Student Names: Rohan Mehta (rm3500), Nanda Kishore Siddabasappa (ns3212)

Instructor Name: Predrag Jelenkovic

EECS E6690 Topics in Data-driven Analysis and Computation: Statistical Learning in Biological & Information Systems

9 May 2018

Statistical Analysis of Cloud Datacenter Workloads

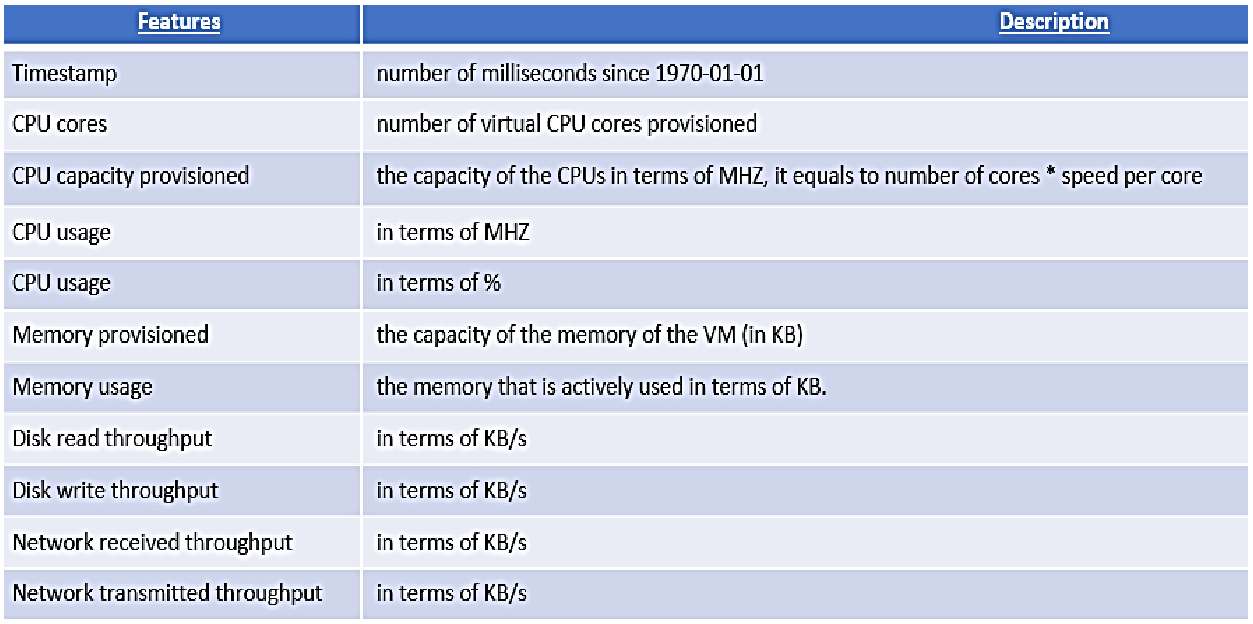
Introduction

Cloud datacenters are being widely used in today's world. They consume about 2 to 3 percent of the total annual power generated in the United States. It will be very beneficial for the cloud hosting companies if they can predict their resource usage beforehand to improve their efficiency and performance. To do so, we need to understand the underlying workload characteristics in order to efficiently plan and manage the resources. Hence, we are analyzing workload traces from a distributed datacenter servicing various types of application workloads and formulating a time series for the features. The traces contain information about memory, CPU, network I/O, and disk I/O.

In the upcoming sections you will see how we are using the models and forecasting techniques such as Polynomial Regression, ARIMA, and Prophet to predict usage of different resources after comparing the results obtained by these models.

Data Set and Summary of Reference Paper

Our dataset comes from a distributed datacenter from *Bitbrains* containing performance metrics of 1250 VMs. They are a specialized service provider with expertise in managed hosting and business computation for enterprises. Numerous major financial institutions like ING Group, International Card Services, Aegon N.V. Life Insurance, etc. are their customers. *Bitbrains* hosts applications that are usually used at the end of a financial quarter (for example, financial reporting softwares). The VMs in our trace are connected to fast storage area network (SAN) storage devices. It includes a large quantity of application servers and compute nodes having high performance storage devices. The performance metrics of our dataset are for one-month duration collected at interval of 5 minutes for each observation (i.e. approximately 8600 observations per VM). The format of each file of our dataset is row-based. Each row represents an observation of 11 performance metrics. The format of each row is:



Features	Description
Timestamp	number of milliseconds since 1970-01-01
CPU cores	number of virtual CPU cores provisioned
CPU capacity provisioned	the capacity of the CPUs in terms of MHZ, it equals to number of cores * speed per core
CPU usage	in terms of MHZ
CPU usage	in terms of %
Memory provisioned	the capacity of the memory of the VM (in 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

Table-1: Dataset features and their descriptions

In the reference paper, a comprehensive study of business-critical workloads hosted in cloud datacenters has been done by the authors. Two large scale and long-term workload traces corresponding to requested and actually used resources in a distributed datacenter have been collected and studied in-depth. Mostly, their study involves plotting cumulative distribution functions (CDF) of all the parameters, and then trying to find correlation between them. Their four main findings from it are:

1) Greater than 60% of VMs used less than 8GB worth of memory and 4 cores of CPU. Between memory and CPU, a strong non-negative correlation was noticed.

2) Less than 10% of requested resources are actually used, on an average.

3) Depending on the type of resource, variation in peak workloads was observed to vary from 10 times of mean workload to 10,000 times of mean workload.

4) Short-term predictions for memory and CPU usage was often possible; whereas, every-day patterns were observed in network I/O and disk I/O.

Reproduction of Results from Reference Paper

Since the entire paper is majorly talking about plotting CDF of various parameters and then finding auto-correlation and correlation between them, we have gone ahead and taken the liberty of reproducing only some of these results. Further, we use these plots later on to do statistical learning on the data and then attempt to predict near future values.

Figure-1 is the CDF plot for number of CPU cores requested at the VMs. The x-axis values are 1, 2, 4, 8, 16, and 32 cores. The y-axis sums up to 1250 VMs (as is the total number of VMs in this dataset). Note that this is not happening in our plot because some of the VMs are requesting 0 CPU cores. These have been excluded from the plot because virtual machines using 0 cores simply means that the particular VM is not being used by any application.

