Towards a Software as a Service for Biodigestor Analytics

Ricardo Pieper[1], **Dalvan Griebler**[1,2], **Adalberto Lovato**[1]

[1] Bacharelado em Sistemas de Informação
Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC)
Faculdade Três de Maio (SETREM) – Três de Maio – RS – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

ricardo.pieper@gmail.com, dalvan.griebler@acad.pucrs.br,

proflovato@terra.com.br

Abstract. *The field of machine learning is becoming even more important in the last years. The ever-increasing amount of data and complexity of computational problems challenges the currently available technology. Meanwhile, anaerobic digesters represent a good alternative for renewable energy production in Brazil. However, performing efficient and accurate predictions/analytics while completely abstracting machine learning details from end-users might not be a simple task to achieve. Usually, such tools are made for a specific scenario and may not fit with particular and general needs. Our goal was to create a SaaS for biogas data analytics by using a neural network. Therefore, an open source, cloud-enabled SaaS (Software as a Service) was developed and deployed in LARCC (Laboratory of Advanced Researches on Cloud Computing) at SETREM. The results have shown the SaaS application is able to perform predictions. The neural network's accuracy is not significantly worse than a state-of-the-art implementation, and its training speed is faster. The user interface demonstrates to be intuitive, and the predictions were accurate when providing the training algorithm with sufficient data. In addition, the file processing and network training time were good enough under traditional workload conditions.*

1. Introduction

Anaerobic digestion is a process of degradation that breaks multi-molecular substances and produces a gas composed of methane, carbon dioxide and others. This gas is also known as biogas, which is often used to produce electric energy [Aslanzadeh 2014]. Research on biogas is important in developing countries, because the implementation of anaerobic digesters is often less costly than comparable renewable technologies (*e.g.* diesel-like fuels). Moreover, production of electric energy through biomass shows great potential in Brazil due to the large agricultural and livestock activity. However, poor management and lack of process control often causes efficiency losses [Labatut and Gooch 2012, Gutiérrez-Castro et al. 2015, Alves et al. 2015].

Meanwhile, cloud computing technology is helping to deliver software and infrastructure. Cloud Computing is an umbrella term that describes services characterized by their on-demand offering, per-use cost, elastic capacity and virtualized resources. Cloud

computing services are usually divided into SaaS, PaaS and IaaS. SaaS (Software as a Service) is a model of delivering applications to end users through web portals. Users can access the application through the Internet instead of using a locally installed application. IaaS (Infrastructure as a Service) distributes computational infrastructure without requiring the user to purchase physical equipment [Vogel et al. 2016a, Roveda et al. 2015a, Thome et al. 2013, Roveda et al. 2015b], and PaaS (Platform as a Service) provides a development environment on which it is possible to create applications using a predefined and abstracted technology stack [Buyya et al. 2010].

The biogas technology in the IT field is becoming even more important, and several SaaS applications are available for different contexts in the anaerobic digestion field, such as monitoring the biogas plant ([Gas Data 2015] and [Green Lagoon 2015]), evaluating the financial gains from biogas operations ([BT IT 2015]) and simulating the digestion process in laboratory ([Bioprocess Control 2015]). In addition, [Sota Solutions 2016] designed a software that uses Neural Networks to predict and optimize biogas production. Scientific publications regarding neural networks and machine learning have also shown that it is possible to use neural networks to extract information (such as biogas volume output, methane percentage and others) from biogas data [Kusiak and Wei 2014, Qdais et al. 2010]. Differently, our work seeks to provide an intuitive user interface to apply machine learning algorithms for anaerobic digester data sets.

Machine learning may help users to improve the biogas production process by performing predictions based on historical data. These predictions might suggest improvements in the biogas production process, generating more financial gains or producing more energy. However, the application of such algorithms often requires the user to have prior knowledge about programming. Also, to allow laypersons on machine learning to extract smart information from data sets, it is important to abstract the low-level details of these algorithms (training, modeling, implementing and processing). In this paper, our goal is to provide a general idea of a SaaS application for anaerobic digester data analytics. This SaaS seeks to serve users with the necessary tools without requiring knowledge on machine learning and programming. The user will be able to perform predictions for biogas or methane. Therefore, our main contributions are:

- A first version of a future SaaS application for anaerobic digester data analytics and biogas predictions.
- An user-friendly graphical interface which abstracts machine learning aspects.

This paper is organized as follows: Section 2 presents related publications in the area of machine learning applied to agricultural contexts, as well as SaaS applications for biogas and anaerobic digesters. Section 3 presents the developed SaaS application. Section 4 discusses the application's architecture and deployment. Section 5 shows the results of experiments performed with the application and with the neural network. Finally, Section 6 presents the conclusion of this work as well as future works.

2. Related Work

In this section, several scientific publications regarding machine learning applied to biogas and other agricultural contexts are going be presented. A survey of SaaS applications for biogas was performed, and its results are also presented afterwards.

2.1. Machine Learning Applied to Biogas Environment

[McQueen et al. 1995] discuss the application of machine learning methods applied to agricultural data, specifically on dairy herd culling data. The work tried to create a small decision tree that evaluates whether a given animal should be culled or not. In order to achieve good results with the C4.5 algorithm, the authors had to improve the quality of the data set, which contained several flaws regarding lack of important data along with irrelevant data (*e.g.* identification, mating date of the animal, and others). When the original data set was used, the algorithm generated a very complex decision tree, taking into account several irrelevant parameters for the problem. After improving the quality of the data and removing unnecessary features, the C4.5 algorithm resulted in a decision tree that is compact and plausible from a farming perspective.

[Coopersmith et al. 2014] evaluated whether it is possible to predict soil drying only from publicly accessible data in the United States. The authors found gaps in the literature about soil drying prediction related to information availability, mostly caused because land owners might not install soil sensors due to financial limitations, limited accessibility or technological complexity. The information used by the work is likely to become available globally from satellite sensors in near future, making the research important in order to improve information accessibility. The work uses three algorithms: Classification Trees, K-Nearest-Neighbors (KNN) and Boosted Perceptrons. While the classification trees had the worst performance with 88% accuracy, boosted perceptrons and KNN algorithms performed with 92-93% accuracy on validation data, respectively. The work shows that even in a scenario where information is limited, machine learning algorithms may help to solve problems when no other alternative is suitable.

