An Exotic IWD - SVR Based Approach for Failure Prognostication in Cloud-Based Scientific Workflows

Sridevi S (✉ srideviresearch2017@gmail.com)

Velammal Engineering College

Jeevaa Katiravan Jeevaa Katiravan

Velammal Engineering College

Research Article

Keywords: Failure prognostication, Intelligent Water Drops, Machine learning, Support Vector Regression, Proactive fault tolerance

Posted Date: August 2nd, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-716843/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

An Exotic IWD - SVR based Approach for Failure Prognostication in Cloud-based Scientific Workflows

Sridevi S¹ and Jeevaa Katiravan²

^{1,2}Department of Computer Science and Engineering, Velammal Engineering College, Chennai, INDIA

srideviresearch2017@gmail.com¹

jeevaakatir@gmail.com²

Abstract. Scientific workflows deserve the emerging attention in sophisticated large-scale scientific problem-solving environments. Though a single task failure occurs in workflow based applications, due to its task dependency nature the reliability of the overall system will be affected drastically. Hence rather than reactive fault tolerant approaches, proactive measures are vital in scientific workflows. This work puts forth an attempt to concentrate on the exploration issue of structuring an Exotic Intelligent Water Drops - Support Vector Regression-based approach for task failure prognostication which facilitates proactive fault tolerance in scientific workflow applications. The failure prediction models in this study have been implemented through SVR-based machine learning approaches and its precision accuracy is optimized by IWDA and various performance metrics were evaluated. The experimental results prove that the proposed approach performs better compared with the other existing techniques.

Keywords: Failure prognostication, Intelligent Water Drops, Machine learning, Support Vector Regression, Proactive fault tolerance.

1. Introduction

Cloud is the buzzword in the computational technologies that has brought a paradigm shift in the way data is stored and computing is performed. Cloud computing is a subscription-based service that delivers computation as a utility. The key characteristics of cloud computing are providing elastic, on-demand delivery of services with dynamically configurable resources and innovative pricing models. In the past decade the concept of scientific workflow has emerged as a booming paradigm for modeling large scale complex data in diverse computing domains. Scientific workflows are abstractions composed of activities and data with intricate dependencies managed by complex engines.

Fault tolerance is one of the critical factors to guarantee reliability of the cloud services. Fault tolerance is the ability of the system to keep working in presence of one or more faults but with decaying performance. Fault tolerance strategies in cloud are classified as two types: reactive and proactive. Reactive fault tolerance strategies are the techniques used to effectively troubleshoot a system upon occurrence of failure(s). Various reactive fault tolerance strategies are: Replication, Task resubmission, checkpointing and so on. The execution of such techniques automatically results in the performance degradation of the system. Proactive fault tolerance makes use of a prediction approach to anticipate the failures in advance thereby reducing the downtime of the system.

Failure prediction influences the delivery of on-demand services in the cloud. Accurate predictions of failure prone machines can help in mitigating the impact of failures in a proactive manner. Hence this work emphasis on a mathematical prediction-based approach to accurately anticipate the failure prone hosts in the cloud resource pool which facilitates the early migration of the virtual machines to other active hosts with no performance degradation.

2. Literature Review

2.1 Machine Learning Approaches for Failure prediction

Cagatay Catal (2011) suggested Naive Bayes as the robust machine learning algorithm for supervised software fault prediction [1]. Malhotra and Ankita Jain (2012) observed that the random forest and bagging gave the best results for fault predictions [2]. But they did not consider the effect of size, resource related issues on fault proneness and its severity. Sadeka Islam, et. al (2012) proved that the prediction accuracy of Error Correction Neural Network (ECNN) demonstrated superior effectiveness for forecasting resource utilization in the cloud [3].

Ilenia Fronza, et.al (2013) introduced a new approach for predicting failures based on Support Vector Machines (SVM) and Random Indexing (RI) [4]. The results of their work proved that weighted SVMs perform better in improving sensitivity. This approach proved reliability in predicting both failures and non-failures. Anju Bala and Inderveer Chana (2015) have designed intelligent task failure prediction models that have been implemented through machine learning approaches like ANN, Naive Bayes, Random Forest, and LR using evaluated performance metrics [5]. The approaches proposed in this paper only predicts whether the task fails or not but does not determine the rate of failure of each task.

