National College of Ireland

Hybrid Machine Learning Model for Auto Scaling using CPU Utilization

MSc Research Project
Cloud Computing

Aaditya Pardeshi
Student ID: x22101781

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aaditya Pardeshi
Student ID:	x22101781
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	14/12/2023
Project Title:	Hybrid Machine Learning Model for Auto Scaling using CPU Utilization
Word Count:	6065
Page Count:	20

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

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

Signature:	Aaditya Pardeshi
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

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

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

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

Hybrid Machine Learning Model for Auto Scaling using CPU Utilization

Aaditya Pardeshi
x22101781

Abstract

This project discusses an essential requirement in today's world, that is by utilising strong machine learning models to increase auto-scaling capabilities, ensuring that cloud architecture readily aligns with changeable workloads. Inspired by the need for adaptable and cost-efficient serverless computing, the project introduces and evaluates four very known and important models such as Linear Regression model, Cat Boost Regressor model, Random Forest Regressor model and also a hybrid model that is a Voting Regressor ensemble. Dataset from Materna that contains CPU utilization metrics from VM's is used. The implementation is conducted on AWS SageMaker. By using feature engineering and data preprocessing, the research investigates the effect of each model on system performance. Evaluation metrics like R-squared score, MSE that is Mean Squared Error, as well as Root Mean Squared Error (RMSE) is used for the evaluation. With an R2 score of 0.664, the proposed hybrid model's performance was determined to be optimal and more efficient as compared with other individual models.

1 Introduction

In today's computing frameworks, cloud computing has grown more and more significant, changing the way resources are allocated and managed. This research project intends to make a contribution to this expanding topic by dealing with the important challenge of accurately determining CPU utilisation in cloud systems. The study was motivated by the critical relevance of effective resource allocation, which has a direct influence on operating expenses and system performance in cloud services. As cloud computing grows at an exponential rate, the requirement for strong and accurate forecasting models becomes critical. How can an advanced hybrid machine learning model be effectively used to optimize auto-scaling in serverless environments by predicting CPU Utilization? To answer this issue, the research objectives include the construction, assessment, and comparison of machine learning models for forecasting CPU utilisation, using Random Forest Regressor, CatBoost Regressor, Linear Regression and Ensemble-based Voting Regression. The arguments underlying this study involve determining how effective these models are at making correct predictions.

The structure of this report unfolds with an detailed exploration of the methodology, detailing the steps for preprocessing of the data and model training processes. Subsequently, the results part of the report provides an evaluation of all the developed

models, including performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared scores. Findings that are taken into consideration in the discussion, focuses on both the positives and possible areas for development. In conclusion, the summary encapsulates the main discoveries and suggests future research directions in this dynamic field.

1.1 Background

Cloud computing's explosive growth in recent years has drastically changed how major companies maintain their computing infrastructure. As explained by Armbrust et al. (2010) that, with the flexible and on-demand availability of computing resources that cloud computing offers, businesses may effectively adjust to their changing computational demands. This transformative shift towards cloud technology has introduced new challenges, particularly in the optimal utilization of computing resources to balance performance and cost-effectiveness Buyya et al. (2009). One of the most crucial performance indicators for assessing a platform's actual time workload efficiency is CPU utilisation Kok and Ong (2020). Workload prediction emerges as a pivotal aspect in addressing these challenges.Its important to note that accurately forecasting workload changes is essential for cloud service providers to allocate resources judiciously and enhance overall system efficiency. The need for robust workload prediction models becomes even more critical in the context of unpredictable and dynamic workloads inherent in cloud environments.

This study focuses on execution of various regression models for workload prediction, emphasizing CPU usage as a key metric. ML models that hearby are disscused include Linear Regression, Random Forest Regressor, CatBoost Regressor, and a collaborative approach using a Voting Regressor. Taking from the authors on machine learning applications in cloud computing and workload forecasting Le et al. (2019), this study aims to give important answers into the present discussion over the best way to use cloud resources.

1.2 Research Question

How can an advanced hybrid machine learning model be effectively used to optimize auto-scaling in serverless environments by predicting CPU Utilization ?

1.3 Aim and Objectives

Aim :
The primary focus of this project is to develop and perform evaluation on machine learning models for accurate prediction of CPU utilization in serverless cloud computing environments for better auto-scaling. By using historical data and advanced regression techniques, the project seeks to enhance the efficiency of resource allocation and management, contributing to the optimization of cloud infrastructure.

Objectives:

- To implement engineering techniques to extract relevant information from the dataset and utilize data visualization tools to explore patterns, trends, and gain insights into CPU utilization variations over time.

- Implement three distinct regression models: Linear Regression, CatBoost Regressor, Random Forest Regressor and apply a collaborative approach by creating an ensemble model using a Voting Regressor.
- To compare the performance of the individual models and the hybrid model, and assess each model's performance using metrics like MSE, RMSE, and R-squared score.

2 Related Work

Serverless computing, in the domain of cloud computing, enables users to run applications without the need to manage or provision the underlying infrastructure. Instead of dealing with servers and their maintenance, developers focus solely on coding, while the cloud provider dynamically handles resource allocation. Auto-scaling is urgent in serverless conditions as it guarantees that assets are scalled effectively because of fluctuating responsibilities. So the below section gives an overview on different articles on Serverless Computing, Auto-Scaling,Machine Learning Algorithms, Timeseries Predicition, Window Rolling and more.

2.1 Serverless Computing and Machine Learning Algorithms

Serverless computing is a cloud computing worldview where clients can run applications without overseeing or provisioning the fundamental foundation. Rather than managing servers and their support, designers centre exclusively on composing code, and the cloud supplier handles the asset portion progressively. This approach offers a few benefits, including decreased functional above, cost proficiency, and programmed scaling in light of real interest. Auto-scaling is urgent in serverless conditions as it guarantees that assets are apportioned effectively because of fluctuating responsibilities. The capacity to naturally change assets up or down in real-time is basic for improving execution and limiting expenses, making auto-scaling a critical component in serverless computing.

When it comes to how this serverless computing is a revolution in the industry Rajan (2018) addresses how serverless computing is developing in the cloud and forecasts easier, less expensive, and more effective resource management. It highlights the possibility of wider applications and the advent of "deviceless edge computing" as a fresh field for study. Technical difficulties including safe resource supply, network tolerance, and smooth scalability are noted by the authors. They also highlight open difficulties and future opportunities for serverless computing research. The relevance of both the serverless computing concept and its testing stage in academia and industry are reaffirmed in the conclusion.

When it comes to what is serverless computing, the paper Li et al. (2023) covers this wide field of serverless computing, offering an in-depth overview of its features, problems, and progress. It highlights the limits of current benchmarks and the significance of taking different application combinations into account, emphasising the necessity for extensive benchmarking to handle the numerous uses of serverless computing. The writers also go into the architectural ramifications of serverless computing, concentrating on particular systems like Azure Functions and Apache Openwhisk. In their conclusion, they call for

additional research on serverless computing, especially when it comes to how it might be integrated with cutting-edge innovations like IoT-based systems and autonomous vehicles. The purpose of this thorough review is to encourage scholars to investigate the exciting prospects in serverless computing.

