Cloud Resource Forecasting and Prediction Methodology Framework

MSc Research Project
Data Analytics

Karim Ladouari
Student ID: 20168195

School of Computing
National College of Ireland

Supervisor: Mr Zahid Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Karim Ladouari
Student ID:	20168195
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Mr Zahid Iqbal
Submission Due Date:	01/02/2023
Project Title:	Cloud Resource Forecasting and Prediction Methodology Framework
Word Count:	7824
Page Count:	21

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:	[illegible]
Date:	1st February 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):	

Cloud Resource Forecasting and Prediction Methodology Framework

Karim Ladouari
20168195

Abstract

Cloud computing has dramatically changed the management of web applications, software development and information technology. Anyone, anywhere in the world, can provision a virtual machine on the platform of one of the Cloud computing service providers. One of the many advantages of Cloud computing is to provide resources on demand. The challenge for Cloud customers is to size and optimise Cloud applications while controlling over-capacity. The current research established a mechanism to select the optimal time series forecasting technique to predict Cloud systems requirements. A Deep learning Recurrent Neural Network (RNN) time series technique combined with Extreme Gradient Boosting (XGBoost) feature selection algorithm is designated to produce the optimal short-term prediction, while Seasonal Auto-Regressive Integrated Moving Average with exogenous factors (SARIMAX) combined with Random Forest is for long-term predictions. The defined prediction methodology framework would help Cloud application owners maintain their running cost and associated low carbon footprint and improve performance.

1 Introduction

Cloud computing plays an essential part in people's lives across the globe by primarily impacting the growth of the world economy and influencing individuals interactions through the rise of Cloud-based social networks.
Oracle corporation[1] has forecasted that Cloud application and associated data will grow by 50% by 2025 and by 80% by 2028, with the Cloud computing market size expected to more than double from 450 billion US dollars in 2021 to over one trillion US dollars by 2028.
International Business Machine (IBM) described in a publication[2] the last decades' Cloud computing innovation and transformation with the creation of virtual machines, the exponential increase of systems computing power and manufacturing costs reductions. To a certain extent, Cloud computing transformed how web-based applications are designed, deployed and scaled. The side effect of this transformation is for Cloud computing customers to maintain and scale deployed applications as the complexity of technologies increases over time. The challenge for commercial applications is to forecast and identify the business growth and the associated systems requirements.
Major Cloud computing corporations like Amazon Web Services, Microsoft Azure or

[1]https://www.oracle.com/
[2]https://www.ibm.com/Cloud/blog/Cloud-computing-history

Google Cloud offer auto-scaling solutions, allowing clients to horizontally or vertically scale their deployed systems automatically based on thresholds. For some clients, auto-scaling is unsuitable, and systems with associated computing resource sizing are based on intuition or experience. As a result, the absence of autoscaling or forecasting mechanisms leaves web-based application architects to oversize their system infrastructure requirements and consequently their running costs.
This research aims to provide Cloud computing web-based application owners with the most suitable technique to forecast, predict and optimise Cloud servers' resource requirements. The motivation is to manage allocated systems resources efficiently and, by association, improve systems performance and quality of service improvement and reduce energy consumption and carbon footprint. The research question for this survey is defined as what is, based on a selection of commonly known methods, the most optimal forecasting technique combined with feature selection to predict cloud resource workload. The time series data from an existing commercial company Cloud-based application is leveraged to achieve benchmark results. The experimental approach for this study consists of unique application features from the collected data associated with time series forecasting techniques such as Triple Exponential Smoothing (TES), Seasonal Auto-Regressive Integrated Moving Average (SARIMA), SARIMAX, RNN, Long Short-term Memory (LSTM), and Gated Recurrent Unit (GRU) combined with Linear regression, Random Forest and XGBoost machine learning feature selection algorithms.
This paper is designed as follows Section 2 outlining the related work and background for time series forecasting techniques and features selection for Cloud applications resource consumption, and Section 3 describes the research methodology, Section 4 the design specification and Section 5 the implementation. Section 6 discusses the experiment results, Section 7 the discussion and Section 8 the conclusion and further work.

2 Related Works and Background

2.1 Cloud Computing

Lisdorf (2021) described the evolution steps of Cloud computing from its creation to its current state. The author started with the notion of Cloud computing which appeared in the middle of the nineties with a group of engineers from the Netscape corporation working on sharing computing resources. The term Cloud computing emerged in the beginning of the 21st century, defined by the industry at the time as data and server services located somewhere in the "Clouds".
From there, Cloud computing evolved into a self-service industry offering shared, scalable and elastic IT infrastructure through the internet. Cloud computing is today defined by three service models. Software as a service (SaaS) where the client access an application over the internet entirely managed by the vendor. Infrastructure as a Service (IaaS) is the service offered to clients providing data and computing resources. Finally, Platform as a service (PaaS) is the service providing consumers with development and web applications environments coupled with infrastructure.
Additionally, consumers' access infrastructure can be defined into three types of Cloud deployment models. The private Cloud is the deployment model where the Cloud resources are dedicated and accessible to only one client. The public Cloud is where the resources are shared across multiple tenants via the internet, and the hybrid Cloud combines private and public Cloud.

There are multiple Cloud computing vendors in the market, but the two principal ones are Amazon web services (AWS) and Microsoft Azure. AWS is the market leader with more than 33%, followed by Microsoft Azure with 21 percent. The author defined AWS as a Cloud computing service provider prioritizing business over technology with a low margin and high volume strategy, focusing on the customer experience, and offering consumers a simple and modular service like EC2 compute power and S3 storage. Microsoft Azure transitioned later to Cloud computing but was able to catch mainly due to a strong culture in computer and operating systems and a strategy focusing on building full-service ecosystems and cultivating solid relationships with enterprises of any size.
Hameed et al. (2016) wrote a comprehensive survey about the management of Cloud computing resource allocation and its associated challenges. The authors started their work by describing the mechanism for Cloud clients to add or remove a resource's capacity to an environment using Simple Object Access Protocol (SOAP) Restful application programming interface (API) utilities or elastic scaling apparatus.
Elastic scaling describes the critical feature offered to Cloud computing consumers to manage and scale their resource allocation dynamically. Elastic scaling can be defined as horizontal when the scaling is completed at the virtual machine level or as vertical at the server capacity level. The elastic scaling functionality allows clients to scale up or down on-demand Cloud service or any resource type such as system capacity (CPU and RAM) or virtual machine(server, pod or container). The main benefits are fast optimisation results with reduced labour requirement; however, its application is not exhaustive. The authors described auto-scaling as three methods, reactive, proactive and hybrid.
The reactive auto-scaling is actioned when the resource level reaches the designated threshold, proactive is based on the resource prediction load, and hybrid is a combination of reactive and proactive auto-scaling. The reactive auto-scaling adoption is the most prevalent, even though challenges exist, particularly with capacity throughput variability or provisioning performance degradation. Proactive auto-scaling is the most promising method, but it needs significant effort and data as it leverages machine learning, and results are not always accurate. The authors also pointed out that many Cloud consumers do not leverage the elastic scaling service for multiple reasons, even if it is widespread.

