Amazon EC2 Spot Price Prediction Using Regression Random Forests

Veena Khandelwal, Anand Kishore Chaturvedi, and Chandra Prakash Gupta

Abstract—Spot instances were introduced by Amazon EC2 in December 2009 to sell its spare capacity through auction based market mechanism. Despite its extremely low prices, cloud spot market has low utilization. Spot pricing being dynamic, spot instances are prone to out-of bid failure. Bidding complexity is another reason why users today still fear using spot instances. This work aims to present Regression Random Forests (RRFs) model to predict one-week-ahead and one-day-ahead spot prices. The prediction would assist cloud users to plan in advance when to acquire spot instances, estimate execution costs, and also assist them in bid decision making to minimize execution costs and out-of-bid failure probability. Simulations with 12 months real Amazon EC2 spot history traces to forecast future spot prices show the effectiveness of the proposed technique. Comparison of RRFs based spot price forecasts with existing non-parametric machine learning models reveal that RRFs based forecast accuracy outperforms other models. We measure predictive accuracy using MAPE, MCPE, OOB Error and speed. Evaluation results show that $MAPE <= 10\%$ for 66 to 92 percent and $MCPE <= 15\%$ for 35 to 81 percent of one-day-ahead predictions with prediction time less than one second. $MAPE <= 15\%$ for 71 to 96 percent of one-week-ahead predictions.

Index Terms—Amazon EC2, compute instances, one-day-ahead prediction, one-week-ahead prediction, regression random forests, spot instances, spot price prediction

1 Introduction and Motivation

The on-demand scalability characteristic of cloud computing forces cloud service providers to over-estimate their resources to meet peak load demand of its customers which happens at different time periods and may not overlap. Due to over-estimation, a large number of cloud resources are idle during off peak hours. Cloud providers also face the problem of allocating resources, keeping in view user's different job requirements and data center capacity. Different types of users, multiple types of requirements further alleviate the resource allocation problem. Also, demand for cloud resources fluctuate due to today's usage based pricing plans. In order to manage these demand fluctuations more flexible pricing plans are required to sell resources according to real time market demand. Spot pricing was introduced by Amazon EC2 in December 2009 to minimize operational cost, combat under utilization of its resources and make more profit. Similar to on-demand instances, spot instances offer several instance types comprising different combinations of CPU, memory, storage and networking capacity. Amazon Web Service (AWS) is not the only participant in the spot instance realm. Google Compute Engine launched its preemptible Virtual Machines on September 8, 2015 designed for such type of workloads that can be delayed and are fault tolerant at the same time. Users can bid for spot instances (SIs) where prices are charged at lowest bid price, whereas, pricing on Google Preemptible VMs is fixed at per hour rate. The distinguishing feature of Amazon Elastic Compute Cloud (EC2) spot instance is its dynamic pricing.

From customer's perspective, spot instances offer prospects of low cost utility computing at a risk of out-of-bid failure at any time by Amazon EC2. Spot instance reliability depends on the market price and user's maximum bid (limited by their hourly budget).

Spot prices vary dynamically with real-time based on demand (user's bid) and supply (resource availability) for spot instance capacity in the data centers across the globe. Users bid for spot instances and control the balance of reliability versus monetary cost. The price for spot instances sometimes can be as low as one eighth of the price of on-demand instances. On the other hand, it is also not uncommon that spot prices surpass on-demand prices in cloud data centers. When the demand is low, spot prices are low because less numbers of users are bidding for the same instance. Therefore, a bidder's probability of incurring less monetary cost is higher. On the other hand, when the demand is high, users are willing to pay high to get access and hence spot prices increase.

Spot pricing in particular is a pricing model targeted for divisible computing jobs that can shift the time of processing to when the computing resources are available at low cost [1]. The primary requirement is that the applications

- *V. Khandelwal is with the Department of Computer Science & Engineering, Rajasthan Technical University, Kota, RJ 324009, India. E-mail: vn.khandelwal@gmail.com.*
- *A. K. Chaturvedi is with MLV Textile and Engineering College, Bhilwara, RJ 311001, India and with Rajasthan Technical University, Kota, RJ 324009, India. E-mail: chaturvedi101@gmail.com.*
- *C.P. Gupta is with the Department of Computer Science & Engineering-Rajasthan Technical University, Kota, RJ 324009, India. E-mail: guptacp2@rediffmail.com.*

Manuscript received 5 Jan. 2017; revised 28 Sept. 2017; accepted 1 Dec. 2017. Date of publication 6 Dec. 2017; date of current version 11 Mar. 2020. (Corresponding author: Veena Khandelwal.) Recommended for acceptance by J. Le. Digital Object Identifier no. 10.1109/TCC.2017.2780159

TABLE 1
Applications and Their Infrastructure Requirements

Application Type	Memory Requirement	Cores Requirement
Audio Encoding	16 MB	Dual Core
Video encoding	64 MB	Dual Core
Web Crawling	8 GB	8 Cores
Protein Analysis	2 GB	Dual Core
Computer Aided drug design	8 GB	Dual Core
Virtual reality	64 GB	16 cores
Genome Analysis	64 GB	62 cores
Bioinformatics application	64 GB	8 cores
Hadoop Application	24 GB	8 cores

must be time flexible, do not have a steep completion deadline and should be interrupt tolerant. Spot instances are also required for executing certain sudden tasks which do not need reserved instances.

The ability to predict spot price lends itself to a variety of applications. Just a few examples of applications suitable to be executed on spot instances are: geospatial analysis, image and video processing, scientific research and computing, data processing, financial modeling and analysis, big data analytics, testing, web crawling and large scale simulations. Table 1 lists various real life applications and their resource requirements.

The infrastructure requirements of most of the applications in terms of memory and number of cores are high which can be met by compute optimized instances easily. We therefore attempt to predict spot prices of Amazon EC2 compute optimized instances only in this work.

Auction-based spot pricing presents challenges for cloud computing markets. Spot price prediction is challenging because there is very little information available on the underlying operating mechanism of the spot market by Amazon EC2. Spot prices change at unequal intervals which eliminate the use of standard time series forecasting methods. Furthermore, bid once made cannot be changed until the instance is interrupted. Forecasts sufficiently ahead in time equal to job length is required in reality to bid spot instances. Job execution reliability and cost savings at a given bid price greatly depend on job length [16].

Spot prices vary considerably across Amazon EC2 regions on different weekdays and during different hours of a day. There is a great surge of mechanisms for accurately predicting spot prices across various regions sufficiently far in advance to plan future job executions. In order to achieve maximum cost savings and reliability of spot instances for fault tolerant and time flexible jobs execution, one should be able to accurately predict one-week-ahead and one-day-ahead spot prices across Amazon EC2 regions sufficiently fast.

Research for mechanisms that enable more informed bids in a highly dynamic cloud computing market justifies the relevance of the Spot Price Prediction problem. Accurate prediction of spot prices can help organizations moderate the risk of losing spot instances due to out-of-bid failure thus increasing the reliability of spot instances. It will also help users not to over-bid on an instance thus reduce the risk of increased execution cost [2]. Developing mechanisms to predict forthcoming bid prices would also be helpful to reduce rollback overheads during task execution.

Dynamic pricing model is not followed by any other cloud service provider. However, we do not see any issues in using our approach for forecasting spot prices of service providers who follow dynamic pricing policy.

The objective of this work is to present and evaluate a predictive model for spot price prediction that can predict future prices with increased accuracy and speed, minimize forecasting errors and predict spot prices sufficiently far in advance to assist cloud spot users in bid decision making process with increased reliability. We compare prediction accuracy of the state of the art non-parametric supervised machine learning algorithms with Regression Random Forests (RRFs) model.

The major contributions of this paper are:

1. An analysis of the length of time epoch durations when spot price is less than on-demand price to raise users confidence level in opting for spot instances.
2. Identify Feature Importance in spot pricing using RRFs model to aid bid decision making process regarding when (weekday, time) to acquire spot instances with increased reliability and minimum cost.
3. Propose a new metric called Mean Consequential Percentage Error (MCPE) adapted from Mean Consequential Error (used for classification problems) in order to measure prediction accuracy.
4. Spot price forecasting. We resort to machine learning based ensemble method namely RRFs for one-week-ahead and one-day-ahead spot price prediction. The approach focuses on both prediction accuracy and speed.

Significant implications of RRFs include increased robustness, accuracy at a fraction of the training time, low implementation effort in comparison to single models, stability to outliers and less prone to over-fitting. To the best of our knowledge, this is the first use of RRFs for spot price prediction. The novelty of our approach is its ability to predict one-day-ahead as well as one-week-ahead spot prices in less than 1 second.

The paper is organized as follows. After this introduction, Section 2 presents related work. Section 3 presents analysis of one year compute optimized spot instance price history traces in order to raise user's confidence in opting spot instances. Section 4 compares various non- parametric machine learning approaches for spot price prediction. Section 5 discusses various techniques to reduce prediction bias and variance in RRFs, RRFs prediction algorithm, parameters and cross validation method (out-of-bag error) used in RRFs. Section 6 presents various RRFs model parameters tuned to fit the model to spot instance pricing dataset in order to increase model's prediction accuracy. Section 7 presents experimental spot price prediction results and analyse the RRFs prediction model accuracy and speed. Section 8 presents conclusion.

2 Related Work

Several works focus on using machine learning and ensemble methods for solving prediction problems of various applications.

