A Cloud-based Architecture for Condition Monitoring based on Machine Learning

Fernando Arévalo, Mochammad Rizky Diprasetya, Andreas Schwung
Department of Automation Technology
South Westphalia University of Applied Sciences, Germany
Email: {arevalo.fernando,schwung.andreas}@fh-swf.de, mochdiprasetya@outlook.com

***Abstract*—In the framework of the digitalization of the industry, there is an increasing trend to use machine learning techniques to assess condition monitoring, fault detection, and process optimization. Traditional approaches use a local Information Technology (IT) framework centralized in a server in order to provide these services. Cost of equipment and IT manpower are associated with the implementation of a condition monitoring based on machine learning. Nowadays, cloud computing can replace local IT frameworks with a remote service, which can be paid according to the customer needs. This paper proposes a cloud-based architecture for condition monitoring based on machine learning, which the end-user can assess through a web application. The condition monitoring is implemented using a fusion of classification methods. The fusion is implemented using Dempster-Shafer Evidence Theory (DSET). The results show that the use of DSET improves the overall result.**

***Index Terms*—Cloud-based Architecture, Condition Monitoring, Machine Learning, IaaS, MQTT, OPC-UA, DSET**

I. Introduction

Nowadays, an Industrial PC (IPC) is used as a monitoring system, which possesses a data analytics software for the industrial process. This IPC could be a dedicated PC, which is separated from the automation system. When a data analytics software uses a complex algorithm, which needs more resources, a powerful IPC is needed [1]. Often, a data analytics software relies on a software framework, which offers services such as access to technical documentation, databases, and connectivity to mobile applications and to the automation platform of the factory [2]. These services are normally contained in a local Information Technology (IT) Infrastructure. The up-front cost for an IT infrastructure can be high e.g. a suitable room, equipment cost, IT manpower, maintenance. Moreover, a user who does not have experience in setting up an IT infrastructure, needs more time to establish a running system [3], as well as to receive technical training. There is an increased use of cloud computing as an alternative for a local IT infrastructure. A cloud service enables a user to share and access her system with minimal management effort, often over the internet. It allows the user to minimize the up-front IT infrastructure cost and the maintenance cost. An Infrastructure as a Service (IaaS) can be used for a user who wants to implement her system in the cloud [4]. The user does not have to configure the IT infrastructure, since the cloud provider offers it. Machine learning is popular in literature for system monitoring, e.g. to detect abnormalities in credit card fraud [5] and fault detection for industrial machinery [6]. Information fusion has been used to improve the results of machine learning methods [7] using Dempster-Shafer and Yager rule of combination. These solutions are implemented in local IT infrastructures.

The main contribution of this research is the use of an information fusion methodology to combine the strength of different machine learning algorithms in a cloud environment, in order to enhance the performance of the condition monitoring system. The information fusion methodology is presented in section V. The fusion results are compared with the results of individual machine learning classifiers in section VI. A second contribution of this paper is the definition of a software development methodology for a web application, see section IV. The software development uses an IaaS cloud computing model for the web application. An IT framework is developed in the cloud service, which provides the aforementioned services such as database, web application, a data analytics module, and connectivity to the automation platform. The information fusion algorithm is implemented in the data analytics module. The results are displayed in the web application for the end user.

II. Related Work

Ferrer et al. [8] explain web-based IoT devices and cloud based manufacturing platform. To achieve the internet of things concept, an Information and Communication Technologies (ICT) framework is implemented. ICT helps machines, sensors, embedded devices and software components to be connected with each other, in order to use the information for monitoring purposes. Cloud Collaborative Manufacturing Networks (C2NET) are implemented for the cloud-based platform. The C2NET architecture uses WebSockets, Message Queue Telemetry Transport (MQTT) and RESTful API for the connection between the IoT devices and the cloud-based platform. Elazab et al. [9] designed a Cloud Based Condition Monitoring (CBCM) system for power plants. The application was based on an IaaS model and a LAN connection for the data collection. Zhen-wu et al. [10] developed a cloud infrastructure for industrial applications based on OpenStack. An automation platform is integrated with the cloud infrastructure.

