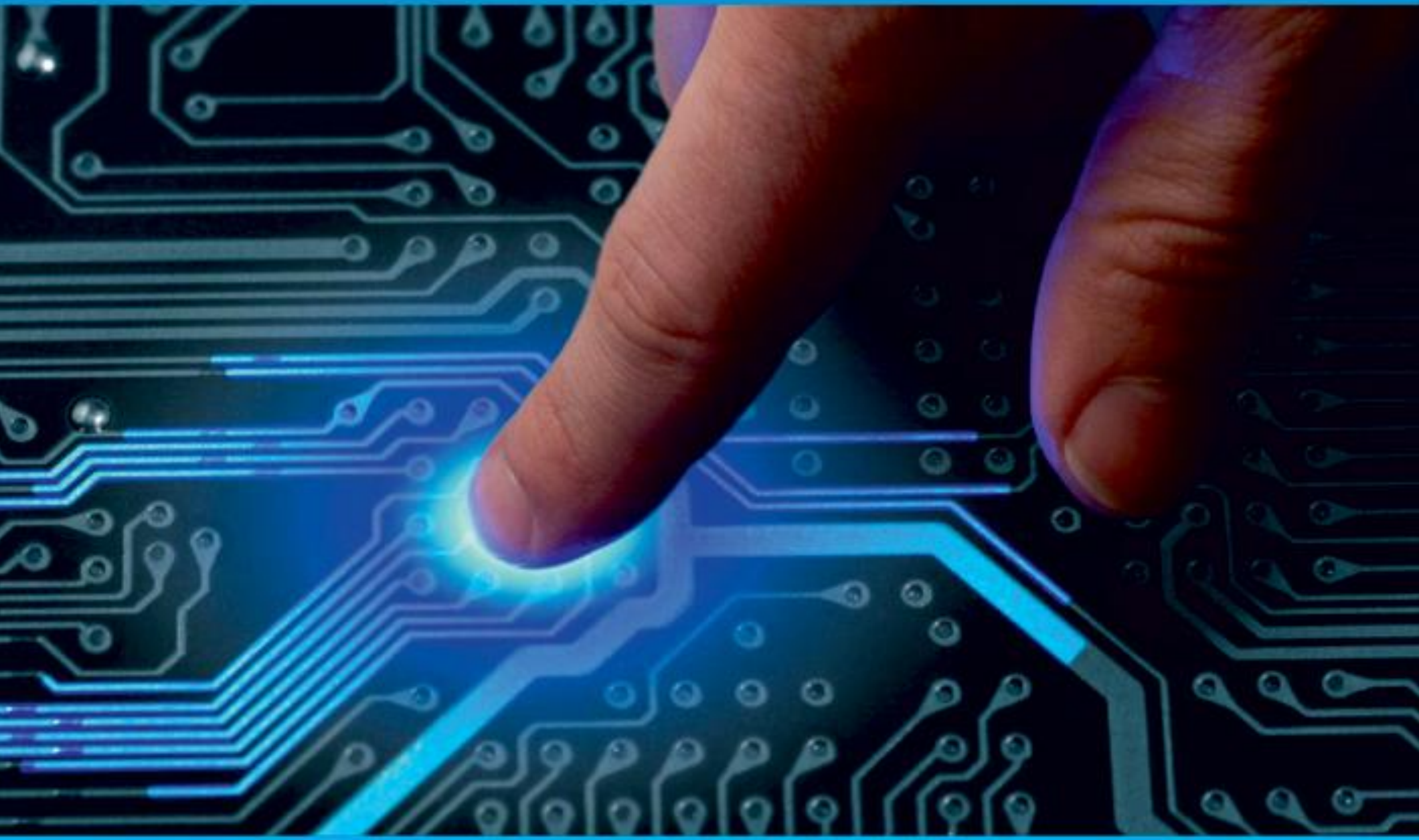


# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 4, April 2024

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204308 |

# Analysis of Task Resource Usage and Prediction of Failures in Cloud Environment

**Vinutha M R, Sheetal S Joshi, Shreeram Kaushik K V, Anusha S**

Department of Information Science & Engineering, Malnad College of Engineering, Hassan, India

