Evaluating Amazon EC2 Spot Price Prediction Models Using Regression Error Characteristic Curve

Batool Alkaddah

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

March 2023

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Batool Alkaddah**

Entitled: **Evaluating Amazon EC2 Spot Price Prediction Models Using Regression Error Characteristic Curve**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Wahab Hamou-Lhadj

_____ External Examiner
Dr. Nizar Bouguila

_____ Examiner
Dr. Wahab Hamou-Lhadj

_____ Supervisor
Dr. Anjali Agarwal

Approved by _____
Dr. Yousef Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2023

_____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Evaluating Amazon EC2 Spot Price Prediction Models Using Regression Error Characteristic Curve

Batool Alkaddah

Amazon EC2 offers inactive virtual machines (VM) as spot instances at up to 90% discount. In return, the least expensive option requires the customers' usage to be tolerated with a low availability level agreement. Thus, many studies proposed forecasting and prediction mechanisms to assess finding the best set of maximum prices.

In this research, we study the model's efficiency in predicting spot EC2 prices by assessing the performance of forecasting algorithms: RFR, XGBoost, k-NNR, and SVR. We evaluate the models using six metrics, including MAPE, RMSE, MAE, and MSE, commonly used in related work, as well as the Regression Error Characteristics (REC) curve and the Area under the curve (AUC-REC). Our experiments consider dataset time per year, training window (1-day, 1-week, and 1-month ahead), and instance location.

The REC curve and AUC-REC are superior performance measurements for evaluating models over different accuracy-loss thresholds. Our findings suggest that the cross-validation technique is unnecessary to improve the models' accuracy, except for the SVR model. Our study highlights the limitations of using threshold-based metrics, which can be misleading, and the importance of using representative metrics

such as AUC-REC to evaluate machine learning models.

Our study has limitations, including the choice of algorithms, which may impact the results. Additionally, our experiments are limited to AWS cloud services, and our results may not be generalizable to other cloud providers. In future work, we plan to evaluate other forecasting methods, including deep learning and statistical methods, and investigate the results of other regions and training windows.

Acknowledgments

I would like to express my genuine appreciation to my advisor, Prof. Anjali Agarwal, for her assistance, direction, and motivation throughout this research dissertation. Her knowledge, input, and understanding have been priceless in shaping the quality and direction of this thesis.

I also want to thank my mother and father, Dr. Soheil Alkaddah and Ms. Boshra Alkaddah, for their infinite care, assistance, and drive. Their unwavering trust in me has been a constant source of inspiration and endurance.

To my beloved husband, Mohammed Bin Shehab, I am grateful for your constant assistance and empathy. Your affection, inspiration, and tolerance have been my anchor during this challenging journey. I learned a lot from your valuable skills and knowledge in this field.

Also, I would like to recognize my brothers and sister, Mohammed, Ismael, Abrar, and Modar, for their care, assistance, and inspiration throughout my academic journey. Their steadfast belief in me has been a constant source of motivation and gratitude.

Lastly, I want to thank my companions in Montreal and Kuwait for their camaraderie, assistance, and constant friendship.

Thank you all for your constant support, inspiration, and care. I am truly fortunate to have you in my life.

Contents

Acronyms

ADF Augmented Dickey-Fuller. 28

AUC-REC Area Under the REC Curve. 26, 27, 58, 60, 82, 90, 96

AWS Amazon Web Services. 1

CDF cumulative distribution function. 26

DT Decision Tree. 32

IaaS Infrastructure as a Service. 1, 10, 11

k-NNR K Nearest Neighbor Regressor. 31, 32

KPSS Kwiatkowski–Phillips–Schmidt–Shin. 28

LR Logistic Regression. 32

LSTM Long Short-Term Memory. 32

MAE Mean Absolute Error. 27, 58

MAPE Mean Absolute Percentage Error. 27

ML Machine Learning. 25, 27, 31, 45, 60

MSE Mean Squared Error. 27, 30, 58, 67

REC Curve Regression Error Characteristics Curve. 26, 27, 58, 60, 66, 67, 90, 96

RFR Random Forest Regressor. 27, 30–32

RMSE Root Mean Squared Error. 27, 58

SI Spot Instance. 14, 57, 58, 61, 89

SVR Support Vector Regressor. 27, 31, 32

VMs Virtual Machines. 1

XGBoost eXtreme Gradient Boosting. 8, 27, 31, 36

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

In recent years, there has been a significant increase in demand for cloud computing services from a wide range of clients, including large corporations, university research groups, and individuals. Many cloud providers, such as Amazon Web Services (AWS), offer Infrastructure as a Service (IaaS), a cloud computing service that provides virtualized computing resources, such as servers, storage, and network capabilities, over the internet. This service allows customers to access and use these resources on demand without investing in or maintaining their physical infrastructure. Cloud companies typically use a combination of different pricing models to price their IaaS resources, including fixed or dynamic pricing plans explained in more detail in section 2.1. Leasing prices vary depending on the plan type, and different payment options are available to meet customers' needs.

Our research focuses on AWS's dynamic pricing scheme, which offers a special class of Virtual Machines (VMs) called spot instances at significantly reduced rates. This pricing option serves as a solution for cloud providers with idle resources by leasing out short-term instances at a lower cost, providing budget-conscious users

with a more affordable alternative.

1.1.1 Spot Instances (SI)

Spot or Preemptible instances are virtual machines (VMs) offered at lower pricing options, with prices generally ranging from 50-90% less than the price of on-demand VMs [2]. The concept of spot instances originated from the idea of using spare or unused cloud computing capacity. Spot instances offer significant price reductions of up to 90% without requiring any long-term commitment from the user or the cloud provider. This pricing model allows spot users to benefit from a substantial reduction in hourly costs compared to on-demand standard prices [3].

Empirical evidence suggests that, on average, spot instances are more cost-effective than on-demand instances for workloads lasting less than approximately two weeks [4]. Spot instances are highly attractive to large-scale applications, and the market for spot instances is the foundation of the dynamic pricing scheme. This VMs model serves as a solution for cloud providers with idle resources by leasing out short-term instances at a lower cost, providing budget-conscious users with a more affordable alternative [2, 5].

However, the spot model also comes with its own set of challenges. This pricing model offers a solution for cloud providers with idle resources, enabling them to lease out short-term instances at a lower cost and provide budget-conscious users with a more affordable option [2]. These lower prices result in significant disruptions for cloud users who rely on spot instances. To address the issue of execution interruptions, cloud users would greatly benefit from a reliable method for predicting spot prices, enabling them to manage the demand for their instances more effectively [3]. By predicting spot prices, users can anticipate supply and demand changes, reducing the mismatch between spot virtual machine supply and user usage and minimizing

interruptions in instance availability.

1.1.2 Forecasting and Prediction Algorithms

The field of regression analysis has come a long way since its inception. Initially, regression models relied solely on statistical techniques, such as linear regression [6], to forecast outcomes. However, with the advent of machine learning algorithms, regression models have become increasingly sophisticated, offering greater accuracy and improved performance. Predictive machine learning models, such as Random Forest Regressor (RFR), k-Nearest Neighbor Regressor (k-NNR), and XGBoost, have become widely used in regression problems and have proven to be highly effective in accurately forecasting outcomes [7–10]. These models leverage complex algorithms and data-driven methods to analyze large amounts of data and generate more accurate predictions than traditional statistical models. By incorporating the latest machine learning techniques and advanced algorithms, these models have helped practitioners in the regression field significantly improve their predictions and better understand the underlying relationships between variables [11].

While developing spot price prediction models has become increasingly applicable in recent years, the field remains complex and challenging. An inaccurate prediction could result in significant financial loss for the user, with a few cents difference sometimes being the deciding factor between success and failure [3]. To address these challenges, recent studies have proposed a range of different prediction models, utilizing a variety of forecasting techniques, including statistical forecasting methods [12, 13], deep learning models [14–16], machine learning algorithms [17–19], and combinations thereof.

1.1.3 Performance Metrics in Spot Models

We compare the performance of different classic machine learning algorithms in forecasting spot instance prices by generating optimal regression models through a series of steps. First, opt for the ultimate ML algorithms to build regression models used in the field of SI price prediction, Random Forest Regressor (RFR) [7, 20], k-Nearest Neighbor Regressor (k-NNR) [8] from related work compared to XGBoost [10]. Second, validating the models using the cross-validation technique, followed by tests that focus on tuning the hyper-parameters λ associated with the algorithms for each training iteration based on window size, instance region, and year of timestamp data. The final stage is measuring model performance using the most commonly used evaluation metrics in the field against the Regression Error Characteristic Curve (RECC) and Area Under the Curve (AUC-RECC) [21].

While not yet widespread in the field, using the RECC and AUC-RECC as visual assessment metrics can yield results comparable to or surpass those commonly used metrics (e.g., MAE and MAPE). Noteworthy that RECC is equivalent to the Receiver Operating Characteristic Curve (ROCC), which is a standard tool for evaluating overall model performance in classification problems [21, 22].

Evaluating the performance of these models is no easy task, as different regression evaluation metrics are often used, ranging from scale-free percentage measures, such as the Mean Absolute Percentage Error, to scale-dependent metrics, such as Root Mean Squared Error, Mean Absolute Error, and Mean Squared Error [23–25]. Determining the most appropriate metric for a given task and data set can be difficult, with the choice of metric often dependent on the data's shape and the forecasting task's specifics [26].

1.2 Motivation

The motivation for this thesis comes from the significant surge in demand for cloud computing services in recent years. With the rise of cloud computing, companies, and individuals increasingly rely on virtual computing resources for their market services, resulting in an increased demand for cloud computing solutions. However, the dynamic pricing model used by cloud providers, such as Amazon Web Services (AWS), comes with challenges, including instance availability and execution interruptions.

The pricing for spot instances is dynamic and fluctuates significantly over time, leading to multiple interruptions in instance availability. Likewise, it results in significant financial loss for users who rely on these instances for their computing needs. While the demand for cloud computing resources continues to grow, predicting the fluctuating spot prices of these resources remains a critical problem. Predicting Spot Instances (SI) prices using proper predictive models is a promising solution, leading to the more efficient and cost-effective use of cloud computing resources. Utilizing the best accuracy metrics to achieve optimal prediction results is important. The performance metric should be unbiased and suitable for all data groups of distribution.

In this thesis, we aim to build and evaluate the performance of various classic machine learning models in predicting EC2 spot prices. The study will focus on three significant contributions, including dataset time's impact and training window size's effect on the model's effectiveness. More importantly, we focus on evaluating the performance of various models using a new visualizing measurement called the Regression Error Characteristic Curve (RECC). Also, the Area under the Curve (AUC-REC) is used as a single-point accuracy metric.

We aim to provide a better understanding of the factors that affect the accuracy of machine learning models in predicting spot prices and to contribute to the ongoing efforts to optimize resource allocation and reduce costs in cloud computing.

1.3 Problem Statements

The main objective of the thesis is to evaluate the effectiveness of machine learning models in predicting the price of Spot Instances (SI) in cloud computing. The problem statement of the thesis is centered around this objective, and the research questions aim to address the key challenges to evaluating the accuracy and efficiency of these models. Ultimately, the thesis seeks to contribute to developing more effective and efficient machine learning models for price prediction of Spot Instances in cloud computing.

This thesis firstly aims to dive into the issue of visualizing the performance of SI-trained models across different data distributions. While current assessment metrics, such as MAE, MSE, RMSE, and MAPE, only evaluate the model outcomes at a specific threshold of data distribution, comparing the performance of different models accurately becomes a challenge. It is difficult to claim that model A outperforms model B with a particular data distribution because the data distribution type directly impacts the model's performance [21, 26, 27]. Furthermore, by gaining more insight into the data distribution point where SI model performance begins to improve or deteriorate, we can better understand the strengths and limitations of these models and thus improve their effectiveness in real-world applications.

Secondly, we explore whether machine learning techniques need to retrain the model with new hyperparameters and identify the optimal time to do so.

An additional aspect to consider in our research is our models' optimal training window size. Following 2017, the price changed and became smoother. The window size, measured in days, weeks, or months, is a critical variable affecting predictions' accuracy [17, 18]. It is important to examine this factor thoroughly when evaluating our models to ensure that our predictions are as precise as possible. Our research aims to explore this area thoroughly and provide valuable insights into the influence

of training window size on the performance of our models.

The fourth research question aims to investigate whether the timing of the collected data affects the model's price prediction accuracy, even after the price change's smoothness in 2017.

Overall, this study aims to contribute to developing more accurate and efficient machine learning models for SI price prediction in cloud computing.

1.4 Objectives

This thesis aims to investigate the performance of machine learning models in predicting the prices of spot instances in the EC2 cloud computing market. We use a new visualizing measurement called the Regression Error Characteristic Curve (RECC) [21] to assess the performance of price-prediction models for the spot price. Also, we investigate the potential of using machine learning models for price-prediction models of SI to achieve simpler and more accurate models. This is done by analyzing training time's impact on the performance of price-prediction models for SI. In addition, this thesis assesses the effect of training window size on performance, considering the smooth change of prices. Moreover, the need for changing the model hyperparameters over time is investigated, and the best time to identify the best time to retrain the model with new parameters for accurate predictions of spot prices is identified.

1.5 Thesis Contributions

This thesis significantly contributes to cloud computing and EC2 Spot price prediction. The following are the key contributions of this research:

(1) Study of Machine Learning Regressors: This thesis compares the performance of

several classic machine learning algorithms, including eXtreme Gradient Boosting (XGBoost), Random Forest Regressor (RFR), k-Nearest Neighbor Regressor (k-NNR), and Support Vector Regressor, in predicting EC2 Spot prices. The study aims to identify the most effective algorithm for spot price prediction.

(2) Tuning of hyperparameters: We investigate the effects of the hyper-tuning of the four models hyperparameters using k-fold cross-validation and Randomized search.

(3) Time Effect: The research evaluates the effect of new fluctuations on the models across the years by using the data after the smooth change of prices in 2017. This allows the study to understand the impact of timing on the performance of the prediction models.

(4) Window Size Training Effect: The study explores the training window size and its effect on the performance of machine learning models. It uses four training window sizes and tunes the ML models' hyper-parameters based on each size. As a result, 180 models are built in the study, making it a comprehensive analysis of the training window size effect.

(5) Evaluation Metrics: This thesis makes a significant contribution by evaluating the overall performance of the models using a new visualizing measurement called the Regression Error Characteristic Curve (RECC) and the Area Under the Curve (AUC-RECC). The evaluation results are compared with the commonly used metrics in the field, including MAPE, RMSE, MAE, and MSE. The study concludes that using RECC outperforms these traditional metrics and provides an easier interpretation of the results.

Our research concludes that the training window size and timing are crucial factors in achieving optimum results. At the same time, using RECC as an evaluation

metric is an innovative approach that provides a comprehensive and straightforward visualization of the performance of the models.

1.6 Thesis Organization

The rest of the thesis is structured as follows. Chapter 2 is about Background and Related work. This chapter will discuss the background and related work correlated to the thesis topic. The first section overviews the fixed and dynamic pricing strategies used in EC2 SI's. We include a discussion of market trends, particularly the shift towards using EC2 Spot instances in 2017 and its impact on price fluctuations. The second section reviews recent studies addressing the EC2 Spot price prediction. We go over the various studies in this field of predictions, organizing them based on their forecasting technique. We include reviewing statistical, deep learning, and classical machine learning algorithms. Chapter 3 shows our approach to solving the price prediction problem and evaluates the models' performance regardless of data distribution. We discuss the terms and definitions used while analyzing the data set concerning its properties. We continue by stating the ML algorithms used in building the algorithms, followed by the evaluation metrics used to assess the performance and accuracies of the models.

Chapter 4 describes the experimental framework used to implement the methodology, followed by a presentation of the data preparation phase. We discuss the techniques we used for training and validation, followed by the results of the evaluation techniques. Chapter 5 summarizes the contributions of this thesis and provides insights into potential areas for future research.

Chapter 2

BACKGROUND AND RELATED WORK

This chapter of our thesis provides a foundation for the study by presenting the existing knowledge on the cloud pricing of IaaS and highlighting the research objectives. In particular, we demonstrate the difference between static and dynamic pricing schema, including the on-demand, reserved, and spot instance pricing models. Since 2009, AWS has provided spot instances with pricing based on market auctions with users' bids, leading to significant price fluctuations [3, 28]. However, in 2017, AWS shifted the pricing mechanism to retail control, where customers set the maximum price they are willing to pay per hour [2, 4]. Although this change emerged, spot instances remain in high demand because they provide a cost-effective option contrary to on-demand instances. Nevertheless, using spot instances carries challenges, including potential execution interruptions due to the spot price directly affecting instance availability. Cloud users require a reliable method for predicting spot prices to manage their instance demand better. Thus, the related work section 2.2 sums up the proposed forecasting and prediction models as a solution for the spot prices problems. Additionally, we provide an overview of the shift in the spot price in 2017

and its impact on the pricing models.

2.1 Background

This section aims to provide a comprehensive reference for understanding the financial expenses and common characteristics of today's IaaS cloud computing platforms. As more and more businesses rely on cloud-based infrastructure as a service, the market for these services is expanding rapidly [29].

The use of Infrastructure as a Service (IaaS) cloud computing platforms is growing rapidly, and cloud providers are fiercely competing for market share. They use various pricing methods to compete, including static [30] and dynamic pricing [31–33] schemes. This study analyzes the pricing methods in the public infrastructure cloud and focuses on the four IaaS types: On-demand, Reserved, Sustained, and Spot Instances.

We gathered data from the top five cloud providers according to Gartner [29], Amazon Web Services, Microsoft Azure, Google Cloud, Alibaba Cloud, and IBM, and compared the common price parameters across and within each pricing model. To demonstrate our analysis parameters, we constructed tables that show the correlation between cost and service types.

Cloud providers offer services and pricing options for Infrastructure as a Service (IaaS). To facilitate equitable insight into the IaaS cloud pricing market, this study focuses on the two IaaS schemes, including four types: On-demand, Reserved, Sustained, and Spot Instances.

Table 2.1 displays the affinity and disparity of standards collected from four models. The billing criteria used to measure billing related to a specific model include the price perspective, commitment term, and duration period for both the company and users. The table compares the price deductions included within the billing, which

reach their maximum in the spot model and offer a free tier in the reserved model. The time unit or charge unit charges a defined amount of money per unit of time, such as minutes, hours, months, or years. Table 2.1 presents prospects under types divided into three sets for each model. The types are classified into three main groups: Billing, General, and Change, inherited from the five cloud providers mentioned before.

In the billing section, the price metrics will be classified as Static (S) or Varied (V) and charged per second (/s), per hour (/h), per month (/m), or per year (/y). For the general capacity criteria, the table denotes whether the capacity is available, using a plus sign (+) to indicate that the Spot Request continues to make the launch request until capacity becomes available automatically. A minus sign (-) indicates that the request is stopped. Special characteristics are mentioned for some options, and the table uses symbols such as (|) to show dependency on the service provider and (‡) to indicate that the capacity is unavailable. The (ICE): Insufficient capacity error and Ø: No Commitment.

Table 2.1: Summary of Fixed and Dynamic Instance Pricing Schema

Type	option/ model	On-demand	Reserved	Sustained	Spot
Billing	**Price Philosophy**	Pay the most	Save money	Cost reduction	Inexpensive
	Free Tier	×	✓	×	×
	User Commitment	×	✓	×	×
	Company Commitment	✓	✓	✓	×
	Discount	0%	(40 – 72) %	(25-60) %	(50 – 90) %
	Time Unit	/s /h /m	/h /m /y	/h /m /y	/h
	Hourly price (S or V)	S	S	$V*$	$\mid (S+V)*$
General	**Term Commitment**	Ø	1 - 3 years	Ø	Ø
	Capacity Request	‡ – ICE	Available	Available	‡ +
	Launch time	Manually	Auto	Manually/Auto	Auto/Manually
Change	**Size(resizing)***	✓	✓	✓	×
	Migration*	✓	✓	✓	×

2.1.1 Fixed Pricing Schema

The fixed price scheme in the cloud refers to a pricing model where customers pay a predetermined price for a set amount of resources or services over a fixed period. Cloud computing commonly uses this approach to give customers greater

cost certainty and budget control.

Cloud service providers often use static pricing as a simple and predictable model for customers. It allows customers to budget and plan their usage of cloud resources without worrying about unexpected pricing changes. There are three classes worth highlighting in this schema, on-demand, reserved, and sustained-usage plans [32, 33].

For example, in the case of IaaS, customers can pay a fixed price for reserved instances, allowing them to use specific resources for a set period. This pricing model can be contrasted with on-demand or pay-as-you-go pricing, which charges customers for the resources they use as they go, often at an hourly rate. Fixed pricing is typically offered for longer-term use cases or customers requiring a specific performance or capacity level. It can be used in pay-as-you-go (PAYG), reserved instances, or sustained plans allowing users to choose a pricing model that meets their business needs and budget. First, the on-demand or pay-as-you-go model charges customers for the resources they use as they go, often at an hourly rate. Fixed pricing is typically offered for longer-term use cases or customers requiring a specific performance or capacity level. In contrast, the reserved instances are a pricing model where customers commit to using specific cloud resources for a set period, typically one or three years, in exchange for a significant discount on the hourly rate. This is suitable for long-term usage requirements.

At the same time, sustained-use instances offer a discount on the hourly rate for instances that run for a significant portion of the month, regardless of whether they are reserved. This pricing model is useful for customers with predictable workloads who do not want to commit to specific resources over a longer period.

2.1.2 Dynamic Pricing Schema

Dynamic pricing in cloud computing is a flexible pricing strategy involving continuous price adjustments for virtual machines (VMs). The advent of this pricing approach has attracted significant attention for its potential to lower costs for organizations and individuals. Spot instances were first introduced in 2009 to optimize the utilization of spare compute capacity in the cloud [2]. Spot instances offer users the chance to attain considerable price reductions of up to 90% in exchange for a lack of commitment from the cloud provider. This makes cloud computing more accessible and cost-effective for a broader range of organizations and individuals. Researchers in [3,4] indicate that, on average, spot instances are more cost-efficient than on-demand instances for workloads shorter than approximately 340 hours or approximately two weeks. Dynamic pricing transformed how organizations and individuals access and utilize cloud computing resources. By continuously adjusting prices in response to changes in demand and supply, this pricing strategy ensures that resources are utilized optimally and costs are minimal. This, in turn, has the potential to drive innovation and foster the growth of cloud computing as a whole. Figure 2.1 shows the spot instance timeline based on the AWS cloud provider. The Spot Instance (SI) started for the first time in 2009. The spot historical prices of up to three months were available for all users in 2014. The fluctuations in prices were high from 2009 to 2017. In 2017 AWS announced new pricing standards making the prices more stable and less fluctuating.