This article Santos et al. (2023) addresses the difficulties in scaling microservices in contemporary architectures and suggests the gym-hpa framework, an approach that leverages reinforcement learning to overcome these difficulties. The authors stress how crucial it is for auto-scaling systems to take application response time and microservice interdependencies into account. They show that A2C is better suited for extension by comparing several RL algorithms. It has been shown that RL agents may be taught offline using simulations and then verified in real-life situations using the suggested simulation method, which performed marginally better than the cluster approach. The authors come to the conclusion that gym-hpa helps researchers assess their auto-scaling methods in actual cloud settings and fills in the gap among RL and automatic scaling

2.2 Auto-Scaling in Serverless Environments

One of the vital difficulties in cloud computing is tracking down ways of limiting the total handling time and cost of microservices while proficiently auto-scaling computing assets in light of client demands. At the point when assets are over-provisioned, it brings about wastage, while under-provisioning can prompt execution issues in cloud servers. Subsequently, an ideal auto-scaling framework ought to have the option to evaluate the capacities of the dynamic physical or virtual machines in a server farm regarding their capacity to adjust to responsibility changes. Also, holders offer incredible adaptability for simultaneous arrangement on cloud servers because of their lightweight nature. A mindful auto-scaling strategy has to concentrate on two basic viewpoints in the cloud climate: a) limiting expenses by relegating microservices to proper physical or virtual machines in light of their necessities, and b) expanding asset use through compelling auto-scaling systems. A paper intends to propose a clever planning procedure to improve the handling of microservices inside a cloud climate. The procedure is supplemented by a auto-scaling strategy to improve the whole proficiency Srirama et al. (2020)

When discussed about serverless computing, the authors Phung and Kim (2022) mention that there are lot of difficulties in auto-scaling and provide a two-state solution for this. The first optimises performance and latency by determining optimal parameter values for the core service function, whilst the second employs a prediction model (Bi-LSTM) to adaptively determine the number of pods depending on request trends. The authors demonstrate better performance when compared to Knative's default setting. They also emphasise the need of choosing proper concurrency levels to minimise latency difficulties. Based on their experiences, the report finishes by recommending further work to deploy the proposed technique in a production setting and to improve Knative's Autoscaler.

2.3 Timeseries prediction, Existing ML Algorithms for Auto-Scaling and their Approaches

Regarding cloud, fog, and edge computing performance assessment measures, the writers Alipour (2019) present a taxonomy of metrics and discuss their importance in evaluating

the efficiency of various optimization techniques. For example, resource allocation, , load balancing, service composition and task scheduling. The paper mentions time series as a method for analyzing performance metrics in the domain of computing in edge, fog and cloud. It discusses how time series data can be analyzed to extract meaningful statistics and other characteristics.

A complete framework for the Cloud-to-Things (integration of IoT with Cloud Computing (CC)) environment is presented in another review, which also outlines the environment of improvement techniques within this model. Aslanpour et al. (2020) . It further presents an order of quantifiable execution measurements intended for every layer of calculation. In the layer of the cloud, differentiation are given in order to address the assorted necessities of different cloud models. The work of the analyzer includes the assessment of checked boundaries and the age of additional exact qualities obtained from these boundaries, which are in this manner used by the organizer. In this situation, the most relevant measurements are factual in nature. In particular, to survey the exactness of predictions, ordinarily used measurements incorporate "MAPE", "MAE", "MSE", "R2", "RMSE", "Average", "Median", "Tail", and "PRED (Prediction Error)".

A thorough summary of time series forecasting techniques can be found in the publication Liu et al. (2021) where it tackles challenges presented on by time series data's growing size and complexity. The authors categorise and contrast the efficacy of current modelling techniques. Additionally, they point up other lines of inquiry, such parallel computing and data preparation. In order to meet the problems brought on by the exponential expansion in data size and complexity, the paper's conclusion emphasises the necessity of ongoing study and improvement in time series forecasting.

When it comes to the techniques used in forecasting the authors Krishna et al. (2022) discusses the importance of ML techniques for traffic flow forecasting. The proposed model, LSTM RNN, is compared to other models like Random Walk (RW), SVM, one layer FFNN, and Stacking Auto Encoder (SAE). The findings demonstrate that the suggested model outperforms the other models in terms of predictions. The suggested model, according to the authors, may be used to increase the precision of traffic flow forecasting and is effective for immediate traffic prediction.

The concept of window rolling is explained in Gašperov et al. (2020) that explores reinforcement learning to determine the optimal size for rolling window for portfolio optimization, with a focus on the global minimum variance portfolio. The authors build a unique state-space description using Frobenius norms to identify correlation shifts related to financial comes back and they show that the resulting agent performs better when compared to traditional standards. The study's assumptions on trade costs and its inclusion of a single class of state attributes are among its weaknesses. Future research will examine a larger range of neural network topologies and add more elements to the state space. All things considered, the authors propose that employing reinforcement learning-based techniques as a supplementary tool in combination with more traditional ways might be a workable step towards more effective portfolio management strategies.

The authors Dong et al. (2018) explain the creation of a scalable machine learning model that uses distributed computing, distributed databases, and machine learning to

forecast household energy usage. The authors store data in AWS S3 and use MongoDB for distributed data storage and access. They process data using Apache Spark on AWS EMR and employ a Random Forest model for prediction. The report emphasizes on importance of efficient storage of data and retrieval for smart meter systems and demonstrates the advantages of using distributed systems and machine learning for accurate energy usage prediction.

The concept of Random Forest is disscussed in Dutta et al. (2020) where the authors use ML models, for example, decision tree and random forest classifiers, for recognizing handwritten Kannada alphabets. The authors emphasize the importance of transforming processed data into useful information in today's data-driven society. They highlight the advantages of random forest classification, which aims to avoid overfitting by utilizing a random set of features for each tree in the ensemble. The methodology involves preprocessing datasets and creating models for binary and multiclass classification using MATLAB. When comparing the effectiveness of random forest and decision tree classifiers, the authors point out that overfitting in a single decision tree is a drawback that is mitigated by the random forest model.