**ABSTRACT:** In the realm of cloud computing systems, enhancing reliability stands as a pivotal objective, necessitating a profound comprehension of failure dynamics and the ability to forecast failures preemptively. A meticulous statistical scrutiny of workload data within cloud environments unveils crucial insights into failure patterns, serving as a linchpin for bolstering system reliability. This paper delves into a comprehensive statistical analysis of task resource utilization data extracted from the expansive Google cluster dataset. Furthermore, novel failure prediction algorithms are devised to prognosticate failures with heightened accuracy. Through our investigation, we discern notable disparities in resource utilization trends, execution durations, and resource consumption between failed and successfully completed tasks. Leveraging diverse resampling techniques in conjunction with the XGBoost classifier, we achieve substantial advancements in failure prediction within highly imbalanced datasets. Our findings highlight that the fusion of Synthetic Minority Oversampling Technique (SMOTE) with XGBoost yields remarkable precision of 92% and a commendable recall rate of 94.8% in predicting task status, thereby underscoring its efficacy in bolstering system reliability within cloud computing infrastructures.

**KEYWORDS:** Bi-LSTM, Cloud Sim

## I. INTRODUCTION

Large-scale systems such as cloud computing and High-Performance Computing (HPC) are pivotal in driving computing capabilities to unprecedented scales and speeds. On the other hand, they are extremely vulnerable to multitude of failures, incurring significant costs for service providers and subscribers in terms of income and standing. Consequently, ensuring resiliency emerges as a paramount concern in the effective deployment of such large-scale systems. This manuscript undertakes an in-depth exploration into the attributes specific to the task failures and endeavors to forecast these failures proactively to enable preemptive actions aimed at avoidance or mitigation of their adverse effects. The investigation is conducted utilizing the Google's publicly accessible cluster data collection, which furnishes comprehensive insights into asset usage, scheduling strategies, priorities, and execution statuses of jobs and tasks. While analogous datasets from Azure and Alibaba also offer task execution details, they lack information pertaining to failed tasks. Prior research has primarily concentrated on analyzing Google cluster data in terms of scheduling strategies and priority features, this study diverges by empirically investigating the resource usage patterns of failed tasks & completed tasks. Serving as an extension to existing literature, this study unveils previously unexplored insights into task failures.

The primary contributions and key discoveries of this paper are as follows:
1. Distinct disparities in resource usage patterns are observed between failed and completed tasks, with failed tasks exhibiting a tendency to consume more resources compared to their successfully completed counterparts.
2. The resource usage of failed tasks demonstrates a pronounced positive correlation, whereas that of completed tasks exhibits a more typical correlation pattern. This distinct pattern in resources age serves as a predictive indicator for identifying potential task failures.
3. Failed jobs tend to undergo prolonged execution durations before eventual failure, potentially indicative of progress stalls during task execution.
4. Leveraging Synthetic Minority Oversampling Technique (SMOTE) in conjunction with XGBoost, task status prediction achieves an impressive precision of 92% and a commendable recall rate of 94.8%.

## II. OBJECTIVES AND DATASET DESCRIPTION

The proposed study encompasses two primary objectives: firstly, to scrutinize the resource usage patterns of both failed and completed tasks with the aim of identifying discriminative patterns in real-time to facilitate early failure detection; secondly, to devise a prediction model capable of forecasting task failures in advance, thereby enabling timely interventions to prevent or mitigate such occurrences.

Among the publicly accessible large datasets, the Google cluster dataset stands as a unique repository containing comprehensive information regarding job and task failures. Spanning data collected from 12,500 nodes over a 29-day period, this dataset comprises six distinct data tables capturing machine events, machine attributes, job events, task events, task usage, and task constraints.

The task events table furnishes intricate details regarding tasks, encompassing parameters such as priority, scheduling class, user identity, job and machine identifiers, resource requisitions, and event types. These event types encapsulate various stages of a task's lifecycle, including submission, scheduling, eviction, failure, completion, cancellation, and loss.

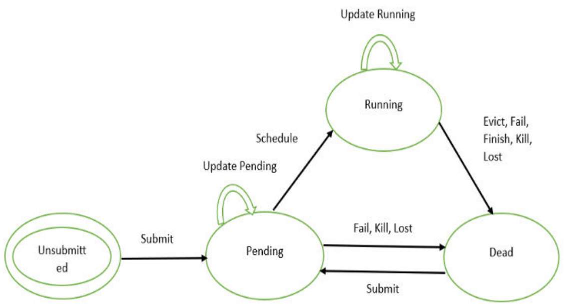


**Fig. 1: State transition diagram of tasks**

Complementing the task events table, the task resource usage table offers periodically profiled metrics pertaining to resource utilization, encompassing peak and average CPU usage, memory consumption, total page cache, disk I/O activity, cycles per instruction, as well as disk and memory accesses per instruction. Notably, both tables share common primary keys, namely time, Job ID, and Task ID, facilitating seamless integration and analysis of task-related data.

