مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(٧) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (7) 2023

Software Faults Detection in Cloud Computing Systems Using Machine Learning Algorithms

Prof. Yaroub Dayoub*
Eng. Aisha Hamedi**

(Received 12/7/2023 . Accepted 10/9/2023)

□ ABSTRACT

Cloud computing is a new technology in the field of distributed computing, and its use is increasing daily and with the wide development of cloud computing systems and due to the growing complexity of cloud systems and the large volume and noise of data. All that causes the increased possibility of faults in these systems, which can lead to high-severity failures such as wasting resources and data losses. Thus, cloud computing providers will lose their credibility. This paper presents an effective novel approach that provides fault detection, reduces the impact of error, and provides accurate predictions within a sufficient period of time in order to ensure reliability and quality of service, reduce wastage, and improve accuracy. We applied the approach in the OpenStack cloud computing platform to fault injection data and data pre-processing. The approach leverages machine learning algorithms by applying a number of algorithms to improve the accuracy of predictions. The algorithms are the Support Vector Classification (SVC), Random Forest (RF), k-Nearest Neighbours (KNN), Decision Tree (DT), and Logistic Regression (LG). The results indicate that the prediction accuracy of our model using the SVM classifier when predicting faults is 97% accurate and effective compared to other algorithms. These results show that our method can effectively predict future system and application faults.

Key Words: Software Faults, Fault Injection, error, Cloud Computing, cloud, Machine Learning Algorithms (MLAs), Preprocessing Data.

[*] Professor in Department of Information Technology, Faculty of Information and Communication Technology Engineering, Tartous University, Syria
[**] Master student at Information Technology Engineering Department, Faculty of Information and Communication Technology Engineering, Tartous University, Syria.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(٧) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (7) 2023

كشف الأعطال البرمجية في أنظمة الحوسبة السحابية باستخدام خوارزميات تعلم الآلة

أ.د. يعرب ديوب*
م. عائشة حميدي**

(تاريخ الإيداع ٢٠٢٣/٧/١٢ . قُبِل للنشر في ٢٠٢٣/٩/١٠)

□ ملخّص□

الحوسبة السحابية هي تقنية جديدة في مجال الحوسبة الموزعة، ويتزايد استخدامها يوميًا ومع التطور الواسع لأنظمة الحوسبة السحابية وبسبب التعقيد المتزايد للأنظمة السحابية والحجم الكبير وضجيج البيانات. كل هذا سبب زيادة احتمالية حدوث أعطال في هذه الأنظمة، مما قد يؤدي إلى حالات فشل شديدة الخطورة مثل إهدار الموارد وفقدان البيانات. وبالتالي، سيفقد مزودو الحوسبة السحابية مصداقيتهم. تقدم هذه الورقة نهجًا جديدًا فعالًا يوفر اكتشاف الأعطال، ويقلل من تأثير الخطأ، ويوفر تنبؤات دقيقة في غضون فترة زمنية كافية من أجل ضمان موثوقية وجودة الخدمة، وتقليل الهدر، وتحسين الدقة. لقد طُبِقَ النهج في منصة الحوسبة السحابية OpenStack لأجل حقن بيانات العطل والمعالجة المسبقة للبيانات. يستفيد النهج من خوارزميات التعلم الآلي من خلال تطبيق عدد من الخوارزميات لتحسين دقة التنبؤات. الخوارزميات التي تطرق إليها البحث هي مصنف المتجه الداعم (SVC)، والغابة العشوائية (RF)، و الجار الأقرب (KNN)، وشجرة القرار (DT)، والانحدار اللوجستي (LG). تشير النتائج إلى أن دقة التنبؤ بالأعطال بالنموذج المقترح باستخدام مصنف SVC دقيقة وفعالة بنسبة ٩٧٪ مقارنة بالخوارزميات الأخرى. تظهر هذه النتائج أن طريقتنا يمكن أن تتنبأ بشكل فعال بأعطال النظام والتطبيقات المستقبلية أيضاً.

الكلمات المفتاحية: الأعطال البرمجية، حقن العطل, خطأ، الحوسبة السحابية، السحابة، خوارزميات تعلم الآلة، المعالجة المسبقة للبيانات.

[*] أستاذ دكتور في قسم تكنولوجيا المعلومات –كلية هندسة تكنولوجيا المعلومات والاتصالات – جامعة طرطوس – طرطوس– سوريا.

[**] طالبة ماجستير في قسم تكنولوجيا المعلومات–كلية هندسة تكنولوجيا المعلومات والاتصالات– جامعة طرطوس– طرطوس– سوريا.

1- Introduction

A Cloud computing provides many services in our lives, such as healthcare, transportation, and other fields where high reliability is necessary. It also assembles many computing, storage, and network resources into a data center. To prevent service outages, cloud system designers need to know in advance how their software system behaves under errors that cause failure, as shown in (Fig.1), before deploying it in operation. Knowledge of fault and repair characteristics is valuable for designers in order to ensure the quality of service, reduce waste, and mitigate errors and failures.

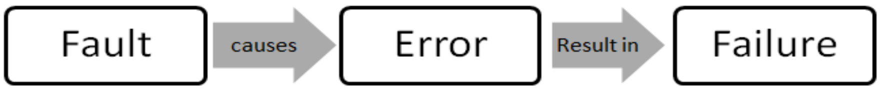


Fig.1. Occurrence of Failure [1]

To get data about software fault injection [2], i.e., (such as resource exhaustion, software bugs, connection loss, etc.) into a system, these experiments produce a large amount of fault data.