Machine to Machine (M2M) communication enables the interaction between different machines. One proper way to establish M2M communication is through IoT. Domenech et

978-1-5386-4829-2/18/$31.00 ©2018 IEEE

al. [11] designed a smart industrial environment using IoT. The M2M communication is implemented through a Representational State Transfer (REST) Application Programming Interface (API) over HTTP. Low et al. [12] designed a Machine Learning and Data Mining (MLDM) application based on the GraphLab framework for the prediction of movie ratings and Named Entity Recognition (NER) using web content. It requires large-scale computing and storage resources to execute the MLDM algorithm. In this case, an IaaS cloud computing model is chosen as its main processor for the MLDM algorithm.

III. A Cloud-based Architecture for Condition Monitoring using a CoreSys SW Framework

The completion of this research comprises the understanding of different topics such as software framework, cloud computing, communication protocols and development of end-user applications.

A. The CoreSys Framework

The CoreSys framework is designed to include different IT services for supporting the end-user, see Fig. 1. CoreSys is designed to work as a local IT Infrastructure. The main modules of CoreSys are: Machine Learning and Data Analytics (MLDA), the Apache Server, Historical Data, and Knowledge Modeling. The purpose of MLDA is to provide functions for fault detection assessment and condition monitoring. The historical data serves for training and testing models, as well as to run data analytics. CoreSys collects data from the automation platform through data communication protocols.

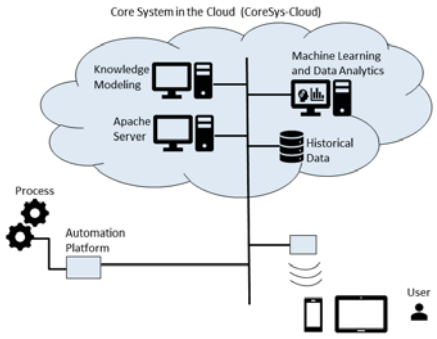

Fig. 1. CoreSys Project

CoreSys was designed initally to run as a local system in an IPC. Nevertheless, given the increasing trend in the industry to use cloud computing, a cloud framework CoreSys-Cloud is proposed. The web application allows the end-user to analyze data using machine learning techniques, which can be visualized through a PC or a smartphone through the web service. All the modules of the CoreSys-Cloud are implemented directly in the cloud server. Flask is a framework for designing the web application for the end-user. It uses a REST mechanism for the communication between the server and the client. The database management system MySQL stores the data, which will be used by the web application.

B. Condition Monitoring based on Machine Learning

The condition monitoring is implemented in the MLDA module of CoreSys-Cloud using machine learning techniques. The condition monitoring uses classification methods to predict the machine status based on its input variables. Dempster-Shafer rule of combination is used as an information fusion method, in order to combine the results of the classification methods. This fusion improves the overall performance of the system. The information fusion results are stored in the database of the CoreSys-Cloud, which are displayed afterwards in the web application.

C. IT Technology

1) Cloud Computing: There are three basic models in cloud computing [4]: Infrastructure as a Service (IaaS), Software as a Service(SaaS), and Platform as a Service (PaaS). The IaaS model offers hardware, software, server, security, storage, and maintenance. A computer software, such as Oracle Virtualbox, VMware, or Hyper-V, runs a virtual machine as a guest in a server. One server can run multiple virtual machines. The server capability maintains the stability while running the virtual machine. There are many providers that offer IaaS such as Google Cloud, Amazon AWS, and DigitalOcean. The SaaS model offers only the software. The software is developed in the cloud infrastructure. The user can only access the software. The PaaS model offers to the user: a development toolkit, operating system, web server and a database. The user can only access the development environment. She cannot access the whole infrastructure of the cloud.

