Predicting Workload Instability and Resource Utilization in Cloud Virtual Machines

Subhan Chaudry
schaudry@utexas.edu
University of Texas at Austin
Austin, Texas, USA

Abstract

As organizations move away from traditional on-premises workloads towards cloud deployments, they rely on reactive workflows in order to scale resources attached to virtual machines (VMs). This typically requires thresholds to be crossed before a VM is able to be migrated or scaled up to accommodate more compute or memory resources. In this project, we investigate whether machine learning models trained on VM telemetry data can predict excess CPU utilization events before they occur. Using the GWA-T-12 Bitbrains fastStorage trace dataset, we engineered features alongside raw data for training predictive machine learning models. The defined prediction task was to determine, for a given VM at a point-in-time, whether CPU utilization would exceed 85% within the next 15 minutes.

We trained Logistic Regression, Random Forest, and XGBoost models using chronological train-test splits. The results showed that tree-based models outperformed our logistic regression model. Random Forest, in particular, was able to achieve an accuracy of 99.66% on this prediction. Feature importance and SHAP analysis demonstrated that the maximum CPU usage percentage over the 15-minute and 30-minute intervals were the key indicators for our models. This confirmed that workload momentum is the key to predicting the need for additional VM resources.

1 Introduction and Research Background

Cloud computing has seen immense adoption within the last decade with cloud providers like Amazon Web Services, Azure, Google Cloud, and Oracle Cloud hosting millions of VMs for critical enterprise workloads [5]. This has led to challenges in keeping performance stable for cloud resources which may experience unpredictable demand depending on time and customer requirements [16].

The key issue has to do with how cloud resources are allocated to different workloads. When a VM experiences utilization beyond the capacity with which it was originally provisioned, this results in slow response times or even outages. These can violate Service Level Agreements (SLAs) with customers and result in loss of revenue or reputation

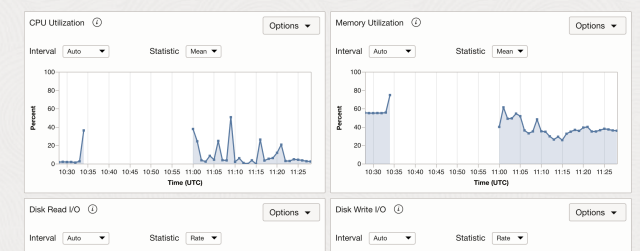

Figure 1. Oracle Cloud VM Utilization Monitoring

[17]. Attempting to over-provision cloud resources avoids these challenges but uses higher amounts of power and increases cloud costs [10]. The typical approach has been to leverage orchestration platforms that migrate or resize VMs once a specific threshold has been met for a set period of time [23]. As an example, a migration could be initiated once CPU utilization exceeds 90% for 5 minutes. This is a simple model to apply to most cloud infrastructure environments but can result in delays between when a CPU spike occurs and when the system responds and reallocates resources [32].

This project aims to predict these spikes in CPU utilization ahead of time. We specifically look at whether machine learning models can be trained to predict whether CPU utilization will cross 85% within the next 15 minutes. This is valuable as organizations can act proactively with migrations or resizing and avoid having any performance issues before CPU utilization exceeds capacity.

1.1 Related Work

Initial solutions for predicting cloud workload utilization relied on ARIMA and exponential smoothing [6], which perform well for stable workloads but struggle with typical enterprise spikes in utilization [22]. Prevost et al. [28] looked at both stochastic and neural models for predicting data center load and came to the same conclusion. Islam et al. [20] looked at linear regression, neural networks, and SVR for predicting CPU utilization and saw that neural networks performed slightly better but were much slower in practice to train. Duggan et al. [14] performed research that showed ensemble tree methods were able to perform stronger than single models.

Lorido-Botran et al. [23] studied cloud infrastructure deployment patterns and saw that reactive solutions were the

most common by far despite performance issues. Qu et al. [29] applied ARIMA with Random Forest on the Bitbrains dataset for proactive scaling predictions. Toka et al. [32] similarly applied reinforcement learning but had issues with their findings for real deployments.