Unfortunately, fault analysis is difficult due to the size and complexity of fault data. Therefore, in this work, it was proposed a novel approach for efficiently identifying faults in cloud data. The approach leverages preprocessing data and supervised machine learning to overcome the challenges of noise and complexity in the feature space. Our approach applies feature engineering to the data to automatically transform the raw fault data into a compact set of features.

The proposed approach adopted software fault injection to accelerate the occurrence of errors caused by software bugs [٣]. The approach injects bugs into one of the system components and analyzes the reaction of the cloud system. The fault injection tool was based on information on software bugs reported by OpenStack developers.

Furthermore, the proposed approach was evaluated on a dataset from OpenStack [٤], a popular platform used in private and public cloud computing systems. As an additional, we compare the proposed approach with a fine-tuned classification technique. The results show that the proposed approach can identify classifications with high accuracy and a low computational cost.

Related Work:

Many studies have been applied to various predictions in cloud computing. From the perspective of research objectives. The researchers analyzed failures of Compute Cloud APIs by performing fault injection and analyzing the impact of the resulting failures in terms of fail-stop behavior, failure detection through logging, and failure propagation across components. They couldn’t find the failures that spread across several sub-systems before being notified and that cause persistent effects that are difficult to recover [٥]. The researchers presented a new approach for detecting failure in the environment of virtualized HPC systems and applications. The approach is based on time series and machine learning techniques. Because machine learning algorithms heavily rely on the training dataset, parameters, and features that are chosen, which require extensive network expertise, they cannot be said to be accurate. They also handle with a specific type of fault [٦]. The researchers proposed a fault detection method for multi-tier web applications in cloud computing based on the Fuzzy One-Class Support Vector Machine (FOCSVM). This problem is addressed through a three-step approach, where first proposed the fuzzy one-class support vector machine (FOCSVM) model to detect abnormal data based on the decisive boundary. Then, the exponentially weighted moving average chart, which monitors the decisive boundary value of the FOCSVM model, is applied to detect faults in multi-tier web applications. Finally, the Random Forest ranking feature algorithm is applied to determine if suspicious metrics are the root of faults [٧]. The researchers suggested a fault diagnosis method that can effectively identify and locate intermittent faults originating from (but not limited to) processors in the cloud computing environment. The method is end-to-end in that it does not rely on artificial feature extraction for applied scenarios, making it more generalizable than conventional neural network-based methods. It can be implemented with no additional fault detection mechanisms and is realized by software with almost zero hardware cost [٨].

2- Research Problem

With an increasing number of using application and platforms running on cloud system, it is challenging for the cloud providers to offer uninterrupted services with high Quality. Identifying the faults modes of cloud computing systems is a difficult and time-consuming task, due to the growing complexity of such systems, and the large volume and noisiness of failure data.

3- Research Objectives and Relevance

This research aims to provide an effective approach to detect Faults, mitigate cloud errors and the impact of failures, and provide accurate predictions within a sufficient period of time in order to ensure reliability and quality of service to reduce waste and improve accuracy.
The significance of this research lies in its application during runtime and in the usage of fault injection based on the results of advanced data Pre-processing, which aims to:

- Cleaning data and extracting new features.
- Find the correlation between features and the target label to determine the injected faults.
- maintain a good classification.

The extraction of new features from data to find a stronger correlation with the target column, which increases the accuracy of identifying the faults that will be injected into the database and reduces the time required for training and reaching the required classification accuracy

4- Research Methods and Materials:

This research was implemented using the Python programming language, which is one of the most popular languages in the field of artificial intelligence and machine learning and contains many software libraries that facilitate work in this field. Google Colab has been selected as a work environment—which is one of the most important cloud services that transfer calculations and storage space to the cloud [9]. offered by Google — that allows work within Jupyter Notebook without any installation of software libraries and language requirements on the local device. Some language libraries, such as Numpy, have been used to handle matrices, Matplotlib to draw charts, Pandas to handle tables in the dataset, and Sklearn because they contain algorithms that help in machine learning.

Therefore, using Google Colab helps enhance the module's abilities to detect faults in real time and guarantee accurate results. In order to maximize their benefits, we also focused on developing a module that combines high preprocessing data with machine learning techniques for improved data visualization. In addition, it ensures high accuracy and mitigates the impact of the error.

5- FAULT DETECTION APPROACH:

The proposed approach is slightly different from previous research in many ways, like applying a new strategy to detect faults at runtime by injecting faults like software faults while VMs are working, using Artificial Intelligence techniques with real-time applicability. The previous researches haven't focused on this part. All of them focus on failure without considering the previous step's error or fault.

This approach, as shown in (Fig.2), contains several steps that begin with importing data, then processing and preparing it, and end with the use of several classification algorithms in training, testing, comparing results, and evaluating them.