2) MQTT Communication Protocol: Nowadays, a web-based internet of things is used in Industry for monitoring, analyzing, and tracking. One of the known communication protocols, which is used for web-based internet of things is the Message Queue Telemetry Transport (MQTT) Protocol [13]. MQTT is a simple publish/subscribe communication protocol that uses a message broker. The broker listens to all clients that are subscribed to it. A client can be either a publisher or a subscriber, or both. Publisher of the client publishes a message to the broker with a certain topic (broker label). When the broker updates a topic, the subscriber retrieves the message from the topic.

IV. Front-End Development

The development of this Front-End considers a Waterfall Software Development model [14]. The main stages for the software development are: planning, definition of software requirements, design, implementation, testing and maintenance. This research includes only the software requirements, design, implementation and testing stages.

A. Software Requirements

This section comprises the software specifications, which define the functionality of the application. The Front-End requirements are summarized as follows:

- MQTT data communication protocol as an interface between the the automation platform and the Back-End.

- MQTT data communication protocol shall be used between the automation platform and CoreSys-Cloud.
- Access to results of the data analytic and machine learning module.
- Access to current data of the automation platform.
- Implement the Front-End using a Web Design.

B. Design

The design is specified using Unified Modeling Language (UML) diagrams. The design is based on the information recollected in the software requirements. Two pages are developed in this project: a homepage and an automation page. The homepage consists of a short introduction about the author and a short description about the project. The homepage has a navigation bar, which has two navigation menus: Home and Automation. It is located on the header of the page. The automation page consists of a figure, which displays the result of the condition monitoring assessment.

The UML diagrams employed are the sequence diagrams, which are the front-end sequence and back-end sequence diagrams. The front-end sequence diagram is shown in Fig. 2. When the user accesses the web application, the homepage will be displayed. The web application requests the home page data from the web framework Flask. Flask retrieves the data from a MySQL database and provides the data to the homepage. The same scenario is applied when the user accesses the automation page. There is a recurrent function in the automation page, which implements this process in a predefined time.

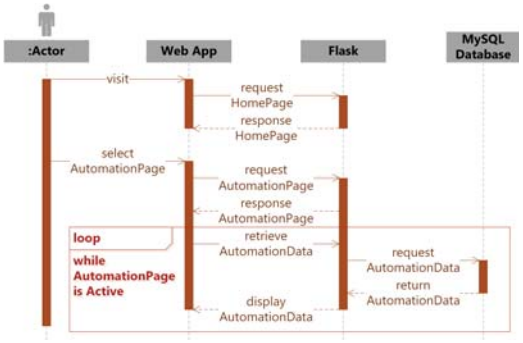

Fig. 2. Front-End Sequence Diagram

The back-end sequence diagram is shown in Fig. 3. This diagram describes the data collection between CoreSys-cloud and the automation platform. In the first step, the automation platform sends data to the MLDA module through MQTT. In the next step, MLDA analyzes the data for the condition monitoring. Then the result is sent to Flask. Flask stores the result in MySQL. This proccess is iterated in a loop, so that the data will be continuously updated in MySQL.

C. Implementation

The architecture of the integration between the CoreSys Cloud and the automation platform is shown in Fig. 4.

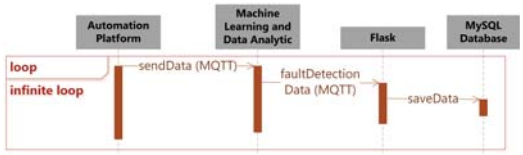

Fig. 3. Back-End Sequence Diagram

The main components are: the automation platform, the communication protocol MQTT, the framework CoreSys-Cloud, an OPC-UA/MQTT gateway, and the Web Application. The gateway enables the communication between the OPC Server of the automation platform and the MQTT broker in CoreSys-Cloud. MQTT establishes the communication between CoreSys-Cloud and the web application.

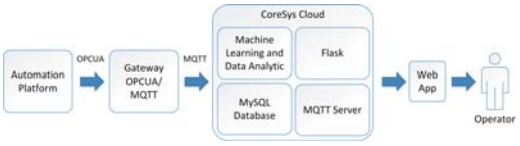

Fig. 4. CoreSys Cloud and Automation Platform

The software implementation is divided in four steps, see Fig. 5. The steps are: choosing the cloud provider, implementation of the OPC-UA/MQTT gateway, development of web application using Flask, and definition of a function for the web application.