The work of [Kusiak and Wei 2014] tried to predict methane production in a waste water treatment facility. A data set of 577 records was used to train an ANFIS - Adaptive Neuro-Fuzzy Inference System algorithm available in Matlab 10.0. Another data set with 148 records was used as a test data set. The authors also tested the K-Nearest-Neighbors algorithm. However, the authors concluded that the ANFIS algorithm yielded the best results, because the predictions were closer to the actual observations than the other algorithms, achieving a correlation coefficient of 0.99. The K-Nearest-Neighbors yielded the worst results, since the predictions do not fit the observed data very well.

The work of [Qdais et al. 2010] tried to create a neural network to optimize the output of methane gas. The data analyzed contains 177 records and several features, such as Total Solids, Total Volatile Solids, pH and and temperature. The resulting neural network was tested against another data set with 50 records. [Qdais et al. 2010] consider that the neural network was able to predict the methane output with a correlation coefficient of 0.87. To optimize the methane output, the authors developed a genetic algorithm that uses the neural network as a fitness function. The genetic algorithm discovered a set of variables that yield a methane production of 77% (relative to the total production of gas), while the recorded data shows a peak of methane production at 70,1%. Therefore, the results suggest that the power plant could improve the methane production by 6,9%.

[Oliveira-Esquerre et al. 2002] studied the application of neural networks and Principal Component Analysis (PCA) to the problem of simulating a wastewater treatment plant. The goal was to predict the Biochemical Oxygen Demand (BOD) of the output stream of a wastewater treatment plant. The research used a data set of 71 records and

8 parameters, but not all of these parameters were meaningful, since they did not contribute much to the variation of the output variable. PCA was used to improve prediction accuracy. The research found out that a simple feedforward neural network with a single hidden layer did not provide satisfactory results. Before using PCA, the neural network achieved a correlation index of 0.60, while the neural network with PCA achieved a correlation index of 0.77. The research indicates that neural networks can represent highly nonlinear relationships, and also shows that preprocessing the data with techniques such as PCA can lead to improvements on neural network performance.

Despite the fact that the studied works did not use large amounts, they have had good results using machine learning algorithms, including neural networks. Furthermore, some of these works had to deal with several problems, including lack of data and poor data set quality. Significant effort is done ensuring that the training data set has good quality in order to reduce mispredictions. The works also suggest that neural networks often perform well in the context of biogas prediction.

2.2. SaaS for Biogas Environments

We performed a survey focused on SaaS biogas software. The software presented here are focusing on monitoring the biogas production process. An example is Click! System from Gas Data ([Gas Data 2015]). Click! focuses on monitoring an array of sensors installed in the biogas plant as well as offering a cloud-based software from which the user can control the system. The solution also includes data transmission via TCP/IP and implements standardized output formats.

Carbon Cloud ([Green Lagoon 2015]) from Green Lagoon provides the same cloud-based software features as Click! System, allowing to control the equipment installed in the biogas plant and visualize data, monitoring and notifying via email and SMS. Carbon Cloud's website[1] provides more information about the system.

BOGIS ([BT IT 2015]) is a SaaS solution that focuses on the economic approach of biogas plants[2]. BOGIS evaluates the profit obtained from biogas operations as well as energy and biogas production in a given time period. BOGIS is also independent of biogas plant manufacturers, and receives data directly from biogas plant interfaces.

In respect to the current work, Click! System and Carbon Cloud are able to get data from sensors and present them to the final users. The technical team working on the biogas plant can get more insight about the digestion process, helping on the diagnostic of problems and optimizations of the production process. Therefore, a biogas system should display general information about the biogas processes.

However, discovering methods to improve the biogas production can be done with alternative methods, as shown by [Sota Solutions 2016]. Sota Solutions provides software focused on simulation and optimization of energy production processes. The company developed a software that predicts energy production from biogas based on historical data[3] with neural networks. The company found out the neural network algorithm yields predictions with 5% more quality than other comparable procedures such as linear regression.

[1] http://www.glt.my/downloads/glt-cloud-monitoring-and-reporting-system.pdf

[2] http://www.bt-it.de/download/flyer.bogis.en.pdf

[3] http://sota-solutions.de/wordpress_en/#anwendungen

The applications here surveyed shows that the Information Technology regarding the biogas context is well established, covering several needs including biogas prediction and data visualizations. Along with several scientific publications, Sota Solution's software suggest that the use of neural networks for biogas is plausible even in a commercial context. However, this work's SaaS application focuses on providing a high-level interface for machine learning algorithms, specifically neural networks in order to performing predictions in a generic way without requiring the user to have prior technical knowledge in the areas of programming, mathematics and machine learning.

3. Prophet: The SaaS application

The application developed is called "Prophet". In order to abstract low-level details of machine learning, the interface should provide only the necessary configurations, such as the input and output variables. Low-level details must be taken care by the application's code or predefined configurations. The biogas production process needs to be understood before developing the concepts that the application uses.

An anaerobic digester can process many types of substrates, including vegetable and animal waste as well as rests of food and other organic substances. However, different types of substrates can possibly produce different amounts of biogas composed of different proportions of methane gas. If the same digester is used to digest two or more types of substances, collected data should be separated in the system to provide accurate predictions. If all data is used to train a single neural network and the data was collected during the digestion of two or more types of substances, the predictions might not be accurate. In order to enable this level of segmentation, the concept of "models" was introduced.

The concept should be applicable not only for different types of substrates, but also for any condition that could dramatically change the gas output. A model is a temporary configuration of an AD, and the data collected represents the events that happened while that configuration was taking place. When the same digester is used to process two or more types of substrates at separate times, the user should create two or more models in the system, and each model should have the appropriate data to perform the model training. By creating models, any important change of configuration can be isolated from the others. It is a simple way to solve the problem by allowing the user to isolate different situations as needed. A model consists of a name that describes the situation, a set of input variables present on the data set which will be used to predict an output variable, and the output variable itself which will be predicted based on the input variables. Prophet enables the user to select the desired variables by their description, as shown in Figure 1.

