALA constructs a cluster of weighted classifiers. Instead of re-training one model with all the existing data, ALA adds new classifiers that are trained using only the new data, and updates weights of classifiers based on the previous prediction results, i.e. feedbacks from doctors. The prediction is accomplished by taking the votes from all the available classifiers from the cluster.

A person's health status might also change over time caused by various factors such as alcohol consumption, diet, tobacco smoking, medications, and pregnancy. Such temporal evolution features are important to identify the correct event, stimulus or cause associated with the corresponding model or sequence of the sensor data. Integrating ALA with Wiki-Health not only provides the computational capabilities of training and make predictions, but also allows the trained models and additional information such as prediction logs, to be persisted, tagged, searched, and reused for any future studies.

The parameters and the procedures of ALA are defined as the following:

TABLE 1. PARAMETERS AND SEMANTICS OF ALA

Parameters	Semantics
H	A cluster of trained and weighted OCSVM classifiers: $H = \{w_1h_1, \dots, w_Th_T\}$
T	The size of cluster H
h_t	A trained OCSVM classifier from H where $h_t \epsilon H, t\epsilon[1,T]$
w_t	Weight for classifier h_t
α_t	Total number of correct predictions by h_t
λ_t	Total number of samples trained by h_t
ϑ	ϑ is the maximum number of training samples that a single OCSVM classifier can take
β	Minimum number of samples to be reached before removing a classifier
γ	Accuracy threshold for removing the classifier
x^i	Feature vectors, $x^i \epsilon\, \mathcal{X}$
y_i	Binary label, $y_i \epsilon \{-1, +1\}$
L	Last training set which h_T used to train.

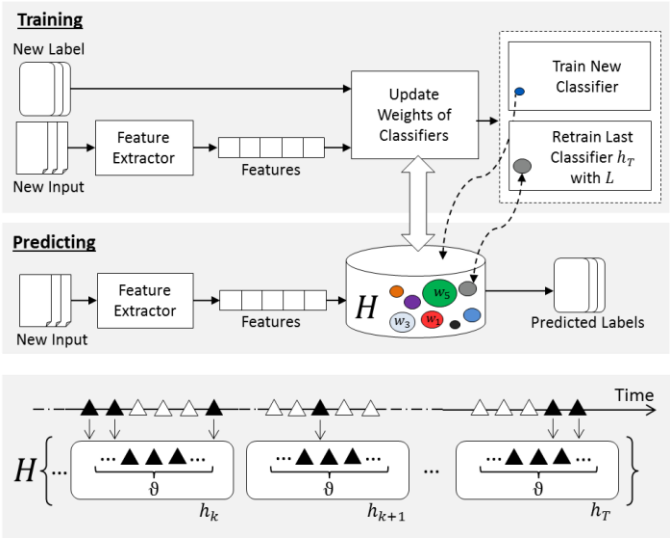

Figure 9. The overview of ALA for a long-term health monitoring service.

Procedure 1 illustrates how ALA updates the PEM model. When new training data arrives, it goes through every classifier from H and updates the corresponding weight. Numbers of correct predicted total predicted times are stored and updated. The weight is increased with more correct predictions and decrease with incorrect ones. Constants such as β and γ are introduced to eliminate "out-of-date" classifiers from H; i.e. no longer able to make good predictions. A boundary ϑ is configured to limits the total number of training samples a classifier can train and also decides when to train and add a new classifier. To make the procedure simple to understand, we only demonstrate the pseudo code for a single input of x and y; it can be easily modified for batch inputs of x and y.

Procedure1: Update PEM Model
Inputs: x, y

x Training features
y Binary label corresponding to x

1. **Begin**
2. T=**sizeof (**H**)**
3. **for** t=1:T
4. *prediction*=model_predict(h_t, x)
5. **if** (*prediction*== y)
6. α_t=α_t+1
7. **end**
8. λ_t= λ_t+1
9. **set** w_t= α_t/ λ_t
10. **if** (w_t<γ and $\beta \leq \lambda_t$)
11. remove $H(t)$
12. **end if**
13. **end for**
14. **append** x and y to L
15. **if** (sizeof (L) == ϑ)
16. **set T=T+1**
17. train model h_T with L
18. **add** h_T to H
19. **set** λ_T=sizeof(L), w_T= *eps*, λ_T=0, α_T=0, L={} //eps is floating-point relative accuracy
20. **end if**
21. **return** H
22. **End**

Procedure 2 illustrates how ALA is used to detect outliers. Only positive weighted classifiers are elected to vote for the results. The voting power depends on the weight of each classifier. The sign of the combination votes decides the final predictions.

