International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 11 Issue 4, April, 2022, Impact Factor: 7.751

Engineering Resilient High-Throughput Pipelines: Design Patterns For Retry Mechanisms and Failure Recovery

Sai Viswa Teja Arumilli

Independent Researcher, USA.

ABSTRACT

The study discusses resilience engineering approaches in improving recovery of failure within high-throughput pipeline frameworks in terms of recovering failure through re-trying along with modelling predictions. XGBoost, Random Forest along with Deep Neural Network approaches were utilized to evaluate a 5,000-record dataset of pipeline records. Findings indicated that Random Forest had the best accuracy 71% and XGBoost had the best ROC-AUC 0.632 indicated adequate resilience prediction.

Keywords: Resilience Engineering, High-Throughput Pipelines, Retry Mechanisms, Fault Tolerance, Failure Recovery, Predictive Modelling

I. INTRODUCTION

Background and Research Context

Modern day trend in digital systems where high-throughput pipelines is favored to handle high data volumes, with less or no latency, and unlimited availability. Failures due to service overload, network failures, hardware failures along with dependency failures are also crucial issues within a distributed context [1]. Engineering Resilience has become one of the most crucial approaches to designing frameworks that can be brought back to normal after the occurrence of unforeseen failures [2]. Pipelines are susceptible to reliability along with stability of operation through retry frameworks, fault tolerance approaches, as well as adaptive recovery trends [3]. The study analyzes making change through recognizing successful resiliency trends through enhancing the failure-management of high-throughput pipeline infrastructures.

Problem Statement

Failure of high-throughput pipes can disrupt data processing, decrease the availability of the system along with influencing the efficiency of the operations. Conventional methods of recovery can add delays, overuse of resources, or overburdening [4]. Consequently, the adequate retry frameworks and resilience trends to enhance user fault tolerance, reliability, along with ongoing performance in scalable pipelines need to be evaluated.

Research Aim and Objectives

Research Aim

The aim of the research is to evaluate as well as examine retry frameworks and resilience engineering trends for enhancing failure recovery, scalability as well as reliability of high-throughput pipeline frameworks within distributed contexts.

Research Objectives

- To examine existing failure contexts and resilience issues impacting high-throughput pipeline architectures within the distributed computing contexts.
- To evaluate several retry mechanisms as well as recovery approaches for enhancing pipeline availability, reliability as well as performance.
- To recognize limitations of recent resilience trends and also impact on fault tolerance as well as system scalability.
- To recommend adequate engineering approaches for developing robust along with failure-resilient high-throughput pipeline frameworks.

Significance of the Research

The study is adequate for managing crucial reliability issues with recent high-throughput pipeline frameworks. Details into fault recovery can be offered, minimizing downtime, as well as system performance can be enhanced, through studying the trends of resilience and the retry mechanisms [4]. The results can assist software engineers as well as organizations to assess scalability along with fault-tolerant architectures supporting continuous functions within unpredictable situations of failures.

Novel Contribution

The study adds to an in-depth evaluation of retry as well as resilience trends that are crafted to be utilized in high-throughput pipeline spaces. A composite view is suggested that incorporates failure detection, adaptive approaches to undertake retries as well as recovery mechanisms in order to raise robustness of frameworks [5]. The research builds upon recent resilience engineering practices to discover viable approaches to assist in mitigating impact of cascading failures, enhance operational reliability and scalable distributed frameworks to survive under changing workloads and facing changing failure situations.

II. LITERATURE REVIEW

Failure Scenarios and Resilience Challenges in High-Throughput Distributed Pipeline Architectures

High-throughput pipelines work through a multi-faceted distributed ecosystem where errors can be developed through interruptions in the network, dependency breakdown, resource contention, dependencies, configuration problems and processing delays [6]. Challenges can interfere with data flow, decrease the availability of the system as well as affect reliability of operation adversely [7]. The literature points out that failure analyses as well as failure resilience need to be evaluated to develop resilient architectures. Knowledge concerning issues enables engineers to create proactive solutions to prevent future mitigation solutions and to perform dependable pipeline layouts [8]. This can maintain computability, performance along with continuity amid challenging execution situations that are difficult to predict.