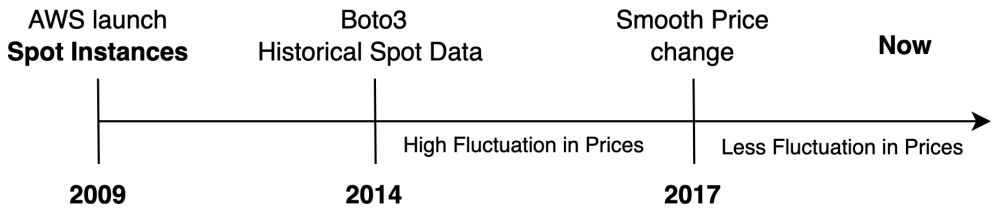


Figure 2.1: The AWS spot instances timeline from 2009 to nowadays.

Spot Pricing Shift in 2017

Price ambiguity is one of the users' biggest problems in the spot market. Pricing criteria were unrevealed [3]; for many years, various studies have proposed extricating bidding strategies and maximizing revenue from bid price [28]. Rapid changes and fare biding were the foremost characteristics ruling the spot market. Baughman et al. [4] critically analyze the effect of the 2017 dynamic change in the AWS Spot instance market. According to this change, one of the obvious results is its drive-off implementation of all extensive prior work on the spot market. AWS pricing criteria introduced the new spot market to refine the user experience for SI [34]. The presented model largely focuses on supplying users with more stable fare standards through short intervals such as days or weeks. Before this model, the work of instances was evicted when the bidding price went up. This continues if the price rises till it reaches the highest offer. Now, you pay only for the current hour for the instances you launch.

Therefore, maximum bidding prices are not required anymore. In addition, you can find the price that's in effect for the current hour in the EC2 console [2]. The new system sets the maximum price to the standard on-demand price. However, users still have control to submit their max price. Furthermore, interruption notably decreased in the new system due to the separate Spot pool and responded to an interruption notice by stopping or hibernating rather than terminating instances when capacity is no longer available.

2.2 Related Work

Since the spot instance market launched, spot pricing has been a significant concern for researchers. Although, before the smooth shift in spot prices [2], many

prediction models were suggested to find reliable price predictions providing different evaluation techniques. Hence, a considerable number of researches conducted to forecast spot prices using different approaches, such as statistical approaches, ML models, and deep learning. We limited our related work to studies implementing Statistical, ML models, deep learning, and their evaluation measurements.

2.2.1 Statistical Models

As a part of forecasting strategies established earlier in the area of forecasting Time-Series problems in various fields like (Energy and stock prices, etc.), many studies investigate the mathematical and statistical techniques for spot price predictions (e.g., [35–38]).

In their study, Duan et al. (2017) [35] proposed a new prediction method called HMM-E, which stands for Hidden Markov Model and Expectation. This approach utilizes Markov processes to model the demand markets and fluctuation degrees of spot prices as separate hidden states and observations, respectively. The authors found that this method outperformed existing autoregression-based forecasting methods (e.g., ARIMA) in predicting spot prices for less than five hours for short forecast periods. These results suggest that HMM-E could be a promising tool for predicting spot prices in cloud computing, especially for short-term forecasting.

In 2018 Cai et al. [36] proposed statistical and mathematical models to predict spot prices using the characteristics of spot prices. They used dynamic ARIMA (D-ARIMA) and two Markov regime-switching auto-regressive dynamic model-based forecasting methods (DMRS-AR-L) and (DMRSAR-SW). The dataset used was collected from 144 days of spot instance price history. They deemed they are providing methods for predicting the prices depending on the prediction length as a regime. They found that D-ARIMA usually converges too fast on the mean of the total time

series. Therefore, it is not suitable for long-term prediction. They found that DMRA-AR-L performed best for forecast periods shorter than 24 hours in most cases, while DMRA-AR-SW was more effective for longer forecast periods. Their finding depends on the VM types and application spans. We argue that these models are hard to implement since they require categorizing techniques like types.

Where Khan et al.(2022) [37] analyze the prices of the GPU spot instances, then predict their prices using four different statistical models. The work was trained based on three training windows and used the MAPE as an evaluation metric. The drawback of using the AR, ARIMA, and ETS models is that they do not predict sudden fluctuations in data or prices. In contrast, the GARCH model is unsuitable for predicting the dataset's seasonality and trend.

In a recent study conducted by Caton et al. (2022) [38], it was argued that complex machine learning models, such as deep learning models, are unnecessary for predicting spot market prices after the pricing mechanism shift in 2017. Instead, the study utilized statistical models such as Holt-Winters, autoregression, and ARIMA. The authors collected data before and after the pricing change and compared the performance of these statistical models to that of the LSTM model used in their previous work [13]. The results showed that the statistical models performed similarly or better than the LSTM model. However, it is important to note that the new study did not test the resistance of the proposed models in the face of sudden changes in the data, and previous data suggests the inferiority of statistical models compared to machine learning models. Therefore, while the study demonstrates the usefulness of statistical models in predicting spot market prices, it is still unclear whether they can withstand sudden changes in the data.

2.2.2 Machine Learning Models

Khandelwal et al. (2017) [17] proposed a detailed price analysis using the Random Forest Regressor (RFR) Algorithm for predicting AWS spot instance prices. The study aimed to evaluate the accuracy and speed of prediction using various comparisons of five machine learning models, including SVM, Multilayer FeedForward NN, Decision Trees (DT), and Regression Tree Ensembles (RTE). The RFR model was found to be the most accurate. The study used data collected between April 2015 and March 2016 for 12 months, limiting the dataset to spot prices lower than the on-demand market. The model's accuracy was measured using MAPE, Mean Consequential Percentage Error (MCPE), and speed metrics. The study concluded that the RFR model performs best for prediction periods 1-day and 1-week ahead. However, this study did not consider the differences in spot instance prices between different availability zones and day periods. The authors established several de-correlated trees to overcome this limitation and calculated out-of-bid errors for different leaf sizes. They determined the optimal leaf size to address the problem of overfitting. Their proposed method could predict the spot instance prices for the next week or day in less than one second. Compared to other machine learning methods, the RFR model was highly robust in price prediction and was easy to implement.

k-nearest neighbors Regressor (k-NNR) was implemented by Liu et al. (2020) [18] as a predominant algorithm. The model's performance is evaluated using 88 days of spot instance prices from 4 regions and 9 instances and is compared with several other models, including LR, SVR, RF, MLPR, and gcForest. Two prediction window sizes were used for training, 1-day, and 1-week ahead. The accuracy estimated using MAPE demonstrates the superior k-NNR performance in both windows, with a slight enhancement of accuracy with 1-week ahead predictions. Again, the data used in this study is from before the price change in 2017, in addition to the limitations of the

k-NNR algorithm in real-world data implementations.

Several previous works on Long/Short-Term Memory (LSTM) recurrent neural networks were discussed in [13,15,16]. Matt et al. (2018) [13], the LSTM model used a dataset of one instance type. The data was collected for 8 days in Sept 2016 and used the difference between on-demand and spot prices as an input of the model. The model was re-trained as a pre-processing step to minimize the training cost, and the seasonal regulation was investigated, showing slight improvement in the results. The accuracy metrics Mean Square Error (MSE) is the main metric in comparison, in addition to the MAPE, RMSE, and MAE as accuracy metrics. While the model was well implemented using the LSTM, except that the accuracy metric MSE is not an appropriate metric to measure the model's loss or accuracy in this case. Thus, in 2022 [38], the authors criticized their use of these metrics and used the MAPE only as an evaluation measure for the new work.

Similarly, authors in 2018 [16] used the LSTM neural network with a dataset of 11 months, 90 days from Dec. 2017 to Feb. 2018, and 8 months from Mar. to Sep. 2016. The K-fold technique was applied to find the best neural network hyperparameters. The RMSE was employed as an accuracy measure, one of the weakest measures used in the field of regression according to [23].

Kong et al. (2021) in [19] analyze three months of spot price history data from August 2018 to November 2018, identifying instance types that still have fluctuations and analyzing the impact of different Operating Systems and zones on price. A k-AMSE approach is proposed to evaluate the volatility of price data. They used a basic GRU network and added two layers. The machine learning prediction algorithms Random Forest, ExtraTree, and ANN are used for price prediction as a comparison. At the same time, the GRU network is found to have higher accuracy than other prediction methods, with RMSE used as the evaluation parameter. However, no

details about the classical ML models used or any other evaluation metrics rather than RMSE were investigated.

Al-Theiabat et al. (2018) in [15] proposed a deep learning approach using RNN-LSTM with data of 90 days collected between Dec 2017 and March 2018. The accuracy metrics were the RMSE, MAPE, and the Mean Absolute Scaled Error (MASE). The MASE is used as a new measurement to evaluate performance in the field evaluating spot prediction. They used two approaches for prediction, the deep learning model (LSTM) and a statistical model AutoRegressive Integrated Moving Average (ARIMA). The LSTM outperforms the ARIMA demonstrated by both MAPE and MASE.

Baughman et al. (2018) [4] train RNNs (Recurrent Neural Networks) with the historical prices from all regions, availability zones, and instance types to predict the prices of one instance type. They consider the LSTM (Long/Short-Term Memory) characteristics that can identify and re-member potential features in uncertain periods and regard it as the main component of RNN. Their experiments show that the network structure that can obtain the best prediction results is a five-layer network (i.e., one LSTM layer, two dense layers, one LSTM layer, and a final dense layer). Two preprocessing steps are adopted to reduce the overhead of training the model, which can adjust the price and consider the seasonality. Their prediction model can reduce training error by as much as 95 % and adapts to the old and new pricing models.

Authors in 2022 [39] proposed a deep learning method of predicting Amazon EC2 spot price using the Temporal convolution network and improving it with an attention mechanism so it could catch the historical features most relevant to the predicted price from the historical sequence. Then, they deemed that they performed extensive experiments to determine if their approach's accuracy was higher than any

other baseline models: LGB, LSTM, CNN, and LSTM-xfea. The MAE, RMSE, and MAPE were adopted as evaluation metrics of the models. Their work is unreliable since they provided data for 1 year and made a tremendous mistake in processing the features of ML models. They used the TimeStamp feature as it does not show trustworthy results about how they processed the data hourly.

In [40], the authors proposed a modified gated recurrent unit (MGRU) model and compared its performance to five other statistical and deep learning methods: ARIMA, RNN, LSTM, GRU, and Transformer. To evaluate the performance of each method, the authors used a grid search to assess all varieties of hyperparameters for each model, and the best combination of them was selected for each model. The prediction accuracy of all models was evaluated using RMSE, MAE, the coefficient of determination (R2), and MAPE. The data used in the study was collected from one region for three instances, with a duration of 90 days from March 7 to June 7, 2016. However, it is worth noting that the data used in this study is outdated, and since the research was conducted in 2022, AWS has already changed its pricing strategy, making prices more stable since 2017. Therefore, the comparison made in this study may not reflect the current state of AWS spot pricing challenges.

Table 2.2 presents the models utilized in the related studies. The models are classified into statistical and deep/machine learning. An asterisk (*) is used to indicate the models preferred by the authors. The remaining columns indicate the data specifications used in the studies.

Table 2.2: Related work models and other important features of used data

Authors Names	Statistical Model	Machine/ Deep learning Model	Data length	Region/Zone	Operating System	Instances Types
Song et al. (2022)	None	TCN*, LGB, LSTM LSTM_xfea, CNN	2019	14 regions	Not determined All OS Systems	Not determined
Khan et al. (2022)	ARIMA, AR, ETS, GARCH	None	May 2019 - Jul. 2019	7 Zones	Linux/UNIX	GPU spot instance
Caton et al. (2022)	Holt-Winters, AR, univariate LR, ARIMA	None	Jul–Sep 2017 Jan–Mar 2018 Dec 21 - Feb 22	us-east-1,us-east-2, us-west-1, and us-west-2	Linux/UNIX	70 instances AWS spot instance types
Seyed et al. (2022)	ARIMA	MGRU*, RNN, LSTM, GRU, and Transformer	March 7 to June 7, 2016	Oregon region	Not determined	3 instances C3.2xlarge, M3.2xlarge, M3.medium
Alkaddah and Agarwal (2022)	None	RFR, SVR, k-NNR, XGBoost	2019, 2020	5 regions	Linux/UNIX, SUSE LINUX, Windows, RED HAT	10 instances Compute Optimized (C3,C4) c3.large,.., c3.8xlarge c4.large,.., c4.8xlarge
Kong et al. (2021)	None	LSTM* RF, ExtraTree, ANN	90 days 2018-08-28 to 2018-11-24	us-east-1	Linux/UNIX, SUSE LINUX, Windows, RED HAT	All instances
Liu et al. (2020)	None	k-NNR*, LR, SVR, RF, MLPR, gcForest	88 days	4 regions us-east-2a, ap-northeast-2a, ap-south-1a, ca-central-1a	Linux/UNIX	9 instances c4.large, c4.xlarge, c4.2xlarge m4.large, m4.xlarge, m4.2xlarge, r4.large, r4xlarge, r4.2xlarge
Khandewal et al. (2017)	None	RFR*, SVM, MFF-NN, DT, RTE	Apr., 2015 – Mar., 2016 simulation	eu-central, us-west EU, US, Asia Pacific Canada regions	Linux/UNIX, Windows	10 instances Compute Optimized (C3,C4) c3.large,.., c3.8xlarge c4.large,.., c4.8xlarge
Cai et al. (2018)	D-ARIMA,DMRS-AR-L DMRSAR-SW	None	Between period from April, 2016 to May,2016	us-east	Linux/UNIX	All instance types
Al-thiabat et al. (2018)	ARIMA	CNN-LSTM*	90 days Dec 2017 to March 2018	us-west-1c, us-west-1c, ap-southeast-1a, us-east-1a	Linux/UNIX	4 instances c3.2xlarge, c1.xlarge, m4.large, c3.4xlarge
Baughman et al. (2018)	ARIMA	CNN-LSTM*	8 days Spet. 2016	us-east-1b	Linux/UNIX	1 instance c3.2xlarge
Sarah et al. (2018)	ARIMA, AR, MA	LSTM-kfold*, LSTM	Dec. 2017 to Feb. 2018 Mar. to Sep. 2016	10 regions	Not determined	Not determined
Duan et al. (2017)	D-ARIMA, HMM-E	None	Apr. , 2016 to Jul. , 2016	us-east-b/c	Linux/UNIX	2 instances
Agarwal et a. (2017)	None	CNN-LSTM*	90 days	us-west-1, ap-southeast-1, ap-southeast-2, us-west-2	Not determined	7 instances

2.2.3 Measurements of Evaluation

Several regression metrics are used to assess the forecasting model's performance. Table 2.3 lists the most current related work with the evaluation metrics used to measure their prediction model accuracy. While there are 7 distinct measurements utilized, they are not all appropriate to measure the loss in the predicted results. The most used metrics are in the order of MAPE, RMSE, and MAE.

First, The MAE is a metric used to describe the average magnitude of errors between predicted and true values. It is calculated as the average of the absolute differences between the predicted and true values without regard to the direction of the errors. While it can be useful for evaluating overall accuracy and comparing different models, it only provides information on the average magnitude of the errors and not their variability, distribution, or frequency. Therefore, it may be necessary to use additional error metrics to obtain a complete understanding of the accuracy of the predictions.

RMSE is a popular error metric in many fields, such as engineering, statistics, and data science. It is calculated as the square root of the mean of the squared errors between the predicted and true values. It provides a measure of the average magnitude of the errors and considers the direction of the errors, unlike the MAE. One issue with RMSE is that outliers can heavily influence the metric in the data, leading to bias in favor of models that perform better on the majority of the data but worse on outliers. Additionally, RMSE is sensitive to differences in the scale of the data, making it harder to compare the magnitude of the errors to the magnitude of the data. Armstrong in [41] argued that using RMSE is untrustworthy and inappropriate for comparing accuracy across time-series predictions. Therefore, it is important to consider the limitations of RMSE, such as its sensitivity to outliers and differences in data scale. It may not always be the best choice, depending on the data's specific

characteristics and the analysis's goals.

MAPE is a percentage-based error metric that measures the average percentage difference between the predicted and true values. It is useful when the data has a strong seasonality or when the absolute magnitude of the errors is more important than their direction. However, MAPE has some limitations [41], such as being sensitive to small denominators and having infinite values for actual values equal to zero. Also, it is asymmetric, meaning that the error for an overestimation may be weighted differently than for an underestimation. This biased evaluation of the forecasting model might not reflect the price change due to the lower values of the spot prices.

Table 2.3: Related work metrics

Name/ Metric	**R2**	**MAE**	**MSE**	**RMSE**	**MASE**	**MCPE**	**MAPE**	**REC Curve**
Song et al. (2022) [39]		✓		✓			✓	
Khan et al. (2022) [37]							✓	
Caton et al. (2022) [38]							✓	
Nezamdoust et al. (2022) [40]	✓	✓		✓			✓	
Alkaddah and Agarwal (2022)		✓	✓	✓			✓	✓
Kong et al. (2021) [19]		✓	✓	✓			✓	
Liu et al. (2020) [18]							✓	
Khandewal et al. (2017) [17]						✓	✓	
Cai et al. (2018) [36]							✓	
Al-thiabat et al. (2018) [15]				✓	✓		✓	
Baughman et al. (2018) [13]		✓	✓	✓			✓	
Sarah et al. (2018) [16]				✓				
Duan et al. (2017) [35]							✓	
Agarwal et a. (2017) [14]							✓	

2.2.4 Review Summary

The data used in previous related work on predicting spot instance prices have been divided into two periods: one before the smooth price change in November 2017 and another after this change with less fluctuation and stable prices. This shift in data distribution has made predicting spot instance prices less complicated, as the data is more stable and has fewer spikes [4, 38, 42].

A literature review reveals that previous works have used various statistical and

Machine Learning (ML) approaches, including deep learning models and classical ML regression algorithms. Many researchers have compared their proposed approaches with baseline models, often using ARIMA (e.g., [13, 15, 36]) or other statistical methods (e.g., [13, 38]). Other researchers have compared their results with classical machine learning models (e.g., [17, 18]).

However, the deep learning approaches used in previous work have had weaknesses. Their data was outdated and required pre-processing techniques like normalization to fit the complexity of the deep learning training task. Similarly, statistical methods performed poorly in most cases before the price change compared to machine learning and deep learning techniques [13, 15, 36].

The debate was on implementing the SI price prediction task based on the deeper learning approach. Despite this debate, recent work by Caton et al. [38] has shown that statistical models are becoming more effective in predicting spot instance prices due to the change in the data distribution. They demonstrated this by objecting to their previous work, which used LSTM models [13] and instead implemented statistical models in the new data version. They used no classical ML algorithms to prove the same deep learning vision. Thus, we investigate the efficiency of using the machine learning models after the price Shift in 2017.

Measuring the accuracy and errors in regression problems is crucial in understanding the performance of machine learning models. Accurate predictions are essential for making informed decisions, and poor accuracy can result in significant financial losses or harm to individuals. Therefore, it is necessary to evaluate the accuracy of regression models before deploying them in the real world.

Measuring the accuracy of regression models allows us to quantify the difference between the actual and predicted values. This can help us identify the strengths

and weaknesses of different models and make informed decisions on which model to use for a specific problem. The evaluation metrics in the related work were mostly about using the dependent scale or dependent-free metrics. In [24], Botchkarev highlights that regression errors are random variables and could only be measured using the probability density function if exit. Thus, we aimed to measure cumulative distribution function (CDF) as an Area Under the REC Curve (AUC-REC) based on the visualization metric proposed by Bi [21], Regression Error Characteristics Curve (REC Curve). Our work is concerned with finding the best evaluation metric by considering the most utilized ones in the related work alongside two new metrics in this field.

Our research focuses on predicting spot EC2 prices using classical machine learning algorithms. We chose to evaluate the data after 2017, including two conservative years. We preprocessed the data to ensemble it into hourly series based on the instance, region, and operating system. Then, we used two tests to check if the data was stationary. Our main concern was about evaluating the performance of the non-parametric ML regression algorithms and proving their effectiveness in solving this time-series problem. We used six evaluation metrics, including the Regression Error Characteristics (REC) curve and the Area under the curve (AUC-REC), which are superior to conventional metrics. While building the models, we also considered three aspects: dataset time per year, training window as 1-day, 1-week, and 1-month ahead, and instances location. Our findings suggest that classical machine learning algorithms can effectively predict spot instance prices and that the REC curve and AUC-REC are useful metrics for evaluating the performance of these models.

Chapter 3

METHODOLOGY AND MODELS

This chapter describes the methodology and models employed in our study of evaluating price prediction models. Our research focused on exploring the impact of various factors on the performance of the models, such as data time, data location, and training window effect (e.g., 1-day, 1-week, 1-month). In addition, we seek to find a precise assessment measurement to estimate both the accuracy and the loss of the used models. Therefore, we utilized multiple supervised ML techniques and statistical analyses to achieve this. For model evaluation, we used selected performance metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), AUC-REC and REC Curve.

We employed non-parametric ML algorithms for modeling, which are , Random Forest Regressor (RFR), Support Vector Regressor (SVR), and XGBoost to capture the complex relationships between input and output variables in our data. These non-parametric models are flexible and can capture complex relationships that may not be apparent in the data [6]. This makes them ideal for our study, where we explore the impact of various data factors on model performance.

Before training the models, we conducted a set of data preprocessing steps, including data cleaning, unit root test, series generating, and feature extracting. The features were selected based on the related work [17, 18] to identify the most significant features previously used to train the models. Also, the unit root test was conducted to check the stationarity of the data using Augmented Dickey-Fuller (ADF) and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) Tests [43, 44]. Moreover, to ensure that the models were well-tuned and optimized, we tuned the hyperparameters for each model using a combination of randomized search and cross-validation [1, 45], selecting the optimal set of parameters that maximized the model's performance on the evaluation metrics.

3.1 Regression Problems

Regression analysis is a statistical method widely used in various fields, such as economics, engineering, and social sciences, to model the relationship between a dependent variable and one or more independent variables [46]. In machine learning, regression is a supervised learning problem that predicts a continuous output variable based on input variables [11]. Mathematically, regression aims to learn a function that maps the input variables to the output variable, given a set of labeled training examples [47].

More formally, let X be a n x p matrix representing **(n rows and p columns)** of the input variables, where each row of X corresponds to a data point, and let y be an n-dimensional vector representing the output variable. The goal of regression is to learn a function $f(X)$ that maps X to y, such that:

$$y = f(X) + \varepsilon$$

where y is the output variable, X is the input variable, $f(X)$ is the function that maps the input variable to the output variable, and ε is the error term or residual, which represents the difference between the actual value of y and the predicted value of y. The regression equation can be written as follows:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n + \varepsilon \tag{1}$$

where Y is the dependent variable, $X_1, X_2, \ldots, X_n$ are the independent variables, b_0 is the intercept, $b_1, b_2, \ldots, b_n$ are the coefficients that determine the slope of the line of best fit, and ε is the residual term [48]. Supervised machine learning algorithms can solve regression problems by learning a function that approximates the relationship between the input and output variables based on the labeled training examples. This learned function can predict the output variable for new, unseen input samples. The idea behind fitting the function is that each machine learning regression algorithm tries to minimize the residuals in different ways to predict the next output [48].

