Workload Prediction of Alibaba Cluster Dataset

Mansi Mehta
Computer Science Department
University of Georgia
Athens, USA
mansi.mehta@uga.edu

Niyati Shah
Computer Science Department
University of Georgia
Athens, USA
nhs01063@uga.edu

Priyadarshini Das
Computer Science Department
University of Georgia
Athens, USA
pd11197@uga.edu

***Abstract*—Workload prediction is a crucial part for almost every organization in order to efficiently manage the resources and smooth uninterrupted flow of work. During holiday shopping seasons the websites witnesses a heavy influx of workload, which may cause the website to experience very high response time, eventually hampering the sales of the company. Hence predicting the future workload would avert such problems. Predicting the future workload is difficult for particular cloud applications. This paper presents various workload prediction models of four different classes (Naïve model, Regression Model, Time series-based model, Neural Network model) to predict the workload of Alibaba cluster dataset. After implementing these models, their performance has been evaluated through two distinct evaluation metrics 1) Root Mean Square Error 2) Mean Absolute Percentage Error. The evaluation result shows that the main approach LSTM (Long Short-Term Memory) outperforms the other implemented models from the accuracy and comparative computational overhead point of view. The accuracy obtained for LSTM (Long Short-Term Memory) is found to be 93.6%.**

Keywords—Time series analysis, Sliding window approach, Workload prediction, Evaluation metrics

I. INTRODUCTION

A study last year revealed that the average cart abandonment rate was 65%. Best Buy lost millions when its website and applications went down for almost three hours. Even Kohl's experienced worse downtime when its website was down for most of the first half of the day due to heavy traffic [3]. Spotify which had an offer of google home mini for 0.99$, on normal days, it is expected to have an average workload but suddenly because of this offer a lot of people used the website because of which it had a very high response time. Similarly, during Black Friday peak hours Amazon was down for 20 minutes, Best Buy had periodic outages over 4 days and Walmart had an average download time of ~260 seconds. It was because too many users lead to overload, that led us to the idea of our project that if we can predict the workload for a particular time and if there is huge workload predicted for that particular time interval then necessary actions (example: provide more numbers of CPU's/ containers) can be taken in order to encounter such situations.

So, the questions arise, such as: Is the website or application all set to handle the heavy influx of traffic and offer outstanding customer experience? Shouldn't the statistics of the previous years be used to determine the amount expected workload in the future for optimizing the performance of website enabling seamless online business transactions? [3]

The motivation behind this idea is that there is a very high workload on a website during the holiday shopping season and because of this deviation of the website from its normal behavior cause the website to slow down or even crashes after some time. The shopping experience becomes too frustrating to many customers due to website outages. According to a recent survey Harris 2015 poll, 46% of shoppers have said that, they will never come back to a slow website. Moreover, as per a recent study by SOASTA, 18% of shoppers have cited too slow web pages as the main cause to abandon carts. [3] Hence, this stated problem can be solved when we can predict the future workload, so that the resource allocation to the scheduler can be done in much advance.

The goal of this work is to comprehensively evaluate workload predictors and to find out which predictor model works the best for our dataset. This leads us to the first question we wish to seek the answer to:

Question#1: In order to predict future data which workload predictor has the lowest error for our dataset (in terms of RMSE, MAPE) i.e. highest accuracy for the number of jobs prediction?

In order to decide on the portion of the data to be chosen for training and testing so that the predicted results are more accurate arises us to the next question:

Question#2: What should be the proportion of Training and testing dataset?

It is normally observed that when a lot of data is taken for training, the model overfits. Hence, if we take the recent past that is the sliding window concept in which we take the recent past of a particular size of the window and predict the value of the time instance after the last time interval of the window. This leads us to the next question.

Question#3: In case of Sliding window approach, what should be the appropriate window size for optimum accuracy?

This paper describes the implementation of 10 workload predictors and their evaluation to identify the best predictor. Apart from this the paper also brings to notice about the computational time of these predictors. Section-II describes the related work, Section-III explains the approach of the project, Section-IV contains the evaluation details, Section-V is about discussions and lessons learned from the project and finally, we concluded the paper with Section-VI.

II. RELATED WORK

A lot of research is conducted for prediction of issues like stock market rate, google trace analysis, weather etc. We are the first group to predict future workload of Alibaba Cluster data Trace. According to Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, Tongxin Bai in Imbalance in the Cloud: An Analysis on Alibaba Cluster Trace in 2017 IEEE International Conference on Big Data (BIGDATA) mentions about their work on performing a deep analysis on Alibaba dataset. The analysis reveals several important insights about different types of imbalance in the Alibaba cloud. It