The frequency distribution of task events in the Google Cluster dataset is depicted in Table 1. Analysis of this distribution reveals a notable imbalance within the dataset, particularly evident in the distribution between the failed and finished classes.

Specifically, the failed class constitutes approximately 29% of the dataset, while the finished class accounts for 38%. This substantial skew in class representation underscores the need for careful consideration and appropriate handling of imbalanced data within the context of this dataset.

| Event Status | Tasks |
|---|---|
| Failed | 13825994 |
| Finished | 1207622 |
| Killed | 10292068 |
| Evicted | 57520568 |

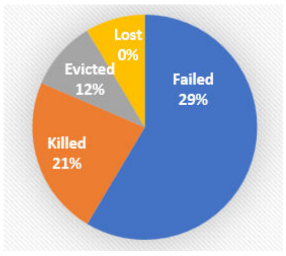


**Table 1. The distribution of task instances based on event type frequencies**

The primary objective of classification is to accurately predict the failure of tasks, particularly those towards the class representing the minority within the dataset. However, the challenge posed by imbalanced datasets lies in the propensity of classification algorithms to exhibit bias towards the majority class, consequently leading to misclassification of the minority class. Various methodologies have been proposed to address this issue, including under sampling, oversampling, synthetic data generation, among others [1-4]. Among these techniques, random under sampling emerges as a viable solution, involving the random reduction of instances from the majority class. This technique is particularly effective for large datasets where the frequency of the majority class significantly outweighs that of other classes. In the context of this study, our proposed approach explores a range of techniques discussed in Section V to mitigate the challenges associated with imbalanced datasets.

## III. NOVEL FRAMEWORK AND IMPLEMENTATION PROPOSAL

This section delves into the experimental configuration and the proposed framework devised for the analysis and prediction of failures. Our experimental setup encompasses a test environment comprising a cluster comprising three machines. The coordinator machine is equipped with a 64-bit Mac OS High Sierra operating system, an Intel Core i7 processor, 16 GB DDR3 RAM, and 50 GB of disk space. Additionally, two worker nodes are included in the cluster, each featuring a 64-bit Mac OS High Sierra operating system, an Intel Core i5 processor, 4 GB DDR3 RAM, and 50 GB of disk space.

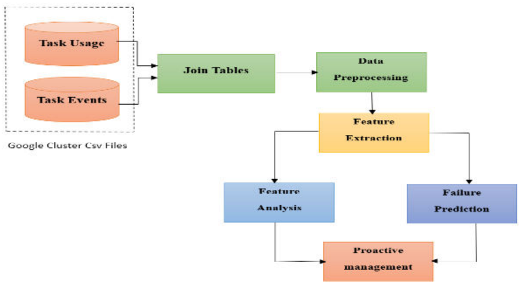


**Fig. 2: Analysis of Failure and framework prediction**

The software stack employed in our experimentation includes Presto version 0.194 and Apache Cassandra version 3.11.2 for data management, and for analytical purposes, R Version 3.5.0 is utilized.
Outlined in Figure 2, our proposed framework for failure analysis and prediction comprises distinct modules, each serving a specific function. Below, we provide a detailed description of these modules.

**Preprocessing of Data:** which encompasses several essential steps to prepare the data for subsequent analysis and prediction tasks. This preprocessing stage addresses various challenges such as joining disparate tables, filtering out invalid and missing entries, and handling data imbalance. Specifically, the task involves consolidating information from two distinct tables: the Task Usage Table containing task resource usage data, and the Task Events Table containing task execution status. Integration of these tables is facilitated using the Job ID and task ID fields as primary keys. Notably, the Task Events data occupies approximately 2 GB, while the Task Usage data is approximately 40 GB in a compressed state. Given the substantial size of the data, traditional approaches using SQL databases such as MySQL proved to be inefficient, taking several days to complete the join operation. To address this challenge, we leveraged distributed database query engines, notably Presto DB in conjunction with Cassandra, to expedite the preprocessing phase.