Upasna Kothari and Moe Momayez (2018) proposed the use of machine learning to predict the time of failure [6]. The results of the study proved that Machine Learning provided prediction values that are 86% of the time closer to the actual time of failure when compared to the traditional methods.

P.Padmakumari, et.al [7] used the ensemble with the combination of machine learning methods in workflow environment to improve accuracy and efficiency of the prediction model. Till now no work has been proposed using the combination of SVR and IWD in the failure prediction of scientific workflows.

2.2 Nature Inspired Support Vector Regression

Xiang – Ming, et.al. [8] (2017) constructed a PV power prediction model based on EMD and ABC-SVM and proved that the method is superior to other approaches. The model proposed by Jui-Sheng Chou, et.al. [9] (2015) is a hybrid of LS-SVR and SFA in which SFA integrates an artificial firefly colony algorithm with chaotic map, adaptive inertia weight, and Levy flight. It uses SFA to optimize LS-SVR hyper parameters (i.e., regularization parameter and sigma parameter) and then uses LS-SVR for prediction.

3. Approaches Used for Task Failure Prognostication

3.1 Support Vector Machines for Regression

The concept of Support Vector Machine (SVM) [10] was first introduced by Boser, Guyon and Vapnik. SVM is a supervised machine learning algorithm based on statistical learning theory to fix both non-linear classification and regression challenges. It has great performance since it can handle a non-linear classification efficiently by mapping samples from low dimensional input space to high dimensional feature space with a non-linear kernel function. The key parameter of SVM is the type of kernel function used. Kernel functions are used for mapping into a higher dimensional feature space. Mercer's theorem is used for the construction of positive definite kernel for SVM regressor.

Support Vector Regression (SVR) has been applied to many actual issues like predicting the performance of compact heat exchangers (Peng and Ling, 2015) [11], modeling of heat transfer in a thermo syphon reboiler (Zaidi, 2015) [12], predicting the sorption capacity of lead (II) ions (Nusrat Parveen et. al, 2016) [13], predicting the price in a car leasing application (Mariana Listiani, 2009) [14], modeling and predicting Turkey's electricity consumption (Kadir Kavaklioglu, 2010) [15], analyzing prognosis of infants with congenital muscular torticollis (Suk-Tae Seo, 2010) [16], and so on.

The architecture of Support Vector Regression is given as follows:

Fig.1: Architecture of a regression machine constructed by the support vector algorithm [10]

3.1.1 Selection of Kernel Functions and Parameters

The key of SVR is the selection of kernel functions and model parameters. They have a great influence on the prediction accuracy of the SVR. Some of the kernel functions are

<i>Type of Kernel</i>	<i>Formula</i>
Linear kernel	$k(x, y) = x^T y + c$
Polynomial kernel	$k(x, y) = (\alpha x^T y + c)^d$
Gaussian kernel	$k(x, y) = \exp(-\gamma x - y ^2)$
Laplace Radial Basis Function	$k(x, y) = \exp(- x - y /\sigma)$
Sigmoid kernel	$k(x, y) = \tanh(\alpha x^T y + c)$
Anova Radial Basis	$k(x, y) = \sum_{k=1}^n \exp(-\sigma(x^k - y^k)^2)^d$
Gaussian RBF	$k(x, y) = e^{-\frac{ x-y ^2}{2\sigma^2}}$

Table 1: Types of kernel functions for SVR

3.2 Intelligent Water Drops Algorithm for Optimization

The behavior of natural resources acts as a rich source of inspiration for us in tackling numerous real life optimization problems. Intelligent Water Drops Algorithm (IWDA) [\[17\]](#) is one among them in which the flow of water drops along the twist-and-turn path of the natural rivers to determine an optimal path towards its destination helps us to implement it in the form of an algorithm to solve optimization issues. The IWD algorithm is a population based constructive optimization algorithm whose idea depends on the responses of the natural water drops in rivers.