highlights that the data trace has spatial and temporal imbalances. The paper clearly states that batch and online jobs have serious imbalanced instance numbers, resource utilization and duration.[15] Referring to the paper by Abdelhadi Azzouni and Guy Pujolle "A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction" in journal: CoRR, in the year 2017, we found that they have used the similar approaches as ours. In this paper also, they designed models using LSTM, ARIMA, AR etc. They also implemented the sliding window approach for the same. Even they achieved high accuracy for LSTM without sliding window.[16] In the paper, "Time Series Modelling and Forecasting: Tropical cyclone Prediction using Arima Model" by A.Geetha and Dr. G.M. Nasira in the 2016 IEEE conference proposes ARIMA technique for predicting tropical cyclones in India. It introduces different steps involved while implementing different time series models for predicting the cyclones.[17] In the paper, by Ayodele A. Adebiyi, Aderemi O. Adewumi "Stock Market Prediction using Arima model" in IEEE 2014 presents an extensive process of building stock price predictive model using the ARIMA model.[18] However, the goal of our project is different from others. Firstly, we predict the workload of Alibaba cluster dataset and furthermore we are geared towards the performance of workload predictors used for predicting our dataset.

III. Approach

We are using the Alibaba Cluster dataset, which is in the csv file format. It provides the number of instances arriving in different time stamps for a total of 12 hours duration. In order to predict the future workload, we have used various forecasting techniques from four different classes: 1) Naïve Model 2) Regression Model 3) Time Series based Model 4) Neural Network Model

A. *Naïve Model:*

Naïve approach is the very basic approach for workload prediction. Two workload predictors of this category implemented in our project are- naïve predictor and mean based predictor.

B. *Regression Model:*

In this class we have used linear regression as a predictor.

C. *Time Series Based Models*

In time series-based approach various predictors exist. In our project work we have used six different predictors of this class- MA (Moving Average), WMA (Weighted Moving Average), ES (Exponential Smoothing), AR (Autoregressive), ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average)

D. *Neural Network Model*

Under this class we have used LSTM (Long Short-Term Memory) for predicting the future workloads.

Note: AR, ARMA, ARIMA and LSTM models are implemented both without sliding window and with sliding window.

System Design: First, we preprocessed and resampled our dataset. Then this resampled dataset is passed through the above mention workload predictors. As an output it gives the predicted data.

Figure 1. System Design Schematic representation

Challenges: During the whole process we faced some difficulties in implementing the models for our project. First, our dataset is random with very wide range of data. So, it affected the prediction accuracy of our models. Second, in selecting the suitable parameters we used the hit and trial method (principle of grid search algorithm). Third, it is a very tedious job to find suitable batch size and epoch value in LSTM model. Another challenge we faced in deciding the training and testing proportion of the dataset. We will discuss in detail what steps were taken to overcome these challenges.

Implementation: We implemented all the above-mentioned workload prediction models in python3 programming language using jupyter notebook. For reading the data from csv files, plotting graphs and for various other operations in the program, python library packages like csv, pandas, matplotlib, numpy are imported. The predictors AR, ARMA and ARIMA are implemented by using the statsmodels library of python. Python's scikit-learn library and tensor flow library is imported for implementing the LSTM model.

The detail description of how all these models are implemented is as follows.

For implementing these models, the dataset is divided into train and test dataset. The model is first trained with the train sample. Then for the test dataset the predicted values are obtained by the model. These predictors are tested with different proportions of test and train dataset and the optimum proportion is considered finally for evaluation. For LSTM 80% of the dataset is taken as train data and the rest 20% is used for test dataset. For other models (AR, ARMA, ARIMA) 65% of the dataset is taken as train data and the rest 35% is used for test dataset.

Let,

A_t = Actual number of jobs at time period t

F_t = Forecasted number of jobs at time period t

1) ***Naïve Predictor:*** This predictor forecasts the very next period's number of tasks based on the just previous period's number of tasks. $F_t = A_{t-1}$

2) ***Mean Predictor:*** In this predictor, the mean of the whole past data is considered as the predicted value.

3) ***Linear Regression:*** This is an alternative approach to address the prediction problem. It models the relationship between one or more input variables x and a dependent output variable y by using a linear equation to observed data.

The linear regression approach determines the unknown parameter values during the training phase, that yields the best-fitted model with the given data points.

4) ***Moving Average (MA):*** In this approach, current periods number of tasks is predicted by considering the average of some numbers of past periods data.

5) ***Weighted Moving Average (WMA):*** The predicted value is a weighted sum of the observed past data. The sum of the weights of each observed data is 1.

6) ***Exponential Smoothing (ES):*** In ES predictor forecast for the current period is obtained by adding the forecast in the last period to a fraction (α) of the error from the last period.

$$F_{t+1} = F_t + \alpha(A_t - F_t) \qquad (1)$$

Value of α must be within the range 0 to 1. The model is tested with different values of α and that value of α is considered which predictor gives the optimum accuracy.

7) ***Autoregression Model (AR):*** AR is a linear combination of the past data. Represented as AR(p), where p is the order of AR. Performance-based approach is used for selecting value of the parameter p. The AR model is tested with different values of parameter p for different combination of test and train dataset.