Presto DB, an open-source distributed SQL query engine renowned for its high-speed querying capabilities, was instrumental in accelerating data processing tasks [15]. Its versatility extends to querying diverse databases such as Hadoop and Cassandra. Our framework capitalizes on the distributed database capabilities of Cassandra [16], with Figure 3 illustrating the Presto-Cassandra architecture employed for querying the Google Cluster dataset. This architecture enables seamless operations including table joins, extraction of specific task records (e.g., finished, killed, failed), and database analytics.
Initially, we employed MySQL for loading and preprocessing tasks, which proved to be time-consuming, taking up to two days to execute basic select queries. However, with the adoption of the Presto and Cassandra architecture, the processing time significantly improved, reducing the duration of operations such as select and join to 2 to 5 hours. This transition underscored a notable enhancement in operational efficiency facilitated by the adoption of distributed SQL databases and the Presto engine.

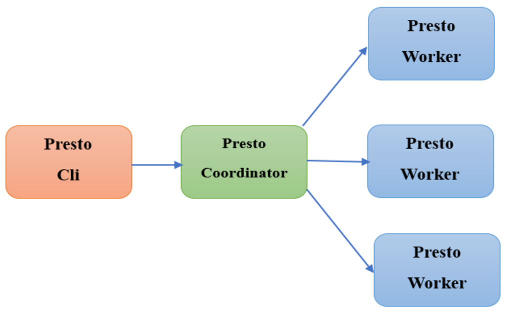


**Fig. 3: Architecture of Presto-Cassandra**

The subsequent preprocessing step involves addressing the challenge of class imbalance. To mitigate this issue, we experiment with various techniques including random under sampling (RUS), random oversampling, and Synthetic Minority Oversampling Technique (SMOTE). These approaches aim to create a more balanced distribution of instances across classes.

**Feature Extraction:** This crucial step involves the meticulous selection of features for model training. Given the study's focus on the resource usage characteristics of tasks, key metrics sourced from the Google cluster dataset are utilized, including mean CPU usage, mean memory usage, page cache memory, mean disk usage, job length, and event type. Notably, features such as priority and scheduling parameters are omitted from this study, as existing literature has extensively analyzed these aspects.

**Failure Analysis:** This module is dedicated to the comprehensive examination of failure characteristics through the analysis of features extracted in the previous step. Its primary objective is to discern patterns that differentiate failed tasks from successfully completed ones, thereby enhancing our comprehension of potential failure scenarios. Detailed insights into failure analysis are expounded upon in Section IV of this manuscript.

**Failure Prediction:** Leveraging the features derived from the feature extraction module, this module employs XGBoost classifier for training models aimed at classifying and predicting the termination status of tasks. A comprehensive discussion of the failure prediction module is provided in Section V of this manuscript.

**Proactive Management:** This module extends the scope of our work by addressing failure remediation strategies based on predicted outcomes from the failure prediction model. Potential proactive strategies include task migration, checkpointing and rollback, virtual machine migration, resource scaling, and software rejuvenation, among others [20].

## IV. ANALYSIS OF FAILURE CHARACTERISTICS

This section conducts a statistical analysis of the resource usage patterns exhibited by failed, killed, and finished tasks. More specifically, we examine and contrast failed tasks with finished tasks in terms of resource consumption and explore the correlation between resource usage and execution time.

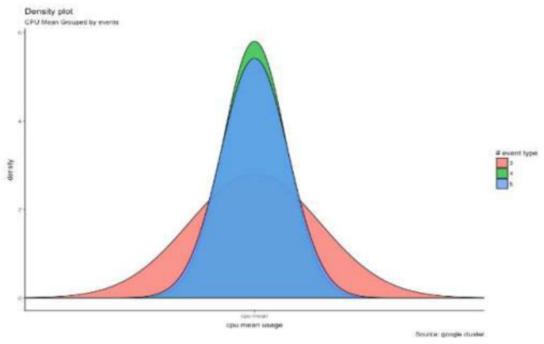


**Fig. 4. (a) Density plot of mean CPU Usage of failed (3), finished (4), and killed (5) tasks in google cluster dataset**