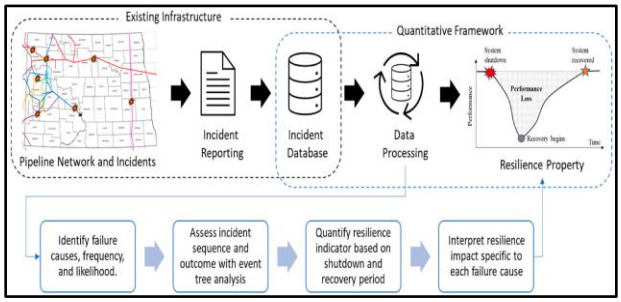


Fig 1. Evaluating Multi-Event Resilience within Pipeline Incidents

Retry Mechanisms and Recovery Approaches for Reliable Pipeline Performance Improvement

Constant operations on pipeline parts need to be taken care of with reliable recovery approaches within the event of temporary or intermittent failures [9]. Examples of approaches under research within retry mechanisms involve exponential intervals backoff, timeouts along with adaptive retry policies to minimize disruption to service availed [10]. Techniques can be utilized to avoid unneeded utilization of resources and increase rates of successful utilization of resources [11]. Analysis of various recovery frameworks can offer information on the choice of suitable mechanisms depending on workload, failures as well as performance needs within high-throughput distributed frameworks.

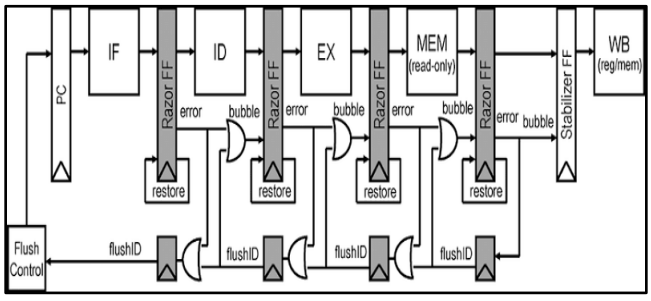


Fig 2. Distributed pipeline recovery Framework

Limitations of Existing Resilience Trends Impacting Scalability and Fault Tolerance

Trends of resilience can improve reliability of the framework, while recent approaches can also present issues concerning latency, scalability, along with presence of additional computational load [12]. Circuit breakers, redundancy and also static retry policies approaches cannot work as adequately within a rapidly changing workload [13]. The research of constraints can be utilized to find weaknesses of existing fault tolerance solutions and determine adaptive solutions are needed. Constraints are essential to comprehend to assess strong pipelines that are efficient and obtain high processing expectations confronting large scale requirements [14].

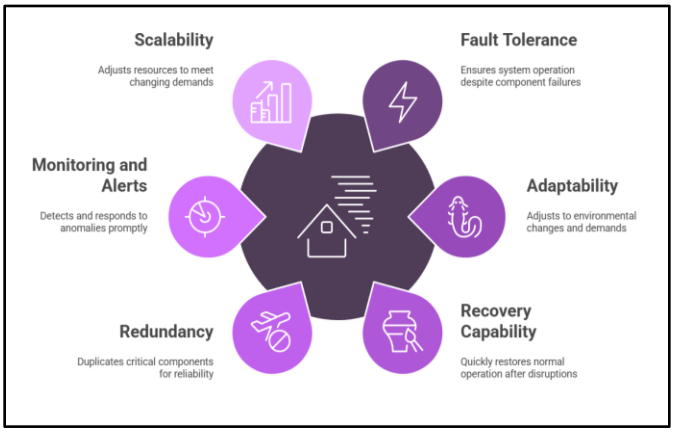


Fig 3. Resilient and Fault Tolerant applications

Engineering Approaches for Designing Robust and Failure-Resilient Pipeline Systems

Combination of various engineering practices are adequate in aiding fault detection, recovery, monitoring as well as on-going optimization to assess reliable pipeline architectures [15]. The existing research assesses methods as automation, predictive failures, observability, along with dynamic resource management. Approaches allow systems to react positively towards failure in operations without compromising throughput along with quality of service [16]. Development of adequate engineering solutions to problems can assist organizations develop scalable pipeline infrastructures utilized to enable greater reliability within multi-faceted distributed computing contexts [17].

