Predicting Amazon Spot Prices

Abhishek Chatrath
Computer Science Dept.
University of Georgia
Athens, Georgia
ac06389@uga.edu

ABSTRACT

Amazon spot instances provide preemptable computing capacity using an unused EC2 instance that is available that is available for less than On-Demand price. Spot instances are charged at the current spot price: a mutable price which changes based on the supply and demand of these instances. Because the spot instances are available at steep discounts (usually up to 90%), it helps lowering the use of Amazon EC2 cost significantly. However, these instances are highly unpredictable and can be rescinded by Amazon with as little as two minutes warning. Given its significant discounts – there has been a significant interest in the scientific cloud computing community to move towards spot instances for workload and that are fault-tolerant and flexible. Thus, the need of predicting the accurate pricing in the future is important for a cost-effective use of this instance. We will explore the current models being used for the same and try to implement other models to compare with the existing models.

1 INTRODUCTION

Amazon EC2 service allows you to rent virtual instances with the specifications you want and at the time one needs. These On-demand instances are available with a few mouse-clicks and usually does not have any wait-time; Amazon maintains an EC2 pool of spare computational resources to allocate. Since these spare resources were not used completely by Amazon. In ordered to use these spare resources, Amazon decided to provide these on-demand at significantly lower prices as `Spot Instances`. Spot instances are not dedicated instances i.e. they are preemptable. Due to their preemptable nature, the offered prices are very low as compared to the On-demand EC2 instance. The low price however, comes at the cost of reliability – the instance can be terminated with, as little as, a two minutes notice.

Amazon organizes spot market for spot instances for each instance type, availability zone and region. Prior to November 2017, the users were required to bid the maximum value they were willing to pay for a particular type of instance, and the spot price were periodically recalculated as per the current bids made since the allocation. If the spot price was equal or higher than the price it was allotted on, the allocated instance was revoked. Thus, the spot price was directly related to revocation and therefore users aimed to compute a bid price that was above the spot price (to avoid revocation) but not significantly so (to reduce financial risk). In November 2017, the spot market was updated, and the bidding process was removed. Now, Amazon sets the spot price instead based on the current trends of supply and demand. Amazon can still revoke spot instances at their discretion and the requirements have changed for revocation. In addition, to spot price increasing to more than the last specified price, conditions like Capacity – if not enough unused EC2 instances available to meet the demand for spot instances and constraints – user defined constraints not met, also lead to interruption and revocation of a spot instance.

Amazon provides up to three months of price histories for each instance type, availability zone, and region. This information provides a basis from which one can try to forecast prices and derive patterns.

Researchers have tried variety of methods to accurately predict the pricing of spot instances that include parametric models, Markov models and cost minimization approaches. Further, there are several models available for price prediction in other fields, such as, the stock market, models like the autoregressive integrated moving average (ARIMA), generalized autoregressive conditional heteroskedasticity (GARCH), neural networks, and many others have been used. While many of these methods can accurately model spot prices, but they are not universally accurate. Perhaps due to the fact that spot prices are not driven solely by demand but rather by Amazon's introduction of hidden externalities that affect pricing. In this project, we will implement ARIMA and LSTM-dense neural network architecture and other models, if time permits.

2 RELATED WORK

A wide range of predictive models have been explored to predict spot prices. For example, researchers have used parametric models based on exponentials used using Expectation-maximization ; Constrained Markov Decision Processes to minimize the expected cost of an instance. Closely related prediction models have been developed to support application migration and checkpointing to avoid service downtime.

A novel framework is presented in [2] integrating empirical mode decomposition and support vector regression for interval forecasting of electricity demand. Bagged Neural Networks (BNN) are used in [3, 4] for prediction. The authors show that BNN re-duce forecasting errors, show higher prediction accuracy and have less computational time.

Several previous works focus on Amazon EC2 spot price modeling and prediction. Authors in [5] make use of simple moving average, weighted moving average and exponential moving average methods to predict the next hour spot prices using estimations. Reverse engineering on how prices are set is performed in [6] to construct a model that generates prices consistent with existing price traces. A probabilistic model is presented by authors in [7] to enable user to optimize monetary costs, performance and reliability. The authors employ normal approximation to model the spot price distribution which focuses on pre-computation of a fixed bid.