Fig. 5. The Implementation Steps

In the first step, the cloud provider DigitalOcean is chosen. DigitalOcean is a well-known cloud provider along with Google Cloud and Amazon AWS. It has a simple cloud configuration. It automatically creates the virtual server with predefined specifications offered by DigitalOcean. The user chooses the specification that suits the needs for her application. After DigitalOcean creates the virtual server, the IP address of the virtual server is given to the user. The user can access the virtual server through a Secure Shell (SSH) session.

In this project, Ubuntu is installed in a virtual server. Linux environment is used for the project development. There are four services implemented in CoreSys-Cloud:

1) Machine Learning and Data Analythics: The machine learning and data analytics (MLDA) module is used for the condition monitoring assessment. MLDA is written in Python. The library scikit-learn [15] is used for the implementation of the machine learning functions of MLDA.

2) Flask: Flask is a software framework that is used for the web application development. The reasons for choosing Flask are its lightweight framework, and it is a Python library. This allows for an easier integration of Flask to the MLDA, since both are written in Python.

3) MySQL Database Server: The database stores the results of the MLDA assessment.

4) MQTT Server: MQTT server is used to establish a communication between the OPC-UA/MQTT gateway and CoreSys-Cloud. The MQTT server communicates with Flask. It provides an authentication method using username and password, which prevents an unknown connection to the MQTT server.

The second step is the implementation of an OPC-UA/MQTT gateway. The gateway enables the communication between the OPC server of the automation platform and the CoreSys-Cloud MQTT broker. The gateway communicates with the automation platform through OPC-UA. The third step is the development of a web application using Flask. Flask serves as an interface between the MQTT broker of CoreSys-Cloud and the MySQL database. The websites Homepage and Automation Page interact with Flask, in order to request or retrieve data. The fourth step is to create functions for the web application. The web application obtains data from the OPC-UA/MQTT gateway through MQTT. A function is called whenever the gateway publishes data to the MQTT broker. This function stores the published data to the MySQL database. A recurrent function based on JQuery is used in the automation page, in order to update continuously the results of MLDA module. This recurrent function calls a GET method in the web application through REST architecture. The GET method retrieves data from MySQL database, and sends the data to the recurrent function. The recurrent function sends the data to the website to be displayed. The database can only be accessed through the GET method. Fig. 6 shows the visualization of the Automation Page customized for the application on a Bulk Good System.

Fig. 6. Automation Page

V. Condition Monitoring based on Machine Learning

This section describes the methodology for the implementation of the condition monitoring using the MLDA module. A theoretical background for the machine learning methods is provided.

A. Preliminary

MLDA has its backbone in classification methods, as well as information fusion techniques. The selected classification methods are Random Forest, Multilayer Perceptron Artificial Neural Network, and AdaBoost. The classification methods were chosen due to good performance and wide use in research [16] [17] [18]. Dempster-Shafer is chosen as an information fusion method, due to its capability to combine different information sources [7]. A summary of each method is provided, for a deep mathematical understanding refer to [17] [18].

1) Random Forest: Random Forest method combines randomized decision trees, and accumulates their result by averaging and voting methods. Each decision tree uses a part of the features to generate a model of classification [19].

2) Multilayer Perceptron Artificial Neural Network: Normally, a Multilayer Perceptron (MLP) network consists of three layers of nodes: input layer, hidden layer, and output layer. Each layer uses a non-linear activation function. The backpropagation algorithm is used to train the network [20].

3) Adaptative Boosting: Adaptative Boosting (AdaBoost) combines many types of learning algorithms to improve the overall performance. Every learning algorithm has its own parameters and configurations. It needs to be adjusted to achieve an optimal performance. The output is combined by using a weighted sum of each algorithm [5].