Figure-2 is a similar CDF plot for amount of memory requested at the VMs. The x-axis values are 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512 GB. The y-axis should sum up to 1250 VMs. Note that this not happening in our plot because some of the VMs are not using any memory and are only serving basic computational functions. These have been excluded from the plot since they provide no insight on usage of memory in a datacenter.

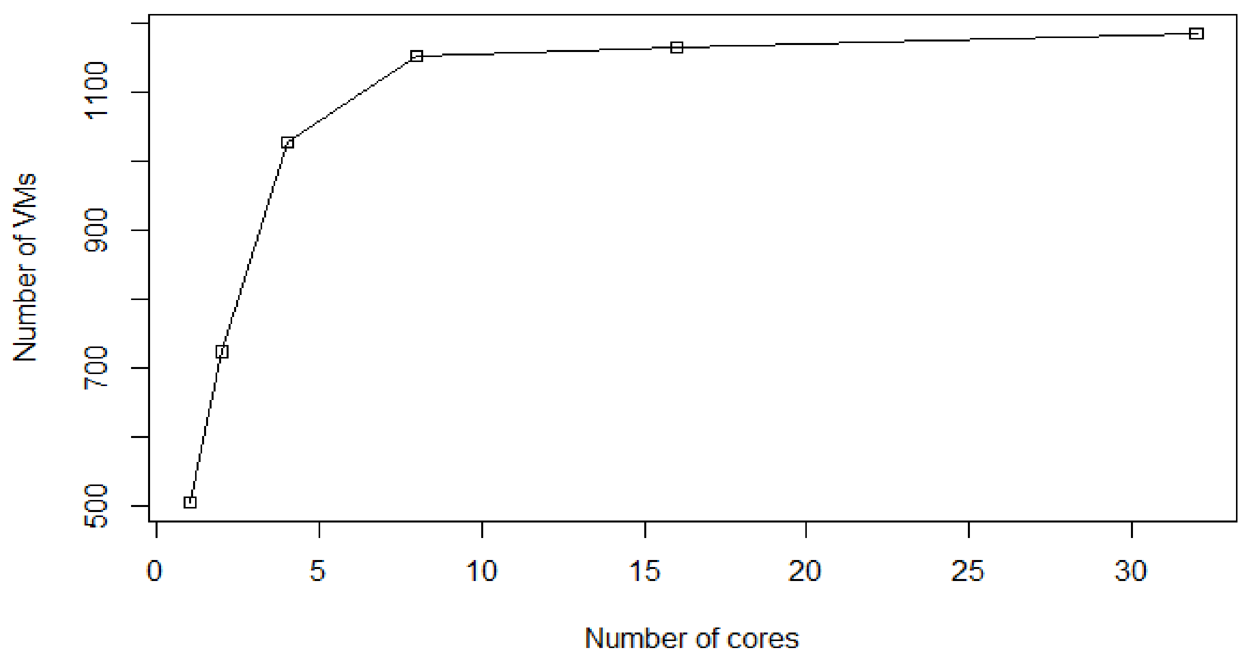


Figure-1: CDF of number of requested CPU cores

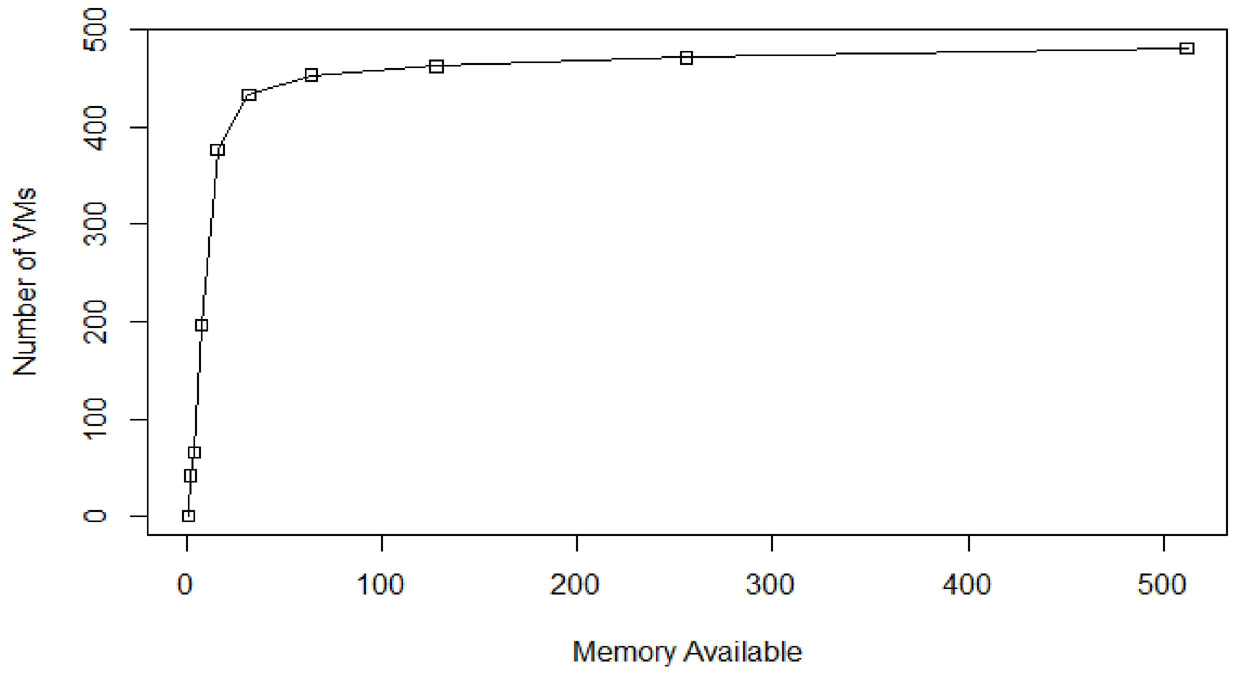


Figure-2: CDF of amount of requested memory

We have also reproduced plots for CPU and memory usage over time, as well as plots for auto-correlation functions for CPU and Disk I/O usage. We would like to discuss about them in the next section since it also involves our new approach of learning and prediction.

New Approach and Results

We have focused our problem on studying percentage CPU usage, percentage memory usage, total Disk I/O usage, and total network bandwidth usage over a period of one month (as is the data collected). We have plotted these parameters as a time series. After that, we applied three statistical learning models on each of these four parameters and then figured out which model gives the best prediction.