Procedure2: Outlier Detection (Voting by classifiers from H)
Inputs:
x Features for prediction

23. **Begin**
24. T=**sizeof (**H**)**
25. **for** t=1:T
26. *local_prediction*= model_predict(h_t, x)
27. *votes*= *votes* + $w_t \times$ *local_prediction*;
28. **end for**
29. *predictions* = sign(*votes*);
30. **return** *predictions*
31. **End**

F. *Classification for GMA*

K-Nearest Neighbor (*K*-NN) algorithm is used as the classifier for the global training and classifications for GMA. To determine the nearest neighbour, various distance measures can be used. For example, between two points a and b, with k dimensions, these distances can be calculated according as below:

$$\text{Euclidean distance} = \sqrt{\sum_{j=1}^{k}(a_j - b_j)^2}$$

$$\text{City Block distance} = \sum_{j=1}^{k}|a_j - b_j|$$

$$\text{Correlation distance} = 1 - \frac{\sum_{j=1}^{k}(a_j-\overline{a})(b_j-\overline{b})}{\sqrt{\sum_{j=1}^{k}(a_j-\overline{a})^2}\sqrt{\sum_{j=1}^{k}(b_j-\overline{b})^2}}$$

V. Implementation

Figure 10 shows the overview of the proposed health monitoring service application deployed on top of Wiki-Health platform. An Android mobile app is developed as part of the Wiki-Health platform. The application supports data collection with an array of embedded sensors including accelerometers, gravity sensors, gyroscopes, GPS, and also external sensors such as ambient air temperature, heart rate, and ECG. It also allows users to tag and enter additional information about their activities during specific hours of a selected date. Such information is annotated and linked to the collected sensor data. The app is used to upload the data, trigger alerts and interact with feedback from the Wiki-Health through the platform API.

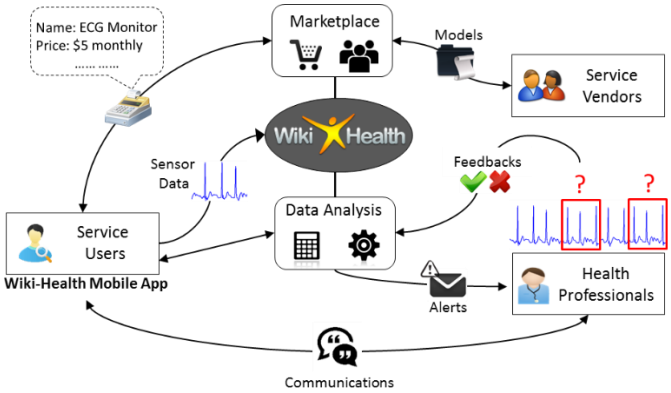

Figure 10. Implementation of ECG-based health monitoring service with Wiki-Health platform.

Model Execution Engine is currently implemented using Java and Octave [27]. Octave is a high level programming language that is most compatible with MATLAB – one of the most commonly used programming language in many domains such as bioinformatics, physics, and chemistry. All data processing functions and analysis models presented in the paper are developed in MATLAB [26]/Octave programming language. LIBSVM library [32] is used for implementing the OCSVM. Task Scheduler is currently implemented using Java Quartz Scheduler [33]. All the tasks are defined using time-based CRON expressions.

The executing function files are uploaded into the Wiki-Health's model repository and published as a health monitoring service in the data and models marketplace. Task Scheduler is used to launch different tasks based on pre-defined schedules, such as detecting and labelling R peak for new data every 2 minutes, detecting personal abnormalities for R peak labelled data every 5 minutes, and share suspicious signals and predicted labels with doctors every 6 hours.

For users, the enabling of a personal health monitoring service with Wiki-Health requires almost no knowledge about the computation. The user only needs to select the service from the data and models marketplace and configure the mappings of inputs and outputs, such as selecting the ECG data stream to be used for the health monitoring service and entering the email address of the doctor for triggered alerts.

All Wiki-Health components outlined by this paper are deployed across different virtual machines (VMs) on top of IC-Cloud [34, 35]. IC-Cloud is a generic IaaS cloud-computing infrastructure. It allows for the rapid design and deployment of a cloud environment in a flexible manner. Data analysis computational tasks are currently allocated on the VM level [36, 37].

