Improving AWS EC2 Spot Instance Price Prediction Accuracy using XGBoost

MSc Cloud Computing
Research Project

Ankit Dutta
Student ID: x20185502

School of Computing
National College of Ireland

Supervisor: Mr Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ankit Dutta
Student ID:	x20185502
Programme:	MSc Cloud Computing
Year:	2022
Module:	Research Project
Supervisor:	Mr Vikas Sahni
Submission Due Date:	15/08/2022
Project Title:	Improving AWS EC2 Spot Instance Price Prediction Accuracy using XGBoost
Word Count:	5148
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ankit Dutta
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improving AWS EC2 Spot Instance Price Prediction Accuracy using XGBoost

Ankit Dutta
x20185502

Abstract

Amazon Web Services offers one of the cheapest types of computing instances as spot instances which can be upto 90% cheaper that on-demand instances. This is especially beneficial to resource constrained projects. This allows users to test and develop using spot instances and finally deploying the service on on-demand or reserved instances. This allows for huge cost savings, thus making AWS spot instance price prediction with reduced error a necessity. In this evaluation 5 AWS regions were considered; namely, 'ap-south-1', 'eu-west-1', 'us-east-1', 'us-west-1', and 'us-west-2'. The metrics used for comparison were Mean Square Error (MSE), Coefficient of Determination (r2_score), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). After comparing these metrics for both the XGBoost and Random Forest models, it was observed that out of 20 total metrics, XGBoost had either comparable or better performance in 15 of them. The highest r2_score (0.6315) was achieved for the XGBoost model in the 'ap-south-1' region.

1 Introduction

According to the Gartner Cloud Computing report of 2022, about 69% of businesses worldwide have accelerated cloud adoption over the past 12 months.[1] Furthermore, there is expected growth in Cloud IT infrastructure from 41% to 63% in the next year and half. This is brought about by organisations recognising the value of cloud computing, as 60% of IT decision makers stand by their word that shifting to cloud has benefited their organisation's revenue and have made it more sustainable in the past year. In the past two years, cloud has grown at a much faster rate due to more and more people working from home. This has affected all domains of cloud, i.e. Infrastructure-as-a-Service (IaaS), Platform-as-a-service (PaaS) and Software-as-a-service (SaaS). Thus, it becomes important to provision cloud services from any domain at the cheapest possible price while maintaining Service Level Agreements (SLAs). To address this issue, the focus on compute facilities of the cloud. There are multiple offerings of the same by different providers, with Amazon Web Services (AWS) being the most popular. The compute service of AWS is called, Amazon Elastic Compute Cloud (EC2). This allows users to lease cloud compute services. Users can also choose what type of compute service they require, whether it is be a single core requirement or a workstation configuration. There also several different pricing models on-demand, reserved and spot. Spot Instances (SI) are the cheapest but have a downside that if the spot price is more than the bid price

[1] `https://www.gartner.com/en/information-technology/insights/cloud-strategy`

then the user stands to lose that instance. Even though SIs have such a downside, it is still on of the most widely used pricing models for reserving EC2 instances as it is about 90% cheaper than on-demand pricing.

SI price prediction has been done using various statistical and machine learning methods in previous scholarly works, with each trying to prove that it is a more accurate or efficient method. The aim of this research is to improve upon existing SI price prediction models. The main contribution of the research work is that of implement Extreme Gradient Boosting (XGBoost) for predicting SI prices in different AWS regions and observing whether this approach yields better results than existing scholarly work. To execute this, the project makes use of SI prices which is available directly from AWS.

The latest work with regards to time-series forecasting and AWS SI price forecasting using statistical and machine learning models has been done in Section 2. The methodology and design specification of the project has been discussed in Section 3 and Section 4. Finally, the implementation and evaluation of this research project has been done in Section 5 and Section 6.

2 Related Work

There have been several research publications that have studied the prediction of AWS SI pricing using various statistical and machine learning methods. Furthermore, in this review, previous works in the field of prediction and forecasting have been done so as to provide better alternatives to the already present methods of AWS SI price forecasting. It helps in evaluating more relevant metrics instead of metrics justly like accuracy, e.t.c.

2.1 Statistical Algorithms

The work presented by (Lucas-Simarro et al.; 2015) introduces a cloud broker which provides the best cloud provider for a given time period by developing a statistical model for it, and analysing dynamic pricing data of the respective cloud providers. The focus of this paper is on the scheduler (The cloud manager, The Virtual Machines (VM) manager, and the cloud scheduler, comprise the three modules of the broker), which is modelled using the historical pricing data of the cloud providers. This model can then use the data to provide the next hour forecasting price data and provide the best cloud provider option for that hour. This scheduler was designed keeping in mind the objective of keeping client investment to the minimum. The client can limit the transfer of active resources between the providers. Furthermore, the implementation can be done in both dynamic and static settings. The static scenario comes into play when future hardware requirements are known. In case of dynamic scenario, the broker switches to the cheapest provider. This implementation has seen that the performance of the statistical model is better when using several different types of instances.

2.2 Machine Learning Algorithms

Several different types of machine learning algorithms have been used for forecasting purposes and in specific forecasting AWS SI pricing.

Work done by (Larivière and Van den Poel; 2005) evaluates the evaluation capability model customer retention and profitability which uses Random Forests (RF) and Regression. It is compared and contrasted against other similar machine learning approaches and linear regression. It successfully identified different important features that have helped with predictions. Further evidence from different publications (Criminisi et al.; 2010), (Antipov and Pokryshevskaya; 2012), (Booth et al.; 2014) provide evidence for better performance than other models (machine learning or statistical), and also better stability while dealing with seasonality (Khaidem et al.; 2016). Work by (Khandelwal et al.; 2017) is one of the latest regarding predicting AWS Spot Price prediction. It successfully establishes RFF as the most accurate method for forecasting SI prices. The most predominant method was to perform one-day ahead forecasting. Mean Absolute percentage Error (MAPE) $\leq$ 10% for 66%-92% of the regions, Mean Consequential Percentage Error (MCPE) $\leq$ 15% for 35%-81% of the regions. In the case of one-week ahead predictions, MAPE $\leq$ 15% for 71%-96% of the regions.