Polynomial Regression: We first applied simple linear regression to each of these four features. After that, we used polynomial regression with order 3 and 25. As expected, we see a better fitting curve for order 25 as compared to order 3 and linear regression. We then created a new data-frame to predict resource utilization 33.33 hours in future. We have checked this prediction for all four features. It was observed that the prediction values are extremely bad and imprecise.

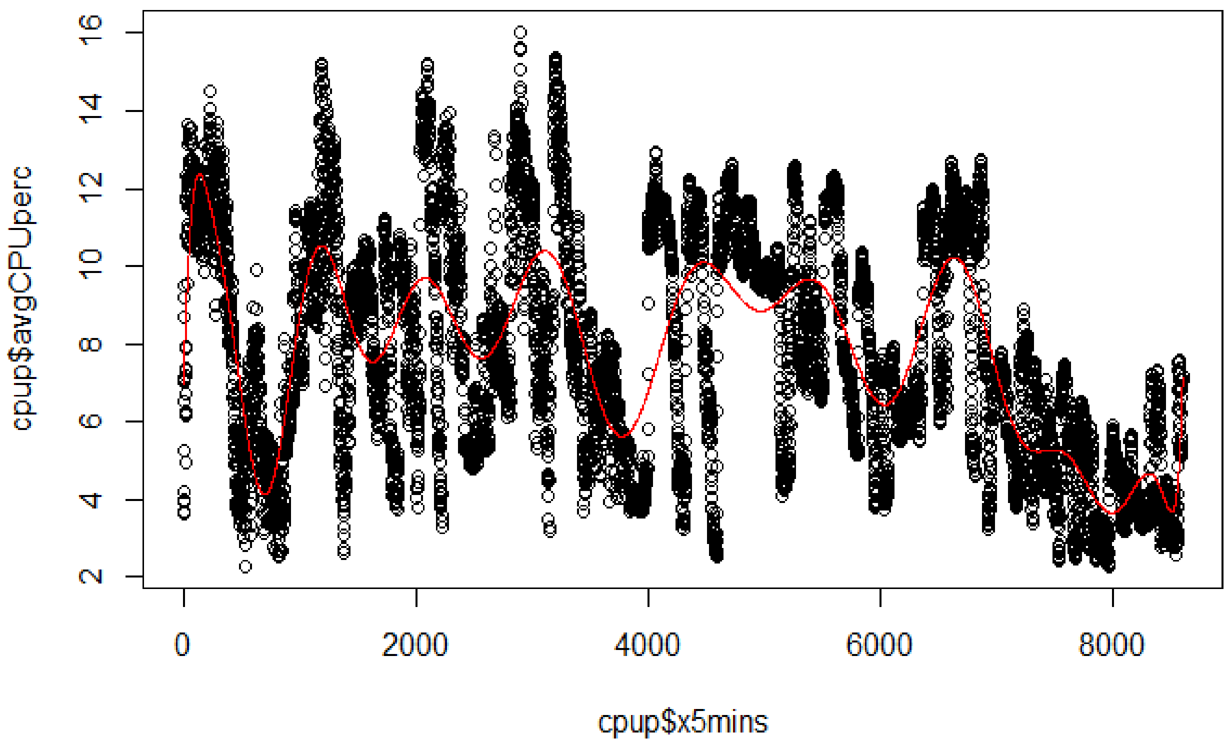


Figure-3: Polynomial Regression of order 30 to % CPU analysis

ARIMA (Autoregressive Integrated Moving Average): These are a fairly old (year 1976) set of forecasting models used widely across many businesses for time series analysis. It is also known as Box-Jenkins approach.

ARIMA checks for stationarity and constant variance in the data to study autocorrelations. A random variable that is a time series is stationary if its statistical properties do not depend on the time at which the series is observed. So, periodic or seasonal trends will not be considered as stationary as their future output will depend on the value of time series at that moment. If a series is not stationary, it can be made stationary using techniques like differencing.

A non-seasonal ARIMA model is written as "ARIMA(*p*,*d*,*q*)" model. The auto regressive component *p* is the number of past values used in the regression equation. For example, AR(3), or equivalently, ARIMA(3,0,0), can be written as:

$$Y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varphi_3 y_{t-3} + e_t$$

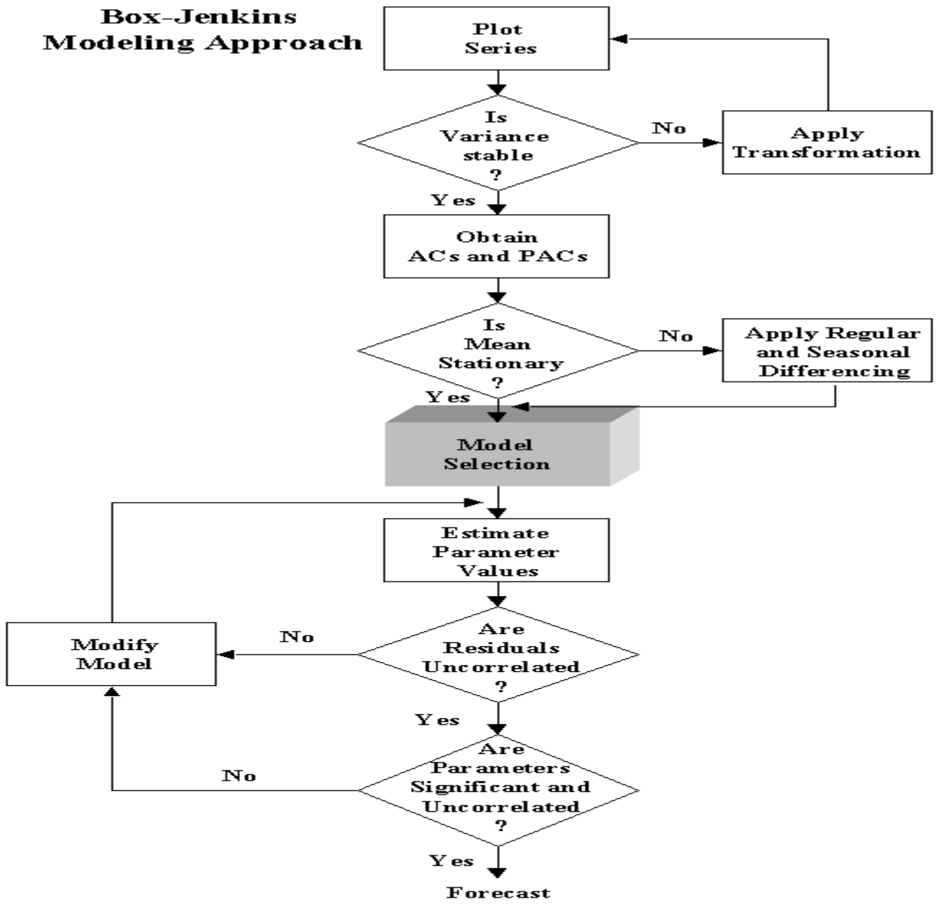
The *d* represents the number of non-seasonal difference terms in the integrated (I(d)) component needed for stationarity and stabilizing the series. "Differencing of a series" involves subtracting its current and previous values for *d* times.