Authors in [8] assume spot prices are normally distributed and focus on achieving availability guarantee with spot in-stances. The authors use a quantile function of the ap-proximate normal distribution to predict future spot prices when the auto-correlation of current and past spot price is weak. When the autocorrelation is strong, a simple linear regression prediction

model is adopted. Ac-cording to Javadi et al. [9] Gaussians distribution shows better fit. Analysis of predictability of the time-varying spot instance prices in Amazon EC2 is performed in [10].

Neural network is a field that is still being explored. Because of limitations of resources, Neural Network was not really used or applied for major researches. Neural Networks was applied for identifying cancerous tumors[11], listening to and generating raw audio [12] and even beating Go without any prior knowledge of the game [13] and transferring what was learnt to games like chess and Japanese chess [14]. With advent of readily available GPU's, Neural network has emerged to be an effective deep learning model for image processing and other predictive fields.

A predictive model based on artificial neural networks is presented in [15] for short-term prediction of future spot prices of m1.linux, m1.windows. The authors predict spot price for the following time interval for approximately every 1.3 hours. Our project is based on [1] where the authors introduced model to predict the spot price for a particular instance type, availability zone, and region for a specific time in the future based on LSTM.

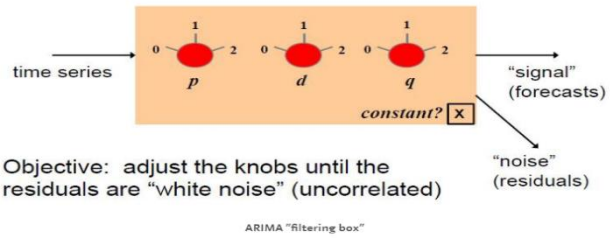
3 APPROACH

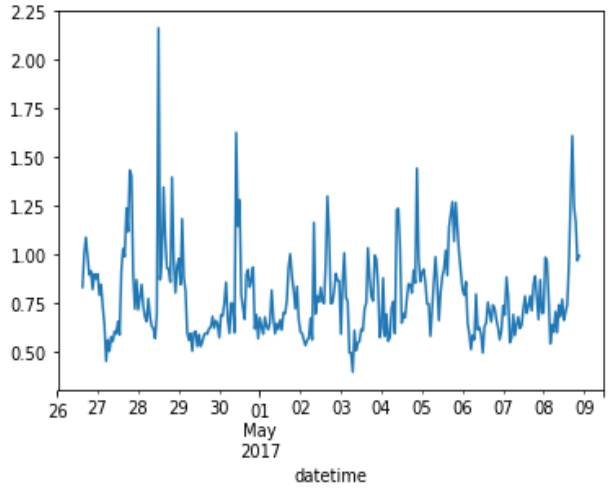
For our approach, we decided to use the data available on Kaggle. The data has been taken for two weeks and is detailed down to the millisecond for three server type – Windows, Linux and Unix. Since the data is a time-series (a series of values of a quantity obtained at successive times, often with equal intervals between them). The statistical model ARIMA[16] was used as a baseline for project and we implemented LSTM (Long short-termed Memory) networks to improve the accuracy that is provided by standard statistical models.

3.1 ARIMA

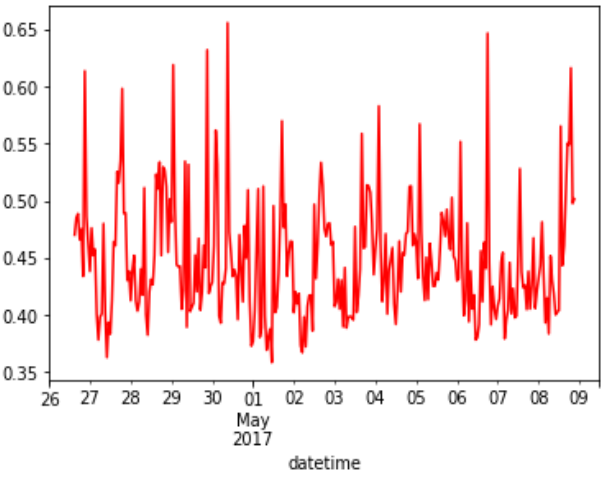
ARIMA stands for auto-regressive integrated moving average. for predicting future points in the series), in such a way that :a pattern of growth/decline in the data is accounted for (auto-regressive), the rate of change of the growth/decline in the data is accounted for (integrated), noise between consecutive time points is accounted for (moving average).