The Bitbrains dataset [30] is a publicly available benchmark that contains 1,250 VM traces taken from various European financial institutions. It has been frequently used for workload classification [13, 26, 30], VM consolidation [15], and scheduling for better energy usage [3]. Mason et al. [25] demonstrated that using a rolling variance for CPU data was a better indicator of high utilization trends than current values. Shwartz-Ziv et al. [31] showed that ensemble tree models typically outperformed deep learning models when applied to tabular data.

Despite extensive studies on cloud utilization, the majority of these studies apply regression rather than classification approaches. Very few studies split train and test data by chronology which risks temporal leakage into models from research by Arlot et al. [2]. It was also very difficult to find studies that look at whether a feature like disk I/O predicts another such as CPU spikes. This project addresses these gaps while predicting high CPU utilization in VMs.

2 Materials and Data Sources

2.1 Dataset

For this project, we utilized the GWA-T-12 Bitbrains fastStorage trace dataset [30] which contains telemetry data from 1,250 VMs connected to fast storage area network (SAN) storage devices from a distributed datacenter. It includes data collected at 5-minute intervals over a one-month period, primarily for financial workloads. Each row includes the following information [12]:

- Timestamp: number of milliseconds since 1970-01-01
- CPU cores: number of virtual CPU cores provisioned
- CPU capacity provisioned (CPU requested): the capacity of the CPUs in terms of MHz
- CPU usage: in terms of MHz
- CPU usage: in terms of percentage
- Memory provisioned (memory requested): the capacity of the memory of the VM in terms of KB
- Memory usage: the memory that is actively used in terms of KB
- Disk read throughput: in terms of KB/s
- Disk write throughput: in terms of KB/s
- Network received throughput: in terms of KB/s
- Network transmitted throughput: in terms of KB/s

After joining the individual CSVs for each VM, this dataset contained around 10.8 million individual pieces of data.

2.2 Data Cleaning and Preparation

The dataset provided was well maintained so limited preprocessing was required to prepare it for machine learning

	Timestamp [ms]	CPU cores	CPU capacity provisioned [MHZ]	CPU usage [MHZ]	CPU usage [%]	Memory capacity provisioned [KB]	Memory usage [KB]	Disk read throughput [KB/s]	Disk write throughput [KB/s]	Network received throughput [KB/s]	Network transmitted throughput [KB/s]
0	1376314848	4	11703.99824	10912.027692	93.233333	67108864.0	6.129274e+06	0.133333	15981.800000	0.000000	2.133333
1	1376315148	4	11703.99824	10890.570362	93.050000	67108864.0	6.755624e+06	1.333333	19137.333333	0.000000	2.600000
2	1376315448	4	11703.99824	10434.114431	89.150000	67108864.0	8.947846e+06	2.533333	19974.933333	535.666667	23.933333
3	1376315748	4	11703.99824	10539.450415	90.050000	67108864.0	1.879048e+07	5.466667	8791.800000	349.666667	5.466667
4	1376316048	4	11703.99824	10951.041020	93.566667	67108864.0	9.305761e+06	5.400000	15679.533333	0.000000	2.066667

Figure 2. Bitbrains fastStorage Dataset

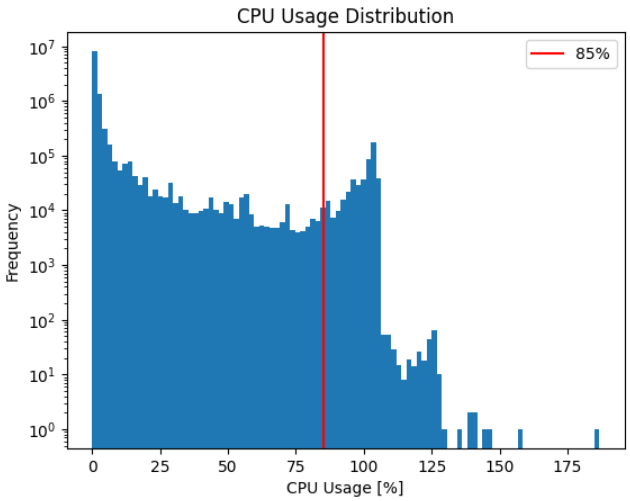

Figure 3. CPU Usage Distribution

training. We renamed the columns for better readability in our results sections. We also created a memory usage percentage column to go along with our CPU usage percentage. Any rows with nulls or negative values were dropped, duplicates were removed, and all timestamps were converted to a datetime format.