After creating a model, the user can build the data set by uploading files containing the variables selected in the model. Currently, the application supports CSV file uploads. After validating the file, its contents are copied into the database. Furthermore, the current architecture allows for further improvements that should enable the application to read more file formats (such as JSON and XML) without changing the neural network code. After uploading a file, the model is ready to be trained when requested by the user.

The user can perform predictions after training the model. A prediction consists of a range variable (which is displayed as the horizontal axis in a chart), minimum and maximum values for the range variable, number of predictions to be performed and fixed values for the other variables in the model.

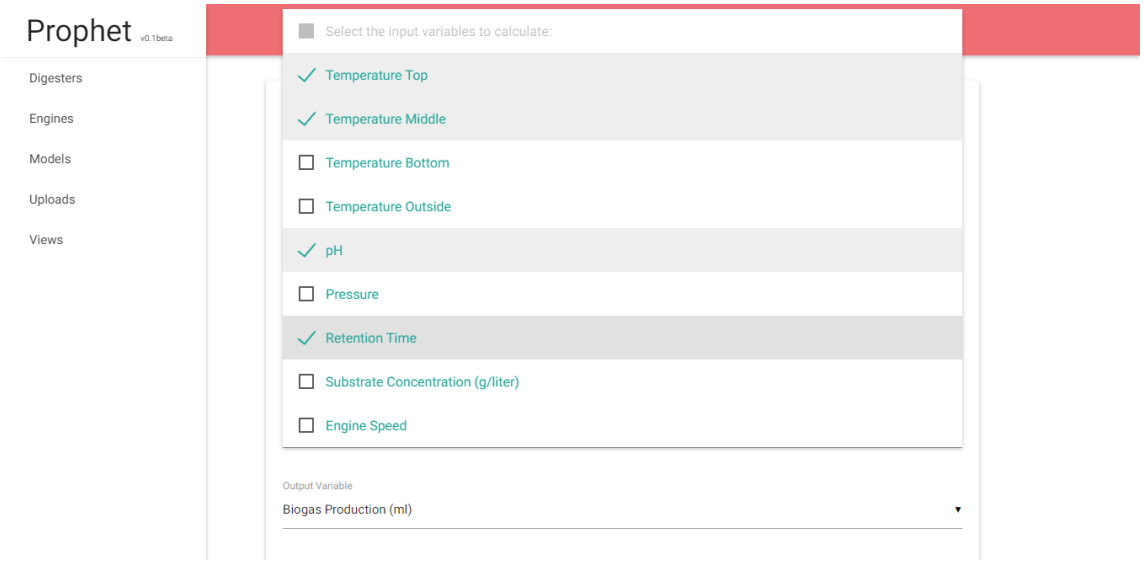

Figure 1. Selecting Variables

An example of a prediction can be stated as follows: The user wants to see the variation of biogas production in a given AD during a period of time. If the user has configured the model to use the AD temperature, substrate pH, pressure and retention time, all the variables must be set in order to perform a single prediction. It is not necessary to choose the output variable in the prediction configuration, as it is already defined in the model. Figure 2 shows a chart rendered from predicted information produced by the neural network.

Prophet Web also includes views, which allow the user to add several charts together in order to compare multiple results at once, as shown in Figure 3. When the user configures and renders a prediction, Prophet allows to add the chart to an existing view, or create a new one containing the chart. Views allows the user to compare multiple information, for example: biogas production and engine energy output. Figure 3 shows the concept of views.

4. Deployment and Architecture

Prophet is composed of 2 parts: Prophet Web and Prophet Service. Prophet Web is a Node.js[4] web application responsible for the frontend. Node.js enables the use of Javascript in the server side using the V8 Javascript engine, which is also used in the Chromium web browser (and Google Chrome). Prophet Web does not perform machine learning and file processing tasks. These tasks are performed by Prophet Service, written in C++. The service process the uploaded files, train the neural network and perform predictions, while the web frontend just displays the information provided by the service. To clarify how the user actually uses Prophet, Figure 4 provides an activity diagram that represents the steps performed by the user on Prophet Web.

Prophet Service's architecture is very simple. The code is composed of a base Task class and three tasks: "TaskProcessUpload", "TaskTrainModel" and "TaskPerformPredictions", as shown in Figure 5. The 3 classes are named after the job they perform.