$$y_d = Y \text{ differenced } d \text{ times}$$

The moving average component *q* refers to the number of error terms required in the model. It is a cumulation of previous error terms e_t; similar to the *p* component. For example, ARIMA(3,0,3) can be written as:

$$Y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varphi_3 y_{t-3} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \theta_3 e_{t-3}$$

ARIMA is a very complex technique to use and requires high level of expertise. An entire set of cleaning and identification steps are involved to figure out these parameter values. We have used generic commands after reading from R Documentation and other sources to implement it to the best of our understanding. In simple words, we can use the analogy of a sugarcane juicer. In one iteration, it is difficult to extract all of the juice from a sugarcane stick. That is why we run the molasses through the juicer multiple times until no more juice can be extracted from it. ARIMA uses same ideology. The final residual should resemble white noise. If that is not the case, then it means that more information can still be extracted from the data. In other words, an ARIMA model is analogous to a "filter" that separates a signal from its noise, and then extrapolates the signal to get future forecasts. Figuring out ways to repetitively is what separates an experienced and seasoned user of ARIMA from others.


Box-Jenkins Modeling Approach
Plot Series
Is Variance stable ?
No
Apply Transformation
Yes
Obtain ACs and PACs
Is Mean Stationary ?
No
Apply Regular and Seasonal Differencing
Yes
Model Selection
Estimate Parameter Values
Are Residuals Uncorrelated ?
No
Modify Model
Yes
Are Parameters Significant and Uncorrelated ?
No
Yes
Forecast

ARIMA models work best on long and stable series because they rely on past values directly. And for this reason, they cannot explain the underlying data mechanism and can only approximate the historical patterns.

We followed the steps as mentioned in reference number [6] to implement ARIMA in our scenario. We first cleaned the data, then decomposed it into seasonal components, then did the augmented Dickey-Fuller (ADF) test on our resulting stationary time-series to get the values of *p*, *d*, and *q*. Then we ran the autocorrelation function (ACF) and partial autocorrelation function (PACF) to understand their namesakes with the lags and accordingly re-choose the parameters, if necessary. Next, we fit an ARIMA model using *auto.arima()* function with appropriate parameters as obtained earlier. Finally, after evaluating the ACF and PACF as obtained from the ARIMA model, we made a forecast for 75 hours in future. The output shown will have two confidence intervals, viz. 95% and 92%, with an average line shown in between.

We did all these steps for moving average of order 7 as well as of order 30.