2.3 Data Exploration and Visualizations

Looking specifically at CPU usage distributions in Figure 3, we can see that most usage readings are below the 10% with a very small amount concentrated at the tail past our 85% threshold. This means that spikes are not very common in our dataset and are a rare event but need to be weighted properly for accurate predictions. In Figure 4, we show a time series for a few example VMs and can see that the behavior can vary greatly. Some VMs have frequent bursting periods while others are mostly stable with the occasional spike in usage. This means that a rolling window approach would be best to avoid missing these trends.

We also visualized the data with a correlation heatmap in Figure 5 which demonstrated that CPU usage in MHz and percentage were highly correlated. CPU usage and memory usage were also similarly correlated with one another. Interestingly, disk and network metrics did not show strong correlation with CPU usage and appear to be independent. For the binary classification, we visualized our class distributions in Figure 6 and can confirm again that spikes are rare with a strong imbalance.

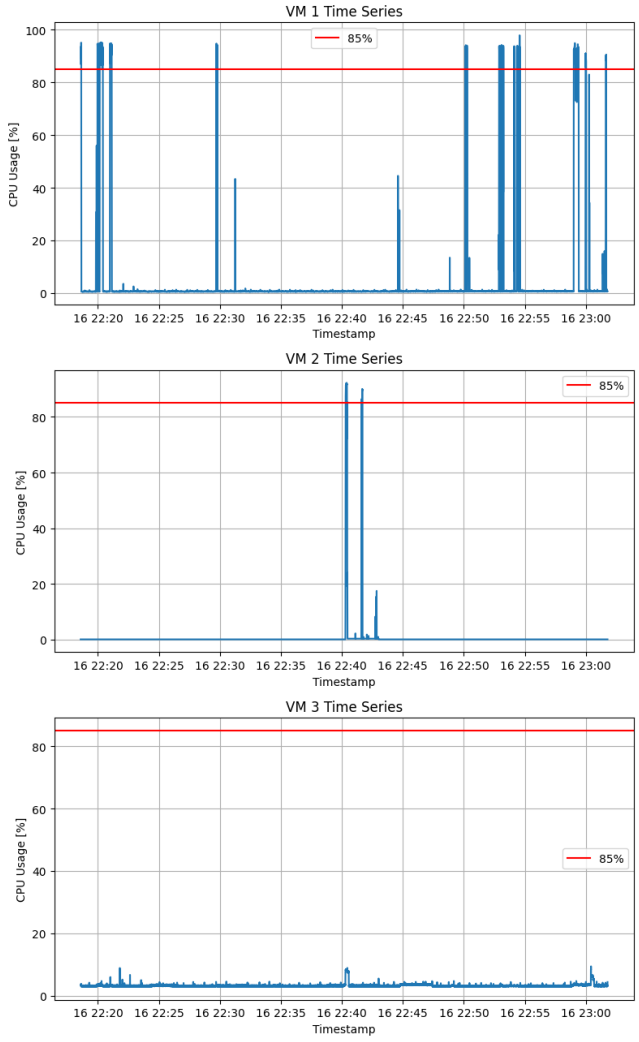

Figure 4. Examples of VMs Time Series

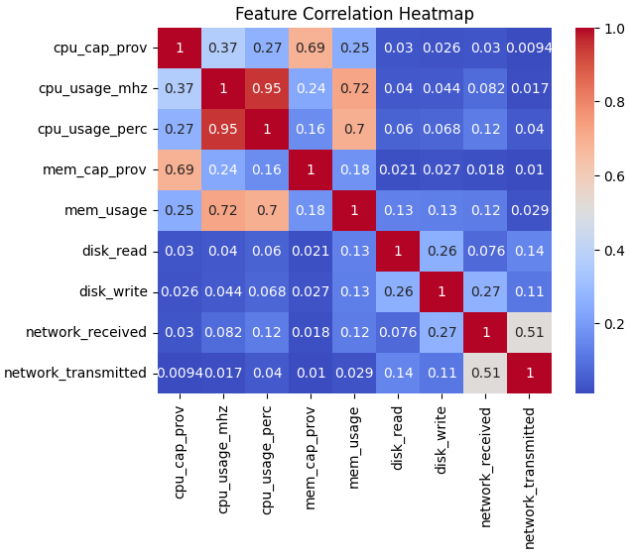

Figure 5. Feature Correlation Heatmap

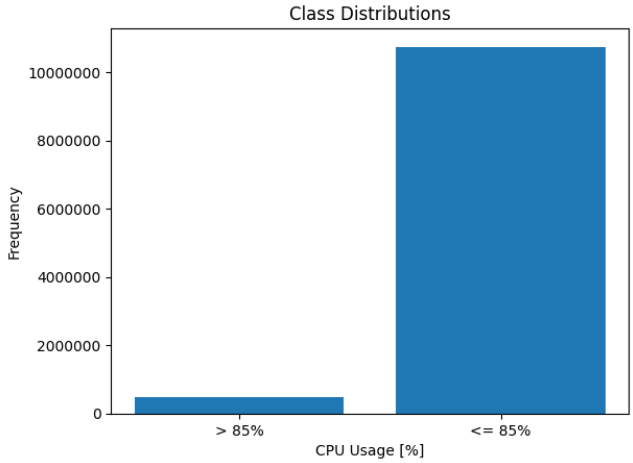

Figure 6. Class Distributions

3 Research and Methods

3.1 Feature Engineering

By looking at a single entry in the fastStorage dataset, we would not have been able to determine much information to predict a spike in CPU usage. For this reason, we created a rolling window to look at a 15-minute period and a 30-minute period of metrics to guide our predictions. The 15-minute period was designed to capture faster spikes and changes in usage, while the 30-minute period provided broader information on trends. This could be adjusted further to potentially capture slower movement as well.

We looked at 6 metrics (CPU usage, memory usage, disk read, disk write, network received, and network transmitted) and computed the rolling mean, median, standard deviation, and max over both of these time periods. We also calculated the difference in CPU usage for these time periods to capture how much of a jump there may have been in usage. This provided additional context to the data point for our machine learning models to leverage.

3.2 Target Variable