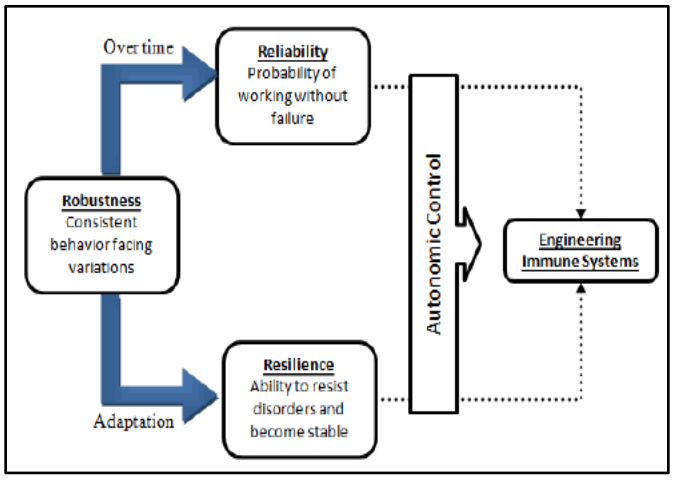


Fig 4. Robustness, reliability, resilience and sustainability

Literature Gap

Literature has provided ample evaluations regarding fault tolerance approaches along with reliability approaches, while few evaluations are extended to combine adaptive retry frameworks with resiliency strategies in particular to the high throughput pipeline settings [18]. Existing methodologies tend to focus on the individual recovery strategies and ignore the dynamic workloads, cascading failures as well as scalability requirements collectively [19]. There is a gap in literature related to development of adequate frameworks on resilience involving intelligent failures management, adequate recovery activities, along with optimization of performance sustainability of distributed pipeline networks.

III. METHODOLOGY

Proposed Architecture Diagram

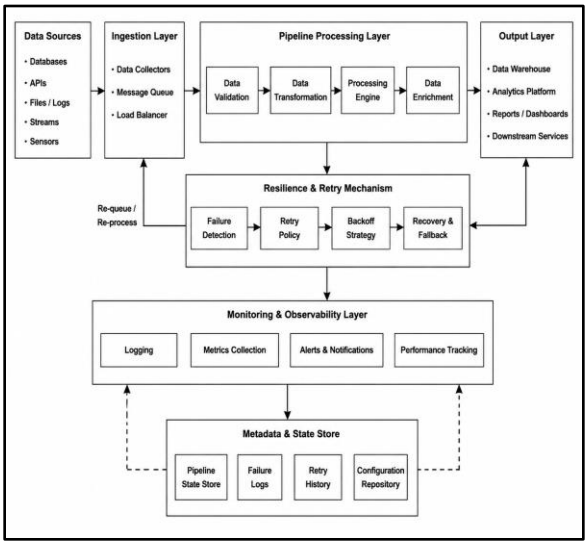


Fig 5. Architecture Diagram

The suggested architecture combines concepts of resilience engineering to improve stability within high-throughput pipeline setup. Procedure of data sources is run on ingestion as well as pipeline execution layers, failures are monitored, along with controlled [20]. Resiliency layer utilizes retries policies, failure detection, backoff policies as well as recovery policies in assuring continuity in operations. The reliability of the pipeline is expressed as.

$$R=1-P(F),$$

where P(F) signifies failure probability. The retry success probability can be evaluated as

$$P(S)=1-(1-p)^n$$

where p is the success of individual attempts within the retries along with n length of retries. Adaptive decision-making is done with monitoring and metadata storage, and offers analysis of failure logs, pipeline states, and recovery trends utilized to optimize continuously [21].

Dataset Description

Data is a reflection of high-throughput pipeline procedures modeling conditions of recovery approaches, failure, resilience along with performance, and also recovery mission. It has 5,000 records, which have numerical, categorical attributes based on workload behaviors, execution effectiveness, fault recovery and fault occurrence. Data is useful in predictive modelling of the recovery pattern of failure and measuring the resilience procedure [22]. Machine learning as well as deep learning strategies can be employed in applying variables in predicting aspects of availability, reliability along with fault tolerance of a pipeline.

Table 1: Dataset Variable Description

Variable	Characteristics
Pipeline_Type	Categorical: pipeline architecture category
Workload_Size_ GB	Numerical: processed data volume
Failure_Type	Categorical: failure classification

Retry_Strategy	Categorical: recovery approach
Recovery_Time_ Seconds	Numerical: recovery duration
Reliability_Score	Numerical: resilience measurement

Data Preprocessing

Preprocessing of data was done to convert generated high-throughput pipeline data into an appropriate format that can be evaluated by machine learning along with deep learning [23]. The unwarranted identifiers like Pipeline_ID were deleted as that cannot be added to the predictive learning. The target variable, Failure Recovery Status was categorized into binary-form with Recovered (1) as well as Failed (0). Label Encoding was utilized to encode categorical variables, involving type of failure, type of pipeline, strategy of retraining, level of monitoring and allow them to be subject to computational procedures. Numerical attributes were standardized with the assistance of StandardScaler increasing convergence of models in minimizing difference within feature magnitude.