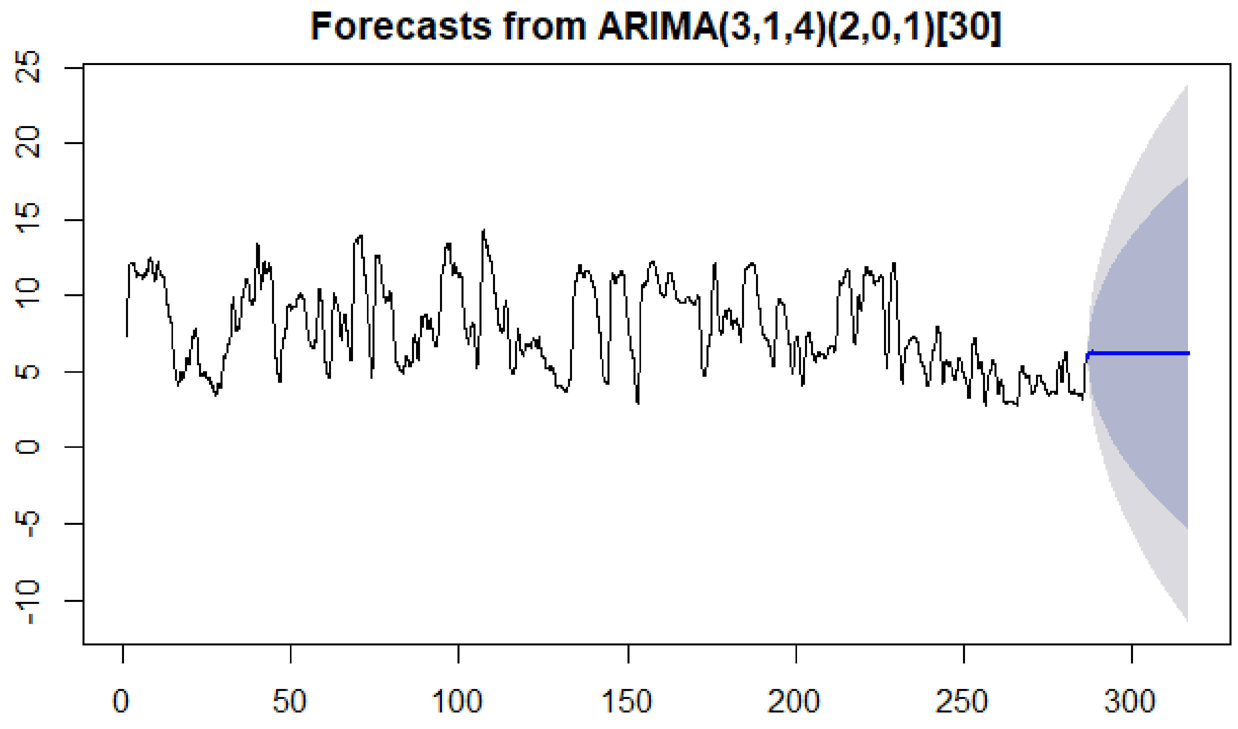


Figure-4: ARIMA forecasting for % CPU usage at moving average of order 30

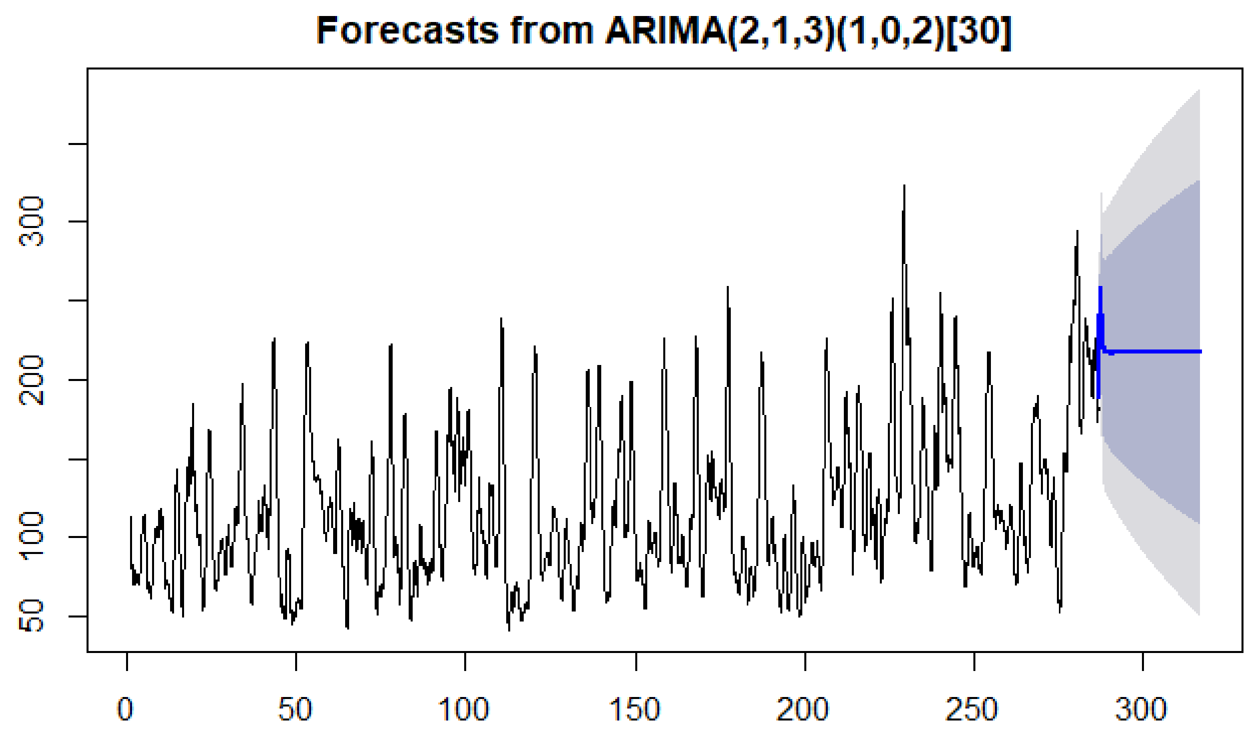


Figure-5: ARIMA forecasting for total network throughput at moving average of order 30

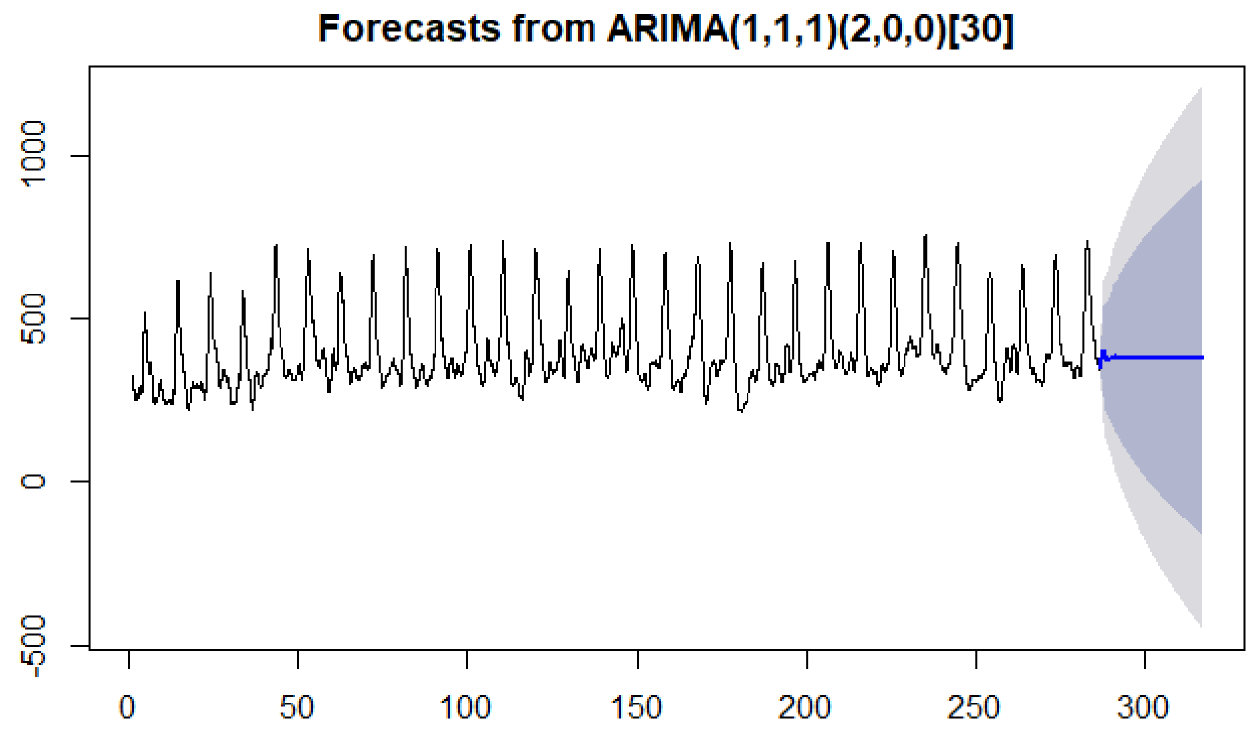


Figure-6: ARIMA forecasting for total disk throughput at moving average of order 30