Since we are aiming to predict future spikes above 85% in the next 15 minutes, we had to determine the target variable by looking ahead at the next few readings in that time frame. To accomplish this, we calculated the max CPU usage over the next three readings (15 minutes) and set the target variable to 1 if it exceeded 85% and 0 otherwise. Since we need to look ahead, we had to drop the last three rows for each VM since there would be no future CPU usage data for a given data point. We can observe the distribution of this in Table 1 which we address through class weighting in our model training [7, 19].

Table 1. Target Value Counts

Target	Count
0	9129503
1	529217

3.3 Train-Test Split by Chronology

As discussed in the Related Work section, we need to avoid temporal leakage and make sure that our models only use past data for predictive learning [2]. For this reason, we split our train and test datasets chronologically. The first 80% of a VM's telemetry data enters our training set and the remaining 20% enters our test set. We scaled these train and test datasets with a StandardScaler that was fitted on our training data. One observed issue during runs was the introduction of infinity values due to feature engineering so these were replaced by 0 to preserve training stability.

3.4 Models

For predicting whether a VM would exceed 85% CPU usage in the next 15 minutes, we trained the following machine learning models:

- Logistic Regression (LR) as a linear baseline with balanced class weights and 1000 iterations [27].
- Random Forest (RF) using balanced class weights, 100 trees, a max depth of 15, and 20 samples per leaf [4].
- XGBoost (XGB) with scaled positive weights, 100 estimators, a max depth of 8, a learning rate of 0.1, and evaluation metric as AUC-PR [8].

These models were selected to compare the linear baseline with ensemble tree methods to see which would perform stronger based on similar work discussed in the Related Work section.

3.5 Evaluation Metrics

Due to a strong imbalance in our two target classes, we compared a number of different metrics including Accuracy, Precision, Recall, F1, ROC-AUC, and PR-AUC to evaluate performance of these machine learning models. PR-AUC is particularly useful for assessing performance when dealing with data that has strong imbalances in classes [11].

When we look at false positives, they would likely represent unnecessary migrations or resizing which would incur additional cost. False negatives can be understood as failing to identify spikes in CPU usage which could cause performance issues or violate SLAs.