In the IWD algorithm, each intelligent water drop is created with two properties: velocity of the water drop and amount of soil each water drop carries. These properties commence with an initial value and change during the flow of an IWD from a source to its destination. The trip of an IWD begins with an initial velocity and zero soil.

During its journey, the IWD flows in discrete steps from its current location to its next location. Hence the velocity of the IWD increases non-linearly proportional to the inverse of the amount of soil between the two locations. Therefore, a path with less soil lets the IWD move with greater velocity than a path with more soil. An IWD gathers soil during its journey which is non-linearly proportional to the inverse of the time needed to travel from the current location to the next. The time taken is proportional to the velocity of the IWD.

The IWD uses a mechanism to select its path to the next location. According to this mechanism, it prefers to travel in the path with low soil so that it can move with greater velocities. The same mechanism can be adopted to find an optimal solution in several real-time problems. Some of the popular applications of IWDA are Vehicle routing problem, Multiple Knapsack problem, Job Shop scheduling, Travelling Salesman problem and so on.

The pseudo-code of an IWD-based algorithm might be indicated in eight stages:

[Step 1] Initialization of static and dynamic parameters
[Step 2] Random distribution of Intelligent Water Drops (IWDs) along the graph nodes and edges.
[Step 3] Updation of visited node list of each IWD.
[Step 4] Select the next node.
[Step 5] Local soil updation and velocity updation for each moving IWD.
[Step 6] Finding the local-best solution and update the soil on its path.
[Step 7] Global soil updation
[Step 8] Finding the Total-best solution and continue the process until all water drops completed.

Fig.2: Pseudocode for Intelligent Water Drops Algorithm

4. The Proposed IWD – SVR Approach

Fig.3: The proposed IWD-SVR approach

The proposed model seeks to adopt intelligent water drops algorithm to optimize the support vector regression parameter which involves the following steps:

- [Step 1] Let the input dataset be T . The sample size of dataset be n . Divide the dataset into training set and test set, the sample size for training set is k and the sample size for testing set is $n - k$.
- [Step 2] Select the kernel function type of SVR as Gaussian Radial Basis Function kernel function.
- [Step 3] Set initial global optimal value $T^{tB} = \min(\text{mid}(\text{SVR penalty factor}), \text{Gaussian RBF})$
- [Step 4] If MSE of anticipation function meets accuracy requirement, goto step 10; if do not meet, perform step 5, carry out SVR parameter adjustment with IWD algorithm.
- [Step 5] Set $N_{iB} = N_1 + N_2$ where N_1 = position of the number of significant digit of SVR penalty factor and N_2 = position of the width of Gaussian RBF.
- [Step 6] Generate parameter adjustment path profile C and σ using IWDA.
- [Step 6.1] Initialize static parameters
 Water Droplet Quantity = N ;
 Velocity updating parameters ($a_v = 1, b_v = 0.01, c_v = 1$);
 Soil updating parameters ($a_s = 1, b_s = 0.01, c_s = 1$)
 initial velocity = 0, initial soil = 100
- [Step 6.2] Initialize dynamic parameters :
 Soil^{IWD}: Soil of IWD
 Vel^{IWD}: Velocity of IWD
 Velc: Visited node list
 Edgesoil(i, j) : Soil along the edge, while flowing from i^{th} node to j^{th} node.
- [Step 6.3] Update the visited node list
- [Step 6.4] Update the dynamic parameters of IWD as
- $$\text{vel}^{\text{IWD}}(t + 1) = \text{vel}^{\text{IWD}}(t) + \frac{a_v}{b_v + c_v * \text{soil}^2(i, j)}$$
- $$\text{soil}^{\text{IWD}} = \text{soil}^{\text{IWD}} + \Delta \text{soil}(i, j)$$
- $$\Delta \text{soil}(i, j) = \frac{a_s}{b_s + c_s * \text{time}(i, j : \text{vel}^{\text{IWD}}(t + 1))}$$
- So that
- $$\text{time}(i, j : \text{vel}^{\text{IWD}}(t + 1)) = \frac{\text{HUD}(i, j)}{\text{vel}^{\text{IWD}}(t + 1)}$$
- Where HUD(j) is the heuristic undesirability defined for the problem. The soil along the path is updated using
- $$\text{soil}(i, j) = (1 - \rho_n) \cdot \text{soil}(i, j) - \rho_n \cdot \Delta \text{soil}(i, j)$$
- [Step 6.5] Update the soil along the path based on the current iteration best solution.