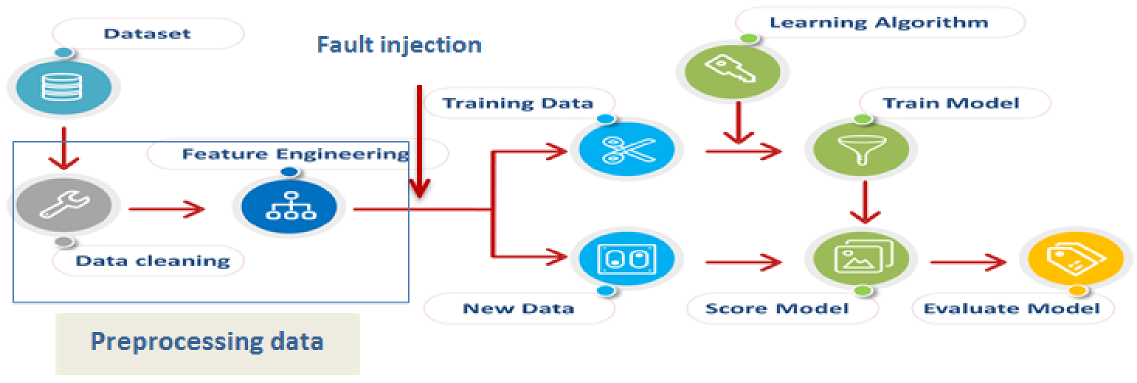


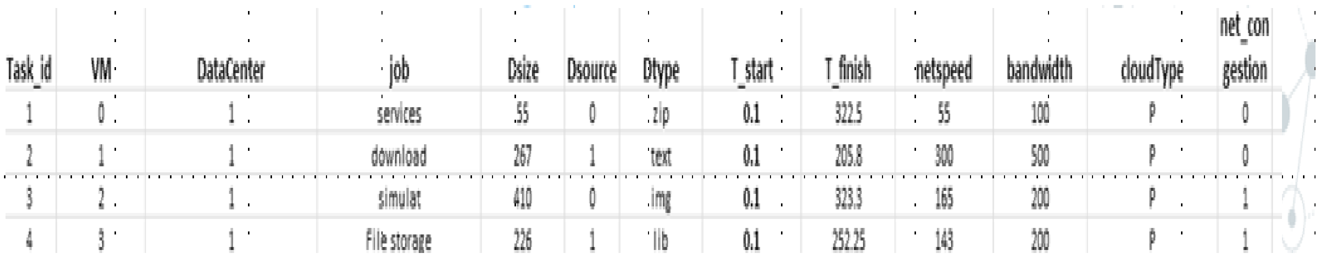
Fig.2: Diagram of the proposed System Model

Steps of building the Proposed System Model:

5-1 Dataset:

The module relies on getting data as a first step. Therefore, we use the OpenStack platform, which provides a cloud computing system consisting of processes distributed across a data center, where each data center includes more than one host, and the number changes according to what is simulated and according to the tasks required. All data center

information is stored within the cloud information system (CIS), and the data is restored from it by using "broker" ,and without it, the simulation process is not done, and it's up to the user to decide which data center (DC), which host, and which virtual machine (VM) will do the task.
We have the dataset from the OpenStack cloud computing platform. So we imported it to see the features. Each row describes a task's details, such as its ID number, time of execution, etc. Then move on to the next step to analyze the data and get more information about it (Fig. 3).



Task_id	VM	DataCenter	job	Dsize	Dsource	Dtype	T_start	T_finish	netspeed	bandwidth	cloudType	net_congestion
1	0	1	services	.55	0	.zip	0.1	322.5	55	100	P	0
2	1	1	download	267	1	text	0.1	205.8	300	500	P	0
3	2	1	simulat	410	0	.img	0.1	323.3	165	200	P	1
4	3	1	File storage	226	1	lib	0.1	252.25	143	200	P	1

Fig.3. Some Features of the Database

5-2 Preprocessing data:

Initially, this step involves analyzing the data and knowing its distribution using histogram schemes. The distribution of some features in Database as shown in (Fig.4).

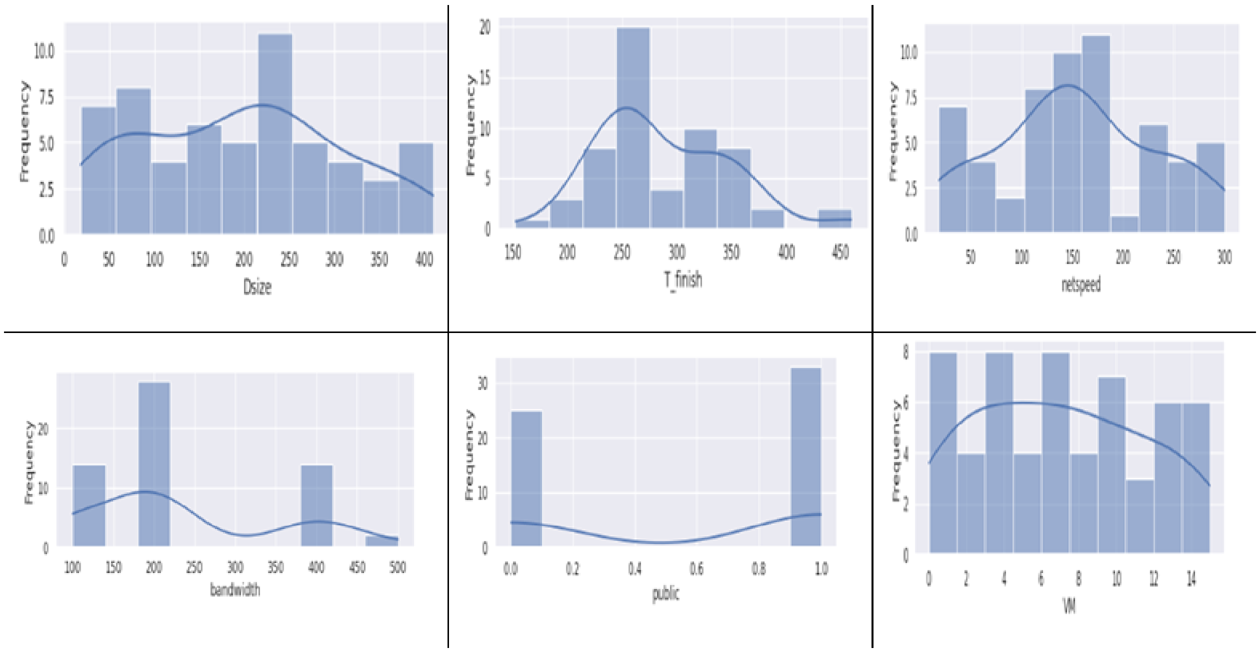
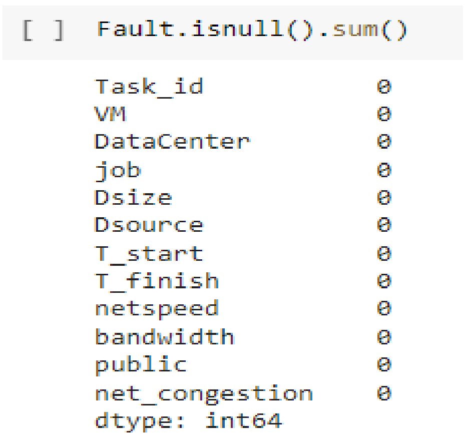


