Failure Characterization and Prediction of Scheduling Jobs in Google Cluster Traces

Mohammad S. Jassas
Department of Computer Science
Umm Al-Qura University
Makkah, Saudi Arabia
msjassas@uqu.edu.sa

Qusay H. Mahmoud
Department of Electrical, Computer and Software Engineering
Ontario Tech University
Oshawa, ON, L1G 0C5 Canada
qusay.mahmoud@uoit.ca

2019 IEEE 10th GCC Conference & Exhibition (GCC) 978-1-5386-9477-0/20/$31.00 ©2020 IEEE 10.1109/GCC45510.2019.1570516010

***Abstract*—In cloud computing, all services including infrastructure, platform, and software experience failures due to their large scale and heterogeneity nature. These failures can lead to job failure execution that may cause performance deterioration and resource waste. Most studies have focused mainly on failure analysis and characterization while there is limited research has been done on failure prediction. In this paper, the overall aim is to develop a failure prediction framework that can early detect failed jobs, and the real advantages of this framework are to decrease the resources waste and to increase the performance of cloud applications. Our failure analysis and prediction are based on Google cluster traces. We have developed a failure prediction model for job failure execution based on applying different machine learning algorithms and selecting the best accurate model. Moreover, we evaluate the model performance using different types of evaluation metrics to ensure that the proposed prediction model provides the highest accuracy of predicted values. Finally, we apply different feature selection techniques to improve the accuracy of our proposed model. Our evaluation results show that our model has achieved a high rate of precision, recall, and f1-score.**

***Index Terms*—Cloud computing, Google cluster traces, failure prediction, failed and finished jobs, decision trees.**

I. Introduction

While the number of cloud services has been increased, the cloud architectures have become more complicated due to their large scale and heterogeneity nature. Thus, cloud consumers are concerned in term of the cloud availability and reliability because recently several cloud services, including software and infrastructure services, experience failures. In fact, the reliability and availability concerns have become one of the most significant challenges facing both the traditional High Performace Computing (HPC) [1] and the cloud environments so that the complexity of cloud architectures can increase the probability rate of failure [2], [3], [4].

Failed jobs consume a significant amount of memory and computational resources. As a result, when the number of failed jobs increases, the cloud resources such as CPU, memory, and disk space will be wasted. Several prior studies [2], [4] have characterized and analyzed Google cluster traces [5]. The vast majority of the work in this area has focused on failure analysis and characterization while there is limited research investigating the job failure prediction approach [6], [3], [7], [8]. Failure prediction can be applied as one of proactive fault tolerance techniques to increase the availability of the cloud applications.

A failure prediction approach has designed and implemented depending on measuring the most critical metrics of the cloud applications. The primary objective of failure prediction is to examine the frequent changes and the behaviour of cloud applications to decrease the number of failed jobs and to increase cloud application performance.

This paper extends our previous research [9], and in this paper we introduce a new model of failure prediction for cloud job execution and answering the following question: how can we accurately classify and predict the incoming job event types (fail, success) before the management system schedules them. To the best of our knowledge, no study has focused on design and implement the failure prediction model based on comparing different classification algorithms and evaluate the model performance using different evaluating metrics to ensure that the proposed prediction model provides high accuracy of predicted values.

The main contributions of this paper are:

- A model for predicting the job/task failure.
- An implementation of the failure prediction model based on a supervised learning approach and applying different classification algorithms include Decision Trees (DTs), Logistic Regression (LR), K-Nearest Neighbors (KNN), Naive Bayes (NB), Random Forests (RF), and Quadratic Discriminant Analysis (QDA).
- Performance evaluation model using different types of evaluation metrics in order to ensure that the proposed prediction model provides high level accuracy of predicted values.
- Applications of feature selection techniques such as SelectKBest, RFE, and feature importance to increase the accuracy of our failure prediction model.

Section II describes the scheme of Google cluster traces and presents basic statistics and behavior for failed jobs. Section III discusses the related work. The proposed framework and data preprocessing and filtration steps are explained in Section IV. The evaluation matrices and results are reported in Section V. In Section VI, we present a comparison of different classification models while we explain some techniques that can be utilized to increase the accuracy of our model in Section VII.