$$\text{soil}(i, j) = (1 + \rho_{\text{IWD}}) \cdot \text{soil}(i, j) - \rho_{\text{IWD}} \cdot \frac{1}{(N_{\text{IB}} - 1)} \text{soil}^{\text{IWD}}_{\text{IB}} \forall (i, j) \in S^{\text{IB}}$$

Where N_{IB} , is the number of nodes in the iteration best solution, and S^{IB} is the iteration best solution.

- [Step 7] Update the global best solution based on the objective function.
- [Step 8] Increment the iteration count, and repeat the steps until the stopping criteria is met. After all IWD iteration complete, N number of water droplet obtains N group parameter C and σ .
- [Step 9] The parameter C searched for each IWD and σ , utilizes study collection to carry out the method for model training according to step 3.
- [Step 10] Adopt these values to carry out the determination of failure prognostication; the accuracy of the process is high.

5. Evaluation Criteria

This work correctly identifies which tasks could be failed due to resource overutilization. For predicting its performance, the data were collected from different scientific workflows with 25, 50 and 100 tasks at fixed intervals using the CloudSim [\[18\]](#) and WorkflowSim [\[19\]](#) tools. The task failure prediction accuracy has been evaluated using some evaluation metrics. Pegasus [\[20\]](#) takes care of abstract mapping to concrete workflows.

Sensitivity (Recall)

Sensitivity is a measure of the proportion of actual positive cases that got predicted as positive (or true positive).

$$\text{Sensitivity} = (\text{TP}) / (\text{TP} + \text{FN})$$

- TP - True Positive = Tasks that are labeled as failure and also evaluated as failed tasks.
- FN - False Negative = Tasks that are labeled as non-failure but evaluated as failed tasks.

Specificity

Specificity is defined as the proportion of actual negatives, which got predicted as the negative.

$$\text{Specificity} = (\text{TN}) / (\text{TN} + \text{FP})$$

- TN – True Negative = Tasks that are labeled as non-failure and also evaluated as non-failed tasks.
- FP - False Positive = Tasks that are labeled as failure but evaluated as non-failed tasks.

Precision

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

$$\text{Precision} = (\text{TP}) / (\text{TP} + \text{FP})$$

F1 Score

F1 Score is the Harmonic Mean between Precision and Recall. The range for F1 Score is [0,1]. The greater the F1 Score, the better is the performance of our model.

$$F1 = 2 * \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Classification Accuracy

Classification Accuracy is the ratio of number of correct predictions to the total number of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

6. Experimentation results and discussions

Our experimental setup has four steps:

- Data collection
- Evaluation of performance metrics using percentage splits
- Comparison of proposed approach with existing models for various scientific workflows.

6.1 Data Collection

CloudSim [18] and WorkflowSim [19] classes are used to collect the dataset for failure prognostication. For predicting its performance, the data were collected from different scientific workflows with 25, 50 and 100 tasks at fixed intervals using the CloudSim [18] and WorkflowSim [19] classes. The attributes considered in the failure prediction process is listed below:

Attribute Name	Description
T_{id}	Task id
VM_{id}	Virtual Machine id
D_{id}	Data Center id
U.BW	Utilization of Bandwidth in percentage
U.CPU	Utilization of CPU in percentage
U.RAM	Utilization of RAM in percentage
U.Mem	Utilization of Memory in percentage
U.Disk	Utilization of disk in percentage
$Time_S$	Start time of execution
$Time_F$	Finish time of execution
Status	Success or Failure
EET	Estimated Execution Time

Table 2: Attributes considered for dataset collection

The maximum dynamic threshold is set using the utilization parameters such as CPU, Bandwidth, RAM, Memory, Disk and EET based on the current values and the historical data. If the current utilization value is greater than the maximum threshold value, the task is categorized as failure.

T_{id} and VM_{id} denote the id of the failed task and its corresponding Virtual Machine. D_{id} denotes the id of the datacenter where the task failure occurs. Percentages split used for training and testing data are 66% and 34% respectively.