8) ***ARMA Model:*** ARMA model is a combined model of AR and MA. Represented as ARMA(p,q), where p=order of AR and q=order of MA. On trial and error basis the ARMA model is tested with different p and q values for different train and test dataset. (performance-based or grid-search approach for selecting values of the parameters p and q)

9) ***ARIMA Model:*** ARIMA is the modified form of ARMA. It provides predictions by integrating AR and MA. Represented as ARIMA(p,d,q), where p= order of AR, q=order of MA, d= order of differencing model.

For implementing ARIMA model the dataset must be stationary. Stationarity of the data is verified by the "Dickey-Fuller test". For stationary dataset, the p-value in the results of the "Dickey-Fuller test" must be less than 0.05 and the test statistics must be less than the 5% critical value. As per the results obtained from "Dickey-Fuller test" our dataset is stationary. To determine the suitable values of the parameters p, d and q performance based or grid-search approach is used for our project.

Higher the order of the models like AR, ARMA and ARIMA is not desirable, because it will increase the computational time.

10) ***LSTM Model:*** In LSTM (Long Short-Term Memory) We design a sequential model of a recurrent neural network in which we add layers: the input layer, LSTM layer and the output layer. We compile the model with a loss function, optimizer and number of epochs(evaluation metric) then use the training data to fit the model and then input the testing set to predict values for the test data. The detail implementation steps of the LSTM model are as follows.

Step 1: Pre-processing data: We perform Feature Scaling using MinMax Scaler().Then perform reshaping of data using numpy.reshape. We decide the proportion of training and testing data.

Step 2: Building RNN: We first initialize the model by building a sequential model using Sequential(). Then we add the input layer, LSTM layer and the output layer. We design a sequential model of a recurrent neural network in which we add layers input layer, LSTM layer and output layer. For the LSTM layer, we choose "tanh" as our activation function. We compile the model with an optimizer and loss function. We fit the RNN to the training set and set batch size(training ground) and the number of epochs(evaluation metric) by testing it for various combinations of the values in order to get the minimal loss function.

Step 3: Making Predictions: We then input the testing data to the model to predict values for the test data. After the predicted values return from the model, each value is inversed transform by MinMax Scaler.

Step 4: Visualizing Results: Plotting the actual values and predicted values us pyplot.

Step 5: Evaluations: Using the Evaluation metrics RMSE, MAPE the predicted values and, actual values are mapped and the amount of error is found.

Sliding Window Approach: The sliding window approach maps an input sequence of window size w, and then outputs the predicted data of the very next timestamp in front of the window by using the prediction models. In our project, we have implemented the sliding window approach for AR, ARMA, ARIMA and LSTM models. Workloads are predicted through these models by using a window of different sizes. We considered those window size for the respective model which gives optimum accuracy. In case of our dataset window size 10 for AR, ARMA and ARIMA model and window size 20 for LSTM model predicted the workload with better accuracy than other window sizes.

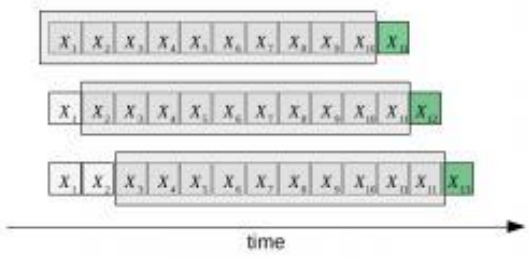

Figure 2. Schematic representation of Sliding Window Approach

IV. EVALUATION

As discussed in the Approach section, we trained and tested the prediction models without sliding window for different proportions of train and test data, and with sliding window strategy. After implementing all these models, our work is evaluated by measuring the accuracy of the predictors in predicting the workloads. The evaluation platform for our project is our laptop with the following specification.

Manufacturer:	Lenovo
Model:	ThinkPad X1 Carbon 5th Signature Edition
Processor:	Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 2.90 GHz
Installed memory (RAM):	16.0 GB (15.8 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

A baseline provides the platform for comparison in forecast performance. It is a point of reference for all other concerned models. The forecasting technique to be considered as a baseline should be easy to implement and naïve of the problem-specific details. For our project, we have considered two baselines, one is "Naïve Predictor" and another one is "Mean based Predictor". These are the most the commonly used baselines. Our main approach is the LSTM Model.

Now, the question arises that "Why have we selected "Naïve predictor" and "Mean based Predictor" as our base model?". MASE is one of the widely preferred evaluation metrics. For calculating MASE, the numerator is the forecast error for a given period and denominator is the mean absolute error of the "Naïve Predictor". MASE can be easily interpreted, as values greater than one indicate that the naïve method performs better than the forecast values under consideration. So, this reflects the importance and prevalence of Naïve Predictor. That is "Naïve Predictor" is considered as a baseline to evaluate and compare the performance of other forecasting models.

Another criterion for baseline selection is that, the baseline forecasting model should be easy to implement. The "Mean based Predictor" is a trivial and very simple predictor which is also considered as baseline in some cases.

