UDC: 004.93 https://doi.org/10.15407/jai2024.04.050

O. Artamonov[1], P. Balych[2]
[1,2]Lviv Polytechnic National University, Ukraine
12, Stepan Bandera str., Lviv, 79013
[1]oleksandr.o.artamonov@lpnu.ua
[2]pavlo.y.balych@lpnu.ua
[1]https://orcid.org/0009-0000-8295-733X
[2]https://orcid.org/0009-0002-0250-9786

ARTIFICIAL INTELLIGENCE IN OPTIMIZATION OF CLOUD RESOURCES

Abstract. Artificial intelligence (AI) is emerging as a transformative force in cloud product optimization, enabling organizations to achieve efficiency, scalability, and cost-effectiveness that were previously unattainable. The complexity of managing cloud resources, including cost, performance, and reliability, has increased dramatically with the widespread adoption of cloud computing.

AI techniques, such as machine learning, deep learning, and reinforcement learning, can be leveraged to address these complexities by predicting workload patterns, automating resource allocation, and ensuring optimal performance through proactive monitoring and adjustments.

This article provides an in-depth exploration of AI-based methods used for optimizing cloud infrastructure, focusing on real-world scenarios like dynamic resource allocation, pricing prediction, and service reliability.

Additionally, we present the challenges of AI adoption in cloud optimization and outline potential directions for future research.

Keywords: artificial intelligence, cloud optimization, machine learning, resource allocation, predictive analysis, cost efficiency.

Intoduction

The rapid advancement of cloud computing has significantly transformed the way organizations operate and manage their IT infrastructure. Cloud computing provides scalable, flexible, and cost-efficient solutions, allowing businesses to focus more on innovation rather than on managing physical hardware. [1] However, with the increasing adoption of cloud services, the complexity of managing cloud resources, including computing power, storage, and network bandwidth, has grown. Ensuring cost-effectiveness, performance, and resource optimization has become a challenge that many organizations face. Artificial Intelligence (AI), with its predictive and automation capabilities, is increasingly seen as a solution to address these challenges. By leveraging AI, cloud environments can achieve enhanced efficiency, scalability, and reliability. [2] This paper explores the role of AI in optimizing cloud product performance, analyzing the benefits, challenges, and implementation strategies for AI-driven cloud optimization.

Cloud systems have become an integral part of modern IT infrastructure, hosting numerous products and services. Managing and optimizing these systems, however, has become increasingly complex due to the growing number of services, varying user demands, and the dynamic nature of workload requirements. AI techniques such as machine learning, deep learning, and reinforcement learning offer solutions to these challenges by enabling real-time decision-making, automation, and predictive analysis. [3]

For instance, cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud offer several services to help organizations manage cloud resources more effectively. However, the selection of appropriate resources, effective scaling, and cost management remain difficult tasks. AI has the capability to significantly enhance these aspects through intelligent prediction and optimization models. Many companies still rely on traditional approaches to handle resource allocation, which often lead to overprovisioning or underutilization of cloud resources. This mismanagement results in increased costs and potential system performance issues. [4]

The relevance of optimizing cloud systems cannot be overstated. As more

businesses migrate to cloud environments, maintaining cost efficiency while ensuring optimal performance becomes a critical task. According to a report by Gartner, cloud spending is predicted to reach $679 billion by 2025, highlighting the growing reliance on cloud services and the necessity for effective optimization strategies. [5] Leveraging AI-based approaches to enhance resource allocation, load balancing, and overall cost management can lead to significant savings and improved system reliability. Moreover, the adoption of predictive and proactive methods allows organizations to forecast future requirements, adjust capacity accordingly, and prevent unexpected issues.

A particularly challenging aspect of cloud optimization is resource selection-whether to use On-Demand or Spot instances, which vary greatly in cost and availability. While tools like AWS's own EC2 Instance Selector provide recommendations based on current prices and instance categories, a more sophisticated approach that incorporates historical data and AI for predictive analysis can yield far better results. This paper aims to delve into AI's role in tackling these complexities, providing a comprehensive understanding of the approaches that can be used to optimize cloud products. [6]

Analysis of recent research and publications

Recent studies have highlighted the use of reinforcement learning in cloud resource management, allowing systems to dynamically adjust computing power based on usage patterns. For example, research by Amazon's AWS Labs has shown that reinforcement learning models can predict incoming traffic and adjust the required server capacity to handle the anticipated load, minimizing the cost of overprovisioning.