[4]http://nodejs.org

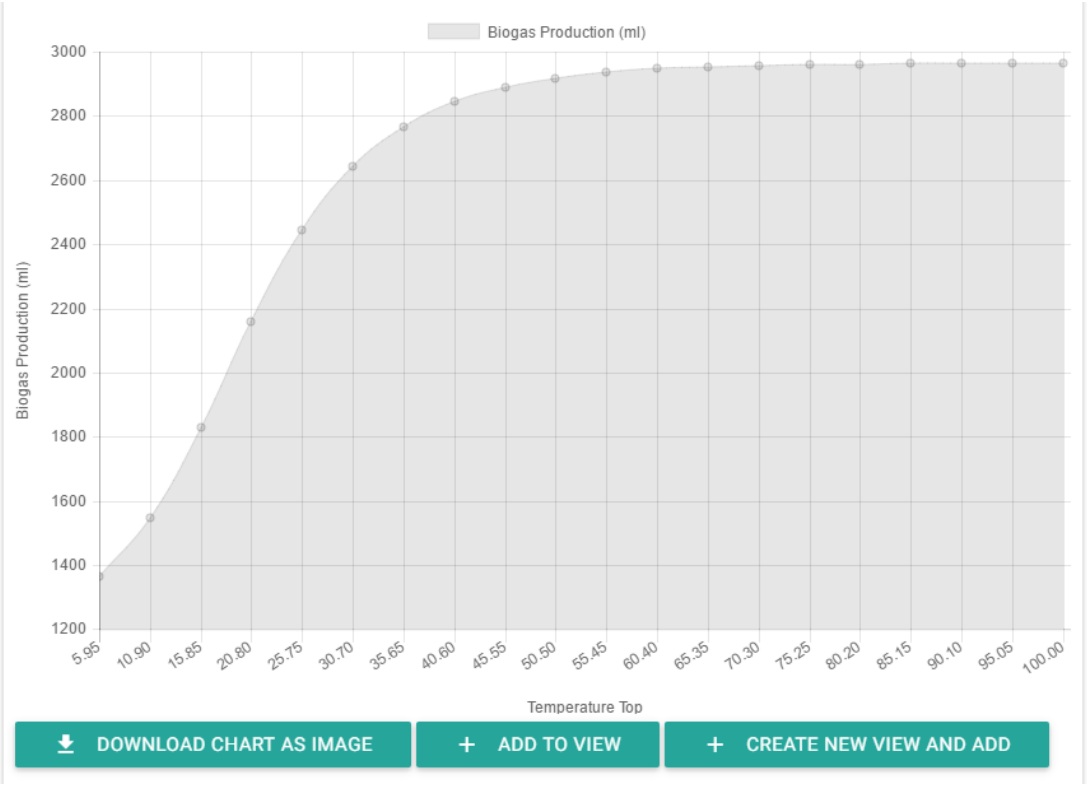

Figure 2. Prediction Chart

"TaskProcessUpload" validates the file and copies it to the database. "TaskTrainModel" implements a neural network training algorithm that takes care of preprocessing the data and performing the backpropagation algorithm. "TaskPerformPredictions" is responsible for loading the training model parameters and performing predictions as configured by the user.

The application was deployed to LARCC[5](Laboratory of Advanced Researches on Cloud Computing) at SETREM. LARCC implements efficient private cloud IaaS solutions [Vogel et al. 2016b, Maron et al. 2014, Adriano Vogel 2015], where we used Apache CloudStack[6] for allowing us to create instances with custom computing power on demand. We created three Ubuntu 14.04 instances as they are described in Table 1.

Table 1. Virtual Machines at LARCC using Apache CloudStack

Name	CPU Cores	RAM	Storage	Purpose
ProphetWeb	2	2GB	40GB	Prophet Web
ProphetService	4	8GB	20GB	Prophet Service
ProphetDB	4	12GB	100GB	Cassandra Database

This work uses the Cassandra Database. Cassandra is a column-oriented NoSQL database focused on scalability and write performance. The database was created at Facebook due to the need of a database system that is both resilient and scalable in order

[5]http://larcc.setrem.com.br
[6]https://cloudstack.apache.org

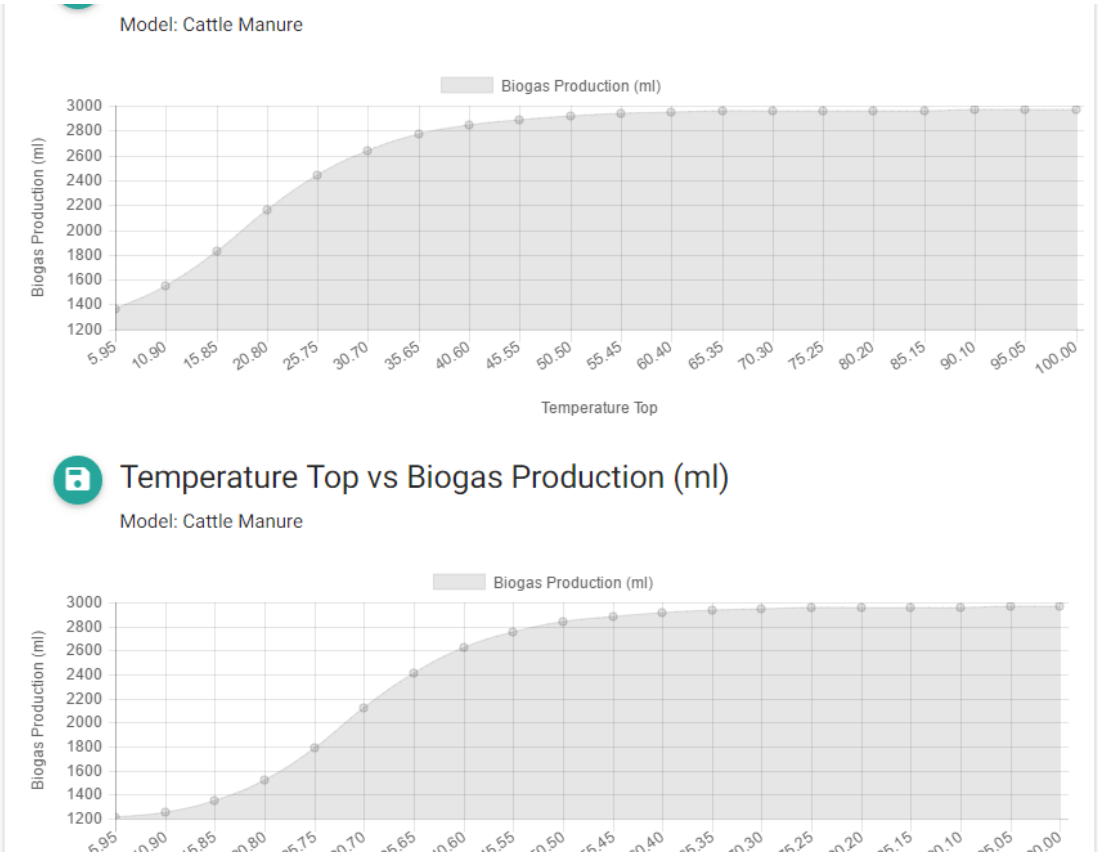

Figure 3. View Multiple Charts

to support the continuous growth of the platform [Lakshman and Malik 2009]. Cassandra is maintained by the Apache Foundation with the help of DataStax, which provides extensive documentation and opensource drivers for C++, Node.js and other languages.

Figure 6 shows the current deployment architecture: ProphetWeb is the web frontend server running Node.js, ProphetService is the background service server in which the C++ service (Prophet Service) runs. The Cassandra database is installed on ProphetDB.