TABLE I: TRACES OVERVIEW [9]

Trace Characteristic	Value
Number of users	393 users
Submitted jobs	150,038 jobs
Scheduling jobs	150,037 jobs
Finished jobs	89,541 jobs
Failed jobs	1,819 jobs
Number of tasks	5,189,267 tasks
Submitted tasks	11,758,184 tasks
Number of scheduling jobs	11,535,100 tasks
Finished tasks	4,142,762 tasks
Failed tasks	3,942,709 tasks

The conclusion of this study and future work are presented in Section VIII.

II. BACKGROUND

A large publicly available dataset names as Google cluster traces [5] has been released by Google. This dataset contains many files of monitored data that are generated from a large system includes more than 12,500 nodes. The Google cluster traces contain 672,074 jobs and more than 26 million tasks, and these jobs are submitted from May 1st to May 29th. Fundamental statistics in the first week of Google cluster traces are summarized in Table 1. Moreover, we notice that each task has different characteristics based on its class schedule, priority, requested resources and actual resource usage.

A brief description of the most important tables, which are used in this study, are presented as follows:

A. *Job Events Table*

Jobs events table shows jobs ID with the event types which indicate the status of each job. There are seven different event types including submitted, scheduled, evicted, failed, finished, killed, and lost jobs.

The basic scenario of the job life cycle is presented in Figure 1: a job is submitted and then a job will be inserted into a pending queue. After that, the job will be scheduled onto an available machine, and it starts running. Then, after a period which considers as execution time, the job is finished successfully. Otherwise, if a job or task is failed for any reasons, the cluster system attempts to restart tasks that have a failed type [5], [9].

B. *Task Events Table*

This table presents a detailed description of each task related to a particular job. Each job has one or several tasks, and each task has some processes. Task Events Table has number of attributes which are a timestamp, event type, index of tasks, priority, and information about resource requests.

We have statistically analyzed and characterized the job status of the Google traces in order to understand the job/task failure behaviors and also compare the failed cases with another termination status. The results are presented in Figure 2. In the first week of Google dataset, we notice that the lost task termination rate is nearly zero. However, the evicting, failure, finishing, and killing rates are 12.5%, 33.4%, 35.12%, and 18.9% respectively.

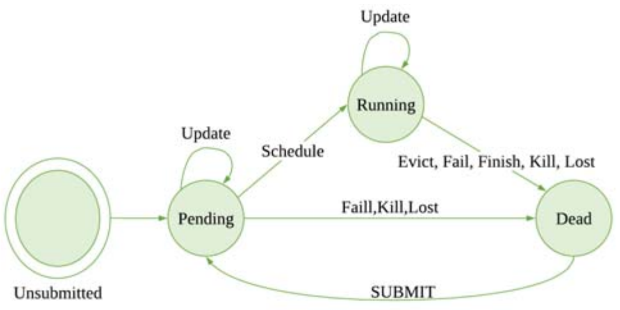

Fig. 1: State diagram fo jobs and tasks [5]

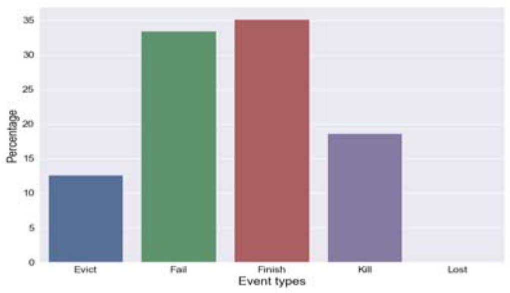

Fig. 2: The percentage of different task status in a week

Approximately 34% of scheduled tasks are recorded as failed tasks, most of these tasks are resubmitted many times. This high rate of failures leads to waste a large amount of cloud resources and increase the utilization time. Thus, applying failure job prediction can help to identify failed jobs early, and also enhance the efficiency of cloud resource utilization. On the other hand, the cloud management can make the appropriate decision based on the output of the failure prediction model.