Work done by (Oliveira and Torgo; 2015) recognises Bagged Decision Trees (BDTs) as being on the up-and-up, and is specifically aimed at diversity generation of features. From further work that was analysed, (Mendes-Moreira et al.; 2012) mention the three stages to ensemble learning; generation stage, pruning stage and the integration stage. (Yu et al.; 2015) provides evidence that Bagged Decision Trees have lower computational complexity and improve the prediction accuracy. Finally, work by (Nur et al.; 2011) sees BDTs being used in real life scenarios, which establishes BDT as a suitable option for ensemble based prediction.

Bagged Neural Networks (BNNs) was the first implementation, of bagging Neural Networks which helped reduce variation in prediction accuracy, and the reduction of load forecasting error (Khwaja et al.; 2015). Two studies were analysed, which had used BNNs for forecasting. The first one (Ghani; 2005), provides insurance prices to online sellers by predicting auction prices. The second one predicts electrical load. This method outperforms statistical approaches like autoregressive–moving-average (ARMA), and machine learning based approaches like bagged regression trees, single artificial neural-network (ANN), Support Vector Machines (SVM), similar day-based wavelet neural network (SIWNN). Of the above mentioned machine learning methods, RF have the least variation in prediction accuracy and lowest forecasting error. This is followed by BDT, and then BNN. Extending this study scope, to using just Neural Networks (NNs) for predicting SI prices is done in the study by (Wallace et al.; 2013), the average relative estimation does not surpass 4% and the number of outliers (i.e., those whose relative prediction error is greater than 10%) is not greater than 5% for the overall amount of the prediction results, the experimental modeling results on the Amazon EC2 spot instances showed high correlation accuracy of the proposed approach. This demonstrates that neural networks are effective at making predictions and are valuable to users who are placing bids on spot instance services. (Singh and Dutta; 2015) have suggested Gradient Descent Algorithm, an unique algorithm in the study to forecast spot instance pricing. The accuracy of the proposed algorithm has been carefully evaluated for both short-term (one-day forecasting) and long-term (one-week forecasting) predicting periods. The findings demonstrate that the suggested model can estimate spot prices up to five days in advance with a maximum error rate of 20% and one day in advance with an error rate of 9.4%.

Similar to the previous studies of predicting spot prices, the work by (Lancon et al.; 2019) uses long short-term memory (LSTM) NNs for the same. The study shows that the proposed LSTM model reduces MAPE by 95% on comparing it to the baseline ARIMA forecast. An advancement to this is the work by (Kong et al.; 2021), it used Gated Recurrent Units (GRU) for AWS SI price forecasting. On comparison to other approaches, it achieves a root mean square error (RMSE) of 1.58e-3.

Finally, to justify the use of Extreme Gradient Boosting (XGBoost), the study by (Zamani Joharestani et al.; 2019) is evaluated. This study evaluates PM 2.5 concentration levels using Tehran's air pollution data provided by the National Department of Environment and Municipality of Tehran city. It is already established that RFF is the best method for forecasting time-series data based on earlier scholarly work documented in this section. This particular study extrapolates the above mentioned dataset using both RFF and XGBoost models. The results show that XGBoost helps eliminate unimportant features and also outperforms RFF, while achieving a mean-squared error of 0.81, MAE = 9.98, and RMSE of 13.58.

In Table 1, the relative performance of machine learning algorithms based on the related works.

Table 1: Relative Performance of Machine Learning Models.

Algorithm	**Relative Performance**
Bagged Neural Networks	✓
Bagged Decision Trees	✓
RFF	✓✓
XGBoost (Proposed Method)	✓✓✓

3 Methodology

One of the most important things to do is to establish an appropriate methodology. This section is aimed at describing all the steps from data collection to the final results. It starts from data collection and processing to feature extraction and modelling as the main point of study is effectiveness of the XGBoost algorithm on the AWS SI price history dataset. The flow of processes is represented in Figure 1. The first step being data collection, the main attempt is to retrieve the AWS SI price history dataset. The price history dataset is available directly from [2]. More specifically in this project, the 'describe-spot-price-history' module of the AWS Command Line Interface (CLI) is used along with AWS 'boto3'[3]. The required data can be directly accessed from the AWS SDK with the appropriate AWS profile credentials applied in the folder directory of the local machine. Once the AWS profile is set up, the required data can be filtered on the basis of a few filters which are:

- **availability-zone:** The availability zone for which SI history prices are desires.
- **instance-type:** The type of deployed instance for which the price is desired. In this criteria, the types represent different computational abilities.
- **product-description:** This basically represents the operating software of the host instance, different host OS's have licensing costs.
- **spot-price:** The spot price for desired instance in USD($).

[2]`https://docs.amazonaws.cn/en_us/AWSEC2/latest/UserGuide/using-spot-instances-history.html`

[3]`https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-spot-price-history.html`

- **timestamp:** The time for which the spot price is valid. It follows the UTC format.

The spot price data that can be retrieved from AWS, is only for 90 days from the day of retrieval. Since the largest possible dataset is desired for a more accurate and reliable machine learning model, 90 days worth of SI price data is retrieved. The specific filters used for this research project are:

- **'m4.large'**
- **'Linux/UNIX (Amazon VPC)'**
- **'us-east-1'**

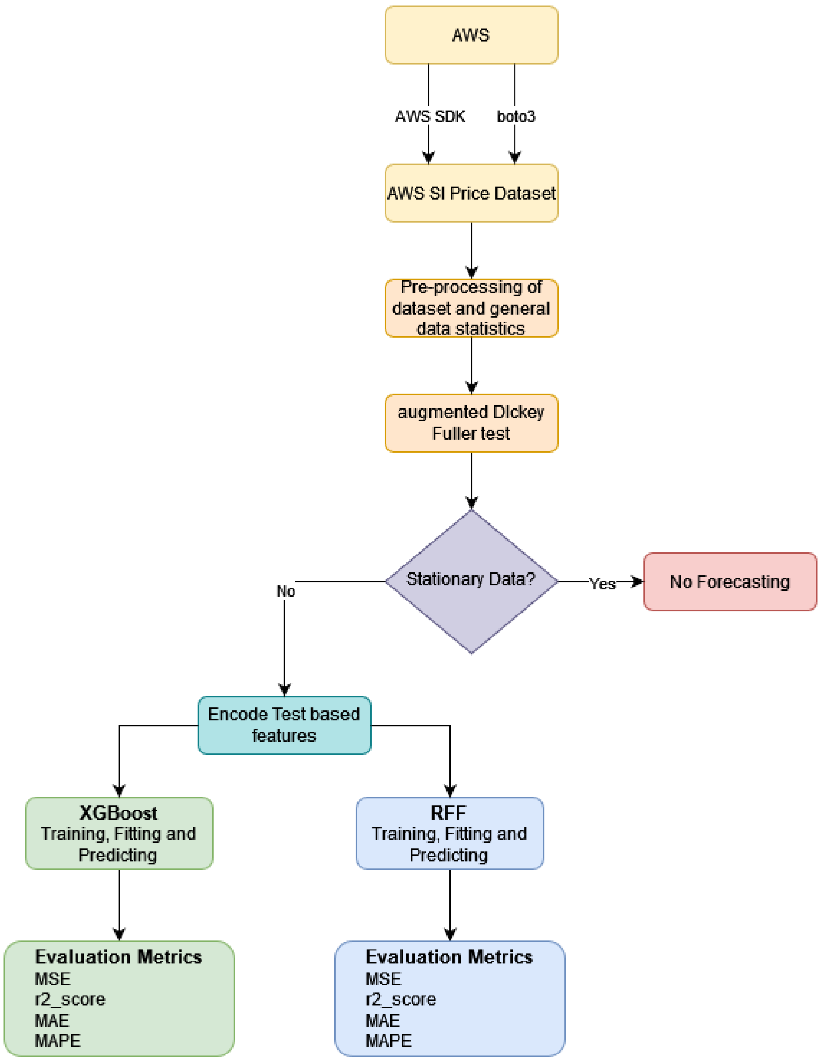

AWS
AWS SDK
boto3
AWS SI Price Dataset
Pre-processing of dataset and general data statistics
augmented DIckey Fuller test
Stationary Data?
Yes
No Forecasting
No
Encode Test based features
XGBoost Training, Fitting and Predicting
RFF Training, Fitting and Predicting
Evaluation Metrics MSE r2_score MAE MAPE
Evaluation Metrics MSE r2_score MAE MAPE

Figure 1: Flowchart explaining the Methodology of this project.

After retrieving the time-series pricing history data for AWS SI, it is important to analyse the data. In specific, it is important to analyse whether the time-series data is stationary or not. This is due to the fact that if the data is stationary, it has no significant dependence on time. This would then fail to prove the objective of the project. Once this has been established, it is then time to forecast AWS SI prices using the XGBoost algorithm.

To begin explaining the implementation of XGBoost, it is important to learn about Boosting. Boosting is a sequential strategy that functions like an ensemble method. A group of weak learners are combined, and the prediction accuracy increases. The model results are weighted depending on the results of the preceding instant 't-1' at any instant 't'. Accurately predicted outcomes get a lower weight than incorrectly categorized ones, which have a larger weight. Keep in mind that a poor learner is one that is just marginally superior than random guessing.Building a weak model, drawing conclusions about the relative relevance of different features, and then utilizing those observations to create a new, stronger model in an effort to minimize the misclassification error of the prior model, is the fundamental notion underlying boosting algorithms.To move onto XGBoost, one needs to be familiar with XGBoost's tree ensemble base learners.They're a collection of classification and regression trees (CART) make up the tree ensemble model. Trees are produced one after the other, and future iterations make an effort to lower the misclassification rate.

Before making the model using the XGBoost Algorithm, the dataset has to be processed for a proper model and a validation test set, for this the dataset is split into training and validation set. The dataset is split into a 70%-30% ratio, with 70% being the training set and the 30% being a test (validation set). Once this is done, the model can be successfully built. For this research project, the XGBoost algorithm, is the point of focus, with comparisons being made using models built using RFF, linear regressions and ARIMA models (models used in prior literature). After building the model with the necessary parameters, it needs to be evaluated. Certain evaluation metrics are used for this:

- Mean Squared Error:

$$\sum_{i=1}^{D}(x_i - y_i)^2$$

  Where x_i is actual value and y_i is predicted value (Bickel and Doksum; 2015).

- Coefficient of Determination (R^2 Score):

$$R^2 = 1 - \frac{RSS}{TSS}$$

  Where RSS is sum of squares of residuals and TSS is total sum of squares (Draper and Smith; 1998).

- Mean Absolute Error (MAE):

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|A_t - F_t|$$

  Where A_t is actual value and F_t is forecasted value (Willmott and Matsuura; 2005).

- Mean Absolute Percentage Error (MAPE):

$$\mathrm{MAPE} = \frac{1}{T}\sum_{t=1}^{T}100\left|\frac{A_t - F_t}{A_t}\right|$$

  Where A_t is actual value and F_t is forecasted value (Hyndman and Koehler; 2006).

These metrics have been used for evaluations in prior work referred to in this literature. Furthermore, we can expand the testing to different regions and instance types. Which helps in measuring the performance of the model in different real-life scenarios.

4 Design Specification

Machine learning is employed in this work to create the classification model. These models are able to generate predictions based on the dataset and learn from it. Additionally, supervised learning is the foundation of the classification model in this instance. Both input and output variables are employed in this machine learning approach, and the mapping function is used in the algorithm to map the input variables to the output variable. This data learning method is beneficial because it enables easy data classification and separation so that data may be filtered or separated as new data is received. As a result, data are used to develop an algorithm, which is then used to forecast future data. This idea forms the basis of every machine-learning categorization model. The three key ideas that underpin all of the findings are listed below. They are XGBoost, RFF, and ARIMA. XGBoost and RFF based learning is examined to provide a basic overview of the key concepts they include.

4.1 Boosting Trees (RFF)

Decision trees are algorithms that group instances according to the values of their features. A decision tree's nodes represent features in instances that are waiting to be categorized. To divide the training data, a feature is chosen, and that feature becomes the root node. Following a similar process, the tree begins to branch out and create sub-trees until the same class subsets are established. Diagonal portioning is one of decision trees' drawbacks. However, (Fu et al.; 2012) offered suggestions for additional properties that may be combined with operators like conjunction, negation, and dis-junction to create multivariate trees.

The most used algorithm for creating decision trees is C4.5 (Quinlan et al.; 1996). It selects which characteristic may be the separating feature based on the Information Gain. It creates a classification tree and takes symbolic and numeric inputs. In order to create subsets of the same class, it divides recursively. The sole drawback is that it performs badly when certain areas have low point densities and is allergic to diagonal partitioning. However, multivariate trees have recently been created to address the aforementioned issue (Brodley and Utgoff; 1995).

Numerous classification trees make up a random-forest classifier (Breiman; 2001). The k^{th} classification tree is a classifier that is indicated by an input vector that is unlabeled and an output vector that is created randomly by choosing random features from the training data for each node. The same distribution approach was used to construct a randomly generated vector of various categorization trees in the forest that are unrelated to one another. Each tree provides a forecast or vote for unlabeled data, so labeling is completed. Many other algorithms, like the J48 method and others, are available for usage.