Arima is a generalized random walk model which is fine-tuned to eliminate all residual autocorrelation. It is a generalized exponential smoothing model that can incorporate long-term trends and seasonality. A random variable of this form can be viewed (as usual) as a combination of signal and noise, and the signal (if one is apparent) could be a pattern of fast or slow mean reversion, or sinusoidal oscillation, or rapid alternation in sign, and it could also have a seasonal component. An ARIMA model can be viewed as a "filter"[17] that tries to separate the signal from the noise, and the signal is then extrapolated into the future to obtain forecasts. To implement ARIMA, the dataset needs to be checked for trends.

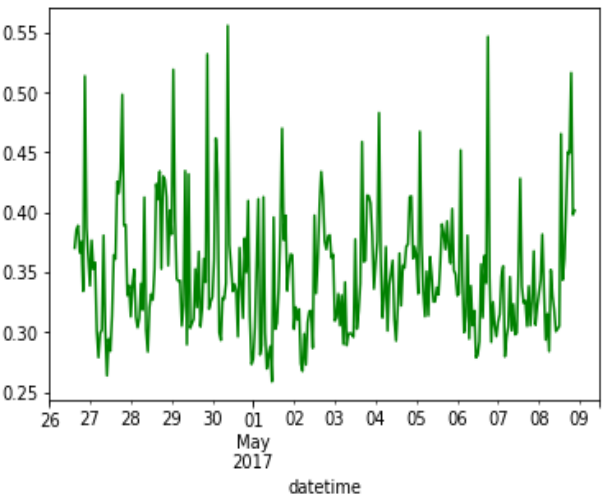




(I)

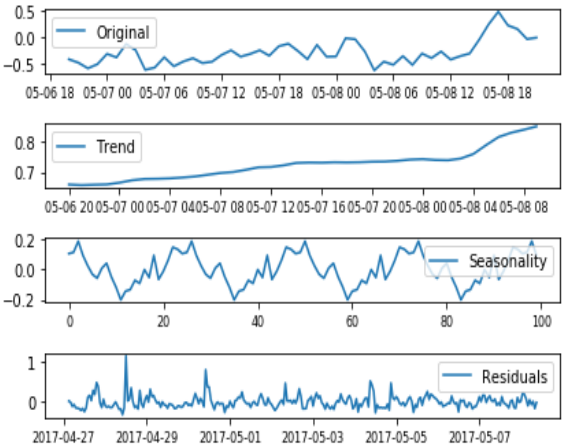


(II)

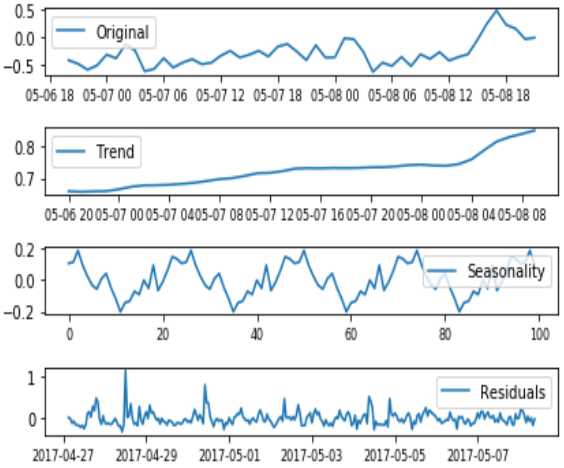


(III)

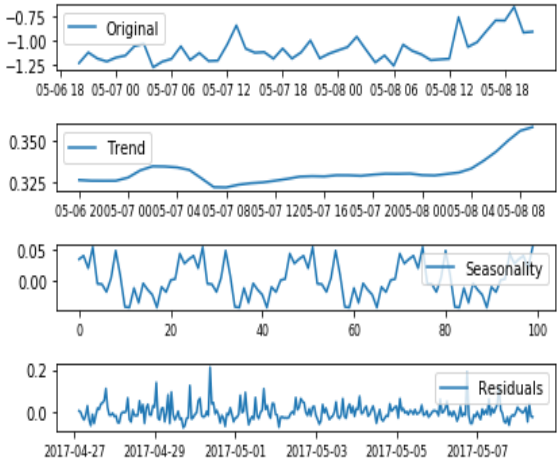
Figure 1 shows the hourly subsample plot for each server types. (I) Windows (II) Linux (III) Unix



(I)



(II)



(III)

Figure 2 : Seasonal decomposition of each server type. (I) Windows (II) Linux (III) Unix

Since ARIMA can only be applied to stationary data, the trends and seasonality have to be removed to make the time series stationary. A random variable that is a time series is stationary if its statistical properties are all constant over time. A stationary series has no trend, its variations around its mean have a constant amplitude, and it wiggles in a consistent fashion, i.e., its short-term random time patterns always look the same in a statistical sense.