Prophet Service was developed on Windows using Visual Studio, but the code uses only standard C++11 libraries available on Windows and Linux. The MSVC compiler allows some syntax that GCC 4.8 does not, therefore, some code had to be changed in order to support compilation with GCC 4.8, as well as some minor bugs needed to be addressed. Prophet Service also uses OpenBlas[7] to speedup calculation times. However, OpenBlas needed to be compiled again in the Apache CloudStack instance, since the library uses low-level and CPU architecture-dependent instructions, which are detected at compile-time. A specific test has shown that the library allowed the application to reduce training times from 4 minutes to 10 seconds without changing the application code. The FLENS[8] library was also used in order to perform matrix operations. The library provides a Octave/Matlab-like syntax to C++, making it easier faster to write code. FLENS also allows to use BLAS libraries by adding a simple compiler flag, and uses a default, non-

[7]http://www.openblas.net
[8]http://apfel.mathematik.uni-ulm.de/ lehn/FLENS/index.html

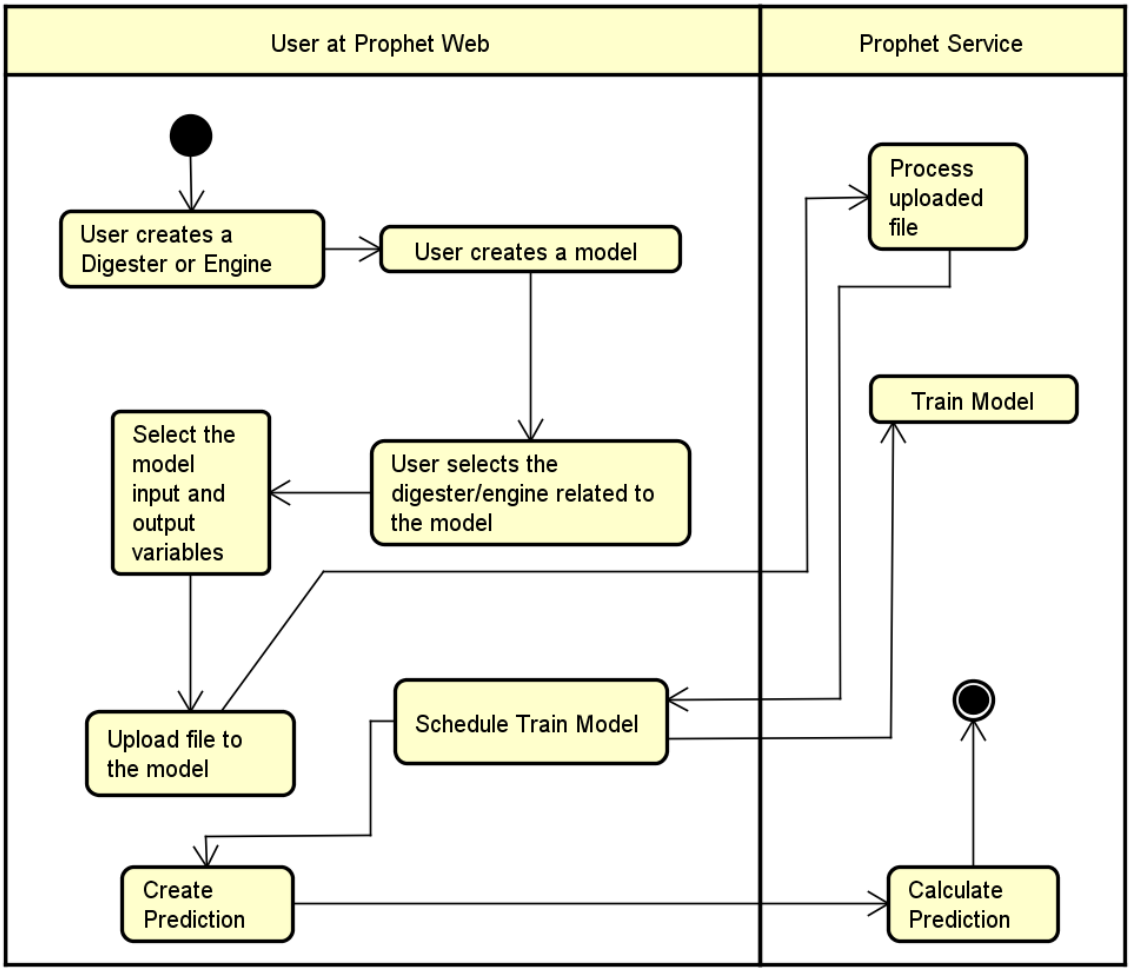

Figure 4. Activity diagram of Prophet Web

optimized implementation when no BLAS library is specified. Most of Prophet Service's performance during trainings and predictions is attributed to FLENS and OpenBlas.

5. Software Experiments

Prophet Service implements a neural network to process biogas data. The neural network uses a feed-forward algorithm to perform predictions. The number of hidden nodes can be configured in code, but the current implementation only supports one hidden layer. Each node in the hidden and output layers implements a sigmoid activation function. We also implemented a backpropagation algorithm to perform network training [Nielsen 2016].

We were not able to get real world biogas data, as there was no available infrastructure and hardware to collect data at SETREM at the moment being. Therefore, Prophet was tested using a data set of images of handwritten digits and a simple synthetic data set with 2 input variables and 1 output variable.