4.2 XGBoost

The core of XGBoost is based on tree algorithms. The characteristic features or columns of the dataset are taken into consideration by the tree methods, and these features then serve as the internal or conditional node. The tree now divides into branches or edges in response to the state at the root node. The leaf node is the branch's end that does not create any more edges, and splitting is often done to arrive at a conclusion.

Additionally, XGBoost classifies the data in accordance with decision-tree techniques after applying them to a known dataset. Gradient-boosted trees serving as the main approach in XGBoost are based on supervised learning. In essence, supervised learning is a method where target values (s_i) are predicted using the input data, often the training data (p_i) with several features.

Based on training data (p_i), the mathematical algorithm (also known as the model) generates predictions (s_i). A linear model, for instance, bases its prediction on a collection of weighted input characteristics, such as $\hat{s}_i = \sum j\theta_j p_{ij}$[4]. It is necessary to learn the parameters from the data. Typically, θ is used to denote the parameters, and the number of parameters might vary depending on the dataset. Whether it be regression, classification, ranking, or another kind of problem, the predict value s_i aids in categorizing the issue at hand. The primary goal is to identify the suitable parameters from the training dataset. The performance of the model is originally described by an objective function. It must be noted that each model might vary based on the chosen parameter. Consider a dataset in which the characteristics "length" and "height" are present. Because of this, a variety of models may be created from the same information, depending on the parameters.

The objective function has two parts: a training loss and a regularization.

$$obj(\theta) = TL(\theta) + R(\theta) \tag{1}$$

The training loss is the first portion of the goal function. R stands for the regularization term, while TL stands for training loss. Simply said, the TL is a measurement of the model's predictive power. Regularization helps to prevent issues like over-stacking or over-fitting of data, which may result in a less accurate model, and keeps the model's complexity within desirable bounds. All of the trees created from the dataset are simply added together by XGBoost, who then optimizes the outcome. The model of tree ensembles is followed by Random Forest as well. As a result, it can be claimed that the boosted trees and random forest are similar in terms of their algorithmic structure but distinct in terms of it is trained. Both random trees and boosted trees should be compatible with a tree ensemble prediction service. This is supervised learning's key benefit. Optimizing the objective function is the simplest technique to achieve the primary goal of learning about the trees.

The issue here is how the trees are configured in respect of the input parameters. The trees' structure and the corresponding leaf scores must be determined in order to establish these parameters (generally represented by a function f_t). The simultaneous training of all the trees is not an easy process. At each stage, XGBoost adds a tree while attempting to optimize the learnt tree (training). The decision of which tree to add at each stage is one that occurs, and the solution is to add the tree that achieves the goal of maximizing our objective function. The new objective function normally includes up

[4] `https://xgboost.readthedocs.io/en/stable/tutorials/model.html`

to the second order and takes the shape of an expansion as stated by Taylor's theorem (let's assume at step t).

$$obj^{(t)} = \sum\nolimits_{i=1}^{n}[m_i f_t(p_i) + \frac{1}{2}c_i f_t^2(p_i)] + R(f_t) \tag{2}$$

In the Equation 2, the inputs are m_i & c_i. For the new tree that wishes to join the model, the outcome reflects the intended optimization. In order to handle loss functions like logistic regression, XGBoost does it in this manner. To continue, regularization is crucial in determining the complexity of the tree $R(f)$. Tree $f(p)$ may be more precisely defined as,

$$f_t(P) = w_{q(p)}, w \epsilon R^L, q : R^d \longrightarrow \{1, 2, 3, 4, ...L\} \tag{3}$$

In the equation above, function q assigns leaves to the relevant data points, and w represents a vector of leaf scores (same score for data points utilizing the same leaf). The number of leaves is given by L. Furthermore, the complexity in XGBOOST is given by,

$$R(f) = \alpha L + \frac{1}{2}\beta \sum_{j=1}^{L} w_j^2 \tag{4}$$

The new optimized objective function at step t, or the t^{th} tree, may be obtained by inserting the regularization equation into Equation 1. The resulting model is a representation of the newly revised tree model and an evaluation of the quality of the tree structure $q(p)$. Since it is impossible to concurrently compute all tree possibilities, the tree structure is determined by computing the regularization, leaf scores, and goal function at each level. As a leaf is divided into a left leaf and a right leaf, the gain is computed at each level. The gain is then calculated at the present leaf using the regularization obtained at any subsequent leaves that may be feasible. That branch is cut off if the benefit is less than the extra regularization value (concept also called pruning). This is how XGBoost classifies data and delves deeply into trees; as a result, accuracy and other parameters are computed.

5 Implementation

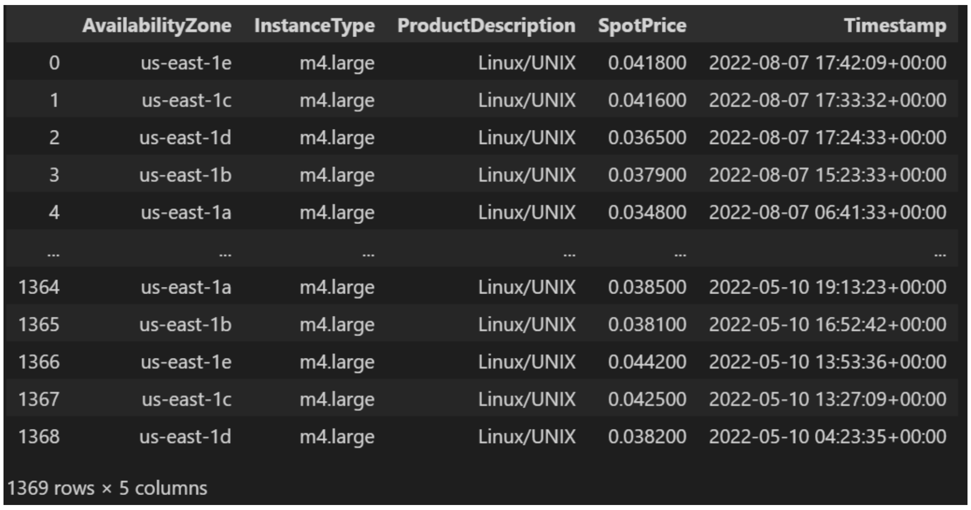
This project is based on 'Python' and 'Jupyter' notebooks which is executed on 'VSCode'. It makes use of the 'Sci-kit Learn' library for the machine learning tasks, 'pandas' and 'numpy' libraries for data manipulation, and the 'datetime' library to change the timestamp into a datetime object. In Table 2, the libraries and software used in the project are detailed.