The scaling procedure was shown as:

$$X' = \frac{x-\mu}{\sigma}$$

where X′ shows the transformed feature, μ showing the mean value, as well as σ indicates standard deviation. The data was split into training as well as testing subsets utilizing 80:20 split in evaluate model generalization. Feature-target separation was done as:

$$D= \{(X_i, Y_i)\}_{i=1}^{n}$$

where X_i shows pipeline characteristics as well as Y_i shows recovery status. Preprocessing steps assure data consistency, minimize bias, and prepare data for developing XGBoost, Random Forest, as well as DNN approaches for resilient prediction of the pipeline failure.

Flowchart



Fig 6. Proposed Flowchart Diagram

High-throughput pipelines operation flow starts with the data ingestion as well as processing [24]. This illustrates failure detection framework, applying of failure-recovery mechanism, failure-retry approaches, failure-backoff, assures system availability over failure, fault-tolerance and stable pipeline operation.

Pseudocode

```
Start
Input pipeline data and execution parameters
Monitor pipeline operations for potential failures
If failure detected, apply retry policy
Execute backoff strategy before retry attempt
If retry succeeds, resume pipeline processing
Else activate recovery or fallback mechanism
Record failure logs and performance metrics
Update resilience status
End
```

Fig 7. Pseudocode Structure

Pseudocode is an automated resilience workflow of high-throughput pipeline monitoring functioning and acts as a reaction to failures [25]. Status updates as well as logging are helpful to inform the reliability, optimization of recovery decisions and the fault-tolerant performance of pipeline in case of disrupted operation.

Model Training and Model Evaluation

Two machine learning approaches, which are XGBoost as well as Random Forest, and "Deep Neural Network (DNN)" were trained on preprocessed dataset, in predicting outcomes of pipeline failure recovery. Training of models was performed on a train-test split of 80:20 with evaluation based on precision, accuracy, recall, F1-score as well as ROC-AUC. The prediction objective was shown as:

$$\hat{Y} = f(X;\theta)$$

where X shows pipeline resilience features as well as θ indicates model parameters.

Table 2: Summary Of Developed Predictive Models

Model	Learning Approach	Evaluation Metrics
Random Forest	Ensemble Classification	Accuracy, F1, ROC-AUC
XGBoost	Boosting Classification	Accuracy, Precision, Recall
DNN	Deep Learning Prediction	Accuracy, Loss, ROC-AUC

The respective evaluation procedure recognizes adequate resilience trends for adequate failure recovery.

Ethical Consideration

Resilient high-throughput pipeline frameworks need to be designed with ethical considerations; concerning safety of data and also privacy, and accountable automation [26]. The study makes sure that there is a security in the procedure of processing pipeline data without revealing sensitive operation data. In the context of model-based failure prediction methods, transparency and an aversion to bias as well as decision-making should be acceptable and human controls in critical recovery activities encouraged [27]. Monitoring information needs to be utilized responsibly, properly stored and be implemented to assure that information is not misused while the system is reliable and trusted by organizations [28].

IV. RESULTS AND DISCUSSION

Seconds	Recovery_Time_Seconds	Circuit_Breaker_Status	Monitoring_Level	Resource_Utilisation_Percentage	Scalability_Level	Pipeline_Reliability_Score	Failure_Recovery_Status
64.55	564.67	Disabled	Advanced	77.09	High	24.10	Recovered
100.24	72.11	Disabled	Advanced	83.48	Medium	24.05	Recovered
70.92	392.31	Enabled	Intermediate	25.58	Medium	90.51	Recovered
52.75	260.95	Enabled	Intermediate	41.79	Medium	64.69	Recovered
34.15	229.25	Disabled	Intermediate	81.62	Low	5.38	Recovered
...	...	...	...	...	...	...	...
42.81	532.33	Enabled	Advanced	86.30	Low	3.14	Failed
83.57	303.58	Enabled	Basic	74.30	Low	19.25	Failed
118.57	366.18	Enabled	Intermediate	84.01	Low	61.50	Recovered
11.00	405.32	Disabled	Intermediate	87.70	Medium	44.64	Recovered
50.65	492.37	Enabled	Intermediate	98.42	Medium	41.38	Failed