Residuals

The importance of residuals in evaluating the accuracy and performance of machine learning models cannot be overstated. In a regression function [1], the residual term (ε) is the difference between the target variable and the predicted value. In essence, the residual represents the discrepancy between the actual and predicted values of the dependent variable. An effective model should predict the dependent variable with minimal error or residual; therefore, the smaller the residual, the more accurate the model is [48, 49].

Residuals are a crucial measure of model performance because they reflect the errors made by the model in predicting the outcome variable. Large residuals suggest

that the model is inaccurate in predicting the outcome variable and requires improvement, while small residuals indicate a higher accuracy. Additionally, residuals can be useful in detecting outliers or influential points that could adversely impact model performance.

Analyzing the residuals can help identify trends or patterns in the data that the model is not capturing, which can aid in improving the model. Thus, residuals play a crucial role in evaluating model performance and identifying areas for improvement.

As an example, the RFR employs an ensemble of decision trees, where each tree predicts the output value. The final prediction is obtained by averaging the predictions of all the decision trees. In the case of RFR, the residual is the difference between the actual and predicted values, and the algorithm aims to minimize the MSE of the residuals [7].

3.2 Parametric and Non-parametric ML Algorithms

Determining between parametric and non-parametric machine learning models is an important consideration in any research project involving data analysis. Parametric models are often preferred when the underlying distribution of the data is well-understood and fits the model's assumptions. They tend to have a smaller number of parameters, which can make them more efficient to train and easier to interpret. However, parametric models can be overly restrictive when the underlying distribution of the data is complex or unknown, leading to poor performance and inaccurate predictions [11, 50].

Non-parametric models, on the other hand, are more flexible and can capture complex patterns in the data without making assumptions about the underlying distribution [6]. They are often preferred when the data is highly variable and does not fit the assumptions of a parametric model. However, non-parametric models can

be more computationally intensive and require more data to train, and their large number of parameters can make them difficult to interpret.

Our research considered the trade-offs between parametric and non-parametric models to select the best approach for our data analysis needs. We ultimately chose non-parametric models such as K Nearest Neighbor Regressor (k-NNR) [8], RFR [7], SVR [9], and XGBoost [10], as they were well-suited to the complex patterns in our data and offered high predictive accuracy. By leveraging the flexibility and power of non-parametric machine learning models, we could create robust and accurate models that captured the complex relationships between input and output variables in our data, whether before or after the smooth change in the distribution.

3.2.1 Models Hyperparameters

Hyperparameters in machine learning are parameters that are not learned from the training data but are set before the training begins. They are used to control the learning process and can have a significant impact on the performance of a ML model. Examples of hyperparameters include the learning rate in gradient descent optimization, the number of hidden layers in a neural network, the regularization parameter in a linear regression model, and the kernel function and its associated parameters in a support vector machine.

Finding the optimal hyperparameters is an important step in building a model that can perform well on new, unseen data. This process is often referred to as hyperparameter tuning or hyperparameter optimization. It involves searching over a range of hyperparameters to find the values that produce the best performance on a validation set.

3.3 Machine Learning Prediction Algorithms

Many recent studies have proposed the utilization of various machine learning algorithms, such as RFR, Decision Tree (DT), k-NNR, SVR, Logistic Regression (LR), and gC forest [17–19]. Additionally, deep learning approaches such as Long Short-Term Memory (LSTM) have been employed to predict prices over time [13,15,16,19]. Our research methodology focuses on four supervised machine learning algorithms: SVR, k-NNR, RFR, and Extreme Gradient Boosting Regression (XGBoost) to predict accurate AWS spot instance prices over time.

3.3.1 Support Vector Regression (SVR)

Support Vector Machine (SVM) is a supervised machine learning technique that requires labeled data to train and build knowledge [9]. The labeled data is used during the training phase to discover the decision boundaries of different target classes. These boundaries separate data on multiple dimensions (hyper-planes) rather than a single dimension like the linear regression algorithm (See Figure 3.1). The SVM also supports various kernels, including linear, polynomial, radial basis function (RBF), and sigmoid [1]. These different kernel levels provide adequate flexibility in recognizing nonlinear boundaries that divide data into groups.

Similarly, SVR works for discrete values. SVR finds a hyperplane that best fits the data by minimizing the error between predicted and actual values [51]. The algorithm uses training data to learn the relationship between the input features and the target values and then applies the learned function to make predictions on new data. SVR's performance largely depends on its hyperparameters, especially the C and gamma hyperparameters. The C parameter determines the trade-off between achieving a low training error and a low testing error. In contrast, the gamma parameter defines the influence of a single training example and can greatly affect the shape of the decision

boundary [1]. The tuning of these hyperparameters is crucial to ensure the optimal performance of the SVR model.

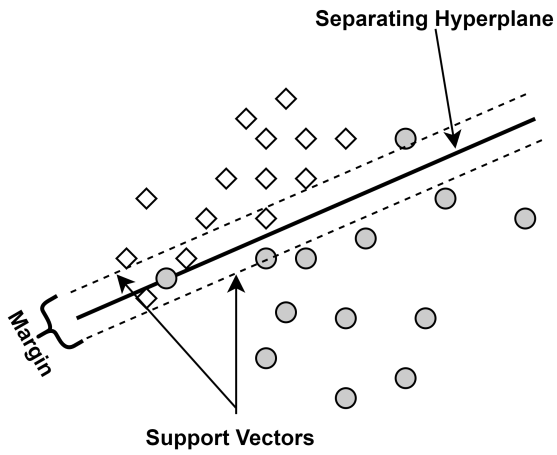


Figure 3.1: Support Vector Regression (Linear Kernel)

In Figure 3.1, the SVR is trained on the preprocessed data. The training process involves finding the hyperplane that maximizes the margin, the distance between the hyperplane, and the closest data points. Then, it uses two hyperparameters: the regularization parameter C and the kernel function. The regularization parameter controls the trade-off between the margin and the error. At the same time, the kernel function transforms the input features into a higher-dimensional space to make them more separable.

3.3.2 k-Nearest Neighbors Regression (k-NNR)

k-NNR (k-Nearest Neighbors Regression) is a non-parametric algorithm for regression problems. It searches for the k-nearest training data points to a new test point based on a distance metric such as Euclidean distance. As distance measurements, we select Minkowski distance because it represents the generalized formula of both (Manhattan and Euclidean) distances [52]. The predicted output value is the average of the output values of the k-nearest neighbors. The value of k is chosen based on

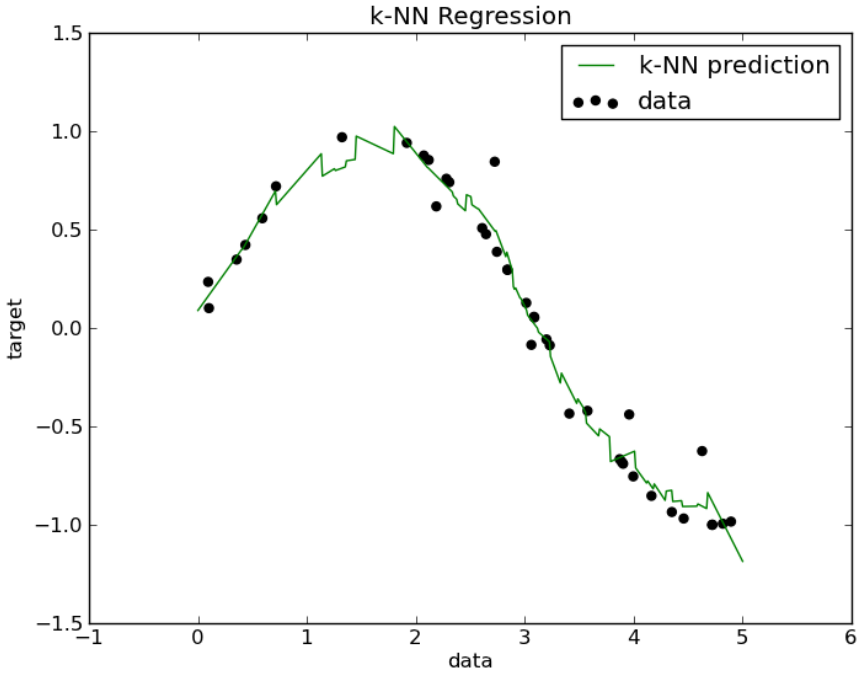


Figure 3.2: Example of regression using nearest neighbor [1]

the data and the problem being analyzed.

We tuned the n_neighbours parameter to find the optimal number of neighbors for the algorithm to consider when making a prediction. A smaller n_neighbours makes the model more sensitive to noise, while a larger value makes the model smoother but may miss important patterns [53].

The algorithm parameter specifies the nearest neighbor algorithm used to compute distances. The brute-force algorithm is computationally expensive for large datasets, while ball tree and KD tree algorithms use tree-based structures for efficient computation. We tuned this parameter to find the optimal algorithm that maximizes the model's performance. See Figure 3.2.

k-NNR is a type of instance-based learning which memorizes the training data for prediction. In other terms is called the lazy model. This makes it suitable for complex nonlinear relationships between input features and output targets. However,

k-NNR has limitations, including sensitivity to distance metric choice, the need for large training data, and high computational costs for large datasets.

3.3.3 Random Forest Regression (RFR)

Random Forest Regression (RFR) is an ensemble learning method that uses multiple decision trees to predict the output variable, as shown in Figure 3.3. First, the RFR uses an ensembling technique called bootstrapping to take a sample of data with a replacement for each tree. The trees randomly sample the training data subsets and input features to reduce overfitting [7]. Each decision tree is constructed by recursively splitting the data to minimize variance until the specified maximum depth is reached. To make a final prediction, RFR averages the predictions of all the decision trees (Result 1, Result 2,.., etc.) to improve accuracy.

To optimize the RFR model's performance, we tuned the maximum depth of the tree and the number of features used for each split. The maximum depth determines the number of splits allowed in each decision tree, capturing more complex relationships but risking overfitting [54]. The number of features determines the number of input features randomly selected for each decision tree, reducing overfitting but may result in lower accuracy. These hyperparameters can be adjusted to find the optimal values for a given dataset and problem.

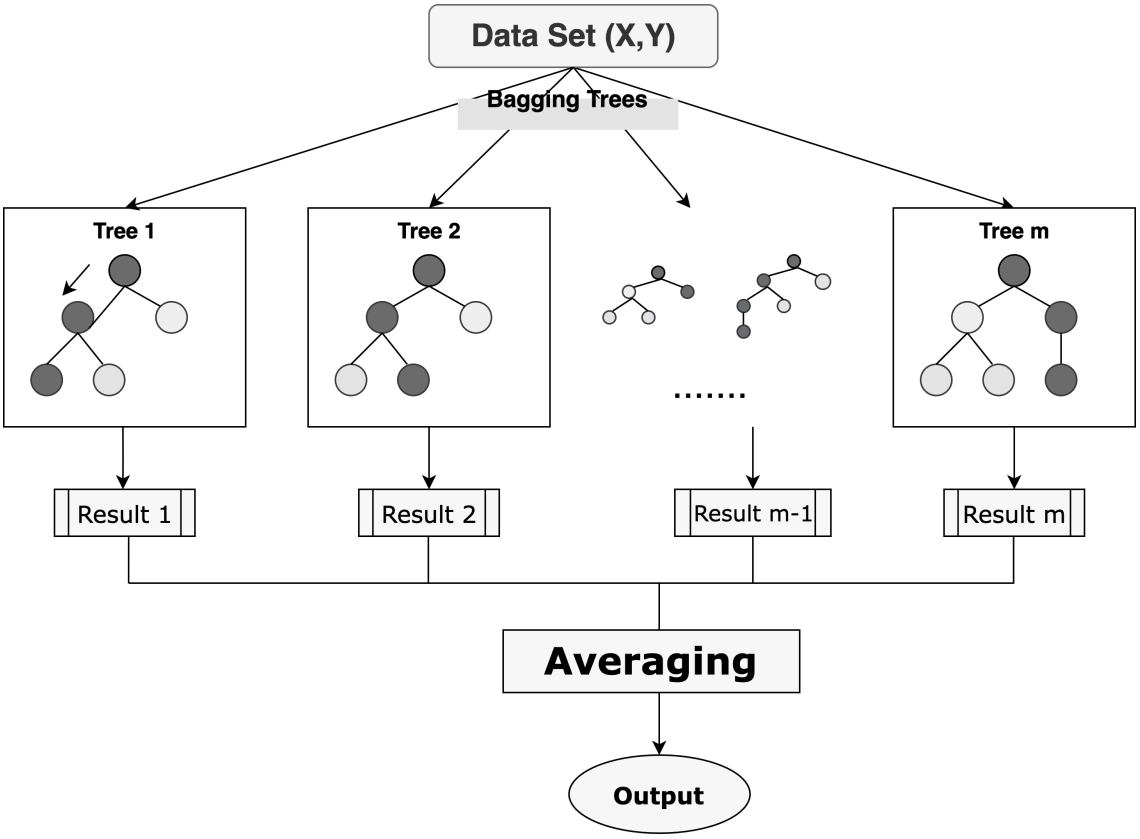


Figure 3.3: Random Forest Regressor Bagging Trees

3.3.4 Extreme Gradient Boosting Regression (XGBoost)

XGBoost is a machine learning algorithm for classification and regression problems. It works by constructing a series of decision trees, where each tree tries to correct the errors made by the previous tree. It optimizes the regularized loss function by applying the second-order Taylor series approximations within the context of gradient descent boosting [27]. During training, XGBoost selects the best-split points for the data based on how well the trees perform. It then adds new trees to the ensemble by continuously splitting features to fit the residual of the previous prediction. Once all the trees are built, XGBoost predicts a numerical value by summing up the predictions of each individual tree. The final output is a weighted sum of the predictions from all the trees, where the weight for each tree is determined by how

well it performed during the training phase [10].

Figure 3.4 shows how XGBoost works: It starts with Tree 1 by finding the best-split point for the entire dataset. Then, the XGBoost algorithm continually calculates the loss at each decision tree node and selects the leaf node with the largest loss reduction. The algorithm then adds a new tree to the ensemble, each tree learning a new function $f_m(X, \theta_m)$ that fits the residual of the previous prediction, where X is the subset of features and θ_m is the gradient boosting tree structure.

After training, the XGBoost model has a number of decision trees M, and each prediction sample corresponds to a leaf node in each tree, corresponding to a score. The scores of each tree are combined to produce the final prediction value for the sample.

In summary, XGBoost builds an ensemble of decision trees by continuously splitting features to fit the previous prediction's residual and combines each tree's scores to make a final prediction. The diagram of the XGBoost algorithm is shown in Figure 3.4.

We used a set of parameters while implementing the model to enhance its performance. The studied parameters include the colsample_bytree to identify the percentage of features used per tree, eta to prevent the model over-fitting, and max_depth of the gradient trees. The mean squared errors (MSE) are the loss function that reduces the preceding tree's residuals.

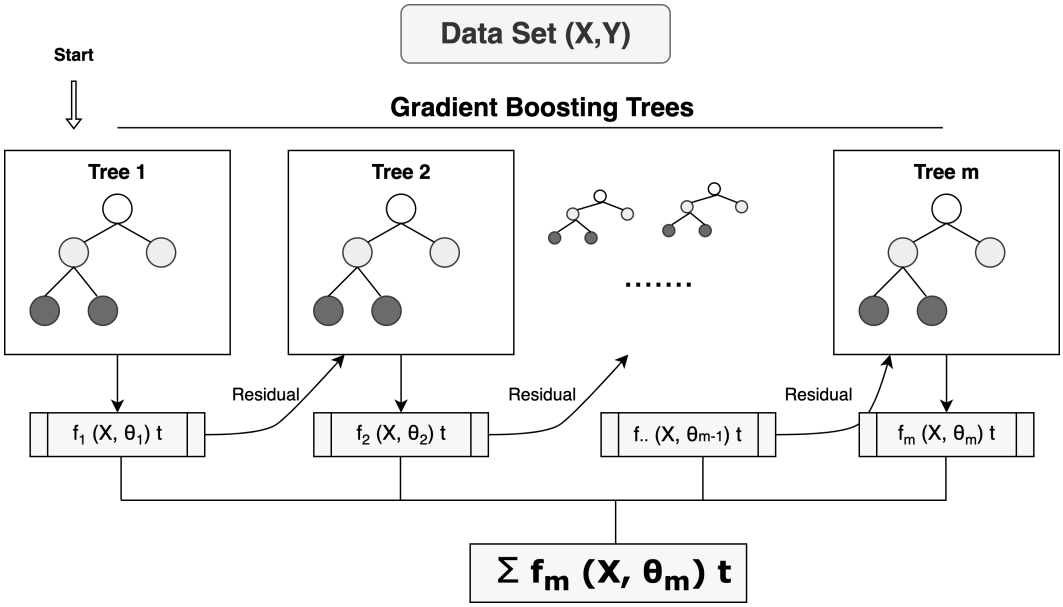


Figure 3.4: XGBoost Regressor Boosting Trees

3.4 Data Collection, Analysis, and Processing

In the spot model, if the price changes, then the SI prices will change hourly. The hourly preprocessing was done previously in most related works (e.g., [13, 16–18]). Hourly data may be necessary to capture the underlying patterns and fluctuations that occur over short periods of time. In financial forecasting, hourly data may be used to capture short-term trends and fluctuations in market conditions, which in our scenario is applicable over the spot price change of data.

3.4.1 Data Collection

The dataset used in this study consists of the historical spot price data of Amazon EC2 instances in various regions from 2019 to 2020. This data was collected as archived files by Ardi Calvin [55] from the University of Southern California; the data was collected daily based on the event level. Each instance was categorized based on its unique operating system data to prepare the data for analysis. The time

stamps were grouped into days, with hourly unit records generated for each day. The resulting dataset is organized in ascending order as a time series, with approximately 350,400 records per year. Given that this is a time-series problem, the model was trained using past data to forecast future data [56]. For instance, when the model was trained using a one-month window, the training data from January was used to predict the testing data in February. The study investigated the models' performance using three different time window sizes: day, week, and month. Previous research by Khandelwal et al. [17] and Liu et al. [18] suggested that the 1-week ahead prediction interval yielded the best results. However, the findings of this study indicate that the 1-day ahead window actually yields the best results, while the 1-month window yields the worst results.

3.4.2 Data Preprocessing

In our research, we encountered the challenge of processing training data initially provided at a detailed event level. Each event corresponded to a specific virtual machine instance running on a cloud platform. However, to use this data effectively in machine learning models, we needed to aggregate it hourly. Each hour record would represent a synopsis of the events during that hour. This conversion allowed us to create 200 unique time series, each representing a unique scenario of instance type, region of instance location, and operating system. To perform this conversion, we used the *'.ffill'* function from the popular scikit-learn library in Python [1], which allowed us to fill in any missing data values in the hourly data by forward-filling the previous value. This resulted in a complete and consistent dataset that we could use for further analysis and modeling. In Figure 3.5. This study chooses 5 regions and 10 featured instances, and 4 operating systems as shown in Table 3.1. These instances were selected previously in [17] from the Compute-Optimized family with

two generations **C3** and **C4**. The preprocessing flow diagram in Figure 3.5 shows how we processed the data in each region to generate 40 series as a dataset, and we have the R1 as an example. We started by considering each region separately, then looped over every OS. A filter for each instance is applied before start aggregating the event-level data to convert it into hourly. The output series was saved and then combined to produce each region dataset.

Table 3.1: Regions, Instances types and Operating System (Abbreviations and names)

Region		**Instance**				**Operatig System**	
R#	**Zone name**	**I#**	**Type**	**I#**	**Type**	**O#**	**Names**
R1	us-east-1a	**I1**	c3.large	**I6**	c4.large	**O1**	Linux/Unix
R2	us-west-1a	**I2**	c3.xlarge	**I7**	c4.xlarge	**O2**	Windows
R3	us-west-2a	**I3**	c3.2xlarge	**I8**	c4.2xlarge	**O3**	SUSE Linux
R4	eu-west-1a	**I4**	c3.4xlarge	**I9**	c4.4xlarge	**O4**	RedHat Linux
R5	ap-northeast-1a	**I5**	c3.8xlrge	**I10**	c4.8xlarge		

3.4.3 Data Analysis

Stationarity is a key assumption in many technical analysis models, as it implies that the statistical properties of the time series, such as the mean and variance, do not change over time [23]. It is important to know whether the data is stationary because the statistical properties of a time series can change over time if the time series is non-stationary. A non-stationary time series can exhibit trends, cycles, and other patterns that make it difficult to model, forecast, and make inferences about the data. For example, if a time series is non-stationary, it may exhibit a trend over time [44, 49]. This trend can make detecting other patterns or changes in the time series difficult; Figure **??** shows the different distributions of the series from regions R1 and R3. Similarly, if a time series is non-stationary, it may exhibit seasonal variations that can mask the underlying patterns in the data. We can use appropriate prediction

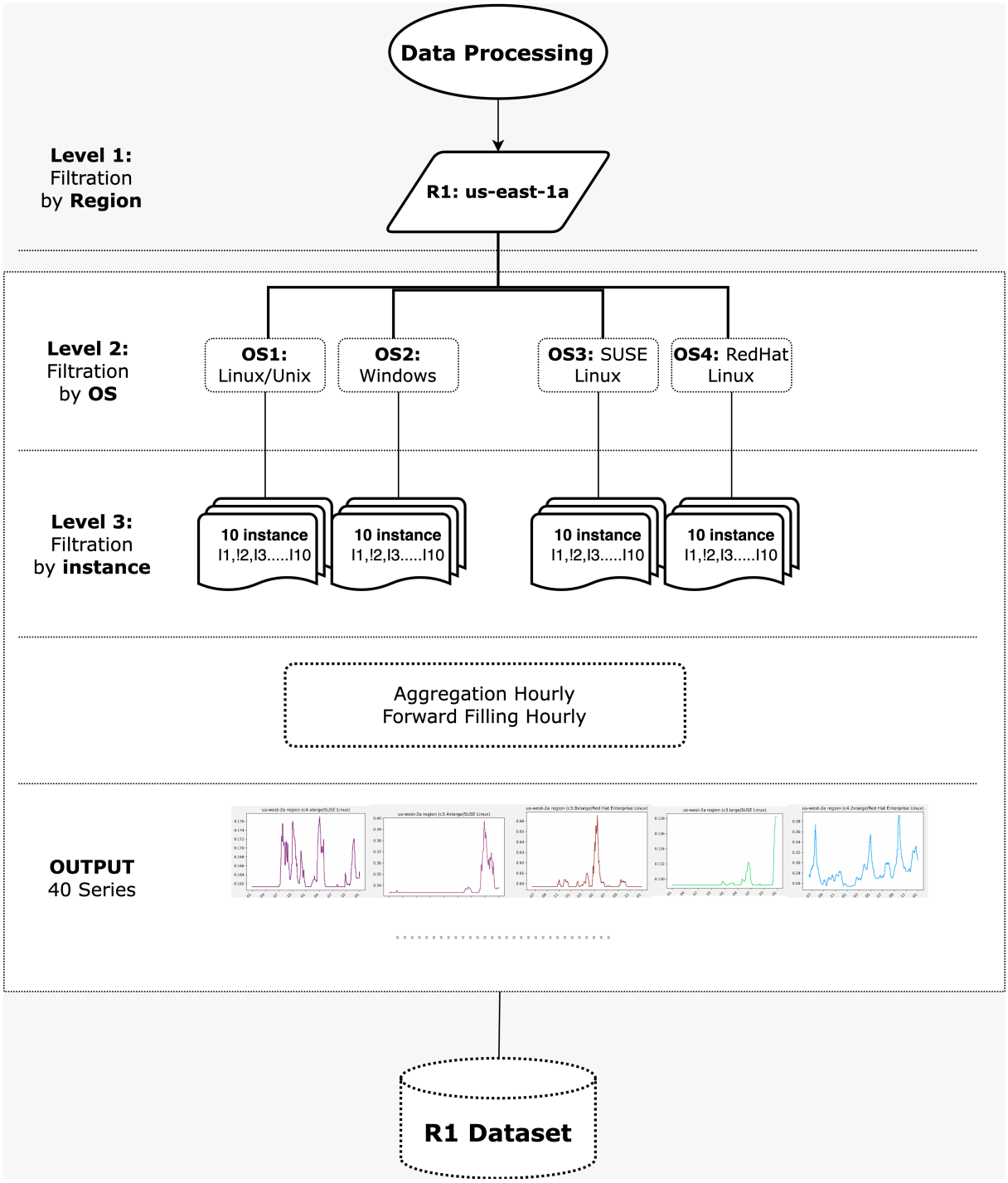


Figure 3.5: Processing steps of series in each region (e.g., R1 region)

techniques to model, analyze, and forecast the data by identifying whether the time series is stationary.