4 Results

After training these models, we observe strong performance in being able to identify CPU usage exceeding 85% within the next 15 minutes with our training and evaluation approach.

Table 2. ML Models Metrics Comparison

Model	Acc.	Prec.	Recall	F1	ROC	PR
LR	0.9891	0.8146	0.9883	0.8931	0.9982	0.9336
RF	0.9966	0.9355	0.9950	0.9643	0.9999	0.9987
XGB	0.9961	0.9260	0.9952	0.9594	0.9999	0.9977

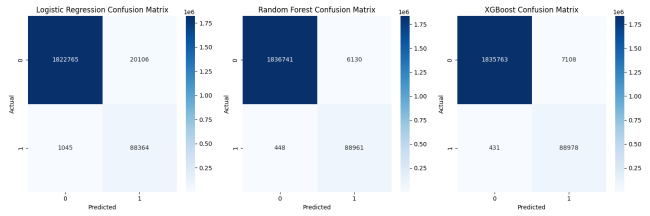

Figure 7. Confusion Matrices

4.1 Model Performance

Table 2 displays the key metrics that we observed for our machine learning models on our chronological test set.

We see that Random Forest has the highest overall performance with an F1 score of 0.9643 and PR-AUC of 0.9987. XGBoost also performed strongly at 0.9594 and 0.9977. Logistic Regression seems to be less effective when we look at its lower precision at 0.8146 causing too many false positives. All of our models performed very well when it came to recall, which means that they all were good at identifying spikes in CPU usage.

4.2 Confusion Matrices

The confusion matrices for our models can be seen in Figure 7. All of our models predicted similar amounts of true positives when we look at this table. Logistic Regression produces 20,106 false positives and 1,045 false negatives. Random Forest cuts our false positives down by 70% to 6,130 and false negatives down to 448. XGBoost reduces false positives to 7,108 but produces 431 false negatives which is the fewest missed spikes in usage.

4.3 ROC and Precision-Recall Curves

The ROC curves in Figure 8 for all of our models are tightly bound to the top left corner which shows they are doing well. Logistic Regression has a less steep trajectory with an AUC of 0.9982. Random Forest and XGBoost were nearly identical with the same AUC of 0.9999.

We can also see the Precision-Recall Curves in Figure 9. Here we can see some differences in the model behavior as we examine the tradeoff between precision and recall. Logistic Regression shows a dip in precision. Random Forest and XGBoost show almost perfect precision across recall. When we assess workloads of thousands to millions of VMs, a small difference in precision could result in significant costs to organizations.