For evaluating the accuracy/performance of these workload predictors we have used RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error) as the performance metrics. These are calculated by using the following formula.

$$RMSE = \sqrt{\frac{1}{n}\sum(Actual - Forecast)^2}$$

$$MAPE = \left(\frac{1}{n}\sum\frac{|Actual - Forecast|}{|Actual|}\right) * 100$$

So, the next question is, what is the justification behind using RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error) to evaluate the performance of models used in our project?

There are many ways of measuring the accuracy of a forecast. RMSE is a frequently used measure of the accuracy to compare forecasting errors of a particular dataset and not between datasets, as it is scale dependent. Lower the RMSE value, better is the performance of the model. As in our project, we have only a single dataset, to compare the performance of these models we are using RMSE as an evaluation metric.

RMSE is the square root of the average of the squared errors. The effect of each error in RMSE is proportional to the size of the squared errors. Thus, larger errors have a disproportionately larger effect on RMSE. Consequently, sensitive to outliers.

MAPE (Mean Absolute Percentage Error) is another evaluation metric which we have used in our project for measuring the accuracy of the predictors. MAPE is more intuitive than RMSE. Following example clarifies this. Let for dataset A, the actual value is 10 and the predicted value is 12. Another dataset B has the actual value 4000 and the predicted value is 4002. Then RMSE for both A and B is 1.41. But MAPE value for A is 20% and for B 0.05%. Thus, MAPE is more intuitive. So, in our project for evaluating and comparing the predictors we have used both MAPE and RMSE.

Methodology: To feed the input data to the predictors, the dataset needs to be preprocessed. From Alibaba cluster dataset which we downloaded from the cloud, we used the file batch_task.csv. It has 80554 number of rows with various attributes in columns. Out of all these columns attributes in this file we are only concerned with two attributes, 1) Timestamp 2) number of instances. The timestamps are in milliseconds. We preprocessed and resampled the dataset into 3 minutes time interval and number of instances during these time intervals, which eventually resulted into a dataset with two columns (time_interval and no_of_instances) and 340 rows. For workload prediction, the predictors used this resampled dataset.

Results: After implementing the models both without sliding window and with sliding window, next their performance was evaluated through evaluation metrics RMSE and MAPE. The results of predictors without sliding window approach is shown is Table-I. Table-II contains the results of predictors with the sliding window approach.

Table-I: Evaluation Results of Predictors (without sliding window)

MODEL	RMSE	MAPE (%)
Naïve Predictor	43k	20000
Mean Based Predictor	33k	31752
ES	32k	9354
MA	33k	3008
WMA	35k	1733
AR	52k	18895
ARMA	45k	16761
ARIMA	52k	4581
Linear Regression	22k	39.12
LSTM	577	6.4

Figure-3 to Fig-12, shows the graphs plotted between predicted (red line) vs expected (blue line) workload of the model without using any sliding window.