To check whether our data are stationary or non-stationary, we choose two tests to assess our data and the proper models to use. The First test is the Augmented Dickey-Fuller test (ADF) [43] and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test [44]. The ADF test checks whether the data has a unit root, indicating that the series is non-stationary. In contrast, the KPSS test checks whether the data has a trend, indicating non-stationarity. Authors in [38] used those tests to check the stationarity of the data. Also, Hyndman and Athanasopoulos' in their book Forecasting: Principles and Practice [49], recommended them as a time series stationarity test.

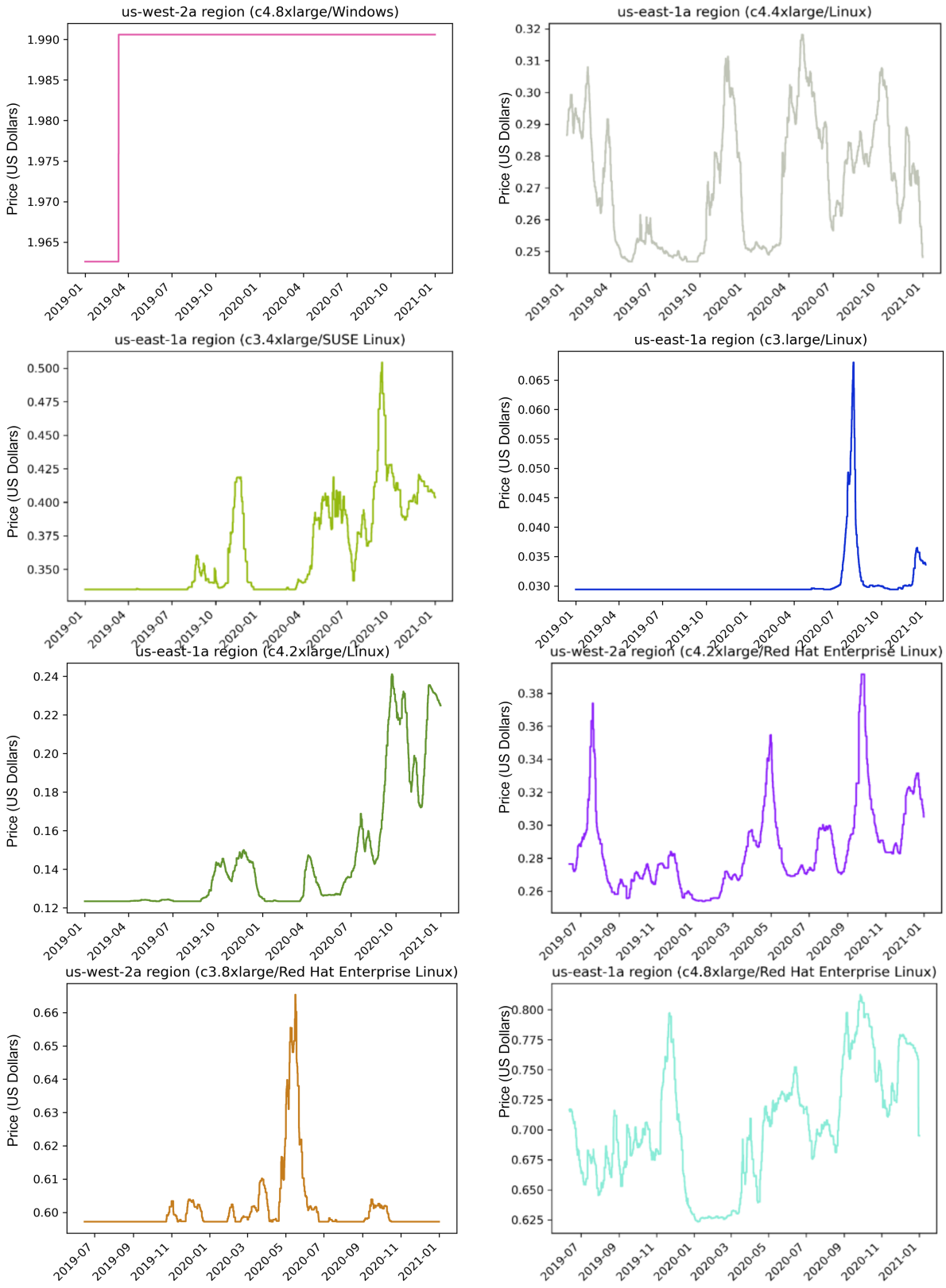


Figure 3.6: Series Distribution Examples in region R1 and R3.

Augmented Dickey-Fuller test

We use this statistical test to check for data stationarity. The Augmented Dickey-Fuller (ADF) test null hypothesis is that the time series has a unit root, which means that the time series is non-stationary. The alternative hypothesis is that the time series is stationary [43].

To check the null hypothesis in the ADF test, the p-value is considered. Suppose the p-value and test statistics are less than your chosen significance level (usually 0.05). In this case, the null hypothesis is rejected and it is concluded that the time series is stationary. If the p-value exceeds the significance level, the null hypothesis cannot be rejected, and the time series is non-stationary.

- **Null Hypothesis:** The time series has a unit root and is non-stationary.
- **Alternative Hypothesis:** The time series is stationary and has no unit root.

We applied this test to all the series scenarios we obtained from processing the data, and we found most of the series are non-stationary. The table of the ADF tests, including the p-values for each series are attached in Appendix A, Table A.2. The table concludes if each is stationary in region R1 as an example.

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

The KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test is a statistical test used in technical analysis to determine whether a time series is stationary. The KPSS test works by comparing between the actual values and the estimated trend of the time series. If the p-value of the test is below a certain significance level, typically 0.05, then the null hypothesis of stationarity is rejected, indicating that the time series is non-stationary. On the other hand, if the p-value is above the significance level, then the null hypothesis is not rejected, suggesting that the time series is stationary and

can be analyzed using various technical analysis techniques. The table of the KPSS tests, including the p-values for each series attached in Appendix A, Table A.1. The table concludes if each is stationary in region R1 as an example.

- **Null Hypothesis:** The time series is stationary around a deterministic trend.
- **Alternative Hypothesis:** The time series is non-stationary, meaning it has a unit root or a stochastic trend.

3.5 Training, Validation, and Testing

When testing and training the spot data, we adopted two distinct approaches to ensure the robustness and accuracy of the models. The first approach employed was the fixed rolling window method, which involves partitioning the data into fixed-size windows based on time, each consisting of a specified number of consecutive time intervals. In our study, we used window sizes of one day, one week, two weeks, and one month for training, with the models trained to forecast the next day, week, two weeks, and month's spot prices. This method is advantageous in capturing any time-dependent patterns or trends in the data.

The second approach we used is the randomized search cross-validation technique, which involves exploring the hyperparameters of the ML models. By investigating the most effective hyperparameters for each model, we aimed to achieve optimal performance and minimize the error rate of the models. This approach enables us to train the models on various data subsets and assess their performance using k-fold cross-validation in the training phase.

The fixed rolling window method and randomized search k-fold cross-validation approach offer a comprehensive and robust framework for testing and training spot data. These approaches allowed us to develop more generalized models to accurately

predict future spot prices and be applied to different regions and timeframes.

The training and testing phases were executed in three steps. We split the data into fixed rolling windows based on time for both training and testing. It is important to note that the training set always consisted of past data to predict future observations in the testing set. During the training phase, we utilized either default or tuned hyperparameters, with the latter requiring investigation using an appropriate test of the model validity. In the hyper-tuning process, we employed k-fold cross-validation, which involves splitting the training data into train and validate sets. This approach ensures that the models are not overfitting to the data and can generalize well to new, unseen data. By testing the model on a separate validation set, we can assess its performance on data it has not been trained on, providing a more accurate measure of its generalization ability. We employed the randomized search approach [45] with five folds (K= 5) for each model tuning procedure to ensure robustness and avoid bias. This technique enabled the model to learn from diverse data distributions and construct optimal decision boundaries, as demonstrated in previous studies. For each fold, the data is divided into five splits; four of these are used for model training, and one of the splits is reserved for validation. As illustrated in Figure 3.8, the cross-validation process disregards time since the model is tested with future data (which was held out during the data splitting phase). This approach enables the model to learn from diverse data distributions to construct optimal decision boundaries [6]. Finally, after constructing the best model, we evaluated the models' performance on the testing data.

Training and Testing Datasets

In time series problems, fixed testing and training window sizes of 24 hours for each (1-day ahead) mean that the dataset is divided into consecutive windows, where each window is 24 hours long. The model is trained on the data in one window and tested on the data in the next window, which is also 24 hours long.

For example, in our work, if we have a time series dataset with hourly data over a period of one year, we can divide the data into contiguous 24-hour windows . Each window will contain 24 hourly observations, and we can train a model on the first 24 hours of data and test it on the next 24 hours of data for 40 series in each region. We continue this process for the entire dataset length, where each window overlaps with the previous window by 24 hours since we used the sliding window approach to create training and testing sets. This approach is illustrated in Figure 3.7

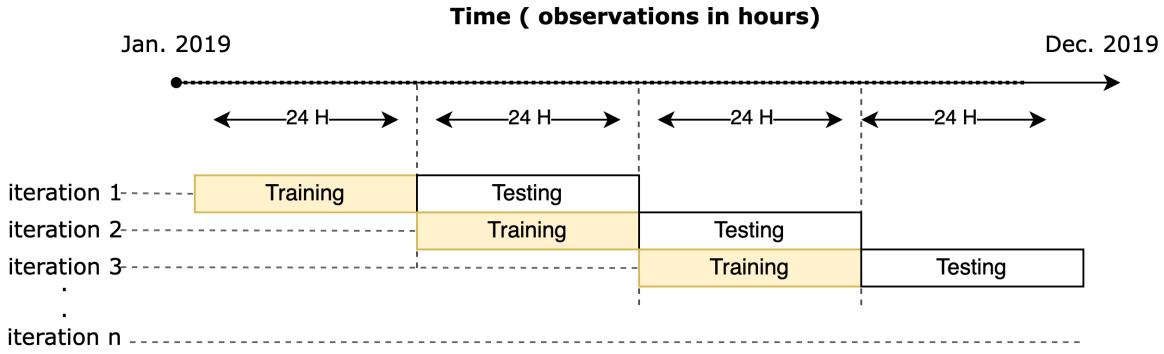


Figure 3.7: One-day ahead testing and training sliding window technique.

Randomized Search Cross-Validation

Randomized search cross-validation is a technique used to explore the hyperparameters of machine learning models to improve their performance. Hyperparameters are settings that can be tuned to optimize the model's performance. The randomized search involves randomly sampling the hyperparameters of a model within specified

ranges rather than exhaustively trying all possible combinations. This allows for a more efficient search of the hyperparameter space [45].

During the randomized search process, the data is partitioned into k subsets or "folds". The model is trained on k-1 folds and validated on the remaining folds. This process is repeated k times, with a different fold used for validation each time. The model's performance is evaluated by averaging the results from the k-fold validation. In our case, we used 5 K-folds to achieve our best performance.

The randomized search approach can help identify the most effective hyperparameters for a given model and dataset by exploring different hyperparameters and evaluating their performance through cross-validation. This can ultimately lead to better performance and more accurate predictions.

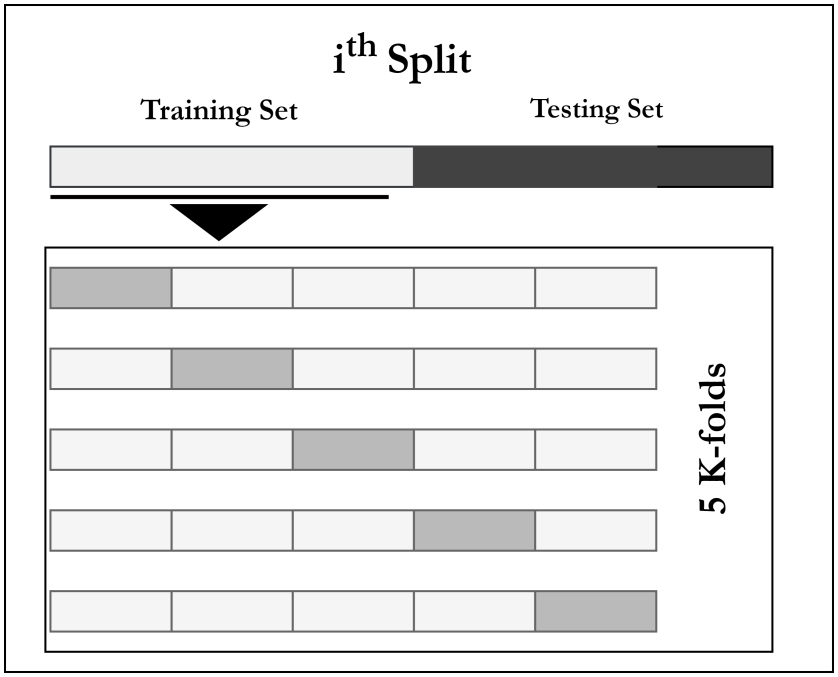


Figure 3.8: Cross-validation Splitting Approach

3.5.1 Hyper-tuning Parameters

The primary aim of hyper-parameter optimization is to attain the best possible performance for a specific algorithm. Since the distribution of data samples can be unpredictable and inevitable, it is necessary for researchers and scientists to continuously explore the hyper-parameters, particularly the hyper-parameter λ associated with the implemented algorithms (i.e., RFR), to reduce model error loss rates. In our study, we used the Randomized Search proposed by James and Benjio [45] to determine the optimal λ for each region yearly.

Although hyper-tuning is widely known for reducing errors and improving accuracy, our five-fold cross-validation in hyper-tuning parameters for all models showed no significant improvements except for the SVR model. Table 3.2 describes our models' hyperparameters and whether it affects results improvement in the predicted prices.

Table 3.2: Tuned Hyperparameters in the Trained Models

Model	Hyperparamter	Description	Default	Affect results?
RFR	max_depth	maximum depth of the tree	None	No
	max_features	Number of features to consider when looking for the best split	1	No
k-NNR	n_neighbors (k)	Number of neighbors to use by default	5	No
	algorithm	Algorithm used to compute the nearest neighbors	auto	No
XGBoost	colsample_bytree	Subsample ratio of columns when constructing each tree.	1	No
	eta	Boosting learning rate	0.3	No
	max_depth	Maximum tree depth for base learners	6	No
SVR	gamma	Kernel coefficient	scale	Yes
	kernel	kernel type to be used in the algorithm	rbf	Yes
	C	Regularization parameter. The strength of the regularization is inversely proportional to C	1	Yes

In this study, we conducted a hyper-tuning process for the SVR model to optimize its performance by adjusting the parameters C and gamma. We found that the RBF kernel was the optimal choice for the SVR model, and we focused on tuning C and

gamma to enhance its performance. It is worth noting that the tuning process must be carefully conducted to avoid degradation in model performance, which would require repeating the search process.

The tuning results, presented in Table 3.3, demonstrate the impact of the hyper-tuning process on the SVR model's performance based on the year and region. In particular, we observed that adjusting C and gamma resulted in improved performance, as indicated by lower RMSE and MAPE values, which are shown and discussed in Chapter 4. We also found that different regions and years required different values of C and gamma, highlighting the importance of tuning the parameters for each dataset.

While hyper-tuning improved the performance of the SVR model, it is worth noting that other models, such as RFR, k-NNR, and XGBoost, are more stable and do not require tuning of their parameters. Therefore, the cost of hyper-tuning can be reduced when using these models. Our study demonstrates the importance of carefully selecting and tuning parameters to optimize model performance and achieve accurate results.

Table 3.3: SVR Hyper-parameters changes in R1 and R5 (2019-2020)

Region	metrics	2019	2020
R1	Kernel	rbf	rbf
	gamma	1	1
	C	1	2
R5	Kernel	rbf	rbf
	gamma	10	1
	C	1	2

The Algorithm 1 shows the training and testing used in our development process.

This algorithm describes a process for training, validating, and testing spot price prediction models. Here is a step-by-step explanation of the algorithm:

The input data is sorted by time using the function "Sort_by_time". A list of years (2019 and 2020) and a list of regions (R1, R2, ..., R5) are defined. A list of time windows (day, week, and month) is defined. For each region in the list of regions:

(1) The input data is filtered by region using the function "Filter_by".

(2) For each year in the list of years:

- The filtered data is further filtered by year using the function "Filter_by".
- For each time window in the list of time windows:
  - The filtered data is split into training and testing sets using the function "Split_data_byTime".
  - A model with default parameters is trained on the training set using the function "Train_Model".
  - A model is tuned using Randomized Search Cross Validation with 5 folds on the training set using the function "Randomized_Search_CV".

(3) A report is generated using the testing set using the function "Generate_report".

Our process is designed to iterate over all possible combinations of region, year, and time window and generate a report on the performance of the models on each combination. The Randomized Search Cross Validation approach is used to find the optimal hyperparameters for the model, and the testing set is used to evaluate the performance of the final model.

Algorithm 1: Process of training, validation, and testing of spot price prediction models

Data: $Data_{Region,Year}$

Result: $Results$

$Data \leftarrow Sort\_by\_time(Data_{Region,Year})$

$Years \leftarrow \{2019, 2020\}$

$Regions \leftarrow \{R1, R2, ..., R5\}$

`/* The Time_Window can be day, week, or month. */`

$Time\_Windows \leftarrow \{day, week, month\}$

foreach $R \in Regions$ **do**

  $Data_r \leftarrow FilterBy(Data_y, R)$

  foreach $Year \in Years$ **do**

    $Data_y \leftarrow FilterBy(Data, Year)$

    foreach $Window \in Time\_Windows$ **do**

      $Data_{Train}, Data_{Test} \leftarrow Split\_data\_byTime(Data_r, Window)$

      $Model_{default} \leftarrow Train\_Model(Data_{Train}, Default_{parameters})$

      `/* Five folds cross-validation is applied using the Random Search approach. */`

      $Model_{tuned} \leftarrow Randomized\_Search\_CV(Data_{Train}, K - folds = 5)$

  $Results \leftarrow Generate\_report(Data_{Test})$

3.6 Experimental Design

In order to train all of the machine learning models, a specific set of features was selected and considered. These features considered in training the models have listed in Table 3.4

By incorporating these features into the training process, the models could better

Table 3.4: Features used to train the models

Feature	**Description**
Month	12 months as January, Febraury, ..., December
Days	7 days as Monday, Tueday,...., etc.
Instance	10 instances (I1,I2,.....,I10)
Operating System	4 operating systems (OS1, OS2,..., etc.)

understand the relationship between these variables and the target variable (the AWS spot instance price), thus improving their predictive capabilities. We build models based on year, window time size, and region, resulting in 160 scenarios for evaluation (2 years x 4-time windows x 5 regions x 4 models). We aim to compare and measure the performance of these models; see Figure 3.9.

3.6.1 Experiment Setup

The study was implemented in Python 3.6, and the code, dataset, and results are freely available as an Eval-EC2Spot-REC in Github[1]. We believe that allowing the source code of our research represents reliable research and allows the bigger community to enhance cloud technologies for our society. Different python libraries were used, including scikit-learn, matplotlib, and slickML, to implement the REC Curve [1, 57, 58]. The device specifications are Intel Xeon(R) CPU @ 3.60GHz, RAM of 32 GB, and Windows 10 Enterprise.