3 Methodology

3.1 Dataset Selection

In the project's data collection phase, three datasets were acquired, each encompassing the performance metrics of VMs from a Materna-operated distributed data center [1]. With over 35 years of experience, Materna stands as a prominent full-service provider in the IT industry, collaborating with renowned German entities and European public sector organizations. The datasets, denoted as Materna-trace-1, Materna-trace-2, and Materna-trace-3, contain performance metrics from 520, 527, and 547 virtual machines, respectively. These metrics were recorded over a three-month period in the Materna Data Centers in Dortmund, with each trace spanning one month. The VMs within these traces primarily host critical business applications for international companies. Performance

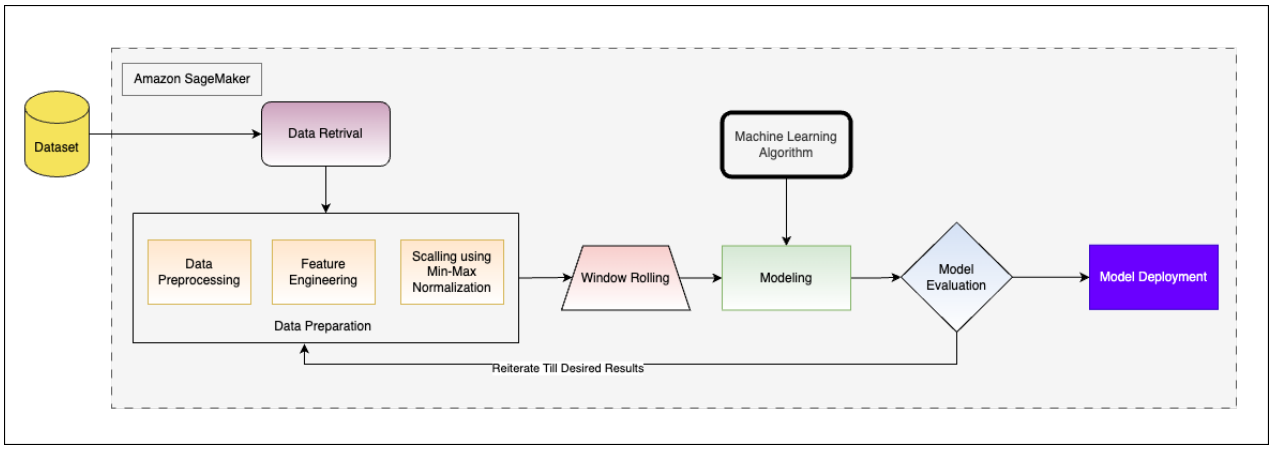


Figure 1: Architecture of the Proposed ML Model

[1] http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna

metrics include timestamp, virtual CPU core count, provisioned CPU capacity, CPU usage, allocated memory, memory usage, disk write throughput, total disk size, network received throughput, and network transmitted throughput.

This rich dataset, drawn from a real-world IT environment, offers invaluable insights into VM performance metrics within a distributed data center, laying the groundwork for subsequent analysis and modeling endeavors in the project.

3.2 Importing Libraries

The code uses various essential Python libraries are imported, including 'pandas,' 'numpy,' 'matplotlib.pyplot,' 'seaborn,' 'plotly'. When combined, these libraries provide a complete environment for activities related to visualisation of the data, it's manipulation, and ML. The 'sklearn' library is extensively used for ML functionalities, featuring modules for model selection ('train_*test*_split'), metrics calculation ('mean_absolute_error,' 'mean_squared_error,' 'r2_score'), and preprocessing with 'MinMaxScaler.' Notably, the code integrates regression models such as 'LinearRegression,' 'RandomForestRegressor,' and 'CatBoostRegressor,' showcasing a diverse set of tools for predictive modeling. Ensemble learning techniques, specifically 'VotingRegressor,' are also included. For applications involving regression-based ML methods and related analysis of data, the code creates a strong basis.

3.3 Loading Dataset

The 'glob' library is used to generate a list of file paths for all CSV files in a directory. It then iterates through the first 150 files, reading each file's content using 'pd.read_csv' from the 'pandas' library. The data from each file is appended to a list called 'datavalues'. The data is then concatenated into a single DataFrame called 'tracedataframe', which serves as a comprehensive representation of the information in the CSV files. The decision to load only 150 files is a deliberate choice, to manage the dataset's size during the initial stages of development or testing.

3.4 Data Cleaning

Missing values must be detected and addressed as part of the data cleaning process to ensure the dataset's completeness and trustworthiness. Hence, after loading the dataset into the DataFrame, a thorough analysis was performed to examine the continuous and categorical columns, as well as the statistical properties of each column. This study aided in gaining a thorough knowledge of the data summary. The data was examined for the existence of null values, and special symbols within the data were discovered and replaced with null values for further analysis. To verify data integrity, duplicate items in the dataset were also reviewed. To discover and measure the values that are not present in each column, the code uses the 'tracedataframe.isna().sum()' function. For data preprocessing, the conversion of 'Timestamp' column into a datetime data type for accurate handling and analysis of temporal data is performed, enabling precise temporal analysis.

3.5 Data Exploration and Visualization:

The code then performs various analysis and visualizations on the time series data. The libraries like Plotly Express and Matplotlib have been used to create graphs and time series plots that reveal patterns and trends in the performance metrics dataset, illustrating CPU utilization year-wise, month-wise, day-wise, hour-wise, and minute-wise mean CPU utilization that are vividly portrayed through engaging funnel plots, bar graphs, pie, line charts, and even scatter plots. These visualizations offer a dynamic representation of how CPU utilization fluctuates over different temporal dimensions The visualizations also reveal intricate relationships between hours, minutes, and days, providing a nuanced understanding of their combined influence on CPU utilization. This comprehensive approach enhances interpretability and provides a solid foundation for future analysis and predictive modeling in this project.