Table 2: Software packages configuration details.

Software/Library	**Version**
VSCode	1.70.0
Python	3.10.5
Scikit-Learn	1.0.2
Pandas	1.4.3
Numpy	1.21.5

This project is split into two main parts. The first one being focused on dataset evaluation, and the second one being focused on model creation and model performance on dataset. For this purpose two Jupyter Notebooks have been created, ‘data_eval.ipynb’ and ’model_perf.ipynb’ respectively.

‘data_eval.ipynb’ is firstly tasked with importing the dataset from AWS. This can be seen in Figure 2.

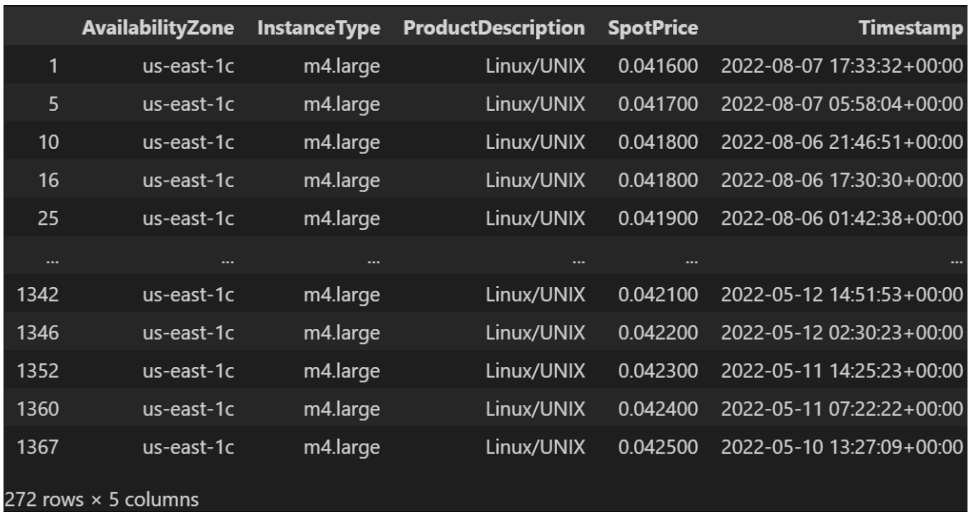


	AvailabilityZone	InstanceType	ProductDescription	SpotPrice	Timestamp
0	us-east-1e	m4.large	Linux/UNIX	0.041800	2022-08-07 17:42:09+00:00
1	us-east-1c	m4.large	Linux/UNIX	0.041600	2022-08-07 17:33:32+00:00
2	us-east-1d	m4.large	Linux/UNIX	0.036500	2022-08-07 17:24:33+00:00
3	us-east-1b	m4.large	Linux/UNIX	0.037900	2022-08-07 15:23:33+00:00
4	us-east-1a	m4.large	Linux/UNIX	0.034800	2022-08-07 06:41:33+00:00
...	...	...	...	...	...
1364	us-east-1a	m4.large	Linux/UNIX	0.038500	2022-05-10 19:13:23+00:00
1365	us-east-1b	m4.large	Linux/UNIX	0.038100	2022-05-10 16:52:42+00:00
1366	us-east-1e	m4.large	Linux/UNIX	0.044200	2022-05-10 13:53:36+00:00
1367	us-east-1c	m4.large	Linux/UNIX	0.042500	2022-05-10 13:27:09+00:00
1368	us-east-1d	m4.large	Linux/UNIX	0.038200	2022-05-10 04:23:35+00:00

1369 rows × 5 columns

Figure 2: Dataset imported as a DataFrame in the Jupyter Notebook

It then performs analysis on the data by specifying the ‘AvailabilityZone’ and ‘InstanceType’. The shortlisted data is shown in Figure 3.

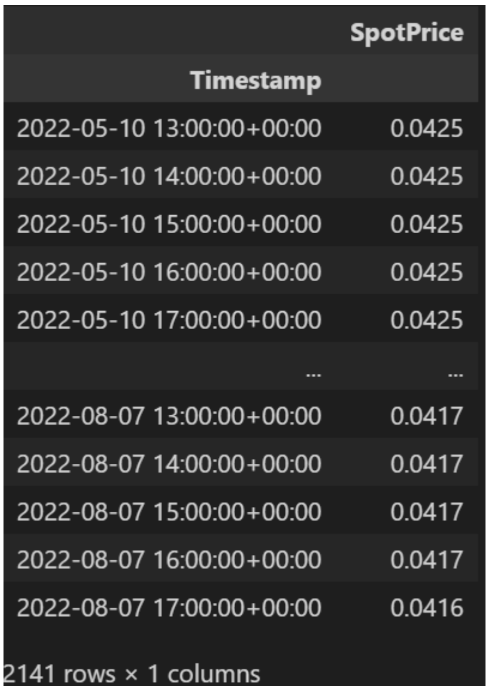


	AvailabilityZone	InstanceType	ProductDescription	SpotPrice	Timestamp
1	us-east-1c	m4.large	Linux/UNIX	0.041600	2022-08-07 17:33:32+00:00
5	us-east-1c	m4.large	Linux/UNIX	0.041700	2022-08-07 05:58:04+00:00
10	us-east-1c	m4.large	Linux/UNIX	0.041800	2022-08-06 21:46:51+00:00
16	us-east-1c	m4.large	Linux/UNIX	0.041800	2022-08-06 17:30:30+00:00
25	us-east-1c	m4.large	Linux/UNIX	0.041900	2022-08-06 01:42:38+00:00
...	...	...	...	...	...
1342	us-east-1c	m4.large	Linux/UNIX	0.042100	2022-05-12 14:51:53+00:00
1346	us-east-1c	m4.large	Linux/UNIX	0.042200	2022-05-12 02:30:23+00:00
1352	us-east-1c	m4.large	Linux/UNIX	0.042300	2022-05-11 14:25:23+00:00
1360	us-east-1c	m4.large	Linux/UNIX	0.042400	2022-05-11 07:22:22+00:00
1367	us-east-1c	m4.large	Linux/UNIX	0.042500	2022-05-10 13:27:09+00:00

272 rows × 5 columns

Figure 3: Filtered Dataset on the basis of ‘AvailibiltyZone’ and ‘InstanceType’

The dataset is further manipulated by indexing on the basis of hour. One issue with this is the AWS SI price is only updated once there is a change, so there are a lot of columns which have ‘NaN’. This is mitigated by using the forward fill method. The final data can be seen in Figure 4.