Fig. 4. Data Distribution

This schemes help us to know if there is an outlier values in the dataset.

Then we applied advanced processing to the data because it had some repeating values and missing values; as a result, we cleaned the data to remove anomalous and extreme values that negatively affect the model using attribute mean to deal with missing values in the data and deleted all the repeated rows. The cleaned data was then validated using this Python script, as shown in (Fig. 5):



```
[ ]  Fault.isnull().sum()

     Task_id           0
     VM                0
     DataCenter        0
     job               0
     Dsize             0
     Dsource           0
     T_start           0
     T_finish          0
     netspeed          0
     bandwidth         0
     public            0
     net_congestion    0
     dtype: int64
```

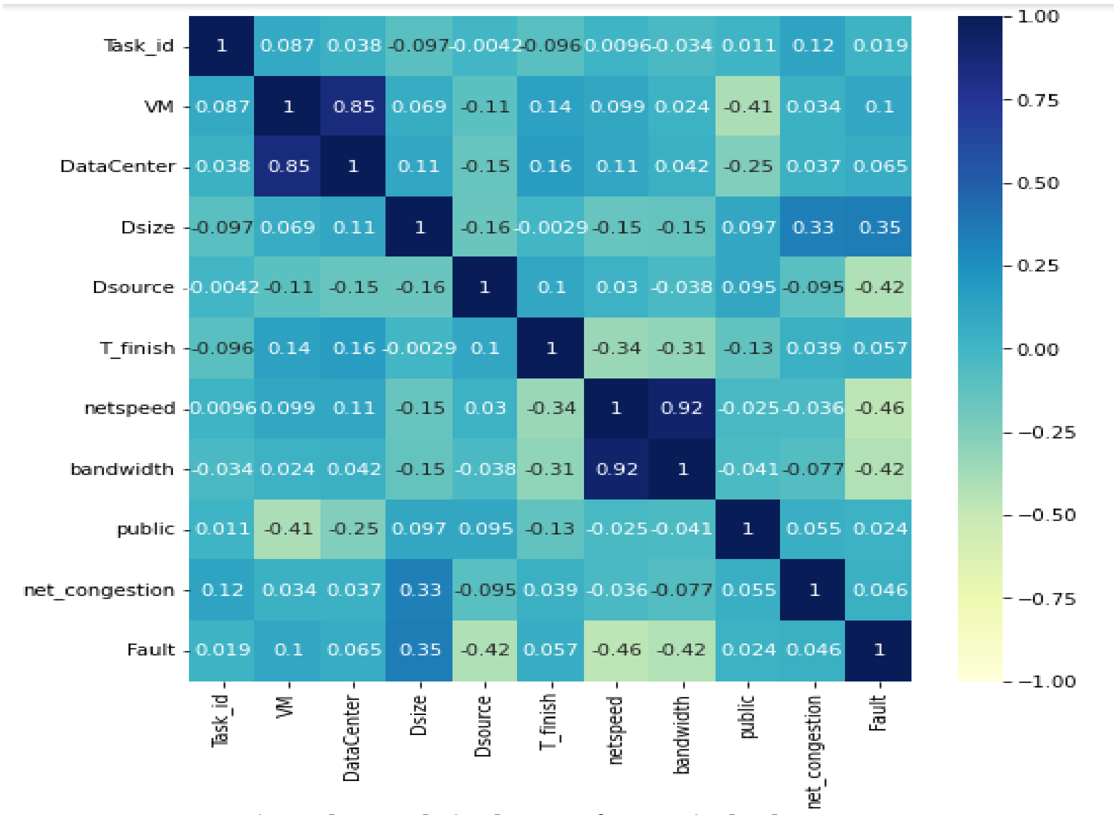
Fig.5. Cleaned the data.

Data transformation was applied because the data contained text values by using the Pandas library of Python libraries to digitize textual information. For example, the data contains text features like the job column. By applying the instruction (panadas.get_dummies ( )), each value of this feature was replaced by a serial number.

Extraction new features from existing data, such as congestion based on Dsize, net_speed, and bandwidth, as well as total time between T_start and T_finish. After that, we examined the data to determine how the data relate to each other and to the target (whether the task is fault or not), as shown in (Fig. 6).

Correlation explains how one or more variables are related to each other. These variables are input data features that have been used to forecast our target variable [10]. It gives us an idea of the degree of the relationship between the two variables. It's a bivariate analysis measure that describes the association between different variables.