To check for trends first we resampled the data on an hourly basis as shown in Fig1. Since the trend was not clear enough, we performed the Dickey-Fuller test to check the stationarity and decompose the data further into trend and seasonality as shown in Fig 2.

ARIMA model for forecasting univariate time series data can be parameterized through three attributes, (p, d, q), where p is the number of autoregressive terms, d the number of differencing, and q the order of the moving average. The values of p and q were determined by the Partial Autocorrelation function (PACF) and Autocorrelation function(ACF) respectively.

Once the data is stationary and we have the p and q parameters are decided, the ARIMA model is ran and fit over the data. ARIMA can be used in two ways for forecasting – (1) In-Sample (2) Out-of-Sample. Since the amount of data after resampling reduced quite a bit, Out-of-Sample would produce a problem of overfitting which is general problem with the ARIMA models. Hence, we used In-Sample forecasting to measure the predicting capabilities of our model.

The results with root mean square error can be seen in Fig 4. Blue is the original price and red represents our prediction.

3.2 LSTM

Given the relative complexity of this task and the potential for many latent features to influence the spot price (e.g., seasonality) recurrent neural network (RNN) can provide better accuracy. RNNs are becoming increasingly popular as a means of modeling time series data such as meteorological data and stock prices. For our task, we use long/short-term memory (LSTM) as the main building blocks of the RNN. LSTMs are able to identify and remember latent features over an unspecified number of time periods, making them a versatile tool in time series prediction.

Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies. These algorithms take time and sequence into account, they have a temporal dimension.

LSTM networks manage to keep contextual information of inputs by integrating a *loop* that allows information to flow from one step to the next. These loops make recurrent neural networks seem magical. But if we think about it for a second, as you are reading this, you are understanding each word based on your understanding of the previous words. You don't throw everything away and start thinking from scratch at each word. Similarly, LSTM predictions are always conditioned by the past experience

of the network's inputs. Figure 3 shows an overview of LSTM unfolded.

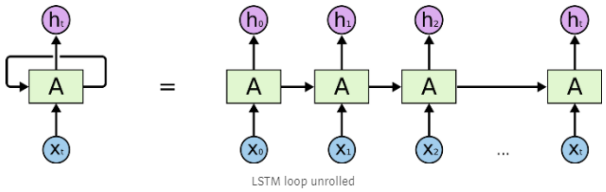


Fig 3 : LSTM unfolded [18]

LSTMs help preserve the error that can be backpropagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely. This is one of the central challenges to machine learning and AI, since algorithms are frequently confronted by environments where reward signals are sparse and delayed, such as life itself.

LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computer's memory. The cell makes decisions about what to store, and when to allow reads, writes and erasures, via gates that open and close. Unlike the digital storage on computers, however, these gates are analog, implemented with element-wise multiplication by sigmoids, which are all in the range of 0-1. Analog has the advantage over digital of being differentiable, and therefore suitable for backpropagation.

3.2.1 Network Structure

Our Network architecture is based on [1]. The final RNN shown in Figure 5, consists of 2 LSTM layers and one Dense layer. There is a Dropout layer after each of the LSTM layers to prevent overfitting. Each LSTM layer consists of 32 units.

Like other RNNs, the LSTM units can be assembled into layers of the neural network that are able to accept data sequentially(i.e., the data is processed through each unit such that the previous data point is able to be related to the last). The advantage held by LSTM over traditional RNNs is that it possesses three internal "gates" that allow it to explicitly forward or forget data ,allowing it to relate data from point to point and "forget" earlier relations over a certain period, if these earlier relations turn out to be irrelevant. This functionality provides a mechanism to use information from surrounding inputs to affect a given input while remaining unaffected by inputs more distant.