Fig 8. Dataset Sample Overview

Overview of the data shows typical records of the data of high-throughput pipeline resilience generated. Vital features involving status of circuit breaker, recovery time, monitoring level, reliability score, resource usage, scalability, and also result of recovery.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Pipeline_ID                     5000 non-null   object
 1   Pipeline_Type                   5000 non-null   object
 2   Workload_Size_GB                5000 non-null   float64
 3   Processing_Throughput_MBps      5000 non-null   float64
 4   Execution_Time_Minutes          5000 non-null   float64
 5   Failure_Type                    5000 non-null   object
 6   Failure_Frequency_Per_Day       5000 non-null   int64
 7   Retry_Strategy                  5000 non-null   object
 8   Maximum_Retry_Attempts          5000 non-null   int64
 9   Average_Retry_Delay_Seconds     5000 non-null   float64
 10  Recovery_Time_Seconds           5000 non-null   float64
 11  Circuit_Breaker_Status          5000 non-null   object
 12  Monitoring_Level                5000 non-null   object
 13  Resource_Utilisation_Percentage 5000 non-null   float64
 14  Scalability_Level               5000 non-null   object
 15  Pipeline_Reliability_Score      5000 non-null   float64
 16  Failure_Recovery_Status         5000 non-null   object
dtypes: float64(7), int64(2), object(8)
memory usage: 664.2+ KB
```

Fig 9. Dataset Structure and Feature Information

The figure depicts a dataset with 5,000 records and 17 attributes assuring that there are numbers and categorical variables indicating failure behaviors, work load features, and retries required within predictive modelling.

	0
Pipeline_ID	0
Pipeline_Type	0
Workload_Size_GB	0
Processing_Throughput_MBps	0
Execution_Time_Minutes	0
Failure_Type	0
Failure_Frequency_Per_Day	0
Retry_Strategy	0
Maximum_Retry_Attempts	0
Average_Retry_Delay_Seconds	0
Recovery_Time_Seconds	0
Circuit_Breaker_Status	0
Monitoring_Level	0
Resource_Utilisation_Percentage	0
Scalability_Level	0
Pipeline_Reliability_Score	0
Failure_Recovery_Status	0

dtype: int64

