Enterprise Application Outage Prediction Using XGBoost and LSTM

Ali Hassan
IT Operations & Site Reliability Engineering
Royal Bank of Canada
Toronto, Canada
ali.hassan@rbc.com

Shahram Shah Heydari
Faculty of Business and Information Technology
University of Ontario Institute of Technology
Oshawa, Canada
Shahram.Heydari@ontariotechu.ca

***Abstract*—A large percentage of the global GDP is dependent on the Internet servicing millions of applications, therefore monitoring the health of these applications is critical to businesses around the world. To this date, majority of the mechanisms to support applications have all been reactive in nature; that is, reacting when an application goes down and the business has suffered financial loss. The use of Machine Learning in this space has been very limited until very recently. In this paper, we aim to employ a multi-modal model in determining the health of an application based on several tangible metrics from the infrastructure. Our results indicate that our method is able to predict the health of the application successfully.**

***Index Terms*—Application, Outage Prediction, XGBoost, Long-Short Term Memory (LSTM), ITSM, Regression, Classification**

I. Introduction

A. Application Outages

Over the past several years, Site Reliability Engineering has been used to employ a mixture of both proactive and reactive measures to prevent end-user impact when an issue arises with the application. To do so, IT Service Management (ITSM) data, which includes but is not limited to metrics from servers, operating systems, network, hardware, software and analytical metrics are gathered regularly to ensure logging is up to standard. This will ensure all the information required to troubleshoot is sufficient to determine the root cause and provide troubleshooting in an acceptable time for the business. It should be the goal of every SRE team to reduce reactive work in a proactive manner.

To accomplish this task, the Service Level Indicators (SLI) are defined to provide quantitative measures into how reliable the system is; these indicators are very important to monitor in order to understand the health of the system, using a variety of modules in for classifying and predicting application health status. Such modules can employ a variety of techniques, including artificial intelligence (AI).

AI is one of the fastest growing areas of research, and its applications have been tried and tested in areas such as stock markets, real estate, big data, and more. However, adoption in the area of application monitoring, classification and regression has been limited due to the unique and specific nuances of each application, making it a challenge to deploy artificial intelligence.

Among different AI techniques, the Long Short Term Memory (LSTM) model has been most successfully used for prediction of patterns [2] for a wide-range of use-cases such as stocks, anomaly detection, failure prediction etc. However, LSTM models are more computationally expensive and normally take longer time to train due to their complexity and vulnerability to overfitting. In order to improve the performance, the use of regularizing Gradient Boosting has been proposed [3] due to its speed, performance and limited data requirement. XGBoost is an open-source regularizing gradient boosting framework that has recently been used along with LSTM in a number of applications, with promising results.

In this paper we will discuss our contribution to the area of Application Prediction using XGBoost and LSTM by proposing a Machine Learning solution combining two algorithms together in order to predict the health of an enterprise application. We will consider the performance of our model as satisfactory if we are able to make a successful prediction of the health, given specific metrics.

This paper is broken down into the following sections: Section II which discusses work in this area that is closely related to application health regression, our model in Section III which will describe the framework of our algorithm, our results and analysis in Section IV, and the conclusion and further areas of research in Section V.

II. Related Work

In our previous work [4], LSTM was utilized for anomaly detection and prediction, confirming its validity to be used in this research as well as one of the inputs. In [5], it was specifically concluded that prediction bounds can be clearly used for detection of anomalous data, an approach similar to that of our research. This research also pointed out that LSTM models perform better than Error, Trend, and Seasonal (ETS) and Seasonal Auto-Regressive Integrated Moving Average (SARIMA) models. Another example of using a different algorithm can be seen in [6] where Hidden Markov Models are used to predict maintenance for asset management using multi-sensor time-series data for prediction. In this research, hardware failure/degradation was predicted accurately and we aim to have similar results in our research, using a combined LSTM/XGBoost model. However, our scope is very different.