III. RELATED WORK

A. *Improving the dependability*

In the distributed computational area, there are several studies focus on improving the reliability and availability of systems. Recognizing and analyzing the prevalence of failures that occur in all layers of a distributed system is a significant factor to increase the system dependability. Many techniques have been utilized for design and implement fault-tolerant systems, especially for distributed systems [10]. Adding a redundancy technique is one of the common strategies to increase the dependability of different system components and layers. In term of fault-tolerant hardware features, a redundant array of independent disks (RAID) can be used instead of focusing on tolerating failures in software for increasing systems availability. Barroso et al. [11] apply a replication technique to increase applications performance and the reliability of job executions.

B. Failure analysis and prediction

Failure analysis and characterization have been studied widely in grid computing, cloud cluster and supercomputer [12]. Fadishei and et al. [13] study the workload features such as memory usage, CPU speed and utilization, the job running time and other monitoring metrics. They find that the correlation between the failed jobs and the workload attributes. Liang et al.[12] utilize log files from the BlueGene machine of IBM to predict failure based on investigating the characteristics of fatal failure events and finding the correlation between non-fatal and fatal events. Pan et al. [14] propose an approach called Ganesha which is a black-box diagnosis technique that utilizes the operating system metrics in order to identify and diagnose faults in MapReduce systems. Some studies focus more on characterizing the reliability of cloud computing hardware. For example, Vishwanath et al. [15] presented a detailed analysis of failure predictors and failure characteristics. The main objective of their study is to understand the hardware dependability in the cloud computing environment. The results of the paper show that hard disks are one of the lower reliable components, and approximately 8% of all servers can expect a minimum of one hardware failure for a year.

C. Google cluster traces

The Google cluster traces [5] are used in different research studies, including workload trace characterization [3] and applying statistical methods in order to compare Google datacentres, which consider as a cloud environment, to Grid or HPC systems [16]. Garraghan et al. [17] have used the same set of Google traces to analyze the server characteristics and resource utilization on the Google platform. They also study the impact of tasks that are terminated before successfully completed regarding wasted resource utilization per server. Liu et al. [18] analyzed and characterized the Google dataset, including machine distribution. They also present some statistics that are related to the job, task, and machine events. Reiss and et al. [4] analyze the Google dataset to highlight the heterogeneous and highly dynamic behavior of the big-data trace. However, this study does not take into consideration job failure analysis and prediction, so their work is limited to the general analysis of the Google cluster trace.

Most studies have focused mainly on failure analysis and characterization while there is limited research has been done on failure prediction [8], [6], [3], [7]. Rosa et al.[8] have proposed an online technique for predicting job status. Their integrated method is based on Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Linear Regression (LR). The classification model is designed to be updated at the end of each day. El-Sayed et al. [6] have designed a job failure prediction model using a machine learning algorithm which is Random Forests (RF). The results show that they can successfully recall up to 94% of all failed jobs with at least 95% precision. The limitation of this previous work is that it is only applied based on one classification algorithm without comparing their models with other classification models in order to ensure the accuracy of their results. Thus, we applied different classification algorithms which are DTs, LR, KNN, NB, RF, and QDA. Then, we selected the best model based on applying different evaluation metrics and feature selection techniques. As a result, a comparison with the previous studies, we have achieved the highest precision, recall, f1-score, and we applied different evaluated methods to ensure that our failure prediction model is accurate.