Fig 10. Missing Value Analysis

The missing value checking suggests that 5,000 records provide attributes in the dataset with no missing values. This defines that there is a high level of data completeness and the need to use imputation approaches is obviated.

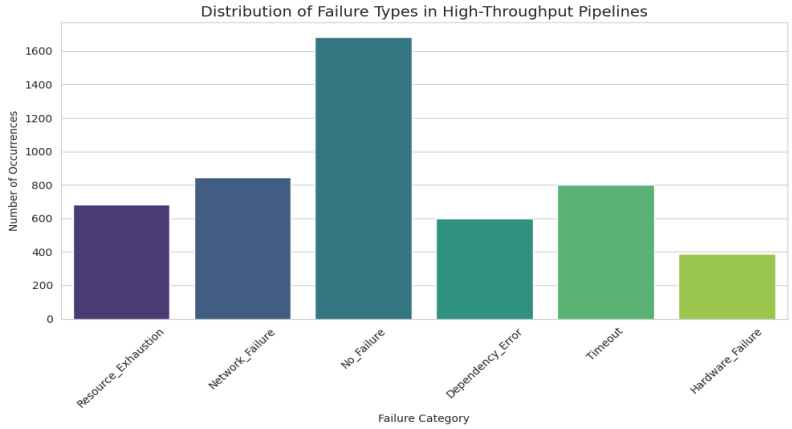


Fig 11. Distribution of Failure Types in High-Throughput Pipelines

Distribution of Failure Types visualization determines the key failure types impacting the operations of pipelines. Highest frequency is the no-failure cases and the rest of the operational risks are indicated by network failures, timeouts, dependency errors, resource exhaustion and hardware failures.

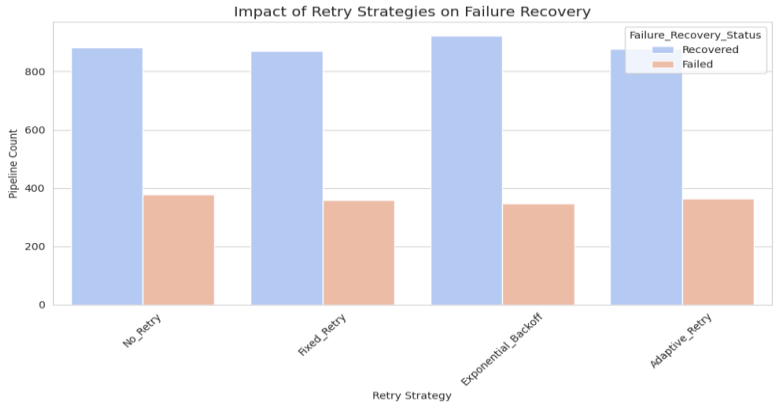


Fig 12. Impact of Retry Strategies on Failure Recovery

The adaptive retry, exponential backoff, fixed as well as no retry approaches exhibit differences in successful recovery rates. The discussion underscores the benefits of smart retry functions to enhance fault tolerance as well as availability of the pipeline.

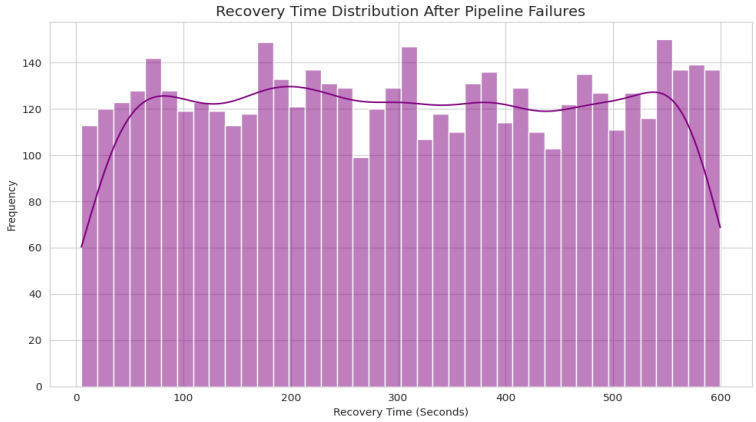


Fig 13. Recovery Time Distribution After Pipeline Failures

The illustration demonstrates the distribution of time taken in recovering after disruption of pipelines. The distribution shows recovery times are varying amongst events of failures implying that resilience mechanisms and recovery plans have an impact.

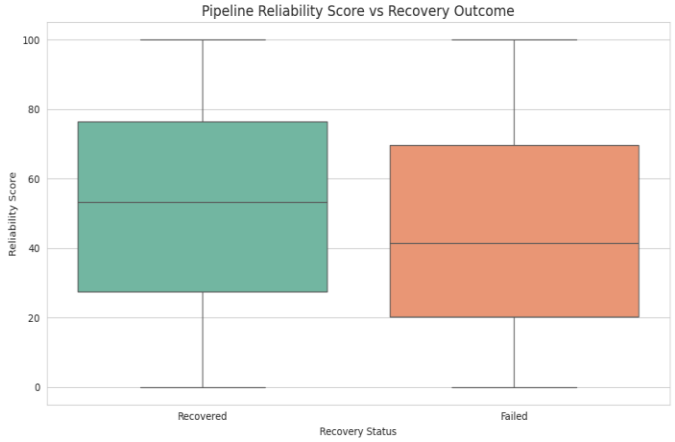


Fig 14. Pipeline Reliability Score vs Recovery Outcome

The developed chart demonstrates recovery outcomes as well as reliability scores of recovered operations and failed operations of pipelines. The cases that are recovered tend to have more reliability traits whereas the cases presenting a failure exhibit relatively lower levels of reliability.

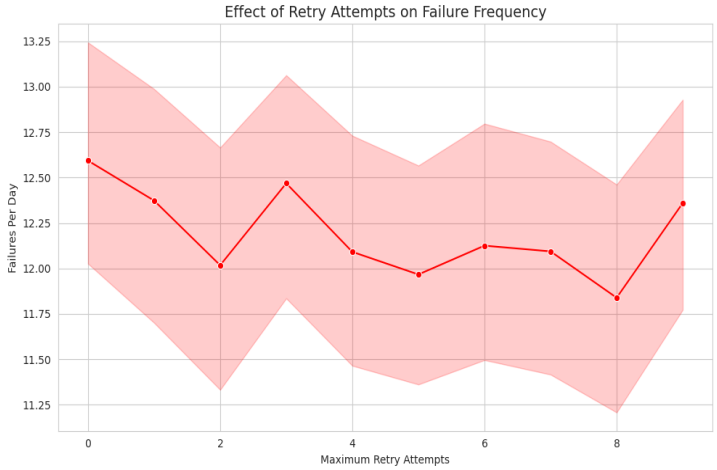


Fig 15. Effect of Retry Attempts on Failure Frequency

The trend shows the changes in frequencies of failures in weaknesses with respect to the failure limits of each retry that too many or too few retries can be adequate in impacting efficiency of the system. Retry configuration needs to be adequate to balance between recovery success as well as utilization of resources.