2.2 Time Series Forecasting: State of Art

This section describes the definition of the time series forecasting algorithms utilised for this research.

2.2.1 ARIMA

Autoregressive integrated moving average (ARIMA) is one of the most popular time series forecasting techniques introduced more than fifty years ago by Jenkins et al. (1970). ARIMA is a unidimensional stochastic time series analysis methodology. ARIMA combined the autoregressive model (AR) and the moving average model (MA) techniques. AR is used to analyse the dependencies between observations and their past values, while MA is the approach describing the linear dependency between observations and their residual error terms. Finally, the I in ARIMA describes the number of order d of differencing needed to transform the time series to a stationary state. The authors define the stationary state as a time series without a trend, seasonality or periodic changes and with a constant variance or autocorrelation over time. They describe ARIMA(p, d, q) with the orders of the model required (p, d, and q) respectively for AR, I and MA to capture the

main dynamic data attributes.
AR model of order p with x_t as a stationary observation, α as a constant, ϕ_i as the auto-correlation coefficients at lags 1 to p and ϵ_t as the residual are expressed as follows:

$$x_t = \alpha + \sum_{i=1}^{p} \phi_i x_{t-1} + \epsilon_t \tag{1}$$

MA model of order q with μ a constant and θ_i the past forecast errors coefficient at lags 0 to q are expressed as follows:

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \epsilon_{t-1} \tag{2}$$

The technique SARIMA and SARIMAX are the extension of ARIMA that explicitly and respectively supports univariate and multivariate time series data with a seasonal component m for (P,D,Q) m.

2.2.2 Neural Network

An artificial neural network (ANN) is a succession of algorithms aspiring to identify a dataset's underlying relationships through a process reproducing how the human brain functions. A neural network consists of a system of multiple neurons. Neural networks are made of multi-layer interconnected nodes, also known as perceptrons consisting of elements receiving data through an input layer, then to one or more hidden layers computing the information to an output layer delivering final results. Hidden layers perform nonlinear transformations by applying an activation function to the result of each node value and a weight multiplication.
Essentially, the purpose of the activation function is to determine whether or not a neuron should be activated. The author defines back-propagation as the essential algorithm optimising the prediction accuracy of a neural network model training process. Back-propagation works back from output nodes to input nodes optimising weights one layer at a time using the gradient of the selected loss function. The back-propagation depends strongly on the choice of the optimiser algorithm and its learning rate. Each model training iteration passes the whole training set forward and backwards. Ultimately the process is completed when it reaches the specified number of iterations or epochs.

2.2.3 Recurrent Neural Network

A RNN is a method of handling sequential data types such as time series, text or language processing or speech recognition. RNN was designed to include dependencies between time series data points. Standard artificial neural networks are not structured to process the relationship between a time series data point and its previous outputs. RNNs contain a concept of memory allowing to save the previous inputs states to generate the following sequence output.
For each input timestep x_t, hidden state sequence h_t, the output o_t, the bias vectors b_h, b_y, the weights W_{hx}, W_{hh}, W_{yh} and the respective activation functions f and g are expressed as follows:

$$h_t = f(w_{hx} x_t + w_{hh} h_{t-1} + b_h)$$

$$o_t = g(w_{yh} h_t + b_y)$$

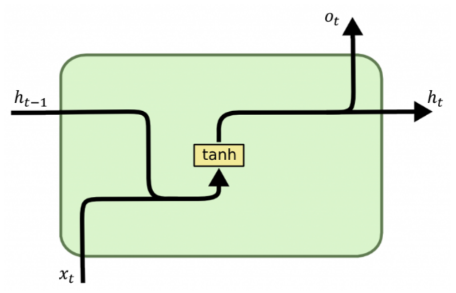

Figure 1: RNN Architecture [3]

The hidden state captures and memorize all the information previous time sequence. RNNs pitfall is the vanishing gradients problem with long data sequences, which makes is not the right choice for large time series. A vanishing gradient happens when the back-propagation gradients computing the weight from the previous output will, with time, get close to zero and prevent improving new weights. LSTM and GRU were designed to circumvent the vanishing gradient problem.

2.2.4 Long Short Term Memory Network

Hochreiter and Schmidhuber (1997) proposed the algorithmn LSTM as an improvement to overcome the long-term dependencies gap introduced by RNN. Figure 2 describes the LSTM networks designed by the authors.

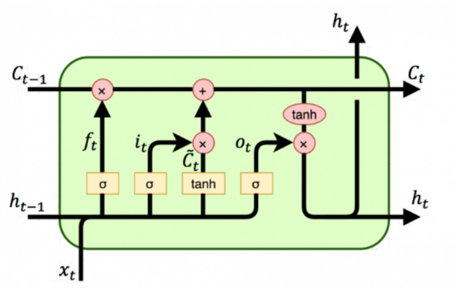

Figure 2: LSTM Architecture[3]

LSTM cell state is the top horizontal line going through the neural network (Figure 2). It is an essential part of LSTM, conveying the information through the network, computing the output value and deciding the information to store or remove. The algorithm contains gates with sigmoid activation functions regulating the data towards the cell state. Gates filter and let through information only if the sigmoid function results are different from 0. The forget gate layer f_t is responsible for eliminating what information needs to be discarded. The input gate layer i_t decides what input values to update from the memory state. The tanh gate layer C_t looks at the candidate value addition to the cell state and finally the output gate layer o_t computes the final output.
For each input timestep x_t, hidden state sequence h_t and C_t, the bias vector b_f, b_i, b_C, b_0, the weights W_f, W_i, W_C, W_0 and the respective activation functions σ and tanh are ex-

[3]Source: http://dprogrammer.org/rnn-lstm-gru

pressed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

2.2.5 Gated Recurrent Unit

GRU neural networks were introduced by Chung et al. (2014). GRU is the most recent addition to recurrent neural networks, and like LSTM, it seeks to solve RNN's vanishing gradient problem. It is similar to LSTM but with a few distinctions. The authors established the hidden state as the vehicle to transfer information through the network. GRU has two gates , the update gate, which is dedicated to deciding what information to keep or forget, and the reset gate looks at how much past data to ignore.

