DOI: 10.21917/ijsc.2024.0469

A STACKED GENERALIZATION BASED META-CLASSIFIER FOR PREDICTION OF CLOUD WORKLOAD

Sanjay T. Singh and Mahendra Tiwari
Department of Electronics and Communication, University of Allahabad, India

Abstract

Cloud computing has revolutionized the way software, platforms, and infrastructure can be acquired by making them available as on-demand services that can be accessed from anywhere via a web browser. Due to its ubiquitous nature Cloud data centers continuously experience fluctuating workloads which demands for dynamic resource provisioning. These workloads are either placed on Virtual Machines (VMs) or containers which abstract the underlying physical resources deployed at the data center. A proactive or reactive method can be used to allot required resources to the workload. Reactive approaches tend to be inefficient as it takes a significant amount of time to configure the resources to meet the change in demands. A proactive approach for resource management is better in meeting workload demands as it makes an appropriate number of resources available in advance to cater to the fluctuations in workload. The success of such an approach relies on the ability of the resource management module of a data center to accurately predict future workloads. Machine Learning (ML) has already proven itself to be very effective in performing prediction in various domains. In this work, we propose an ML meta-classifier based on stacked generalization for predicting future workloads utilising the past workload trends which are recorded as event logs at Cloud data centers. The proposed model showed a prediction accuracy of 98.5% indicating its applicability for the Cloud environment where SLA requirements must be closely adhered to.

Keywords:

Cloud Computing, Auto-Scaling, Workload Prediction, Stacked Generalization, Machine Learning

1. INTRODUCTION

Due to its inherent characteristics, Cloud has emerged as a computing platform of choice for businesses both large and small [1]. Cloud workload corresponds to the various inputs/ requests by users for various services/applications offered by the Cloud service providers which are hosted at a Cloud data center. Each of these workloads has its unique resource demands and performance requirements. These workloads are required to be executed to meet the Service Level Agreement (SLA) parameters. SLA parameters are the constraints that ensure an acceptable level of performance so that the response time, which is one of the Quality of Service (QoS) attributes, is within the specified range. Due to the extremely dynamic nature of workloads, Cloud performance is impacted [2].

This means that at certain times of day, a large number of users are attempting to utilise the service simultaneously, yet at other times just a small number of users are parallelly accessing the Cloud services. As a result, static resource allocation to workloads is ineffective because, during periods of low demand, there will be more resources available than required, costing the service provider and during periods of high demand, there may not be enough resources available, resulting in low QoS and a loss of customers [3].

Hence, resources must be provisioned dynamically to match the demands of fluctuating workloads in order to prevent their overallocation or under allocation [4].

To provide resources dynamically, deciding on the correct number of resources to be provisioned at a particular time interval is a necessary and non-trivial task [3]. The approaches for handling this task can be categorized as reactive or proactive. The reactive approach works by adding/ removing resources when certain thresholds are reached. This approach is slow and results in SLA violation as VMs or containers on which workloads are deployed require some fraction of time to be up and running. On the contrary proactive approaches towards resource provisioning are faster as they work by estimating the required number of resources in advance and keep resources ready if they are to be deployed in the future or if the resources are to be removed, it does so in a timely manner [5].

This eventually ensures that SLA parameters are upheld and also the electricity consumption of the data center is optimised. So, a proactive auto-scaler for dynamically provisioning resources as per the variation in workload trend must be equipped with the ability to forecast the workload by observing the historical data for workload trends. For the purpose of prediction of workload various parameters corresponding to workloads such as job arrival rate, CPU statistics, disk transfer rates, memory capacity details, etc. can be utilised. These details are usually monitored and logged in a data center [6]. ML has proven to be a very effective tool for performing predictive tasks in various domains as ML models learn from the relationships between the attributes of data given to them as input and based on this information, they predict the outcome. Researchers have successfully used ML to accomplish the task of workload prediction in the past [5].

The goal of this study is to utilise the predictive capabilities of ML to handle the challenge of correctly predicting the Cloud workload trend, which will ultimately aid in the optimisation of auto-scaling choices. To this end, we propose a meta-classifier model that combines multiple ML models by stacking them and then using a baseline classifier to make a final prediction [7]. In this approach, each of the ML models in the model stack performs their predictions and then the prediction probabilities from all the models are used as input to the meta-classifier. In order to demonstrate the proof of the efficacy of the proposed model we have compared it with some existing works.