In Figure 4(a), the density plot illustrates the mean CPU usage of failed, killed, and finished tasks. Our analysis reveals that the CPU usage of failed tasks exhibits a wider spread, indicating greater variability compared to both killed and finished tasks.
Furthermore, Figures 4(b) and 4(c) present box plots illustrating the CPU and memory utilization of failed (3), finished (4), and killed (5) jobs. Analysis of these box plots reveals that the minimum, maximum, and median CPU and memory usage values for failed tasks exceed those of finished tasks. This indicates that failed jobs tend to utilize resources more extensively compared to both killed and finished jobs. Additionally, the high inter-quartile range (variance) observed for failed jobs suggests greater variability in resource usage among failed tasks. Similar trends are observed for disk and cache usage. These analyses depicted in Figure 5 collectively demonstrate that failed tasks exhibit resource consumption and greater variance in resource usage compared to both finished and killed tasks.

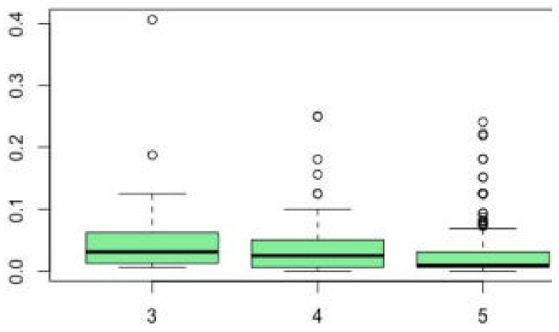
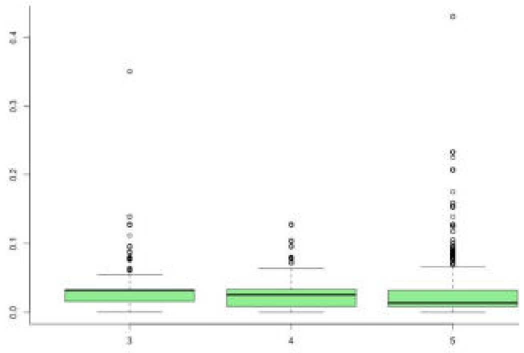


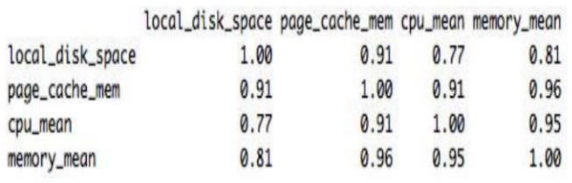
**Fig. 4. (b) Box plot of CPU Utilization(y-axis) of failed (3), finished (4), and killed (5) Jobs in google cluster**



**dataset.**

**Fig. 4. (c) memory Utilization(y-axis) of failed (3), finished (4), and killed (5) tasks in google cluster dataset**

Motivated by the diverse patterns of resource usage illustrated in Figure 6, we conducted a correlation analysis of various resources including local disk space, page cache memory, mean CPU usage, and mean memory usage for both failed and finished tasks. Figures 5(a) and 5(b) depict the correlation plots between different resources for failed and finished tasks, respectively. Our analysis reveals that for failed tasks, the correlation between resources exhibits significantly high positive values ranging from 0.7 to 0.96, whereas for finished tasks, the correlation values fall within the normal range of 0.43 to 0.82. This disparity suggests that failed tasks demonstrate an atypical resource usage behavior characterized by simultaneous increases in all resource consumption metrics before failure. Such anomalous behavior can serve as a predictive indicator for identifying tasks likely to fail soon.

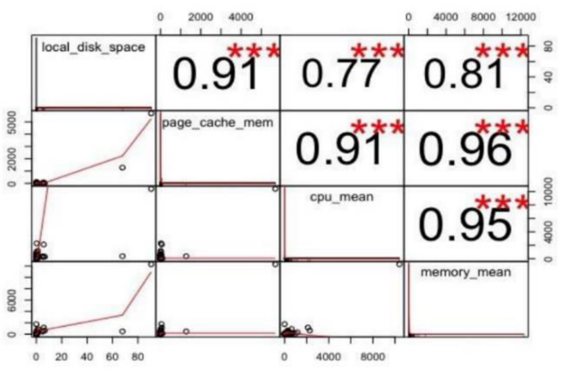


| | local_disk_space | page_cache_mem | cpu_mean | memory_mean |
|---|---|---|---|---|
| local_disk_space | 1.00 | 0.91 | 0.77 | 0.81 |
| page_cache_mem | 0.91 | 1.00 | 0.91 | 0.96 |
| cpu_mean | 0.77 | 0.91 | 1.00 | 0.95 |
| memory_mean | 0.81 | 0.96 | 0.95 | 1.00 |