4) Dempster-Shafer Rule of Combination: Dempster Shafer Evidence Theory (DSET) is an information fusion method, which allows to model uncertainty and to combine knowledge of different sources. The knowledge is defined as a mass function. Each mass function is defined as a set of evidence. The DSET rule of combination combines the mass functions of the different knowledge sources [7].

B. Fusion of Classification Methods using DSET

The prediction of a classification method is transformed into a set of evidence using:

$$m_{prediction} = [K * w \quad \frac{(1-K)}{n-1} * w \quad ... \quad \frac{(1-K)}{n-1} * w \quad (1-w)] \tag{1}$$

$$k = 1 - 10^{F} \tag{2}$$

where k is the sensitivity to zero [7] factor, F is the approximation factor, w is the confidence weight of each method and n is the number of target classes.

C. Methodology

MLDA module collects the machine data, trains the classification models, and feeds up current data into the models to obtain a prediction of the target class. MLDA uses the predicted target class for the machine condition assessment. The methodology for the implementation of the MLDA follows the steps: Data collection, data preparation, method selection and model training, and model testing. The Data Collection is divided in two types. The first type collects the data once or on a regular basis, which is stored as historical data. MLDA uses the data for training, validation and testing of the models. The second type collects data continuously and

feeds it into the trained models, in order to get the predictions. The Data Preparation consists of splitting the data into training, validation and testing data. A two-step split strategy is followed. The first step divides the data into validation(30%) and model data(70%). The second step divides the model data into training(70%) and testing data(30%).The data is scaled to improve the performance. The data features are normalized using a normalize function, which scales the features into a number between zero and one [18].

The Method Selection is assessed through a cross validation method. This method separates the training data into a predifined number of folds [17]. Each fold is divided into training and validation data. The purpose of this method is to use all the data for training and validation of the models. The method provides a performance score for each fold. An average of all performance scores for the folds is taken. This average for each classification methods serves as a metric to evaluate the performance of each method. It is used as a confidence weight in the DSET fusion. The Model Testing involves changing of parameters, and checking the performance of the methods. This includes testing all the classification methods individually, as well as the DSET fusion of the classification methods.

VI. Use Case: A Cloud-based Architecture for Condition Monitoring of a Bulk Good System

The Bulk Good System (BGS) Laboratory plant is a bulk good plant in a small scale [6]. It is a machine, which contains four stations namely loading, storage, weighing and filling. Each station can operate in standalone modus, or in a configuration of at least two stations. The automation platform of the BGS possesses a controller on each station, specifically an Industrial PC for the loading station and a PLC for the rest of the stations. Profinet serves as a communication protocol between the stations. MLDA collects the data from the BGS, trains the models, and feeds current BGS data into the models to obtain a prediction of the target classes (normal, warning, critical) of the machine. The machine assessment is then performed using current features to obtain the machine status based on the target class.

A. Experimental Data Analysis and Results

The Data Collection from the BGS extracted data in a lapse of 2 hours using a sample rate of 1s. This data is used as historical data, in order to perform the training, validation and testing of the models. Table I displays the features and classes of the BGS data, for more details regarding the dataset refer to [6]. The Data Preparation consists of splitting the data into training, validation and testing data. The features are normalized. Even though MLP presents better results using standard data, as it shows the information fusion more clearly. The Method Selection is performed through a cross validation method using 10 folds. The average of the cross validation is used as a confidence weight for each method, see Table II.

Table III shows the performance of each method and DSET fusion for each station. The station performance is divided per class. The performance indicator is built using a confusion matrix, which compares the class prediction with the test class. The station Loading shows a total performance above 83.99%. The class critical using Multilayer Perceptron presents the lowest result with 52.25%. The rest of the results are predicted with an accuracy above of 99.31%. The station Storage presents a total performance above 71.31%. The lowest result 18.05% belongs to the class critical using MLP. The rest of the results remain homogeneous with values above 95.88%. The station Weighing has a total performance above 84.78%. The lowest result for the class critical is 71.72% using MLP. The lowest results for the warning class belongs to MLP and AdaBoost with values of 82.74% and 86.76% respectively. The rest of the results presents values above 99.76%. The station Filling displays a total performance above 87.80%. The lowest result belongs to the class warning with a value of 63.91% using AdaBoost. The rest of the results remain homogeneous with values above of 94.32%. DSET fusion presents the same results as the method Random Forest for the stations Loading, Storage and Filling with values above 99.83%. The station Weighing shows a slight difference between the two approaches.