A novel modeling framework is presented in [3] integrating bivariate empirical mode decomposition and support vector regression for interval forecasting of electricity demand. Bagged Neural Networks (BNN) are used in [4], [5] for prediction. The authors show that BNN reduce forecasting errors, show higher prediction accuracy and have less computational time. Bagged decision trees are employed as predictive models in [5], [6], [7], [8], [9] as they can incorporate a large number of predictor variables and also incorporate the interactions of predictor variables into the final models and these do not have to be specified. Random Forests (RFs) are proposed for minimizing forecasting errors in various applications in [10], [11], [12], [13], [14]. Authors in [15] employ artificial neural networks, multivariate linear regression, RFs, gradient boosting for discrete event simulations and evaluate that RFs and boosting methods outperform multivariate linear regression.

Several previous works focus on Amazon EC2 spot price modeling and prediction. Authors in [16] 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 [17] to construct a model that generates prices consistent with existing price traces. Spot prices are determined artificially by an AR (1) reserve price algorithm. The prices do not represent real client bids around 98 percent of the time. Moreover, simple regressive models might fail to provide insights on how the spot price might evolve in the long run since spot price change at second's granularity [28].

A probabilistic model is presented by authors in [18] 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. Large cost savings are achieved by using CPU optimized instance types. Authors emphasize the need for tuning the parameters such as task length and bid price for a suitable balance between monetary cost and desired service levels. Authors in [19] assume spot prices are normally distributed and focus on achieving availability guarantee with spot instances. The authors use a quantile function of the approximate 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. According to Javadi et al. [20] Gaussians distribution shows better fit. Analysis of predictability of the time-varying spot instance prices in Amazon EC2 is performed in [21]. The authors find that the best achievable prediction is insufficient to provide a close approximation to the actual prices. Statistical analysis for all spot instances in four Amazon's EC2 data centers is provided in [20]. The authors use a mixture of Gaussians distribution to model spot price dynamics and inter-price time of each spot instance and determine the time correlation in spot price in terms of hour-in-day and day-of-week. The authors propose probability density functions (pdfs) for the calculation of spot price and the interval of price spot change.

Empirical distributions models [18], [19], [20], [21] although provide an improved treatment of longer term properties than regressive models; they discard valuable temporal information regarding continuity of spot price staying below different bid values. Therefore, they may not succeed to capture well the continuity of service availability which is of prime concern for applications with long execution time. Such models are unable to distinguish between more frequent and less frequent bid failures with same availability level under a given bid [28].

Authors in [22] model spot instance lifetime by building a Markov Chain and predict spot instance uptime at different bid prices given a starting market price. A discrete semi-Markovian chain to model the spot price variations and verify the Markovian property of the spot price sequences is applied in [23]. However, such multidimensional models might suffer scalability limitations when considering multiple Amazon EC2 regions for better availability and cost minimization [28].

A predictive model based on artificial neural networks is presented in [24] 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. One prediction step takes about 12 seconds in their model. Gradient Descent algorithm is presented in [25] to predict one-day-ahead and one-week-ahead spot prices. The authors take global (monthly and daily) and seasonality trends into account and assume month and hour as important predictor variables. DRAFTS a non parametrical statistical method is presented in [26] which while taking target probability as a parameter predict minimum bid price and the duration the price will meet or exceed the market with at least that probability. Less than 10 percent of the bid prices predicted using DRAFTS meet the probability of success for the duration greater than one hour. Also, the model does not predict spot prices sufficiently ahead in time.

Authors in [27] present a comparative evaluation of prediction algorithms namely Neural Networks, Support Vector Machines, Linear Regression and Gaussian Process. The authors conclude that Support Vector Regression method for spot price forecasts outperforms other methods in terms of MSE, MaxPE, and MeanPE.

Previously proposed models perform data pre-processing to accomplish hourly time intervals to maximize reliability and to present representative values of hourly spot prices. No such data pre-processing is performed in the RFs model used in this work. The prediction horizon in most of the works done earlier is next hour only and it takes sufficiently large time up to 12 seconds to predict next hour price.

In terms of prediction horizons our work closely resembles the work done by [25] where the authors use gradient descent algorithm to predict future prices. The authors use 3 months dataset as training set which makes gradient descent algorithm slow on large datasets to converge. Moreover, spot prices vary depending on demand and supply and usage trends change very fast. It is not justifiable to use a large window size of 3 months for training dataset.

3 Analysis of Spot Price History Data

3.1 Amazon Global Infrastructure

Amazon Web Service is composed of several regions [29] and availability zones as shown in Table 2 to reduce the consequences of outages and provide facility to launch instances in regions that are closer to user thereby reducing latency. Each region has multiple redundant availability zones that provide fault tolerance to cloud resources.

Amazon EC2 announced c3 compute optimized instances in November 2013 and c4 instances in January 2015. The c3

TABLE 2
Amazon EC2 Regions and Availability Zones

Region	Name	Launch Date	Zones
us-east-1	US East (N. Virginia)	2006	5
us-west-2	US West (Oregon)	2011	3
us-west-1	US West (N. California)	2009	3
eu-west-1	EU (Ireland)	2007	3
eu-central-1	EU (Frankfurt)	2014	2
ap-southeast-1	Asia Pacific (Singapore)	2010	2
ap-northeast-1	Asia Pacific (Tokyo)	2011	3
ap-southeast-2	Asia Pacific (Sydney)	2012	2
sa-east-1	South America (Sao Paulo)	2011	3
ap-northeast-2	Asia Pacific (Seoul)	2016	2

TABLE 3
Amazon EC2 Computed Optimized Instance Matrix

API Name	vCPU /cores	Memory (GiB)	ECU	Clock Speed (GHz)
c3.large	2	3.75	7	2.8
c3.xlarge	4	7.5	14	2.8
c3.2xlarge	8	15	28	2.8
c3.4xlarge	16	30	55	2.8
c3.8xlarge	32	60	108	2.8
c4.large	2	3.75	8	2.9
c4.xlarge	4	7.5	16	2.9
c4.2xlarge	8	15	31	2.9
c4.4xlarge	16	30	62	2.9
c4.8xlarge	36	60	132	2.9

TABLE 4
Abbreviations Used

R#	Amazon EC2 Regions	Linux/UNIX Instances		Windows Instances	
R1	ap-northeast-1a	**I1**	c3.large	**I11**	c3.large
R2	ap-southeast-1a	**I2**	c3.xlarge	**I12**	c3.xlarge
R3	ap-southeast-2a	**I3**	c3.2xlarge	**I13**	c3.2xlarge
R4	eu-central-1a	**I4**	c3.4xlarge	**I14**	c3.4xlarge
R5	eu-west-1a	**I5**	c3.8xlarge	**I15**	c3.8xlarge
R6	sa-east-1a	**I6**	c4.large	**I16**	c4.large
R7	us-east-1a	**I7**	c4.xlarge	**I17**	c4.xlarge
R8	us-west-1a	**I8**	c4.2xlarge	**I18**	c4.2xlarge
R9	us-west-2a	**I9**	c4.4xlarge	**I19**	c4.4xlarge
		I10	c4.8xlarge	**I20**	c4.8xlarge

instance and c4 instance type each is available in five different sizes. c3 and c4 instance types provide with the highest performance processors and the lowest price/compute performance compared to all other Amazon EC2 instances. Table 3 shows the API names, number of cores, RAM memory, total processing capacity in EC2 Compute Units and clock speed of compute optimized instance types [30], [31], [32], [33].

Table 4 lists the various abbreviations used for representing Amazon EC2 regions and instance types.

3.2 Analyzing Suitability of SIs for Long Jobs

Any observation that is distant from other observations is an outlier. In the context of spot pricing, we term any spot price as an 'outlier' if it is greater than or equal to on-demand price. In our study, we do not drop any spot price observation just because it is an outlier. They are the legitimate observations and are the most interesting ones. At any point of time the presence of outliers in spot price history is

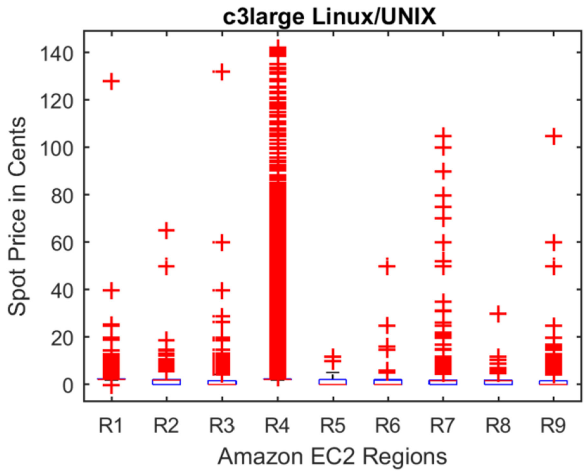

Fig. 1. Box plot showing variations in spot prices in different regions for c3.large Linux/UNIX instance type.