VI. Experimental Evaluation

This section first describes an experiment used to test the performance of our proposed analysis method and features, and then it evaluates the effectiveness of ALA from different aspects, and finally it accesses the performance of ECG classifications using the global model of abnormalities (GMA).

ECG records from the MIT-BIH database [29, 30] are used for the experiments. Each record contains two channels of 30-minute recordings, which were digitised at a sample rate of 360 samples per second. There are approximately 650,000 samples per record per channel. The signal data from the first channel (modified limb lead II) is used. The entire database is separated into two groups. The first group of 23 records were used for the evaluation of *Part A* and *B*. The remaining records are assigned as the second group, which is used as a training set for GMA in *Part C*.

True positive (*TP*) refers to the number of abnormal ECG beats that are correctly detected. True negative (*TN*) refers to the number of normal beats that are correctly labelled. The following statistical parameters are used to compare the detection algorithms:

- $Classification\ Rate = \frac{TP+TN}{TB}$
- $Specificity = \frac{TN}{FP+TN}$
- $Sensitivity = \frac{TP}{TP+FN}$

The results were tested with different kernels, and the summary is displayed in Table 4. A 5-fold cross-validation is used for each training set to select the parameters. LIBSVM's degree variable is set to 3 for the Polynomial kernel, while the gamma variable is set to be 0.001 for Gaussian RBF kernel [38]. The trade-off parameter ν is set to be 1% for all tests (at most, 1% of the training samples being misclassified). The detailed results from Table 5 are obtained using Gaussian radial basis function kernel (RBF).

A. Using single fixed one-class SVM

The purpose of this experiment is to test the performance of the selected features with a single fixed OCSVM (SF). The first group of 48 records are split into two sets. To simulate the scenario of using a small portion of a user's typical previous ECG shapes and detecting personal abnormalities after, for each record, the first 3 minutes of the data is used for training the fixed OCSVM model and the remaining 27 minutes as the testing data.

As shown in Table 5, SF provides an average classification rate of 85.5%, specificity of 81.1%, and sensitivity of 98.1%. SF's classification rate and specificity are not as good as the sensitivity for some patients such as 116, 202, 205, 222 and 223. For ECG-based health monitoring services, sensitivity is important so that we capture and highlight more alerts than missing one. In the next experiment, we demonstrate how ALA can improve classification rate and specificity but still maintain high sensitivity based on feedback mechanisms.

B. *Effectiveness of the Adaptive Learning Approach*

To simulate the scenarios of receiving feedbacks from the professionals, we created two different cases for PEM:

- VS: using a single OCSVM as the classifier for PEM to trigger the alerts (i.e. abnormalities), these alerts are compared with the labels and the false positive ones are sent back. The PEM is then re-trained with existing normal data together with newly added false positive ones.
- ALA: using the Adaptive Learning Approach by creating a cluster of weighted OCSVMs for PEM to detect personal abnormalities. Weights are adjusted and newly trained classifiers are added to the cluster when false positive alerts are triggered. Constants ϑ and β are both set to 250, with γ set to 0.6.

Similar to the previous experiment, the first group of 23 records are also split into two sets. For each record, the first 3 minutes of the data are used for training and the remainder for testing and feedbacks. As shown in Table 5, classification rate and specificity are increased for both cases. However, the sensitivity of VS is dropped down more than 10% as the size of training samples grows. ALA is able to achieve 96.6% sensitivity.

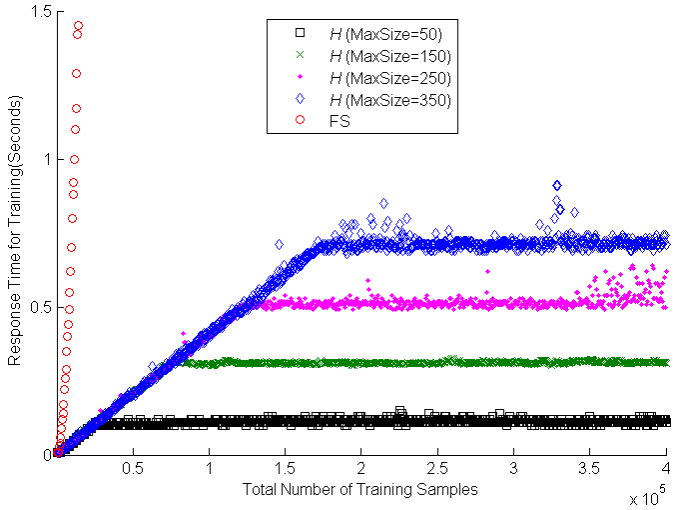

Figure 11. Relationship between system response time for training and the number of training samples.