[1]https://github.com/BatoolKad/Eval-EC2Spot-REC.git

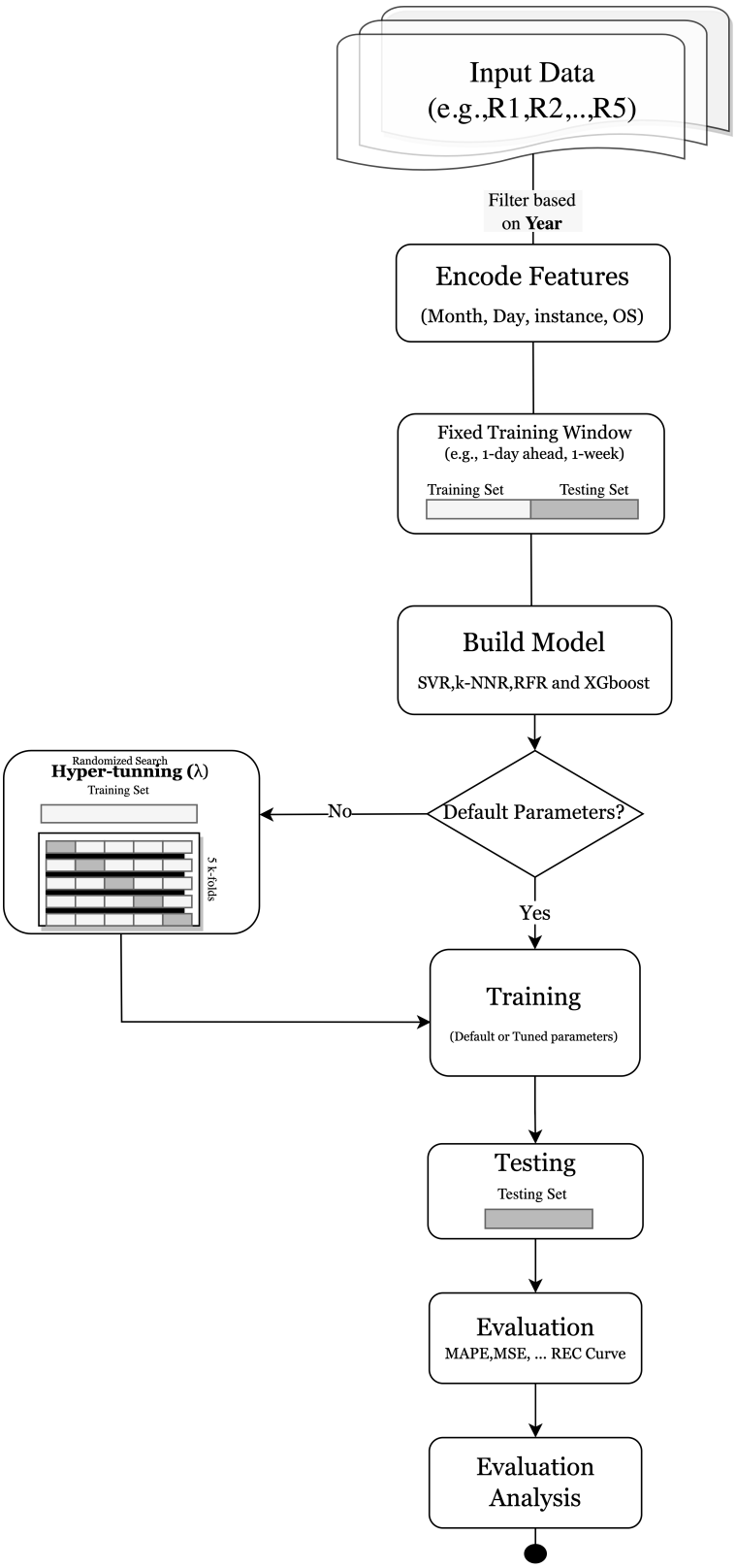


Figure 3.9: The Flowchart of the experiments

3.7 Tuned and Default Models

We compared the performance of default models, which use the pre-set hyperparameters provided by the algorithm developers, to that of tuned models, which have parameters specifically adjusted to suit the data distribution. We used the randomized cross-validation technique to tune the hyperparameters of our machine learning models, which allowed us to search the hyperparameter space efficiently and identify optimal parameter values.

After training and tuning our models, we found that k-NNR, RFR, and XGBoost performed similarly with either default or tuned parameters. However, the SVR model showed a slight performance improvement when its parameters were tuned, depending on the region and timing of the data. This suggests that the SVR algorithm is more sensitive to the hyperparameters than the other algorithms.

Overall, our proposed machine learning algorithms are simple and can handle data fluctuations and changes in price patterns. Tuning the hyperparameters can provide a small performance boost for some models, but it is not always necessary. Default parameters may be sufficient when data is limited or training time is a concern. However, tuning the hyperparameters is recommended for more complex problems or when maximum performance is desired.

3.8 Summary

In this chapter, we present the methodology used in our research, which involves the application of non-parametric supervised machine learning algorithms to address the spot price as a regression problem by minimizing the model's loss errors while training. We also discuss the pre-processing of the data used in our study and assess its stationarity using the ADF and KPSS tests. Then, the experimental design of this

study is presented, including the training, validation, and testing phases. The randomized cross-validation technique is utilized to tune the hyperparameters, and the rolling window method is employed during training and testing to ensure the robustness and accuracy of the models. This chapter provides a comprehensive overview of the methodology employed in our research, except for the evaluation step, which is covered in the next chapter.

In this chapter, we present the methodology we used in our research to predict spot prices using non-parametric supervised machine learning algorithms. The main objective is to minimize the errors in the model's predictions during training by applying different techniques. First, we pre-process the data and evaluate its stationarity using two statistical tests: Augmented Dickey-Fuller (ADF) and Kwiatkowski–Phillips–Schmidt–Shin (KPSS).

Then, the experimental design includes three main phases: training, validation, and testing. During training, we use the rolling window method, which involves training the model on a fixed number of historical data points and then moving the window forward to predict the next data point. This method ensures that the model is robust and accurate in time-series forecasting. Lastly, we tune the hyperparameters while training using a randomized search cross-validation technique, randomly selecting data subsets to train the model and validate its performance. This technique helps prevent overfitting, where the model becomes too specialized to the training data and does not perform well on new data. A final training using default and tuned hyperparameters attractively was done to compare between them.

Overall, this chapter provides a comprehensive overview of the methodology used in the research. However, the evaluation of the model's performance is covered in the next chapter, which will provide more insight into the effectiveness of our approach.

Chapter 4

PERFORMANCE EVALUATION

The process of evaluating prediction models is crucial in both classification and regression tasks. This study assesses the performance of ML regression models. We developed and evaluated 4 models (XGBoost, RFR, k-NNR, and SVR) in the context of SI price predictions. Our models consider various factors that affect the nature of the spot data, including the time for 2 years of observed data, the location of launched instances (e.g., R1:us-east-1a, R3:u-west-2a), and time window size (e.g., day, week, and month). The total number of built and evaluated models is 120 (4 models, 5 regions, 2 years, and 3 window sizes). We use new metrics to evaluate these models that study the trade-off between deviation and accuracy compared to the previous studies that did not consider these factors.

To evaluate the accuracy of our predictions, we need to estimate the **forecast error** using an appropriate evaluation metric. It involves calculating the difference between the predicted and actual price points, with the predicted value being the output of the trained model [11, 41, 49]. It means the error should always be positive since it is the distance between two points, except if we want to know the direction of the error, as negative or positive, to optimize the working models. Various methods can be used to measure forecast error, broadly classified as either scale-dependent

or scale-free metrics [25]. We consider using both types of metrics, as well as the scaled error metric [23]. In addition, we consider the usage for the REC Curve and the area under the REC (AUC-REC) as newly utilized metrics in the field of SI price predictions.

4.1 Forecast Error Metrics Types

Estimating the forecast errors accurately of Spot Instances (SI) price predictions is crucial since it can cause delays in customer work. Choosing the right error metric is vital in indicating forecast prices accurately. In our study, we focus on measuring forecast errors, regardless of their magnitude or distribution, since any error in bidding or setting the maximum price for the spot instance can cause work to fail. The distribution of the errors also affects the evaluation metrics, and squared error, for instance, always penalizes large errors, making them positive. However, it is not useful when errors are small, less than, or equal to 1. Several methods measure this error, largely divided into scale-dependent and scale-free metrics [24, 25] to assess model performance. We consider the usage of both of them. We also use the REC metric, a visualization tool that helps experts study model performance insights. The REC metric and area under the REC Curve (AUC-REC) are newly utilized metrics in SI price predictions and provide a unique perspective on model performance.

4.1.1 Scale Dependent and Scale-free Error Metrics

Scale-dependent errors are metrics affected by the predicted data's scale or data unit (e.g., US dollars, Celsius, and Fahrenheit) [23, 59]. A range of evaluation metrics is considered from this type of metric, including MAE, MSE, and RMSE. These metrics give equal weight to each observation, regardless of the magnitude of the

prediction error. They are useful for evaluating the accuracy of predictions when the scale of the data is important, such as in physical measurements, financial forecasting, or stock price prediction [25]. In this research, the scale of the data is important, but these metrics did not give the right insight in the end due to their limitations and the way they estimate the forecast errors. A detailed explanation of these errors metric is discussed in section 4.2.

On the other hand, scale-free errors are metrics not affected by the scale of the predicted data. Examples of scale-free error metrics include mean absolute percentage error (MAPE) and symmetric mean absolute percentage error (SMAPE). These metrics give more weight to smaller prediction errors and less weight to larger prediction errors. They are useful for evaluating the accuracy of predictions when the scale of the data is less important, such as in predicting proportions or rates of sports outcome prediction [23, 49].

The main difference between scale-dependent and scale-free error metrics is their sensitivity to the magnitude of prediction errors. Scale-dependent metrics are more sensitive to large errors, while scale-free metrics are more sensitive to small errors [23, 60].

4.1.2 Single and Multi-point Accuracy Metrics

Single-point accuracy metrics for regression problems are used to evaluate the performance of regression models as a single averaged value [23, 25, 60]. The most common single-point accuracy metric is Mean Absolute Error (MAE), which measures the average absolute difference between the actual and predicted values [59]. Another popular single-point accuracy metric is Root Mean Squared Error (RMSE), which measures the average magnitude of the error [41]. These metrics can provide valuable information about the accuracy of a regression model for a specific prediction. It can

be used with overall accuracy metrics, such as R-squared, to get a complete picture of a model's performance.

Multi-point accuracy metrics are evaluation metrics used in regression problems to measure the accuracy of predictions across all points of the target variable concerning their error threshold [21]. Unlike single-point metrics, multi-point metrics comprehensively evaluate model performance by considering the overall distribution of errors across multiple points in the target variable and visualizing it. The regression error characteristic curve (RECC) plots the cumulative distribution of errors between the predicted and actual values, providing a visual representation of how the errors are distributed across the target variable. This can help to identify areas of weakness in the model, as well as to identify potential biases in the predictions. In general, multi-point accuracy metrics provide a more comprehensive evaluation of model performance than single-point metrics. They can help identify areas of weakness in the model and inform the decision-making process for model selection and optimization.

4.2 Model Performance Evaluation Metrics

We evaluate the price predictions of 120 ML models using the four algorithms XGBoost, RFR, k-NNR, and SVR. We consider the same criteria of development in the evaluation process. The considered data criteria were location and time effects. At the same time, the training measures are considered as the training window, hyperparameters tuning effects. We applied the most used evaluation metrics from the related work: MAE, MSE [13], RMSE [19], and MAPE [17, 38, 52]. In addition, we used the REC Curve and AUC-REC as the primary metrics in evaluating the prediction models.

To understand how each metric works or how to calculate errors using a specific

metric, we explain the operational principle and the mathematical formula that explains each metric. Thus, the explanation of each metric is provided below to help understand how it measures forecast errors and its strengths and limitations. Consequently, use it for evaluating the SI price prediction models.

4.2.1 Mean Absolute Error

The Mean Absolute Error (MAE) is a popular evaluation metric employed in regression models to assess the accuracy of predictions [41]. It is defined as the average of absolute differences between the predicted and actual values, making it a simple and easy-to-interpret error measure. The mathematical formula is shown in the equation 2 below:

$$MAE = \frac{\Sigma_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2}$$

where n is the number of observations, y_i is the true value of the ith observation, and $\hat{y}_i$ is the predicted value for the ith observation. The absolute value function $||$ ensures the errors are positive.

Despite its simplicity, MAE has a drawback in that it lacks information about the direction of the error [59,61]. This can result in an imperfect evaluation of the model's performance, particularly when forecast errors are relatively small. Averaging small values can change their interpretation by masking important information in the data. For instance, if we have a set of errors where most of the values are large and a few values are small, then averaging all the values together can hide the fact that a few small values may be significant in some way.

4.2.2 Mean Squared Error

The Mean Squared Error (MSE) is a widely used metric in regression models to evaluate the accuracy of predictions. It measures the average of the squared differences between the predicted values and the actual values. The main advantage of using MSE is its mathematical properties, which provide a more comprehensive analysis of the error between the predicted and actual values. However, this advantage has a drawback: MSE is sensitive to data distribution [23]. When the target variable has a high proportion of large values, MSE tends to heavily penalize large errors, which can result in a distorted evaluation of model performance [62]. This sensitivity to data distribution can make MSE less suitable for modeling certain problems, such as skewed or non-normal data distributions, where outliers can significantly impact the MSE value. The equation of MSE is listed below:

$$MSE = \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{3}$$

where n is the total number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

4.2.3 Root Mean Squared Error

The Root Mean Squared Error (RMSE) is a widely used evaluation metric for regression models. However, it has a major weakness: it is highly sensitive to data distribution [23, 62]. RMSE estimates the difference between the predicted and actual values and calculates the average of these differences. This makes it vulnerable to outliers, which can significantly impact the overall score and make it difficult to compare different [49]. Additionally, RMSE does not provide information about the

direction of the errors, making it difficult to identify patterns or biases in the predictions [59,61]. As a result, it is important to consider the distribution of the data and the presence of outliers when using RMSE as an evaluation metric. Furthermore, using other metrics, such as mean absolute error or median absolute error, is advisable, which are less sensitive to the data distribution and provide a more comprehensive view of the model's performance.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{4}$$

where n is the total number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value. Note: the equation under the root is the MSE equation. See equation (3).

4.2.4 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) is a favored measurement utilized in regression models described in the related work Chapter 2, section 2.2 to determine the precision of predictions. It calculates the percentage difference between the actual and predicted values and takes the average of the absolute values of these differences [63]. However, MAPE has a considerable drawback as it is prone to be impacted by data distribution. When the target variable has a high proportion of zero or near-zero values, the percentage error becomes undefined or extremely large [23], leading to a distorted evaluation of the model's performance, as the metric fails to consider the

actual size of the error.

$$MAPE = \frac{1}{n} * \sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i}| * 100\% \quad (5)$$

4.2.5 REC Curve and AUC-REC

The Regression Error Characteristic Curve (REC Curve) is a technique for visualizing and comparing regression results, such as the Receiver Operating Characteristic (ROC) Curve for classification models. While the AUC-REC is a single scalar representing the performance associated with every point on the REC curve, it gives a comprehensive understanding of the accuracy along each error tolerance rate [21].

The ROC is an evaluation metric that is used to visualize classifier performance based on true positive rate (TPR on the y-axis) versus false positive rate (FPR on the x-axis) [22]. Researchers utilize the ROC curve to evaluate model performance at various thresholds providing visualization of the detector performance and the selection of optimum operating points without committing to a single decision threshold [22]. However, the ROC is not adapted to visualize the performance of regression models because the regression results are continuous values. Bi and Bennett [21] proposed REC as a visualization metric for regression models.

The REC curve is drawn with points based on 2 values (absolute deviation on the x-axis and accuracy on the y-axis) as shown in Figure 4.1. The range of the y-axis is between [0, 1] intervals, while the x-axis range could be modified based on the problem. We adopted absolute deviation error as the optimal loss function [21] to determine the deviation on the x-axis, with a threshold of 1 for the highest approximated error. On the contrary, the y-axis represents the proportion of points that fit inside the error tolerance known as accuracy. The value is optimal until the top left corner approaches 1 for accuracy and 0 for deviation. At this point, the REC

curve attains its highest accuracy and lowest deviation. In comparison, the bottom right corner achieves 0 accuracy's and 1 absolute deviation, which means the worst value on the REC curve [22]. The diagonal line in the middle represents the 0.50 for each point values of deviation and accuracy (also known as a random model).

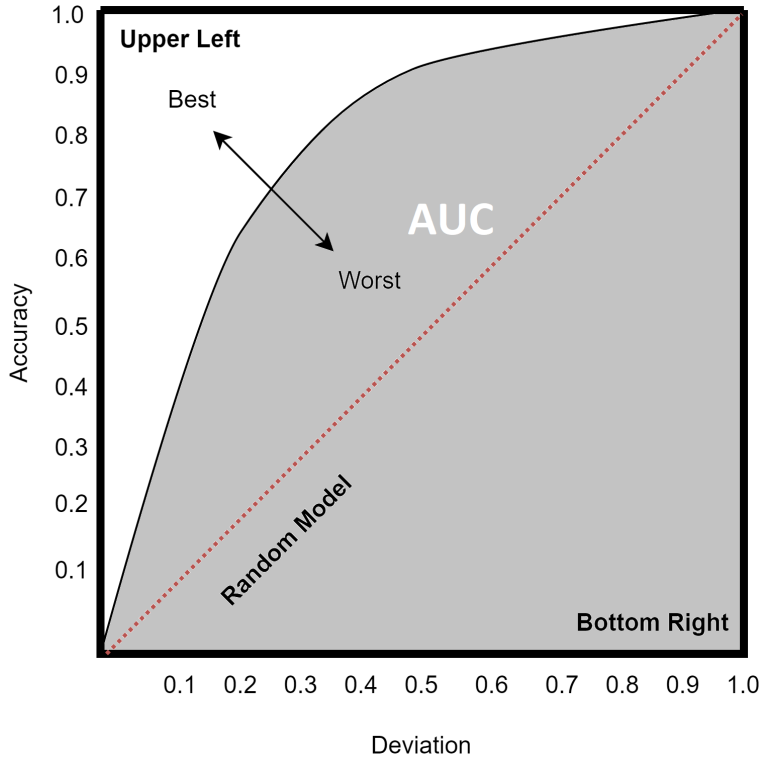


Figure 4.1: Regression Error Characteristic (REC) interpretation

As a result, the best model performance is the one that draws the nearest point to the upper left corner of the REC curve. Finally, we calculate the area under the curve to represent all these points from the REC curve as a single value called the model's accuracy (AUC-REC). Equation 6 used the Simpson method as an integral method implemented in [64] to estimate the cumulative distribution function of the error as a curve [21].

$$AUC = \int_0^{\epsilon} F_0(x) f_1(x)\, dx \tag{6}$$

The AUC is a single statistic that measures expected performance as accuracy with well-understood statistical meaning.

4.2.6 Summary of Evaluation Metrics

The error metrics discussed earlier are designed to estimate errors by aggregating them using an average of all errors, including MSE, MAE, RMSE, MAPE, and AUC-REC. However, one error is associated with each predicted point calculated in all metrics: the deviation or distance between the actual price point y_i and the predicted value $\hat{y}_i$ [23, 25, 60]. Therefore, relying on the overall errors based on a single point can be unreliable and open to debate, as noted in previous studies [25, 41, 61]. To address this issue, the REC Curve visualizes the trade-off between errors for each prediction [21, 65]. This allows a more accurate estimation of the prices and their associated errors. The REC Curve is a corresponding metric of the Receiving Operating Characteristics Curve (ROCC) used in classification.

4.3 Evaluation Metrics Insights

The metrics' efficiency required an investigation of all metrics outcomes using their error distribution and how it reflected in determining the models' performance. Thus, we used the Boxplot [57] as an illustration tool to analyze the prediction forecast errors using exploratory data analysis, providing a visual representation of the distribution of forecast errors using all utilized metrics and allowing for the identification of potential outliers and comparisons between datasets.

Boxplots, also known as box-and-whisker plots, are useful data visualization tools for displaying the distribution of a dataset. They are particularly useful for identifying outliers and comparing multiple datasets' distribution.

Earlier in section 4.2, we outlined the metrics used to evaluate the Spot price predictions. We ordered the metrics with the lowest insight to the highest based on their forecasted error distribution in the four utilized models: MSE, MAE, RMSE, MAPE, AUC-REC, and REC Curve.

4.3.1 MSE Evaluation Insights

Figure 4.2 displays the errors estimated using MSE for three different training windows in one region for 2019. It can be observed that all models perform similarly, except for SVR, which has outliers when predicting with a 1-day window. However, MSE is not a good metric for evaluating the performance of models in SI price predictions. The MSE errors for models XGBoost, RFR, and k-NNR are underestimated, with a median and quintile of zero for the Boxplots. These models reflect errors based on the squared specialty, but when the loss is less than 1, the loss exponentially decays until it reaches zero. The same issue is in Figure 4.3 for data from 2020. The MSE shows that all models are performed with low error rates.

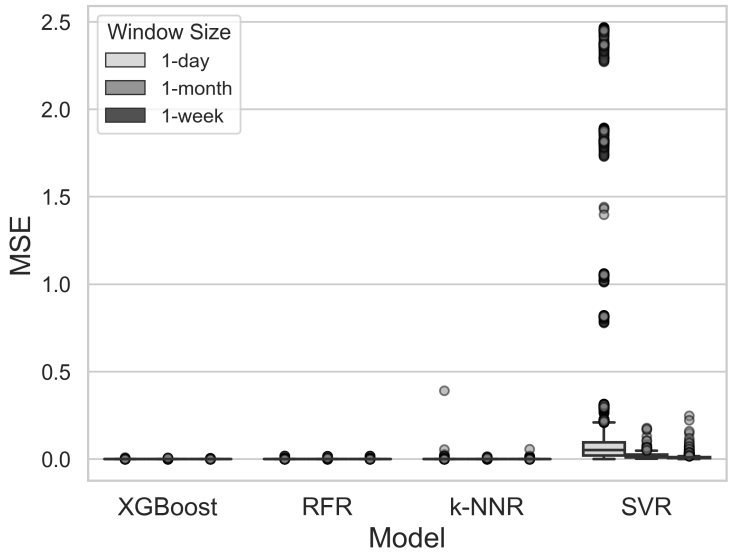


Figure 4.2: The distribution of MSE errors in region R1 (2019).

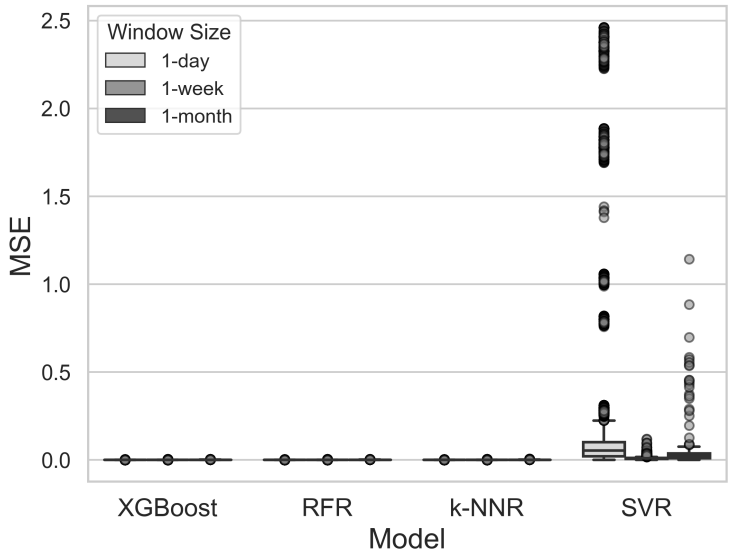


Figure 4.3: The distribution of MSE errors in region R1 (2020).