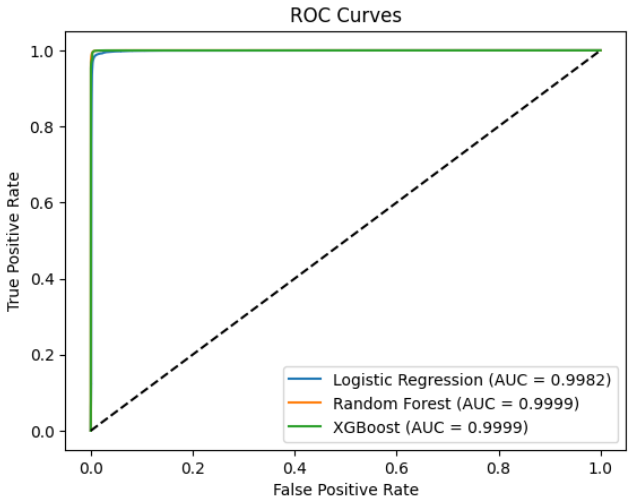

Figure 8. ROC Curves

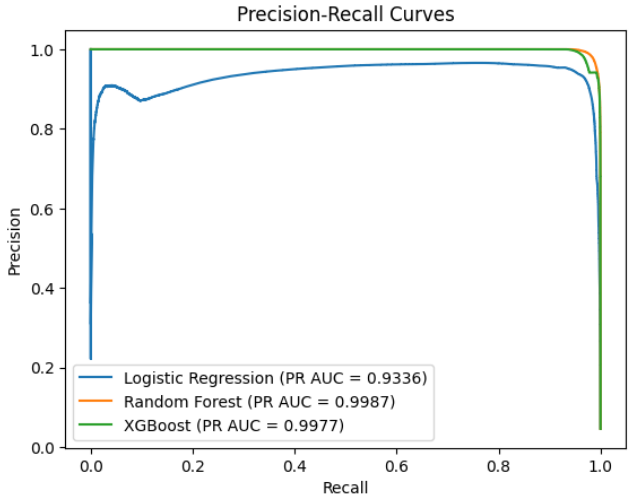

Figure 9. Precision-Recall Curves

4.4 Feature Importance

For Random Forest and XGBoost, our stronger models, we can take a deeper look at the feature importance rankings in Figures 10 and 11 to determine which contributed the most to our machine learning.

Random Forest applied importance across a number of different features, particularly the different engineering features for CPU usage. The rolling max CPU usage for 15 minutes is the most important of these which means it favors shorter periods. The raw CPU percentage at the time is valued the second highest.

Interestingly, XGBoost places almost all importance on the rolling max CPU usage for 30 minutes. This means that this feature alone is so strongly associated with correct predictions that it concentrates all the signal there. This makes

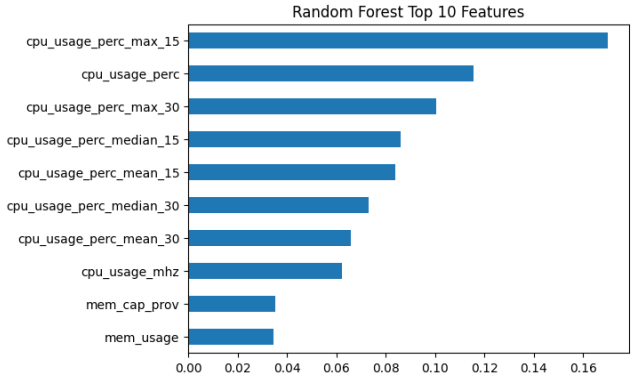

Figure 10. Random Forest Features

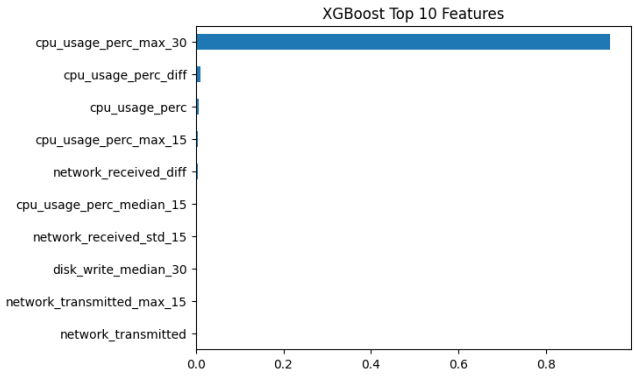

Figure 11. XGBoost Features

sense since XGBoost tends to prioritize features with the strongest predictive signal.

4.5 SHAP Analysis

We also used a SHAP analysis to determine the directionality that may have been missed by feature importance alone [24]. This plot can be seen in Figure 12 with the summary for each of the features. We can see that the rolling max CPU usage for 30 minutes has the widest spread and pushes the predictions positive or negative. The CPU usage percentages at that time have a similar pattern. We can also observe that the memory capacity provisioned has an effect on pushing the predictions positive, which may be due to the fact that larger memory shapes tend to have larger more spiky workloads attached to them.

5 Conclusion

These results were very positive for our project and showed that we can predict CPU usage spikes with strong accuracy. Notably, Random Forest showed an F1 score of 0.9643 and PR-AUC of 0.9987 which catches 99.5% of our spikes in usage with a precision of 93.6%. Since we applied a chronological