In comparing the computation time between VS and ALA, we simulated the scenario of adding 500 training samples each time for VS and ALA. To keep experimental results consistent, all the computation was performed in the VM on top of IC-Cloud with two allocated cores and 8GB memory. Each time, a new classifier is trained and added until the size of H reaches the maximum. Different maximum sizes are used for H. The comparison of the system response time for training is shown in Figure 11. Compared to VS, ALA reduces the time of training and updating the PEM model significantly. A similar test is completed to evaluate the system response time for prediction. As shown in Figure 12, the prediction time for ALA is dependent on the size of cluster H and it increases linearly with the number of predicting samples.

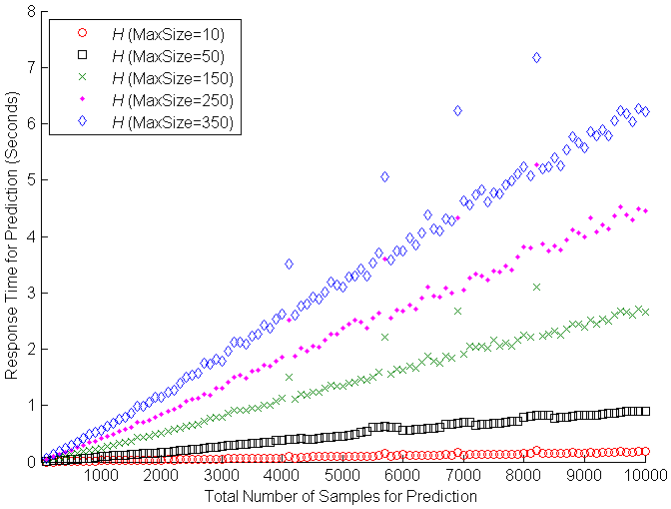

Figure 12. Relationship between system response time for prediction and the number of samples.

C. *Performance of Global Model of Abnormalities*

In this experiment, different distance metrics and values of K are used to test the performance for GMA. The following most common types of beats are extracted: atrial premature beat (A), ventricular premature beat (V) and right bundle branch block beat (R) from the second group of data to form the training data. The triggered beats from Part A are used as the test data. The results shown in Table 2 and Table 3 are obtained by applying three different distance metrics and five different values of K. The best result is obtained using City Block distance (CB).

TABLE 2. COMPARISON OF ACCURACY BETWEEN DIFFERENT DISTANCE METRICS AND VALUES OF *K* FOR GMA. ALL ITEMS ARE IN PERCENT (%).

Distance Metrics	***K=1***	***K=2***	***K=3***	***K=4***	***K=5***
City Block	82.2	72.9	71.9	72.3	71.3
Euclidean	78.5	72.7	70.2	71.4	69.9
Correlation	77.6	72.5	69.9	71.0	69.7

TABLE 3. PERFORMANCE OF GMA FOR DIFFERENT BEAT TYPES.

Beat Types	**A**	**R**	**V**	**Total**
Train Beats (Group 2)	1642	4176	1669	7487
Test Beats (Group 1)	785	2795	4666	8246
Correctly Labelled (CB)	506	2012	4258	6776
Accuracy (%) (CB)	64.5	72.0	91.3	82.2

VII. CONCLUSION

This paper presents the key components for enabling "Health Monitoring as a Service" in a cloud-based personal health sensor data management platform, named Wiki-Health. An ECG-based health monitoring service application is illustrated with a specific focus on the analysis model. The proposed Adaptive Learning Approach (ALA) within our analysis model is able to detect personal abnormalities and update the model based on received feedbacks. A comparison has been made to compare the training time required for updating the model and predicting performance. The positive

performance of the approach is supported by experimental results and shows significant potential for real-world applications.

For future research, one direction is to improve the scalability and performance of the Wiki-Health Analysis Framework by employing adaptive scaling and allocating mechanisms that distribute the analysis workload more efficiently.

REFERENCES

[1] Y. Li, L. Guo, C. Wu, C.-H. Lee, and Y. Guo, "Building a cloud-based platform for personal health sensor data management," in Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference on, 2014, pp. 223-226.

[2] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, et al., "Aurora: a new model and architecture for data stream management," The VLDB Journal—The International Journal on Very Large Data Bases, vol. 12, pp. 120-139, 2003.