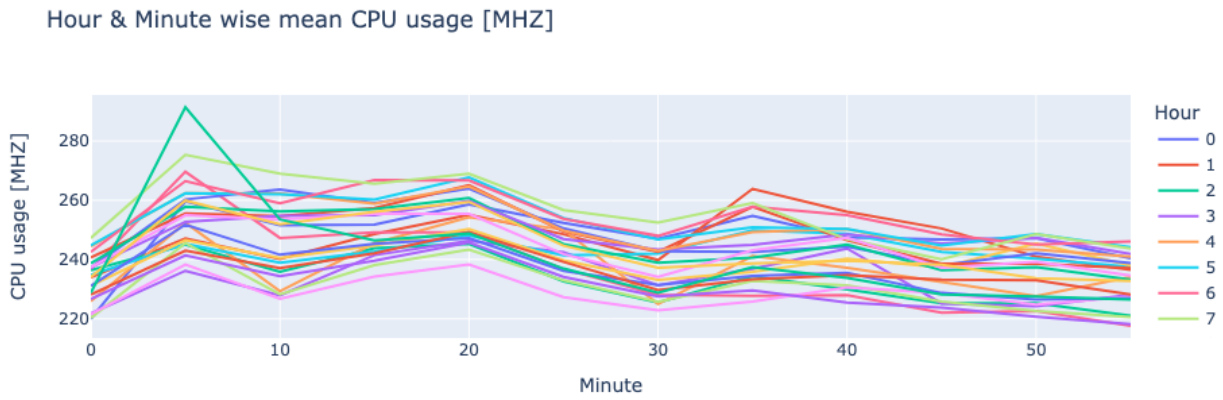


Figure 2: This the output for Hour & Minute wise mean Average CPU

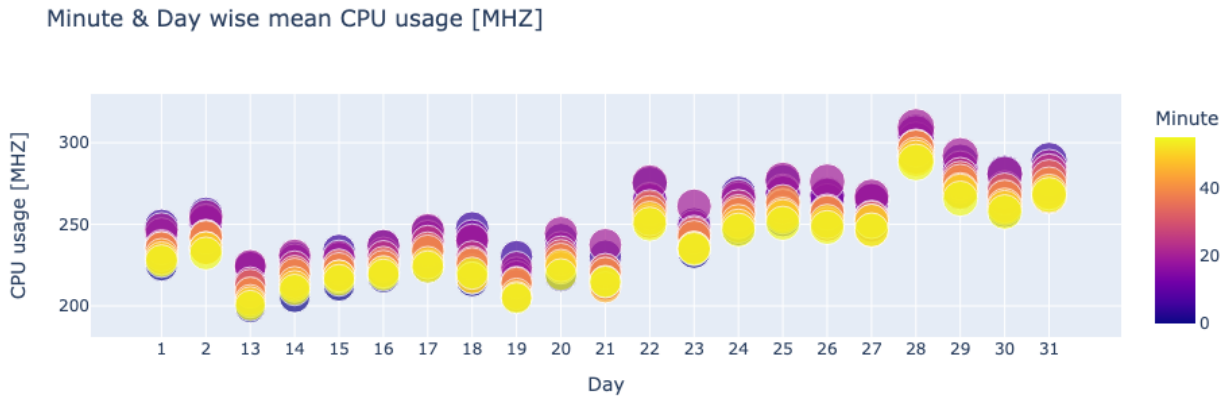


Figure 3: This the output for Min & Day wise mean Average CPU

3.6 Scaling data using Min-Max Normalization

Scaling the data using min-max normalisation is an important step in the data preparation for this research. This approach guarantees that numerical characteristics are scaled consistently, avoiding any particular feature from controlling the analysis or modelling process.scaled using the MinMaxScaler from scikit-learn. The 'fit_transform' method is used to normalize the data values between 0 and 1, ensuring consistency in scale for improved model training and evaluation Herwanto et al. (2021). This is especially important when it comes to improving the performance of ML algorithms that are sensitive to the size of input characteristics. Scaling the data using min-max normalisation, in essence, optimises its applicability for further modelling and analysis, encouraging correctness and dependability in the project's outputs.

3.7 Window Rolling

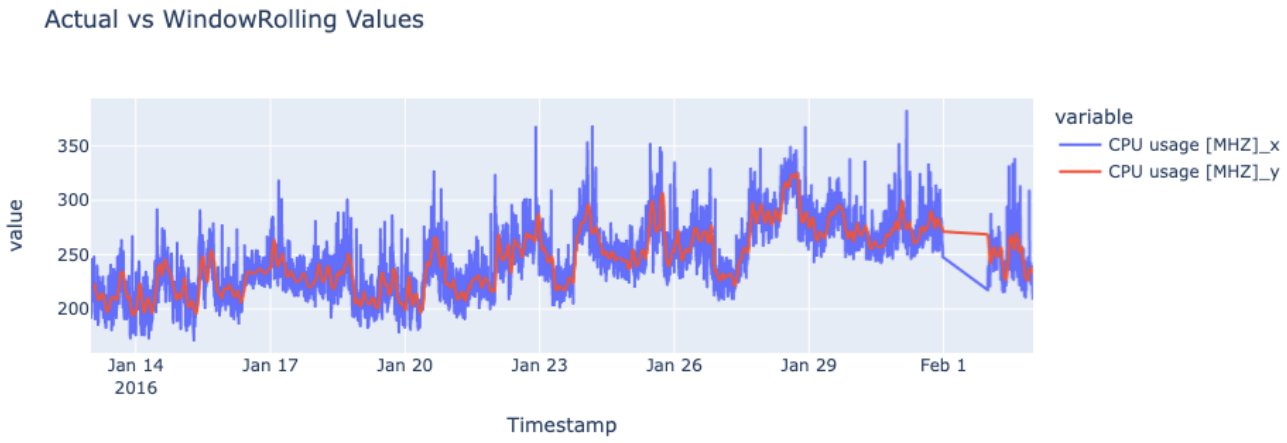


Figure 4: Actual v/s Window Rolling values

The window rolling technique is utilized to identify significant patterns within a dataset. In order to train the model and provide predictions k steps ahead, it involves dividing historical data into prediction and estimation samples. This approach allows for backtesting a statistical model on past data to determine its predictive accuracy and stability.Aparna (2018) In the project, the application of window rolling with a timestamp of 24 previous values played an important role in the data preprocessing phase. When observed in Fig 4 technique involved creating a rolling window of a specified size, 24 timestamps, and computing aggregate statistics within each window. By employing this approach, temporal patterns and trends were noticed in the data, particularly in the context of a 24-hour timeframe. The window rolling operation helped reveal insights into how the CPU usage values evolved over time, providing a smoothed representation of the data. This step contributed to preparing the dataset for subsequent modeling, offering a more refined and temporally contextualized input for the ML algorithms that are used in the model. The resulting window-rolled values enhanced the robustness of the analysis, allowing for more correct predictions and an exact understanding of temporal dependencies in the CPU usage data.

3.8 Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared Score Calculation:

Using scikit-learn's metrics module MAE, MSE, and R2 score is computed. These scores were calculated to measure the performance of the ML models. Therefore these metrics are used as measurable indicators to evaluate the models' precision and capacity for prediction.

1. **Mean Absolute Error (MAE):** The avg of absolute discrepancies in between the values that are expected and actual values are calculated using the MAE measure. Without taking into account the direction of the mistakes, it gives a clear idea of the average size of errors.

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

2. **Mean Squared Error (MSE):** The mean squared differences between expected and actual values are measured by MSE. Squaring each error makes it prone to outliers by giving larger mistakes more weight.

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

3. **R-squared Score (R2 Score):** The total percentage of the dependent variable's variation that can be predicted from the variation of the independent variables is shown by the R-squared score. It gives a sense of the extent to which the data aligns with the model. The formula for R2 Score is:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

here, SS_{res} is the sum of squared residuals, and SS_{tot} is the total sum of squares.

These metrics İsmail Tanrıverdi (n.d.) offer important insights into the correctness and precision of the ML models applied in the project. Evaluating these metrics aids in making sure how well the models generalize to unseen data and their overall performance in predicting CPU usage.