Additionally, machine learning models are commonly used for predictive load balancing, which helps reduce service latency and optimize costs. A notable example is Google’s use of AI to manage the cooling of their data centers, which has helped reduce energy consumption by up to 40%. Many researchers also emphasize the importance of proactive failure detection using deep learning to ensure uninterrupted services. AI-driven tools like Datadog, integrated with cloud systems, provide monitoring and alerts to identify potential points of failure, preventing unexpected downtimes. [7]

On the other hand, there remain significant challenges and research gaps, particularly concerning multi-cloud environments where multiple providers are used to achieve redundancy or specialized services. The integration of heterogeneous cloud services requires AI models capable of understanding the different SLAs, pricing, and performance metrics offered by each provider. Multi-cloud resource optimization is still underexplored, presenting an opportunity for future AI solutions to make significant impacts.

Advantages and disadvantages of using predictive AI models for cloud optimization of EC2 resources against regular methods:

– **Cost efficiency**: Predictive AI models can analyze historical data to forecast future demand, allowing better decisions between using On-Demand or Spot instances. This approach can lead to significant cost savings by taking advantage of lower-cost instances when availability is high.

– **Scalability**: AI-driven systems can automatically adjust resource allocation based on predicted workloads, reducing the need for manual intervention and allowing systems to scale efficiently in response to changes in demand.

– **Reduced Human Error**: Automation through predictive AI minimizes human error in resource planning and allocation, leading to more reliable cloud operations.

– **Proactive Resource Management**: Predictive AI allows for proactive adjustments to cloud resources, preventing performance bottlenecks and reducing downtime.

From the business point of view there a few disadvantages we can admit:

– **Complexity:** Implementing predictive AI models requires expertise in both cloud computing and machine learning, making it more complex than traditional resource allocation methods.

– **Data Dependency:** The accuracy of predictions heavily depends on the quality and quantity of historical data available.

– **Initial Costs:** Developing and deploying AI models involves higher upfront costs compared to regular manual methods.

Fig. 1. Benefits of AI Cloud Computing

Objectives of the study. The primary objective of this article is to demonstrate how AI can be used to predict AWS EC2 pricing using historical dataset:

– **Leverage AI techniques:** to analyze past pricing data for various AWS EC2 instance types and categories (spot and on-demand).

– **Cost optimization:** by predicting future prices, the goal is to provide user with better insights for optimizing their cloud infrastructure.

– **Showcase solution:** the article aims to get you through data collection integration, model training, prediction and visualization of a complete cohesive solution for cloud optimization.

Methods and dependencies

To achieve the expected result the following technologies and methods where used:

– **AWS Account**: For accessing cloud resources like EC2, S3, and CloudWatch.

– **Python Libraries**: NumPy, pandas, and scikit-learn, matplotlib.pyplot for data processing, building predictive models and drawing MATLAB-like graphics.

– **Machine Learning Model**: Random Forest Regressor used to create predictive models for resource optimization.

– **Historical Data**: Collected from AWS EC2 instances, including instance type, region, and pricing data.