[3] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," ACM Sigmod Record, vol. 31, pp. 9-18, 2002.

[4] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing," in Network-Based Information Systems (NBiS), 2010 13th International Conference on, 2010, pp. 1-8.

[5] Xively. Available: https://xively.com/

[6] AliveCor heart monitor. Available: http://www.alivecor.com

[7] C. A. Volgman, S. Wang, D. Mehta, N. Nazir, S. Alexander, K. Krishnan, et al., "O016 AliveCor Heart Monitoring: Is it a practical alternative to a traditional ECG monitor for a developing nation?," Global Heart, vol. 9, pp. e4-e5, 2014.

[8] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," Biomedical Engineering, IEEE Transactions on, pp. 230-236, 1985.

[9] D. Benitez, P. Gaydecki, A. Zaidi, and A. Fitzpatrick, "The use of the Hilbert transform in ECG signal analysis," Computers in biology and medicine, vol. 31, pp. 399-406, 2001.

[10] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," Biomedical Engineering, IEEE Transactions on, vol. 51, pp. 1196-1206, 2004.

[11] T. Ince, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of ECG signals," Biomedical Engineering, IEEE Transactions on, vol. 56, pp. 1415-1426, 2009.

[12] C. Ye, M. T. Coimbra, and B. V. Kumar, "Arrhythmia detection and classification using morphological and dynamic features of ECG signals," in Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, 2010, pp. 1918-1921.

[13] M. R. Homaeinezhad, S. Atyabi, E. Tavakkoli, H. N. Toosi, A. Ghaffari, and R. Ebrahimpour, "ECG arrhythmia recognition via a neuro-SVM–KNN hybrid classifier with virtual QRS image-based geometrical features," Expert Systems with Applications, vol. 39, pp. 2047-2058, 2012.

[14] Z. Dokur and T. Ölmez, "ECG beat classification by a novel hybrid neural network," Computer methods and programs in biomedicine, vol. 66, pp. 167-181, 2001.

[15] M. Engin, "ECG beat classification using neuro-fuzzy network," Pattern Recognition Letters, vol. 25, pp. 1715-1722, 2004.

[16] P. Li, K. L. Chan, S. Fu, and S. M. Krishnan, "An abnormal ecg beat detection approach for long-term monitoring of heart patients based on hybrid kernel machine ensemble," in Multiple Classifier Systems, ed: Springer, 2005, pp. 346-355.

[17] B. Babusiak and M. Gala, "Detection of Abnormalities in ECG," in Information Technologies in Biomedicine, ed: Springer, 2012, pp. 161-171.

[18] M. C. Chuah and F. Fu, "ECG anomaly detection via time series analysis," in Frontiers of High Performance Computing and Networking ISPA 2007 Workshops, 2007, pp. 123-135.

[19] A. Lourenço, H. Silva, and C. Carreiras, "Outlier detection in non-intrusive ECG biometric system," in Image Analysis and Recognition, ed: Springer, 2013, pp. 43-52.

[20] Y. Li, L. Guo, C. Wu, C.-H. Lee, and Y. Guo, "Building a Cloud-Based Platform for Personal Health Sensor Data Management," presented at the IEEE-EMBS International Conferences on Biomedical and Health Informatics, Valencia, Spain, 2014.

[21] A. HBase. Available: http://hbase.apache.org/

[22] A. Khetrapal and V. Ganesh, "HBase and Hypertable for large scale distributed storage systems," Dept. of Computer Science, Purdue University, 2006.

[23] Y. Li, L. Guo, and Y. Guo, "CACSS: Towards a Generic Cloud Storage Service," presented at the CLOSER, 2012.

[24] Y. Li, L. Guo, and Y. Guo, "An Efficient and Performance-Aware Big Data Storage System," in Cloud Computing and Services Science, ed: Springer, 2013, pp. 102-116.

[25] Y. Li, L. Guo, A. Supratak, and Y. Guo, "Enabling Performance as a Service for a Cloud Storage System," presented at the 7th IEEE International Conference on Cloud Computing, Alaska, USA, 2014.

[26] I. MathWorks, MATLAB: the language of technical computing. Desktop tools and development environment, version 7 vol. 9: MathWorks, 2005.

[27] Octave. Available: http://www.gnu.org/software/octave/

[28] R. C. Team, "R: A language and environment for statistical computing," 2012.

[29] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, et al., "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," Circulation, vol. 101, pp. e215-e220, 2000.