The primary contributions of our work are given below:

- Proposing a meta-classifier for Cloud workload prediction.
- Novel implementation of the proposed model over the Kingdom of Saudi Arabia (KSA) Ministry of Finance dataset [8].
- Evaluating the performance of the proposed model by comparing it with some earlier works.

2. RELATED WORK

Dynamic provisioning of resources cannot be realized to the fullest without the ability to predict Cloud workloads efficiently and accurately. Predicting workloads is essential in order to optimize resource utilisation, ensuring QoS and reducing costs which is advantageous for both Cloud service providers and customers. In this section, we discuss some of the works done in this domain by various researchers in the past.

Wamba et al. [9] proposed two complementary ML-based workload prediction models based on Neural Networks and Constraint Programming. To forecast workload in the Cloud, Kumar et al. [10] suggested a hybrid model combining LSTM and BLSTM that uses association learning to capture the link between the associated resource indicators. They used Google cluster trace data and the data obtained in a virtual environment based on Docker to demonstrate the performance of the suggested technique. Liu et al. [11] suggested an ensemble learning approach which they call Tr-Predictor. Their approach used LSTM as well as sample weight transfer for workload prediction. To test the predictive power of the suggested approach, they randomly choose different sets of short-workload data from the Google and Alibaba clusters respectively. Maiyza et al. [12] proposed a hybrid VTGAN workload prediction model. Their model forecasts both the trend in workload (high or low) as well as future workloads. VTGAN models are very effective in handling long-term nonlinear interdependence of Cloud workloads. In their hybrid prediction model, LSRU, Shuvo et al. [13] integrated the GRU and LSTM models. They demonstrated that the LSRU outperforms the LSTM or the GRU model in terms of accuracy if they are to be used as standalone models. A Linear Regression based workload prediction model, called LiRCUP was proposed by Farahnakian et al. [14] to reduce violations of the SLA parameters and control the energy costs. Their method performs an approximation function by using the most recent resource usage trend i.e., of almost an hour ago. Based on the present utilisation in each host machine, the approximation function predicts the short-term future utilisation. To forecast how a VM will be deployed in the Azure Cloud platform, Cortez et al. [15] employed gradient boosting trees with random forest models. To provide sufficient time for scheduling of tasks based on the projected workload, Gao et al. [16] proposed the m-gap prediction approach to do workload prediction ahead of time. For better accuracy of forecast, they also suggested a workload prediction approach based on clustering. In order to forecast the future workload of individual requests, their technique first groups jobs with identical workload trends into clusters, after which it generates a workload forecast model for respective clusters, and then employs the associated model. Yu et al. [17] proposed a workload prediction that uses the statistical characteristics of a pool of existing workload to forecast the workload patterns of new jobs. Their method combines both machine learning and clustering to maximise learning efficiency and, as a result, enhances the accuracy of workload prediction. Through experiments they proved the effectiveness of their model in providing better utilisation of resources and reduction of energy consumption. Yazdanian et al. [18] processed a lengthy sequence of Google trace data using a combination of a stack of LSTM blocks and 1D Convolutional Networks to provide a precise and efficient forecast of RAM and CPU demands in the future.

3. PROPOSED MODEL

The purpose of this section is to describe the proposed model, shown in Fig.1, which works on a stack of five ML models namely: K-Nearest Neighbor, Decision Tree, Gradient Boosting, Histogram Gradient Boosting, and Artificial Neural Network. The prediction probabilities of these models are input to the Logistic Regression model that works as a meta-classifier for making the final prediction. The proposed model can learn from past data related to workload arrival trends and their related resource utilisation. After the model is trained, it forecasts the future pattern of workload arrival and the related trend of resource utilisation. Given below are the details of the steps/ components of the proposed model:

3.1 DATA PREPROCESSING

Preprocessing of data is very crucial as this step prepares the data for training the ML model [19]. It encompasses several tasks out of which we have performed two for making the data suitable for learning of our proposed model to take place.

3.1.1 Feature Engineering:

Feature engineering is performed for deriving new features from the various features present in the dataset. For this, we have combined three features pertaining to the number of jobs arriving at time intervals of 1, 5, and 15 minutes respectively to form a new feature for representing the average number of jobs which makes it convenient to analyse the resource utilisation trends.