Fig.6, Show the degree of relationship between features; the closer the value is to one, it means that they have a significant impact on each other. The correlation value is between -1 and +1. For example, the correlation between "net_speed" and "bandwidth" features is 0.92; it means they are related to each other and any change on one of them will effect on the other one. We get it by applying a Python script to Google Colab.



	Task_id	VM	DataCenter	Dsize	Dsource	T_finish	netspeed	bandwidth	public	net_congestion	Fault
Task_id	1	0.087	0.038	-0.097	-0.0042	-0.096	0.0096	-0.034	0.011	0.12	0.019
VM	0.087	1	0.85	0.069	-0.11	0.14	0.099	0.024	-0.41	0.034	0.1
DataCenter	0.038	0.85	1	0.11	-0.15	0.16	0.11	0.042	-0.25	0.037	0.065
Dsize	-0.097	0.069	0.11	1	-0.16	-0.0029	-0.15	-0.15	0.097	0.33	0.35
Dsource	-0.0042	-0.11	-0.15	-0.16	1	0.1	0.03	-0.038	0.095	-0.095	-0.42
T_finish	-0.096	0.14	0.16	-0.0029	0.1	1	-0.34	-0.31	-0.13	0.039	0.057
netspeed	0.0096	0.099	0.11	-0.15	0.03	-0.34	1	0.92	-0.025	-0.036	-0.46
bandwidth	-0.034	0.024	0.042	-0.15	-0.038	-0.31	0.92	1	-0.041	-0.077	-0.42
public	0.011	-0.41	-0.25	0.097	0.095	-0.13	-0.025	-0.041	1	0.055	0.024
net_congestion	0.12	0.034	0.037	0.33	-0.095	0.039	-0.036	-0.077	0.055	1	0.046
Fault	0.019	0.1	0.065	0.35	-0.42	0.057	-0.46	-0.42	0.024	0.046	1

Fig.6. The correlation between features in database.

Based on this, we can decide what faults we want to inject into the database.

5-3 Fault injection:

Injection using a software fault injection tool [11]. It deliberately injects faults into a system to determine whether the system can withstand error conditions. It is a practice of injecting faults, observing how the system responds to the events, and implementing improvements. With this tool, we have avoided the mistake of most human analysts and researchers, which is hand injection. The analysis (preprocessing step) above is allowed to identify the faults we should inject. The faults are:

- Wrong return value: This method returns an incorrect value. In particular, the returned value is corrupted according to its data type (e.g., we replace an object reference with a null reference or replace an integer positive value with a negative one).
- Wrong parameter value: The target method is called with an incorrect input parameter. Input parameters are corrupted according to the data type, as for the previous fault type.
- Delay: The target method returns the result after a long delay. This fault can trigger timeout mechanisms inside the system or cause a stall.
- Missing parameters: A method call was invoked with omitted parameters (e.g., the method used a default parameter instead of the correct one).

After that we get the new database as an Excel file.

5-4 Model build and training:

All the previous steps built our module, and the data is ready to train our model. The dataset is divided into 80% for training and 20% for validation in order to achieve a more concrete estimate of the accuracy of the best model.
As a last step, the algorithms were fitted into our model, then compared and selected the best out of all of them to choose it and apply fine tune to improve its accuracy. The Machine Learning Algorithms we considered are [12]:

- **Support Vector Classification (SVC) algorithm:**

SVR is a model of SVM algorithm, where SVM is one of the types of supervised algorithms which can be used for solving classification problems and regression. Modifications can be applied on SVM algorithm. Using the modified version of SVM algorithm, regression problems can be solved. This can be done by exploiting the hyper plane as the function to be estimated with adding tolerance range.

- **Decision Tree (DT) algorithm:**

DT is a supervised learning technique. In DT There are two nodes, which are the Decision Node and Leaf Node**.** In order to build a tree, we use the Classification and Regression Trees CART algorithm, which is a binary decision tree. Decision Tree building algorithm involves a few simple steps which are as follows: 1.Take labeled input data with a target variable and a list of Independent Variables. 2. Best Split: Find the best split for each of the independent variables. 3. Best variable: Select the best variable for the split. 4. Split the input data into left and right nodes. 5. Continue step (2-4) on each of the nodes until it meets the stopping criteria. 6. Decision tree pruning: Steps to prune decision tree built [13].

- **Random Forest (RF) algorithm:**

The RF algorithm is a collection of multiple random decision trees, and it's much less sensitive to the training data. The process to create an RF is as follows: The first step is to build new data from the original data. Every data set will have the same number of rows as the original one because rows are randomly selected from the original data. The method we used to generate new data is known as bootstrapping. Then, train the decision tree on each of the bootstrapped datasets independently. For prediction, pass the data points through each tree one by one and note down the prediction, then combine all the predictions. This process of combining results from multiple models is called aggregation. For the classification problem, take the majority vote as the final result [14].

- **Logistic Regression (LG) algorithm:**

LG (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class. If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does not (i.e., it belongs to the negative class, labeled “0”). This makes it a binary classifier. Just like a Linear Regression model, a Logistic Regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the logistic of this result.

- **K Neighbors Classifier (KNN) algorithm:**

KNN is a simple supervised machine learning method that commonly used for classification problems. It classifies a data point based on classification of its neighbors. KNN stories all variable cases and categories new cases using a similarly metric. The steps of KNN algorithm are as follow:

1. Choose a value for the K number.
2. Determinate the Euclidean distance between the new data and the entire set of data.
3. Using the estimate Euclidean distance, find the K nearest neighbors to the new data.
4. Allocate the new data points to that category with the greatest number of the neighbor.