[30] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," Engineering in Medicine and Biology Magazine, IEEE, vol. 20, pp. 45-50, 2001.

[31] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Neural computation, vol. 13, pp. 1443-1471, 2001.

[32] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, p. 27, 2011.

[33] Quartz Scheduler. Available: http://quartz-scheduler.org/

[34] L. Guo, Y. Guo, and X. Tian, "IC cloud: a design space for composable cloud computing," in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 394-401.

[35] Y.-K. Guo and L. Guo, "IC cloud: Enabling compositional cloud," International Journal of Automation and Computing, vol. 8, pp. 269-279, 2011.

[36] H. Sijin, G. Li, G. Yike, W. Chao, M. Ghanem, and H. Rui, "Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning," in Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on, 2012, pp. 15-22.

[37] H. Sijin, G. Li, M. Ghanem, and G. Yike, "Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models," in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 574-581.

[38] B. Scholkopf and A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond: MIT press, 2001.

TABLE 4. COMPARISON OF OVERALL PERFORMANCE OF PEM USING DIFFERENT KERNELS. ALL ITEMS ARE IN PERCENT (%).

	Classification Rate			Specificity			Sensitivity		
	FS	*VS*	*ALA*	*FS*	*VS*	*ALA*	*FS*	*VS*	*ALA*
Linear	81.5	95.2	89.5	79.3	97.3	88.6	95.8	85.0	94.7
Polynomial	86.7	96	92	85.7	98	92	94.2	83	93
Gaussian	84.5	96.3	91.8	82.6	98.5	91.1	97.2	85.2	96

TABLE 5. COMPARISON OF PERFORMANCE OF FS, VS AND ALA USING GAUSSIAN RBF KERNEL.

Record			Classification Rate (%)			Specificity (%)			Sensitivity (%)		
ID	*Normal*	*Abnormal*	*FS*	*VS*	*ALA*	*FS*	*VS*	*ALA*	*FS*	*VS*	*ALA*
116	2066	105	73.9	98.6	76.5	72.6	98.5	75.3	100.0	100.0	100.0
119	1386	494	93.3	99.1	93.6	90.9	98.8	91.3	100.0	100.0	100.0
201	1422	408	88.5	96.8	94.3	86.1	98.4	94.2	96.8	91.2	94.9
202	1849	82	29.6	97.3	83.3	26.5	98.5	83.3	100.0	69.5	82.9
203	2276	468	91.4	96.6	92.9	90.3	97.1	91.8	97.0	94.2	98.1
205	2304	98	74.1	97.6	89.9	73.0	98.0	89.5	100.0	88.8	99.0
208	1430	1282	92.8	94.7	94.8	88.7	97.2	92.4	97.4	91.8	97.4
209	2325	403	94.8	85.7	94.9	95.4	98.5	96.0	91.1	11.4	88.8
210	2176	223	93.2	98.1	95.5	92.5	98.6	95.1	100.0	92.8	99.6
212	793	1682	99.5	99.7	99.5	98.4	99.2	98.4	100.0	99.9	100.0
213	2361	601	84.9	94.7	93.6	83.0	99.2	94.5	92.3	77.0	89.7
215	2881	149	96.4	98.9	97.3	96.2	98.8	97.2	100.0	100.0	100.0
217	242	1806	93.0	99.1	93.0	40.5	93.4	40.5	100.0	99.8	100.0
219	1865	212	87.8	98.5	91.9	87.3	99.1	91.4	91.5	92.9	96.7
220	1747	109	54.1	98.0	85.3	51.2	98.1	84.5	100.0	96.3	98.2
221	1831	373	98.5	99.4	98.3	98.1	99.3	98.0	100.0	100.0	100.0
222	1799	556	56.2	79.3	70.6	44.4	95.8	67.2	94.2	26.1	81.7
223	1795	573	79.6	94.0	90.2	73.4	98.5	88.5	98.8	79.9	95.3
228	1540	365	97.6	98.3	96.9	97.3	98.2	96.4	99.2	98.6	99.2
230	2024	191	97.9	99.1	95.4	97.7	99.0	95.0	100.0	100.0	100.0
231	302	1122	97.8	98.9	98.0	89.7	95.0	90.7	100.0	100.0	100.0
233	2021	814	98.3	99.3	98.5	98.0	99.4	98.0	99.1	99.1	99.8
234	2424	55	93.9	99.0	96.0	93.8	99.0	95.9	100	100	100
Average			85.5	96.5	92.2	81.1	98.1	88.9	98.1	87.4	96.6