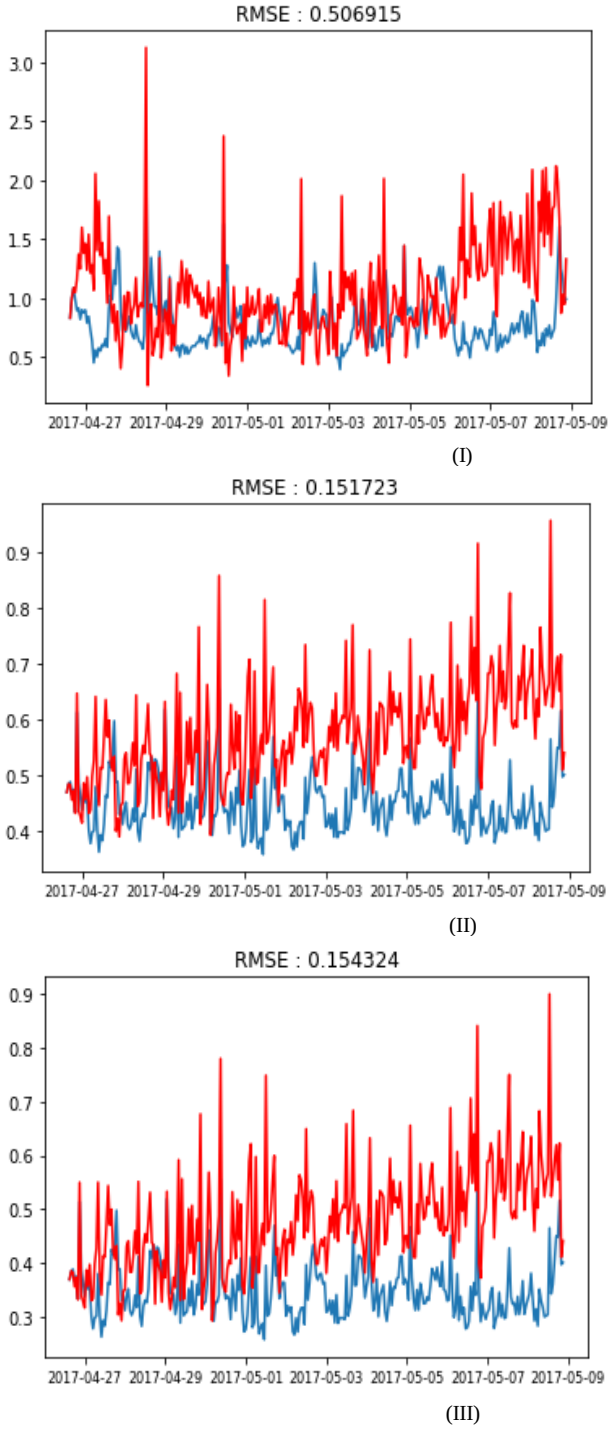


Figure 4 : ARIMA in-sample predictions of each server type - (I) Windows (II) Linux (III) Unix

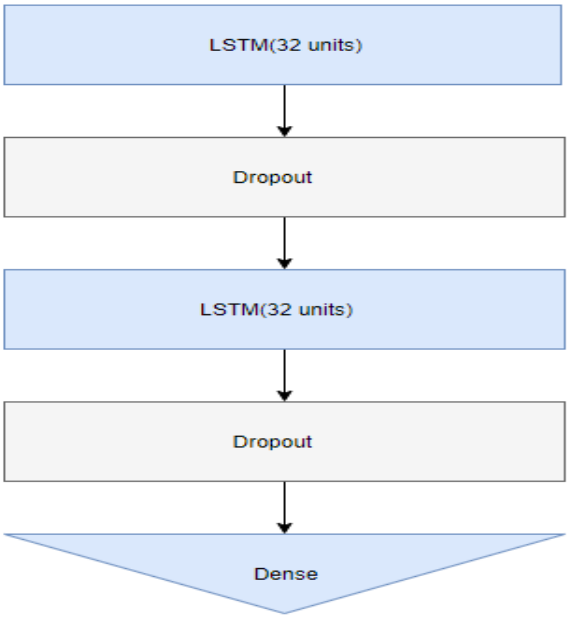


Figure 5 : LSTM architecture consists of 2 LSTM layers and 1 Dense Layer. Dropout layers were added to prevent overfitting

The data was resampled after every 15 minutes with the mean of price for each server types giving us 1178 entries for each server type. Then each sample was divided into training and test data in 80:20 ratio respectively.