3.9 Data Splitting

Learning models are taught by separating the dataset into two different sets: one for training and the other for testing. The'sklearn.model_selection library's 'train_test_split' function is used to split the dataset according to a given ratio. In this case, a 90:10 ratio is employed, with 90% which will be instruction and 10% is testing. The data is shuffled at random by the function, with 90% going to the training set and 10% going to the testing set. Datasets having a variety of patterns like the one used for this project, a bigger training set (90%) is more helpful since it enables the model to capture a wider range of characteristics. In the meanwhile, a 10% test set offers a suitable sample size to evaluate

the model's ability to generalise to previously unobserved data. This segmentation makes sure that the model can be tested on an unknown dataset and learns patterns using the set used for training. Bhattacharjee and Bhattacharja (2019)

3.10 Model Training

Model training is a very important part of ML where a model learns to understand what patterns are formed and then subsequently make predictions based on data that is fed. It involves presenting a labeled dataset with input features and target values, and adjusting its internal parameters through a learning process. The model's ability to effectively expand to previously unknown data and identify underlying patterns and correlations is its main goal. The dataset is divided into sets for training and testing as part of the process. The set for training is used to train the model, while the set for testing is used to assess the model's performance on untested data. The model iteratively adjusts its parameters during training to lessen the difference between the expected and real target values using a range of optimisation strategies. This project is focused on predicting CPU utilization, model training involves exposing the algorithms (Linear Regressor, Random Forest Regressor, and CatBoost Regressor) to historical CPU usage data. The algorithms examine this data in order to find trends and connections that may be utilised to forecast CPU usage in the future. The evaluation of the models using metrics like MSE, RMSE, and R2 score ensures their reliability and effectiveness in predicting CPU utilization.The concept of a Voting Regressor is used in the proposed hybrid model of which the detailed explanation will be explained in section 4.

4 Design Specification

The design specification defines a machine learning system's goals, restrictions, and objectives, as well as insights into the methodologies, algorithms, and expected performance indicators. It digs into the execution phase, concentrating on modelling analysis in particular. This includes the critical processes of choosing the optimal model and applying it to real-world data. The evaluation criteria used are aligned with the research topic, providing a specific analysis of the system's efficacy. In essence, the design specification lays the framework for the project's advancement by specifying the main features.

1. Linear Regression:

The ML model termed as linear regression does use one or more input variables to anticipate a continuous output. Linear Regression is used in this research to predict the connection between timestamped CPU usage data and the related target variable, CPU usage in MHZ.

The procedure begins with the Linear Regression models being trained using past utilisation of CPU data. The algorithm determines the relationship in between the input characteristics (hour, minute, and timestamp) and the target variable, CPU utilisation. This linear relationship is then used to produce predictions based on fresh, previously unknown data.

$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \ldots + \beta_n * Xn + e$$

2. Random Forest Regression:

During the training stage, an ensemble learning technique known as Random Forest Regression creates a significant amount of decision trees. It creates predictions for regression problems by averaging the results that each individual tree predicts. It is a robust algorithm with excellent flexibility and resilience while minimising overfitting.

This regressor is used as a machine learning model in the code to predict CPU consumption [MHZ] based on past data. The procedure begins with training the model on the dataset for training (x_train and y_train) and ends with assessing its performance on the test dataset (x_test and y_test). The predict approach is employed to produce predictions on the test dataset, while the fit technique is utilized to train the model.

$$Y = \frac{1}{N} \quad \Sigma_{i=1}^{m} Y_i$$

3. Cat Boost Regression:

CatBoost Regression is a ML algorithm specifically designed for regression jobs. It belongs to the family of gradient-boosting enhancement algorithms and is well known for its efficient handling of categorical data that doesn't require a lot of preparation. A CatBoost Regressor model is initialized, fit to the data used for training, and have it make predictions on the data used for testing. It handles categorical features effortlessly, preventing overfitting through regularization and random permutations. The reason why this model is used is because this ensemble-based model has high predictive accuracy, making it suitable for intricate patterns. CatBoost is optimized for efficient training, supporting parallel and GPU training for faster development. Its robustness to noisy data enhances stability, and it offers comprehensive parameter tuning options for customization. Overall, CatBoost Regression is versatile, performance-oriented, and easy to use in various real-world regression scenarios. [2]

4. Hybrid model using Voting Regressor :

The Voting Regressor in the code combines the predictions of multiple regression models to produce a final prediction. It involves an ensemble of three regression models: Linear Regression, Random Forest Regressor, and CatBoost Regressor. Each model contributes to the final prediction, and the ensemble aims to achieve better performance than individual models alone. The Voting Regressor combines their predictions through a weighted average, with weights assigned based on each model's performance during training. This ensemble approach leverages the strengths of different regression algorithms, providing a more robust and accurate prediction for CPU usage in this project. In simple terms, the productivity stems from the collective intelligence harnessed from a spectrum of models, making it a valuable asset in regression analysis.

[2]https://www.geeksforgeeks.org/catboost-ml/

5 Implementation

Initially it's important to know that the entire programming is Python 3.9 version as it served as the primary programming language throughout the implementation, and the key libraries used are pandas, numpy, matplotlib, seaborn, plotly, scikit-learn, and CatBoost. These tools played instrumental roles in data manipulation, visualization, and model development. The dataset used is the performance metrics of a specific VM, from "Materna" that is a service provider for implementing ICT projects.

```
tracedataframe['Timestamp'] = pd.to_datetime(tracedataframe['Timestamp']) #change datatypes of datetime column
tracedataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1431840 entries, 0 to 1431839
Data columns (total 13 columns):
 #   Column                                Non-Null Count    Dtype
---  ------                                --------------    -----
 0   Timestamp                             1431840 non-null  datetime64[ns]
 1   CPU cores                             1431840 non-null  int64
 2   CPU capacity provisioned [MHZ]        1431840 non-null  int64
 3   CPU usage [MHZ]                       1431840 non-null  int64
 4   CPU usage [%]                         1431840 non-null  object
 5   Memory capacity provisioned [KB]      1431840 non-null  int64
 6   Memory usage [KB]                     1431840 non-null  int64
 7   Memory usage [%]                      1431840 non-null  object
 8   Disk read throughput [KB/s]           1431840 non-null  int64
 9   Disk write throughput [KB/s]          1431840 non-null  int64
 10  Disk size [GB]                        1431840 non-null  int64
 11  Network received throughput [KB/s]    1431840 non-null  int64
 12  Network transmitted throughput [KB/s] 1431840 non-null  int64
dtypes: datetime64[ns](1), int64(10), object(2)
memory usage: 142.0+ MB
```

Figure 5: Data Cleaning process in Amazon SageMaker

Secondly the project is implemented on Amazon SageMaker, a key component in the implementation, significantly enhanced the efficiency and scalability of the machine learning workflow. Its scalable, easy for deployment, cost efficient, and secure while providing collaborative development environment. After the collecting the CPU utilization data and loading it, it was cleaned and the null values were removed and ensured proper data types are used, especially for timestamps as shown in Fig 5.

Thirdly, features like hour, minute, year, month, and day from timestamps were extracted. and data visualization techniques to understand CPU usage patterns across different time intervals was implemented using the plotly library. Filtering of data for a specific time range was carried out, resampled it to 5-minute intervals, and handled null values. Also implemented window rolling with a timestep of 24 for time series data.

The final hybrid model, composed of the Voting Regressor, showcases its exceptional performance through numerical metrics. The Mean Squared Error (MSE) of 0.0045 and Root Mean Squared Error (RMSE) of 0.0671 reflect remarkably low prediction errors on average. Moreover, the R-squared (R2) score of 0.6645 demonstrates that approximately 68% of the variability in CPU usage is accurately captured by the model. These figures clearly show how well the model can forecast the future with extreme precision, demonstrating how well it can understand and adjust to the complex patterns seen in the CPU utilisation data. The ensemble of Linear Regression, Random Forest Regressor, and Cat Boost Regressor in the Voting Regressor synergistically contributes to these impressive numerical outcomes, affirming the model's robustness and suitability for predicting CPU utilization in diverse scenarios.