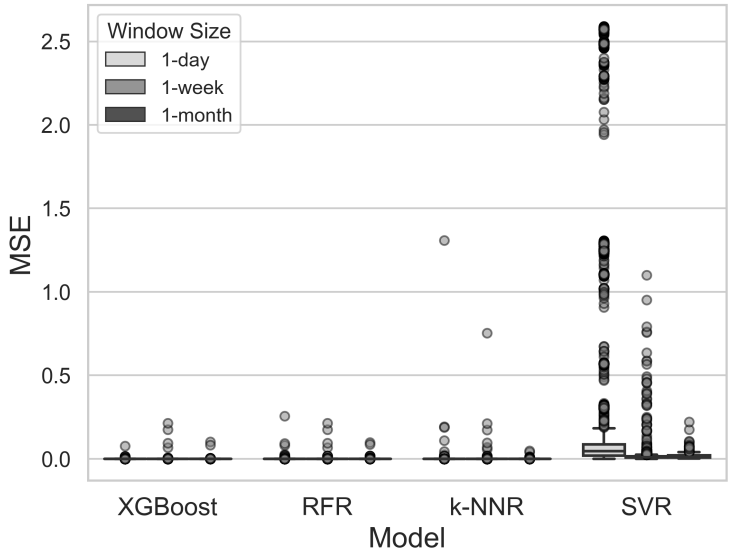


Figure 4.4: The distribution of MSE errors in region R3 (2019).

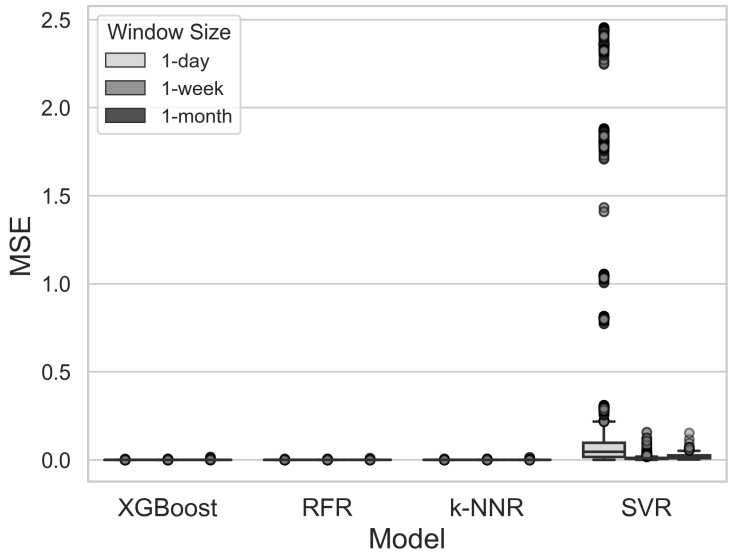


Figure 4.5: The distribution of MSE errors in region R3 (2020).

4.3.2 MAE Evaluation Insights

The MAE results for the four models in R1 and R3 regions for 2019 and 2020 are presented in Figures 4.6, 4.7, 4.8, and 4.9. MAE in 2019 4.6 and 4.8 show the four prediction models: XGBoost, RFR, and k-NNR with an error rate less than 0.2 and 0.4, respectively, except for k-NNR in 1-day has one outlier in the year. At the same time, errors illustrated in figures 4.6 and 4.7 shows the same for the aforementioned models with an MAE less than 0.1.

Except for the SVR, the MAE boxplot figures do not show a significant difference between the training window sizes for the aforementioned models, as noted in the MSE results. The SVR is sensitive to the distribution and density of the data, which makes them produce bad results, especially when the training window is small. The model cannot generalize or learn the data to generate accurate results.

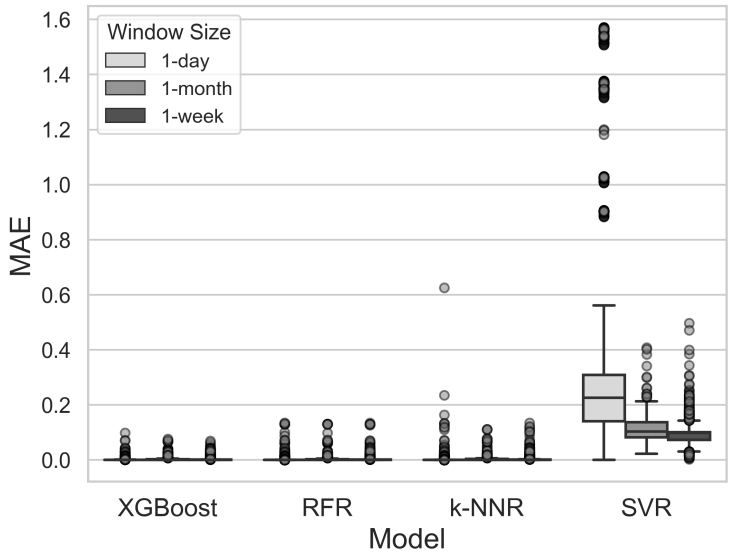


Figure 4.6: The distribution of MAE errors in region R1 (2019).

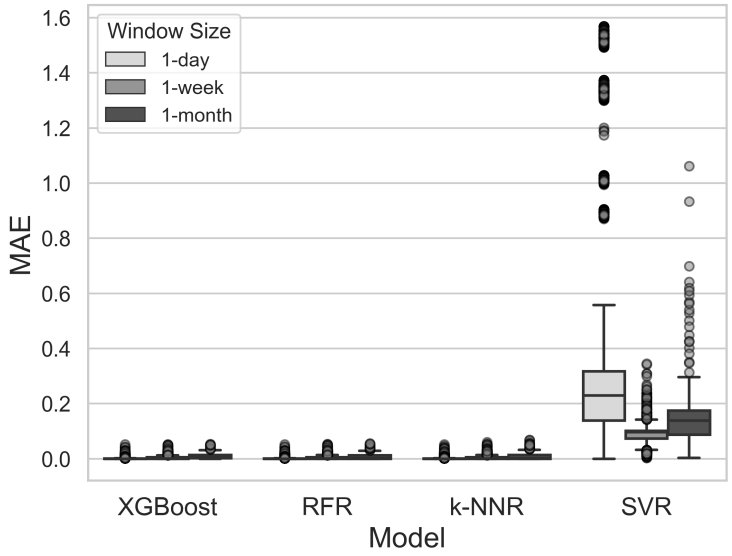


Figure 4.7: The distribution of MAE errors in region R1 (2020).

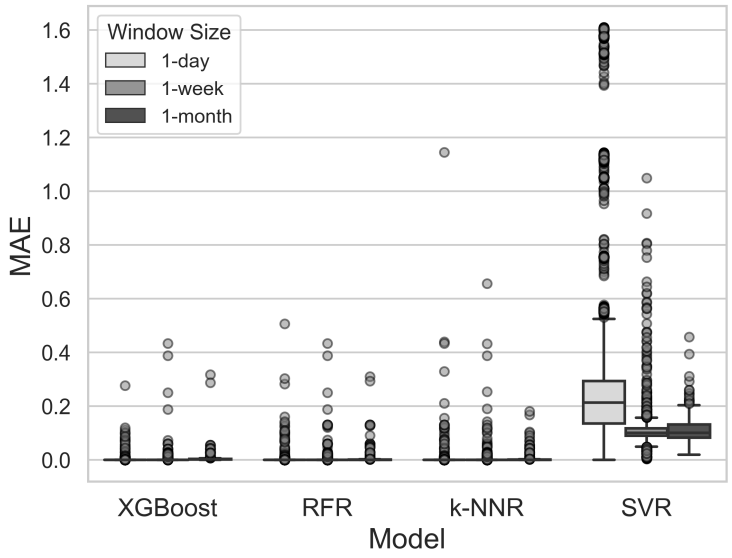


Figure 4.8: The distribution of MAE errors in region R3 (2019).

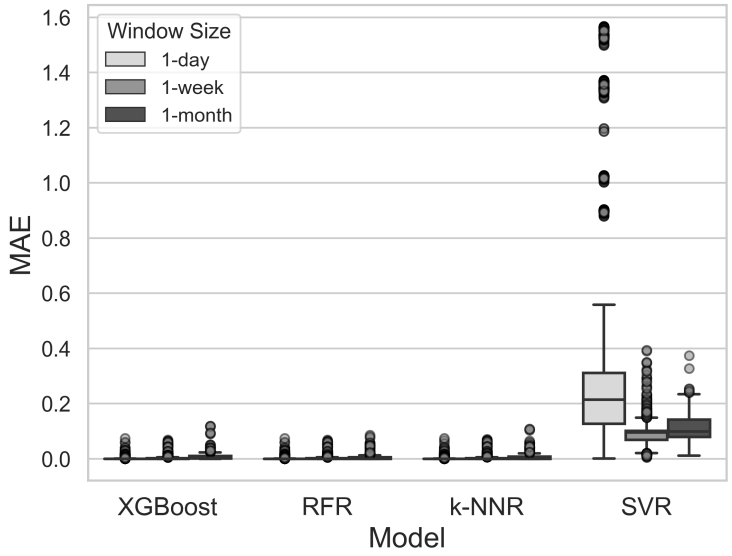


Figure 4.9: The distribution of MAE errors in region R3 (2020).

4.3.3 RMSE Evaluation Insights

Figures 4.10, 4.11, 4.12 and 4.13 depict the errors in RMSE for regions R1 and R3. RMSE is considered a superior metric to MSE in assessing the model performance. The findings of the RMSE are similar to those of MAE, as discussed in section 4.3.2.

The models XGBoost, RFR, and k-NNR demonstrate the best performance, with a few more outliers observed for the k-NNR model. Conversely, the three figures of RMSE show the SVR model has the poorest performance, like MSE and MAE insight.

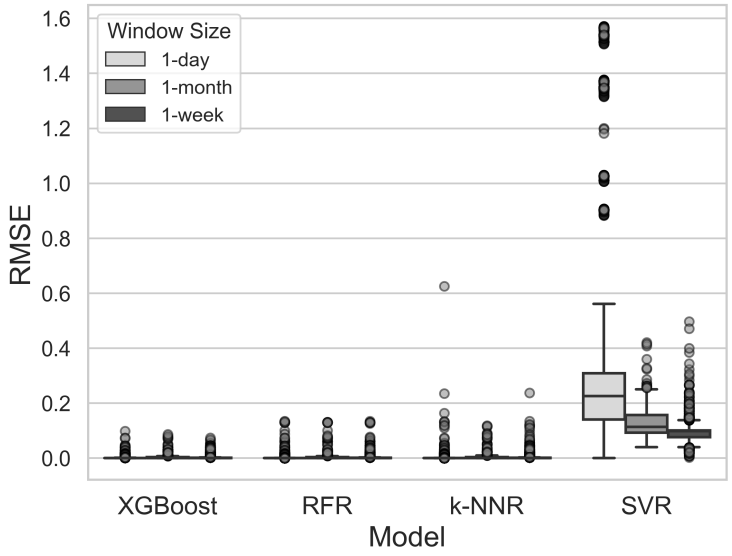


Figure 4.10: The distribution of RMSE errors in region R1 (2019).

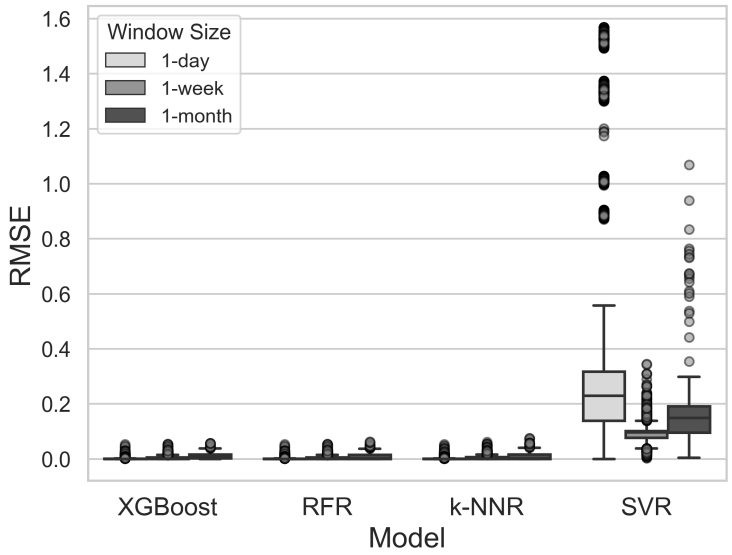


Figure 4.11: The distribution of RMSE errors in region R1 (2020).

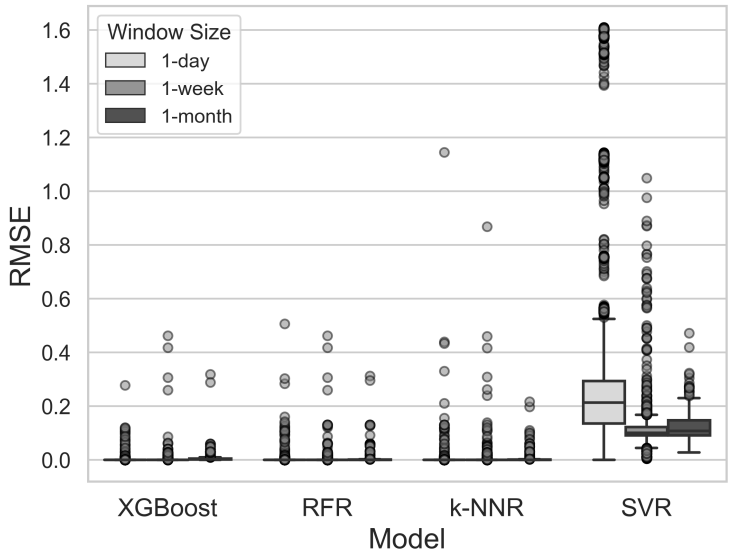


Figure 4.12: The distribution of RMSE errors in region R3 (2019).

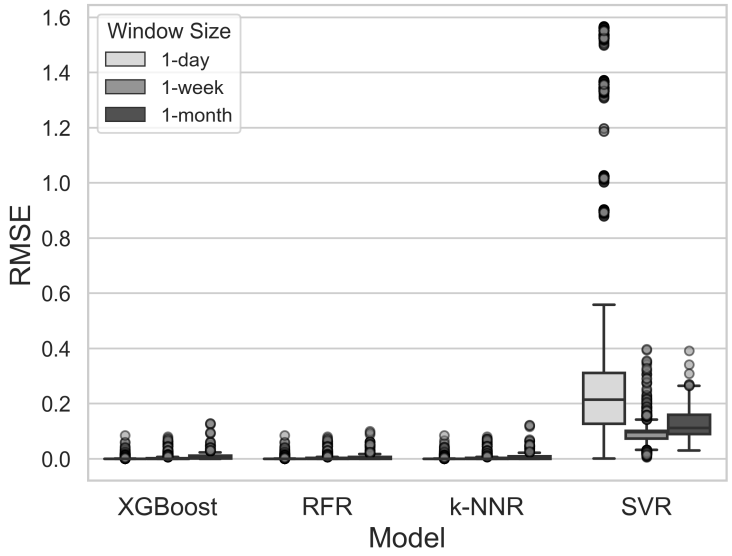


Figure 4.13: The distribution of RMSE errors in region R3 (2020).

4.3.4 MAPE Evaluation Insights

Figure 4.14, 4.15, 4.16, and 4.17, illustrate the percentage of errors observed in 2019 and 2020, R1 and R3 respectively. In 2019, the first three models in the figures depicted an $MAPE$ that is less than or equal to 2%, except for the k-NNR model,

which shows an $MAPE$ of less than or equal to 7%. However, the SVR model continues to exhibit the poorest performance, with the best results observed in the 1-week ahead window. Moreover, in Figure 4.16, which relates to the R3 region, the k-NNR model displays a significant proportion of high outliers, with an $MAPE$ greater than 40%. On the other hand, the XGBoost model shows the most reliable and consistent performance.

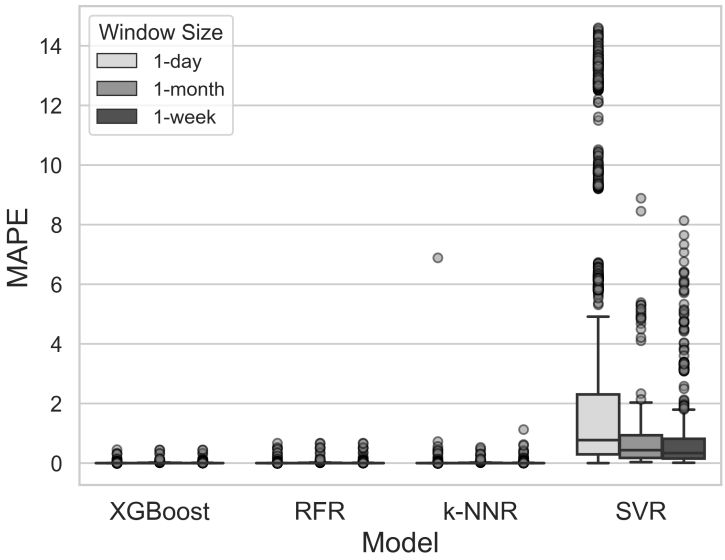


Figure 4.14: The distribution of MAPE errors in region R1 (2019).

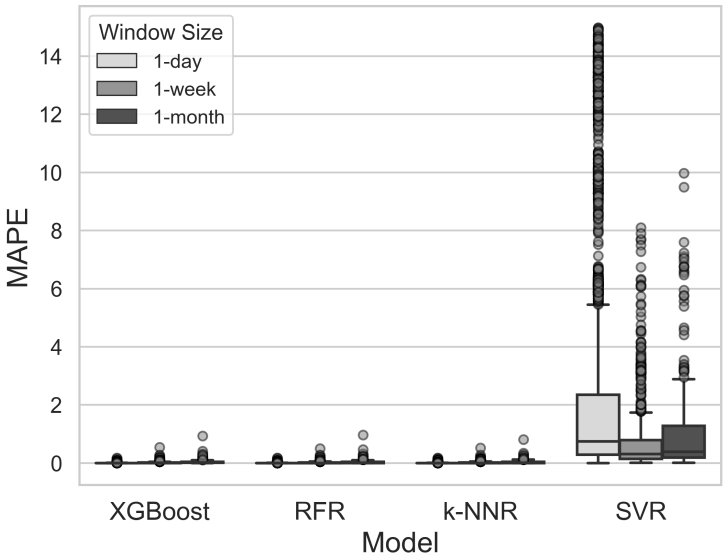


Figure 4.15: The distribution of MAPE errors in region R1 (2020).

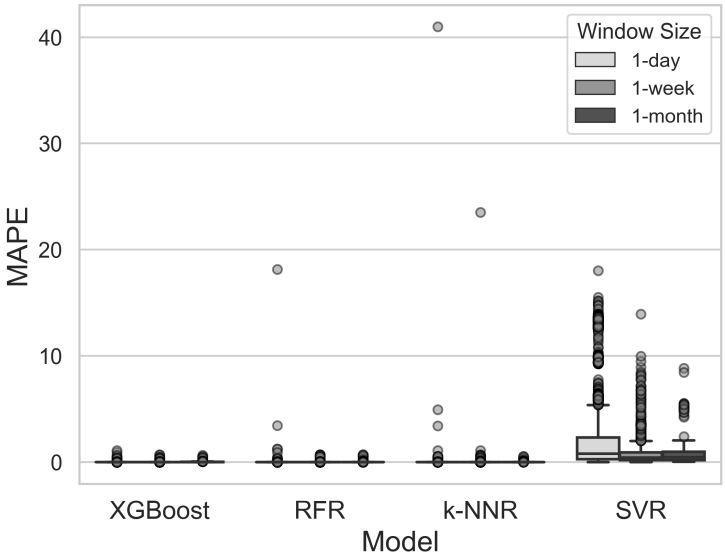


Figure 4.16: The distribution of MAPE errors in region R3 (2019).

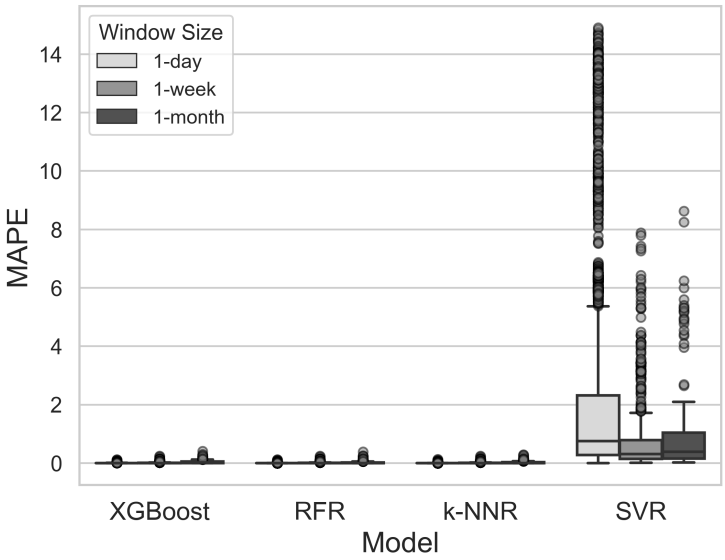
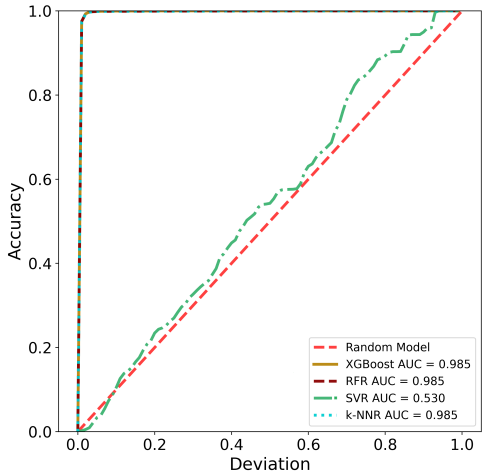


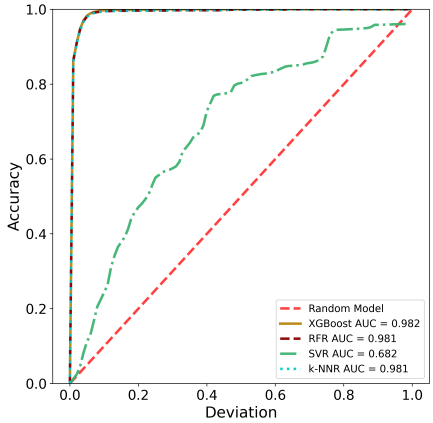
Figure 4.17: The distribution of MAPE errors in region R3 (2020).