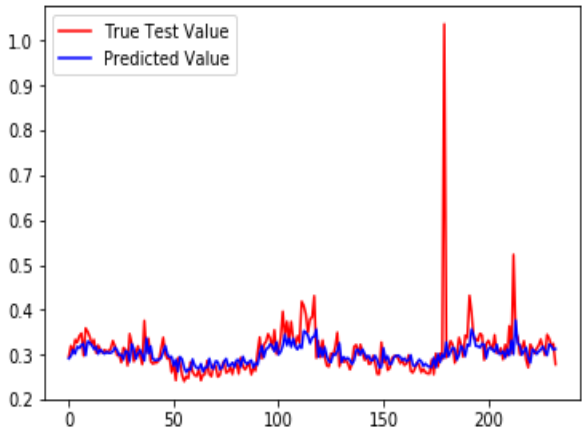
To train our neural network ,we used the ADAM optimization algorithm with Nesterov momentum(NADAM) and Mean Square Error as our loss function.

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 1, 32)	4352
dropout_1 (Dropout)	(None, 1, 32)	0
lstm_2 (LSTM)	(None, 1, 32)	8320
dropout_2 (Dropout)	(None, 1, 32)	0
flatten_1 (Flatten)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

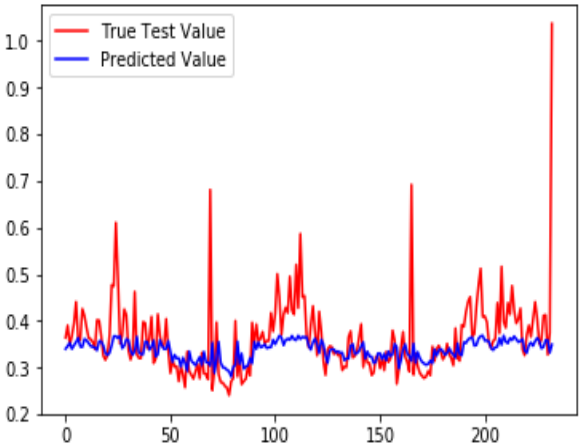
Total params: 12,705
Trainable params: 12,705
Non-trainable params: 0

Figure 6: Summary of our LSTM model

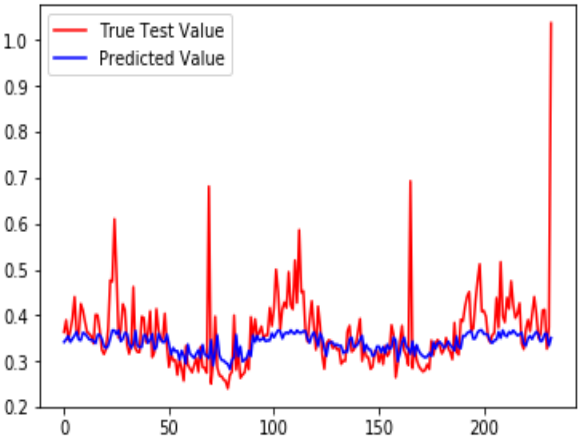
Fig. 7 presents our predictions over the test dataset for each server type.



(I)



(II)



(III)

Figure 7 : Predictions with LSTM for each server type – (I) Windows (II) Linux (III) Unix

4 EVALUATION

For evaluation of our models we are using Root-Mean-Square-Error(RMSE). We are using Mean-Squared Error(MSE) to compare our models**.** RMSE and MSE can be calculated as below:

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y} - y)^2}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y} - y)^2}{n}}$$

where $\hat{y}$ is the predicted price and y is the true price.

Figure 8 shows the MSE plot for ARIMA and LSTM. The points where the plot hits zero means that the prediction was accurate at those points.