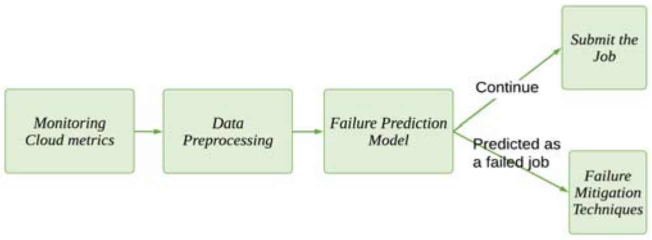

Fig. 3: Proposed Framework

IV. Failure Prediction Framework

A. Overview

Four important phases of implementing the proposed framework are as follows: 1) The cloud management system is responsible for monitoring the application metrics and the requested resources such as memory, CPU, and disk space. Then, the monitored data will be stored in the cloud storage in order to be used for the data prepossessing and failure analysis and prediction phases. 2) Applying the data prepossessing and filtration techniques are the most important step in the data science world as the quality of the failure prediction model is based on this step. 3) The machine learning algorithm will be applied to the Google cluster traces in order to predict the failed job/task. 4) Finally, based on the prediction results, the cloud management system will make the appropriate decision. If the job is predicted as a successful job, the job continues to be submitted and normally scheduled to the available nodes. Otherwise, if the job is predicted as a failed job, failure mitigation techniques will be applied, and it is important to note that this final step will be considered in the future work. The framework phases are presented in Figure 3.

The main aim of the proposed framework is to predict the job/task failure in the cloud with high accuracy using available machine learning classification algorithms, and this technique assists to reduce wasted cloud resources and improve utilization of cloud infrastructures.

B. Data Preprocessing and Filtering

The input of this framework is a feature set which describes the main attributes of jobs and a cloud system. We have filtered out some records from the dataset in order to be ready for applying a failure prediction algorithm. The failure behavior for the first seven days is shown in Figure 4. The data preprocessing steps are presented in detail in our previous research paper [9].

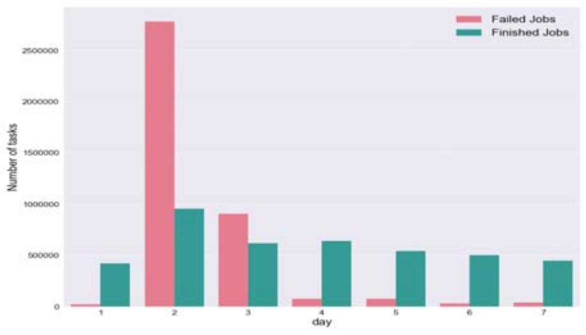

Fig. 4: The failure behavior for the first seven days

C. Prediction Technique

We apply a Decision Trees (DTs) which is one of the supervised learning algorithms. The goal of implementing the DTs is to build a prediction model that can predict the value of target variables, so DTs algorithm applies some decision rules to learn from the data features. The criterion that we use for measuring the quality of a split is entropy. DTs algorithm uses entropy to calculate the homogeneity (similar values) of a sample. The cost complexity can be measured based on the number of leaves in the tree and error rate of the tree. We split the Google trace dataset into 75% for a training set and 25% for a testing set.

V. Evaluation and Results

A. Experimental Setup

The Google cluster trace is stored in Comma Separated Value (CSV) files. The trace size is approximately 200GB which consider as a big data, and key-value structure is utilized to represent the data attributes. We read these data in Panda data frames; then we store the data frames in pickle format in order to decrease the data processing time. Then we use scikit-learn which is machine learning packages in python [19] to implement the failure prediction model.

B. Feature selection

We have manually selected the most relevant features in order to avoid the curse of dimensionality, shorter preprocessing training times, and reducing overfitting.

C. Evaluation Metrics

Evaluating the model performance is the one of the most important task in the data science which indicates how accurate the model is. In order to evaluate the proposed model, it is important to take into consideration these metrics which are accuracy, precision, recall, and F1-score. Before we dig deeper into the evaluation of model performance, we will explain the concept of confusion matrix and evaluation metrics.