978-3-903176-59-1

Bi-LSTM has also been used in [7] which is used to predict job failures in the cloud resulting in over 90% accuracy for task and job failures. This paper hopes to resolve an issue mentioned in [8] where LSTM and RNN were used for failure prediction using metrics such as CPU, memory, cache, etc. However, using just LSTM resulted in lower weights erroneously, but this is very dependent on the dataset. In [9], LSTM, XGBoost, and Generative Adversarial Network (GAN) were used together for time-series anomaly detection and it was concluded that the algorithms combined resulted in a more effective approach to extracting the deep features of time-series data resulting in a high accuracy of 97.25% for anomaly detection. The use of LSTM and XGBoost was discussed in [10] however the models were *not* combined. This paper did however demonstrate a Mean Absolute Percentage Error (MAPE) of 2.32% and 7.21% for LSTM and XGBoost respectively. Another use-case of multiple algorithms together is in [11] where Gated recurrent units (GRU) and XGBoost were used individually and together. Individually, GRU and XGBoost have a Root Mean Squared Error (RMSE) of 2.85% and 2.78% respectively. However, when the models are combined, the RMSE drops to 2.36%.

The use of XGBoost and LSTM used together have been explored in many papers such as [12] which discusses using both the aforementioned algorithms together for predicting the next data $a(t+1)$ and $d(t+1)$ at the next moment for the sequences of data labelled *A* and *D*, where the prediction of *A* is done using LSTM and *D* using XGBoost. This research confirmed the possibility of using both algorithms together, albeit in a different environment. They concluded that the amalgamation of both algorithms provided an excellent approximation and generalization capacity for predictive performance of time-series data, in this case specifically for compression and decompression of data. In [13], short-term load forecasting was accomplished using similar days (SD) selection, empirical mode decomposition (EMD), and LSTM - SD-EMD-LSTM, and XGBoost was utilized for feature importance evaluation. In this paper, SD-EMD-LSTM was compared to SD-Autoregressive Integrated Moving Average (SD-ARIMA), SD-Backpropagation Neural Network (SD-BPNN), and SD-Support Vector Regression (SD-SVR) and it was found that SD-EMD-LSTM model had the lowest Mean Absolute Percentage Error (MAPE) of 1.04%.

Although our approach is similar to these aforementioned papers, the application is vastly different. It has been determined that using multiple models together may increase the accuracy, especially as our approach looks at a plethora of different datasets which LSTM may not be well equipped to handle.

III. Approach

This section introduces a three-step approach for predicting IT incidents and outages for any given application. The outcome of our method is a short-term prediction of a health score for the application, which will then be classified into three states; healthy, unhealthy, and degraded. The thresholds for this classification are identified by researching historical outages for the given application. Our three-step approach for obtaining the health score is outlined below.

1) A neural network model (LSTM) is trained to predict perimeter link network utilization. This is a relatively simple LSTM model employing a univariate dataset for prediction.
2) An ensemble model (XGBoost) is trained in parallel to the model trained in step 1 by utilizing independent multi-modal datasets including Information Technology Service Management (ITSM) data (e.g., Enterprise Change Management), technology infrastructure metrics (e.g., server utilization and application alerts), and publicly available outage information from Down Detector.
3) Predictions of the models above are then transformed into a probability score through customized nonlinear mapping. Lastly, a health score will be derived by calculating the weighted average of those probability scores. Thus far, to the best of our knowledge, all available solutions deal with recovering from a failure after an outage. However, we provide a method to classify and predict outages before they possibly happen based on the patterns identified within our three-step approach. The ability to amalgamate two models together is known to outperform a single model, thereby providing a more accurate prediction.

The first step is utilizing LSTM to predict link utilization. In the enterprise network, traffic entering the organization specifically for the Application in question, does not come via the Internet as other traffic; that is highly insecure. Instead, a content-delivery network (CDN) in the middle relays the user traffic back to the enterprise after performing the required checks within their data-centre to ensure the authenticity and the benign nature of the traffic. After validation checks are done externally, traffic enters the organization at one of many possible links on the Perimeter Router. For our calculation, we are using the summation of the perimeter links for the SCORE.