6.2 Evaluation of Performance Metrics Using Percentage Split

The main contribution of this work is a SVR-IWD based approach in which the accuracy of the failure prognostication is high compared with the existing approaches. The metrics of the evaluation of the proposed approach is chosen as given by Aditya Mishra(2018) [21]. Figure 4,5 and 6 illustrates the comparison of various performance metrics such as sensitivity, specificity, precision, recall and F1 score.

Fig.4: Comparison of Sensitivity and Specificity

Fig.5: Comparison of Recall and Precision

Fig.6: Comparison of F1 Score

6.3 Comparison evaluation of various scientific workflows with existing approaches

To evaluate the algorithm, library of realistic workflows which are used in the scientific community Montage (astronomy), Epigenomics (biology), LIGO (gravitational physics) and SIPHT (biology), CyberShake are used. These are available as abstract workflow in XML format (DAX) in the website [15]. Each workflow application is evaluated using four different sizes, small, medium, large and extra-large. Small workflows have 24 to 30 tasks, medium workflows have 45 to 50 tasks, large workflows have 100 tasks and extra-large workflows have 995 to 1000 tasks.

The Montage is an astronomy application which is used to create custom mosaics of the sky based on the set of input images. The tasks of the Montage workflow are mostly I/O sensitive and require less amount of CPU processing capacity to execute.

Fig.7: Structure of Montage Workflow

The Epigenomics is used in the bioinformatics field and its basic purpose is to automate the execution of various genome-sequence operations. Most of the tasks of the workflow require high CPU capacity to process with low I/O utilization.

Fig.8: Structure of Epigenomics Workflow

The LIGO’s Inspiral Analysis workflow is used in Physics to detect gravitational waves. Maximum numbers of the task of the workflow characterized as CPU sensitive that consumes large memory.

Fig.9: Structure of LIGO Inspiral Workflow

The Cyber Shake workflow is generated by measuring the earthquake hazards by generating synthetic seismograms. Most of the tasks of the workflow are CPU sensitive with high memory requirements.

Fig. 10: Structure of CyberShake Workflow

The SIPHT workflow from the bioinformatics project at Harvard is used to automate the search for untranslated RNAs (sRNAs) for bacterial replicons in the NCBI database. This analyzes data from the coalescing of compact binary systems such as binary neutron stars and black holes. The jobs in this workflow are the most computationally intensive and have relatively high CPU utilization.

Fig.11: Structure of Sipht Workflow

The proposed IWD-SVR based approach is evaluated using the various scientific workflows such as Epigenome, Montage, CyberShake, Inspiral and Sipht. Figure 7 illustrates the accuracy comparison of proposed approach with existing Naive Bayes, Random Forest, Rule Based, and Logistic Regression. The overall accuracy of the proposed approach is 98.2% which is 2.18% higher than the existing Naive Bayes approach.

Fig.12: Accuracy comparison using various scientific workflows.

7. Conclusion

Proactive fault tolerance mechanism is preferred in scientific workflows in cloud environment because they can anticipate the failure well ahead so that there is sufficient time for migration and mitigation of failures. Proactive mechanisms are mainly based on the historical data to predict the future faults. The efficiency of such mechanisms is based on several factors of which the prediction accuracy plays the vital role. Hence the focus of my work is to enhance the prediction accuracy of the proactive fault tolerance mechanism. In our proposed IWD-SVR based approach, the prediction accuracy can be achieved better by nearly 2.18% higher compared to the existing approaches.