4.3.5 AUC and REC Curve Evaluation Insights

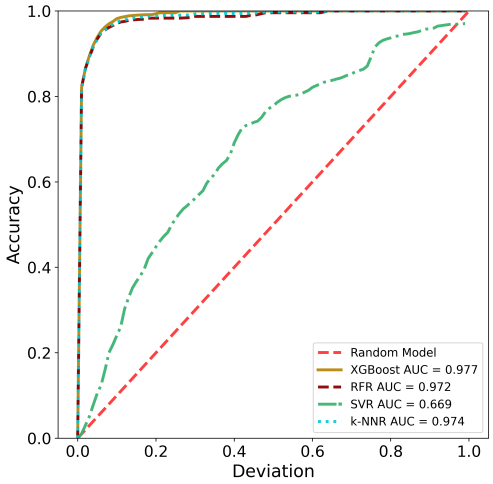
The AUC-REC metric calculates the area under the REC curve, whereas the REC curve gives us more insight into the model performance. As indicated previously, the REC visualizes accuracy with different loss thresholds. Figures 4.18a, 4.18a 4.18a visualize the REC curve of 4 models based on the window size (day, week, and month, respectively) for the year 2019 and region R1. Also, Figures 4.19a, 4.19b and 4.19c display the REC of 4 models of the year 2020. The model performance is better with a window size of 1-day for all models except SVR when we used the 1-day time window in both years (2019 and 2020). On the other side, the performance of models (RFR, k-NNR, and XGBoost) is degraded by 1% of AUC-REC when we used a 1-month window size to train the models. The AUC-REC increased by 14% for SVR when trained using 1-month. All the REC curves are plotted using a series of accuracy and deviations. We can easily represent these values as a single curve to compare models' performance.



(a) REC Curve of 1-day time window (R1/2019)

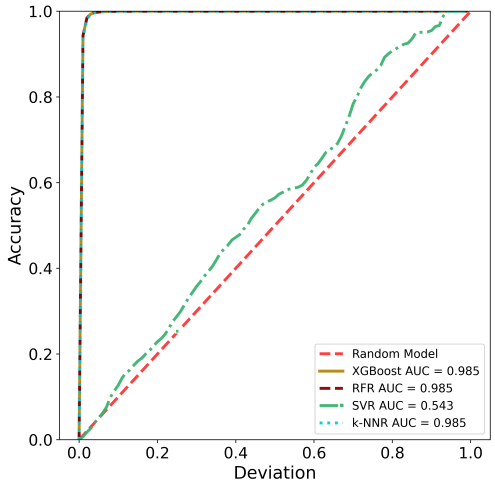


(b) REC Curve of 1-week time window (R1/2019)

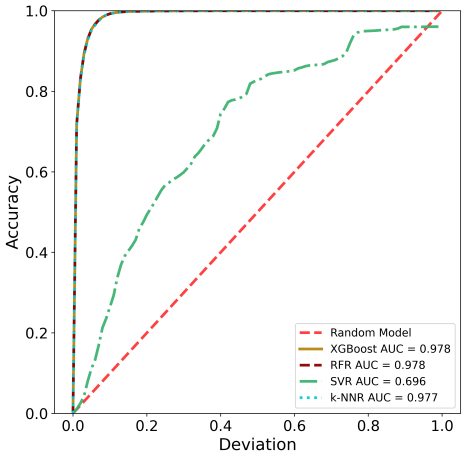


(c) REC Curve of 1-month time window (R1/2019)

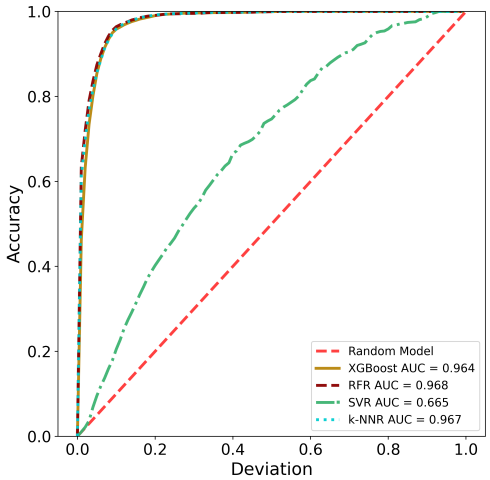
Figure 4.18: The REC visualization of 2019 (R1)



(a) REC Curve of 1-day time window (R1/2020)



(b) REC Curve of 1-week time window (R1/2020)



(c) REC Curve of 1-month time window (R1/2020).

Figure 4.19: The REC visualization of 2020 (R1)

To this end, we illustrated the AUC-REC accuracy of all models using the Box-plots. Figures 4.20 and 4.22 pictured the AUC-REC of 4 models in 2019. The top three models achieve the best AUC-REC with slight variance in XGBoost, while the performance of RFR and k-NNR exhibited high variance with AUC less than 0.6 for some points. In comparison, the SVR model has a median of 55% to 74% with a high variance of results meaning that the model is not confident about the results. The results of 2020 give the same insight with more robust performance. Figures 4.21 and 4.23 show top models with the finest accuracy for a 1-day training window.

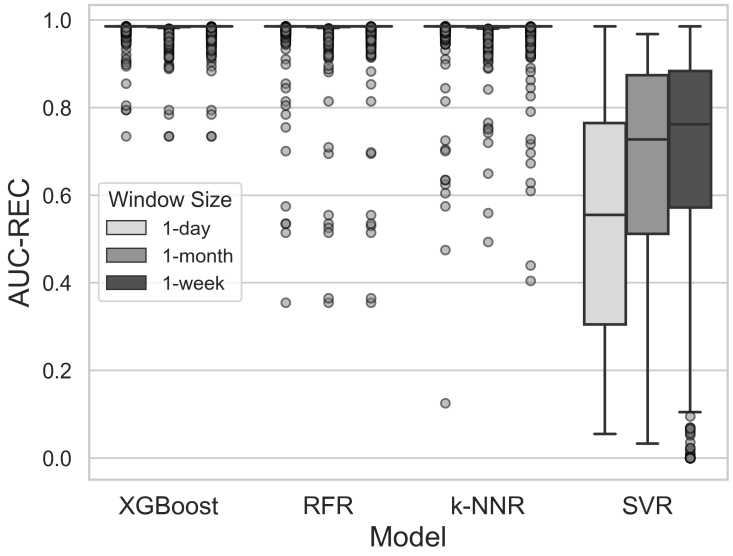


Figure 4.20: The distribution of results for AUC-REC metric of region R1 (2019).

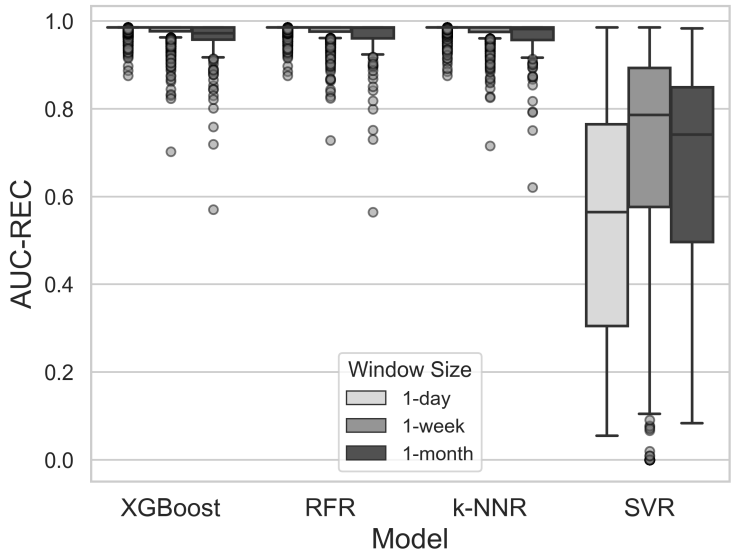


Figure 4.21: The distribution of results for AUC-REC metric of region R1 (2020).

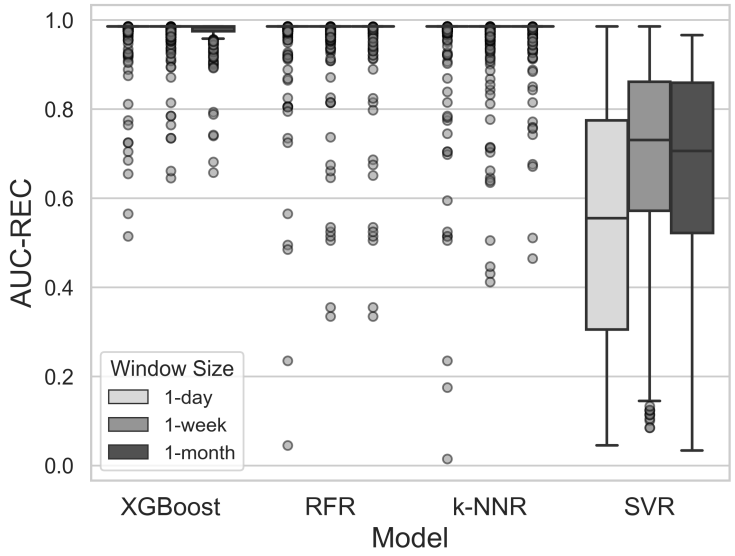


Figure 4.22: The distribution of results for AUC-REC metric of region R3 (2019).

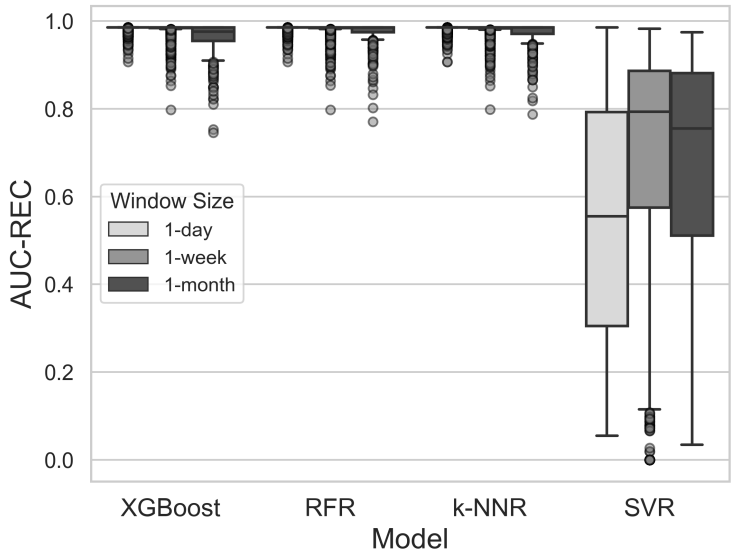


Figure 4.23: The distribution of results for AUC-REC metric of region R3 (2020).

4.3.6 Summary

As shown previously, the MAPE is a widely used metric that has been used for measuring the percentage of errors in regression problems [62]. However, it can not be used if the results contain zeros or values close to zero because it can result in infinite or undefined values; the range is between $[0, \infty)$ [23, 66]. As shown in Figures 4.15 and 4.17, the range of results has zero values for all models except SVR. Moreover, the results of the SVR model contain a high rate of outliers. This issue occurs because the MAPE metric puts a higher penalty on negative errors [26]. As a result, using MAPE is critical when we move the trained model to production. Producing infinite or undefined results can raise an exception (e.g., memory overflow) and problematic understanding of the model results.

Figures 4.2, 4.6, and 4.10 show the overall statistics of RMSE, MAE, and RMSE errors, respectively. The MSE is the worst metric for evaluating errors. The intuition given using the MSE directed by the squared unit used while prediction means the MSE values interpreted in our study as squared US dollars which is not comparable

to the real price values in US dollars. Figure 4.3 illustrated the errors of the year 2020. Since the SVR model is sensitive to the data distribution, it has more outliers.

Both MAE and RMSE are used to evaluate the price prediction model accuracy. Again, the range of these metrics lies within the range $[0, \infty)$. As a result, interpreting the model predictions is not an easy task for normal users. The MAE has more reliable results, as shown in Figure (4.6, 4.7, 4.9 and 4.8), are displaying the SVR as the worst model. The main issue of MAE is confidence in predictions. The variance of results of SVR in Figure 4.6 is smaller, which means that the SVR model has high confidence to predict price accurately, but this is incorrect. Different data distributions hinder the performance of the SVR model, which has the lowest performance.

Nevertheless, Figure 4.2 presents that all models have the same performance measured by MSE, while the SVR has lower accuracy than the other models. All prior metrics depend on the data's dimensions except for MAPE, a scale-free metric [23]. REC is an independent metric used to evaluate predictions with different thresholds regardless of the type of distributions [21]. In addition, the AUC-REC is an accuracy metric used to evaluate the performance with a range between [0,1], making it the only employed metric with a predefined range that could be interpreted efficiently by anyone.

4.4 Location and Timing Effect

In this section, we limit our discussion to location and timing as factors that affect the forecast error regardless of the features used while training. Each value in this table is related directly to one prediction point, respecting both instance type and operating system features. First, we will discuss the timing effect concerning the training window based on the MinMax Table 4.1; the location along timing effect is observed based on the Boxplot from section 4.2. Line plot of average prices and

predictions provided in the previous section 4.4.

Table 4.1 shows the minimum and maximum AUC-REC values of 2 years (2019 and 2020) of overall trained models in region R1. The rows represent the AUC-REC based on years, while the columns display results based on time windows (1-day, 1-window, and 1-month). These values are associated with one observation of the test predictions. Mostly in three training window sizes, the highest AUC-REC in both years is 0.986, recorded by XGBoost, RFR, and k-NNR. In k-NNR, the minimum value was the lowest, 0.125, for these models in 2019. While in 2020, the minimum was observed for 1-month in RFR as 0.564. Overall the results were more stable, with a minimum value of 0.875 in the top three models, in a 1-day training window. The SVR model recorded the worst results, achieving the lowest AUC-REC value of zero in 2020.

Table 4.1: The Min-Max mean Values of AUC-REC in R1 (us-east-1a)

Window	**AUC-REC**	**2019**				**2020**			
		XGBoost	RFR	k-NNR	SVR	XGBoost	RFR	k-NNR	SVR
1-day	**Min**	0.735	0.355	0.125	0.055	0.875	0.875	0.875	0.055
	Max	0.986	0.986	0.986	0.935	0.986	0.986	0.986	0.945
1-week	**Min**	0.735	0.335	0.403	0.000	0.702	0.728	0.716	0.000
	Max	0.986	0.986	0.986	0.986	0.983	0.986	0.986	0.986
1-month	**Min**	0.735	0.355	0.493	0.055	0.570	0.564	0.620	0.084
	Max	0.986	0.986	0.983	0.945	0.983	0.986	0.986	0.938

The location effect is evident when comparing the results of R1 and R3 regions for all evaluation metrics. Where the location effect has its effect while tuning the hyperparameters of the SVR, see Chapter 3 section 3.5.1. We found the price tended to fluctuate less in the 5 regions in 2020. The timing effect pertains to the variation in forecast errors across different years or time periods. While Figures 4.6 and 4.7 show a slight difference in MAE forecast errors between 2019 and 2020, the figures for Region R3, i.e., Figures 4.8 and 4.9, demonstrate the impact of location on forecast errors. Specifically, when comparing Figures 4.6 and 4.8 for the year 2019, the performance

of the models is inferior in Region R3 due to the presence of noisy data. Furthermore, the analysis indicates that the XGBoost model is top-performing when considering the 3-window training.

4.5 Training Windows Results

All figures in the discussion section studied the effect of the training window in predicting results. The overall conclusion from the metrics insight, especially the AUC-REC, indicated that using different training windows affects the model's performance differently. In addition, figures of the real price plot vs. the prediction in the location and timing section 4.4 demonstrated how the training window size affects the models' performance. Figures 4.24 and 4.25 display the average of real prices and predictions for all models with windows day, week, and month, respectively.

4.5.1 One-day ahead

Figures 4.24a and 4.25a show the prices and predictions of all models using a 1-day time window in regions R1 and R3. The models XGBoost, RFR, and k-NNR have the lowest error rates on their predictions compared to the real price in 2019 and 2020. Conversely, SVR could not identify the proper bounds for predicting the actual price with a minimal error rate. In contrast, the SVR shows the worst results when using the 1-day window, with almost random prediction behavior as shown in Figures 4.24a and 4.25a.

4.5.2 One-week ahead

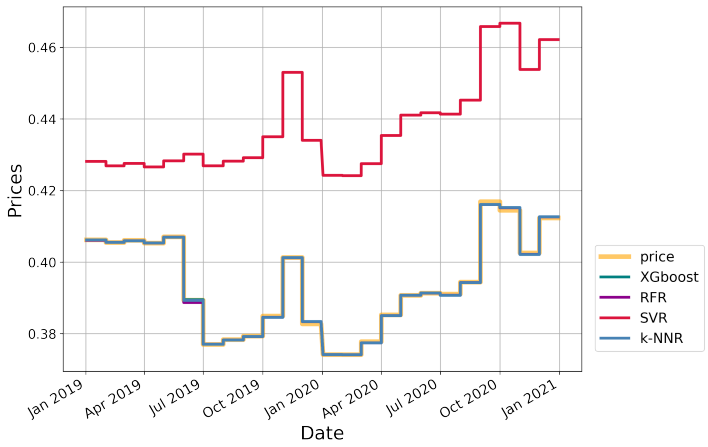
Figures 4.24b and 4.25b visualize the model's predictions and actual prices in regions R1 and R3. The performance of XGBoost, RFR, and k-NNR decreased,

producing more errors with the actual prices. On the other hand, the SVR model performs better than the 1-day and 1-month window sizes. The SVR overlaps on the months (of June (2019) and December (2020), scoring the highest accuracy of AUC-REC at 0.986, see table 4.1.

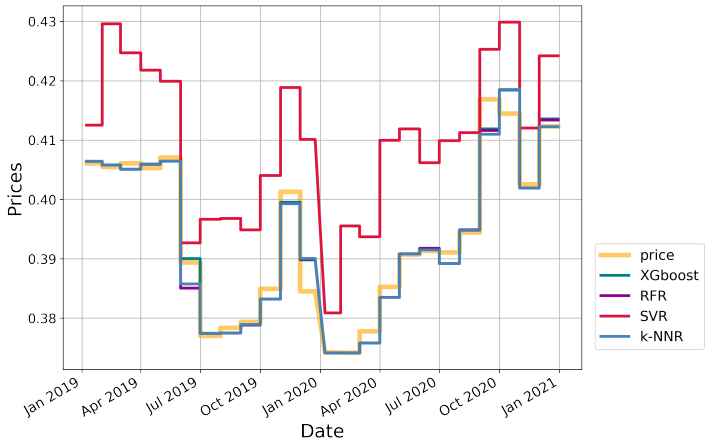
4.5.3 One-month ahead

Figures 4.24c and 4.25c show the prices and predictions with all models trained using a 1-month time window. The forecast error increases for all models with larger time windows. Although, this difference between real prices and predictions is fairly small in XGBoost, RFR, and k-NNR models. They perform better than SVR in all window sizes, including the 1-month ahead. On the other hand, the SVR model has a better average prediction compared to the 1-day ahead. The best SVR fitting is between the period of June 2019 to July 2019. These results can be recognized in Table 4.1, where the maximum AUC-REC 0.945 is in 2019 using 1-month.

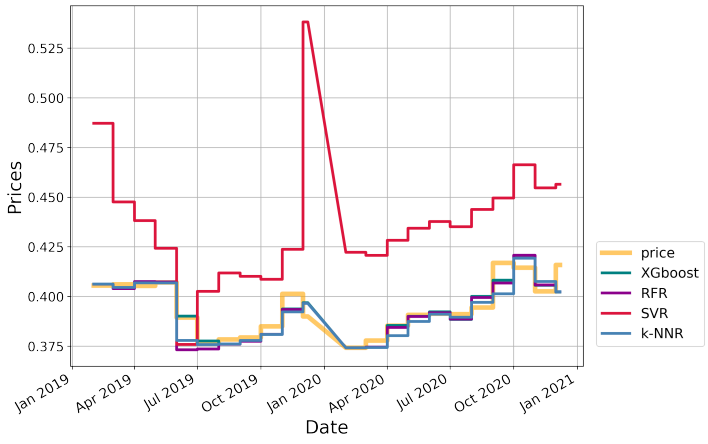
We conclude that studying window sizes directly affects the model's performance. The 1-day ahead window is the best, with predictions reaching 0.986 AUC-REC accuracy, in three models, where the 1-week is the best for the SVR model with 0.986 as the highest AUC-REC accuracy.



(a) 1-day window predictions vs. actual prices

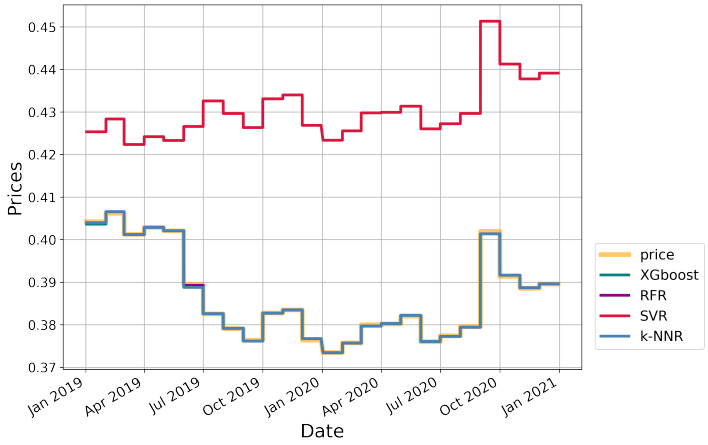


(b) 1-week window predictions vs. actual prices

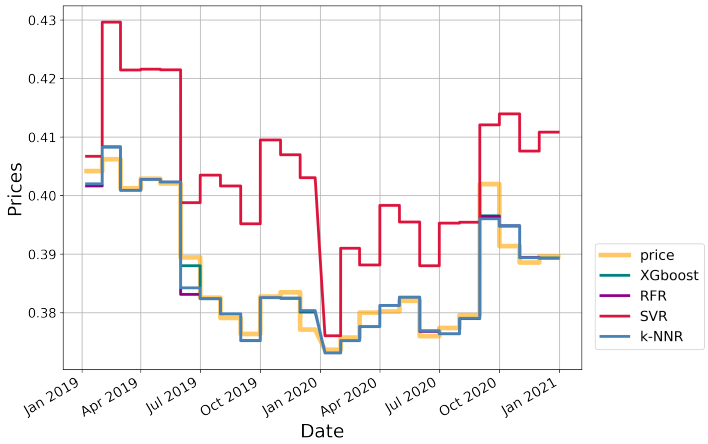


(c) 1-month window predictions vs. actual prices

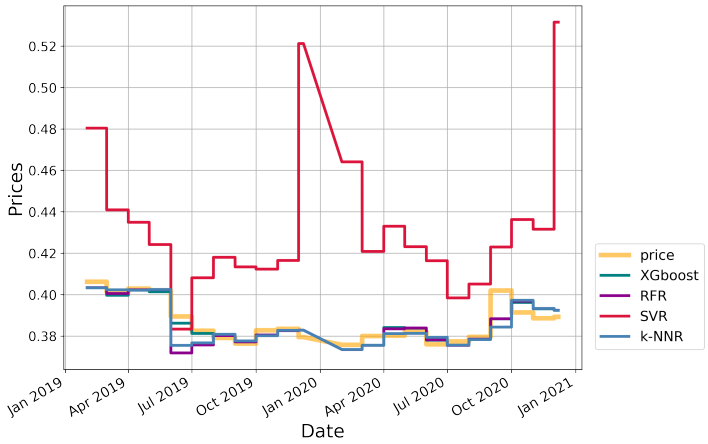
Figure 4.24: The average window predictions vs. actual prices in R1 region (2019-2020)



(a) 1-day window predictions vs. actual prices



(b) 1-week window predictions vs. actual prices



(c) 1-month window predictions vs. actual prices

Figure 4.25: The average window predictions vs. actual prices in R3 region (2019-2020)

4.6 Hyper tuning Effect

One of the research aims is to study the effect of the hyper-tuning of hyperparameters for the four ML models. The investigation was done using the randomized search cross-validation with k= 5. We found that the XGBoost, RFR, and k-NNR do not require tuning to achieve high performance, whereas the tuning affected the SVR results in the three training windows. In section 3.5.1 (Chapter 3), Table 3.3, we indicated how the hyperparameters were affected, especially the gamma and C. Table 4.2 reflects the improvement of region R1 results by using the tuned models reflected in the five metrics we examined earlier. The table below shows each training window's average error or accuracy values. The readings of AUC-REC show the highest enhancement in the results of 1-week, with approximately a 0.072 increase. At the same time, the 1-month has a slight improvement of 0.027 as the lowest enhancement. The same applies to the rest of the error metrics, considering the lower errors are better. For example, the MAE lowered by 0.091 for 1-week in the tuned model. In contrast, the 1-day ahead shows nearly no improvement. At the same time, the 1-month has slight improvements, around 0.059 error reduction.