3.1.2 Resolving Data Imbalance:

Data imbalance is a state in which the dataset set does not have equal number of instances for all the labels of the output variable. Due to this issue the model tends to learn more in favour of those class labels whose instances are more and learns less for those class labels whose instances are less. This eventually affects the performance of the classifier as it may wrongly predict labels when faced with new data.

The KSA dataset for this work has four class labels, namely i) High, ii) Medium, iii) Low, and iv) Very Low. The number of instances for each of these class labels present in the dataset is shown in Table.1. It can be clearly seen that the original dataset suffered from data imbalance as the number of instances for Medium and Low classes were substantially less.

We used random oversampling for handling the class imbalance problem. This technique works by either creating synthetic data points or randomly replicating existing instances, thereby, increasing the number of cases in the minority class. After performing the random oversampling, the number of instances for all four class labels became 13028 and the number of instances in the dataset were increased to 52112 from 25697.

Table.1. Class labels with the number of instances

S. No.	Label Name	No. of Instances
1.	High	13028
2.	Medium	254
3.	Low	2715
4.	Very Low	9700

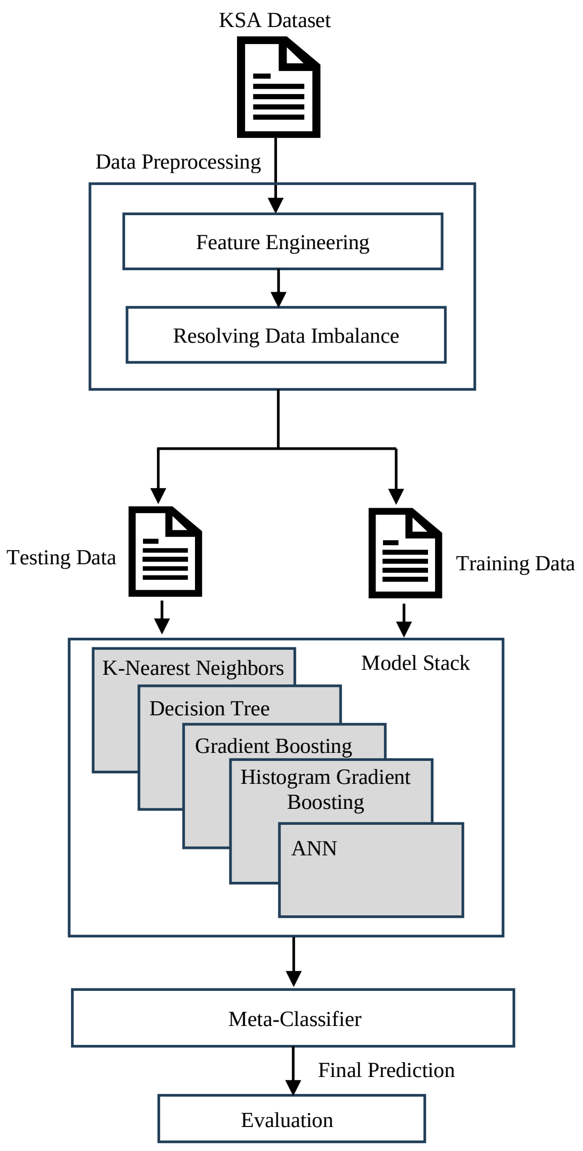

Fig.1. Architecture of the Proposed Model

3.2 DATA SPLIT

After preprocessing the data, we split the dataset into a training set and a testing set. The data in the training set is used to train the model to learn the relationships existing between the attributes in the data. Once the model training is over, the data in the testing set is used to test the predictive performance of the Model. For our work, we used 80% data of the dataset for training and 20% of the data for testing the model.

3.3 MODEL STACK

The core of our proposed model is a model stack. Stacking the various ML models is an ensemble learning approach. In this technique, several base models are combined and they are trained on the same dataset and make their individual predictions. In order to improve the accuracy of the forecast, predictions made by the base models are combined and input as features to the meta-classifier to make the final prediction. Below is the description of the five ML models we used for the model stack.