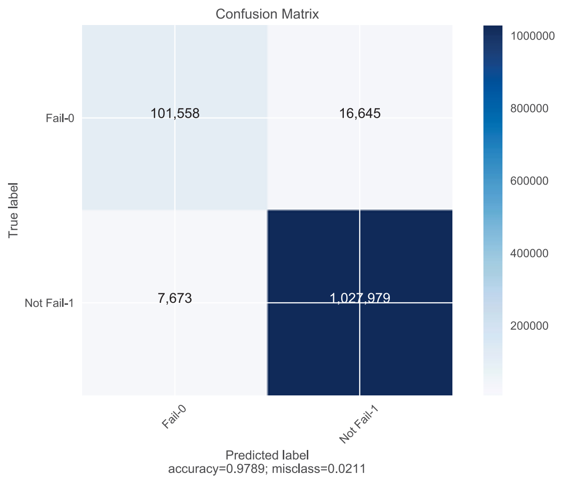

Fig. 5: The confusion matrix for total number of failed and succeed tasks

1) Confusion matrix: a confusion matrix is a table that is used to express the performance of a classification model. Each column represents the observations in an actual class while each row of the matrix represents the observations in a target class. All the evaluation metrics can be calculated based on confusion matrix values. As shown in Figure 5, True Positive (TP) and True Negative (TN) are the observations of the correct prediction results. However, False Positive (FP) and False Negative (FN) are the observations of the incorrect prediction results, so we are interested in minimizing the False Positive and False Negative in order to increase the accuracy of our failure prediction model.

2) Accuracy: accuracy is a first intuitive step to measure the performance of the classification model. Accuracy is calculated based on a ratio of correctly predicted samples to the total samples. It is not necessary that if we have achieved a high accuracy, our model is best. Accuracy is a great measure when the value of false negative and false positive are almost the same. Thus, we require looking at other metrics in order to evaluate the performance of our model. For our failure prediction model, we have achieved high accuracy with 0.97. The accuracy has calculated based on the following equation:

$$Accurcy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

3) Precision: precision is another metrics to measure the model performance based on calculating the ratio of correctly predicted positive samples to the total predicted positive samples. This metric can answer an important question which is how many jobs/tasks actually succeed (finish) among all jobs/tasks that are labeled as finish. Low False Positive Rate (FPR) indicates that we have high precision. We have achieved a high precision in both classes (fail, success) which are 0.93 and 0.98 respectively. The overall average of precision is 0.98. The precision has calculated based on the following equation:

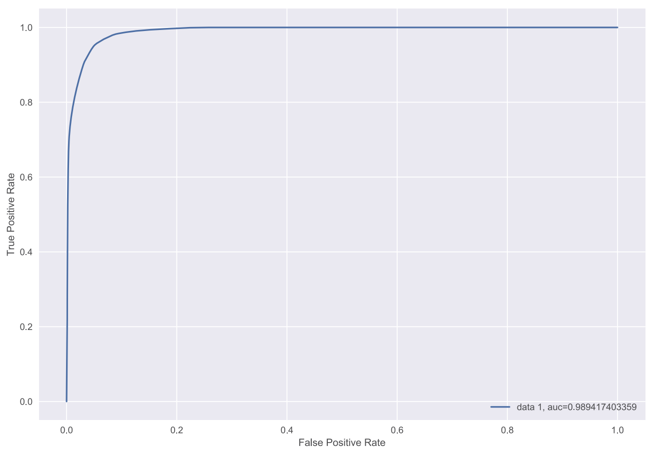

Fig. 6: ROC curve of the failure predictive model using DTs

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

4) Recall: recall is another metrics to measure the model performance based on calculating the ratio of correctly predicted positive samples to the all samples that are in the actual class. The recall answers an important question which is how many jobs/tasks did we correctly label as successfully completed (finish) among all the jobs/tasks that is actually succeeded. We have achieved a high recall in both classes (fail, success) which are 0.86 and 0.99 respectively. The overall average of recall is 0.98. The recall has calculated based on the following equation:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

5) F1-score: F1-score is the weighted average of recall and precision. F1-score in many cases is more important than accuracy. We have achieved a high F1-score in both classes (fail, success) which are 0.89 and 0.99 respectively. The overall average of F1-score is 0.98. The F1-score has calculated based on the following equation:

$$F1 - score = \frac{2 * (Recall * Precision)}{Recall + Precision} \tag{4}$$

D. DTs ROC Curve

The main objective of calculating the Receiver Operating Characteristic (ROC) curve is to visualize the performance of a binary classifier. However, before we explain the ROC curve, we need to define two important concepts in order to calculate ROC curve. These two concepts are sensitivity and specificity. Sensitivity is the same as recall which presents the true positive rate (TPR) as seen in eq (3); specificity presents the true negative rate (TNR) as seen in eq (5) which measures the proportion of negatives that are correctly identified.