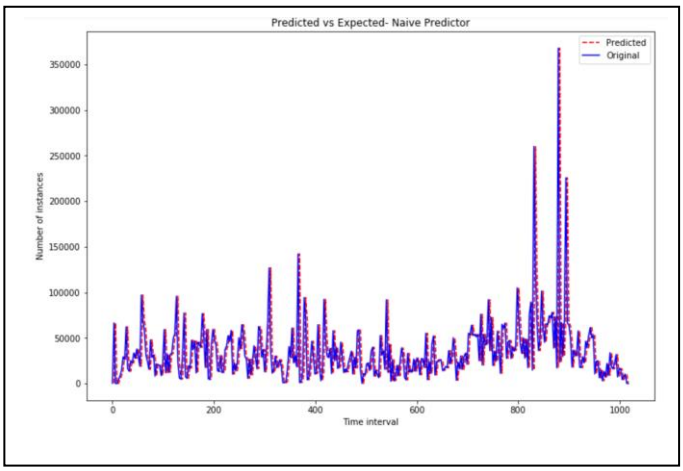

Figure 3. Predicted vs Expected Workload by Naïve Predictor

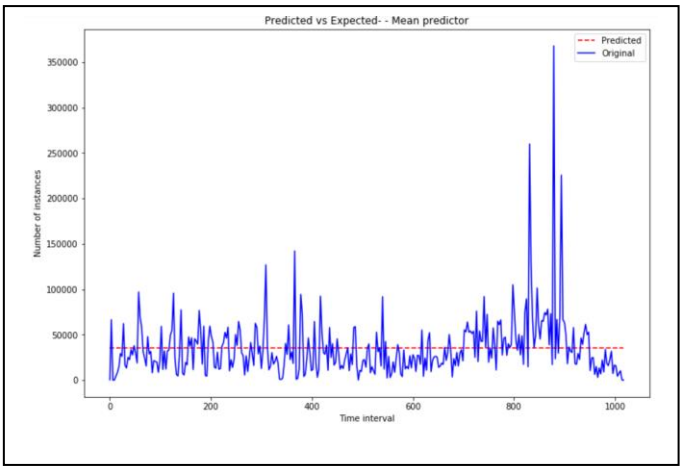

Figure 4. Predicted vs Expected Workload by Mean Based Predictor

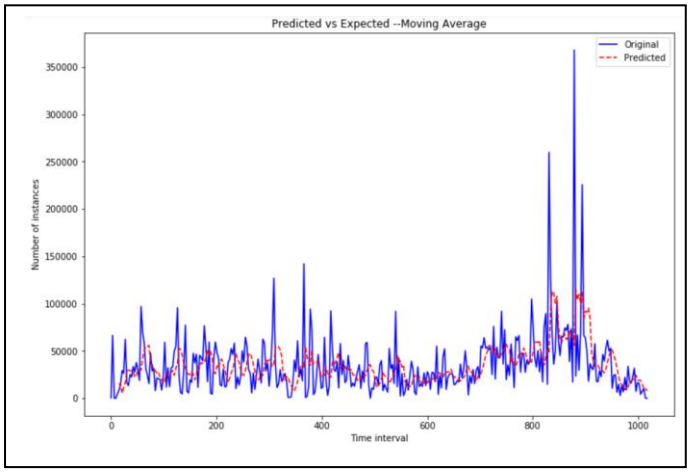

Figure 5. Predicted vs Expected Workload by Moving Average

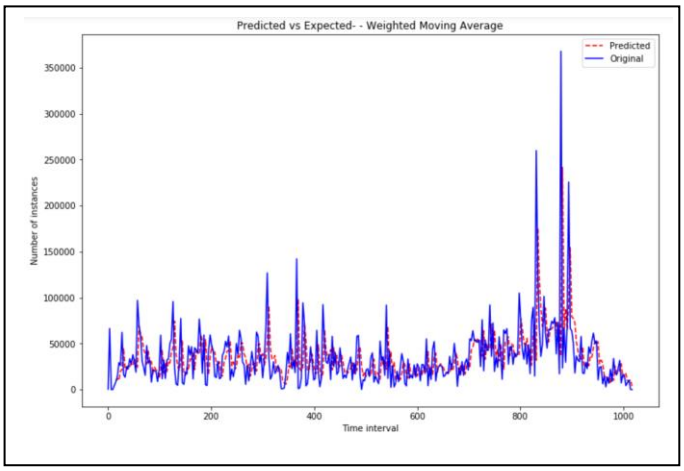

Figure 6. Predicted vs Expected Workload by Weighted Moving Average

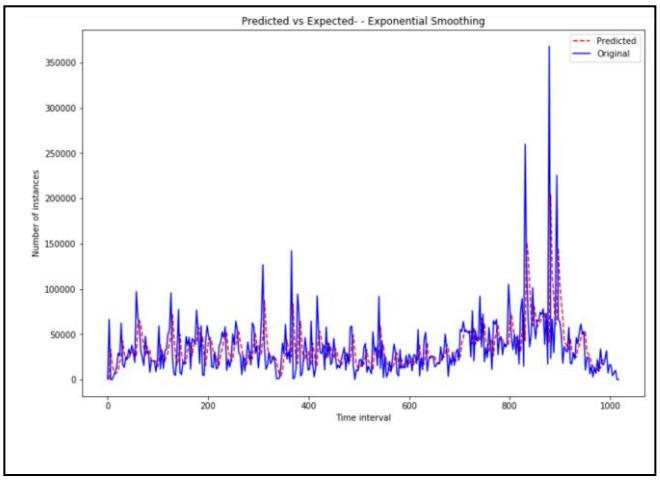

Figure 7. Predicted vs Expected Workload by Exponential Smoothing

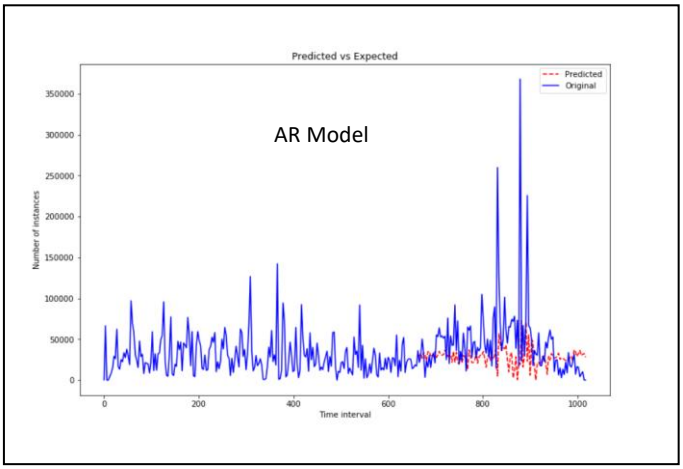

Figure 8. Predicted vs Expected Workload by AR Model

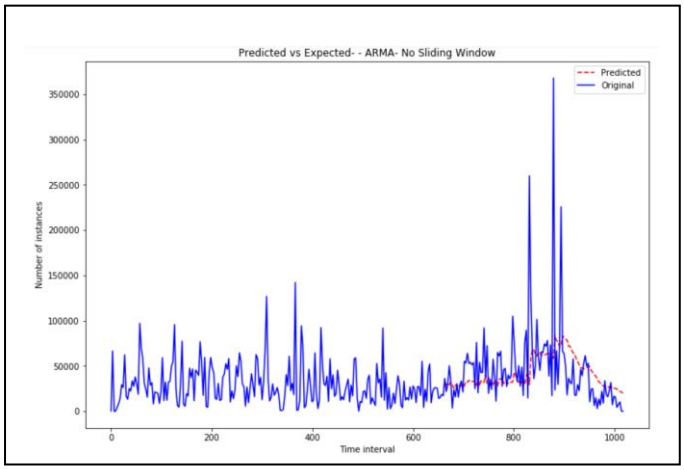

Figure 9. Predicted vs Expected Workload by ARMA Model