Input dataset. Below on (Fig. 2), will be shown an input csv formatted dataset for testing of the model. Here are provided screenshots from the application, along with a detailed explanation and visualizations. The historical pricing data is fetched from an S3 bucket, ensuring the most recent dataset is used for training and prediction. Let’s start with providing historical input dataset about AWS EC2 pricing. Not a complete file will be provided, just a small part of it.

```
date,instance_type,region,category,instance_family,physical_processor,clock_speed,price,year,month,day,weekday
2024-07-01,t2.micro,ap-southeast-1,on-demand,General,Intel Xeon,2.5GHz,0.08608,2024,7,1,0
2024-07-01,t2.micro,ap-southeast-1,spot,General,Intel Xeon,2.5GHz,0.26536,2024,7,1,0
2024-07-01,m5.large,ap-southeast-1,on-demand,Compute,Intel Xeon,3.0GHz,0.4642,2024,7,1,0
2024-07-01,m5.large,ap-southeast-1,spot,Compute,Intel Xeon,3.0GHz,0.10129,2024,7,1,0
2024-07-01,c5.xlarge,ap-southeast-1,on-demand,Compute,Intel Xeon,3.5GHz,0.47674,2024,7,1,0
2024-07-01,c5.xlarge,ap-southeast-1,spot,Compute,Intel Xeon,3.5GHz,0.31982,2024,7,1,0
2024-07-01,t3.medium,ap-southeast-1,on-demand,Burstable,Intel Xeon,2.3GHz,0.16314,2024,7,1,0
2024-07-01,t3.medium,ap-southeast-1,spot,Burstable,Intel Xeon,2.3GHz,0.03162,2024,7,1,0
2024-07-01,r5.large,ap-southeast-1,on-demand,Memory,Intel Xeon,3.1GHz,0.22213,2024,7,1,0
2024-07-01,r5.large,ap-southeast-1,spot,Memory,Intel Xeon,3.1GHz,0.11571,2024,7,1,0
2024-07-02,t2.micro,ap-southeast-1,on-demand,General,Intel Xeon,2.5GHz,0.15593,2024,7,2,1
2024-07-02,t2.micro,ap-southeast-1,spot,General,Intel Xeon,2.5GHz,0.22525,2024,7,2,1
2024-07-02,m5.large,ap-southeast-1,on-demand,Compute,Intel Xeon,3.0GHz,0.14478,2024,7,2,1
2024-07-02,m5.large,ap-southeast-1,spot,Compute,Intel Xeon,3.0GHz,0.20682,2024,7,2,1
2024-07-02,c5.xlarge,ap-southeast-1,on-demand,Compute,Intel Xeon,3.5GHz,0.16157,2024,7,2,1
2024-07-02,c5.xlarge,ap-southeast-1,spot,Compute,Intel Xeon,3.5GHz,0.15744,2024,7,2,1
2024-07-02,t3.medium,ap-southeast-1,on-demand,Burstable,Intel Xeon,2.3GHz,0.01443,2024,7,2,1
2024-07-02,t3.medium,ap-southeast-1,spot,Burstable,Intel Xeon,2.3GHz,0.3305,2024,7,2,1
2024-07-02,r5.large,ap-southeast-1,on-demand,Memory,Intel Xeon,3.1GHz,0.30445,2024,7,2,1
2024-07-02,r5.large,ap-southeast-1,spot,Memory,Intel Xeon,3.1GHz,0.30155,2024,7,2,1
2024-07-03,t2.micro,ap-southeast-1,on-demand,General,Intel Xeon,2.5GHz,0.25057,2024,7,3,2
2024-07-03,t2.micro,ap-southeast-1,spot,General,Intel Xeon,2.5GHz,0.20123,2024,7,3,2
2024-07-03,m5.large,ap-southeast-1,on-demand,Compute,Intel Xeon,3.0GHz,0.21154,2024,7,3,2
2024-07-03,m5.large,ap-southeast-1,spot,Compute,Intel Xeon,3.0GHz,0.32811,2024,7,3,2
2024-07-03,c5.xlarge,ap-southeast-1,on-demand,Compute,Intel Xeon,3.5GHz,0.29168,2024,7,3,2
2024-07-03,c5.xlarge,ap-southeast-1,spot,Compute,Intel Xeon,3.5GHz,0.13062,2024,7,3,2
2024-07-03,t3.medium,ap-southeast-1,on-demand,Burstable,Intel Xeon,2.3GHz,0.23081,2024,7,3,2
2024-07-03,t3.medium,ap-southeast-1,spot,Burstable,Intel Xeon,2.3GHz,0.00786,2024,7,3,2
2024-07-03,r5.large,ap-southeast-1,on-demand,Memory,Intel Xeon,3.1GHz,0.08488,2024,7,3,2
2024-07-03,r5.large,ap-southeast-1,spot,Memory,Intel Xeon,3.1GHz,0.0625,2024,7,3,2
2024-07-04,t2.micro,ap-southeast-1,on-demand,General,Intel Xeon,2.5GHz,0.39113,2024,7,4,3
2024-07-04,t2.micro,ap-southeast-1,spot,General,Intel Xeon,2.5GHz,0.0553,2024,7,4,3
```

Fig. 2. Input CSV dataset

Program overview. Application has a single entrypoint. It coordinates the entire process by calling functions from other modules. It takes user input, including instance type, region, and category, and runs the complete workflow, from data loading and processing to price prediction.

By providing the following cli program call (Fig. 3) we initiate the process of data receive, processing and training the model according to csv file from (Fig. 2) It has to show us price predictions within next month for each day.

```
python main.py --instance_type t3.medium
--region ap-southeast-1 --category spot
```

Fig. 3. Program call

Make another call

As for testing and price comparison purposes, let's make another program call but with different parameters.

Assuming that customer wants to know the on-demand price as well. Usually, on-demand has higher price than spot instances because they are just pure virtual machines that are provided to a requestor, when spot instances are a already allocated cloud resources by hypervisor that are currently not used.