The table below shows a comparison for MSE and RMSE of each server type in both the models

MODEL	WINDOWS	LINUX	UNIX
ARIMA-RMSE	0.5069	0.1517	0.1543
LSTM - RMSE	0.0591	0.0800	0.0768
ARIMA-MSE	0.3955	0.13008	0.1312
LSTM-MSE	0.0034	0.0064	0.0456

Table 1 : Comparison of ARIMA and LSTM

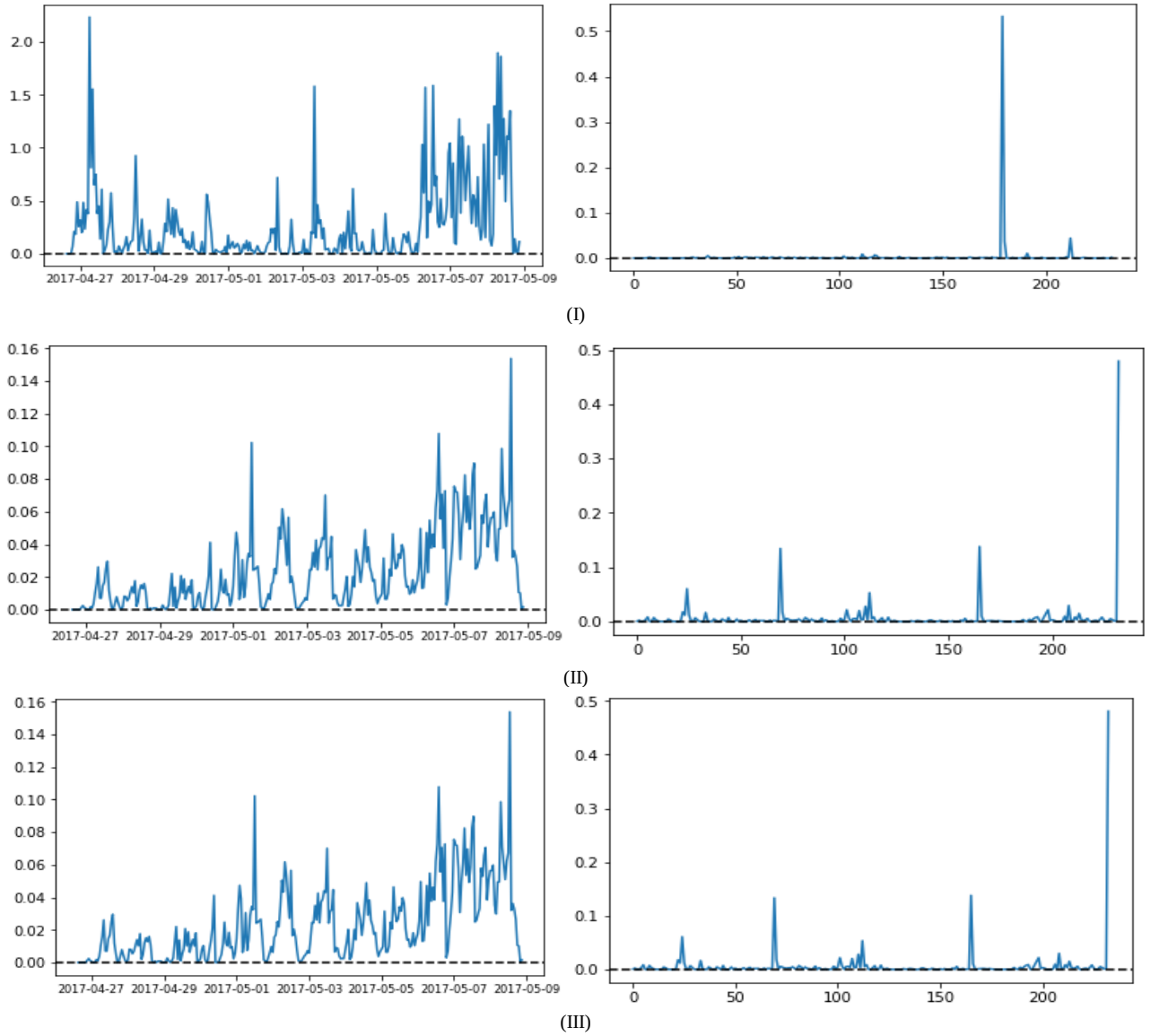


Figure 8 : MSE for ARIMA(LEFT) and LSTM(RIGHT) for each server type –(I) Windows (II) Linux (III) Unix

5 Conclusion

ARIMA as a general model tends to have issues for overfitting. This might be one of the reasons why our results show high RMSE. Another issue that was observed was the closeness of time periods. The closer resample we took, ARIMA stopped understanding them. That is one reason why we had to choose a resample time period of 1 hour which is where the model understood the data. ARIMA is statistical model i.e. it has foundations based on assumptions and observations of humans and when it comes to big data, some patterns tend to be missed. That can be another reason for low accuracy of the ARIMA model with our data.