TABLE I
Features and Classes of the BGS.

Features	**Loading**	**Storage, Weighing**	**Filling**
Filling height min. connected	x	x	
Filling height max. connected	x	x	
Overflow connected	x	x	
Emergency stop active	x	x	x
Normal speed active	x		
Normal pressure active		x	x
Normal suction time active			x
Next station connection active			x
Container active			x
Classes	**Loading**	**Storage, Weighing**	**Filling**
Machine condition normal	x	x	x
Machine condition warning	x	x	x
Machine condition critical	x	x	x

TABLE II
Cross Validation of Classification Methods using Station Data

Method	**Loading**	**Storage**	**Weighing**	**Filling**
Decision Tree	99.92	99.71	99.71	99.58
Random Forest	99.92	99.71	99.71	99.62
ANN MLP	99.03	97.02	96.97	97.18
AdaBoost	75.60	97.61	96.50	96.76
Naive Bayes	99.79	95.20	87.22	94.95

B. Discussion

The results were improved using the DSET rule of combination, which relies on the combination of set of evidences with low uncertainty. The uncertainty was based on the cross validation results, as a basis to determine the confidence of each method. The confidence of each method has values

TABLE III
PERFORMANCE OF CLASSIFICATION METHODS PER STATION

Method	Normal	Warning	Critical	Total
Station Loading				
Random Forest	100.00	99.86	100.00	99.95
ANN MLP	100.00	99.72	52.25	83.99
AdaBoost	100.00	99.31	100.00	99.77
DSET Fusion	100.00	99.86	100.00	99.95
Station Storage				
Random Forest	100.00	100.00	100.00	100.00
ANN MLP	100.00	95.88	18.05	71.31
AdaBoost	100.00	100.00	100.00	100.00
DSET Fusion	100.00	100.00	100.00	100.00
Station Weighing				
Random Forest	99.88	99.76	100.00	99.88
ANN MLP	99.88	82.74	71.72	84.78
AdaBoost	99.88	86.76	100.00	95.55
DSET Fusion	99.88	99.76	100.00	99.83
Station Filling				
Random Forest	99.48	100.00	100.00	99.83
ANN MLP	99.61	96.03	94.32	96.65
AdaBoost	99.48	63.91	100.00	87.80
DSET Fusion	99.48	100.00	100.00	99.83

above 96.50%, only AdaBoost shows a value of 75.60% for the station Loading. The results of the fusion are equal or better than the other classification methods, except for slight differences in the performance of the weighing station. The results are attributed to the nature of the dataset, since there are only 5 features per station, and it does not possess complex relationships between the variables.

VII. CONCLUSION

This paper presented a cloud-based architecture for condition monitoring based on machine learning, as an alternative for a traditional condition monitoring solution based on a local IT infrastructure. A methodology was provided for the implementation of a fusion of classification methods in a cloud environment. The results proved that the Dempster-Shafer rule of combination is a suitable alternative for combining the predictions of different classification methods.

It also presented a methodology for the software development of a web application for the end-user. CoreSys-Cloud was presented as an IT framework alternative for small and middle size companies, in order to have access to services such as databases, machine learning, mobile applications and connectivity to the automation platform.

REFERENCES

[1] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010*, M. Parashar, Ed. Piscataway, NJ: IEEE, 2010, pp. 495–504.

[2] L. Wang and R. X. Gao, *Condition monitoring and control for intelligent manufacturing*, ser. Springer series in advanced manufacturing. London: Springer, 2006.