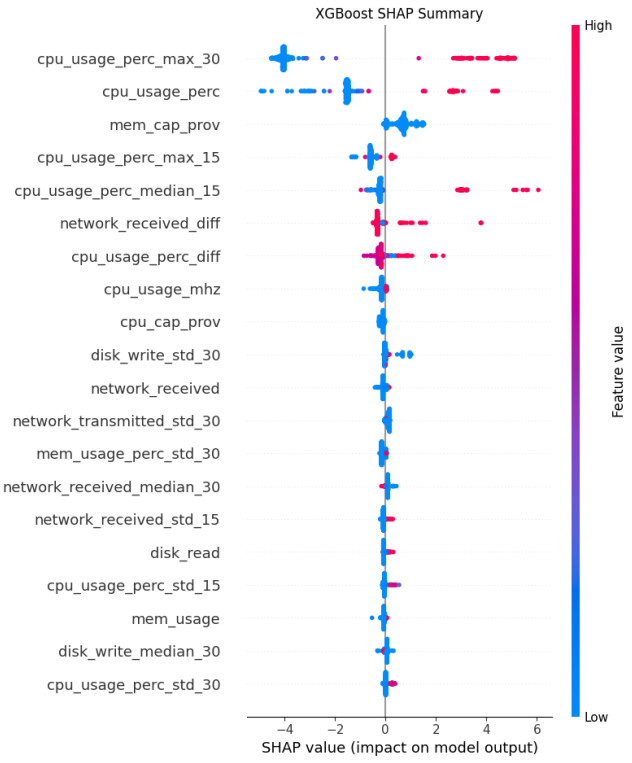

Figure 12. SHAP Summary

split, this is highly applicable to predicting future usage in real scenarios unlike other studies conducted on this.

5.1 Implications of Results

Both of our ensemble tree models outperformed Logistic Regression with Random Forest being the most effective model. The biggest difference across models was seen in precision where Logistic Regression would tend to produce 3x more false alarms than Random Forest with a precision of 0.9355. Since all the models show almost perfect recall, the features seem to have a strong impact in their ability to identify CPU usage spikes.

In particular, the engineered features played an important role in our models' performances with the rolling max usage for 15 and 30 minutes being important to our predictions. The feature importance plots and SHAP analysis show additional impacts of other features on our predictions as well. A monitoring agent deployed by a cloud hyperscaler or organization could leverage a small subset of metrics to capture the majority of performance for these models.

5.2 Tradeoffs

When we look at applying the results from this project to true cloud deployments, the key tradeoff is between precision and recall. There is a clear balance between unnecessary migrations or resizes versus missing spikes in CPU usage. An organization that has tighter SLAs and more sensitive customers might opt towards higher recall since the costs of proactive actions would be overshadowed by violations or customer impact.

However, since recall was high across all models, we can focus on false alarms and minimize costs to organizations. We saw that ensemble tree models achieved far less false alarms in our study. Given the feature importance findings, a 15 or 30 minute prediction window would provide organizations enough time to perform a live migration which typically takes one to five minutes to scale VMs proactively [9, 33].

5.3 Limitations

The GWA-T-12 Bitbrains fastStorage trace dataset only contains limited information for around 1,250 VMs which may not be applicable to larger enterprise workloads. The dataset may also not be as applicable for more modern workloads, especially with its concentration of financial workloads. Modern AI inference workloads and containerized instances operate differently and require additional metrics to predict accurately. This would require a wider prediction window to understand seasonal demand fluctuations as well.

VMs also typically share a physical host and so noisy neighbor effects could affect CPU usage patterns observed. The chronological train-test split does not allow us to look closely at activity across VMs. A real production deployment of this solution would require more detailed feature information for each interval.

5.4 Future Work

To make this project more generalizable and applicable to modern workloads, more recent datasets of traces like Google cluster [13] or Azure workloads [10] would broaden the scope. It could also be interesting to utilize a larger prediction window and see if models could predict spikes in CPU usage even earlier for better resource planning. Sequence models like LSTMs could help to capture this, but tree models may still perform stronger [31]. Measuring true impacts of this in a closed-loop system could help to truly measure impacts on costs and SLA violation reductions.

Cite This Work

Subhan Chaudry. 2026. *Predicting Workload Instability and Resource Utilization in Cloud Virtual Machines.* Technical Report, University of Texas at Austin.

References

[1] Alam, M., Shakil, K. A., and Sethi, S. (2016). Analysis and clustering of workload in Google cluster trace based on resource usage. In *2016 IEEE Intl. Conf. on CSE*, pp. 740–747.

[2] Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4, 40–79.