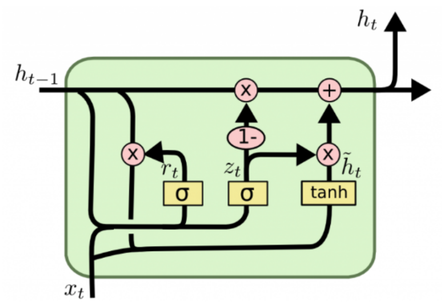

Figure 3: GRU Architecture[4]

For each input timestep x_t, hidden state sequence h_t , the bias vector b_z, b_r, b_h, the weights W_z, W_r, W_h and the respective activation functions σ and tanh are expressed as follows:

$$z_t = \sigma(W_z * [h_{t-1}, x_t] + b_z)$$
$$r_t = \sigma(W_r * [h_{t-1}, x_t] + b_r)$$
$$\tilde{h}_t = \tanh(W_h * [r_t * h_{t-1}, x_t] + b_h)$$
$$h_t = (1 - z_t * h_{t-1} + z_t * \tilde{h}_t)$$

2.3 Related Works

Recent work including Ilyushkin et al. (2017), Podolskiy et al. (2019) and Poppe et al. (2022) concentrated on improving Cloud resource workload management with some relative success. Many techniques and algorithms were leveraged to predict Cloud systems capacity requirements. Starting with Janardhanan and Barrett (2017), who analysed the forecasting and prediction of systems' CPU consumption time series using ARIMA and LSTM with the outcome of LSTM performing better than ARIMA, exhibited by the Root mean square error (RMSE) score from twenty-four hours of predictions. Even though the

[4]Source: http://dprogrammer.org/rnn-lstm-gru

two techniques used are complementary and valuable for time series forecasting, this research lack depth as the data is only univariate.
Duggan et al. (2017), Zhang et al. (2018) and Shaw et al. (2018) provided similar research forecasting and predicting CPU workload using extracted data from multiple major Cloud service providers (CSP). RNN among other techniques, was used to forecast and predict CPU utilisation multiple time steps ahead. As a result, RNN outperformed all other techniques showing high precision in predictions on unseen data but not more than three steps ahead. The side effect of RNN is that it loses efficiency with long time series sequences due to a vanishing gradient.
Yadav et al. (2022) discuss in their article the concept of proactive Cloud computing auto-scaling approach combined with LSTM to optimise service latency through provisioning optimisation. The design of the auto-scaling process incorporates the LSTM optimal prediction for systems resource requirements with the Service level agreement (SLA) violations. The experimentation is essentially the forecast of univariate Cloud systems CPU workload time series using LSTM along ARIMA and Support vector machine (SVM), comparing the prediction accuracy performance score resulting in LSTM outperforming all the other techniques.
Chapram et al. (2018) use the same method and concept and achieve similar results in their research as LSTM outperformed ARIMA to predict workload resources from Cloud web servers. As highlighted in previous research, predicting Cloud systems CPU workload using only univariate data is limited. The applications running on the Cloud systems are more certainly influential to the CPU consumption.
Further, Li et al. (2022) research the impact of combining bi-directional LSTM (BILSTM) and GRU for CPU resource usage forecasting and prediction and compared performance and accuracy against ARIMA, LSTM, BILSTM and GRU models. The experimental outcomes of using a data set extracted from Google Cloud showed that the model combined with techniques BILSTM and GRU improved performance and accuracy compared to the traditional single prediction methods.
Using a similar approach, Ouhame and Hadi (2019) present a method combining a Vector Autoregressive (VAR) model and LSTM forecasting Cloud computing resource workload multivariate time series. Combining VAR and LSTM enables capturing the linear and non-linear trends from the extracted multivariate Bitbrains data. The proposed method results were more effective than ARIMA/LSTM and GRU models. Combining techniques is a good approach as it leverages the strength of both techniques. The authors' motivation lay more on incorporating techniques jointly, ignoring the added value of a multivariate dataset.
Moreno-Vozmediano et al. (2019) propose using the Support Vector Machine (SVM) method to forecast and predict Cloud resource workload univariate time series rather than ARIMA or neural networks, as they suggest methods have limitations. Considering neural networks suffer from potential local minima and do not produce one unique solution, ARIMA holds linear dependencies with the past. However, like any other machine learning algorithm, SVM also has gaps as it is unsuitable for large data sets and does not perform well with data sets with noise, which is one of the times series characteristics.
Calheiros et al. (2015) propose integrating an ARIMA Cloud resource forecasting mechanism into a pro-active auto-scaling process. The forecast produced by the ARIMA model generates an optimisation with an associated action dynamically provisioning or decommissioning a virtual machine when the prediction exceeds the designated threshold. The authors used a simulation to generate the dataset with a pre-determined linear pattern

and reduced variability trend. The pitfall of this research resides in the experiment limitation of using engineered data as it lacks introducing unpredictability in time-series forecasting model fitting.
Chen, Hu, Min, Zomaya and El-Ghazawi (2020) suggest predicting Cloud resource workload with high dimension and high variability dataset by combining a feature selection top-sparse auto-encoder and GRU for time series forecasting. The authors use a neural network auto-encoder and principal component analysis (PCA) dimensionality reduction algorithm for feature selection options. The authors' motivation for not only using PCA is due to its limitation for dimensional reduction of high-variance data. The experiment uses a different workload and various prediction lengths to evaluate every possible scenario. As a result, the selected technique's prediction outperforms RNN, LSTM and GRU.
Ullah et al. (2021) propose a similar concept but with different techniques by combining VAR, Convolutional Neural Network (CNN) and LSTM to forecast Cloud computing resource utilisation. The authors use VAR to filter the linear relationship within the multivariate dataset extracted from 1750 virtual machines from the Bit-brains Cloud data centre. They then extract the influential features using CNN to forecast and predict CPU, memory, network and disk usage using LSTM. As a final step, the authors measure the prediction accuracy of the selected method, compare the score against traditional forecasting techniques and conclude it improves the accuracy by a least 3.8%. Research exploting feature selection highlight how resource workload prediction performance can be improved, but using only one algorithm reduces the accomplishment potential.
Xie et al. (2022) present a prediction system for Cloud micro-services containers resource workload using a hybrid model consisting of ARIMA and TES. The hybrid model combines the predictions from ARIMA and TES. The experimentation uses extracted data from AWS Elastic Compute Cloud to predict diverse workload containers' CPU. The authors measure the hybrid model prediction accuracy using the Mean squared error (MSE) and compare it against ARIMA, TES and neural networks models. As a result, the hybrid model improved the prediction accuracy by at least 20%. The limitation of the authors' research included using MSE exclusively as outliers can impact MSE significantly.
Kumar et al. (2021) conduct a comparative study of multiple predicting and forecasting techniques. This study aims to determine which method among Linear regression, SVM, K-Nearest Neighbors (KNN) and ARIMA is the most accurate in predicting CSP workload resources using two distinctive univariate time-series datasets. The central motivation from the authors is also to compare the forecasting efficiency between traditional machine learning algorithms and a time-series forecasting technique like ARIMA. The authors select MSE, Mean absolute error (MAE) and RMSE as criterion scores to measure the effectiveness of each prediction. ARIMA comes out as the most accurate predicting technique for both datasets. It would have been also valuable to compare ARIMA model's results against neural network time series forecasting techniques.
Mehmood et al. (2018) put forward a research predicting CSP resource workload using stacking KNN and Decision-tree machine learning algorithms. The extracted data is multivariate. The model prediction results are compared and evaluated against machine learning and neural network forecasting techniques. The outcome shows an improvement in accuracy by at least 2%. While the method is very innovative, this research needs to include feature selection to help refine which variables are influential in improving predictions.
Even though the cited articles are inspiring, they all require more depth and breadth. Most of the applied methodologies are based on univariate data set experiments. Mul-

tivariate modelling would undoubtedly introduce more precision as features generated from the Cloud applications would most likely influence and impact Cloud system resource usage. Additionally, there is very little leveraging of feature selection; when there is, it is limited to one algorithm. Introducing multiple feature selection techniques will increase the likelihood of selecting the most influential features to reduce noise and overfitting and increase accuracy.
The most apparent gap in the existing research is in combining multiple feature selection techniques with the most successful time-series forecasting learnings to increase the precision of Cloud resource workload predictions. Finally, most experiments mentioned in the highlighted articles use short-length datasets. The introduction of a more extensive data set would add variability and complexity to the model training process, with results to reduce the risk of overfitting.