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

TABLE II: K-FOLD CROSS VALIDATION

	3-Folds	**5-Folds**	**10-Folds**	**15-Folds**
Precision	98.3%	98.3%	98.4%	98.3%
Recall	99.2%	99.2%	99.2%	99.2%
F1-score	98.8%	98.8%	98.8%	98.8%

TABLE III: ACCURACY COMPARISON FOR FAILED TASKS CLASS

	DTs	**LR**	**KNN**	**NB**	**RF**	**QDA**
Precision	93%	91%	93%	62%	93%	70%
Recall	86%	62%	86%	70%	86%	68%
F1-score	89%	74%	89%	66%	89%	69%

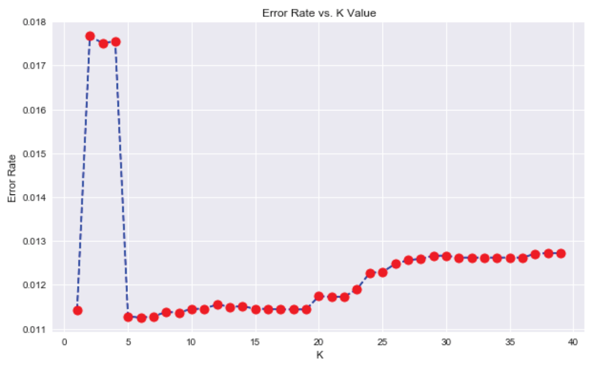

Fig. 7: The elbow method in order to pick the optimum K-Value

Figure 6 shows ROC curve of the predictive failure model. The ROC curve shows the trade-off between the specificity and sensitivity, so it important to note that any increase in specificity will lead to decrease in sensitivity. In order to combine the TPR and FPR into one metric, it is required first to calculate the two metrics with many different thresholds between the value of (0.00 to 1.00). The output of this computation is called ROC curve. When the curve is very close to the left-hand border of the ROC space, we can state that the proposed model is accurate. The area under the curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (failed jobs/finished jobs). Our prediction model has achieved a high AUC with 0.98.

E. K-fold cross validation

In the previous section, we have evaluated our proposed model using different evaluation metrics include general classification accuracy, precision, recall, f1-score, ROC curve. In this section, we apply another evaluated method which is cross-validation. Cross-validation is a solution for solving the problem of guaranteeing that each observation is used for training and testing purpose in an equal number of times. Thus, one of the main advantages of cross-validation is reducing the variance of an accuracy score. The K-fold cross-validation, where k is the number of splits to apply in the dataset is the best way to evaluate the proposed model. We have selected four different K-value for cross-validation test. The results of K-fold cross-validation are shown in Table II.

TABLE IV: TRAINING AND TESTING TIME FOR ALL APPLIED MODELS

	DTs	LR	KNN	NB	RF	QDA
Training time	14.82	7.92	10240.67	0.80	65.96	1.02
Testing time	0.15	0.03	4381.46	0.28	1.50	0.28

VI. Comparing DTs with Other Algorithms

We have achieved high accuracy using DTs algorithm, but we are interested in applying other classification algorithms to select the best model in term of accuracy, performance and complexity. Thus, in this section, we discuss some classification algorithms that can be applied for implementing the failure prediction model. Then, we evaluate the performance of each algorithm based on applying different evaluation metrics.

We have applied six different classification algorithms which are Decision Trees (DTs), Logistic Regression (LR), K-Nearest Neighbors (KNN), Naive Bayes (NB), Random Forests (RF), and Quadratic Discriminant Analysis (QDA). Table III presents the evaluation results of our experiment based on the most important metrics which are precision, recall, and f1-score. The results show that the DTs, RF and KNN have achieved the best accuracy. However, the KNN has a log training time of 10240.67 seconds compare to DTs, LR, NB, RF, and QDA which have training time of 14.82, 7.92, 0.80, 65.96, and 1.02 seconds respectively as shown in Table IV. For the KNN algorithm, we have selected the K-Value=5 based on using elbow method in order to pick the optimum K-Value as shown in Figure 7. Thus, when the K= 5, the confusion matrix shows better results.