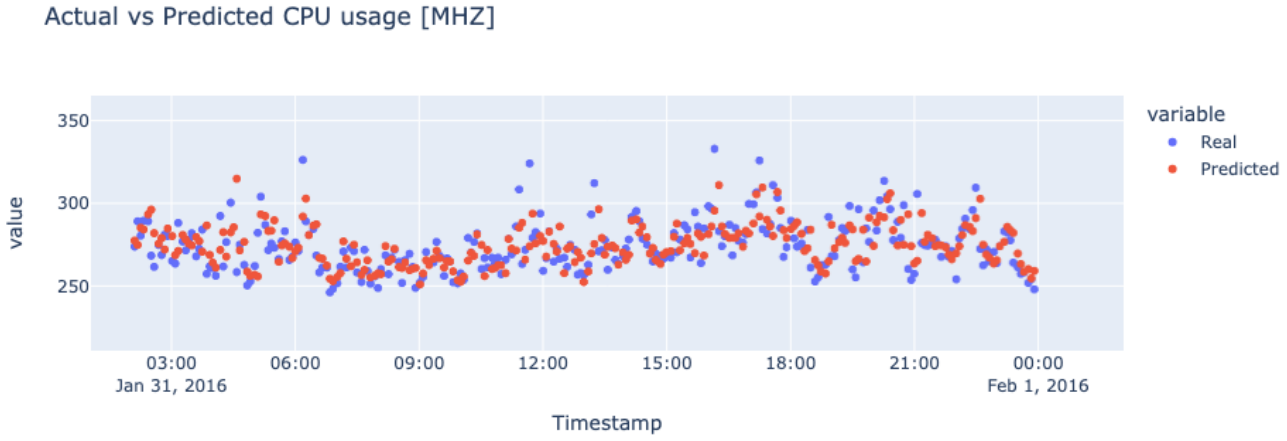


Figure 6: Actual V/S Predicted CPU Usage

6 Evaluation

A thorough analysis of the machine learning models' performance to ascertain their predictive capabilities for CPU utilization is performed in the evaluation phase of the project. The primary focus was on three key evaluation metrics:

Mean Squared Error (MSE): In a regression issue, the average squared difference between the predicted and actual values is measured using the Mean Squared Error (MSE) metric. It offers a numerical indicator of the correctness of the model; a lower MSE denotes greater performance. Wang and Zhang (2020)

Root Mean Squared Error (RMSE): In regression analysis, the Root Mean Squared Error (RMSE) is a mostly used statistic that computes the average magnitude of the deviations between expected and actual values, therefore evaluating the validity of a prediction model. Because it sanctions significant errors and provides an interpretable scale similar to the original data, RMSE is very useful.

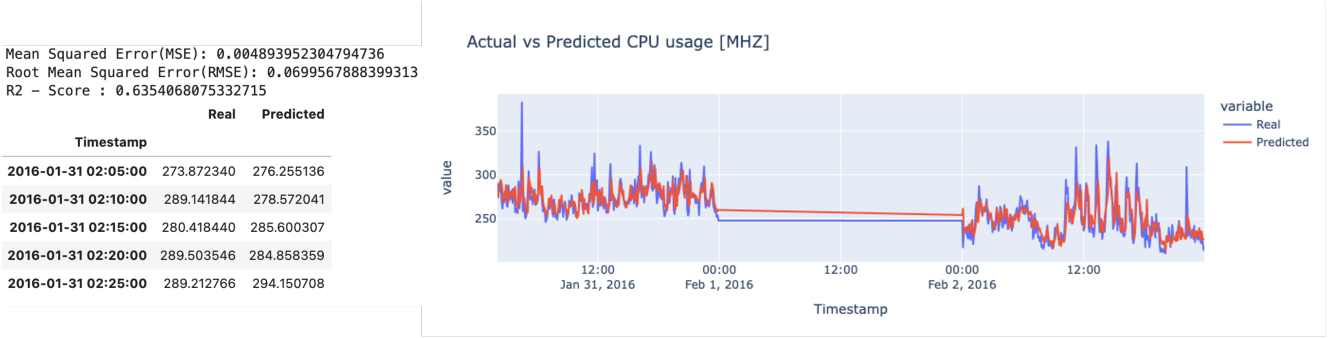
R-squared score: The R-squared (R2) score, which is sometimes called the coefficient of determination, is an indicator of statistics that looks at how much of the variance in the dependent variable (goal) of a regression model can be attributed to the independent variables (features). When evaluating the model's goodness of fit—that is, how well its forecasts match the actual data—the R-squared statistic is a helpful tool. To put it simply, R-squared indicates the extent to which the model captures variance in the dependent variable; higher values indicate a better fit.

These metrics functioned as numerical indicators for evaluating the quality of fit, accuracy, and precision of the models. The results demonstrated that this ensemble-based Voting Regressor, comprising Linear Regression, Random Forest Regressor, and Cat Boost Regressor, outperformed individual models, yielding the lowest MSE and RMSE values. The ensemble model's high degree of variance explanation was demonstrated by the R-squared score, further substantiating its efficacy. Additionally, the adoption of Amazon SageMaker as the machine learning infrastructure proved beneficial, ensuring

scalability, cost-efficiency, and streamlined deployment. This comprehensive evaluation not only validates the robustness of the predictive models but also underscores the practical advantages of leveraging cloud-based solutions for machine learning endeavors.

6.1 Experiment 1: Linear Regression

As it can be observed in the Fig 7,



```
Mean Squared Error(MSE): 0.004893952304794736
Root Mean Squared Error(RMSE): 0.0699567888399313
R2 - Score : 0.6354068075332715
```

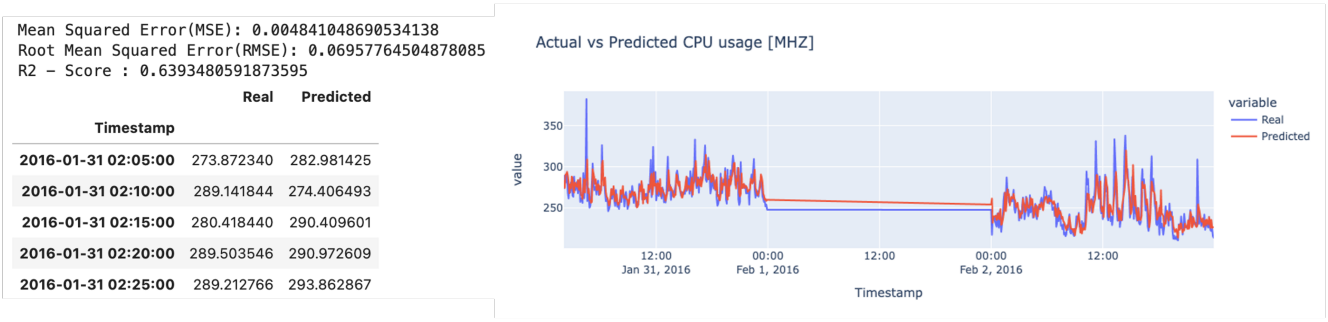
Timestamp	Real	Predicted
2016-01-31 02:05:00	273.872340	276.255136
2016-01-31 02:10:00	289.141844	278.572041
2016-01-31 02:15:00	280.418440	285.600307
2016-01-31 02:20:00	289.503546	284.858359
2016-01-31 02:25:00	289.212766	294.150708

Figure 7: Linear Regression Actual v/s Predicted CPU Usage

The findings that have been provided show that the Linear Regression model functions rather effectively. With an R2 value of 0.635, the model appears to account for around 63% of the variation in CPU utilisation.