Data processing

The following snippet (Fig. 7) demonstrates how data is loaded and processed to generate useful features like year, month, day and weekday, which are essential for training the prediction model.

The load_data_from_s3 function reads data directly from an AWS S3 bucket, ensuring access to up-to-date historical pricing information. The preprocessing step extracts temporal features to help the model capture pricing trends over time.

Below in (Fig. 4) is a console program response output example.

Model training

Code on (Fig. 8) show how the RandomForest model is trained using the processed features.

The dataset is split into training and testing subsets to validate the model's

performance. The RandomForestRegressor is trained on historical pricing data, learning complex relationships between instance features and their corresponding prices to generate accurate future predictions.

Price prediction

The prediction of the EC2 pricing for a future time period using the trained model. The pd.date_range function is used to generate the dates for the next month and can be seen on (Fig. 9).

The data is then prepared with one-hot encoding (pd.get_dummies) to match the feature format expected by the model. The trained model is then used to predict future prices, providing users with insights into the expected cost trends for AWS EC2 instances.

```
Data processing completed successfully and saved as
'processed_ec2_pricing_data.csv'.

Model training completed and model saved successfully.

Predicted price for 2024-11-01 is $0.21 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-02 is $0.22 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-03 is $0.20 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-04 is $0.21 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-05 is $0.21 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-06 is $0.27 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-07 is $0.26 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-08 is $0.26 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-09 is $0.20 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-10 is $0.21 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-11 is $0.18 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-12 is $0.19 for instance type: t3.medium,
region: ap-southeast-1, category: spot

Predicted price for 2024-11-13 is $0.26 for instance type: t3.medium,
region: ap-southeast-1, category: spot
```

Fig. 4. Program output

```
python main.py --instance_type t3.medium
--region ap-southeast-1 --category on-
demand
```

Fig. 5. Program call with on-demand parameter

```
Data processing completed successfully and saved as 'processed_ec2_pricing_data.csv'.

Model training completed and model saved successfully.

Predicted price for 2024-11-01 is $0.29 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-02 is $0.31 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-03 is $0.28 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-04 is $0.27 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-05 is $0.28 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-06 is $0.33 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-07 is $0.33 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-08 is $0.34 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-09 is $0.34 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-10 is $0.28 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand

Predicted price for 2024-11-11 is $0.26 for instance type: t3.medium, region: ap-
southeast-1, category: on-demand
```

Fig. 6. Output with on-demand category

```
# Load data from S3 and preprocess it

data = load_data_from_s3(bucket_name,
object_key, aws_access_key,
aws_secret_key, region_name)

data['date'] =
pd.to_datetime(data['date'])

data['year'] = data['date'].dt.year

data['month'] = data['date'].dt.month

data['day'] = data['date'].dt.day

data['weekday'] =
data['date'].dt.weekday
```

Fig. 7. Data processing

```
# Train the RandomForest model

X_train, X_test, y_train, y_test =
train_test_split(features, target,
test_size=0.2, random_state=42)

model =
RandomForestRegressor(n_estimators=100,
random_state=42)

model.fit(X_train, y_train)
```

Fig. 8. Model training

```
# Predict future prices for the next
month

future_dates =
pd.date_range(start='2024-11-01',
end='2024-11-30')

future_data =
pd.DataFrame(future_features)

future_data =
pd.get_dummies(future_data)

future_data = future_data[model_columns]

predictions = model.predict(future_data)
```

Fig. 9. Price prediction

Conclusions

In this article we have investigated use of AI within cloud compute resources, machine learning capabilities, analyzed advantages and disadvantages of AI implementation while allocating necessary cloud resources. We also discovered one of AI methods how to provide more scalable, flexible, and cost-efficient solutions within cloud compute architectures.

While this approach implies that team or organization already have historical data, it makes your cloud resources more affordable, delivering best performance avoiding unnecessary overprovisioning, making infrastructure more flexible and predictive, reduces risk of human error including the usage of major Python libraries for data analysis and machine learning. By providing the inputs to Python libraries, we can tailor it to the particular use case.

There is still a room for improvements and future investigations: improving the inputs for the AI, extending the application to support more cloud providers and resources, such as Google Cloud Compute, AWS ECS.