One aspect of this project consists of inputting historical link utilization statistics for the following link(s) and running LSTM to output a prediction. Upon training and validation checks, we were able to determine a very accurate result with both loss and validation loss decreasing over each trained epoch and the prediction visualizing the models' accuracy can be seen in Figure 1 with the prediction in blue and the actual traffic for the day in orange. In this case, the validation of the model was done using a different dataset than the one used for training, and weekend data was excluded from the dataset to focus on the critical business hours.

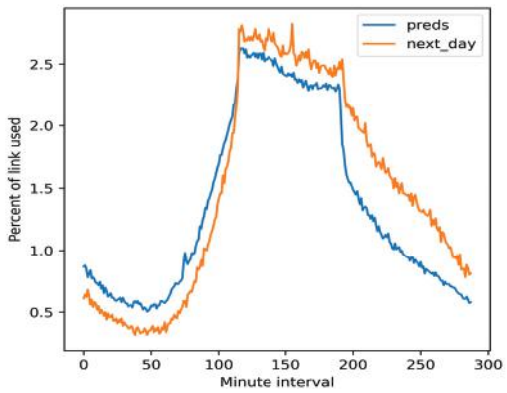

Fig. 1. LSTM Prediction

The second step in our three-step approach is utilizing the Ensemble models. To build this model, multiple features were used to produce a classifier value between 0 and 100, where a predicted score near 0 was classified as unhealthy/down, a score near 45 and below was classified as degraded service, and a score near 100 was classified as healthy and normal business operations. The following features were used to build this Ensemble model for regression:

- Server Logs – count of alerts sent to service desk and filtered for the servers involved in the application functionality whose status we would like to predict
- Change Risk – Change Record count per day of changes of low, medium, or high severity (labelled as CR_L, CR_M, and CR_H respectively) relevant to the application
- Outage Data – application down detector data which returns number of outages reported every 3 minutes for the application
- Status (Classifier Data) – provides definitive answer on whether application is Healthy, Unhealthy, or Degraded: based on the following threshold:
  - 100 – healthy
  - 50 – degraded
  - 0 – unhealthy

For the LSTM model, the predicted value was taken and was activated via non-linear mapping using the formula defined in Algorithm 1 to provide a mapping shown such that if the numerical value was closer to 0, that likely meant an unhealthy/down status, and if the value was closer to 100, then that signified a normal traffic load.

Once the two models had been built, as part of the three-step approach, the final step was producing a weighted average by taking data from both models to output a single value which would classify the performance of the application according to the low threshold and high threshold identified as T_{low} and T_{high} in the following manner:

1) Unhealthy if SCORE $\leq T_{\text{low}}$
2) Degraded if $T_{\text{low}} <$ SCORE $< T_{\text{high}}$
3) Healthy if $\geq T_{\text{high}}$

Algorithm 1 LSTM Final Value

($lower\_bound$ = min(lstm_preds_value))
($upper\_bound$ = max(lstm_preds_value))
(dr = $upper\_bound$ - $lower\_bound$)
(x_std = ($lstm\_preds\_value$-$lower\_bound$)/dr*100)
($final\_lstm\_value$ =(2500-(x_std-50)**2)/25)

Next, let's suppose the value from LSTM is represented as R_{LSTM} , the value from the Ensemble model is represented as R_{Ensemble}, and the weight is represented as K. The final weighted average formula is defined as follows:

Algorithm 2 Weighted Average Formula

R_{LSTM} = Output of LSTM prediction
R_{Ensemble} = Output of Ensemble prediction
$W = K * R_{\text{Ensemble}} + (1 - K) * R_{\text{LSTM}}$

To choose the best value for K, we ran several tests which involved finding the optimal score where the following criteria as in Table I is to be met:

TABLE I
CRITERIA FOR DIFFERENT CASES

Scenarios	SCORE Criteria
Case 1: When R_{Ensemble} = Healthy and R_{LSTM} = Healthy	value within a healthy range of $\geq T_{\text{high}}$
Case 2: When R_{Ensemble} = Unhealthy and R_{LSTM} = Unhealthy	value within an unhealthy range of between $T_{\text{low}} <$ final_result $< T_{\text{high}}$
Case 3: When R_{Ensemble} = Unhealthy and R_{LSTM} = Unhealthy	value within an unhealthy range of $\leq T_{\text{low}}$
Case 4: When R_{Ensemble} = Unhealthy and R_{LSTM} = Unhealthy	value within an unhealthy range of $\leq T_{\text{low}}$
Case 5: When R_{Ensemble} = Unhealthy and R_{LSTM} = Healthy	value where value is between the unhealthy range of $\leq T_{\text{low}}$
Case 6: When R_{Ensemble} = Healthy and R_{LSTM} = Unhealthy	value within a unhealthy range of $\leq T_{\text{low}}$
Case 7: When R_{Ensemble} = Healthy and R_{LSTM} = Unhealthy	value within n unhealthy range of between $T_{\text{low}} <$ final_result $< T_{\text{high}}$
Case 8: When R_{Ensemble} = Unhealthy and R_{LSTM} = Healthy	value within an unhealthy range of between $T_{\text{low}} <$ final_result $< T_{\text{high}}$
Case 9: When R_{Ensemble} = Unhealthy and R_{LSTM} = Unhealthy	value within an unhealthy range of $\leq T_{\text{low}}$

The results of those tests are shown in Table II. Based on actual field results regarding application status, in our simulations we used T_{low} = 56 and T_{high} = 74. It was decided to use the value K = 0.7 as it provided the most accurate result. It is also logically justified, as the LSTM portion is used to predict only one possible outage scenario where the CDN is impacted, therefore a lower weight needs to be given to LSTM. Furthermore, there are few CDNs available to choose from when it comes to hosting application and they are known to have a very high uptime as they employ a network of interconnected servers all over the globe servicing clients. Comparing this to the Ensemble model, the dataset for this

can be used to represent a wide range of issues, such as server error logs, configuration changes, and definitive data from down detector showing an outage, thus it is given more weight.

TABLE II
SUCCESSFUL - MULTIPLIER CHOSEN FOR R_{ENSEMBLE} AND R_{LSTM}

R_{Ensemble}	R_{LSTM}	R_{Ensemble} Multiplier (K)	R_{LSTM} Multiplier	$SCORE$	Prediction Correctness
80	80	0	1	80	Yes
40	80	0	1	80	No
80	0	0	1	0	Yes
80	40	0	1	40	No
80	80	0.3	0.7	80	Yes
40	80	0.3	0.7	68	Yes
80	0	0.3	0.7	24	Yes
80	40	0.3	0.7	52	No
80	80	0.5	0.5	80	Yes
40	80	0.5	0.5	60	Yes
80	0	0.5	0.5	40	Yes
80	40	0.5	0.5	60	Yes
80	80	0.7	0.3	80	Yes
40	80	0.7	0.3	52	Yes
80	0	0.7	0.3	56	Yes
80	50	0.7	0.3	68	Yes
80	80	1	0	80	Yes
40	80	1	0	40	No
80	0	1	0	80	No
80	40	1	0	80	No

IV. RESULTS/EVALUATION

As mentioned in earlier sections, the number of server alerts, outage data, labelling data and change and its severity were collected from various data sources including SQL, Elasticsearch, and Simple Network Management Protocol (SNMP) tools to be used as input in our models. This was fed into the XGBoost for training. Table III demonstrates the unhealthy dataset which was used. It is also worth noting that $> 40\%$ of outages to an application have been attributed to changes, making it by far one of the most important values to monitor to determine the health status.