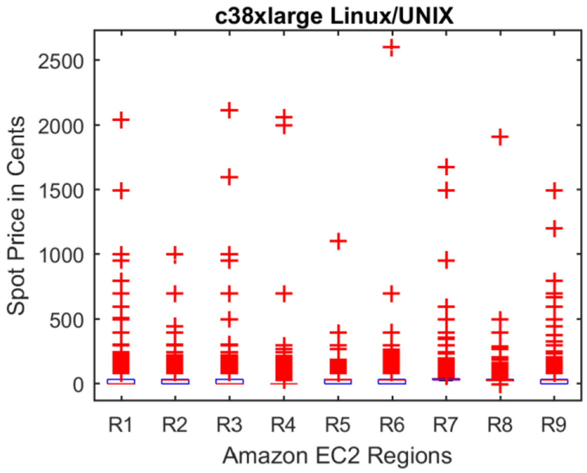

Fig. 2. Box plot showing variations in spot prices in different regions for c3.8xlarge Linux/UNIX instance type.

an important piece of information that can aid in bid decision making. As the spot price data are not normally distributed we use box plots as a non-parametric statistical method to plot historical spot price variations. Fig. 1, Fig. 2 illustrate multiple box plots drawn together to compare spot price history traces of different regions from April 2015 to March 2016 of various compute instance types.

The box plots clearly indicate that there are significant numbers of outliers in spot pricing. Different whisker lengths, number of outliers and variations in box plots clearly indicate that region and instance type both are significant factors in spot pricing. Two different types of fluctuations are observed in spot pricing namely, gradual and steep. Gradual fluctuations in spot pricing are due to changes in demand and supply of compute resources at Amazon EC2 and correspond to users bidding behavior. Rahman et al. [34] showed that fluctuations in Amazon's SIs are much lower than expected values in a free market. This possibly is because of few users for SIs in comparison to other types of reliable resources such as on-demand instances.

We also observe several continuous time epochs when spot prices change abruptly across different regions and instance types occurring at different times. The prices

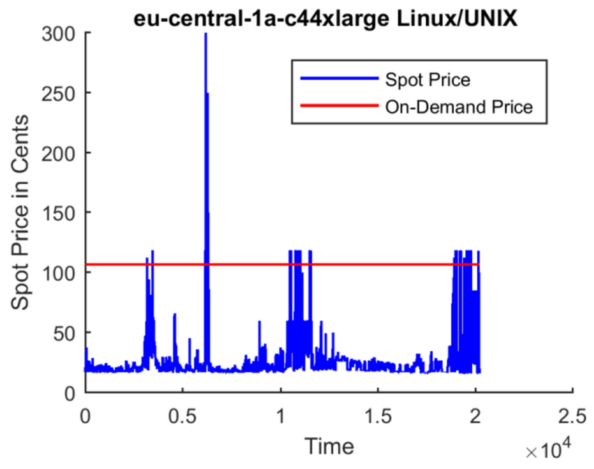

Fig. 3. One year Spot price variations in c4.4xlarge instance.

during these time epochs reach up to 6 to 8 times higher than on-demand prices as shown in Figs. 3 and 4.

We attribute the presence of abruptly high prices to Amazon EC2 spot pricing mechanism itself. This may be due to non-availability of resources in the spot pool in Amazon EC2 regions which terminate all the spot instances allocated. On-demand prices can be considered as an upper bound to bid for spot instances and additionally to increase spot prices. Cloud customers would not choose to acquire unreliable spot instances for long time periods at costs higher than on-demand instances which provide full availability guarantees as it would not result in any monetary savings. We therefore, restrict forecasting period of an instance type to the time period during which spot prices are less than on-demand price for that particular type of instance.

Greater the number of outliers in any region for an instance type, smaller is the forecasting period. Table 5 shows percentage of days (out of 376 days) when spot price is less than on-demand price across various regions and compute instance types for Linux/UNIX instances.

The numbers clearly indicate that spot price is less than on-demand price across most of the regions and instance types for more than 60 percent of days. The numbers also indicate that the general trend in probability of spot prices <on-demand price decreases with the size of instance from I1 to I5. This indicates that the user should opt for instances

TABLE 5
Percentage of Days Spot Price < On-Demand Price

	R1	R2	R3	R4	R5	R6	R7	R8	R9
I1	96	99	95	81	100	99	80	100	94
I2	98	100	91	71	99	99	88	100	82
I3	97	94	86	61	94	96	78	60	85
I4	84	82	53	60	89	88	65	89	75
I5	18	82	41	38	95	74	94	83	78
I6	96	94	96	96	85	NA	88	97	73
I7	82	84	93	94	85	NA	77	92	82
I8	84	85	75	89	77	NA	71	86	81
I9	83	78	74	91	78	NA	59	82	69
I10	73	75	71	88	74	NA	70	79	69

most suitable to the application requirements and not higher infrastructure instances in order to have maximum probability of getting spot instances at low cost with availability guarantee. Regions R5, R2, R8 have greater possibility of getting spot instances.

We plot the length of time epoch durations when spot price < on-demand price in continuation to get a clear picture about the suitability of spot instances for executing fault tolerant applications. The stacked bar graph shown in Fig. 5 clearly indicates that there are sufficient numbers of time epochs of duration ranging from 15 days to more than 150 days and even longer when spot price is significantly less than on-demand price. These time epoch can be effectively utilized by cloud users in order to execute long jobs when spot prices are relatively stable. This raises user's confidence in opting for spot instances.

4 Machine Learning Approaches for Prediction

Machine Learning is that domain of computational intelligence which can be effectively used for constructing computer programs. These programs automatically improve with experience and can be summarized as learning a function (f) that maps input variables (X) to output variables (Y).

$$Y = f(x)$$

Predictive analytics is the area of data mining concerned with forecasting probabilities and trends. The specialized

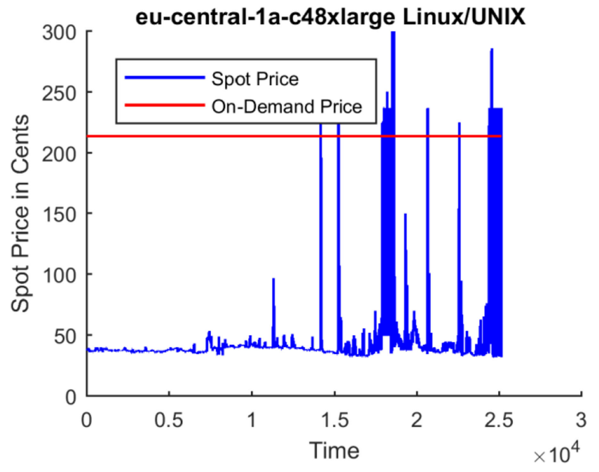

Fig. 4. One year spot price variations in c4.8xlarge instance.

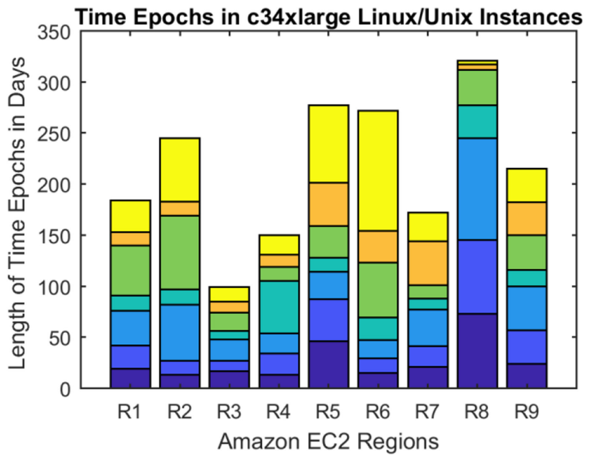

Fig. 5. Time Epochs across various Amazon EC2 regions for c3.4xlarge instance type when spot price<on-demand price.

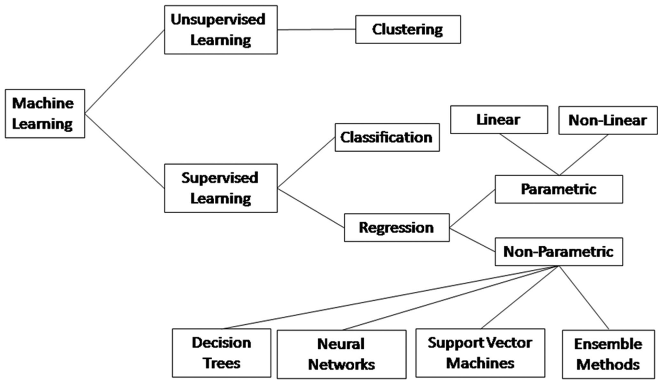

Fig. 6. Machine learning non-parametric algorithms.

statistical techniques required for large scale data analytics separate into two genera: Supervised Learning and Unsupervised Learning. Two basic species of supervised learning methods are regression analysis and classification analysis as shown in Fig. 6.

Classification predicts categorical (discrete, unordered) labels, whereas prediction or regression is most often used for numeric prediction. Algorithms that do not make strong assumptions or make fewer assumptions about the form of the mapping function are called non-parametric algorithms [35]. By not making assumptions, they are free to learn any functional form from the training data and automatically adapt easily to changes in the underlying time dynamics with varying characteristics. Popular non-parametric regression machine learning algorithms are:

- Support Vector Machines
- Multilayer Feed forward Neural Networks
- Decision Trees (Classification/Regression)
- Regression Tree Ensembles
- Random Forests

"Wisdom of crowds" refers to the phenomenon in which aggregated predictions from a large group of people can be more accurate than most individual judgments and can rival or even beat the accuracy of subject matter experts [36]. Ensemble methods are learning models that achieve performance by combining the opinions of multiple learners [37]. In doing so one can often get away with using much simpler learners and still achieve great performance in terms of increased robustness and accuracy.