3.3.1 K-Nearest Neighbor:

K-Nearest Neighbor (KNN) [20] is a straightforward but effective machine learning method utilised for both classification as well as regression problems. It memorises the training data and bases predictions on similarities between new and old data points rather than creating an explicit model. In the training phase, the classifier stores all the data instances along with the labels of the output variable. In the prediction phase, when given a new data sample, the KNN classifier works by calculating its Euclidean distance from all the data samples learned in the training phase. Next, the classifier chooses the top K nearest neighbours with the smallest distances. From the identified nearest neighbours, the occurrences of class labels of each neighbour are counted and then the new sample is classified as belonging to that neighbour class that has the highest number of occurrences.

3.3.2 Decision Tree:

Decision trees [21] are a series of models that rationally integrate a sequence of tests; each test compares a nominal or numeric feature against a range of possible values. In the training phase, this method partitions the input data recursively according to the most useful characteristics and their accompanying thresholds, creating a hierarchical tree-like structure. Each leaf node of the tree carries the predicted class label, while each internal node indicates a judgement based on a characteristic. The tree is improved during its construction to better detect relationships hidden in the training data. A new sample moves through the tree during the prediction phase, following the decision pathways determined by the feature values. The algorithm eventually arrives at a leaf node to obtain the predicted result.

3.3.3 Gradient Boosting:

Gradient boosting [22] is an efficient ensemble learning algorithm for classification as well as regression tasks. It works by gradually constructing an ensemble of classifiers which are generally decision trees. Following the creation of an initial model, known as the base model, further models are created to remove the errors in the ensemble of previous models. The differences between the established target values and the most recent predictions are known as residuals or mistakes. Each new model carefully integrates with the current ensemble to forecast these residuals. The accuracy of the ensemble increases as errors are reduced through an iterative approach. Overall, this approach creates a robust classifier by continuously improving performance and identifying complex data patterns.

3.3.4 Histogram Gradient Boosting:

In order to increase effectiveness and scalability in complex machine learning problems, histogram gradient boosting serves as an advanced version of the conventional gradient boosting approach [22]. Histogram Gradient Boosting uses a binning algorithm to divide data points as discrete intervals instead of the traditional method that takes into consideration precise feature values, resulting in histograms that show the distribution of features. Based on these histograms, the method determines the best-split points during tree construction, therefore significantly lowering computing complexity. This approach speeds up training by using histograms for decision-making since they can be constructed much more quickly than exact split computations. On multi-core architectures, this method also improves memory efficiency and makes parallelization easy. Because of this, Histogram Gradient Boosting provides an efficient way to handle large datasets while retaining reliable forecasting performance.

3.3.5 Artificial Neural Network:

Artificial neural network (ANN) [23] is a computing model motivated by the complex interconnections found in the biological brain. It has an input layer for data intake, one or more than one hidden layer that gradually extracts characteristics and associations, and an output layer that generates predictions. It functions by replicating the behaviour of neurons at hidden layers. Weighted connections are the primary means of communication among neurons. These weights denote the strength of the communication link. Inputs are transferred through the network during forward propagation, and nonlinearity is introduced when neurons apply a function called activation function to their weighted sum of inputs. Afterward, a loss function is used to compare the computed outputs to the actual targets, measuring the error in prediction. A key component of ANN is backpropagation in which iterative calculation of the gradients of the loss is done corresponding to the weights and biases of the network. The optimisation process is guided by these gradients. Consequently, weights and biases are modified using optimisation techniques such as stochastic gradient descent. Internal parameters of the network are improved through its iterative learning process, allowing it to produce predictions that become increasingly accurate over time. A trained ANN demonstrates its adaptability and power in a variety of machine learning tasks by generalising its learned knowledge to new, unseen input.

3.4 META-CLASSIFIER

The predictions made by the model stack were further fed as input to the meta-classifier. A meta-classifier is an ML model based on ensemble learning that combines predictions from many base classifiers to provide well-informed final predictions. It utilises the outputs of the base classifiers while operating at a higher level to improve the accuracy of prediction. This approach follows a series of steps; first using the training set several base classifiers are independently trained to make their individual predictions. These predictions can be in the form of either class labels or probabilities. The meta-classifier then uses these predictions as input features. Finally, the meta-classifier combines predictions made by base classifiers and performs concluding predictions. Once the training phase is over, the base classifiers perform their predictions in response to fresh input which the meta-classifier combines to produce the final prediction by the ensemble. By using the unique abilities of the underlying classifiers, the use of a meta-classifier provides advantages such as a variety of insights, resilience against noise, and a potential increase in performance.