3 Methodology

The research methodology consists of six stages, namely data collection, data pre-processing, data transformation, data selection, data modelling , evaluation, predictions and results, as shown in Figure 4.

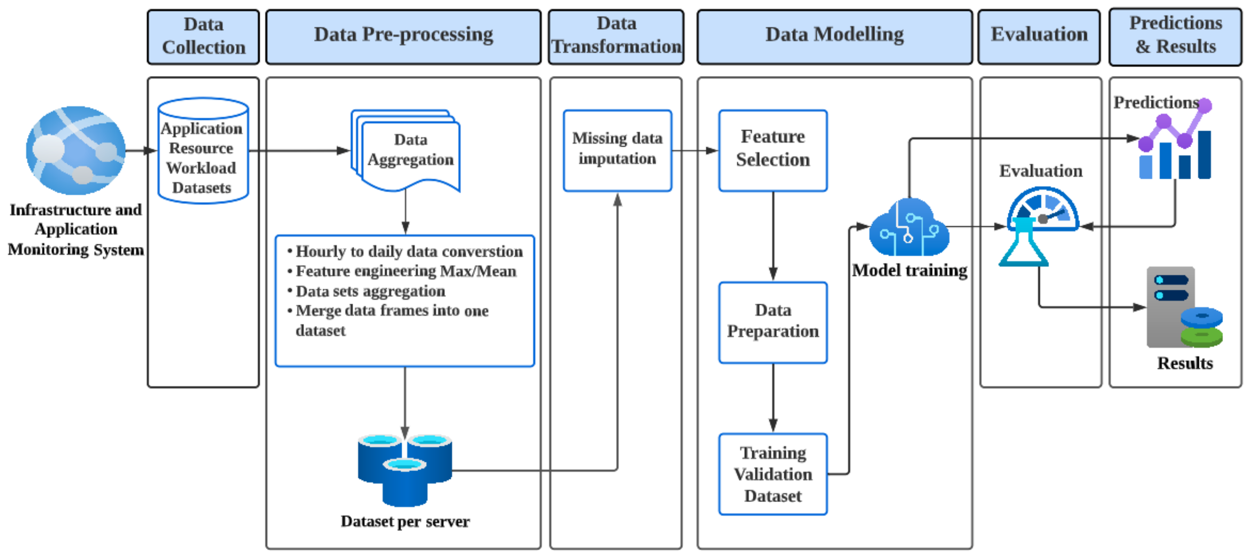

Figure 4: Research Methodology

3.1 Methodology Stages

3.1.1 Data collection

The workload resources (CPU and RAM) and application data are all recorded on the monitoring system. The data is stored on an hourly basis and categorised by customers, application services and resources workload. The data collection stage consists of programmatically querying and extracting the data using multiple Python[5] API calls. All the extracted systems resource performance and associated application features data activity will be saved as dedicated Comma-separated values (CSV) files for each customer.

[5]https://www.python.org

3.1.2 Data Pre-Processing

Data pre-processing involves cleaning and preparing the data for feature selection and model fitting. The data format required for the steps following data pre-processing is time series per server with daily resources workload. However, the extracted data consists of a time series data set for all servers hourly resource consumption and a time series data set for the seven application attributes hourly metrics. The feature engineering process transforms raw data into features, and the hourly to daily conversion task of the times series datasets is an opportunity to introduce new features. The data sets conversion from hourly to daily is not only limited to averaging values. Computing the daily maximum or highest value factor for the capacity consumption and each application component is essential for measuring resources workload capacity. After the data pre-processing section completion, a daily time series for each server representing the relationship between the application components metrics and its resource consumption is computed. The final steps involve aggregating all the time series datasets and finally producing a dataset per server.

3.1.3 Data Transformation

The data transformation consists of identifying and imputing missing data points. Jadhav et al. (2019) propose a complete survey about imputation techniques comparing multiple algorithms such as mean, median, KNN, predictive mean matching, Bayesian Linear regression, Linear regression, non-Bayesian, and random sample. The authors suggest KNN is the best missing data imputation technique as it outperforms all others. Additionally, they recommend not to impute variables containing more than 16% missing data points. As a first step, every server dataset is programmatically scanned to identify and delete their independent variables or data frame columns with more than 16% of missing data points.
Applying the imputation technique on data set features missing data points completes data transformation. Makwana et al. (2013) suggest using the elbow method to determine the number of optimal K cluster for KNN imputation. This method helps choose the optimal K number of clusters to use by selecting the elbow of the curve from the K values and its cost function plot. The identified missing points are replaced using the K-Nearest Neighbor method with the optimal K number of clusters.

3.1.4 Data Selection

The data selection or feature selection is the process of isolating the most influential and relevant features for the selected server data set model fitting.
Guyon and De (2003), Chen, Dewi, Huang and Caraka (2020) and Alsahaf et al. (2022) suggest using Linear regression, bagging Random Forest and boosting XGBoost feature selection methods. Random Forest combined with Recursive Feature Elimination (RFE) uses impurity or Gini importance to identify the significant features. Feature selection is completed by measuring how, on average, each feature decreases the impurity over all trees generated by the Random Forest algorithm. Like Random Forest, XGBoost combined with RFE uses Gini importance to identify influential features.
The difference resides in that XGBoost uses a sequential procedure by adding each feature one at a time and measuring its importance each time. Linear regression feature selection is completed by iteratively running a stepwise selection function against all the predictors and keeping only the one contributing significantly to the model. However,

Linear regression has conditions which should be met before proceeding with predictions and are expressed as follows:

- Variables and residuals are normally distributed
- Linear relationship between independent and dependent variables
- Homoscedasticity
- Reduced independent variables multicollinearity
- Independence of the observations

The data selection phase consists of running the three chosen feature selection techniques against the selected data set to identify and record their influential features. Random Forest and XGBoost are combined with RFE for scoring and ranking the predictors' influence, and only the features scoring more than ten are selected.
In contrast, the preliminary requirements for the Linear regression feature selection are first to check the normal distribution of all variables. Any variables not normally distributed are transformed using the Box-Cox transformation method. Furthermore, highly correlated variables scoring more than 80% at the Pearson score are removed. The Linear regression feature selection proceeds with the stepwise selection function against all the remaining predictors, and only the statistically significant ones with a p-value less than 0.05 are selected. Ultimately, Linear regression assumption plots are generated and verified.