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204308 |



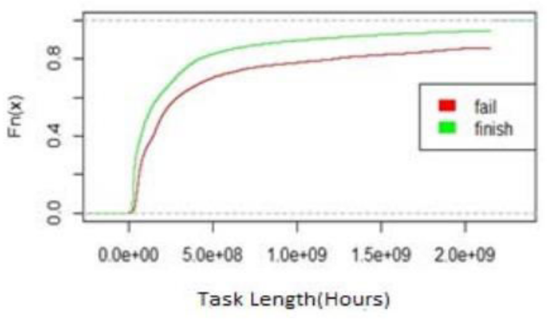
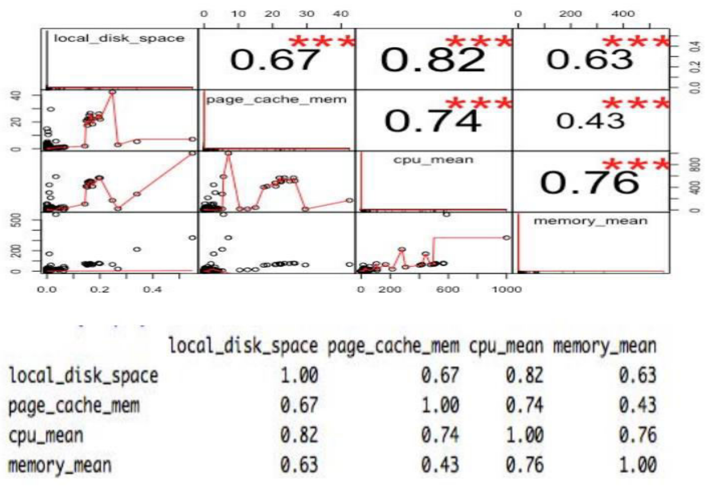


**Fig. 5 (a) Google Cluster correlation failed tasks**

Figure 6(a) illustrates the Cumulative Distribution Function (CDF) for task duration, comparing failed tasks with finished tasks. Notably, finished tasks exhibit shorter execution times in contrast to failed tasks. Additionally, Figure 6(b) presents a box plot depicting the service times of failed, finished, and killed tasks. Analysis of this plot reveals that failed tasks have higher median, maximum, and inter-quartile (variance) values compared to finished tasks. This suggests that failed tasks tend to execute for prolonged durations, potentially experiencing hang-ups or freezes before ultimately failing, whereas finished tasks swiftly complete their execution processes.



```
                 local_disk_space page_cache_mem cpu_mean memory_mean
local_disk_space             1.00           0.67     0.82        0.63
page_cache_mem               0.67           1.00     0.74        0.43
cpu_mean                     0.82           0.74     1.00        0.76
memory_mean                  0.63           0.43     0.76        1.00
```

**Fig. 6(a) CDF of finished and failed task length**

The primary objective of the failure prediction model lies in accurately forecasting failure classes with high precision and recall rates. To overcome the challenge of class imbalance within the dataset, various techniques have been explored, including Random Under sampling (RUS), Random Oversampling (ROS), and Synthetic Minority Oversampling Technique (SMOTE). Upon preprocessing, the dataset underwent resampling using ROS and SMOTE, resulting in an equal instance count of 373,083 for both finished and failed classes Subsequently, an XGBoost classifier

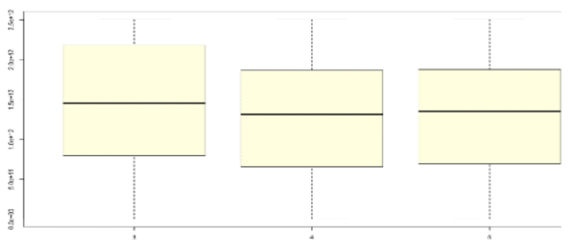


**Fig. 6(b) service length(y-axis) of failed (3), finished (4), and killed (5) tasks in google cluster dataset**

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204308 |

was trained using the resampled dataset to facilitate prediction. Evaluation of the model on a test dataset revealed the precision and recall metrics for different resampling approaches, as detailed in Table 2 and graphically depicted in Figure 7. These assessments highlight Resampling techniques which effectively enhance model performance, ensuring accurate identification of both finished and failed classes.