3.5 EVALUATION

Evaluating the performance of any ML model is very important as it gives an insight into its performance. There are various metrics for evaluating the performance of ML models. Some of these metrics are common to both classification and regression tasks whereas some metrics are suitable only for classification problems and some are exclusive to regression problems. For evaluating the performance of the proposed classification model, we have used the metrics shown in Table.2.

Table.2. Metrics for evaluation

S. No.	Metrics
1.	Accuracy
2.	Precision
3.	Recall
4.	F1-Score
5.	Balanced Accuracy Score
6.	Macro Avg. Precision
7.	Weighted Avg Precision
8.	Macro Avg Recall
9.	Weighted Avg Recall
10.	Macro Avg F1-Score
11.	Weighted Avg F1-Score

4. EXPERIMENTAL SETUP

4.1 DATASET DETAILS

For the proposed work we have used the KSA dataset as provided by [8]. The Table.3 contains the description of the dataset. This dataset contains 28147 instances which were collected from 13 computing nodes. These instances were further divided into two parts, one consisting of 25697 and the other consisting of 2450 instances. For our experiment, we used the file having 25697 instances. The Table.4 shows the features present in this dataset.

Table.3. Dataset details

Dataset Source	Ministry of Finance, Riyadh, Saudi Arabia	**Dataset Accessibility**	[8]
Characteristics of Dataset	Multivariate	**Missing Values**	No
No. of Attributes	9	**Number of Records**	28147
Attribute Values	Real	**No. of Nodes used for Data Collection**	13

Table.4. Dataset features

S. No.	Feature Name
1.	Jobs_per_ 1Minute
2.	Jobs_per_ 5 Minutes
3.	Jobs_per_ 15Minutes
4.	Mem capacity
5.	Disk_capacity_GB
6.	Num_of_CPU_Cores
7.	CPU_speed_per_Core
8.	Avg_Recieve_Kbps
9.	Avg_Transmit_Kbps

As mentioned earlier, this dataset was suffering from a class imbalance problem which could introduce model, evaluation, and decision-making biases we dealt with it by applying random oversampling after which the number of instances became 52112. We used 41690 instances for training the model and the remaining 10422 instances for testing the predictive performance of the proposed model.

4.2 TOOLS USED

For the implementation and evaluation of our proposed model, we used Google Colaboratory which is a Cloud-based environment for Python programming and can be accessed through a browser. For execution we used a laptop having Intel Core i3-1115G4 processor clocked at 3.00GHz and 8 GBs of RAM. The operating system of the laptop was Windows 10 64-bit.

5. RESULTS AND DISCUSSION

In this section, we present the results that we achieved corresponding to the various metrics chosen for the evaluation of the predictive performance of the proposed model. To demonstrate the predictive efficiency of the proposed model we also compare it with the works proposed by [24]–[26].

A confusion matrix is required to be plotted in order to calculate the values for various evaluation metrics. It represents the performance of the classifier by projecting predicted class labels against actual class labels. Following are the terms associated with a confusion matrix:

- True Positive (TP), is the number of instances belonging to a class that are correctly classified to be of that class.
- True Negative (TN), is the number of instances not belonging to a class that are correctly classified as not belonging to that class.
- False Positive (FP), is the number of instances not belonging to a class but wrongly classified as belonging to that class.
- False Negative (FN), is the number of instances belonging to a class that are misclassified as not belonging to that class.

Fig.2 represents the confusion matrix for the proposed classifier. Looking at the confusion matrix it can be observed that the proposed classifier correctly classifies 2539 instances out of 2568 for the class label Very Low, 2569 out of 2640 for the class label Low, 2589 out of 2625 for the class label Medium, and 2568 out of 2590 for class label High. The overall accuracy of the proposed classifier is 0.985.

We computed the precision score of the proposed model. Precision is the percentage of instances that were accurately predicted as positive out of all instances that were projected to be positive. Precision can be defined as:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{1}$$

It can be seen from Table.5 that the meta-classifier gives a balanced performance for all class labels in comparison to the existing works.