4.1 Evaluation of Existing Non Parametric Machine Learning Algorithms with Random Forests based Predictions

We compare one-day-ahead prediction accuracy of RFs with existing machine learning approaches for spot price prediction namely Support Vector Machines [27] and Neural Networks [24] for 4 different compute instance types each in a different region for a period of 120-150 days forecasts. The codes of the algorithms evaluated are freely available. Prediction accuracy is compared using different forecasting error metrics namely Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Maximum Positive Error (MAXPE), Mean Positive Error (MEANPE), Mean Squared Error (MSE).

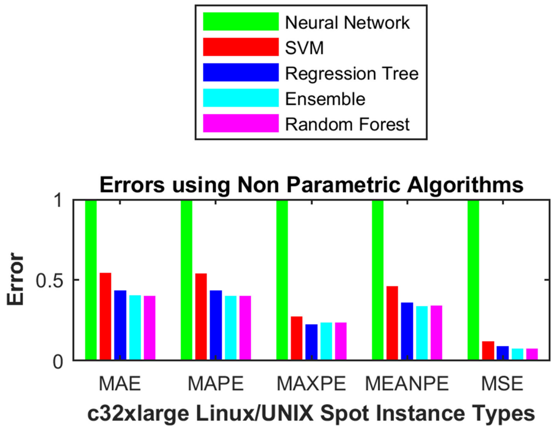

Fig. 7. Comparison of different errors (normalized) in spot price prediction using various non parametric algorithms.

Results show that prediction accuracy of Random Forests outperform prediction accuracy of Neural Networks and Support Vector Machines in all cases. Fig. 7 shows prediction accuracy of ap-southeast-1a c32xlarge instance type for a period of 150 days. The results obtained for SVM and Neural Networks are similar to the results obtained by S. Arevalos et al. [27] who compare Gaussian, Linear Regression, SVM and Neural Networks and show that among the four algorithms compared by the authors, SVM outperforms Neural Network prediction.

5 Random Forests Prediction Model

5.1 Bias and Variance Heuristics

Bias of a learning algorithm (for a given learning problem and a fixed size N of training sets) [38] is the persistent or systematic error that the learning algorithm is expected to make when trained on training set of size N. It measures the accuracy or quality of the algorithm. High bias means a poor match. The error due to bias is the difference between the expected (or average) prediction of any model and the correct value of the response variable which we are trying to predict.

Variance of an algorithm is defined as the expected value of the squared difference between any particular hypotheses and the averaged hypothesis [38]. Variance measures how predictions vary with respect to the average prediction. A high variance means a weak match.

If we denote the response variable we are trying to predict as Y and predictors as X, we may assume that there is a relationship relating one to the other such as $Y = f(x) + \epsilon$ where the error term ϵ is normally distributed with a mean of zero. Considering an estimated model prediction $\hat{f}(X)$ of $f(X)$ using any modeling technique, the expected squared prediction error at a point is

$$Err(x) = E[\ \left(Y - \hat{f}(x)\right)^2\ ]. \qquad (1)$$

This error may then be decomposed into bias and variance components

$$Err(x) = \left[E\left(\ \hat{f}(x)\right) - f(x)\right]^2 + E\left[\hat{f}(x) - E\left[\hat{f}(x)\right]\right]^2 + \sigma_e{}^2)\ . \qquad (2)$$

The third term, irreducible error, is the noise term in the true relationship that cannot fundamentally be reduced by

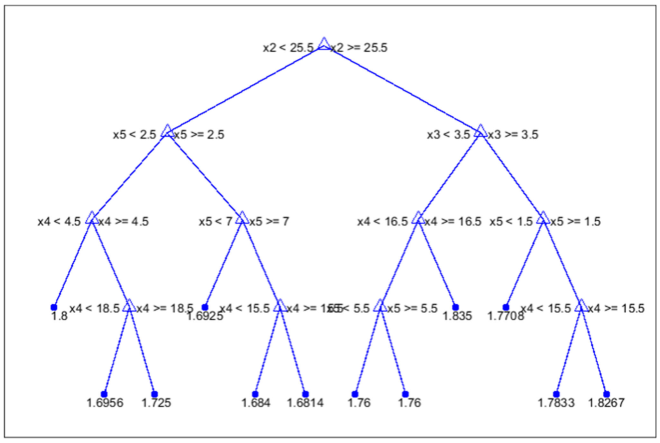

Fig. 8. Regression tree1 obtained from bootstrap samples.

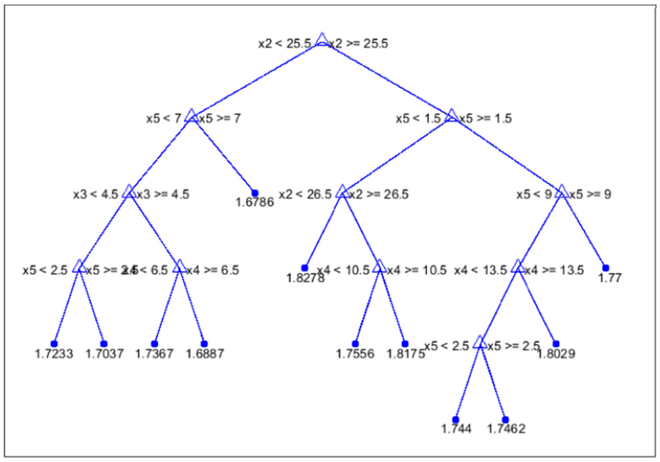

Fig. 9. Regression tree2 obtained from bootstrap samples.

any model [39]. The first term is the bias term which is the squared difference between the true mean $f(x)$ and the expected value of the estimate.

Both high bias and high variance are negative properties for types of models. A model with high bias "under-fits" the data and a model with high variance and low bias "over-fits" the data [40]. Over-fitting is usually a more major problem in machine learning than under-fitting. It is often the case that there is noise (small, random errors) in the data. Over-fitting the data causes the model to fit the noise, rather than the actual underlying behavior. Models with less variance are more robust (fit a better model) in the presence of noisy data. Trees have low bias as they can capture complex interaction structures in the data. Trees have high variance due to the correlation in the predictors.

5.2 Bias and Variance Reduction Techniques

Bootstrapping is a resampling technique used for assessing statistical accuracy [37]. B Training sets $Z^{*b}, b = 1, \ldots, B$ each of size $|Z|$ are drawn with replacement from the original training dataset. This is done B times producing B bootstrap datasets.

$$Z^{*b} = \left\{ x \,\middle|\, \begin{array}{c} x \; is \; drawn \; at \; random \\ with \; replacement \; from \; Z \end{array} \right\} and \; \left|Z^{*b}\right| = |Z|.$$

Variance of bootstrap sample is

$$\widehat{Var}\left[\hat{Y}(Z)\right] = \frac{1}{B}\sum\nolimits_{b=1}^{B}\left(\hat{Y}(Z^{*b}) - \bar{\hat{Y}}^{*}\right)^2 \tag{3}$$

where $\hat{Y}(Z)$ represents prediction and $\hat{\mathrm{Y}}^{-*} = \sum_{\mathrm{b}} \hat{\mathrm{Y}}(\mathrm{Z}^{*\mathrm{b}})/\mathrm{B}$.

Bagging uses the bootstrap to improve the estimate or prediction itself. Each bootstrap tree will involve different features than the original, and might have a different number of terminal nodes as shown in Figs. 8 and 9.

Bagging perturbs the data in an i.i.d. fashion and re-estimates the model to give a new set of model parameters. At the end, a simple average of the model predictions from different bagged samples is computed.

Suppose we fit a model to our training data $Z = (x_1, y_1), (x_2, y_2), \;\; (x_3, y_3), \ldots (x_n, y_n)$, obtaining the prediction $\hat{f}(x)$ of tree at input x. Bagging averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample $Z^{*b}, \;\; b = 1, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{4}$$

Bagging helps under squared error loss as it reduces variance leaving bias unchanged since each tree generated in bagging is identically distributed(i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them. Main caveat here is "independent," and bagged trees are not.

5.3 Random Forests

Random Forests [41], [42] (Leo Breiman, 2001) is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them. The method uses an algorithm for induction of regression trees which combines random subspaces with bagging.

This is achieved in the tree-growing process through random selection of the input variables. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split used at each node is the best split among a random subset of the features. Specifically, when growing a tree on a bootstrapped dataset

$$\begin{array}{c} Before \; each \; split, \; select \; m \; \leq \; p \; of \; the \; input \; variables \\ at \; random \; as \; candidates \; for \; splitting \end{array}$$

The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. This strategy turns out to perform very well compared to discriminant analysis, support vector machines and neural networks, and is robust against over-fitting.

After B such trees $\{T(x; \theta_b)\}_1^{\mathrm{B}}$ are grown, the Random Forests (regression) predictor is

$$\hat{f}_{rf}^{B}(x; \theta_1, \ldots, \theta_B; Z_n) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x; \theta_b; Z_n) \tag{5}$$

where $\theta_1, \ldots, \theta_B$ are independent random variables, distributed the same as a generic random variable θ and independent of Z_n.

5.4 Random Forests Parameters

The algorithm has four important parameters:

1. the data set $Z \in \{1, \ldots n\}$,
2. leaf size or desired depth of the decision trees i.e., $nodesize \in \{1, \ldots n\}$,
3. a number B of total decision trees to build.
4. $m \in \{1, \ldots p\}$: which is how many features chosen randomly from feature variables in X to search over to find the best feature.