3.1.5 Data Modelling

Data Modelling involves fitting and training models using selected algorithms. This research cornerstone lies in model fitting, combining feature selection techniques and time series forecasting algorithms to identify the best method to predict system resource workload. However, the selected algorithms require the completion of preparatory tasks on the dataset before starting fitting models.
As a first preparation task and specifically for the ARIMA technique, the data set is transformed using the Box-Cox method, but only if it is not normally distributed. The time series dataset is normalised and transformed into multiple input/output patterns for recurrent neural network algorithms. Each data point has its current observation as the output combined with its three previous observations as the input. Finally, the training and validation data sets are created using a simple split data type which splits it with an 80/20 hold-out.
Besides ARIMA and TES, all forecasting models are multivariate as they are built using the systems resource workload as dependent variables and the independent variables specified by the features selections (Linear regression, Random Forest and XGBoost). Fourteen models are fitted to forecast any server's resource workload, two univariate and twelve multivariate combining the selected forecasting techniques and the three feature selections. The model fitting process also integrates hyper-parameter tuning to identify the model optimal parameters for model performance maximisation.

3.1.6 Evaluation

Model evaluation consists of using various performance metrics to understand a machine learning model's performance results and related strengths and weaknesses. The performance scoring metrics are computed using testing samples and the trained model's

predicted data. The model evaluation is initially processed during the model training to determine the best feature selection technique for each forecasting algorithm. Shcherbakov et al. (2013) wrote a comprehensive survey of metrics which can be employed for measuring errors level for time series forecasting and predictions. They stress that defined scoring measures are not pre-determined for any prediction scenario. Nevertheless, the authors still recommend using RMSE, MAE, and Mean absolute percentage error (MAPE).
RMSE is the standard deviation of predicted errors, showing how models accurately predict responses. A low RMSE value indicates the model's goodness of fit, allowing for comparison to others. However, it can be influenced by large values or outliers. For each actual observation x_i , predicted observation $\hat{x_i}$ and the number of time series data points n are expressed as follows:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2}$$

MAE is the sum of absolute differences between actual and predicted values. It eliminates direction, positive or negative and is robust to the effects of outliers. Lower value indicates a good model performance.
For each actual observation x_i , predicted observation $\hat{x_i}$ and the number of time series data points n are expressed as follows:

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x_i}|$$

MAPE is the mean of the absolute percentage errors of predictions. A value of MAPE less than 20% shows the goodness of fit of a model.
For each actual observation x_i , predicted observation $\hat{x_i}$ and the number of time series data points n are expressed as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \hat{x_i}}{x_i}\right|$$

The same evaluation steps are used to measure the performance of the unseen data predictions on fitted models.

3.1.7 Predictions

The predictions section involves evaluating the selected models with unseen data. Evaluation scores are the same as the ones used to measure model fitting and are completed for ten, twenty, thirty and forty days.

4 Design Specification

The implementation project is designed into three tiers of architecture: Data persistent, Business logic and Client, as shown in Figure 5. The data persistent tier is where the information is processed and stored. The business logic tier represents the application layer dedicated to data modelling where all models are fitted, and the most performant ones are selected. Ultimately, the client tier describes the user layer where predictions from the best candidate models are produced to determine the required system capacity. As a result, CompanyA can leverage prediction to determine their system capacity needs and define the optimal method from horizontal or vertical scaling.

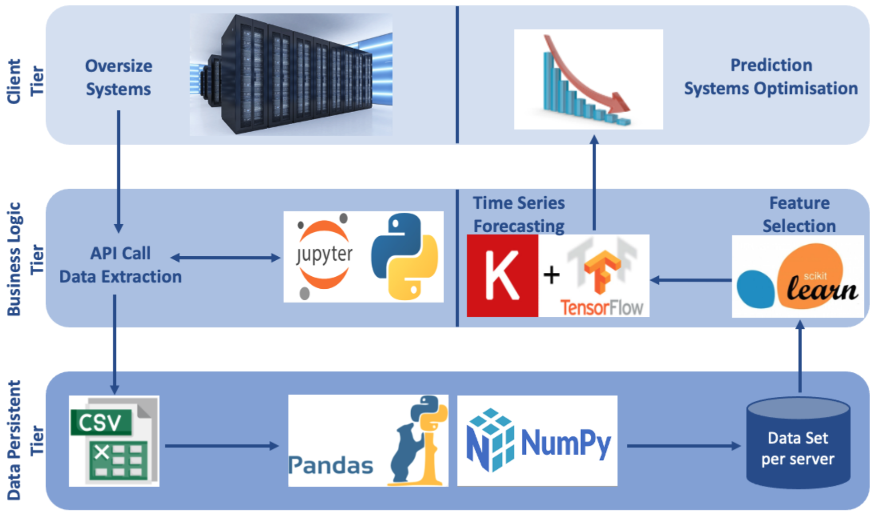

Figure 5: Three Tier Design Architecture

5 Implementation

5.1 Experimentation

The experimentation scenario uses genuine data from CompanyA, providing commercial software as a service to its customers. The solution is a web-designed application for single-tenant dedicated instances per customer. The application consists of multiple servers (or nodes) deployed in a micro-service environment, each containing several pods and multiple containers. CompanyA Systems infrastructure is provided by CSP but does not subscribe to additional infrastructure services. Instead, they manage their own Docker/Kubernetes platform on a provisioned CSP Bare-metal [6], physical and dedicated architecture. For cost and efficiency purposes, CompanyA is not currently planning to subscribe to Cloud virtual systems services and to leverage autoscaling services. CompanyA's intuition is that all their systems run over capacity with no planned appropriate optimisation process. The experimentation will leverage the best performant technique to predict a selected system's daily maximum resource requirement.

5.2 Dataset

The server using the largest amount of resources from the most active end-customer is selected as the experiment case scenario. The first implementation step was to collect data from Server1. The data is programmatically extracted using multiple Python API calls connecting to the monitoring system. Seven dedicated API calls were executed, six for each application attributes activity (API calls, total calls, transaction calls, integration calls, integration2 calls, service calls and database calls) and one for the system resource workload consumption (CPU and memory). The data pulled from each API call is held in a dedicated Python Pandas data frame and then stored in a CSV file on a laptop computer. The data is an aggregation of the selected end-customer all systems. Initial

[6]https://learn.microsoft.com/en-us/azure/baremetal-infrastructure/concepts-baremetal-infrastructure-overview

extracted data is from the 1st of December 2021 to the 31st of June 2022, hour to hour. Afterwards, data from the 1st of July 2022 to the 15th of August 2022 are extracted to compare with the final predictions.