However, even the existing mechanism is powerful enough to help development teams and organizations to allocate and use their resources in significantly more efficient and intelligent way.

```
import pandas as pd

import joblib

import argparse

import matplotlib.pyplot as plt

def predict_prices(instance_type,
region, category):

    model =
joblib.load('ec2_price_model.pkl')

    model_columns =
joblib.load('model_columns.pkl')

    future_dates =
pd.date_range(start='2024-11-01',
end='2024-11-30')

    future_features = []

    for date in future_dates:

        future_features.append({

            'instance_type':
instance_type,

            'region': region,

            'category': category,

            'year': date.year,

            'month': date.month,

            'day': date.day,

            'weekday': date.weekday()

        })



    future_data =
pd.DataFrame(future_features)

    future_data =
pd.get_dummies(future_data)

    missing_cols = [col for col in
model_columns if col not in
future_data.columns]

    missing_df = pd.DataFrame(0,
index=future_data.index,
columns=missing_cols)

    future_data =
pd.concat([future_data, missing_df],
axis=1)

    future_data =
future_data[model_columns]

    future_prices =
model.predict(future_data)

    dates = []

    prices = []

    for date, price in zip(future_dates,
future_prices):

        print(f'Predicted price for
{date.strftime("%Y-%m-%d")} is
${price:.2f} for instance type:
{instance_type}, region: {region},
category: {category}')

        dates.append(date)

        prices.append(price)

    plt.figure(figsize=(10, 5))

    plt.plot(dates, prices, marker='o',
linestyle='-', color='b')

    plt.xlabel('Date')

    plt.ylabel('Predicted Price (USD)')

    plt.title(f'Predicted EC2 Instance
Prices for {instance_type} in {region}
({category}) for November 2024')

    plt.xticks(rotation=45)

    plt.tight_layout()

    plt.grid(True)

    plt.show()

if __name__ == "__main__":

    parser =
argparse.ArgumentParser(description='Pre
dict EC2 instance prices')

    parser.add_argument('--
instance_type', type=str, required=True,
help='EC2 instance type (e.g.,
t2.micro)')

    parser.add_argument('--region',
type=str, required=True, help='AWS
region (e.g., us-east-1)')

    parser.add_argument('--category',
type=str, choices=['spot', 'on-demand'],
required=True, help='Instance category
(Spot or On-Demand)')

    args = parser.parse_args()

    predict_prices(args.instance_type,
args.region, args.category)
```