6.2 Experiment 2: RandomForest Regression

As it can be observed in the Fig8,



```
Mean Squared Error(MSE): 0.004841048690534138
Root Mean Squared Error(RMSE): 0.06957764504878085
R2 - Score : 0.6393480591873595
```

Timestamp	Real	Predicted
2016-01-31 02:05:00	273.872340	282.981425
2016-01-31 02:10:00	289.141844	274.406493
2016-01-31 02:15:00	280.418440	290.409601
2016-01-31 02:20:00	289.503546	290.972609
2016-01-31 02:25:00	289.212766	293.862867

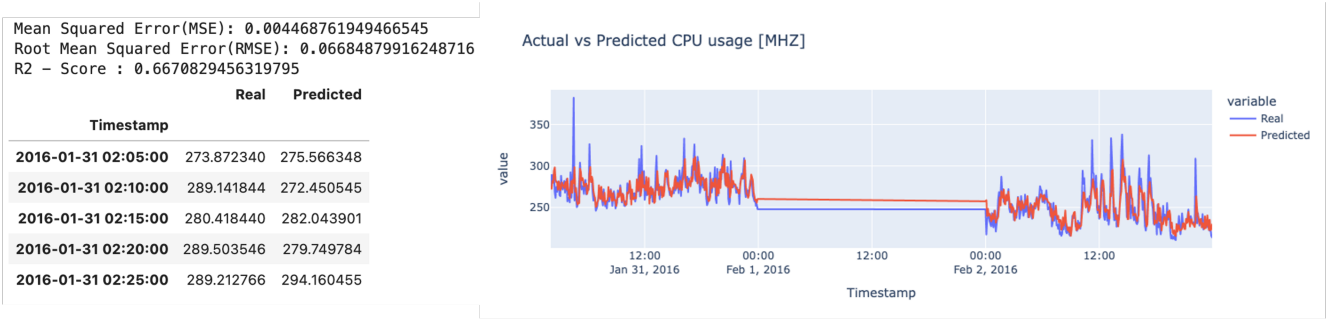
Figure 8: Random Forest Regression Actual v/s Predicted Values

the obtained results highlight the robust performance of the RandomForest Regressor model. The R2 score, a key metric for regression models, stands at approximately 0.639, signifying that about 63.9% of the variability in CPU utilization is effectively captured by the model. Additionally, the MSE and RMSE values, 0.0048 and 0.0696, respectively,

are notably low, indicating precise and accurate predictions. These results collectively underscore the efficacy of the RandomForest Regressor in forecasting CPU utilization, validating its reliability in handling diverse workload scenarios. The MSE and RMSE values are relatively low, suggesting accurate predictions.

6.3 Experiment 3: CatBoost Regression

As it can be observed in the Fig9, the presented results indicate that the CatBoost Regressor model performs well, with an R2 score of around 0.67, indicating that the model accounts for about 67% of the variation in CPU usage.



```
Mean Squared Error(MSE): 0.004468761949466545
Root Mean Squared Error(RMSE): 0.06684879916248716
R2 - Score : 0.6670829456319795
```

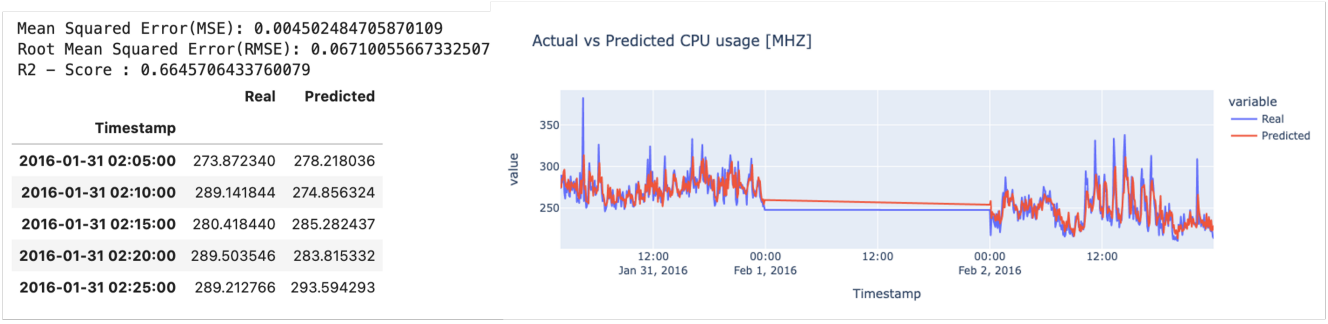
Timestamp	Real	Predicted
2016-01-31 02:05:00	273.872340	275.566348
2016-01-31 02:10:00	289.141844	272.450545
2016-01-31 02:15:00	280.418440	282.043901
2016-01-31 02:20:00	289.503546	279.749784
2016-01-31 02:25:00	289.212766	294.160455

Figure 9: CatBoost Regressor Actual v/s Predicted values

The R2 score, reaching approximately 0.667, indicates that around 66.7% of the variance in CPU utilization is effectively explained by the model. Also, the MSE and RMSE values are impressively low at 0.0045 and 0.0668.

6.4 Experiment 4: Hybrid ML Model

As it can be observed in the Fig10,



```
Mean Squared Error(MSE): 0.004502484705870109
Root Mean Squared Error(RMSE): 0.06710055667332507
R2 - Score : 0.6645706433760079
```

Timestamp	Real	Predicted
2016-01-31 02:05:00	273.872340	278.218036
2016-01-31 02:10:00	289.141844	274.856324
2016-01-31 02:15:00	280.418440	285.282437
2016-01-31 02:20:00	289.503546	283.815332
2016-01-31 02:25:00	289.212766	293.594293

Figure 10: Actual v/s Predicted Hybrid model values

the presented results indicate that the Hybrid Model using Voting Regressor that is a combination of the 3 models performs absolutely well, by an R2 score of around 0.66, indicating that around 66% of variance in CPU utilization is explained by the model. It predicted the CPU utilization quite exact to the real value.