5.3 Implementation Steps

The first step is using the Python Groupby function to convert each of the seven data frames from hourly to daily, as this experiment aims to select the best technique to predict the maximum daily necessary system resource. The Python merge function is used to merge all data frames. Subsequently, a Python function is designed and executed to split the data frame into servers distinctive data frames and save them as CSV files in an allocated directory. The pre-processing implementation tasks are completed by loading the experimentation selected server into a Python data frame.
The transformation phase is achieved by replacing any data frame missing values using KNN imputation function from Fancyimpute library. A Python condition is included in the function for replacing missing data points for data frame columns containing not more than 16% of their total. The optimal K number of clusters is set to five by plotting the K values and its cost function elbow of the curve. From this stage, the dataset is ready for model fitting. It consists of 212 data points with a date variable recording values for the system resource's maximum daily mean CPU consumption as the dependent variable and 26 application-associated independent variables.
Feature selection is first initiated by processing the Scikit-Learn[7] library normalisation function on the data frame. Specifically for Linear regression, a SciPy library box-cox transformation combined with a Pandas correlation function is executed to eliminate from Linear regression feature selection any highly skewed and or correlated features. Linear regression feature selection is completed by designing stepwise Linear regression, including a Python For loop condition leveraging the Statsmodels[8] library Ordinary Least Squares method. Random Forest and XGBoost feature selections are completed by respectively running Scikit-learn library RandomForestRegressor and XGBoost library XGBRegressor methods. The selected influential features from each technique are saved in distinctive lists which will be then used for fitting each model.
Thereafter, a function is designed for each learning incorporating every required step for Model Fitting. The function includes multiple other functions, each with a defined role and is run for each feature selection nominated set of variables and ultimately produces a plot comparing the forecasted and the test datasets and the evaluation scores.
For ARIMA learning, a Python Box-Cox method is first processed to transform all selected variables for SARIMAX and only the dependent variable for SARIMA to ensure normal distribution. Subsequently, two designed functions were successively launched, one to extract the time series seasonality value using from Python package Statsmodels seasonaldecompose function seasonaldecompose and one to determine using library pmdarima's ndiffs if time series required differencing. The ndiffs function uses Augmented Dickey-Fuller (ADF) , Kwiatkowski-Phillips-Schmidt-Shin (KPSS) and Phillips-Perron (PP) statistical tests to determine differencing requirements. The designed function returns, as a value, the best two test results over three.
For Neural Networks, the time series dataset is first normalised using the Scikit-Learn library scaler method. The variables are then transformed into multiple input/output

[7]https://scikit-learn.org
[8]https://www.statsmodels.org

patterns format using a designed function.
The training and testing data sets are constructed for all models by splitting the dataset's Python array into respective temporal orders of 70% and 30%. A hyper-parameter tuning function is designed and processed to identify the model's optimal parameters for model performance maximisation from a large settings selection. Akaike Information Criterion (AIC) and RMSE are the hyper-parameter tuning function scoring methods respectively used for ARIMA and Neural network techniques. Additionally, the Keras[9] EarlyStopping method is incorporated in the neural network hyper-parameter function to stop training models and record the epoch numbers before overfit occurs. A hyper-parameter tuning function is combined with the Python multi-threading method to boost computation time. The optimal model selection also includes the least impact from over or underfitting. The best models are fitted for neural networks using Tensorflow[10] Keras compile and fit methods along mean absolute error as the loss function, and for SARIMAX, the Statsmodels sarimax method. The model training function is completed by plotting the forecasted against the test datasets along with the Scikit-Learn evaluation scores. Specifically for SARIMA and SARIMAX plotdiagnostics method is processed to plot diagnostic checks for model validation.
Predictions for ten, twenty, thirty and forty days ahead are compiled for each forecasting technique's most performant model/feature selection. As previously completed for the model fitting, each prediction is plotted and evaluated using prediction against unseen data.

6 Results

This section presents the results described in the experimentation section, starting with each algorithm's most performant model fitting and feature selection, followed by the multi-step ahead predictions and a discussion highlighting the significance of the outcomes.

6.1 Model Selection

Table 1 shows the results of the model fitting of forecasting techniques and each feature selection except for SARIMA and TES. No feature selection applies to SARIMA and TES as algorithms involve univariate datasets.
TES is the most performant univariate forecasting technique with an RMSE score of 0.6954 and MAPE of 0.5059. However, TES produces 27% more error than the least performant selected multivariate time series forecasting technique.
Table 1 also displays a broad range of results for the fitted model using time series forecasting techniques and feature selection algorithms influential predictors. Although XGBoost MAPE score is for SARIMAX 1.5% lower than Random Forest, Random Forest applied to SARIMAX as a feature selection as has RMSE and MAE errors scores respectively lower by 0.07 and 0.021 compared to XGboost. Additionally, examining the plots comparing the forecasted and testing dataset for each technique show Random Forest capturing better the underlying data. For neural networks, XGBoost is the most performing feature selection learning with an MAE score larger than the following performing learning by approximately 15% for LSTM, 13% for GRU and 17% for RNN and an RMSE score

[9]https://keras.io
[10]https://www.tensorflow.org

larger by approximately 20% for LSTM, 13% for GRU and 18% for RNN. In contrast, the Linear regression feature selection algorithm performed the poorest for almost all forecasting techniques except for RNN outperforming Random Forest by 1%.

Forecasting Technique	**Feature Selection**	**MAE**	**RMSE**	**MAPE**
TES	N/A	0.4942	0.6385	0.4483
SARIMA	N/A	0.5491	0.6954	0.5059
SARIMAX	LinearRegression	0.2656	0.4134	0.2154
	RandomForest	**0.1513**	**0.1808**	**0.1703**
	XGBoost	0.1727	0.2544	0.1558
LSTM	LinearRegression	0.2632	0.4002	0.3331
	RandomForest	0.2365	0.3392	0.4078
	XGBoost	**0.0881**	**0.1333**	**0.1457**
GRU	LinearRegression	0.2678	0.4093	0.2678
	RandomForest	0.2388	0.2819	0.4869
	XGBoost	**0.1023**	**0.1522**	**0.1719**
RNN	LinearRegression	0.2570	0.3938	0.3219
	RandomForest	0.2618	0.3003	0.5576
	XGBoost	**0.0870**	**0.1244**	**0.1409**

Table 1: Model selection based on feature selection

6.2 Multi-step Ahead Prediction

Figure 6 provides results of the multi-step ahead predictions for each of the six forecasting technique. Predictions are made for ten, twenty, thirty and forty days ahead, and error levels are measured using RMSE, MAE and MAPE against unseen data. TES and SARIMA RMSE scores increased by approximately 100% from ten to forty days predictions and MAPE scores by approximately 50% to reach 0.4513 and 0.4712.
The first observation is that Neural Networks RMSE scores are larger than SARIMAX for short-term prediction within ten days. The RNN, LSTM and GRU RMSE predictions score trends are almost identical, and score gaps are marginal. However, there is a shift from twenty days predictions where Neural networks RMSE performance score increase from 0.0822 to 0.2210 to exceed SARIMAX, which remains approximately constant. Neural networks RMSE score sharply increases from the first to the second step prediction but stabilises to reach approximately 8% from one step period prediction to another. On the contrary, SARIMAX RMSE score remains almost constant for all step period predictions except for the increase from 0.1759 to 0.2017 from twenty to thirty days.
Multi-step ahead prediction models' MAE scores show the same pattern as RMSE but with a performance shift from twenty days instead of ten days. This distinction is explained by the fact that RMSE is more sensitive to outliers penalising more significant errors than MAE. The MAE is a constant score, meaning each residual is weighted equally in the average.