Distance function Euclidean is shown in equation (1):

$$\text{Euclidean} = \sqrt{(\boldsymbol{xi} - \boldsymbol{yi})^2} \qquad (1)$$

The model analyses several performance metrics to choose the best classification algorithm, as shown in the Experimental Evaluation section. The last step was to fine-tune the module to improve our chosen module.

5-5 Fine tune the: a few ways you can do but we chose what is suit our work [15]:

Grid Search: this method depends on modifying hyperparameters manually until finding a suitable combination of hyperparameter values. This can be done by using GridSearchCV from Scikit-Learn should perform the search for you. You only need to know which hyperparameters you want the experiment to utilize. Several parameters must be tuned to increase the precision of the model. There are three important parameters to consider:

a) Kernels: The principal function changes a low-dimensional input space into a higher-dimensional space.
b) C (Regularization): this parameter denotes misclassification or an error word. The misclassification or error term informs the algorithm optimization on how much inaccuracy can be tolerated. This is how the trade-off between the decision boundary and the misclassification term can be managed.
c) Gamma: It specifies how far the calculation of the feasible line of separation is influenced. When gamma is high, adjacent points have a large influence; when gamma is low, faraway points are also examined to determine the decision border.

6- Research results and Discussion

a. Performance parameters:

Before proceeding to discuss the results, we will highlight several parameters (performance evaluation parameters), which are inferred by the confusion matrix [16] as shown in Table 1:

TABLE 1. Confusion Matrix for two classes.

Actual Values	Predict Value	
	Normal	Fault
Normal	True Positives (TP)	False Positives (FP)
Fault	False Negatives (FN)	True Negatives (TN)

Performance measures are:

- Accuracy: measures how often the model predicts correctly as shown in equation (2):

$$\textbf{Accuracy} = \frac{\text{Number of correct tasks}}{\text{Number of all tasks}} = \frac{TN+TP}{TN+TP+FN+FP} \qquad (2)$$

- F1-score: gives a better measure of incorrectly predicted labels ,as shown in equation (3):

$$\textbf{F1-Score}= 2* \frac{Precision*Recall}{Precision+Recall} \qquad (3)$$

- Sensitivity: also known as Recall or True Positive Rate (TPR), measures the proportion of data points that were correctly predicted as Faults , as shown in equation (4):

$$\textbf{Sensitivity} = \frac{\text{Number of true positive tasks}}{\text{Number of all negative tasks}} = \frac{TP}{TP+FN} \qquad (4)$$

- Precision: is a measure of the accuracy of positive predictions or fault identification , as shown in equation (5):

$$\textbf{Precision} = \frac{\text{Number of true positive tasks}}{\text{Number of true and false positive tasks}} = \frac{TP}{TP+FP} \qquad (5)$$

b. Experimental Evaluation:

First we imported the dataset by using Google Colab and python language to analysis the data. Also applied all the previous steps as shown in section 5, which built our module. To check the efficiency of the proposed module, we applied the most common supervise machine learning algorithms and estimated their prediction accuracy. In other to achieve more concrete estimate of the accuracy of the best model, we first split the loaded dataset into two, 80 % to train the module and 20% to validate dataset because When learning a dependence from data, to avoid overfitting, it is important to divide the data into the training set and the testing set. We first train our model on the training set, and then we use the data from the testing set to gauge the accuracy of the resulting model. Empirical studies show that the best results are obtained if we use 20-30% of the data for testing, and the remaining 70-80% of the data for training [17]. The following steps were followed:

1. Set-up the test harness to use cross validation to estimate accuracy.
2. Apply the five different algorithms to predict the component fault pattern from the fault data.
3. Furthermore, the dataset is being split into 10 parts, train in 8 and test on 2 and release for all combinations of train-test splits. We used the Performance parameters: Recall, Precision, Specificity, and Accuracy to evaluate our proposed model.

We applied the algorithms using python languages and libraries. to evaluate our developed models we need to compare the models to each other and select the most accurate. Then reset the random number seed before reach run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. This ensures the results are directly comparable. Here we compare the five models and their respective accuracy estimations.

In this section, after training and testing models, we evaluate the five models by contrasting their accuracy predictions.

The results of experiment 1 from Table 2 and (Fig.6) shows that the SVC has the highest prediction accuracy (0.928) and sensitivity **(0.75).** Because of that, we chose the SVC module to be improved and to get better accuracy by applying fine tuning to our proposed module.

TABLE 2. The performance measure of the MLAs algorithms.

ML Algorithms	**Accuracy**	**F1_score**	**Precision**	**Sensitivity**
SVC	**0.928**	**0.80**	**0.78**	**0.75**
DT	0.73	0.65	0.576	0.63
LR	0.857	0.74	0.68	0.71
KNN	0.81	0.77	0.571	0.66
RF	0.75	0.59	0.72	0.65

Fig.6, The figure represents the values of the table as a chart to illustrate the difference between the values. It displays the evaluation matrices of models for the five algorithms.