5.5 Algorithm for Random Forests

Input: Training set Zn, number of trees $B > 0, Z \in \{1, \ldots n\}$, $m \in \{1, \ldots p\}$, leafsize $\in \{1, \ldots n\}$: and $x \in X_1, X_2, \ldots X_p$

Output: Prediction of the Random Forests at x.

```
1) For $b = 1$ to $B$ :
   a) Draw a bootstrap sample $Z^*$ of size $n$ from the training data.
   b) Grow a Random Forests tree $T_b$ from the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum leaf size is reached.
      i) Select $m$ variables at random from the $p$ input variables.
      ii) Pick the best variable/split-point among $m$.
      iii) Split the node into two daughter nodes. The leaves of this tree, where predictions are made, are filled in based on the training data.
2) Compute the predicted value $\hat{f}_n(x; \theta_b; Z_n)$ at $x$ equal to the average of the $y_i$ falling in the cell of $x_i$ $\hat{f}_c = \frac{1}{n_c} \sum_{y \in c} y_i$, the prediction for leaf $c$.
3) Compute the Random Forests estimate $\hat{f}^B_{rf}(x; \theta_1, \ldots, \theta_B; Z_n)$ from ensemble of trees $\{T_b\}_1^B$ and output the estimate at query point $x$ according to (5).
```

5.6 Cross Validation in Random Forests

OOB Error is RFs cross validation method. RFs has its own way of estimating predictive accuracy ("out-of-bag" estimates). It uses bootstrapping while building decision tree model. Bootstrapped data sets will be similar. However, they will not be too similar if N is large. The probability that the first training example is not selected once is $(1 - \frac{1}{N})$ and probability of not getting selected at all is $(1 - 1/N)^N$. As N tends to 100 or more this is equivalent to 0.366. So, only about 63.4 percent of the original training examples will be represented in any given bootstrapped set. The remaining 36.6 percent of observations are "out-of-bag" observations. Spot price training dataset size N for one-day-ahead and one-week-ahead predictions is greater than 100.

OOB Error estimates the mean squared error from the out-of-bag prediction for each observation by computing average of predictions obtained from all trees in the ensemble in which the observation under consideration did not appear (out-of-bag). It is the difference between the out-of-bag predicted responses against the true responses. Out-of-bag average estimates true ensemble error eliminating the need of cross validation or a separate test set. Out-of-bag estimate is as accurate as using a test set of the same size as the training set.

6 Fitting Random Forests Model

Spot price history traces contain the following attributes: (1) region/availability zone, (2) instance type, (3) platform/operating system, (4) spot price, (5) Time

ap-northeast-1a c3.largeLinux/UNIX 0.0286 2016-06-27T05:37:0Z

We use time and spot price attributes for prediction purpose. Let $(x_1, y_1), (x_2, y_2) \ldots .(x_n, y_n)$ be some sequence of spot price data points in training dataset Z.

$$(x, Y) = (x1, x2, x3, x4, x5, x6, x7, Y) \tag{6}$$

The predictor vector x represents time when spot prices change and is used for prediction task. It is decomposed into seven variables. The target/dependent vector Y represents spot price that we are trying to predict.

$$(time,\ spot\_price) = \begin{pmatrix} year, month, day,\ weekday, \\ hour, minute, seconds, spot\_price \end{pmatrix}$$

It is critical to use RFs that makes predictions that are both fast and accurate. Parameters in RFs are either to increase the predictive power of the model or to make it easier to train the model. We tune different RFs design parameters to jointly minimize two objectives: 1) the prediction error on OOB samples and 2) the prediction speed.

6.1 Variable Importance

RFs use the OOB samples to construct variable importance to measure the prediction strength of each input variable. Let j be some variable from $\{1 \ldots p\}$. Variable Importance (VI) of $X^{(j)}$ in Random Forests [43] is defined as mean difference between OOB Error after randomly permuting the values of $X^{(j)}$ in the OOB samples of each tree and OOB Error without randomly permuting the values of $X^{(j)}$ in the OOB samples of each tree when B trees are grown.

$$VI(X^{(j)}) = \frac{1}{B} \sum_{b=1}^{B} (OOBError_b^j - OOBError_b) \tag{7}$$

Fig. 10 illustrates the importance of various variables in spot instance pricing in different instance types and Amazon EC2 regions using RRFs model. We see that $VI^{hour} > VI^{day} > VI^{weekday} > VI^{minute} > VI^{month}$.

Spot prices vary considerably during different hours of the day as shown in Figs. 11 and 12.

6.2 Leaf Size

The leaf nodes of the individual regression trees contain a fixed pre-specified number of nodes. Trees with smaller leaf size (leading to deep trees) are more prone to capture noise in the training data leading to over fitting. Higher leaf size has greater variance which leads to poor predictive performance. Optimal leaf size depends on the training dataset. In order to find optimal leaf size for our use-case we compute out-of-bag error for various leaf sizes. As shown in Fig. 13

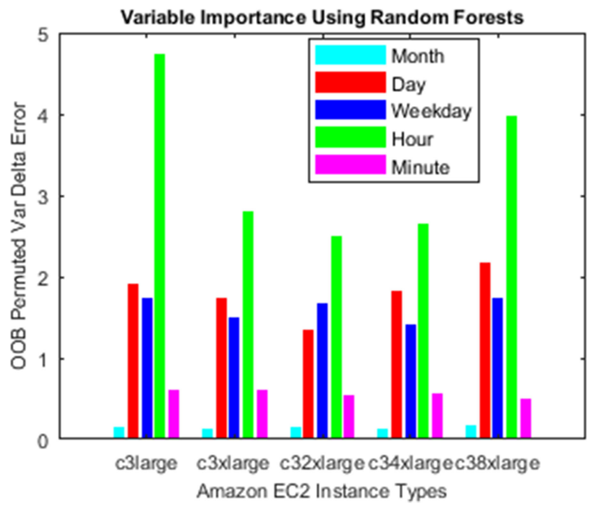

Fig. 10. Variable importance in spot pricing for different instance types.

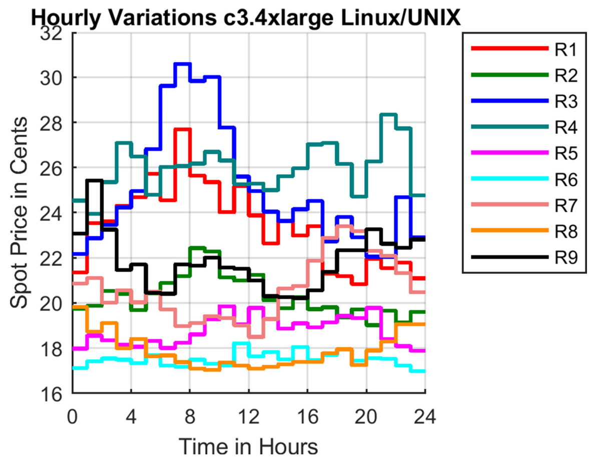

Fig. 11. Average daily spot price variatonsin different regions.

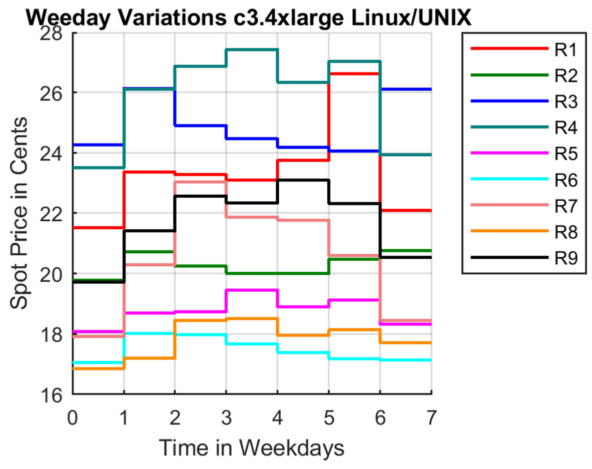

Fig. 12. Average weekday spot price variations in different regions.

we see that leaf size has a significant impact in minimizing OOB Error. We keep optimal leaf size = 5 for the spot price prediction.

6.3 Number of Trees

Random Forests achieve lower test error by means of variance reduction without affecting bias. This is achieved in RRFs by reducing correlation between trees and by increasing the number of trees grown. Fig. 13 illustrates reduction in OOB Error with the number of leaves and the trees grown. Less number of trees leads to high bias leading to high OOB Error. Bias reduces considerably when the number of trees grown is increased. Further increase in number of trees grown results in reduction in variance which reduces OOB Error considerably. When the number of trees grown is increased beyond 50, no significant decrease in OOB Error is observed. We keep number of trees grown = 50 in the model.

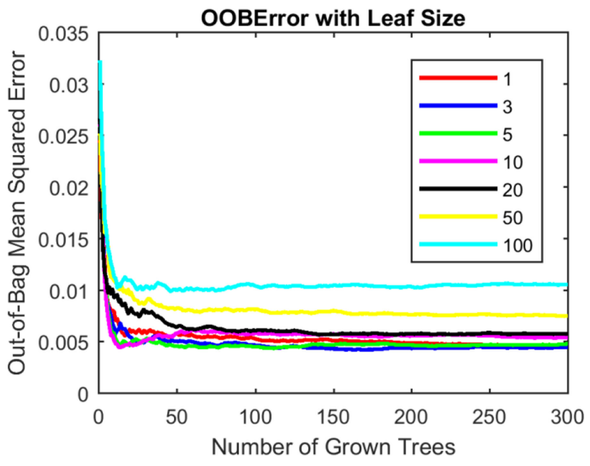

Fig. 13. OOB errors with number of grown trees and leaf size.