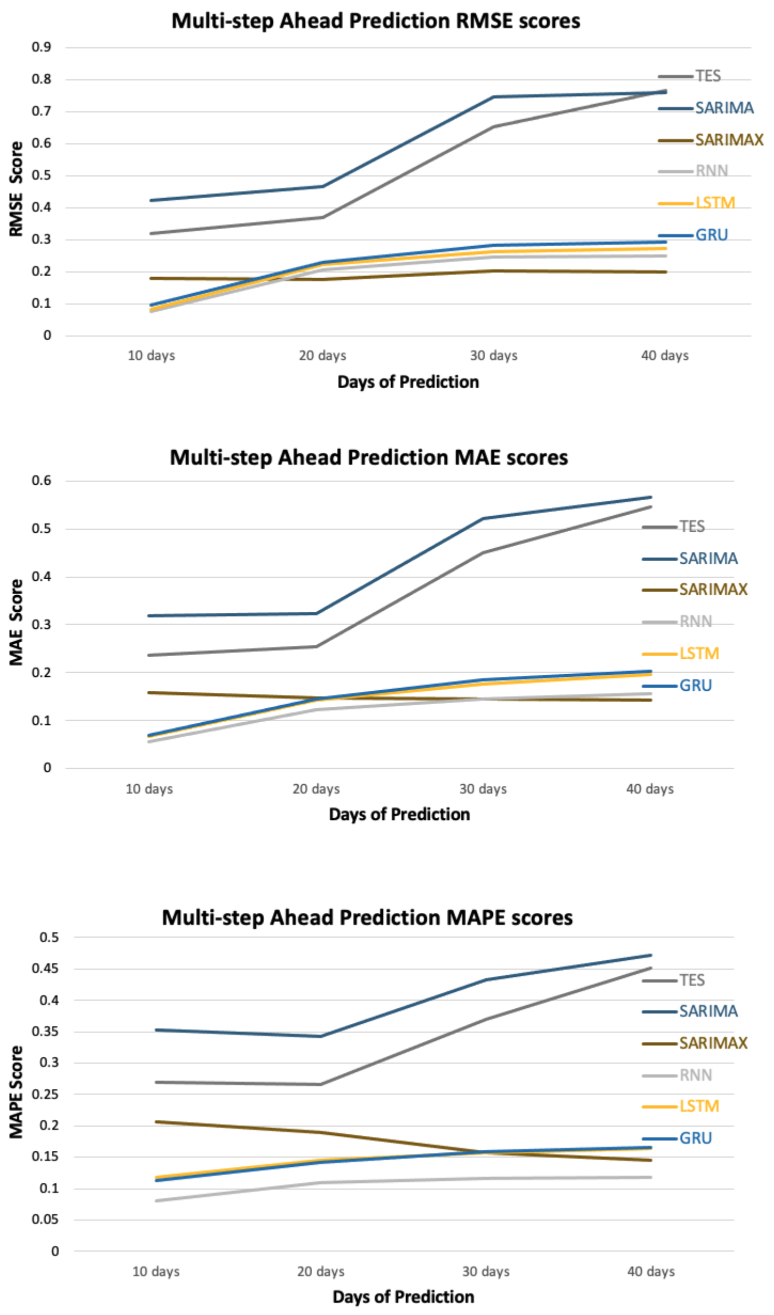

Figure 6: Prediction metrics

7 Discussion

7.1 Model Selection

The first observation of this experiment illustrated how poorly TES and SARIMA univariate performed fitting models. The univariate and the multivariate time series forecasting methodologies developed a significant error gap which can be explained by the univariate methods not having associated predictors

The second observation is the impact of associating features to the time series dependent variable and the technique for selecting the most influential ones. Feature selection is key to determining which variables influence each forecasting technique the most. The

second experiment tested feature selection algorithms and determined the most performant technique for each model. The experiment results show that boosting (XGBoost) and bagging (Random Forest) feature selections for the selected dataset perform better than Linear regression across all forecasting techniques. The performance scores show that Linear regression was insignificant, while XGBoost was proven more effective across all techniques, even if, for SARIMAX, Random Forest negligibly outperformed XGBoost. Whereas for Neural networks, XGBoost significantly surpassed the other two techniques with its low level of errors.
XGboost superior performance can be explained by the presence of a boosting feature selection algorithm to select features by minimizing errors, as displayed by its RMSE and MAE scores. On the contrary, Linear regression feature selection is the learning with the lowest performance, which is explained by the larger residuals resulting from the linear relationship between the dependent and independent variables.
The model fitting experiment confirmed how essential feature selection techniques are to forecast future values and how influential it is to select the highest level of precision from a diverse set of techniques.

7.2 Multi-step Ahead Prediction

One of the experiment objectives was to determine how influential independent variables were in predicting time series values. The first observation from the multi-step ahead prediction experiment is the prediction precision score disparity between univariate and multivariate forecasting techniques. Comparatively, the RMSE and MAE scores for TES and SARIMA for ten days of prediction are approximately 25% to 30% higher than multivariate methods and are proportionally increasing for the following prediction periods
The two univariate techniques have the same prediction trend, although TES outperformed SARIMA for each period except for forty days showing that long-term prediction between the two techniques flattened. Introducing independent variables for predicting Cloud resource workload is an absolute requirement, as using the dependent variable's past value to explain its future value is insufficient. However, it requires more than just adding variables to a dataset. Adding a feature selection mechanism into the equation is essential to define which added variables are significant and influential to the dependent variable and its prediction.
The observation from the multi-step ahead predictions best models revealed the same region of performance for the multivariate techniques. Looking closer at results from RMSE outlined better short-term predictions from neural network models, while SARIMAX offers more precision for long-term predictions. MAE show the same behaviour but with the performance shift happening between twenty and thirty days of predictions instead of ten days for RMSE.
The interpretation of these findings is that MAE evaluates the absolute distance of the observations to the prediction and is less influenced by outliers to the same degree as RMSE. For predicting future values, MAE is differentiated as more robust than RMSE to measure each model goodness of fit. Overall, the prediction scores suggest SARIMAX is more stable than Neural network models, which have less variance over time and better generalise the data. In contrast, MAPE scores showed RNN having a better average percentage difference between the predictions and the actual data sample, even for long-term predictions.
The neural network prediction error variance can be explained by the number of data

points used for that experiment, as a large dataset is recommended to take advantage of neural networks. Insufficient data points can also explain RNN being the most performant neural network while suffering from a vanishing gradient and loss of efficiency with long time series sequences. A vanishing gradient is a problem that RNN face during model training. The gradient becomes exponentially small, causing the back-propagation process to struggle to optimise the network weights. LSTM and GRU were developed to overcome RNN constitutional gaps to improve performance and prediction accuracy, which is contradicted in the prediction experiment as RNN accuracy scores are negligibly better than LSTM and GRU. Multiple factors can explain these results, starting with the shortness of the dataset, the requirement for an extensive hyper-parameter cycle, or the enablement of LSTM and GRU stateful functions to better control each network memory cell.
Nonetheless, accuracy and precision maximisation can be achieved by selecting the model and technique providing the best result for each prediction step to produce the most optimal outcomes. Ultimately, the prediction experiment did not elect a method systematically outperforming others but provided a framework leveraging the best technique for each step ahead prediction.