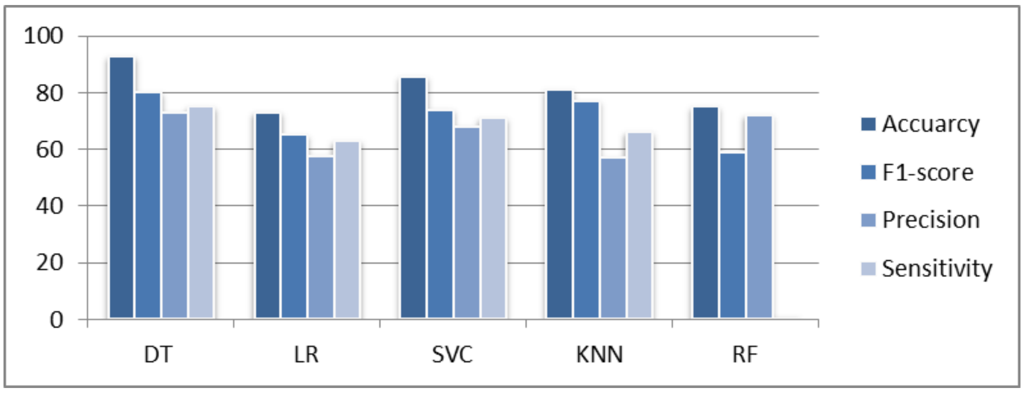
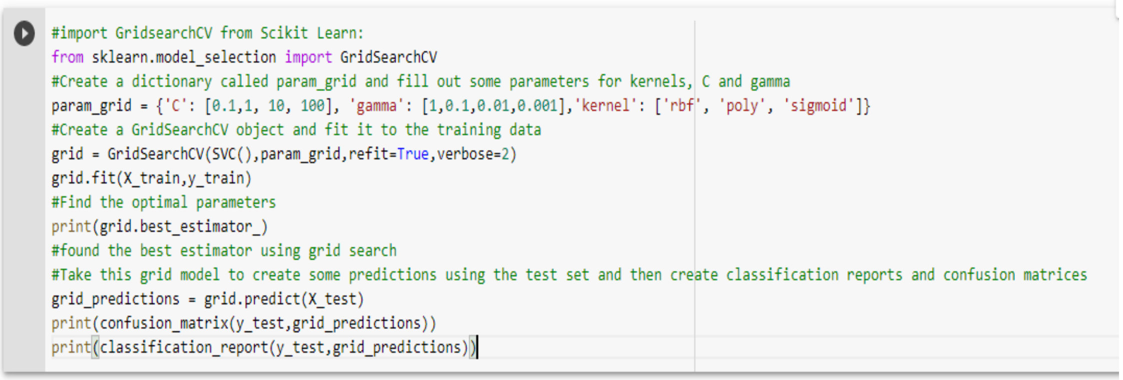


Fig.6: Experiment – 1.Results: Using Evaluation Metrics models for all the experiments

The most widely used library for implementing machine learning algorithms in Python is scikit-learn. The class used for SVC classification in Scikit-Learn is svm.SVC().
sklearn.svm.SVC (C=1.0, kernel='rbf', degree=3, gamma='auto')
Python script on google colab as shown in (Fig.7):



```
#import GridsearchCV from Scikit Learn:
from sklearn.model_selection import GridSearchCV
#Create a dictionary called param_grid and fill out some parameters for kernels, C and gamma
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}
#Create a GridSearchCV object and fit it to the training data
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
#Find the optimal parameters
print(grid.best_estimator_)
#found the best estimator using grid search
#Take this grid model to create some predictions using the test set and then create classification reports and confusion matrices
grid_predictions = grid.predict(X_test)
print(confusion_matrix(y_test,grid_predictions))
print(classification_report(y_test,grid_predictions))
```

Fig.7. GridSearchCV Python script.

After applying fine tune on the selected model in Google Colab as a second scenario we get the result in Table 3.

TABLE 3. Compare of SVCs algorithm after fine tune model.

ML Algorithms	**Accuracy**	**F1_score**	**Precision**	**Sensitivity**
SVC1(without)	0.928	0.80	0.78	0.75
SVC2 (with)	0.974	0.85	0.83	0.87

As per the results in Table 3 for Experiment 2, 'SVC2 (with)'has the higher accuracy of 97.4% than 'SVC2 (without)'.

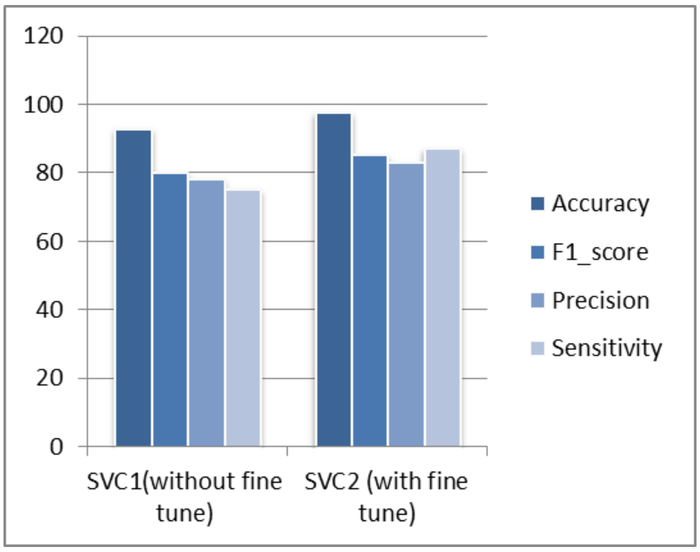


Fig.8. Experiment – 2.Results: Using Evaluation Metrics between two algorithms.

The results For experiment-2 from Table 2 and (Fig.8), it is apparent that the optimal model is SVC2 (with fine-tuning). Consequently, the results demonstrate that our SVC-based model is sufficiently accurate to predict system faults. In addition, SVC-based models provide more accurate predictions than other models. Hence, the results clearly validate that our model using SVC has accurate enough to predict system faults.