TABLE III
UNHEALTHY DATA

Time	Outage	Alerts	CR_L	CR_M	CR_H	Label
00:05	2	1	13	0	0	0
00:09	1	1	13	0	0	0
00:13	1	1	13	0	0	0
00:17	0	1	13	0	0	0
00:20	0	1	13	0	0	0

Once the model was built, and trained, the accuracy of the model was validated with new data, which showed the models' ability to be very accurate in classifying both degraded and unhealthy traffic patterns.In the figure below, we can see that there was a drop in performance of the application from the Classifier data using *actual* data points.

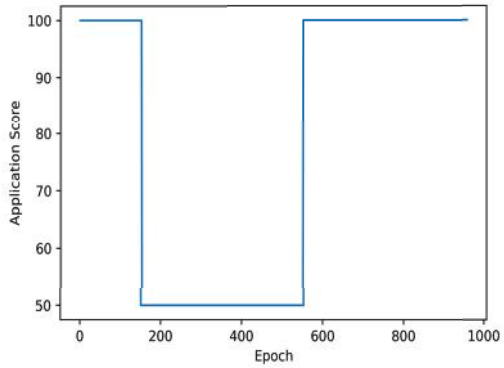

Fig. 2. Degraded - Real Values

Figure 2 shows the *real* health status where we saw the status decrease to 50, which is similar to our prediction in Figure 3 as well. Note that, due to the nature of enterprise applications, if the prediction is around 50 or 0 for either degraded or unhealthy application status, we will mark the entire day as being degraded or unhealthy respectively as we are mimicing the users perspective. If an application is unhealthy for a couple hours, enterprise users will typically note this as the application being unhealthy for the entire day.

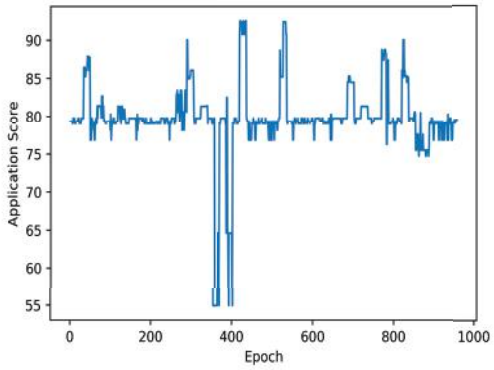

Fig. 3. Degraded - Prediction Values

As our model has successfully been able to distinguish when the application is degraded, we will further confirm that the model can also predict when an application is unhealthy, which is visualized in Figure 4 and 5.

Lastly, to provide more data for further research, a feature importance [14] analysis was done demonstrating the most important features in determining if an application is unhealthy and the result demonstrated that over CR_L and server_alerts together make up over *50%* of the predicting value for determining whether an issue is present in the application. Surprisingly, CR_H did not play as much of a role in terms of feature importance, as the assumption is that high risk changes are scrutinized heavily as part of the Information Technology Infrastructure Library (ITIL) process, requiring approvals from

the change executers in immediate management, VP/SVP approvals and a sign off from the business users. These steps are not required for CR_L changes. This opens the door to a lot of further research as there is a possibility that specific *types* of low, medium, and high risk changes could attribute to an application health than other types of changes and this may be worth investigating.