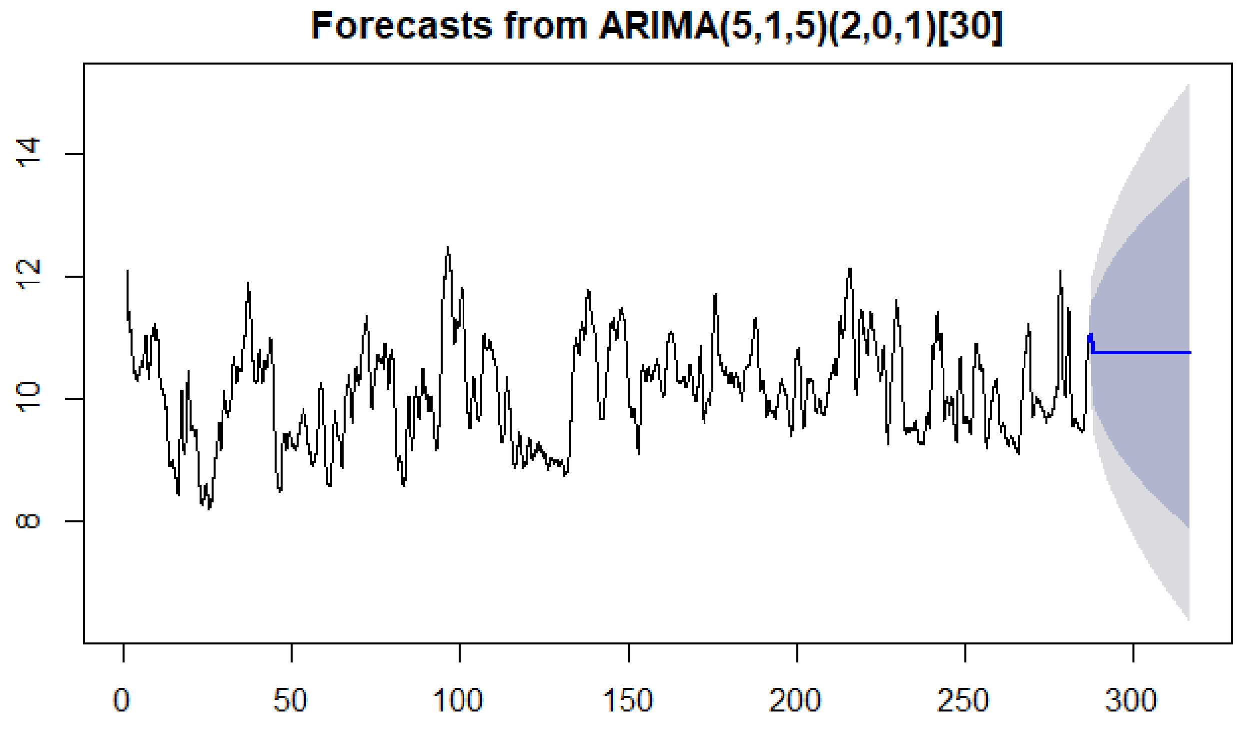


Figure-7: ARIMA forecasting for % memory usage at moving average of order 30

Prophet: As we saw the already existing forecasting model such as ARIMA, Generalized Additive Model (GAMs), etc. need an experienced analyst with good feature knowledge to fine tune the model to get better prediction. It is often hard to find a person or develop a with good analytical skills and feature knowledge. Its proven to be hard to develop a model which can be flexible and also take care of useful assumptions and heuristics.

Prophet was designed by Facebook and open-sourced to the public in February 2017. They implemented a modular regression model with interpretable parameters in a way that amateur as well as novice analysts can intuitively adjust them for forecasting according to their domain knowledge about the time series. They need not know the actual importance of the parameter.

The prophet model primarily contains 3 main components – trend, seasonality, and holidays/events. It is mathematically modelled as follows:

$$y(t) = g(t) + s(t) + h(t) + Error$$

The trend component g(t) corresponds to the changes that are non-periodic. Prophet uses trend model such as the linear trend model, automatic changepoint selection, and nonlinear saturating growth. s(t) represents time series changes that tend to be periodic such as observations that show repetitive behavior in days, months, year, etc. and rely on Fourier series techniques. h(t) takes into account the holidays or any irregularities/outliers that one could expect as a part of normal trend in data. The error term takes care of any peculiar changes which are not included in the model. A parametric assumption of normal distribution can also be made for the error term.

Prophet uses a combination the above stated models and automates the forecast performance evaluation.

In short Prophet model takes the feature that has to be forecasted as a variable "y" and the corresponding timestamp as "ds" and runs these inputs through a pipeline of models which involves numerous combinations of different models as specified in [9]. The function *forecast()* takes the trained model "prophet" and the duration up to which we want the forecast to be made as input and plots the forecast on a time series scale. We made a forecast up till 75 hours in future. The output is shown with some confidence interval.

We did this process for four different sets of training data, viz. 1/3rd, one-half, 2/3rd, and 100% of the available dataset. Plots for all these results are attached in the Rmd output file.