ROC curve plotted using Python language. It is also beneficial to plot the first ROC curve to understand how the SVC1 and SVC2 compare (Fig.9).

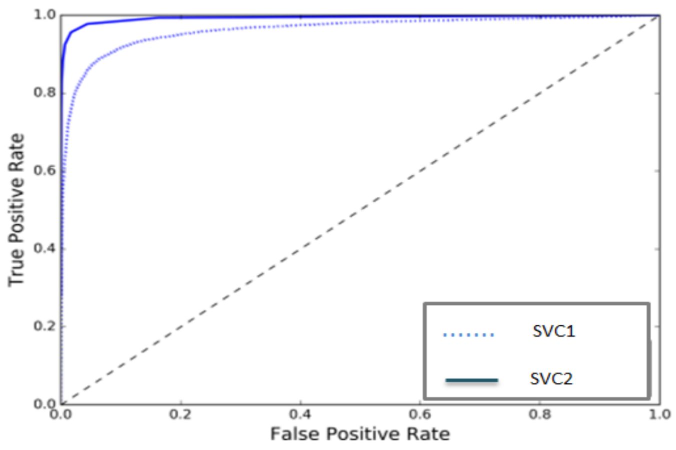


Fig.9. Experiment – 2.Results: using Comparing ROC curves.

Experiment-2 as shown in (Fig.9) shows that the SVC2 ROC curve is significantly closer to the top-left corner. Because of that, the ROC accuracy of SVC2 score is much higher than SVC1's accuracy, F1_score, precision, and sensitivity.

7- Conclusion and Future Study

The proposed module has many advantages, manifested in detecting faults and mitigating their negative impact. The detection happens in real time by implementing a Python script and Google Colab. As well, the approach has been designed to be applied without any a priori information about the types of features in the fault data in order to minimize the manual effort. Besides that, the tested results are highly accurate, and the whole proposed module helps enhance the rate of true positives and reduce the rate of false positives compared to previous studies. This module is not resource-consuming, does not require high hardware requirements to keep computational times acceptable, and is easy to implement in any desired environment. Moreover, this approach can be used as a first step to detect or predict errors that result in failure in cloud systems.

8- References

[1] GILL, S,S; Buyya, R. (2020), *Failure Management for Reliable Cloud Computing*. Computing in Science & Engineering , vol.22, issue.3, pp. 52 – 63.

[2] Lenka, R, K; M, C; Padhi, S, Nayak, K, M. (2018), *Fault injection techniques*. IEEE, vol.30, no.4, pp.75-82.

[٣] Natella, R; Cotroneo, D; Madeira, H, S. (2016), *Assessing dependability with software fault injection: A survey*. ACM Computing Surveys (CSUR), vol.48,issue.3, no.3, pp.1-55.

[٤] OpenStack, Accessed: Nov. 18, 2022. [Online]. URL: http://www.openstack.org/

[٥] Cotroneo, D; Simone, L; Liguori, P; Natella, R; Bidokhti, N. (2019), *How bad can a bug get? an empirical analysis of software failures in the OpenStack cloud computing platform*, in: Proc. ESEC/FSE, ACM , pp. 200– 211.

[٦] Mohammed, B ; Awan, I ; Ugail, H; Younas, M. (2019), *Failure Prediction using Machine Learning in a Virtualised HPC System and application*, Cluster Computing, vol.22, pp.471–485.

[٧] Bui, K, T; Vo, L, V; Nguyen , C, M; Pham, T, V; Tran, H, C. (2020), *A Faults Detection and Diagnosis Approach for Multi-tier Application in cloud computing*. KICS Korean Institute of Communications and Information Sciences, vol.22, issue.5.

[٨] Wang, C; Fu, Z; Huo, Y. (2021), *End-To-End Diagnosis of cloud system Against Intermittent Faults*. Computer Science and Information Systems, vol.18, issue.3, pp.771-790.

[9] Google Colab Platform . Accessed: Feb. 25, 2022. [Online] .URL https://colab.research.google.com/#scrollTo=UdRyKR44dcNI.

[10] What is Correlation In Machine Learning. Accessed: sept. 15, 2022. [Online]. URL: https://medium.com/analytics-vidhya/what-is-correlation-4fe0c6fbed47#:~:text=Correlation%20explains%20how%20one%20or,relation%20with%20the%20other%20variable.

[11] Cotroneo, D; Simone, L, D; Liguori, P; Natella, R. (2020) ,*Profipy: Programmable software fault injection as-a-service*. in: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 364–372.

[12] Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, Second Edition.

[13] Saif, W, S; Esmail, M, A; Ragheb, A, M; Alshawi, T, A ; Alshebeili, S,A. (2020), *Machine Learning Techniques for Optical Performance Monitoring and Modulation Format Identification: A Survey*. in IEEE ÓPTICA PURA Y APLICADA www.sedoptica.es Opt. Sociedad Española de Óptica Communications Surveys & Tutorials, vol. 22, no. 4, pp. 2839-2882.

[14] Random Forest Algorithm For Machine Learning. Accessed: Feb. 12, 2023. [Online] .URL: https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb.

[15] SVM Hyperparameter Tuning Using GridSearchCV. Accessed: Apr. 27, 2023. [Online] .URL: https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/

[16] Understanding Confusion Matrix. Accessed: Jan. 26, 2023. [Online]. URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[17] Gholamy, A; Kreinovich, V; and Kosheleva, O. (2018) .*Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation*. Departmental Technical Reports (CS). 1209. https://scholarworks.utep.edu/cs_techrep/1209