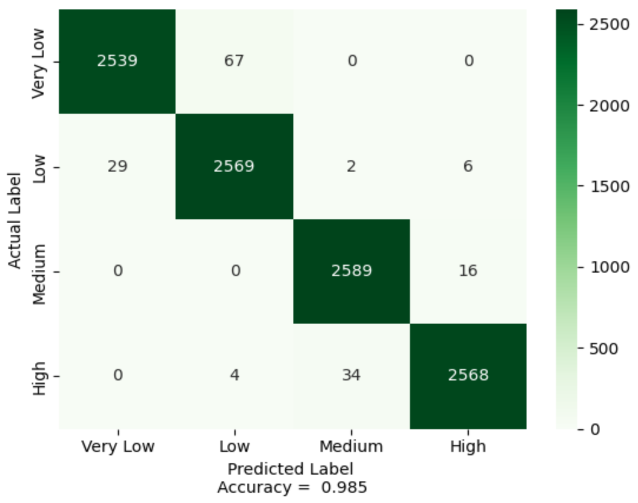

Fig.2. Confusion matrix for meta-classifier

We also computed the recall value for our proposed model and compared it with the recall values of existing works as illustrated in Table.6. Recall is the percentage of accurately forecasted positive instances out of all actual positive instances. Recall can be computed by the formula given below:

$$\text{Precision} = \frac{TP}{TP+FN} \tag{2}$$

The meta-classifier performs better considering the fact that we have implemented the proposed classifier after dealing with the data imbalance problem which was not addressed by the previous works.

Further, a comparison of f1-scores of previous works with the proposed meta-classifier is shown in Table.7. F1-score is the harmonic mean of precision and recall. The formula for calculating F1-score is:

$$\text{F1 Score} = \frac{TP}{TP+0.5(FP+FN)} \tag{3}$$

Table.5. Computation of Precision Values

Class Label	Gaussian Naïve Bayes [24]	Random Forest [25]	Support Vector Machine [26]	Proposed Model
High	1	1	1	0.99
Medium	0.14	0.55	0	0.99
Low	0.08	0.94	0.56	0.97
Very Low	0.92	0.99	0.89	0.99

Table.6. Computation of Recall

Class Label	Gaussian Naïve Bayes [24]	Random Forest [25]	Support Vector Machine [26]	Proposed Model
High	0.99	0.99	0.92	0.99
Medium	0.07	0.76	0	0.99
Low	0.29	0.94	0.9	0.99
Very Low	0.78	0.98	0.87	0.97

Further, we also calculated the accuracy score as shown in Fig.3. Accuracy can be computed as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

Table.7. Computation of F1-Scores

Class Label	Gaussian Naïve Bayes [24]	Random Forest [25]	Support Vector Machine [26]	Proposed Model
High	0.99	0.99	0.96	0.99
Medium	0.1	0.64	0	0.99
Low	0.12	0.94	0.69	0.98
Very Low	0.85	0.99	0.88	0.98

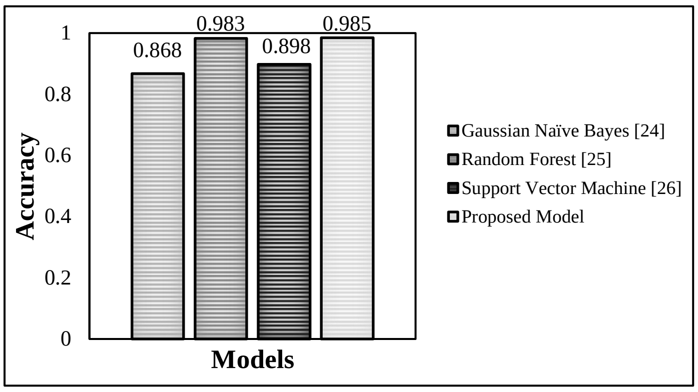

Fig.3. Computation of Accuracy

As it can be observed from Fig.3 the meta-classifier performs better with an accuracy of 0.985 as compared to 0.868 of [24], 0.983 of [25], and 0.898 of [26]. The Table.8 provides a comparison corresponding to other important metrics which are Macro Avg Precision, Weighted Avg Precision, Macro Avg Recall, Weighted Avg Recall, Macro Avg F1-Score, and Weighted Avg F1-Score. Here also, the meta-classifier can be seen to perform better.