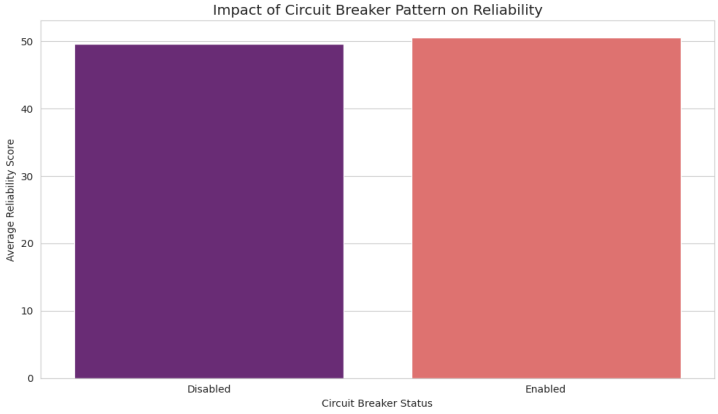


Fig 16. Impact of Circuit Breaker Pattern on Reliability

The figure assesses the level to which implementation of circuit breakers has an effect on the scores of pipe reliability. The enabled circuit breakers have a less good reliability than the disabled setup.

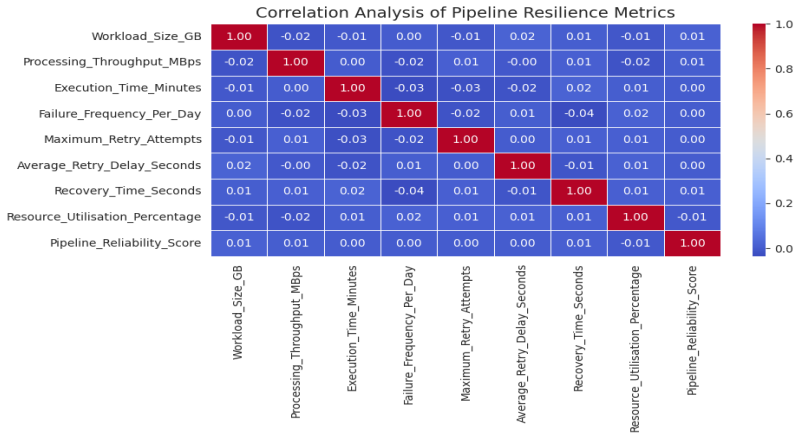
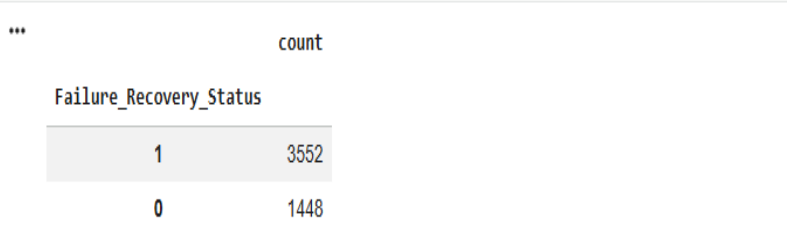


Fig 17. Correlation Analysis of Pipeline Resilience Metrics

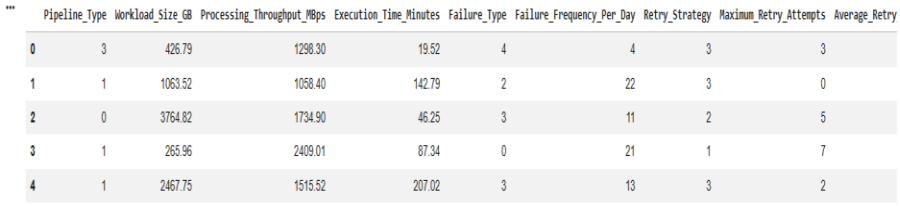
The relationships between the characteristics of the pipes numerically are shown in the mentioned figure. The heatmap determines interactions of features in terms of throughput, workload size, execution time, recovery duration, resource usage as well as reliability score.



Failure_Recovery_Status	count
1	3552
0	1448

Fig 18. Failure Recovery Status Distribution

The mentioned figure shows the target variable distribution with the recovery outcomes having been encoded. The data involving 3,552 recovered as well as 1,448 failed, defines that there is a medium class imbalance.



	Pipeline_Type	Workload_Size_GB	Processing_Throughput_MBps	Execution_Time_Minutes	Failure_Type	Failure_Frequency_Per_Day	Retry_Strategy	Maximum_Retry_Attempts	Average_Retry
0	3	426.79	1298.30	19.52	4	4	3	3	
1	1	1063.52	1058.40	142.79	2	22	3	0	
2	0	3764.82	1734.90	46.25	3	11	2	5	
3	1	265.96	2409.01	87.34	0	21	1	7	
4	1	2467.75	1515.52	207.02	3	13	3	2	

Fig 19. Preprocessed Feature Dataset Representation

Transformed input variables represented as numeric are displayed, where categorical attributes were encoded. Processed characteristics are the pipeline attributes, failure attributes, workload settings, retry settings as well as operational contexts.

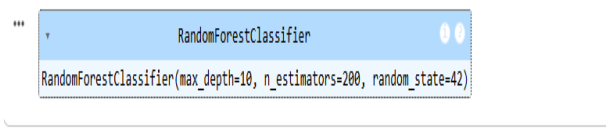


Fig 20. Random Forest Model Training Configuration

Training Configuration of the Random Forest approach shows an ensemble learning approach utilized with specific hyper parameters of maximum depth along with a number of estimators.

Classification Report