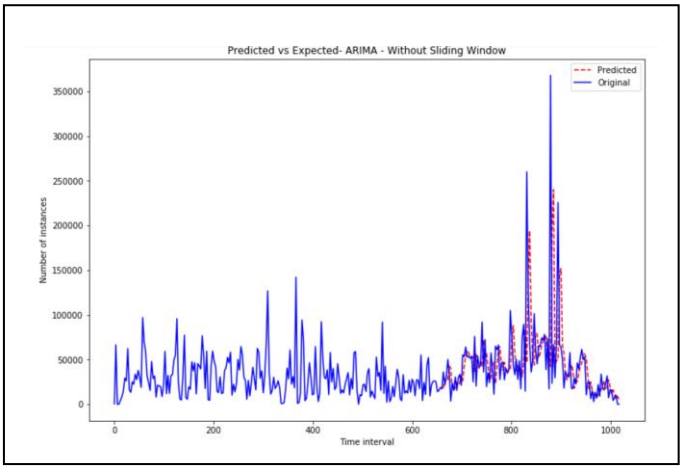

Figure 10. Predicted vs Expected Workload by ARIMA Model

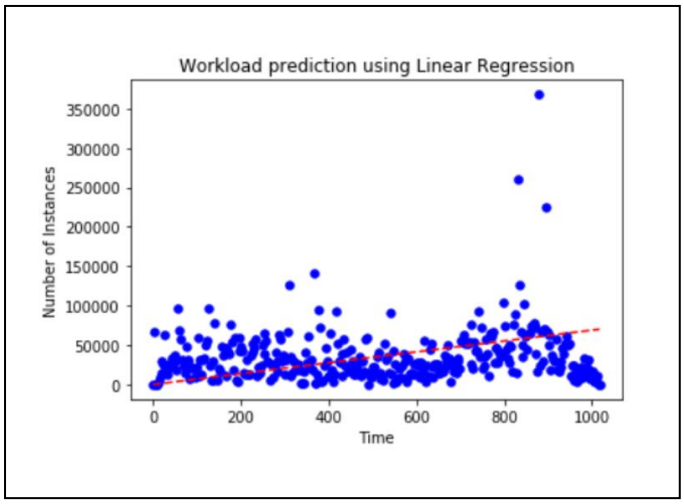

Figure 11. Workload Prediction by Linear Regression Model

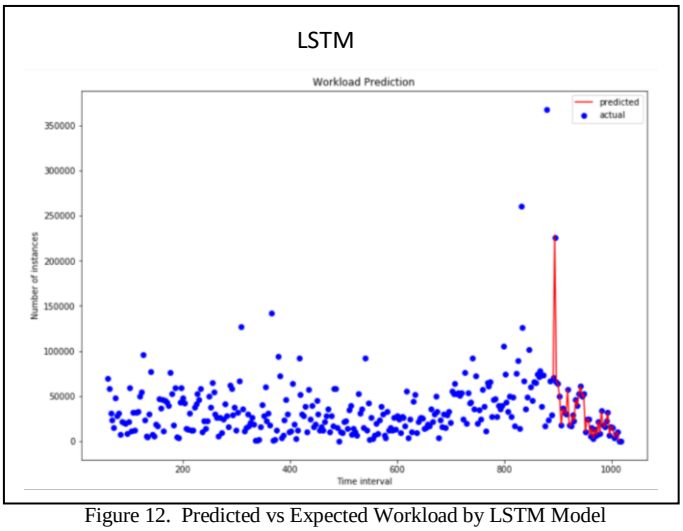

Figure 12. Predicted vs Expected Workload by LSTM Model

Table-II: Evaluation Results of Predictors (with sliding window)

MODEL	**RMSE**	**MAPE (%)**
Naïve Predictor	43k	20000
Mean Based Predictor	33k	31752
ES	32k	9354
MA	32k	5105
WMA	35k	1733
AR	122k	2673
ARMA	32k	3590
ARIMA	39k	1396
LSTM	25617	35.96

The graphs plotted between predicted vs expected workload for the models using sliding window approach. Figure-13 to Fig-16, shows the graphs plotted between predicted (red line) vs expected (blue line) workload.