Table.8. Computation of other metrics

	Gaussian Naïve Bayes [24]	Random Forest [25]	Support Vector Machine [26]	Proposed Model
Macro Avg Precision	0.53	0.87	0.61	0.98
Weighted Avg Precision	0.92	0.98	0.93	0.98
Macro Avg Recall	0.54	0.92	0.67	0.98
Weighted Avg Recall	0.86	0.98	0.9	0.98
Macro Avg F1-Score	0.51	0.89	0.63	0.98
Weighted Avg F1-Score	0.89	0.98	0.91	0.98

Fig.4 presents the comparison of the balanced accuracy score of the proposed model. By taking an average of the recall values of each class, the balanced accuracy score accounts for the class distribution and offers a more accurate indication of the performance. The proposed meta-classifier model has attained a balanced accuracy score of 0.985 against 0.53 of [24], 0.918 of [25], and 0.894 of [26]. As per these values, it is clear that the meta-classifier gives a better predictive performance for all class labels of the output variable.

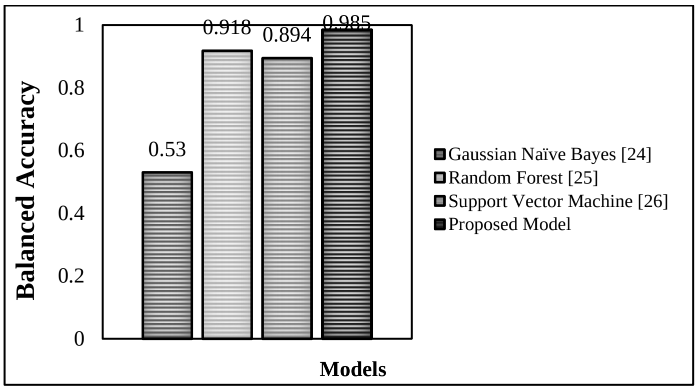

Fig.4. Computation of balanced accuracy score

Considering the scope of this study, we have not explored the scalability aspect of the proposed model as we have evaluated it on only the Ministry of Finance, KSA dataset as provided by [8]. Our work assumes comparing the performance of the proposed model with existing works that utilised the same dataset.

This study utilises a homogenous cloud dataset as its main source of empirical data in order to explore workload prediction in the cloud computing environment. By addressing the effects, issues, and opportunities related to the research problem under consideration in homogenous cloud systems, we want to further knowledge of the dynamics of cloud computing.

6. CONCLUSION AND FUTURE WORK

Considering the highly dynamic nature of the Cloud workload and the need for strict adherence to SLA parameters, provisioning of Cloud resources to meet the demands for services becomes very crucial. In order to provision the required resources proactively, since the reactive approach is slow, the resource management module amongst others must be equipped with the ability to accurately forecast the future workload. With the aim of addressing this issue, we have proposed a meta-classifier model in this work. The results that we obtained by implementing the proposed model were both plausible and coherent. As a future extension, further enhancement in the predictive performance of the proposed model can be explored by different configurations of hyperparameters of the stacked models and meta-classifier as well. Further, the scalability aspect of the proposed model can be evaluated in the future to ascertain its performance on larger datasets.

REFERENCES

[1] Mell P, "The NIST Definition of Cloud Computing", Technical Report, Computer Security Division, Information

Technology Laboratory, National Institute of Standards and Technology, 2011.

[2] M.C. Calzarossa, M.I. Tabash and D. Tessera, "*Workloads in the Clouds*", Springer, 2016.

[3] R.N. Calheiros, E. Masoumi, R. Ranjan and R. Buyya, "Workload Prediction using ARIMA Model and its Impact on Cloud Applications' QoS", *IEEE Transactions on Cloud Computing*, Vol. 3, No. 4, pp. 449-458, 2015.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing", *Communications of the ACM*, Vol. 53, No. 4, pp. 50-58, 2010.

[5] C. Qu, R.N. Calheiros and R. Buyya, "Auto-Scaling Web Applications in Clouds: A Taxonomy", *ACM Computing Surveys*, Vol. 51, No. 4, pp. 1-33, 2018.

[6] B. Dougherty, J. White and D.C. Schmidt, "Model-Driven Auto-Scaling of Green Cloud Computing Infrastructure", *Future Generation Computer Systems*, Vol. 28, No. 2, pp. 371-378, 2012.

[7] D.H. Wolpert, "Stacked Generalization", Neural Networks, Vol. 5, No. 2, pp. 241–259, 1992.