LSTM is a deep learning model which has its foundations on the ability of a machine to understand and identify correlations and patterns. Thus, LSTM doesn't miss most of the patterns and increase the accuracy of our predictions. Another benefit with LSTM is that it can understand the trends and seasonality of the time-series and doesn't need for the data be stationary. The accuracy of the model clearly proves the aforementioned conclusion. Like other neural network models, LSTM tends to overfit the data when you don't have enough data to run the model on. That was the reason we chose resample of 15 minutes to ensure there is no overfitting and our dataset also remains close to the sample used for ARIMA.

6 Summary

Accurate price prediction of spot instances can reduce the cost to company of using the spot instances and can also be used down the road in other areas of research. Using an LSTM-based approach has given us a very high accuracy when compared to our base model ARIMA which suggests using RNN to predict prices is much better than any other statistical model approach.

One thing that we did not expect to see was the seasonality trend in the dataset even though it is based on the bid-system.

The current Amazon market has changed from prices based on bids to prices based on demand and availability of resources, so our results cannot be used to predict the current market, but it would be interesting to see if our models could actually reveal the decision making patterns for spot prices that is being used currently by Amazon.

7 Acknowledgement

This project was successfully completed under guidance of Dr. In Kee Kim and Dr Jaewoo Lee, Computer Science, UGA

8. References

[1] Mat Baughman, Christian Haas, Rich Wolski, Ian Foster, Kyle Chard, 2018 Predicting Amazon Spot Prices with LSTM Networks. https://dl.acm.org/citation.cfm?id=3217881

[2] T. Xiong, Y. Bao, and Z. Hu, "Interval forecasting of electricity demand: A novel bivariate EMD-based support vector regression modeling framework," International Journal of Electrical Power & Energy Systems, vol. 63, pp. 353–362, Dec. 2014.

[3] S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, and B. Venkatesh, "Improved short-term load forecasting using bagged neural networks," Electric Power Systems Research, vol. 125, pp. 109–115, Aug. 2015.

[4] R. Ghani, "Price prediction and insurance for online auctions," Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD ″05, 2005.

[5] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios," Concurrency and Computation: Practice and Experience, vol. 27, no. 9, pp. 2260–2277, Dec. 2012.

[6] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," ACM Transactions on Economics and Computation, vol. 1, no. 3, pp. 1–20, Sep. 2013.

[7] A.Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Aug. 2010.

[8] M. Mazzucco and M. Dumas, "Achieving Performance and Availability Guarantees with Spot Instances," 2011 IEEE International Conference on High Performance Computing and Communications, Sep. 2011.

[9] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," Future Generation Computer Systems, vol. 29, no. 4, pp. 988–999, Jun. 2013.

[10] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, "Optimal Resource Rental Planning for Elastic Applications in Cloud Market," 2012 IEEE 26th International Parallel and Distributed Processing Symposium, pp. 808–819,May 2012.

[11] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. 2017. Brain tumor segmentation with Deep Neural Networks. Medical Image Analysis 35 (2017), 18–31

[12] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol ¨ Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. CoRR abs/1609.03499 (2016). arXiv:1609.03499 hŠp://arxiv.org/abs/1609.03499

[13] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Landrent Sifre, George van den Driessche, Thore Graepel, andDemis Hassabis. 2017. Mastering the game of Go without human knowledge. Nature 550 (2017), 354–359.

[14] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. CoRR abs/1712.01815 (2017). http://arxiv.org/abs/1712.01815

[15] R. M. Wallace, V. Turchenko, M. Sheikhalishahi, I. Turchenko, V. Shults, J. L. Vazquez-Poletti, and L. Grandinetti, "Applications of neural-based spot market prediction for cloud computing," 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), Sep. 2013.

[16] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo. 2003. ARIMA models to predict next-day electricity prices. IEEE Transactions on Power Systems 18, 3 (Aug 2003), 1014–1020.

hŠps://doi.org/10.1109/TPWRS.2002.804943
[17] https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8
[18] https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd
[19] Predicting Spot Exchange Rates with ARIMA in Python - https://dataplatform.cloud.ibm.com/exchange/public/entryview/815137c868b916821dec777bdc23013c