The handwritten digits data set was used to train the neural network. The goal is to correctly classify most of the 5000 digits present on the data set. The neural network was first developed using Octave[9] and then ported to C++. The images have 400 pixels

[9]https://www.gnu.org/software/octave/

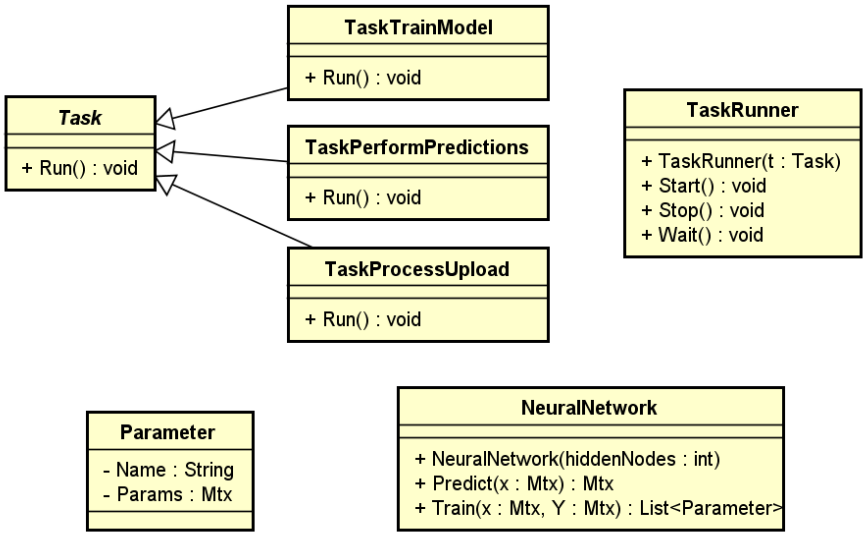

Figure 5. Class Diagram for the C++ background software

(20x20), resulting in 400 input variables. Each image represent a number between 0 and 9, therefore the neural network contains 10 output nodes. There is also 1 hidden layer with 25 nodes. The neural network was able to perform with good accuracy, recognizing correctly 4920 images out of 5000. Another simple test was developed in order to test whether the network can predict a simple linear relation of 2 input variables. The results indicate a correlation of 0.98, and the comparison between the data set and the predictions is shown in Figure 7.

The algorithm was also tested against WEKA using the same data sets previously described. WEKA implements a different algorithm for neural networks, requiring the user to configure some settings before using it. However, this test was performed using WEKA's default settings, changing only the number of hidden nodes to 2. In the simple 2-variable test, WEKA reports a correlation of 0.99, which is better than our implementation. This is attributed to the differences in the algorithm, where WEKA implements a linear activation function, while Prophet implements a sigmoid activation function. The test results for WEKA is shown in Figure 8.

In the handwritten digits data set, WEKA performed with a correlation of 0.833, while Prophet achieved 0.98. The number of hidden nodes was changed to 25. This is also attributed to the differences in algorithm. However, WEKA could perform better by performing some changes in the data set and configurations, as WEKA chooses between sigmoid and linear activation functions depending on the output variable, which is a discrete numeric value representing the handwritten digit (from 0 to 9), but WEKA might be interpreting it as a continuous value. Regardless of the correlation index achieved by WEKA, Prophet performs the training process in around 10 seconds, while WEKA takes around 75 seconds. This speedup is attributed to the usage of OpenBLAS, which is capable of using multiple cores to perform matrix operations as well as using low-level, CPU-architecture specific instructions.

After deploying Prophet Web and Prophet Service, the experience of using the

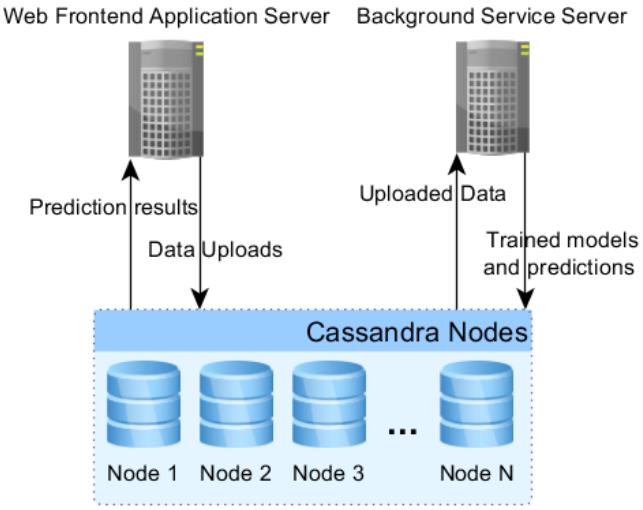

Figure 6. Application Architecture

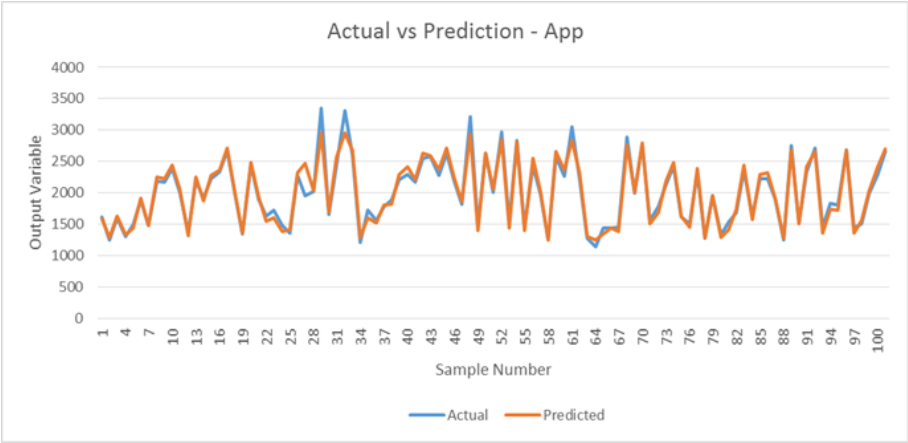

Figure 7. Simple Prediction Test

system was evaluated. Ideally, the user should get results in a timely manner, meaning that not only the system should have good performance and allow improvements with further implementations, but also the time spent by the user on setting the system up (creating models, uploading files and making predictions) should also not take too long.

Five workloads were tested, using data sets with different numbers of rows: 400, 4000, 40,000, 400,000 and 5 million. The files are CSV files with 3 columns. Table 2 shows how much time Prophet spent performing its tasks. The "Prediction time" column shows how much time it takes to perform 20 predictions. The times are measured in the following way:

- DB load: Prophet Web measures the time taken to create 5MB chunks of the original file and finishing upload all the chunks to Cassandra.
- File load: Prophet Service measures the time taken to process the file and load the data into the database.
- Training: Prophet Service measures the time taken to train a model, including the time taken to load the data from Cassandra to memory.