[8] A.M. Al Faifi, B. Song, M.M. Hassan, A. Alamri and A. Gumaei, "Data on Performance Prediction for Cloud Service Selection", *Data in Brief*, Vol. 20, pp. 1039-1043, 2018.

[9] G.M. Wamba, Y. Li, A.C. Orgerie, N. Beldiceanu and J.M. Menaud, "Cloud Workload Prediction and Generation Models", *Proceedings of International Symposium on Computer Architecture and High-Performance Computing*, pp. 89-94, 2017.

[10] S. Kumar, N. Muthiyan, S. Gupta, A.D. Dileep and A. Nigam, "Association Learning based Hybrid Model for Cloud Workload Prediction", *Proceedings of International Joint Conference on Neural Networks*, pp. 1-3, 2018.

[11] C. Liu, J. Jiao, W. Li, J. Wang and J. Zhang, "Tr-Predictior: An Ensemble Transfer Learning Model for Small-Sample Cloud Workload Prediction", *Entropy*, Vol. 24, No. 12, pp. 1-17, 2022.

[12] A.I. Maiyza, N.O. Korany, K. Banawan, H.A. Hassan and W.M. Sheta, "VTGAN: Hybrid Generative Adversarial Networks for Cloud Workload Prediction", *Journal of Cloud Computing*, Vol. 12, No. 1, 2023.

[13] M.N. Hasan Shuvo, M.M. Shahriar Maswood and A.G. Alharbi, "LSRU: A Novel Deep Learning based Hybrid Method to Predict the Workload of Virtual Machines in Cloud Data Center", *Proceedings of IEEE International Conference on Cloud Computing*, pp. 1604-1609, 2020.

[14] F. Farahnakian, P. Liljeberg and J. Plosila, "LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers", *Proceedings of Euromicro Conference on Software Engineering and Advanced Applications*, pp. 357-365, 2013.

[15] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura and R. Bianchini, "Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms", *Proceedings of Symposium on Operating Systems Principles*, pp. 153-158, 2017.

[16] J. Gao, H. Wang and H. Shen, "Machine Learning Based Workload Prediction in Cloud Computing", *Proceedings of International Conference on Computer Communications and Networks*, pp. 1-7, 2020.

[17] Y. Yu, V. Jindal, F. Bastani, F. Li and I. L. Yen, "Improving the Smartness of Cloud Management via Machine Learning Based Workload Prediction", *Proceedings of IEEE Annual Conference on Computer Software and Applications*, pp. 38-45, 2018.

[18] P. Yazdanian and S. Sharifian, "Cloud Workload Prediction using ConvNet and Stacked LSTM", *Proceedings of Iranian Conference on Signal Processing and Intelligent Systems*, pp. 83-92, 2018.

[19] S. García, S. Ramirez-Gallego, J. Luengo, J.M. Benitez and F. Herrera, "Big Data Preprocessing: Methods and Prospects", *Big Data Analytics*, Vol. 1, No. 1, pp. 1-22, 2016.

[20] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification", *IEEE Transactions on Information Theory*, Vol. 13, No. 1, pp. 21-27, 1967.

[21] S.B. Kotsiantis, "Decision Trees: A Recent Overview", *Artificial Intelligence Review*, Vol. 39, No. 4, pp. 261-283, 2013.

[22] J.H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine", *The Annals of Statistics*, Vol. 29, No. 5, pp. 1189-1232, 2001.

[23] S.C. Wang, "*Interdisciplinary Computing in Java Programming*", Springer Science and Business Media, 2003.

[24] A.M. Al-Faifi, B. Song, M.M. Hassan, A. Alamri and A. Gumaei, "Performance Prediction Model for Cloud Service Selection from Smart Data", *Future Generation Computer Systems*, Vol. 85, pp. 97-106, 2018.

[25] S.T. Singh, M. Tiwari and A.S. Dhar, "Workload Prediction Model for Autonomic Scaling of Cloud Resources with Machine Learning", *Proceedings of International Conference on Intelligent Systems and Smart Infrastructure*, pp. 343-351, 2023.

[26] S.T. Singh, M. Tiwari and A.S. Dhar, "Machine Learning based Workload Prediction for Auto-Scaling Cloud Applications", *Proceedings of OPJU International Technology Conference on Emerging Technologies for Sustainable Development*, pp. 1-12, 2023.