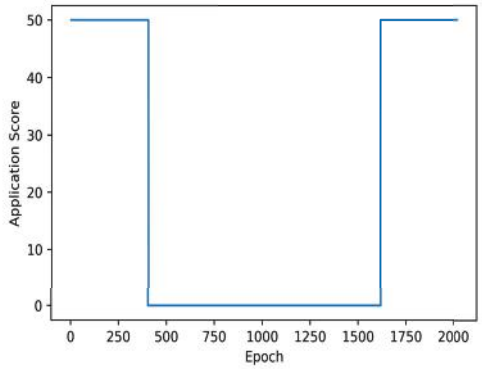

Fig. 4. Unhealthy - Real Values

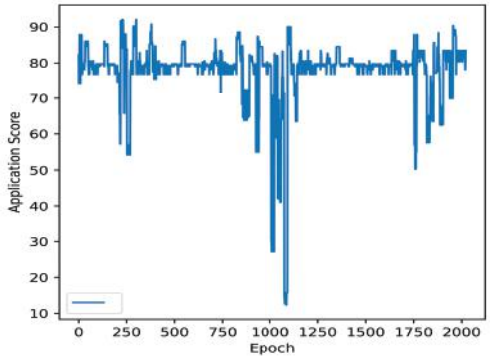

Fig. 5. Unhealthy - Prediction Values

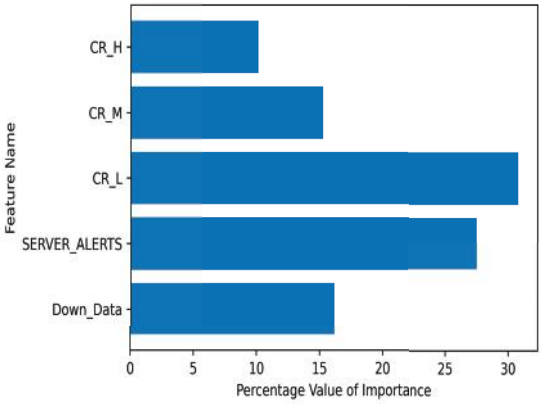

Fig. 6. Feature Importance

V. Conclusion and Future Work

In this paper, we presented a combination of an XGBoost and LSTM approach in determining the health of an application based on various metrics such as server log count, change count, and outage detector count. Through this method, we were able to successfully predict the health of our application, while also documenting the importance of each feature in determining the status.

For future work, our algorithm will be implemented in a production environment for an enterprise, with the end goal of proactive failure management and to eventually one day provide a solution to the various incidents our solution is capable of recognizing, i.e. - a self-healing framework. Furthermore, this algorithm could possibly be enhanced by using other metrics from Load Balancer, Firewall or application-specific metrics and can be scaled to practically any application given the appropriate data is collected. Additionally, as each application has a wide range of dependent applications, it would be of benefit to explore how this application is dependent on other applications for monitoring as well.

References

[1] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[2] G. Benram. extreme gradient boosting. DOIT. [Online]. Available: https://www.doit.com/xgboost-or-tensorflow/

[3] A. Hassan, "M.sc. thesis, TCP congestion control using reinforcement learning." Univeristy of Ontario Institute of Technology, 2022, Available at: https://hdl.handle.net/10155/155710.

[4] A. Jerome, T. Ishii, and H. Chen, "Forecasting and anomaly detection on application metrics using lstm," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2221–2227.

[5] P. Hofmann and Z. Tashman, "Hidden markov models and their application for predicting failure events," in *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part III 20*. Springer, 2020, pp. 464–477.

[6] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE transactions on services computing*, vol. 15, no. 3, pp. 1411–1422, 2020.

[7] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[8] X. Xu, H. Zhao, H. Liu, and H. Sun, "Lstm-gan-xgboost based anomaly detection algorithm for time series data," in *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*. IEEE, 2020, pp. 334–339.

[9] J. Luo, Z. Zhang, Y. Fu, and F. Rao, "Time series prediction of covid-19 transmission in america using lstm and xgboost algorithms," *Results in Physics*, vol. 27, p. 104462, 2021.

[10] X. Li, Y. Huang, and Y. Shi, "Ultra-short term power load prediction based on gated cycle neural network and xgboost models," in *Journal of Physics: Conference Series*, vol. 2026, no. 1. IOP Publishing, 2021, p. 012022.

[11] Z. Yan, J. Wang, L. Sheng, and Z. Yang, "An effective compression algorithm for real-time transmission data using predictive coding with mixed models of lstm and xgboost," *Neurocomputing*, vol. 462, pp. 247–259, 2021.

[12] H. Zheng, J. Yuan, and L. Chen, "Short-term load forecasting using emd-lstm neural networks with a xgboost algorithm for feature importance evaluation," *Energies*, vol. 10, no. 8, p. 1168, 2017.

[13] J. Brownlee. Feature importance and feature selection with xgboost in python. Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/