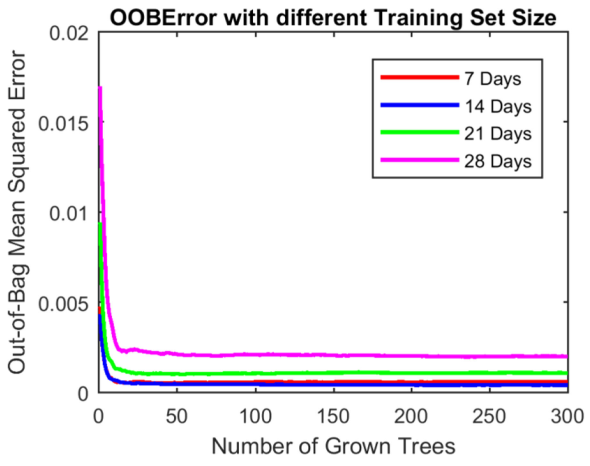

Fig. 14. OOB Error with varying training dataset size.

6.4 Training Dataset Size

Small variation in OOB Error is observed when training dataset size is varied between 7 and 14 days as shown in Fig. 14. Increasing the RRFs training data size beyond 14 days leads to high OOB Error. This clearly shows that patterns in spot prices change considerably within a short period of time and only a small window size of past history traces is required to accurately predict spot prices. We take optimal amount of past spot price history equal to 7 days for training dataset where prediction error are minimal. The dataset size incorporates the effect of all important feature variables in spot pricing namely hour, day and weekday.

6.5 Accuracy Measures

We evaluate the prediction accuracy and speed based on Mean Absolute Percentage Error (MAPE), Mean Consequential Percentage Error (MCPE) and Prediction Time.

TABLE 6
One-Day-Ahead Mape in Linux/Unix Instance Types

Instance type	MAPE 5%	MAPE 10%	MAPE 15%
I1	81.54	93.32	97.29
I2	65.46	87.36	94.05
I3	44.15	72.28	85.93
I4	43.03	68.66	83.52
I5	46.06	77.97	88.97
I6	70.44	86.35	92.79
I7	61.74	81.90	92.06
I8	64.52	84.59	92.99
I9	41.45	68.08	82.83
I10	47.54	72.57	84.87

TABLE 7
One-Day-Ahead Mape in EC2 Regions Linux/Unix

Region	MAPE 5%	MAPE 10%	MAPE 15%
R1	57.09	80.83	90.07
R2	64.37	83.58	92.50
R3	69.75	86.28	92.77
R4	54.89	76.16	86.69
R5	78.14	92.62	96.78
R6	72.29	88.28	94.86
R7	39.82	73.29	89.98
R8	47.09	72.13	83.66
R9	34.57	65.67	81.45

6.5.1 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percent Error (MAPE) is the most common parametric measure of prediction accuracy specifically in trend estimation. We use MAPE as we compare prediction accuracy on several series with different scales (different regions and instance types). It is commonly used when the amount by which numerical predictions are in error is evenly important. MAPE functions best when there are no extremes to the data (including zeros).

$$MAPE = \left(\frac{1}{n}\sum\frac{abs\left(y_i - y_f\right)}{y_i}\right) * 100 \tag{9}$$

6.5.2 Mean Consequential Percentage Error (MCPE)

We propose a new prediction assessment metric Mean Consequential Percentage Error (MCPE) for assessing predictability of regression problems such as spot price prediction adapted from Mean Consequential Error mostly used in Kaggle competitions [44] for classification problems. The mean of "Consequential Error" in regression problems where all errors are equally bad (1) and the only value that matters is an approximate prediction (0) where

$$\frac{abs(y_i - y_f)}{y_i} > m\% \text{ (1), and } \frac{abs(y_i - y_f)}{y_i} < m\% \text{ (0)}.$$

$$MCPE = \left(\frac{\sum\frac{abs\left(y_i - y_{pred}\right)}{y_i} > m\%^{\,1}}{n}\right) *100 \tag{10}$$

6.5.3 Prediction Time

It is the time required by the RRFs model to make spot price predictions. We measure the time elapsed for training and spot price predictions for each day in order to assess the prediction speed of the model.

7 Prediction Anaysis

We use Amazon's spot instance market as the target market for price prediction. We collated spot price history traces of all compute optimized instance types across various regions as listed in Table 1 beginning from April 2015 to March 2016 from Amazon EC2 console using command line interface [45] after installing AWS CLI in Ubuntu which provides past 90 days spot price history.

Matlab 2016a trial version is used as a simulation tool for spot price prediction. We performed experiments with the aim of predicting future spot prices for both medium term and long term horizons. We also evaluate the prediction model based on RRFs in terms of the various forecasting metrics. Since the number of experiments conducted to evaluate MAPE, MCPE, and prediction time is very large, we present here average results only. In evaluating MCPEm percent for RRFs based predictions, we consider values of $m = 5$, 10 and 15 percent. Two categories of experiments namely Linux/UNIX and Windows instances are performed. Under each category, two sets of results are presented. First set of results examine prediction accuracy across Amazon EC2 regions and second set of results examines prediction accuracy in different compute instance types.

7.1 Daily Prediction Accuracy

We simulate 85 use-cases (8 regions*10 instance types+ 1 region*5 instance types) as listed in Table 4 each for both the operating systems Linux/UNIX and Windows for one-day-ahead spot price prediction. Each use-case consists of 50 days to 376 days spot price prediction.

7.1.1 Linux/Unix Instances

Table 6 shows daily prediction accuracy with respect to MAPE in different compute instance types for Linux/UNIX instance types. $MAPE \; <= 10\%$ for 68 to 93 percent of predictions in different instance types. $MAPE \; <= 15\%$ for 82 to 97 percent of predictions in different instance types. Instance types I1, I2, I6 and I8 show better prediction accuracy in terms of MAPE than other instance types.

Table 7 shows daily prediction accuracy with respect to MAPE across various Amazon EC2 regions for Linux/UNIX instance types. $MAPE \; <= 10\%$ for 66 to 92 percent of predictions and $MAPE \; <= 15\%$ for 81 to 96 percent of predictions. Regions R2, R3, R5, and R6 show better prediction accuracy in terms of MAPE than other regions.

Table 8 shows prediction accuracy in terms of MCPE in different instance types for Linux/UNIX instances. $MCPE <= 10\%$ for 28 to 71 percent of predictions in different types of compute instances. $MCPE <= 15\%$ for 40 to 81 percent of predictions in different types of compute instances. Instance types I1, I2, I6, I8 show better prediction accuracy in terms of MCPE than other instance types.

Table 9 shows prediction accuracy in terms of MCPE across various Amazon EC2 regions for Linux/UNIX instances. $MCPE \; <= 10\%$ for 25 to 67 percent of predictions in different regions. $MCPE \; <= 15\%$ for 37 to 81 percent of predictions in different regions. Regions R3, R4, R5,

TABLE 8
One-Day-Ahead MCPE in Different Instance Types

Instance type	MCPE 5%	MCPE 10%	MCPE 15%
I1	49.91	71.60	81.44
I2	31.69	51.78	64.32
I3	14.09	28.20	40.44
I4	18.68	29.05	41.90
I5	18.14	32.08	48.24
I6	44.89	64.41	74.29
I7	33.51	51.28	62.88
I8	36.42	52.39	65.69
I9	20.81	33.70	44.50
I10	20.18	37.58	50.92

TABLE 9
One-Day-Ahead MCPE in Amazon EC2 Regions Linux/Unix

Region	MCPE 5%	MCPE 10%	MCPE 15%
R1	26.80	44.80	57.68
R2	33.41	51.17	66.65
R3	38.79	53.19	68.28
R4	32.05	54.14	58.36
R5	50.00	67.59	81.16
R6	47.67	53.39	70.55
R7	7.85	24.75	37.24
R8	22.84	36.38	48.45
R9	9.19	23.34	35.19

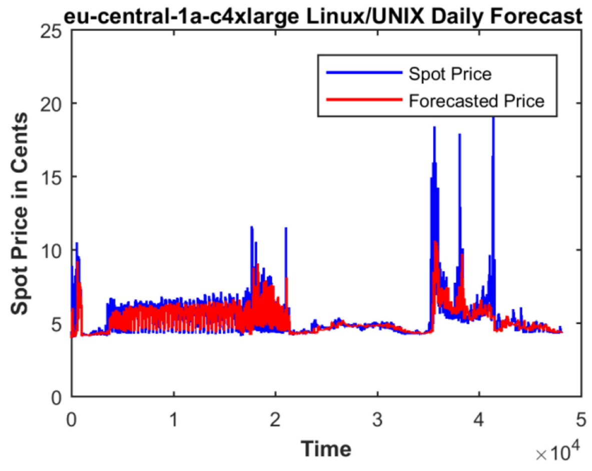

Fig. 15. Amazon EC2 Linux/UNIX spot instance price prediction from April 2015 to March 2016 using RRFs in eu-central-1a region.

R6 show better prediction accuracy in terms of MCPE than other regions.