	SpotPrice
Timestamp	
2022-05-10 13:00:00+00:00	0.0425
2022-05-10 14:00:00+00:00	0.0425
2022-05-10 15:00:00+00:00	0.0425
2022-05-10 16:00:00+00:00	0.0425
2022-05-10 17:00:00+00:00	0.0425
...	...
2022-08-07 13:00:00+00:00	0.0417
2022-08-07 14:00:00+00:00	0.0417
2022-08-07 15:00:00+00:00	0.0417
2022-08-07 16:00:00+00:00	0.0417
2022-08-07 17:00:00+00:00	0.0416

2141 rows × 1 columns

Figure 4: Dataset with 'NaN' replaced using forward fill.

This final version of the manipulated dataset is then plotted thus representing the hourly price plot for 90 days of data. Some of its statistical data like rolling mean, rolling standard deviation have been plotted as well. These can be seen in Figures 5 and 6.

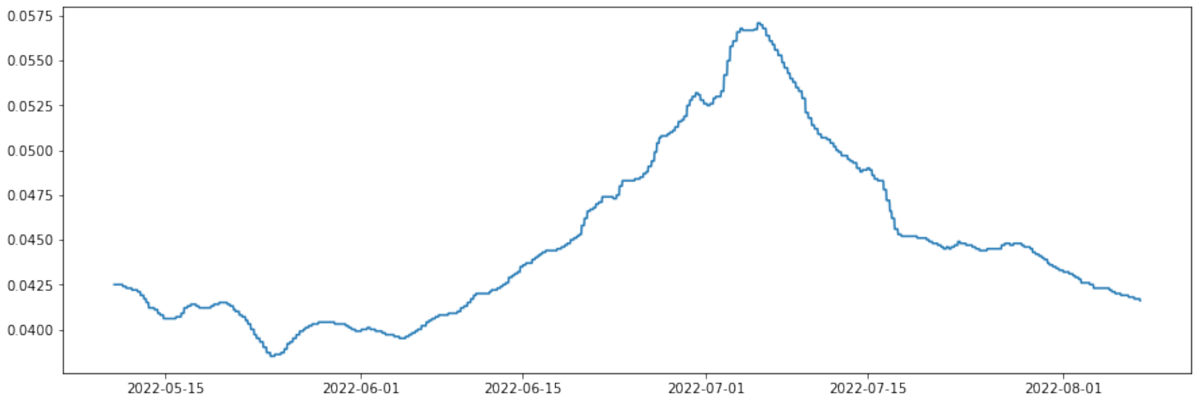


Figure 5: Hourly plots for AWS SI data.

Finally, the stationarity of data is tested because if the data is stationary then there is no point in forecasting future prices.

In the **'model_perf.ipynb'** notebook, the only task is that of model performance. Based on prior related works reviewed in this project, It has been established that RFF and XGBoost are one of the best methods for forecasting time-series data. Since, it is to be proven that XGBoost is a better method then it has to be head-to-head with RFF. For a detailed evaluation. The AWS SI pricing data is collected from five different regions:

- ap-south-1.

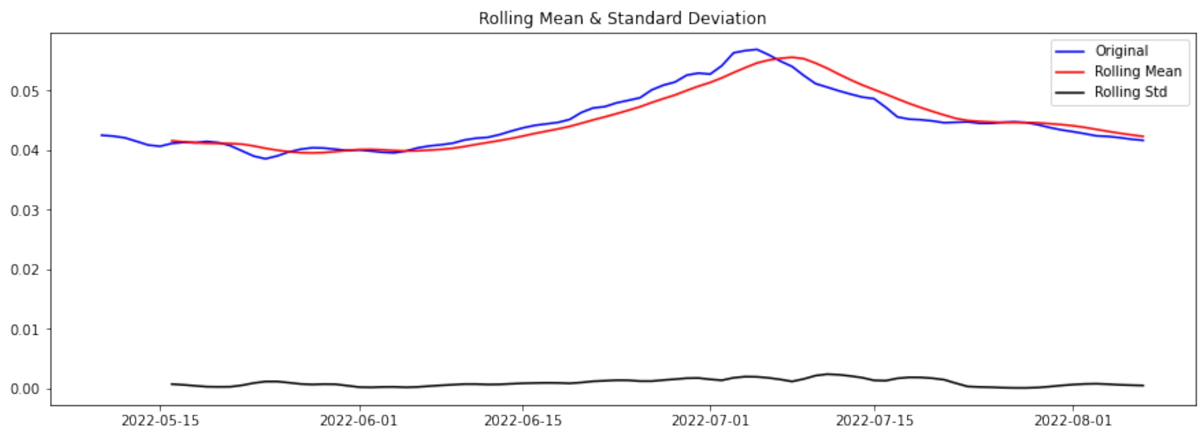


Figure 6: Rolling Mean and Rolling Standard Deviation of data.

- eu-west-1.
- us-east-1.
- us-west-1.
- us-west-2.

The AWS SI price data collected from these regions is used for creating RFF and XGBoost models. Finally, once these models have been made, a few evaluation metrics are also calculated for the same for comparison purposes:

- Mean Squared Error.
- Coefficient of Determination.
- Mean Absolute Error.
- Mean Absolute Percentage Error.

These metrics are then evaluated to establish which method of forecasting is less likely to produce errors.

6 Evaluation

The first part of the evaluation is to prove that the AWS SI pricing data is not stationary. To check for stationarity, the 'adfuller()' function is implemented. This function is an augmented Dickey-Fuller test (ADF), used in statistics and economics, examines the possibility that a unit root exists in a time series sample. Depending on the test version being utilized, the alternative theory may vary, but it is often stationarity or trend-stationarity. It is a supplement to the Dickey-Fuller test (Dickey and Fuller; 1979) for a more extensive and intricate collection of time series models. The test's ADF statistic is a negative value. The work by Mushtaq (2011) shows that the greater the rejection of the unit root hypothesis is, at least at some degree of confidence, the higher the negative

Table 3: Evaluation metrics of the ADF test.