8 Conclusion and Future Work

This research aimed to determine the most optimal forecasting technique based on a selection of commonly known methods. One of the primary outcomes is the design of a flexible model framework combining multiple time series forecasting and feature selection algorithms to determine the most optimal model to predict Cloud workload requirements. Random Forest and XGBoost were the most effective feature selection techniques for SARIMAX and Neural Network, respectively.
The outcomes from the experiment revealed that neural network models and SARIMAX provided the most accurate results for short-term and long-term prediction. However, evidence from the multi-step ahead predictions plot (Figure 6) showed SARIMAX as the model generalising the data best as showing steady residual variability over time. Furthermore, a hybrid model combining RNN for short-term predictions and SARIMAX for long-term would undoubtfully provide the most accurate prediction.
The value of this research resides not only in electing the best technique but also in selecting the finest methods for optimal prediction, as data set composition and nature vary constantly. This research demonstrated that feature selection, coupled with forecasting techniques, was a robust method to identify the best model for optimal prediction.
Future work will focus on adding feature selection algorithms such as neural network auto-encoder or Principal Component Analysis (PCA) to introduce more variation and strengthen the process to build the best forecasting model. Additionally, implementing a stacking ensemble machine learning algorithm combining the predictions from the most performant models could enhance prediction accuracy.

References

Alsahaf, A., Petkov, N., Shenoy, V. and Azzopardi, G. (2022). A framework for feature selection through boosting, *Expert Systems with Applications* **187**.

Calheiros, R. N., Masoumi, E., Ranjan, R. and Buyya, R. (2015). Workload prediction using arima model and its impact on cloud applications, *IEEE Transactions on Cloud Computing* **3**: 449–458.

Chapram, S., Revanth, K., Nupa, S. and Vishal, R. (2018). Workload prediction using arima statistical model and long short-term memory recurrent neural networks, *IEEE International Conference on Computing, Power and Communication Technologies* .

Chen, R. C., Dewi, C., Huang, S. W. and Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods, *Journal of Big Data* **7**.

Chen, Z., Hu, J., Min, G., Zomaya, A. Y. and El-Ghazawi, T. (2020). Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning, *IEEE Transactions on Parallel and Distributed Systems* **31**: 923–934.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Duggan, M., Mason, K., Duggan, J., Howley, E. and Barrett, E. (2017). Predicting host cpu utilization in cloud computing using recurrent neural networks, *The 12th International Conference for Internet Technology and Secured Transactions* .

Guyon, I. and De, A. M. (2003). An introduction to variable and feature selection, *Journal of Machine Learning Research* **3**: 1157–1182.

Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q. M., Tziritas, N., Vishnu, A., Khan, S. U. and Zomaya, A. (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, *Computing* **98**: 751–774.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Computation* **9**: 1735–1780.

Ilyushkin, A., Ali-Eldin, A., Herbst, N., Papadopoulos, A. V., Ghit, B., Epema, D. and Iosup, A. (2017). An experimental performance evaluation of autoscaling policies for complex workflows, *Proceedings of the 2017 ACM/SPEC International Conference on Performance Engineering* pp. 75–86.

Jadhav, A., Pramod, D. and Ramanathan, K. (2019). Comparison of performance of data imputation methods for numeric dataset, *Applied Artificial Intelligence* **33**: 913–933.

Janardhanan, D. and Barrett, E. (2017). Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models, *The 12th International Conference for Internet Technology and Secured Transactions* .

Jenkins, G., Reinsel, G., Ljung, G. and Box, G. (1970). *Time series analysis: forecasting and control*, 5th ed. edn, Wiley.

Kumar, K., Rao, K. G., Bulla, S. and Venkateswarulu, D. (2021). Forecasting of cloud computing services workload using machine learning, *Turkish Journal of Computer and Mathematics Education* **12**: 4841–4846.

Li, X., Wang, H., Xiu, P., Zhou, X. and Meng, F. (2022). Resource usage prediction based on bilstm-gru combination model, *2022 IEEE International Conference on Joint Cloud Computing (JCC)* pp. 9–16.

Lisdorf, A. (2021). *Cloud Computing Basics: A Non-Technical Introduction.*, 1st ed. edn, Apress.

Makwana, P., Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining of cluster in k-means clustering review on determining number of cluster in k-means clustering, *International Journal of Advance Research in Computer Science and Management Studies* **1**.

Mehmood, T., Latif, S. and Malik, S. (2018). Prediction of cloud computing resource utilization, *IEEE 15th International Conference on Smart Cities: Improving Quality of Life Using ICT and IoT* .

Moreno-Vozmediano, R., Montero, R. S., Huedo, E. and Llorente, I. M. (2019). Efficient resource provisioning for elastic cloud services based on machine learning techniques, *Journal of Cloud Computing* **8**.

Ouhame, S. and Hadi, Y. (2019). Multivariate workload prediction using vector autoregressive and stacked lstm models, *ACM International Conference Proceeding Series* .

Podolskiy, V., Jindal, A. and Gerndt, M. (2019). Multilayered autoscaling performance evaluation: Can virtual machines and containers co-scale?, *International Journal of Applied Mathematics and Computer Science* **29**: 227–244.

Poppe, O., Guo, Q., Lang, W., Arora, P., Oslake, M., Xu, S. and Kalhan, A. (2022). Moneyball: proactive auto-scaling in microsoft azure sql database serverless, *Proceedings of the VLDB Endowment* **240**: 7–10.

Shaw, R., Duggan, M., Duggan, J., Howley, E. and Barrett, E. (2018). A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers.

Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A. and evich Kamaev, V. A. (2013). A survey of forecast error measures, *World Applied Sciences Journal* **24**: 171–176.

Ullah, A., Ouhame, S. and Hadi, Y. (2021). An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model, *Neural Computing and Applications* **33**: 10043–10055.

Xie, Y., Jin, M., Zou, Z., Xu, G., Feng, D., Liu, W. and Long, D. (2022). Real-time prediction of docker container resource load based on a hybrid model of arima and triple exponential smoothing, *IEEE Transactions on Cloud Computing* **10**: 1386–1401.

Yadav, M. P., Rohit and Yadav, D. K. (2022). Resource provisioning through machine learning in cloud services, *Arabian Journal for Science and Engineering* **47**: 1483–1505.

Zhang, W., Li, B., Zhao, D., Gong, F. and Lu, Q. (2018). Workload prediction for cloud cluster using a recurrent neural network, *Proceedings - 2016 International Conference on Identification, Information and Knowledge in the Internet of Things* pp. 104–109.