## V. PREDICTION OF FAILURE

The primary aim of the failure prediction model is to achieve high precision and recall in predicting the failure class. To tackle the issue of class imbalance, several techniques such as Random under sampling (RUS), Random Oversampling (ROS), and Synthetic Minority Oversampling Technique (SMOTE) were implemented. Post resampling of the preprocessed dataset [6], ROS and SMOTE resulted in an equal instance count of 373,083 for both finished and failed classes, while RUS yielded 9,741 instances for each class.

Various resampling strategies were explored to address dataset imbalance, resulting in 9,741 instances for both finished and failed classes with RUS, and 373,083 instances for each class with ROS and SMOTE. After resampling, an XGBoost classifier was trained for prediction purposes. The model's performance was evaluated on a test dataset, with precision and recall metrics recorded for different resampling techniques, as outlined in Table 2.
The findings were visually represented in Figure 7. Analysis of Figure 7 indicates that SMOTE outperforms other techniques, demonstrating a precision and recall value of 92% and 94.8% respectively.

| Name of Sampling Technique | Precision | Recall |
|---|---|---|
| Random Under sampling | 78.8% | 83.5% |
| Random Oversampling | 76.6% | 82.4% |
| SMOTE | **92%** | **94.8%** |

**Table 2: Precision and Recall values**

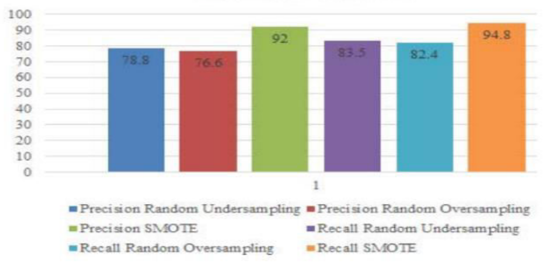


**Fig. 7: Recall values and Precision**

## VI. RELATED WORK

Prior studies on the Google cluster dataset have unveiled intriguing insights into scheduling strategies, user characteristics, and job failures. Notably, analyses have revealed patterns such as higher task resubmission rates in failed jobs, clustering of user profiles based on job failures, and the efficacy of Recurrent Neural Networks (RNNs) in predicting failures with a 40% true positive rate. Furthermore, empirical failure analysis has delineated variance in failure patterns by priority type and distribution characteristics, shedding light on resource utilization issues and the impact of machine locality on job success.

In contrast, recent endeavors have emphasized open-source tools and tackled dataset size and class imbalance issues while focusing on resource usage patterns and task durations of failed tasks. By leveraging accessible methodologies, these studies offer reproducible insights into failure prediction, highlighting the significance of resource limit violations over machine failures and the role of heightened I/O activity in unsuccessful jobs.

## VII. CONCLUSION

Examining task failure characteristics within the expansive Google cluster dataset reveals distinct resource usage patterns between failed and finished tasks, presenting a valuable opportunity for enhancing the reliability of cloud services. To mitigate class imbalance, resampling techniques like RUS, ROS, and SMOTE were employed, followed by classification using the XGBoost algorithm. Results demonstrate the effectiveness of SMOTE in conjunction with XGBoost, yielding promising accuracy levels. However, it's noted that SMOTE may exhibit overfitting tendencies on the dataset. Therefore, future endeavors aim to explore more sophisticated approaches to address class imbalance and extend the analysis and prediction framework to encompass diverse cloud environments beyond the confines of the Google cluster.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204308 |

## REFERENCES

1. Smith, J. R., & Goodwin, S. D. (2008). Task analysis and activity theory: Towards an integrated approach to human work. Ergonomics, 51(7), 985-1009.
2. Stanton, N. A., Salmon, P. M., & Walker, G. H. (2017). A guide to task analysis: The task analysis working group. CRC Press.
3. Vicente, K. J. (1999). Cognitive work analysis: Toward safe, productive, and healthy computer-based work. CRC Press.
4. Endsley, M. R., & Jones, W. M. (2012). Designing for situation awareness: An approach to user-centered design (2nd ed.). CRC Press.
5. Annett, J., Duncan, K. D., & Stammers, R. B. (2000). Task analysis. CRC Press.
6. Rasmussen, J., Pejtersen, A. M., & Goodstein, L. P. (1994). Cognitive systems engineering. Wiley.