Figs. 15 and 16 show medium term (one-day-ahead) spot price prediction for Linux/UNIX instances. Spot price predictions using RRFs are very close to actual spot prices during most of the times excluding when spot prices rise extremely high.

7.1.2 Windows Instances

Table 10 shows prediction accuracy with respect to MAPE across different Amazon EC2 regions for Windows instance types. $MAPE <= 1\%$ for 71 to 99 percent of predictions across various regions. Regions R1, R2, R5, and R9 show better prediction accuracy in terms of MAPE.

Table 11 shows prediction accuracy with respect to MAPE in different compute instance types for Windows

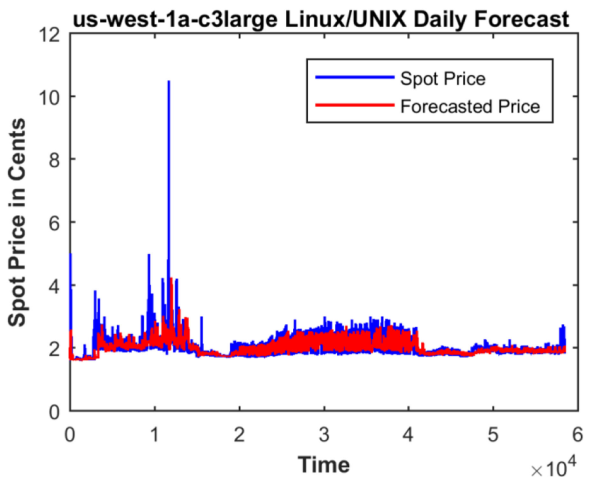

Fig. 16. Amazon EC2 Linux/UNIX spot instance price prediction from April 2015 to March 2016 using RRFs in eu-central-1a region.

TABLE 10
One-Day-Ahead MAPE in Windows Instances

Region	MAPE 1%	MAPE 5%	MAPE 10%
R1	89.42	99.06	99.65
R2	91.37	98.39	99.36
R3	71.18	90.87	98.60
R4	76.63	95.09	98.55
R5	99.44	100.00	100.00
R6	73.07	95.09	99.03
R7	77.20	96.04	98.71
R8	85.24	96.32	98.03
R9	96.08	99.15	99.70

TABLE 11
One-Day-Ahead MAPE in Windows Instance Types

Instance Types	MAPE 1%	MAPE 5%	MAPE 10%
I11	98.22	99.27	99.61
I12	94.90	98.74	99.33
I13	73.66	97.25	99.97
I14	80.43	96.56	99.65
I15	69.01	88.83	95.51
I16	90.93	98.18	99.62
I17	93.67	99.35	99.85
I18	89.20	99.00	99.90
I19	80.85	95.03	98.08
I20	80.99	95.80	99.37

instance types. $MAPE <= 1\%$ for 69 to 98 percent of predictions in different instance types. Instance types I11, I12, I16, and I17 (lower infrastructure) show better prediction accuracy in terms of MAPE than other instance types.

Evaluation results for Windows instances also reveal that $MCPE <= 5\%$ for 77 to 99 percent of spot price predictions across all regions. Also $MCPE <= 5\%$ for 73 to 98 percent of spot price predictions in different Windows compute instance types. Forecast accuracy for Windows instance types is much higher than Linux/UNIX type.

7.2 Weekly Prediction Accuracy

In this sub-section we assess the accuracy of one-week-ahead spot price prediction. The total number of weeks for which spot price history traces were collected is 55 weeks.

TABLE 12
% of Weeks Spot Price < On-Demand Price (Linux/Unix)

	R1	R2	R3	R4	R5	R6	R7	R8	R9
I1	85	96	87	65	98	96	45	98	89
I2	89	100	71	49	96	96	73	100	58
I3	89	76	62	29	76	82	53	29	67
I4	56	58	16	31	67	64	35	73	49
I5	0	53	16	15	87	56	80	49	51
I6	82	84	84	89	76	NA	64	91	51
I7	51	47	73	80	73	NA	53	76	67
I8	60	53	56	71	65	NA	47	65	60
I9	55	40	51	76	62	NA	24	56	42
I10	36	40	38	65	56	NA	40	56	40

TABLE 13
One-Week-Ahead MAPE in EC2 Regions Linux/ Unix

Region	MAPE 10%	MAPE 15%
R1	78.62	87.84
R2	84.33	94.56
R3	86.30	90.59
R4	62.74	81.94
R5	87.38	96.95
R6	80.61	87.63
R7	72.90	93.45
R8	65.69	85.76
R9	53.13	71.81

TABLE 14
One-Week-Ahead MAPE in Different Instance Types Linux/Unix

Instance Types	MAPE 10%	MAPE 15%
I1	93.79	98.01
I2	81.07	90.91
I3	68.49	88.75
I4	58.92	85.78
I5	77.46	92.32
I6	80.51	91.02
I7	78.93	91.35
I8	83.27	91.56
I9	47.67	70.45
I10	71.52	78.82

Table 12 shows percentage of weeks when spot price is less than on-demand price across various Amazon EC2 regions and instance types. The numbers clearly indicate that the general trend in probability of spot prices <on-demand price decreases with the size of instance type from I1 to I5 and from I6 to I10 which is similar to the results shown in Table 5. Regions R6, R5, R8 show greater percentage of weeks when spot prices <on-demand price.

Table 13 shows one-week-ahead prediction accuracy with respect to MAPE across different Amazon EC2 regions for Linux/UNIX instance types. $MAPE \quad <= 10\%$ for 53 to 87 percent of predictions in different regions. $MAPE \quad <= 15\%$ for 71 to 96 percent of predictions in different regions. Similar to one-day-ahead predictions, regions R2, R3, R5, R6 show better prediction accuracy in terms of MAPE than other regions. This shows that the spot prices in these regions are relatively stable (less number of outliers) than other regions and therefore, can be predicted more accurately for a longer forecasting horizon. About 10 percent decrease in prediction accuracy MAPE is observed in one-week-ahead price predictions of different regions in comparison to one-day-ahead price prediction.

TABLE 15
Average One-Day-Ahead Prediction Time in Seconds

Region #	Prediction Time	Instance Type	Prediction Time
R1	0.695	I1	0.585
R2	0.683	I2	0.785
R3	0.619	I3	0.872
R4	0.589	I4	0.795
R5	0.681	I5	0.865
R6	0.656	I6	0.506
R7	0.851	I7	0.603
R8	0.616	I8	0.626
R9	0.741	I9	0.557
		I10	0.593

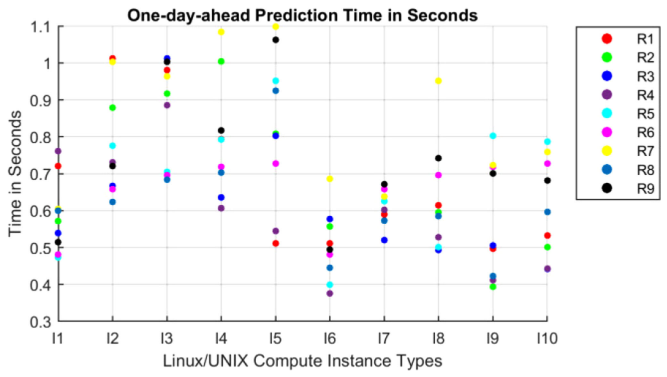

Fig. 17. Prediction time for one-day-ahead prediction in different regions and different compute instance types.

Table 14 shows one-week-ahead prediction accuracy with respect to MAPE for Linux/UNIX type. $MAPE \quad <= 10\%$ for 47 to 93 percent of predictions in different instance types. $MAPE \quad <= 15\%$ for 70 to 98 percent of predictions in different instance types. Similar to one-day-ahead predictions instance types I1, I2, I6, I8 show better prediction accuracy in terms of MAPE than other instance types. About 12 - 20 percent decrease in MAPE is observed in one-week-ahead predictions of different instance types in comparison to one-day-ahead predictions.

7.3 Prediction Time

Table 15 shows average one-day-ahead spot price prediction time in seconds across various Amazon EC2 regions and different compute instance types. Spot price prediction time is greater in higher capacity c3 type instances (I3, I4, and I5) which is directly proportional to the number of training records in training dataset. Increase in prediction time of higher capacity instance types is an indication of greater number of spot price changes in these instance types. Similarly, regions R7 and R9 have higher prediction time than other regions which is an indication of greater spot price fluctuations in these regions.

Fig. 17 shows average spot price prediction time across various regions and instance types in seconds. We see that prediction time for one-day-ahead predictions is less than 1 second. This indicates that the RRFs based predictions due to their efficient tree structure have fast prediction speed.

8 Conclusions

Spot pricing encourages users to shift execution of flexible workloads from provider's peak hours to off-peak hours and thus obtain monetary incentives. Analysis of one year spot price history data shows that there are sufficient number of time epochs of duration ranging from 30 days to more than 100 days and even longer when spot prices are up to 6 to 8 times cheaper than on-demand prices. It is therefore reasonable for users to shift their workloads from on-demand to spot instances. This work presents application of RRFs for Amazon EC2 spot price prediction. We compare several non-parametric machine learning prediction algorithms for spot price prediction in terms of various forecasting accuracy measures and conclude that RRFs outperforms other methods. Evaluation results show that $MAPE \quad <= 10\%$ for 66 to 92 percent and $MCPE \quad <= 15\%$ for 35 to 81 percent of one-day-ahead predictions in different regions. $MAPE \quad <= 15\%$ for 71 to 96 percent for One-week-ahead predictions in different regions. Instance types with lower infrastructure show better prediction accuracy than those with higher infrastructure. The prediction results suggest that spot price predictions using RRFs are more accurate than other machine learning algorithms. Prediction time for all predictions is less than 1 second. This indicates that the RRFs based predictions are fast enough to predict spot prices. Furthermore, RRFs when used for spot price predictions have only a few adjustable parameters namely number of regression trees and leaf size. One-week-ahead and one-day-ahead predictions along with feature importance can be effectively used by spot users to plan job executions in advance and bid effectively leading to significant cost savings and reducing out-of-bid failure probability of spot instances.