As a result, after we have applied different models and evaluated them, it is clear that DTs and RF classification are the best two models in term of accuracy, precision, recall, and f1-score. However, the DTs tree has less complexity and performance than the RF because DTs does not require long training time. Thus, in the next section, we are interested in improving the accuracy of these two models by applying some techniques such as feature selections. Then, we evaluate both models and select the best one to be our proposed failure prediction model.

Figure 8 shows the ROC curves of the different models. DTs and RF has achieved the highest AUC with 98% while the KNN comes in the next place with 89%. Even though the KNN has achieved 89% AUC, the training time was very long compared with other classification algorithms.

VII. Improve the Model Accuracy of Failure Prediction

Based on our experiment for comparing number of classification models in order to select the best model that can be selected as a failure predication model, we find that the DTs and RF have achieved the best accuracy results. In this section, we present some techniques that can be utilized to increase the accuracy of our model.

Features selection is one of the most important methods that can be used to increase the accuracy of our model based on automatically select a number of features that have high contribution to the model output. If the model has trained use irrelevant features, it can decrease the accuracy of the proposed model. There are many other benefits of applying features selection such as reducing overfitting and reducing training time.

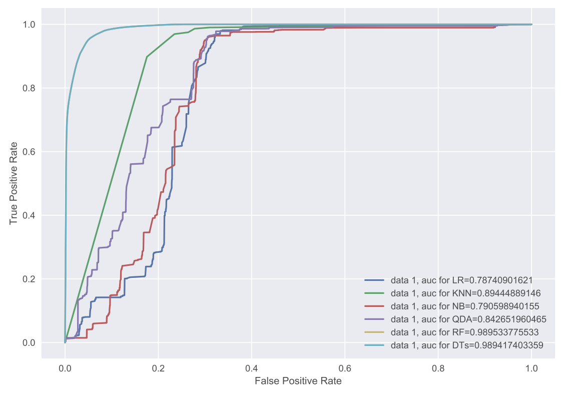

Fig. 8: ROC curve of different classification algorithms

A. *SelectKBest*

The SelectKBest technique is applied based on statistical tests in order to select those attributes that have the strongest relationship with the target variable. After we perform this technique, we have obtained the most important features which are job ID, task index, machine ID, scheduling class, and priority.

B. *Feature Importance*

Bagged decision trees such as Random Forest (RF) can be used to estimate which features have the most contribute in order to predict the highest accuracy. After we apply this technique, we have obtained the most important features which are job ID, day, machine ID, scheduling class, and disk space.

C. *Recursive Feature Elimination*

The Recursive Feature Elimination (RFE) is one of well known feature selection techniques. RFE works by recursively eliminating features and building a model on those features that remain. RFE use the model accuracy in order to identify which features that can contribute the most to predicting the output target. After we apply the RFE feature selection technique, we have obtained the most important features which are job ID, hour, task index, machine ID, and CPU.

Table V presents the evaluation results of performing different feature selection techniques. The RF classification model has achieved the highest accuracy with the average of 99% for precision, recall, and f1-score using RFE technique. As a result, applying feature selection techniques to our job failure prediction has a significant impact on increasing the accuracy of our proposed model.

TABLE V: COMPARISON BETWEEN DTs AND RF CLASSIFICATION ALGORITHMS

	Decision Tree Classifier					**Random Forest Classifier**				
	Precision	Recall	F1-score	Training Time	Testing Time	Precision	Recall	F1-score	Training Time	Testing Time
SelectKBest	92%	92%	92%	34.40	0.25	97%	90%	93%	144.94	2.72
RFE	93%	94%	94%	34.63	0.24	97%	93%	95%	133.82	2.37
Feature Importance	92%	93%	92%	29.31	0.25	94%	92%	93%	124.99	2.49

VIII. CONCLUSION & FUTURE WORK

We developed a job failure prediction model based on applying a machine learning approach that can be implemented in the large datacenters or cloud computing environments. The model can early detect the failed jobs before the cloud management system schedules them. At the beginning of our experiment, we developed our failure prediction model based on the DTs classification algorithm which can predict the job status with an accuracy rate of 98% and an overall average of F1-score is 98%. Then, we have applied some techniques based on different feature selection approaches in order to increase the accuracy of our failure prediction model. Finally, we have achieved higher accuracy based on RF classification algorithm with an accuracy rate of 99%, and an overall average of F1-score is 99%

In future work, we are planning to continue improving the accuracy and performance of our failure prediction model by applying deep learning techniques. Future research will consider the online failure prediction and mitigation policies and techniques.