Funding: No funding

Conflicts of interest/Competing interests: No conflict and Competing Interest

Availability of data and material: Available in paper

Code availability: Available in paper

Ethics approval: Not applicable

Consent to participate: Not applicable

Consent for publication: Not applicable

8. References

- [1] Cagatay Catal : Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, 38 (2011), pp 4626 – 4636.
- [2] Ruchika Malhotra, Ankita Jain: Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality. *Journal of Information Processing Systems*, Vol.8, No.2, June 2012, pp. 241 – 262.
- [3] Sadeka Islam, Jacky Keung, Kevin Lee and Anna Liua: Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28 (2012), pp. 155 - 162.
- [4] Ilenia Fronzaa, Alberto Sillitti, Giancarlo Succi, Mikko Terho, Jelena Vlasenkoa: Failure prediction based on log files using Random Indexing and Support Vector Machines. *The Journal of Systems and Software* 86 (2013), pp. 2 - 11.
- [5] Anju Bala, Inderveer Chana: Intelligent failure prediction models for scientific workflows. *Expert Systems with Applications*, 2015, Vol 40, pp.980 - 989.
- [6] Upasna Chandarana Kothari and Moe Momayez: Machine Learning: A Novel Approach to Predicting Slope Instabilities. *International Journal of Geophysics*, 2018, Article ID 4861254.
- [7] P. Padmakumari, A. Umamakeswari: Task Failure Prediction using Combine Bagging Ensemble (CBE) Classification in Cloud Workflow. *Wireless Personal Communications*, March 2019. <https://doi.org/10.1007/s11277-019-06238-9>
- [8] Xiang - Ming Gao, Shi-feng Yang, and San-bo Pan: Optimal Parameter Selection for Support Vector Machine Based on Artificial Bee Colony Algorithm: A Case Study of Grid - Connected PVSystem Power Prediction. *Hindawi Computational Intelligence and Neuroscience*, Volume 2017, Article ID 7273017, 14 pages <https://doi.org/10.1155/2017/7273017>
- [9] Jui-Sheng Chou, Ngoc-Tri Ngo and Anh-Duc Pham: Shear Strength Prediction in Reinforced Concrete Deep Beams Using Nature-Inspired Metaheuristic Support Vector Regression. *American Society of Civil Engineers*, 2015, DOI: [10.1061/\(ASCE\)CP.1943-5487.0000466](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000466).
- [10] Manikandan, S, Chinnadurai, M, "Effective Energy Adaptive and Consumption in Wireless Sensor Network Using Distributed Source Coding and Sampling Techniques",. *Wireless Personal Communication* (2021), 118, 1393–1404 (2021). <https://doi.org/10.1007/s11277-021-08081-3>, Springer
- [11] Peng, Hao & Ling, Xiang. (2015): Predicting thermal-hydraulic performances in compact heat exchangers by support vector regression. *International Journal of Heat and Mass Transfer*. 84. 10.1016/j.ijheatmasstransfer.2015.01.017.
- [12] Zaidi, Sadaf. (2015): Novel application of Support Vector Machines to model the two phase boiling heat transfer coefficient in a vertical tube thermosiphon reboiler. *Chemical Engineering Research and Design*. 98. 10.1016/j.cherd.2015.04.002.
- [13] Nusrat Parveen, Sadaf Zaidi, Mohammad Danish: Support vector regression model for predicting the sorption capacity of lead (II), *Perspectives in Science*, Volume 8, 2016, Pages 629-631, ISSN 2213-0209, <https://doi.org/10.1016/j.pisc.2016.06.040>.
- [14] Listiani, Mariana et al. "Support Vector Regression Analysis for Price Prediction in a Car Leasing Application." (2009).
- [15] Kadir Kavaklioglu: Modeling and prediction of Turkey's electricity consumption using Support Vector Regression, *Applied Energy*, Volume 88, Issue 1, 2011, Pages 368-375, ISSN 0306-2619, <https://doi.org/10.1016/j.apenergy.2010.07.021>.
- [16] Suk-Tae Seo et.al.: Support Vector Regression-based Model to Analyze Prognosis of Infants with Congenital Muscular Torticollis. *Healthcare Informatics Research*, Dec. 2010; Vol 16, Issue 4, pp. 224 – 230.
- [17] Hamed Shah-Hosseini: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *Int. J. Bio-Inspired Computation*, Vol.1, Nos. 1 / 2, 2009.
- [18] Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., & Buyya, R. (2011): CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41, pp. 23–50.
- [19] Chen, W., & Deelman, E. (2012): WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In *IEEE 8th international conference on E-Science*, pp. 1–8.
- [20] Deelman, E., et al. (2005): Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13, 219–237.
- [21] Aditya Mishra(2018) <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>