References

[1] W. Guo, K. Chen, Y. Wu, and W. Zheng, "Bidding for highly available services with low price in spot instance market," in *Proc. 24th Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2015, pp. 191–202.

[2] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 71–84, Aug. 2015

[3] T. Xiong, Y. Bao, and Z. Hu, "Interval forecasting of electricity demand: A novel bivariate EMD-based support vector regression modeling framework," *Int. J. Electr. Power Energy Syst.*, vol. 63, pp. 353–362, Dec. 2014.

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

[5] R. Ghani, "Price prediction and insurance for online auctions," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 411–418.

[6] N. Nur, et al., "Where the wild things are: Predicting hotspots of seabird aggregations in the california current system," *Ecological Appl.*, vol. 21, no. 6, pp. 2241–2257, Sep. 2011.

[7] L. Yu, Z. Wang, and L. Tang, "A decomposition–ensemble model with data-characteristic-driven reconstruction for crude oil price forecasting," *Applied Energy*, vol. 156, pp. 251–267, Oct. 2015.

[8] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–40, Nov. 2012

[9] M. Oliveira and L. Torgo, "Ensembles for time series forecasting," *Asian Conf. Mach. Learn.*, pp. 360–370, 2014.

[10] B. Larivière and D. Van Den Poel, "Predicting customer retention and profitability by using random forests and regression forests techniques," *Expert Syst. Appl.*, vol. 29, no. 2, pp. 472–484, Aug. 2005.

[11] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu, "Regression forests for efficient anatomy detection and localization in CT studies," in *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging*, Berlin, Germany: Springer, 2010. pp. 106–117.

[12] E. Antipov and E. Pokryshevskaya, "Mass appraisal of residential apartments: An application of Random forest for valuation and a CART-based approach for model diagnostics," *Expert Syst. Appl.*, vol. 39, no. 2, 1772–1778, 2012.

[13] A. Booth, E. Gerding, and F. McGroarty, "Automated trading with performance weighted random forests and seasonality," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3651–3661, Jun. 2014.

[14] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using Random forest." *Appl. Math. Finan., CoRR*, vol. abs/1605.00003, pp. 1–20, 2016.

[15] W. Budgaga, M. Malensek, S. Pallickara, N. Harvey, F. J. Breidt, and S. Pallickara, "Predictive analytics using statistical, learning, and ensemble methods to support real-time exploration of discrete event simulations," *Future Generation Comput. Syst.*, vol. 56, pp. 360–374, Mar. 2016.

[16] 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 Comput.: Practice Experience*, vol. 27, no. 9, pp. 2260–2277, Dec. 2012.

[17] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon EC2 spot instance pricing," *ACM Trans. Economics Comput.*, vol. 1, no. 3, pp. 1–20, Sep. 2013.

[18] A. Andrzejak, D. Kondo, and S. Yi, "Decision model for cloud computing under SLA constraints," in *Proc. IEEE Int. Symp. Modeling Anal. Simul. Comput. Telecommun. Syst.*, Aug. 2010, pp. 257–266.

[19] M. Mazzucco and M. Dumas, "Achieving performance and availability guarantees with spot instances," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, Sep. 2011, pp. 296–303.

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

[21] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, "Optimal resource rental planning for elastic applications in cloud market," in *Proc. Symp. IEEE 26th Int. Parallel Distrib.*, May 2012, pp. 808–819.

[22] N. Chohan, et al., "See spot run: Using spot instances for MapReduce workflows," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 7–7.

[23] M. Zafer, Y. Song, and K.-W. Lee, "Optimal bids for spot VMs in a cloud for deadline constrained jobs," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 75–82.

[24] R. M. Wallace, et al., "Applications of neural-based spot market prediction for cloud computing," in *Proc. IEEE 7th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst.*, Sep. 2013, pp. 710–716.

[25] V. K. Singh and K. Dutta, "Dynamic price prediction for amazon spot instances," in *Proc. 48th Hawaii Int. Conf. Syst. Sci.*, Jan. 2015, pp. 1513–1520.

[26] R. Wolski and J. Brevik, "Providing statistical reliability guarantees in the AWS spottier," in *Proc. 24th High Perform. Comput. Symp.*, 2016, Art. no. 13.

[27] S. Arevalos, F. Lopez-Pires, and B. Baran, "A comparative evaluation of algorithms for auction-based cloud pricing prediction," in *Proc. IEEE Int. Conf. Cloud Eng.*, Apr. 2016, pp. 99–108.

[28] C. Wang, Q. Liang, and B. Urgaonkar, "An empirical analysis of amazon EC2 spot instance features affecting cost-effective resource procurement," in *Proc. 8th ACM/SPEC Int. Conf. Perform. Eng.*, 2017.

[29] AWS Global Infrastructure, Retrieved From: 2015, <https://aws.amazon.com/about-aws/global-infrastructure/>, Accessed on: Mar. 2015.

[30] EC2Instances.info Easy Amazon EC2 Instance Comparison, Retrieved From: 2015, <http://www.ec2instances.info/ >, Accessed on: Mar. 2015.

[31] AWS Blog Now available-New C4 Instances, Retrieved from.: 2015, <https://aws.amazon.com/blogs/aws/now-available-new-c4-instances/>, Accessed on: Mar. 2015.

[32] Amazon EC2 instance types, Retrieved from: 2015, <http://aws.amazon.com/ec2/instance-types/>, Accessed on: Mar. 2015.

[33] What is Amazon EC2?, Retrieved from: 2015, <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ >, Accessed on: Mar. 2015.

[34] M. R. Rahman, Y. Lu, and I. Gupta, "Risk aware resource allocation for Clouds," University of Illinois at Urbana-Champaign, Tech. Rep., 2011. [Online]. Available: http://hdl.handle.net/2142/25754

[35] K. Nordhausen, “The elements of statistical learning: data mining, inference, and prediction, second edition by trevor hastie, robert tibshirani, jerome friedman,” *Int. Statistical Rev.*, vol. 77, no. 3, pp. 482–482, Dec. 2009.

[36] D. G. Goldstein, R. P. McAfee, and S. Suri, “The wisdom of smaller, smarter crowds,” in *Proc. 15th ACM Conf. Economics Comput.*, 2014, pp. 471–488.

[37] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, “Ensemble approaches for regression,” *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–40, Nov. 2012.

[38] T. G. Dietterich and E. B. Kong, “Machine learning bias, statistical bias, and statistical variance of decision tree algorithms,” Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, 1995.

[39] Fortmann-Roe, Scott, “Understanding the bias-variance tradeoff,” 2012, http://scott.fortmann-roe.com/docs/BiasVariance.html

[40] C. John, “Neural network training,” U.S. Patent Application No. 10/629,821, May, 13, 2004.

[41] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[42] L. Breiman, “Bagging predictors,” *Mach. Learn.* vol. 24., no. 2, pp. 123–140, 1996.

[43] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, “Variable selection using random forests,” *Pattern Recognit. Lett.*, vol. 31, no. 14, pp. 2225–2236, Oct. 2010.

[44] Mean Consequential Error, Retrieved From: <https://www.kaggle.com/wiki/MeanConsequentialError>, Accessed on: Dec. 2015.

[45] AWS CLI Command Reference, Retrieved from:<http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-spot-price-history.html>, Accessed on: Mar. 2015.

Veena Khandelwal received the MTech degree in computer science & engineering from Rajasthan Technical University, in 2014. She is working towards the PhD degree in the Department of Computer Science & Engineering, Rajasthan Technical University, Kota, India. Her current research interests include cloud computing, cost optimization, data availability and data security.

Anand Kishore Chaturvedi received the bachelor’s degree from the Department of Mechanical Engineering, Rajasthan Technical University, Kota, in 1991, and the MTech degree in industrial engineering from IIT Delhi, in 2004 and the PhD degree from ABV Indian Institute of Information Technology & Management, Gwalior, in 2012. He is currently working as prinicpal in MLV Textile and Engineering College, Bhilwara affiliated to Rajasthan Technical University. His research interests include optimization, scheduling, grid and cloud computing.

Chandra Prakash Gupta received the bachelor’s degree with honors from the Department of Electrical Engineering, Malaviya National Institute of Technology, Jaipur, in 1987, the MTech degree in computer technology from IIT Delhi, in 1995, and the PhD degree from the University of Kota in Wireless Sensor Networks, in 2014. Currently, he is a professor of Computer Science & Engineering, Rajasthan Technical University, Kota. He has presented papers in various International Conferences and has approx. 20 publications in the field of Ad-hoc Networks, Data Security, Distributed Computing etc. He is the general chair for International Conference on Cyber Security 2017 as well as International Conference on Computers and Management 2017. He is also an editor for International Journal of Information and Computer Security and International Journal of Information Security and Privacy.