Table 4.2: SVR overall metrics results in default and hyper tuning of parameters in R1

Metric	**Default**			**Tuned**		
	1-day	**1-week**	**1-month**	**1-day**	**1-week**	**1-month**
MAE	0.216	0.185	0.171	0.215	0.094	0.112
MSE	0.159	0.099	0.086	0.141	0.011	0.020
RMSE	0.282	0.186	0.173	0.216	0.097	0.128
MAPE	1.924	0.889	0.836	0.871	0.674	0.827
AUC-REC	0.530	0.610	0.640	0.589	0.682	0.667

The highest values of AUC-REC mean better performance, while the higher values of other metrics (MAE, MSE, RMSE, MAPE) means worst performance.

4.7 Discussion and Findings

In this section, we discuss and explore our findings in this research. Firstly, 4 machine learning models predict the SI prices over 2 years (2019 and 2020) for five regions. We found interesting points that need to be considered when machine learning is used in this area, listed in the following:

(1) In this research, we found models such as XGBoost, RFR, and k-NNR are more stable and resistant to data changes over time. The k-NNR uses the lazy process to predict the price by comparing the new observation with stored data from the training set. In contrast, both XGBoost and RFR built the decision trees during training, which takes less time to provide the prediction. Thus, these models are better to be used in the future as live prediction models.

(2) Also, we found that tuning model **hyperparameters** are required for models such as SVR. The SVR is affected directly by data size and distribution. Thus, we must investigate and update some new data's SVR parameters. While XGBoost, RFR, and k-NNR do not need to hyper-tune their parameters because the tuned and default models have the same performance. As a result, using XGBoost, RFR, and k-NNR reduce the time and complexity of investigation for experts.

(3) The **time** of the data measured in years shows that the spot data predictions exhibited different trends and changes after the smooth change of the spot data in 2017. The study of the changes is out of the scoop of our study. Studies [42,67] analyzed the effect of the new change in the distribution and the data price over time and locations. We found the models had better results in the year (2020) when the prices were more stable.

(4) The **training window size** affects the predictions; 1-day is the best of the top three models XGBoost, RFR, and k-NNR. The SVR has the 1-week as the best training window. The 1-day is the best means that the prices have more variance when it exceeds the 24 hours of predictions.

(5) The effect of the **location** was obvious since the distribution of the series differs based on the regions. We limit the study of the location effect to its effect on the behavior of the model's predictions. Although the data distribution differs between regions, the regions have the same behavior when treated as an ML problem. This means the 1-day training window size is always the best regardless of our study region. Additionally, the time effect, whether the predictions have more accuracy and fewer errors generalized for all locations we studied.

(6) We utilized the **REC Curve** and the **AUC-REC** in the area of the spot price predictions for the first time to measure the accuracy of the models against their error tolerance. To compare the significance of this RECC, we used the most utilized error metrics in the related work to measure the forecast errors of the four ML models. We found that no metric reflects the accuracy precisely like AUC-REC with a value between [0,1]. Instead, all metrics were used before to measure the errors with an error range of $[0, \infty)$. The AUC-REC estimates the model's accuracy by finding the area under the REC Curve. While the RECC provides the model's accuracy based on each error threshold. We ranked the evaluation metrics from the lowest to highest insight in the investigation. The ranking is as follows: MSE, MAE, RMSE, MAPE, AUC-REC, and RECC.

To this end, we find the metrics used in the related work did not reflect the difference in the model's performance using different window training sizes, except for

a few outliers in some models. The SVR evaluation using previous metrics indicates how the performance changes through window size due to large prediction errors. This results from the evaluation of the small errors using non-insightful metrics.

First, The MSE errors are determined based on the squared feature. When the point y_i has a loss of less than 1, the $loss_i$ value exponentially decays until it reaches zero at the end. Conversely, large errors receive a high weight to penalize models with high error rates. Although MSE is a scale-dependent metric, the squared unit directs the intuition given using the MSE. The forecast error conveys the MSE values interpreted in our study as squared US dollars which is not comparable to the real price values in US dollars. In the insight, we saw how the

While the MAE estimates the absolute errors, then finds and averages them, maintaining the unit of the predicted. It treats all errors equally and assigns them the same magnitude. However, when averaging small errors, their values may be rounded to zero due to insufficient precision or a limited number of decimal places used to calculate the average.

The RMSE has almost the same insight as MAE, except when there is a big error, it gives more weight before finding its root. The problem of finishing values when squaring small errors arises again. Using the root of MSE is the same as using the MSE when we apply it in the context of small errors, but the root value reflects an estimation of the real unit of prediction (US Dollars). To this end, all the metrics used are scale-dependent, which reflects the change in errors based on the unit of the predicted value.

The last and most utilized metric in related work is the MAPE, a free-dependent scale that gave an overall insight into the models' percentage errors. It has the problem of converging to zero when the errors are small and might have infinity and negative values. In our work, the percentage errors did not reveal how the error

changes between the training window size or the regions.

The RECC and AUC-REC indicate how the model's performance changed when studying all the effects. It uses the absolute deviation as an error threshold.

Chapter 5

CONCLUSION AND FUTURE DIRECTIONS

5.1 Conclusion

Predicting spot prices is challenging due to their dynamic and unpredictable nature. The inherent volatility and variability of spot prices emphasize the importance of employing effective metrics to evaluate the accuracy and efficacy of spot price prediction models. This research evaluated the performance of four supervised and non-parametric machine learning models using six distinct loss measurements. Our primary focus was on the Regression Error Characteristic (REC) curve and the Area Under the Curve (AUC-REC) as novel metrics for evaluating predicted prices.

We investigated the impact of various factors, such as data time, location, and training window size. We found that XGBoost, RFR, and k-NNR are more stable and resistant to data changes over time, making them suitable as live prediction models. Additionally, we used hyperparameter tuning during training. We found that the tuned models did not show improvement for the top three models, as mentioned earlier, unlike SVR, which improved in the three training windows.

Furthermore, the results of our study indicate that the REC curves and AUC-REC are the most reliable metrics for measuring the output quality from the machine learning models used for EC2 spot price prediction. Specifically, the AUC-REC metric estimates the model's accuracy across all error tolerance thresholds. In addition, the REC curve clearly explains the trade-off between error tolerance and expected accuracy at any prediction point. These metrics allow customers in the spot market to make informed decisions based on their specific needs and tolerance for prediction errors.

We also found that other metrics, such as Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Squared Error (MSE), can also provide helpful insights into the performance of the models. However, these metrics provide a different level of precision than the REC and AUC-REC curves.

Our study highlights the importance of choosing the right evaluation metric to minimize work interruptions and reduce costs during the operating time of spot instances. Using the right evaluation metric enables customers to make informed decisions about the maximum price that ensure the spot prices are successfully predicted. The precision of predicting prices is important, especially in the spot market, where incorrect predictions can lead to significant financial losses.

To this end, our research provides new insights into evaluating spot price predictions and lays the foundation for future studies that further improve the accuracy of spot price predictions. The REC curves and AUC-REC can be valuable tools for evaluating the accuracy and efficacy of machine learning models used for EC2 spot price prediction. The other metrics we studied can provide additional insights into the performance of the models.

The outcome of our research is a paper titled: *"Evaluating Amazon EC2 Spot Price*

Prediction Models Using Regression Error Characteristic Curve", published in a peer-reviewed conference: *"The Seventh International Conference on Fog and Mobile Edge Computing (FMEC2022), Dec 2022"*[1].

5.2 Future Work

This section discusses the limitations of our study and proposes suggestions for future research.

We employed four supervised regression machine learning algorithms that have been extensively researched in the literature and followed the standard training, validation, and testing procedure. To evaluate the performance of the regression algorithms, we used the RECC and AUC-REC as the evaluation metric, which we argued is more representative than other metrics like MAPE, RMSE, MAE, and MSE. However, it is worth noting that many other error metrics are proposed as an alternative to the one we used. In future studies, we recommend investigating the relative errors and scaled error metrics [23, 24].

Also, the RECC could use different error functions like squared errors instead of absolute ones. We encourage the usage of the RECC along different functions and apply it in various forecasting fields.

One potential limitation of our study is that we used only supervised regression machine learning models. Although we employed efficient algorithms known for their performance in various regression tasks and research domains, the choice of algorithms may still be a limitation. However, we believe that selecting well-known algorithms partially alleviates this limitation.

It is important to note that we experimented with EC2 AWS spot instances prices for two years. Therefore, we cannot generalize our results to all other cloud providers,

[1] https://ieeexplore.ieee.org/document/10062720

such as Azure, industrial, and other cloud vendors to which we could not access their preemptible history.

For future work, we intend to evaluate how other forecasting methods, such as deep learning and statistical methods, could affect the prediction based on the REC Curve and AUC-REC compared with supervised machine learning methods. Additionally, we only studied the results of five regions and three training time windows in this research. Therefore, the results of other regions and training windows require further investigation, analysis, and discussion on a yearly basis.

Appendix A

Table A.1: KPSS Test in region R1

Time Series	**KPSS Statistic**	**p-value**	**Is Stationary (KPSS)**
c3.large-Red Hat Enterprise Linux	49.33856807	0.01	FALSE
c3.xlarge-Red Hat Enterprise Linux	10.12768256	0.01	FALSE
c3.2xlarge-Red Hat Enterprise Linux	8.126634919	0.01	FALSE
c3.4xlarge-Red Hat Enterprise Linux	3.825770838	0.01	FALSE
c3.8xlarge-Red Hat Enterprise Linux	3.209581884	0.01	FALSE
c4.large-Red Hat Enterprise Linux	48.85372244	0.01	FALSE
c4.xlarge-Red Hat Enterprise Linux	6.078708957	0.01	FALSE
c4.2xlarge-Red Hat Enterprise Linux	9.615312506	0.01	FALSE
c4.4xlarge-Red Hat Enterprise Linux	8.955150278	0.01	FALSE
c4.8xlarge-Red Hat Enterprise Linux	1.921527817	0.01	FALSE
c3.xlarge-SUSE Linux	9.891446213	0.01	FALSE
c3.2xlarge-SUSE Linux	10.54303433	0.01	FALSE
c3.4xlarge-SUSE Linux	6.595452651	0.01	FALSE
c3.8xlarge-SUSE Linux	4.690580699	0.01	FALSE
c4.xlarge-SUSE Linux	9.354523074	0.01	FALSE
c4.2xlarge-SUSE Linux	13.74967376	0.01	FALSE
c4.4xlarge-SUSE Linux	4.534041711	0.01	FALSE
c4.8xlarge-SUSE Linux	4.735641376	0.01	FALSE
c3.large-Linux	76.96647754	0.01	FALSE
c3.xlarge-Linux	9.951165103	0.01	FALSE
c3.2xlarge-Linux	10.55803218	0.01	FALSE
c3.4xlarge-Linux	6.48136545	0.01	FALSE
c3.8xlarge-Linux	4.5060962	0.01	FALSE
c4.large-Linux	76.96647754	0.01	FALSE
c4.xlarge-Linux	8.436511817	0.01	FALSE
c4.2xlarge-Linux	13.69085179	0.01	FALSE
c4.4xlarge-Linux	4.592851492	0.01	FALSE
c4.8xlarge-Linux	4.608498934	0.01	FALSE

Table A.2: ADF Test in region R1

Time Series	**ADF Statistic**	**p-value**	**Is Stationary (ADF)**
c3.xlarge-Red Hat Enterprise Linux	-0.913549063	0.783477777	FALSE
c3.2xlarge-Red Hat Enterprise Linux	-0.62914955	0.864258464	FALSE
c3.4xlarge-Red Hat Enterprise Linux	-1.100695951	0.714794288	FALSE
c3.8xlarge-Red Hat Enterprise Linux	-1.817071118	0.372019897	FALSE
c4.xlarge-Red Hat Enterprise Linux	-1.796865614	0.382018936	FALSE
c4.2xlarge-Red Hat Enterprise Linux	-0.677402149	0.852516387	FALSE
c4.4xlarge-Red Hat Enterprise Linux	-0.799991617	0.81912453	FALSE
c4.8xlarge-Red Hat Enterprise Linux	-0.940770436	0.774249902	FALSE
c3.xlarge-SUSE Linux	-1.492464217	0.537235834	FALSE
c3.2xlarge-SUSE Linux	-0.818693845	0.813569127	FALSE
c3.4xlarge-SUSE Linux	-1.372995577	0.595151744	FALSE
c3.8xlarge-SUSE Linux	-1.984363	0.293471185	FALSE
c4.xlarge-SUSE Linux	-2.090901938	0.24818868	FALSE
c4.2xlarge-SUSE Linux	-1.091959319	0.718267761	FALSE
c4.4xlarge-SUSE Linux	-1.239885291	0.656207161	FALSE
c4.8xlarge-SUSE Linux	-1.464796122	0.550848801	FALSE
c3.xlarge-Linux	-1.343820332	0.608893803	FALSE
c3.2xlarge-Linux	-0.944142164	0.773088564	FALSE
c3.4xlarge-Linux	-2.067219375	0.257898444	FALSE
c3.8xlarge-Linux	-2.616270596	0.089680877	FALSE
c4.xlarge-Linux	-2.176698468	0.550848801	FALSE
c4.2xlarge-Linux	-1.240886731	0.655764747	FALSE
c4.4xlarge-Linux	-1.716416799	0.422674514	FALSE
c4.8xlarge-Linux	-1.942069819	0.530848801	FALSE

Bibliography

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] "Amazon EC2 Spot Instances Pricing." https://aws.amazon.com/ec2/spot/pricing/. Accessed:2022-10-04.

[3] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon ec2 spot instance pricing," *ACM Trans. Econ. Comput.*, vol. 1, 2013.

[4] M. Baughman, S. Caton, C. Haas, R. Chard, R. Wolski, I. Foster, and K. Chard, "Deconstructing the 2017 changes to aws spot market pricing," in *Proceedings of the 10th Workshop on Scientific Cloud Computing*, pp. 19–26, 2019.

[5] S. Shastri and D. Irwin, "Cloud index tracking: Enabling predictable costs in cloud spot markets," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 451–463, 2018.

[6] C. M. Bishop, *Pattern recognition and machine learning*. Information science and statistics, New York, NY: Springer, 2006.

[7] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, vol. SMC-15, no. 4, pp. 580–585, 1985.

[9] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1996.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 ed., 2009.

[12] M. B. Chhetri, M. Lumpe, Q. B. Vo, and R. Kowalczyk, "On estimating bids for amazon ec2 spot instances using time series forecasting," in *2017 IEEE International Conference on Services Computing (SCC)*, pp. 44–51, IEEE, 2017.

[13] M. Baughman, C. Haas, R. Wolski, I. Foster, and K. Chard, "Predicting Amazon Spot Prices with LSTM Networks," in *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ScienceCloud'18, (New York, NY, USA), Association for Computing Machinery, 2018.

[14] S. Agarwal, A. K. Mishra, and D. K. Yadav, "Forecasting price of amazon spot instances using neural networks," *Int. J. Appl. Eng. Res*, vol. 12, no. 20, pp. 10276–10283, 2017.

[15] H. Al-Theiabat, M. Al-Ayyoub, M. Alsmirat, and M. Aldwair, "A Deep Learning Approach for Amazon EC2 Spot Price Prediction," in *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–5, 2018.

[16] A. Sarah, K. Lee, and H. Kim, "LSTM Model to Forecast Time Series for EC2 Cloud Price," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing (DASC)*, pp. 1085–1088, 2018.

[17] V. Khandelwal, A. K. Chaturvedi, and C. P. Gupta, "Amazon EC2 spot price prediction using regression random forests," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 59–72, 2017.

[18] c. Liu, P. Wang, Y. Meng, C. Zhao, and Z. Zhang, "Cloud spot instance price prediction using kNN regression," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1–14, 2020.

[19] D. Kong, S. Liu, and L. Pan, "Amazon Spot Instance Price Prediction with GRU Network," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 31–36, 5 2021.

[20] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[21] J. Bi and K. P. Bennett, "Regression error characteristic curves," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 43–50, 2003.

[22] T. Fawcett, "An introduction to roc analysis," *Pattern Recogn. Lett.*, vol. 27, p. 861–874, Jun 2006.

[23] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[24] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," *arXiv preprint arXiv:1809.03006*, 2018.

[25] A. Botchkarev, "A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 14, pp. 045–076, 2019.

[26] S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International Journal of Forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

[27] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[28] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning cloud dynamics to optimize spot instance bidding strategies," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 2762–2770, IEEE, 2018.

[29] B. G. Raj Bala, "Magic quadrant for cloud infrastructure and platform services," tech. rep., Gartner Inc., 2021.

[30] V. Abhishek, I. A. Kash, and P. Key, "Fixed and market pricing for cloud services," in *2012 Proceedings IEEE INFOCOM Workshops*, pp. 157–162, IEEE, 2012.

[31] E. K. Siham, C. Schlereth, B. Skiera, *et al.*, "Price comparison for infrastructure-as-a-service," 2012.

[32] S. Chen, H. Lee, and K. Moinzadeh, "Pricing schemes in cloud computing: Utilization-based vs. reservation-based," *Production and Operations Management*, vol. 28, no. 1, pp. 82–102, 2019.

[33] C. Wu, R. Buyya, and K. Ramamohanarao, "Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[34] J. Barr, "Streamlined access to spot capacity smooth price changes instance hibernation." https://aws.amazon.com/blogs/aws/amazon-ec2-update-streamlined-access-to-spot-capacity-smooth-price-changes-instance-hibernation/. Accessed:2022-06-27.

[35] D. Liu, Z. Cai, and X. Li, "Hidden markov model based spot price prediction for cloud computing," in *2017 IEEE International symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC)*, pp. 996–1003, IEEE, 2017.

[36] Z. Cai, X. Li, R. Ruiz, and Q. Li, "Price forecasting for spot instances in cloud computing," *Future Generation Computer Systems*, vol. 79, pp. 38–53, 2018.

[37] M. Khan, A. I. Jehangiri, Z. Ahmad, M. A. Ala'anzy, and A. Umer, "An exploration to graphics processing unit spot price prediction," *Cluster Computing*, vol. 25, no. 5, pp. 3499–3515, 2022.

[38] S. Caton, M. Baughman, C. Haas, R. Chard, I. Foster, and K. Chard, "Assessing the current state of aws spot market forecastability," in *2022 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)*, pp. 8–15, IEEE, 2022.

[39] X. Song, R. Lin, and H. Zou, "Amazon ec2 spot price prediction using temporal convolution network," in *ICETIS 2022; 7th International Conference on Electronic Technology and Information Science*, pp. 1–6, VDE, 2022.

[40] S. S. Nezamdoust, M. A. Pourmina, and F. Razzazi, "Optimal prediction of cloud spot instance price utilizing deep learning," *The Journal of Supercomputing*, pp. 1–22, 2022.

[41] J. S. Armstrong and F. Collopy, "Error measures for generalizing about forecasting methods: Empirical comparisons," *International journal of forecasting*, vol. 8, no. 1, pp. 69–80, 1992.

[42] G. George, R. Wolski, C. Krintz, and J. Brevik, "Analyzing aws spot instance pricing," in *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 222–228, 2019.

[43] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American statistical association*, vol. 74, no. 366a, pp. 427–431, 1979.

[44] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?," *Journal of econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[45] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, p. 281–305, feb 2012.

[46] J. S. Armstrong, *Principles of forecasting: a handbook for researchers and practitioners*, vol. 30. Springer, 2001.

[47] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994.

[48] M. Kuhn, K. Johnson, *et al.*, *Applied predictive modeling*, vol. 26. Springer, 2013.

[49] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. Australia: OTexts, 2nd ed., 2018.

[50] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[51] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, may 2011.

[52] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[53] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of Improving K-Nearest-Neighbor for Classification," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, vol. 1, pp. 679–683, 2007.

[54] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[55] C. Ardi, "Amazon EC2 Spot Price History: 2014–2015, 2017–2021," 2022.

[56] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[57] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[58] A. Tahmassebi and T. Smith, "Slickml: Slick machine learning in python," *URL available at: https://github. com/slickml/slick-ml*, 2021.

[59] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[60] S. Makridakis and M. Hibon, "The m3-competition: results, conclusions and implications," *International journal of forecasting*, vol. 16, no. 4, pp. 451–476, 2000.

[61] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[62] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[63] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016.

[64] A. Tahmassebi, "ideeple: Deep learning in a flash," in *Disruptive Technologies in Information Sciences*, vol. 10652, p. 106520S, International Society for Optics and Photonics, 2018.

[65] J. Hernández-Orallo, "Roc curves for regression," *Pattern Recognition*, vol. 46, no. 12, pp. 3395–3411, 2013.

[66] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669–679, 2016.

[67] N. Ekwe-Ekwe and A. Barker, "Location, location, location: exploring amazon ec2 spot instance pricing across geographical regions," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 370–373, IEEE, 2018.