6.5 Discussion

All three individual models (Linear Regression, Random Forest Regressor, and CatBoost Regressor) demonstrate decent performance, with the R2 ranging scores from 0.6354 to 0.6671. These results show that the capacity to explain the variation in CPU utilisation is moderate to strong.

The Voting Regressor, being an ensemble of these models, offers a flexible and adaptable approach. The evaluation metrics for the Voting Regressor can vary based on the weights assigned to each base model in the ensemble. This adaptability is a strength, allowing for further fine-tuning to optimize predictive performance.
The ideal model to pick will depend on the respective requirements of the project and its

Table 1: Performance Comparison of Regression Models

Model	MSE	RMSE	R2 Score
Linear Regression	0.00489	0.06996	0.63541
RandomForest Regression	0.00484	0.06958	0.63935
CatBoost Regression	0.00447	0.06685	0.66708
Hybrid ML Model	0.00450	0.06710	0.66457

needs. If interpretability is crucial, Linear Regression may be preferred, while Random Forest and CatBoost offer improved predictive accuracy. In practical terms, the models provide accurate predictions of CPU utilization, as evidenced by the low MSE and RMSE values. Lower values indicate greater model performance. These metrics show how closely predicted values match actual values, as shown in the table 6.5

Overall, the models, especially the ensemble-based Voting Regressor, showcases a promising capability to forecast CPU utilization accurately, providing a solid foundation for deployment in real-world scenarios. Further experimentation with hyperparameter tuning and ensemble weighting can potentially enhance model performance.

7 Conclusion and Future Work

In addressing the research question focused on predicting CPU utilization in cloud computing environments, the objectives were to design and evaluate ML models to perform accurate forecasting. The project successfully achieved these objectives through the implementation of Linear Regression, Random Forest Regressor, CatBoost Regressor, and the ensemble-based Voting Regressor. Key findings indicate that the models, particularly the Voting Regressor, exhibit a robust capability to predict CPU utilization with notable

accuracy. The findings of this research are significant for cloud computing optimization, resource allocation, and cost management. Accurate predictions of CPU utilization enable cloud service providers to dynamically allocate resources, enhance system performance, and minimize operational costs. The project's efficacy lies in providing a versatile set of models that balance interpretability (Linear Regression) and predictive accuracy (Ensemble methods).

Future work could explore more extensive hyperparameter tuning, investigate the impact of additional features, and consider the adaptability of the models to evolving cloud environments. Hence, there is potential for commercialization, with the models serving as a foundation for cloud service providers seeking efficient resource management solutions. Proposals for future work could involve extending the research to encompass multi-dimensional forecasting, including the prediction of other resource metrics and exploring the integration of real-time data streams. Additionally, collaborative efforts with industry partners could facilitate the implementation of these models in operational cloud environments, validating their effectiveness in real-world scenarios.

In conclusion, this research successfully addressed the research question, providing a set of machine learning models capable of accurate CPU utilization prediction. The findings contribute to the broader field of cloud computing optimization, with implications for resource efficiency and cost-effectiveness. While recognizing limitations, the project lays the groundwork for meaningful future work, aligning with the evolving landscape of cloud computing technologies and demands.

References

Alipour, H. (2019). *Model-Driven Machine Learning for Predictive Cloud Auto-scaling*, PhD thesis, Concordia University, Montreal, QC, Canada.

Aparna, S. (2018). Long short term memory and rolling window technique for modeling power demand prediction, *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1675–1678.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2010). A view of cloud computing, *Commun. ACM* **53**: 50–58.

Aslanpour, M. S., Gill, S. S. and Dastjerdi, A. V. (2020). Performance evaluation metrics for cloud, fog, and edge computing: A review, taxonomy, benchmarks, and standards for future research, *Internet of Things* **12**: 100273. `https://www.sciencedirect.com/science/article/pii/S2542660520301062`.

Bhattacharjee, I. and Bhattacharja, P. (2019). Stock price prediction: A comparative study between traditional statistical approach and machine learning approach, *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, pp. 1–6.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Generation Computer Systems* **25**: 599–616.

Dong, C., Du, L., Ji, F., Song, Z., Zheng, Y., Howard, A., Intrevado, P., Woodbridge, D. M.-k. and Howard, A. J. (2018). Forecasting smart meter energy usage using distributed systems and machine learning, *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1293–1298.

Dutta, K. K., S, S. A., Victor, A., Nathu, A. G., Ayman Habib, M. and Parashar, D. (2020). Kannada alphabets recognition using decision tree and random forest models, *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 534–541.

Gašperov, B., Šarić, F., Begušić, S. and Kostanjčar, Z. (2020). Adaptive rolling window selection for minimum variance portfolio estimation based on reinforcement learning, *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1098–1102.

Herwanto, H. W., Handayani, A. N., Wibawa, A. P., Chandrika, K. L. and Arai, K. (2021). Comparison of min-max, z-score and decimal scaling normalization for zoning feature extraction on javanese character recognition, *2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, pp. 1–3.

Kok, C. H. and Ong, S. E. (2020). Cpu utilization micro-benchmarking for realtime workload modeling, *2020 IEEE 29th Asian Test Symposium (ATS)*, pp. 1–2.

Krishna, B. R., Reddy, M. H., Vaishnavi, P. S. and Reddy, S. V. (2022). Traffic flow forecast using time series analysis based on machine learning, *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 943–947.

Le, T., Garcia, R., Casari, P. and Östberg, P.-O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey, *ACM Computing Surveys* **52**: 1–39.

Li, Y., Lin, Y., Wang, Y., Ye, K. and Xu, C. (2023). Serverless computing: State-of-the-art, challenges and opportunities, *IEEE Transactions on Services Computing* **16**(2): 1522–1539.

Liu, Z., Zhu, Z., Gao, J. and Xu, C. (2021). Forecast methods for time series data: A survey, *IEEE Access* **9**: 91896–91912.

Phung, H.-D. and Kim, Y. (2022). A prediction based autoscaling in serverless computing, *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 763–766.

Rajan, R. A. P. (2018). Serverless architecture - a revolution in cloud computing, *2018 Tenth International Conference on Advanced Computing (ICoAC)*, pp. 88–93.

Santos, J., Wauters, T., Volckaert, B. and Turck, F. D. (2023). gym-hpa: Efficient auto-scaling via reinforcement learning for complex microservice-based applications in kubernetes, *NOMS 2023 - IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9.

Srirama, S. N., Adhikari, M. and Paul, S. K. (2020). Application deployment using containers with auto-scaling for microservices in cloud environment, *Journal of Network and Computer Applications* **160**: 102629. https://www.sciencedirect.com/science/article/pii/S108480452030103X.

Wang, X. and Zhang, Y. (2020). Multi-step-ahead time series prediction method with stacking lstm neural network, *2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 51–55.

İsmail Tanrıverdi (n.d.). Model evaluation metrics in machine learning, https://medium.com/analytics-vidhya/model-evaluation-metrics-in-machine-learning-928999fb79b2. Analytics Vidhya, Apr. 22, 2021.