For our project the baseline is “Naïve predictor” and “Mean based predictor” and main approach is “LSTM”. From the evaluation results, it shows LSTM performs better than the baseline models and also the other considered models. In case of models like AR, ARMA and ARIMA, sliding window approach improvised their prediction accuracy in comparison to the without sliding window approach. But for LSTM the accuracy without sliding window approach is pretty much better than with sliding window strategy.

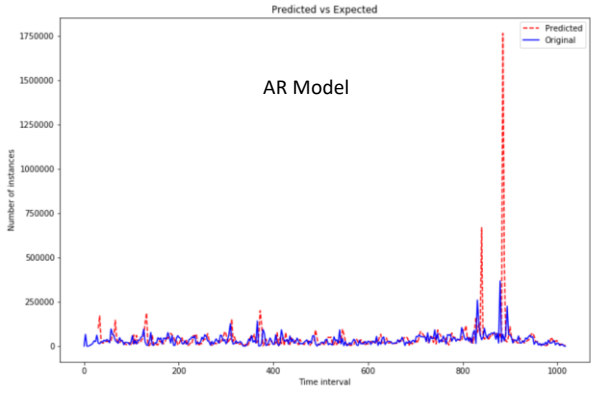

Figure 13. Predicted vs Expected Workload by AR Model

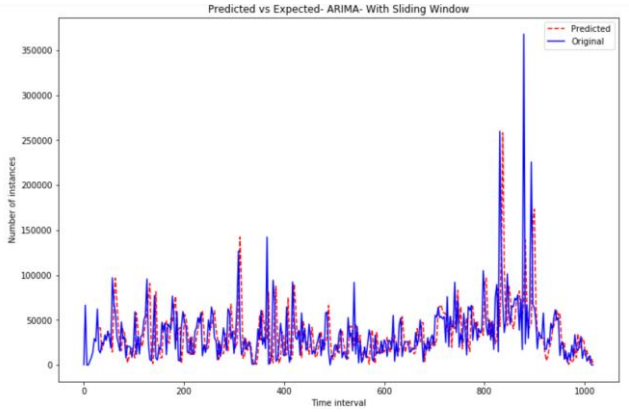

Figure 14. Predicted vs Expected Workload by ARIMA Model

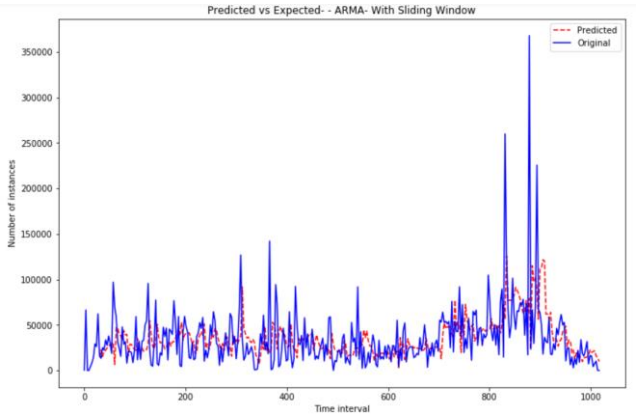

Figure 15. Predicted vs Expected Workload by ARMA Model

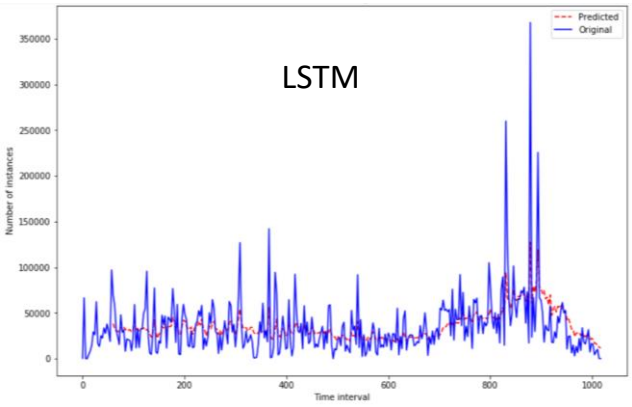

Figure 16. Predicted vs Expected Workload by LSTM Model

Table-II: Evaluation Results of Predictors (with sliding window)

MODEL	**Computational Time (seconds)**
Naïve Predictor	0.26
Mean Based Predictor	0.22
ES	0.24
MA	0.1
WMA	0.27
AR	0.01
AR (with sliding window)	0.67
ARMA	9
ARMA (with sliding window)	36.77
ARIMA	7.72
ARIMA (with sliding window)	37.9
Linear Regression	0.15
LSTM	0.16
LSTM (with sliding window)	900

By using the time.clock() function of python, the computation overhead is measured for all the models (both with and without sliding window). Table-III contains the computational time for all the predictor implemented in our project.

In our dataset the time interval between the next workload is 3 minutes (180 seconds) and it has total 340 number of timestamps. For this dataset, in case of without sliding window approach ARMA is the highest overhead predictor which takes 9 seconds and in case of with sliding window approach LSTM model which takes 900 seconds is the highest overhead predictor. LSTM took a longer amount of time in sliding window approach, whereas in without sliding window approach LSTM took only 0.16 seconds computational overhead. So, the computational overhead of LSTM (without sliding window) is less than our baseline predictors “Naïve predictor” and “Median based predictor”.