[3] T. Hegazy and M. Hefeeda, "Industrial automation as a cloud service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2750–2763, 2015.

[4] M. Moravcik, P. Segec, J. Papan, and J. Hrabovsky, "Overview of cloud computing and portability problems," in *ICETA 2017*. [Piscataway, New Jersey]: IEEE, 2017, pp. 1–6.

[5] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, "Credit card fraud detection using adaboost and majority voting," *IEEE Access*, vol. 6, pp. 14 277–14 284, 2018.

[6] F. Arevalo, T. Mohammed, and A. Schwung, "Fault detection using probabilistic prediction and data fusion on a bulk good system," in *2017 52nd International Universities Power Engineering Conference (UPEC)*. Piscataway, NJ: IEEE, 2017, pp. 1–6.

[7] S. R. Mousavi, *Dempster-shafer theory and modified rules to determine uncertainty in mineral prospection*. Clausthal-Zellerfeld: Universitätsbibliothek Clausthal, 2011.

[8] B. R. Ferrer, W. M. Mohammed, E. Chen, and J. L. M. Lastra, "Connecting web-based iot devices to a cloud-based manufacturing platform," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 10/29/2017 - 11/1/2017, pp. 8628–8633.

[9] E. Elazab, T. Awad, H. Elgamal, and B. Elsouhily, "A cloud based condition monitoring system for industrial machinery with application to power plants," in *2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*. IEEE, 2017, pp. 1400–1405.

[10] L. Zhen-wu, Z. De-min, S. Yun-tao, and S. De-hui, "Research on the architecture and application of industrial cloud experimental platform based on openstack," in *Proceedings of the the 29th Chinese Control and Decision Conference (2017CCDC)*. Singapore: IEEE Industrial Electronics (IE) Chapter, 2017, pp. 7401–7406.

[11] M. C. Domenech, L. P. Rauta, M. D. Lopes, P. H. Da Silva, R. C. Da Silva, B. W. Mezger, and M. S. Wangham, "Providing a smart industrial environment with the web of things and cloud computing," in *2016 IEEE International Conference on Services Computing*, J. Zhang and J. A. Miller, Eds. Los Alamitos, California: IEEE Computer Society, 2016, pp. 641–648.

[12] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[13] K. Grgic, I. Speh, and I. Hedi, "A web-based iot solution for monitoring data using mqtt protocol," in *Proceedings of 2016 International Conference on Smart Systems and Technologies (SST)*, D. Žagar, Ed. Osijek, Croatia: Faculty of Electrical Engineering, Computer Science, and Information Technology Osijek, 2016, pp. 249–253.

[14] E. J. Braude and M. E. Bernstein, *Software engineering: Modern approaches*, 2nd ed. Hoboken, NJ: Wiley, 2011.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] Abraham J. Wyner, Matthew R Olson, Justin Bleich, and David Mease, "Explaining the success of adaboost and random forests as interpolating classifiers," *Journal of Machine Learning Research*, vol. 18, pp. 48:1–48:33, 2017.

[17] K. P. Murphy, *Machine learning: A probabilistic perspective*, ser. Adaptive computation and machine learning series. Cambridge, Massachusetts and London, England: The MIT Press, 2012.

[18] I. H. Witten, C. J. Pal, E. Frank, and M. A. Hall, *Data mining: Practical machine learning tools and techniques*, fourth edition ed. Cambridge, MA: Morgan Kaufmann, 2017. [Online]. Available: http://proquest.tech.safaribooksonline.de/9780128043578

[19] C. Hao, S. Sivanesan, M. Majmudar, and K. S. Rajput, "Combinational feature based random forest classification for enhanced bundle branch block beat detection," in *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. IEEE, 3/4/2018 - 3/7/2018, pp. 319–322.

[20] A. Thakur and D. Mishra, "Hyper spectral image classification using multilayer perceptron neural network & functional link ann," in *Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering*, A. Bansal and A. Singhal, Eds. Piscataway, NJ: IEEE, 2017, pp. 639–642.