Metric	Value
Test Statistic	-1.806237
p-value	0.377370
#Lags Used	5.000000
Number of Observations Used	84.000000
Critical Value (1%)	-3.510712
Critical Value (5%)	-2.896616
Critical Value (10%)	-2.585482

value it is. The hourly Spot Price is passed onto the ‘adfuller()’ function, which then provides data statistics seen in Table 3.

Since the ‘Test Statistic’ metric’s value is greater than the critical values, the null hypothesis is accepted and the data is considered non-stationary. This means that the AWS SI pricing data is not stationary and can be used for forecasting. The next step is to evaluate the XGBoost model against the RFF model. For this purpose, models for both are created for the ‘m4.xlarge’ type instance for five AWS regions; ‘ap-south-1’, ‘eu-west-1’, ‘us-east-1’, ‘us-west-1’, and ‘us-west-2’. The model performance of each region is discussed in the subsections below.

6.1 ap-south-1

The RFF model and XGBoost model performance statistics for the ‘ap-south-1’ region are given in Table 4.

Table 4: Evaluation Metrics for RFF and XGBoost Model for ’ap-south-1’ region

Evaluation Metrics	RFF	XGBoost
MSE	1.8722×10^{-5}	1.6322×10^{-5}
r2_score	0.6289	0.6315
MAE	0.0031	0.0027
MAPE	0.0345	0.0308

From Table 4; it is inferred that the XGBoost model has better MAE and MAPE, and equivalent (within margin of error) r2_score than the RFF model.

6.2 eu-west-1

The RFF model and XGBoost model performance statistics for the ‘eu-west-1’ region are given in Table 5.

From Table 5; it is inferred that the XGBoost model has better r2_score, MAE and MAPE, and equivalent (within margin of error) MSE than the RFF model.

6.3 us-east-1

The RFF model and XGBoost model performance statistics for the ‘us-east-1’ region are given in Table 6.

Table 5: Evaluation Metrics for RFF and XGBoost Model for 'eu-west-1' region

Evaluation Metrics	**RFF**	**XGBoost**
MSE	8.2157×10^{-6}	8.0334×10^{-6}
r2_score	-0.3155	-0.0344
MAE	0.0019	0.0017
MAPE	0.0288	0.0258

Table 6: Evaluation Metrics for RFF and XGBoost Model for 'us-east-1' region

Evaluation Metrics	**RFF**	**XGBoost**
MSE	5.7867×10^{-5}	5.0131×10^{-5}
r2_score	0.0518	-0.0028
MAE	0.0044	0.0044
MAPE	0.0469	0.0477

From Table 6; it is inferred that the XGBoost model has equivalent (within margin of error) MAE and MAPE than the RFF model.

6.4 us-west-1

The RFF model and XGBoost model performance statistics for the 'us-west-1' region are given in Table 7.

Table 7: Evaluation Metrics for RFF and XGBoost Model for 'us-west-1' region

Evaluation Metrics	**RFF**	**XGBoost**
MSE	8.2157×10^{-6}	8.2769×10^{-6}
r2_score	-0.3155	0.0891
MAE	0.0019	0.0019
MAPE	0.0288	0.0284

From Table 7; it is inferred that the XGBoost model has a better r2_score and MAPE, and equivalent (within margin of error) MSE and MAE than the RFF model.

6.5 us-west-2

The RFF model and XGBoost model performance statistics for the 'us-west-2' region are given in Table 8.

From Table 8; it is inferred that the XGBoost model has equivalent (within margin of error) MAE and MAPE than the RFF model.

6.6 Discussion

From subsections 6.1, 6.2, 6.3, 6.4, and 6.5 it is seen that out of 20 error measuring statistics, 15 of them are either equivalent (7) or better (8) for XGBoost. This shows that the XGBoost model outperforms RFF. The XGBoost model generates a feature importance graph. In all five cases represented in subsections 6.1, 6.2, 6.3, 6.4, and 6.5,

Table 8: Evaluation Metrics for RFF and XGBoost Model for 'us-west-2' region

Evaluation Metrics	**RFF**	**XGBoost**
MSE	2.053×10^{-4}	2.0105×10^{-4}
r2_score	-1.4931	-1.6524
MAE	0.0114	0.0115
MAPE	0.1223	0.1230

the F-score of the 'day' parameter is always the highest. In Table 9, the combined report for the evaluation metrics of both RFF and XGBOOST models is observed.

Table 9: Combined report for the evaluation metrics of both RFF and XGBOOST models.

Evaluation Metrics	**Regions**									
	'ap-south-1'		**'eu-west-1'**		**'us-east-1'**		**'us-west-1'**		**'us-west-2'**	
	RFF	XGBoost	RFF	XGBoost	RFF	XGBoost	RFF	XGBoost	RFF	XGBoost
MSE	1.8722×10^{-5}	1.6322×10^{-5}	8.2157×10^{-6}	8.0334×10^{-6}	5.7867×10^{-5}	5.0131×10^{-5}	8.2157×10^{-6}	8.2769×10^{-6}	2.053×10^{-4}	2.0105×10^{-4}
r2_score	0.6289	0.6315	-0.3155	-0.0344	0.0518	-0.0028	-0.3155	0.0891	-1.4931	-1.6524
MAE	0.0031	0.0027	0.0019	0.0017	0.0044	0.0044	0.0019	0.0019	0.0114	0.0115
MAPE	0.0345	0.0308	0.0288	0.0258	0.0469	0.0477	0.0288	0.0284	0.1223	0.1230

Based on evidence from Zamani Joharestani et al. (2019), it has been successfully corroborated that XGBoost has a better performance on time-series data than RFF. Two major observations that has been made are, that the performance of the models suffered due to small datasets. This could be easily rectified by gathering AWS SI prices for more than the 90 days provided by AWS. Also, for all regions except 'ap-south-1' have very low (and in some cases negative) which means that a very small percentage of inputs can explain the observed variation. The highest r2_score (0.6315) was achieved for the XGBoost model in the 'ap-south-1' region.

7 Conclusion and Future Work

The main objective of this project was to establish XGBoost as a superior model for forecasting AWS SI price data. In Section 6, it has been clearly established that XGBoost is a better model that RFF which in turn has outperformed other ensemble and statistical models. Thus the research question for this project has been successfully answered. The main findings of the project are that the AWS SI price dataset is not stationary thus making is suitable for price forecasting. Furthermore, upon modeling this data using XGBoost and RFF it was observed that for most evaluation metrics the XGBoost model outperformed its RFF counterpart. This model also generated a feature importance graph, which on all the five different scenarios of evaluation showed that the 'day' feature has the highest importance. This shows that AWS SI prices doesn't fluctuate much everyday, and that the price changes value gradually.