What is the importance of measuring computational overhead? If the computational time of a predictor is very high, then it may not be possible for the predictor to forecast the next number of workloads by the time the next workload arrives. So, computational time is an important aspect for selecting the suitable predictor. For our project LSTM predictor without sliding window has best accuracy and less computational overhead than the baseline models and rest of the other implemented models.

V. Discussion and Lesson Learned

Discussion: The dataset we used is of type random dataset. Out of all the 10 predictors implemented, the accuracy of LSTM is quite reasonable. Even though the sliding window approach improved the RMSE and MAPE for ARIMA and ARMA models, still it couldn't gain the accuracy up to the desired level. After going through the implementation and experiments on the workload predictors, we realize that the wide range of our dataset (minimum value is 1 and the maximum value is 367838) is one of the chief causes behind the high RMSE and MAPE values.

Learnings: This project is a learning curve for all of us in our team. We learned about different categories of dataset, like cyclic, seasonal, periodic, random, stationary, non-stationary. Our dataset is identified as a random dataset. Through the process of the project we honed our cognizance about python and its various useful library packages (scikit-learn machine learning package for implementing LSTM, statsmodels library package for implementing the time series-based approaches, pandas and matplotlib for working on csv files and plotting graphs, numpy etc.) We researched about different categories of prediction models and implemented few of them. Through the evaluation process the performance of these models were compared and which model suits best for our specific dataset was determined. For our dataset LSTM predicts the workload with highest accuracy which is around 96%. For evaluating our implemented models, we came across various evaluation metrics. In our project RMSE and MAPE have been used for performance evaluation of the predictors. Apart from technical knowledge we also learned about the teamwork and project management skills, which is the basic need for the success of any project.

VI. Conclusion

For predicting the future workload of Alibaba cluster dataset, we comprehensively evaluated 10 workload predictors using RMSE and MAPE as evaluation metrics. From the experiment we found out that LSTM predictor outperforms the other predictors both in case of without sliding window and with sliding window approach. Accuracy of LSTM for our dataset is found to be 93.6%. The proportion of train and test dataset depends upon the type of model and distribution of data. By the hit and trial approach, for obtaining the optimum accuracy, the train and test proportion for our dataset is obtained. In case of LSTM 80% as training data and 20% as testing data, whereas in case of other models (AR, ARMA, ARIMA) the training data is 65% and testing data is 35%. For the sliding window approach, the best window size depends upon the model type. For LSTM the best window size is 20 and for AR, ARMA and ARIMA models, window size 10 is the most suitable size.

Acknowledgment

We would like to thank Dr In Kee Kim for his valuable suggestions and guiding us throughout the project. His motivation inspired us to work on new problems and we successfully completed our project.

References

[1] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. "Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling." (IEEE Cloud 2016)

[2] Sadeka Islam, Jacky Keung, Kevin Lee, Anna Liu,"Empirical prediction models for adaptive resource provisioning in the cloud,Future Generation Computer Systems",Volume 28, Issue 1,2012

[3] https://www.cigniti.com/blog/black-friday-how-to-make-sure-your-website-is-ready/.

[4] https://machinelearningmastery.com/time-series-forecasting/

[5] https://otexts.org/fpp2/

[6] https://www.coursera.org/learn/practical-time-series-analysis/home/welcome

[7] https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks

[8] https://towardsdatascience.com

[9] https://www.business-science.io/timeseries-analysis/2018/04/18/keras-lstm-sunspots-time-series-prediction.html

[10] https://www.researchgate.net/post/How_to_set_up_LSTM_for_Time_Series_Forecasting

[11] A. Garcia-Pedrero and P. Gomez-Gil, "Time series forecasting using recurrent neural networks and wavelet reconstructed signals," *2010 20th International Conference on Electronics Communications and Computers (CONIELECOMP)*, Cholula, 2010, pp. 169-173.

[12] A. Tokgöz and G. Ünal, "A RNN based time series approach for forecasting turkish electricity load," *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, 2018, pp. 1-4.

[13] https://en.wikipedia.org/wiki/Mean_absolute_scaled_error

[14] https://en.wikipedia.org/wiki/Root-mean-square_deviation

[15] C. Lu, K. Ye, G. Xu, C. Xu and T. Bai, "Imbalance in the cloud: An analysis on Alibaba cluster trace," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2017, pp. 2884-2892.

[16] Abdelhadi Azzouni and Guy Pujolle, "A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction" in journal: CoRR, in the year 2017.

[17] A. Geetha and G. M. Nasira, "Time series modeling and forecasting: Tropical cyclone prediction using ARIMA model," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 3080-3086.

[18] Adebiyi, Ayodele & Adewumi, Aderemi & Ayo, Charles. (2014). Stock price prediction using the ARIMA model. Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014. 10.1109/UKSim.2014.67