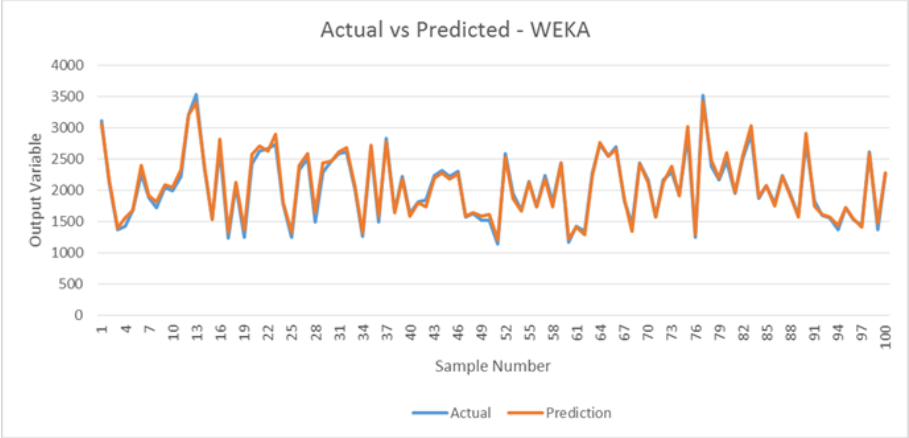

Figure 8. Simple Prediction Test for WEKA

- Prediction: Prophet Service measures the time taken to load the model from Cassandra, generate the prediction data set and executing the feed-forward algorithm.

Table 2. Prophet performance under 5 different workloads

#	Rows	Size	DB load	File load	Training	Prediction	Total
1	400	7.3KB	109 ms	154 ms	361 ms	89 ms	713 ms
2	4,000	78.1KB	10 ms	409 ms	881 ms	91 ms	1631 ms
3	40,000	781KB	32 ms	3163 ms	5291 ms	86 ms	9936 ms
4	400,000	7.62MB	390 ms	30 sec	50 sec	88 ms	81 sec
5	5,000,000	95.3MB	3851 ms	381 sec	489 sec	99 ms	874 sec

As can be seen on the Table 2, the training times increase according to the size of the data set. Test #4 takes around 50 seconds to train, while test #5 takes 489 seconds. Test #5 is 12.5 times larger than Test #4, and takes 9.78 times more than Test #4 to complete. Prediction tasks execute almost instantly in all cases, because the number of calculations increase according to the size of the problem (number of input variables) and not according to the size of the training set.

Column "DB load" shows how much time it takes to store the file in Cassandra. Prophet Web breaks the file into smaller 5MB files. To transfer the file used in test #5, Prophet took 3851 milliseconds (almost 4 seconds). For smaller workloads such as #3 and #4, Prophet should be able to transfer the file in less than a second. Before using the data, Prophet has to process the file first. To perform file processing, Test #4 takes 30 seconds, while test #5 takes 381 seconds. Test #5 takes 12.7x more time to process than test #4 while being 12.5x larger.

The conclusion is that the application is performing well, except for data sets as large as 5,000,000 rows and 3 columns. When used with data sets as large as 100,000 rows and several columns, the training process should not take a long time to run (around 10 seconds). However, the test shows that some optimizations could be applied. For instance: if a model data set grows in an unexpected way (i.e. uploading several gigabytes of data), the algorithm might not be able to train the neural network or perform any prediction due to memory constraints: test #4 used 350MB of RAM during the training process, while

test #5 used 2GB of RAM.

A possible solution is to use a subset of the data set (a smaller sample of the data set) in order to keep training times short and memory usage under reasonable levels that do not compromise the operating system and other tasks being performed concurrently. The memory usage problem exists because the algorithm uses the entire data set 100 times, using a large amount of RAM to store the entire data set in memory during the process.

Even though there are limitations in Prophet regarding sizes of data sets for training, some scientific works often use less than 1,000 records and achieve good results. [Holubar et al. 2002] developed a backpropagation neural network with 9 input nodes, 3 hidden nodes and 2 ouput nodes to predict biogas production, achieving a regression coeficient of 90% using 500 records. [Qdais et al. 2010] also created a backpropagation neural network with 4 input nodes, 3 hidden nodes and 3 output nodes using 177 records, achieving a correlation coefficient of 0.87. [Kusiak and Wei 2014] used the Adaptative Neuro-Fuzzy Inference System (ANFIS) toolbox of Matlab 10.0 to predict methane production a with correlation coefficient of 0.99 using 725 records. Therefore, Prophet should perform well when used with similar or heavier workloads.

6. Conclusions and Future Works

This work discussed the creation of a SaaS software for biogas prediction. The use of machine learning algorithms helps to extract useful information from data sets to improve the process or product. In our case, the software was implemented by using neural networks. The application focused on providing a high-level interface for the user, which is not required to have knowledge on machine learning to use the system.

In the Section 3, Prophet was described and its concepts were introduced. “Models” is the most important concept in Prophet, enabling the user to separate unique configurations of an anaerobic digester in order to provide better predictions. The application also abstracts the low-level details of the algorithm, requiring the user to only upload data and configuring predictions. Section 4 describes the deployment and architecture. The application was designed as a SaaS application. Prophet was deployed to the LARCC infrastructure, and can be accessed anywhere through the Internet.

Moreover, the application and the neural network algorithm were put under tests (Section 5). Our results show that the neural network is behaving well, being able to perform predictions with correlation coefficients greater than 0.90, even in more complex problems such as recognizing handwritten digits. The training times are fast when the data set size is comparable with publications in the area of biogas and machine learning, as described in the Section 5. The algorithm might also have a fast training time with larger data sets. Additionally, the prediction times are usually fast, and future works might improve it to perform on-line predictions in the application. Furthermore, all tests with handwritten digits suggests that the neural network is very flexible and could be applied to several other domains outside the biogas context.