Fig. 10. Core functionality

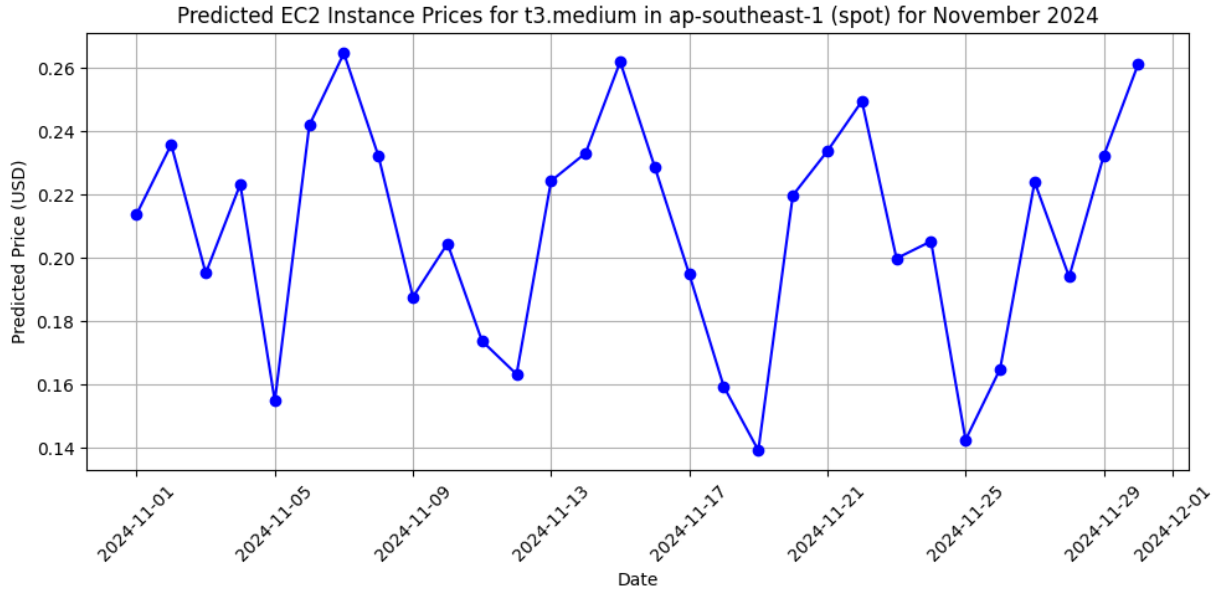

Fig. 11. Finalized results for spot instance using matplotlib

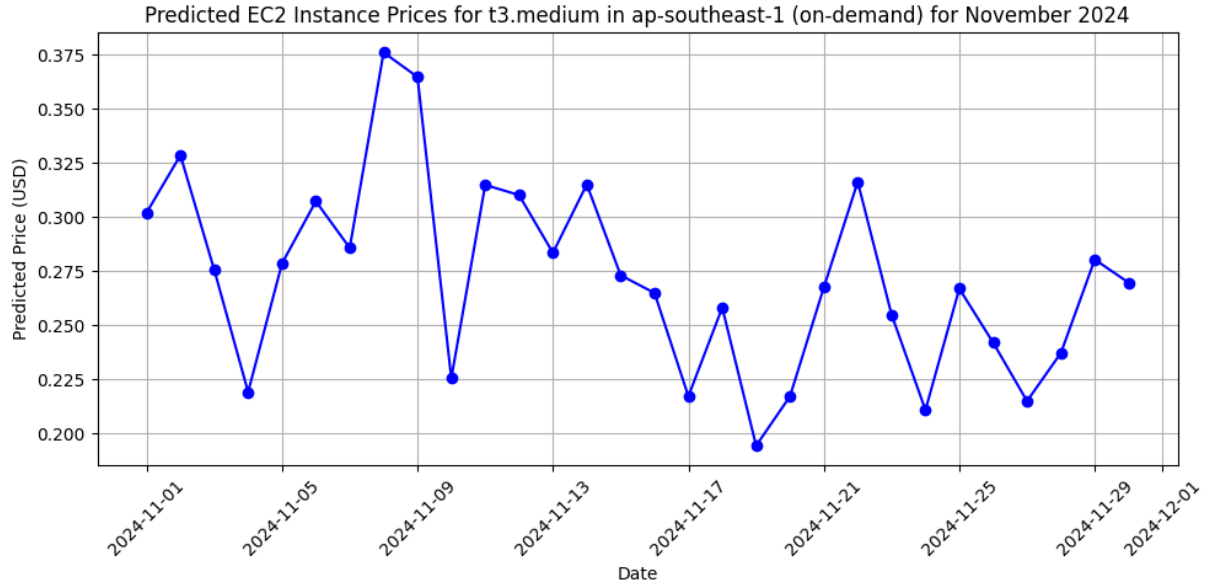

Fig. 12. Finalized results for on-demand instance using matplotlib

References

1. Michael J. Kavis (2019). Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS), 3-11. https://www.everand.com/book/203556393/Architecting-the-Cloud-Design-Decisions-for-Cloud-Computing-Service-Models-SaaS-PaaS-and-IaaS

2. Bernard Marr, Matt Ward (2019). Artificial Intelligence in Practice, 29-37. https://www.perlego.com/book/991892/artificial-intelligence-in-practice-how-50-successful-companies-used-ai-and-machine-learning-to-solve-problems-pdf

3. Jitendra Kumar, Ashutosh Kumar Singh, Anand Mohan, Rajkumar Buyya. (2022) Machine Learning for Cloud Management, 35-47. https://www.routledge.com/Machine-Learning-for-Cloud-Management/Kumar-Singh-Mohan-Buyya/p/book/9780367622565?srsltid=AfmBOopavyj7-gmH2vRVPQIbSyu1KMekwqVxq4riMs7sTb5ftkbul-i8

4. Thomas Erl, Ricardo Puttini, Zaigham Mahmood (2013), Cloud Computing: Concepts, Technology & Architecture, 26-33.

5. Elias Al Helou (2024), EconomyMiddleEast: Gartner predicts $679 bn public cloud end-user spending in 2024. https://economymiddleeast.com/news/gartner-public-cloud-end-user-spending/

6. Dan C. Marinescu (2013). Cloud Computing Theory and Practice, 67-77. https://eclass.uoa.gr/modules/document/file.php/D416/CloudComputingTheoryAndPractice.pdf

7. Mostapha Zbakh, Mohammed Essaaidi, Pierre Manneback, Chunming Rong (2017). Cloud Computing and Big Data: Technologies, Applications and Security, 73-88. https://link.springer.com/book/10.1007/978-3-319-97719-5.

The article has been sent to the editors 04.11.24.
After processing 20.11.24.
Submitted for printing 30.12.24.