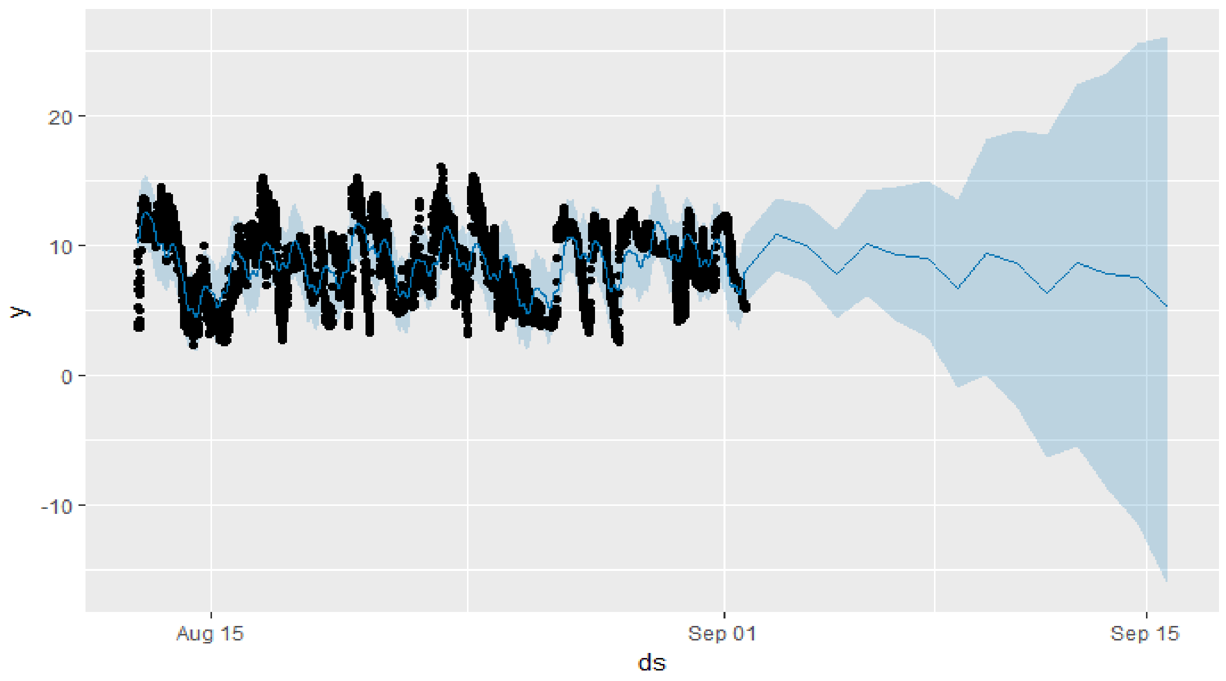


Figure-8: Prophet forecasting for % CPU usage

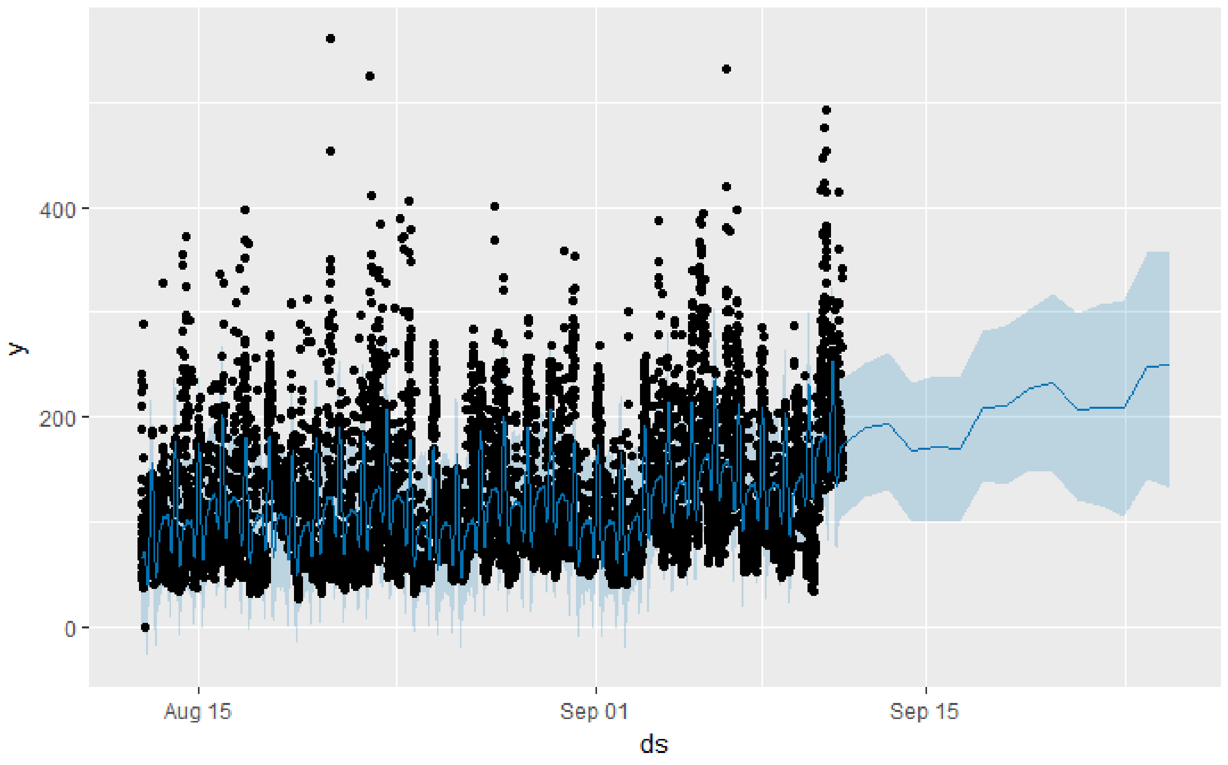


Figure-9: Prophet forecasting for total network throughput

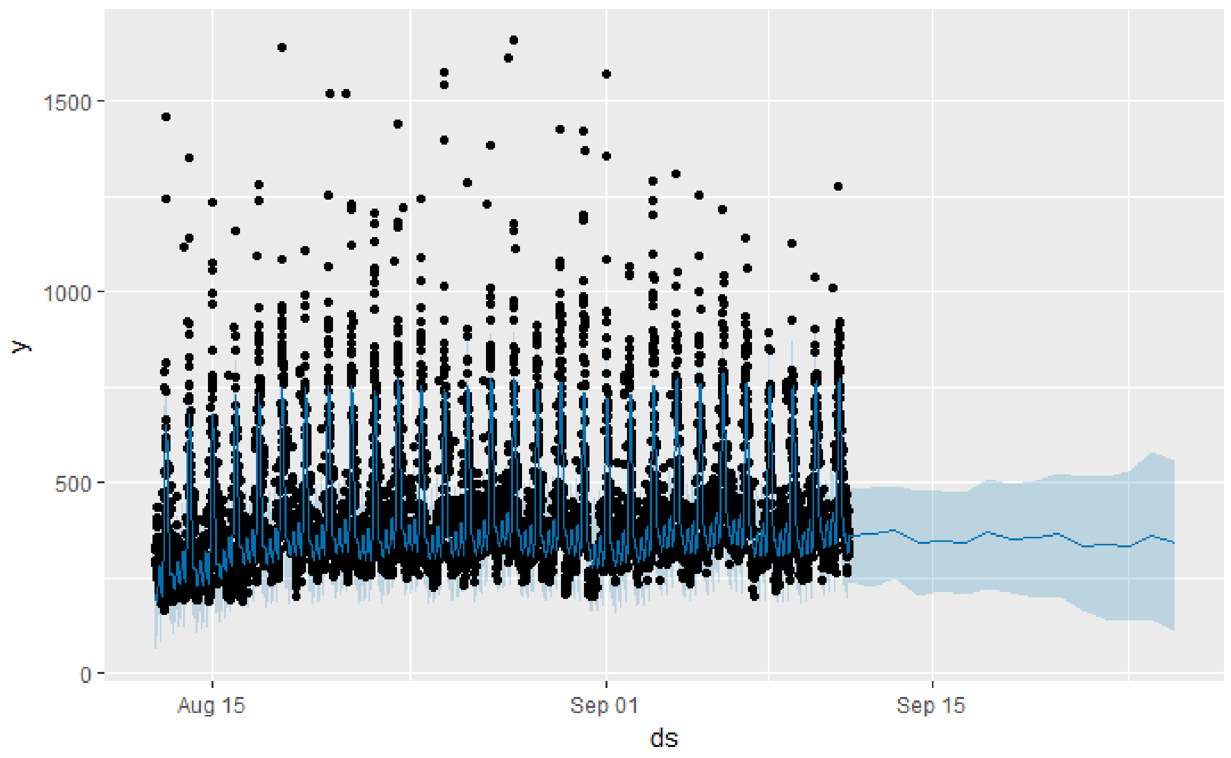


Figure-10: Prophet forecasting for % memory usage

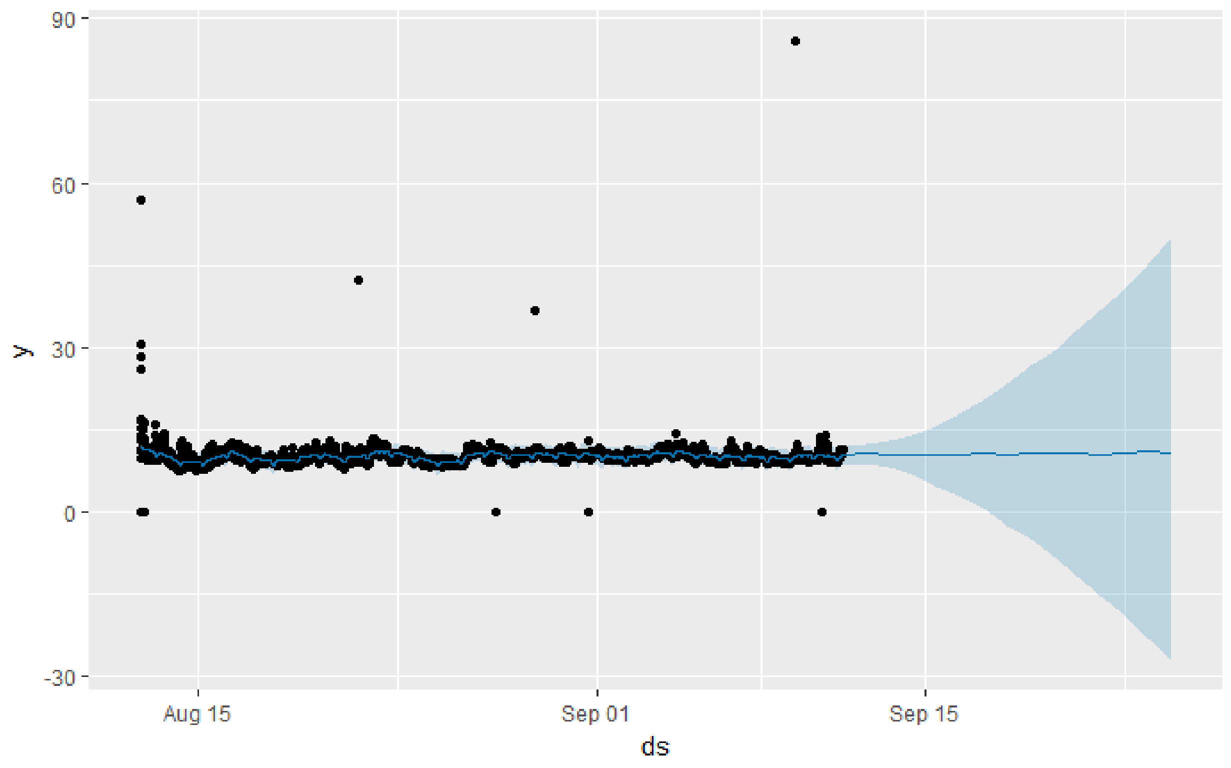


Figure-11: Prophet forecasting for % memory usage

Conclusions

We first used various models as studied in class like polynomial regression, linear regression, etc. for our time series analysis of our workload features. We realized that these basic techniques are extremely inefficient and give us absurd and impracticable future predictions. Clearly, this can be explained by the phenomenon of *overfitting*.

We found out that Autoregressive Integrated Moving Average (ARIMA) forecasting model and Prophet model are two of the best techniques out there to predict a quantity in the future based on time series data, as well as explain its historical patterns; which is what is desired in our case.

Prophet model generally outperforms ARIMA model when it comes to predicting future resource usage. It is also less complicated to use as compared to ARIMA.

We need more data, preferably at least for 1 year, so that we can make more accurate predictions for resource utilization. Both ARIMA as well as Prophet model advocate to use data of at least 1-year duration in order to get acceptable and accurate future predictions.