Unfortunately, we only have performed predictions with synthetic AD data, because we do not have access to real world data at the moment being. As future work, we intend to get sensor data of AD deployed at SETREM and perform usability experiments with users that are not expert on machine learning. Also, we plan to implement

automatic data transmission from the biogas plant to the SaaS application, allowing to perform on-line training and provide real time recommendations to the user. Other data visualization methods could be studied and explored, as well as tools to export Prophet's data to other formats. Different databases could also be tested, such as ScyllaDB[10], a Cassandra-compatible database implemented in C++. Other algorithms such as linear regression and K-Nearest-Neighbors could also be implemented, while keeping the low-level details abstracted from the user.

Acknowledgements

We would like to thanks SETREM institutional support and LARCC team for providing the necessary infrastructure in our work. Also, we thank Prof. Fauzi Shubeita for the fruitful discussions during the development of Prophet.

References

[Adriano Vogel 2015] Adriano Vogel, Carlos A. F. Maron, V. L. L. B. F. S. C. S. D. G. (2015). HiPerfCloud: Um Projeto de Alto Desempenho em Nuvem. In *14th Jornada de Pesquisa SETREM*, page 4, Três de Maio, Brazil. SETREM.

[Alves et al. 2015] Alves, P., Viana, S., Carlos, L., and Aoki, A. (2015). Potential Assessment Tool of Biomass Electricity Generation for the State of Paraná. In *Sustainable Energy and Environment Protection - SEEP 2015*, pages 148–153, Paisley, Scotland.

[Aslanzadeh 2014] Aslanzadeh, S. (2014). *Pretreatment of cellulosic waste and high-rate biogas production*. PhD thesis, University of Borås, Borås, Sweden.

[Bioprocess Control 2015] Bioprocess Control (2015). Biogas Endeavour.

[BT IT 2015] BT IT (2015). BOGIS.

[Buyya et al. 2010] Buyya, R., Broberg, J., and Goscinski, A. M. (2010). *Cloud Computing: Principles and Paradigms*. Wiley, 1 edition.

[Coopersmith et al. 2014] Coopersmith, E. J., Minsker, B. S., Wenzel, C. E., and and, B. J. G. (2014). Machine learning assessments of soil drying for agricultural planning. *Computers and Electronics in Agriculture*, 104:93–104.

[Gas Data 2015] Gas Data (2015). Click! System.

[Green Lagoon 2015] Green Lagoon (2015). Carbon Cloud.

[Gutiérrez-Castro et al. 2015] Gutiérrez-Castro, L., Diez, P. Q., Tovar, L., and López, L. R. (2015). Development of the Large Scale Biogas Technology for Energy Generation in Mexico City. In *28th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact on Energy Systems*, pages 7–8, Pau, France.

[Holubar et al. 2002] Holubar, P., Zani, L., Hager, M., Froschl, W., Radak, Z., and Braun, R. (2002). Advanced controlling of anaerobic digestion by means of hierarchical neural networks. *Water Research*, 36:2582–2588.

[Kusiak and Wei 2014] Kusiak, A. and Wei, X. (2014). Prediction of methane production in wastewater treatment facility: A data-mining approach. *Annals of Operations Research*, 216(1):71–81.

[10]http://www.scylladb.com

[Labatut and Gooch 2012] Labatut, R. A. and Gooch, C. A. (2012). Monitoring of Anaerobic Digestion Process to Optimize Performance And Prevent System Failure.

[Lakshman and Malik 2009] Lakshman, A. and Malik, P. (2009). Cassandra - A Decentralized Structured Storage System. 104.

[Maron et al. 2014] Maron, C. A. F., Griebler, D., Vogel, A., and Schepke, C. (2014). Avaliação e Comparação do Desempenho das Ferramentas OpenStack e OpenNebula. In *12th Escola Regional de Redes de Computadores (ERRC)*, pages 1–5, Canoas. Sociedade Brasileira de Computação.

[McQueen et al. 1995] McQueen, R. J., Garner, S. R., Nevill-Manning, C. G., and Witten, I. H. (1995). Applying Machine Learning to Agricultural Data. *Computers and Electronics in Agriculture*, 12:275–293.

[Nielsen 2016] Nielsen, M. (2016). *Neural Networks and Deep Learning*. , 1 edition.

[Oliveira-Esquerre et al. 2002] Oliveira-Esquerre, K., Mori, M., and Bruns, R. (2002). Simulation of an industrial wastewater plant using artificial neural networks and principal component analysis. *Brazillian Journal of Chemical Engineering*, 19(04):365–370.

[Qdais et al. 2010] Qdais, H. A. A., Bani-Hani, K., and Shatnawi, N. (2010). Modeling and optimization of biogas production from a waste digester using artificial neural network and genetic algorithm. *Resources Conservation and Recycling*.

[Roveda et al. 2015a] Roveda, D., Vogel, A., and Griebler, D. (2015a). Understanding, Discussing and Analyzing the OpenNebula and OpenStack's IaaS Management Layers. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, 3(1):15.

[Roveda et al. 2015b] Roveda, D., Vogel, A., Maron, C. A. F., Griebler, D., and Schepke, C. (2015b). Analisando a Camada de Gerenciamento das Ferramentas CloudStack e OpenStack para Nuvens Privadas. In *13th Escola Regional de Redes de Computadores (ERRC)*, Passo Fundo, Brazil. Sociedade Brasileira de Computação.

[Sota Solutions 2016] Sota Solutions (2016). Sota Solutions.

[Thome et al. 2013] Thome, B., Hentges, E., and Griebler, D. (2013). Computação em Nuvem: Análise Comparativa de Ferramentas Open Source para IaaS. In *11th Escola Regional de Redes de Computadores (ERRC)*, page 4, Porto Alegre, RS, Brazil. Sociedade Brasileira de Computação.

[Vogel et al. 2016a] Vogel, A., Griebler, D., Maron, C. A. F., Schepke, C., and Fernandes, L. G. (2016a). Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack. In *24rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 672–679, Heraklion Crete, Greece. IEEE.

[Vogel et al. 2016b] Vogel, A., Maron, C. A. F., Griebler, D., and Schepke, C. (2016b). Medindo o Desempenho de Implantações de OpenStack, CloudStack e OpenNebula em Aplicações Científicas. In *16th Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS)*, pages 279–282, São Leopoldo, RS, Brazil. Sociedade Brasileira de Computação.