[3] Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of VMs. *Concurrency and Computation*, 24(13), 1397–1420.
[4] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.
[5] Buyya, R., Srirama, S. N., et al. (2018). A manifesto for future generation cloud computing. *ACM Computing Surveys*, 51(5), 1–38.
[6] Calheiros, R. N., Masoumi, E., Ranjan, R., and Buyya, R. (2015). Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Trans. Cloud Computing*, 3(4), 449–458.
[7] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *JAIR*, 16, 321–357.
[8] Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proc. 22nd ACM SIGKDD*, pp. 785–794.
[9] Clark, C., Fraser, K., Hand, S., et al. (2005). Live migration of virtual machines. In *Proc. 2nd USENIX NSDI*, pp. 273–286.
[10] Cortez, E., Bonde, A., Muzio, A., et al. (2017). Resource central: Understanding and predicting workloads for improved resource management. In *Proc. 26th ACM SOSP*, pp. 153–167.
[11] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proc. 23rd ICML*, pp. 233–240.
[12] Dhamane, G. (2023). GWA Bitbrains dataset. Kaggle. https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains
[13] Di, S., Kondo, D., and Cappello, F. (2014). Characterizing and modeling cloud applications/jobs on a Google data center. *J. Parallel and Distributed Computing*, 74(10), 2967–2981.
[14] Duggan, M., Mason, K., Duggan, J., Howley, E., and Barrett, E. (2017). Predicting host CPU utilization in cloud computing using recurrent neural networks. In *2017 12th ICITST*, pp. 67–72.
[15] Farahnakian, F., Ashraf, A., Liljeberg, P., et al. (2015). Using ant colony system to consolidate VMs for green cloud computing. *IEEE Trans. Services Computing*, 8(2), 187–198.
[16] Gill, S. S., Buyya, R., et al. (2019). Transformative effects of IoT, blockchain and AI on cloud computing. *Internet of Things*, 8, 100118.
[17] Guo, Y., Stolyar, A. L., and Walid, A. (2023). Machine learning for predictive deployment of cloud resources: A survey. *ACM Computing Surveys*, 55(12), 1–38.
[18] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.
[19] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Trans. KDE*, 21(9), 1263–1284.
[20] Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gen. Computer Systems*, 28(1), 155–162.
[21] Ismaeel, S. and Miri, A. (2015). Using ELM techniques to predict data centre VM requests. In *IEEE CSCloud*, pp. 80–86.
[22] Khan, A., Yan, X., Tao, S., and Anerousis, N. (2012). Workload characterization and prediction in the cloud. In *2012 IEEE NOMS*, pp. 1287–1294.
[23] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *J. Grid Computing*, 12(4), 559–592.
[24] Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *NeurIPS*, 30.
[25] Mason, K., Duggan, M., Barrett, E., et al. (2018). Predicting host CPU utilization in the cloud using evolutionary neural networks. *Future Gen. Computer Systems*, 86, 162–173.
[26] Moreno, I. S., Garraghan, P., Townend, P., and Xu, J. (2014). Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. *IEEE Trans. Cloud Computing*, 2(2), 208–221.
[27] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. *JMLR*, 12, 2825–2830.
[28] Prevost, J. J., Nagothu, K., Kelley, B., and Jamshidi, M. (2011). Prediction of cloud data center networks loads using stochastic and neural models. In *2011 IEEE SoSE*, pp. 276–281.
[29] Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys*, 51(4), 1–33.
[30] Shen, S., van Beek, V., and Iosup, A. (2015). Statistical characterization of business-critical workloads hosted in cloud datacenters. In *IEEE/ACM CCGrid*, pp. 465–474.
[31] Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81, 84–90.
[32] Toka, L., Dobreff, G., Fodor, B., and Sonkoly, B. (2020). ML-based scaling management for Kubernetes edge clusters. *IEEE Trans. NSMA*, 18(1), 958–972.
[33] Wood, T., Shenoy, P., Venkataramani, A., and Yousif, M. (2007). Black-box and gray-box strategies for virtual machine migration. In *Proc. 4th USENIX NSDI*.

Appendix 1: AI Usage

AI tools were only used for brainstorming topics for exploration, debugging code issues, and collecting potential research papers for references. All code, research paper writing, and interpretation of results were completed independently by the author without any AI assistance.