ACKNOWLEDGMENT

The first author would like to thank Umm Al-Qura University, Makkah, Saudi Arabia for funding this work as part of his graduate scholarship.

REFERENCES

[1] M. Snir, R. Wisniewski, and J. Abraham, “Addressing Failures in Exascale Computing,” 2013. [Online]. Available: http://www.osti.gov/bridge/servlets/purl/1078029/

[2] A. Rosà, L. Chen, R. Birke, and W. Binder, “Demystifying Casualties of Evictions in Big Data Priority Scheduling,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 4, pp. 12–21, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2788406

[3] X. Chen, C. D. Lu, and K. Pattabiraman, “Failure analysis of jobs in compute clouds: A google cluster case study,” in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, Nov 2014, pp. 167–177.

[4] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. a. Kozuch, “Heterogeneity and dynamicity of clouds at scale : Google Trace Analysis,” *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC '12*, pp. 1–13, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2391229.2391236

[5] C. Reiss, J. Wilkes, and J. J. L. Hellerstein, “Google cluster-usage traces: format+ schema,” *Google Inc., . . .*, pp. 1–14, 2011. [Online]. Available: http://googleclusterdata.googlecode.com/files/Google cluster-usage traces - format + schema (2011.10.27 external).pdf

[6] N. El-Sayed, H. Zhu, and B. Schroeder, “Learning from Failure Across Multiple Clusters: A Trace-Driven Approach to Understanding, Predicting, and Mitigating Job Terminations,” *Proceedings - International Conference on Distributed Computing Systems*, pp. 1333–1344, 2017.

[7] C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, and J. Chen, “Predicting of Job Failure in Compute Cloud Based on Online Extreme Learning Machine: A Comparative Study,” *IEEE Access*, vol. 5, pp. 9359–9368, 2017.

[8] A. Rosa, L. Y. Chen, and W. Binder, “Predicting and mitigating jobs failures in big data clusters,” *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015*, pp. 221–230, 2015.

[9] M. Jassas and Q. H. Mahmoud, “Failure analysis and characterization of scheduling jobs in google cluster trace,” in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 3102–3107.

[10] A. S. Tanenbaum and M. V. Steen, *Distributed Systems:Principles and Paradigms*, 2006.

[11] L. A. Barroso, J. Dean, and U. Holzle, “Web search for a planet: The google cluster architecture,” *IEEE micro*, vol. 23, no. 2, pp. 22–28, 2003.

[12] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, “Bluegene/l failure analysis and prediction models,” in *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. IEEE, 2006, pp. 425–434.

[13] H. Fadishei, H. Saadatfar, and H. Deldari, “Job failure prediction in grid environment based on workload characteristics,” in *Computer Conference, 2009. CSICC 2009. 14th International CSI*. IEEE, 2009, pp. 329–334.

[14] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, “Ganesha: Blackbox diagnosis of mapreduce systems,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 8–13, 2010.

[15] K. V. Vishwanath and N. Nagappan, “Characterizing Cloud Computing Hardware Reliability,” *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, p. 193, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1807128.1807161

[16] S. Di, D. Kondo, and W. Cirne, “Characterization and comparison of cloud versus grid workloads,” *Proceedings - 2012 IEEE International Conference on Cluster Computing, CLUSTER 2012*, pp. 230–238, 2012.

[17] P. Garraghan, P. Townend, and J. Xu, “An analysis of the server characteristics and resource utilization in google cloud,” in *2013 IEEE International Conference on Cloud Engineering (IC2E)*, March 2013, pp. 124–131.

[18] Z. Liu and S. Cho, “Characterizing machines and workloads on a Google cluster,” *Proceedings of the International Conference on Parallel Processing Workshops*, pp. 397–403, 2012.

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