One major limitation of this project is the inability to use in mission critical applications as the r2_scores on all the regions except 'ap-south-1' has very low values (in some cases negative) which means that a very small percentage of inputs can explain the observed variation. This research project benifits any user planning on using AWS SI for their projects / system implementation. This significantly reduces costs as SI instances are upto 90% cheaper than on-demand instances[5]. Services that offer Distributed databases benefit greatly from SIs. Distributed databases that have the ability to save data

[5]https://aws.amazon.com/ec2/spot/pricing/

even after instance reboots may be supported by Spot Instances without any problems. Examples include Mongo, Elasticsearch, and Cassandra, all of which are fairly adept at picking up where they left off after an interruption. In addition to this machine learning also is a good domain to use SI instances in as they're cheaper than on-demand instances. Machine learning models, if interrupted during creation due to loss of SI can just restart the process again. Since, most machine learning applications are in the education domain and not mission critical, SIs should be preferred.

Any future work in this domain should aim at studying the computational complexity of machine learning models v/s gains in SI price predicting. Recent works have seen minimal increments in error reduction while forecasting AWS SI price data. Hence, such a study which would provide an idea as to whether a slight reduction in forecasting error is worth the added computational complexity.

References

Antipov, E. A. and Pokryshevskaya, E. B. (2012). Mass appraisal of residential apartments: An application of random forest for valuation and a cart-based approach for model diagnostics, *Expert Systems with Applications* **39**(2): 1772–1778.
URL: *https://www.sciencedirect.com/science/article/pii/S0957417411011894*

Bickel, P. J. and Doksum, K. A. (2015). *Mathematical statistics: basic ideas and selected topics, volumes I-II package*, Chapman and Hall/CRC.

Booth, A., Gerding, E. and McGroarty, F. (2014). Automated trading with performance weighted random forests and seasonality, *Expert Systems with Applications* **41**(8): 3651–3661.

Breiman, L. (2001). Random forests, *Machine learning* **45**(1): 5–32.

Brodley, C. E. and Utgoff, P. E. (1995). Multivariate decision trees, *Machine learning* **19**(1): 45–77.

Criminisi, A., Shotton, J., Robertson, D. and Konukoglu, E. (2010). Regression forests for efficient anatomy detection and localization in ct studies, *International MICCAI workshop on medical computer vision*, Springer, pp. 106–117.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root, *Journal of the American statistical association* **74**(366a): 427–431.

Draper, N. R. and Smith, H. (1998). *Applied regression analysis*, Vol. 326, John Wiley & Sons.

Fu, Z., Papatriantafilou, M. and Tsigas, P. (2012). Mitigating distributed denial of service attacks in multiparty applications in the presence of clock drifts, *IEEE transactions on Dependable and Secure Computing* **9**(3): 401–413.

Ghani, R. (2005). Price prediction and insurance for online auctions, *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 411–418.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy, *International journal of forecasting* **22**(4): 679–688.

Khaidem, L., Saha, S. and Dey, S. R. (2016). Predicting the direction of stock market prices using random forest, *arXiv preprint arXiv:1605.00003* .

Khandelwal, V., Chaturvedi, A. K. and Gupta, C. P. (2017). Amazon ec2 spot price prediction using regression random forests, *IEEE Transactions on Cloud Computing* **8**(1): 59–72.

Khwaja, A., Naeem, M., Anpalagan, A., Venetsanopoulos, A. and Venkatesh, B. (2015). Improved short-term load forecasting using bagged neural networks, *Electric Power Systems Research* **125**: 109–115.

Kong, D., Liu, S. and Pan, L. (2021). Amazon spot instance price prediction with gru network, *2021 IEEE 24th international conference on computer supported cooperative work in design (CSCWD)*, IEEE, pp. 31–36.

Lancon, J., Kunwar, Y., Stroud, D., McGee, M. and Slater, R. (2019). Aws ec2 instance spot price forecasting using lstm networks, *SMU Data Science Review* **2**(2): 8.

Larivière, B. and Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques, *Expert Systems with Applications* **29**(2): 472–484.
URL: *https://www.sciencedirect.com/science/article/pii/S0957417405000965*

Lucas-Simarro, J. L., Moreno-Vozmediano, R., Montero, R. S. and Llorente, I. M. (2015). Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios, *Concurrency and Computation: Practice and Experience* **27**(9): 2260–2277.

Mendes-Moreira, J., Soares, C., Jorge, A. M. and Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey, *Acm computing surveys (csur)* **45**(1): 1–40.

Mushtaq, R. (2011). Augmented dickey fuller test.

Nur, N., Jahncke, J., Herzog, M. P., Howar, J., Hyrenbach, K. D., Zamon, J. E., Ainley, D. G., Wiens, J. A., Morgan, K., Ballance, L. T. et al. (2011). Where the wild things are: predicting hotspots of seabird aggregations in the california current system, *Ecological Applications* **21**(6): 2241–2257.

Oliveira, M. and Torgo, L. (2015). Ensembles for time series forecasting, *Asian Conference on Machine Learning*, PMLR, pp. 360–370.

Quinlan, J. R. et al. (1996). Bagging, boosting, and c4. 5, *Aaai/Iaai, vol. 1*, pp. 725–730.

Singh, V. K. and Dutta, K. (2015). Dynamic price prediction for amazon spot instances, *2015 48th Hawaii International Conference on System Sciences*, IEEE, pp. 1513–1520.

Wallace, R. M., Turchenko, V., Sheikhalishahi, M., Turchenko, I., Shults, V., Vazquez-Poletti, J. L. and Grandinetti, L. (2013). Applications of neural-based spot market prediction for cloud computing, *2013 IEEE 7th international conference on intelligent data acquisition and advanced computing systems (IDAACS)*, Vol. 2, IEEE, pp. 710–716.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance, *Climate research* **30**(1): 79–82.

Yu, L., Wang, Z. and Tang, L. (2015). A decomposition–ensemble model with data-characteristic-driven reconstruction for crude oil price forecasting, *Applied Energy* **156**: 251–267.

Zamani Joharestani, M., Cao, C., Ni, X., Bashir, B. and Talebiesfandarani, S. (2019). Pm2. 5 prediction based on random forest, xgboost, and deep learning using multi-source remote sensing data, *Atmosphere* **10**(7): 373.