Future Work

If more data is available, we can do similar analysis for different topological models and get insights on resource management principles.

We can try different time-series forecasting models (like Neural Networks, Exponential Smoothing, etc.) as alternative learning techniques.

Outliers/Irregularities in data can be studied further to identify types of malicious attacks in a cloud datacenter (e.g. DDoS).

Works Cited

[1.] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. 2nd edition, Springer, 2009.

[2.] Lecture slides

[3.] Siqi Shen, Vincent van Beek, and Alexandru Iosup. *Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters*, the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015, Shenzhen, China

[4.] https://www.rdocumentation.org/

[5.] http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains - data set

[6.] https://www.datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials - ARIMA model

[7.] https://www.bistasolutions.com/resources/blogs/5-statistical-methods-for-forecasting-quantitative-time-series/ - ARIMA model

[7.] https://research.fb.com/prophet-forecasting-at-scale/ - Prophet Model

[8.] https://towardsdatascience.com/using-open-source-prophet-package-to-make-future-predictions-in-r-ece585b73687 - Prophet Model

[9.] Sean Taylor and Benjamin Latham. *Forecasting at Scale*, PeerJ Preprints, https://doi.org/10.7287/peerj.preprints.3190v2, CC BY 4.0 Open Access, 27 Sep 2017