```
rf_accuracy = accuracy_score(
    y_test,
    rf_prediction
)
print(
    classification_report(
        y_test,
        rf_prediction
    )
)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       290
           1       0.71      1.00      0.83       710

    accuracy                           0.71      1000
   macro avg       0.35      0.50      0.42      1000
weighted avg       0.50      0.71      0.59      1000
```

Fig 21. Random Forest Classification Report

The Classification Report of the Random Forest shows performance of a model to predict pipeline failure recovery. The model had an 71% accuracy rate having higher precision (0.71) as well as recall (1.00) along with F1-value (0.83) within the class 1 recovery detection.

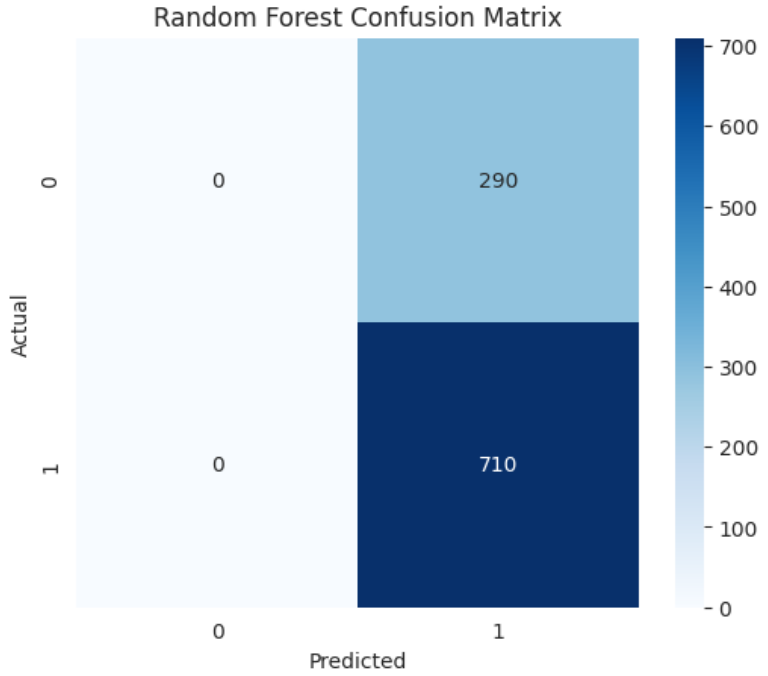


Fig 22. Random Forest Confusion Matrix

Confusion matrix of the random forests on prediction of recovered and failed pipeline states is depicted. The model was able to classify most of the recoveries 710 as well as failed cases 290 while at the cost of wrongly listing failed cases as recovered.

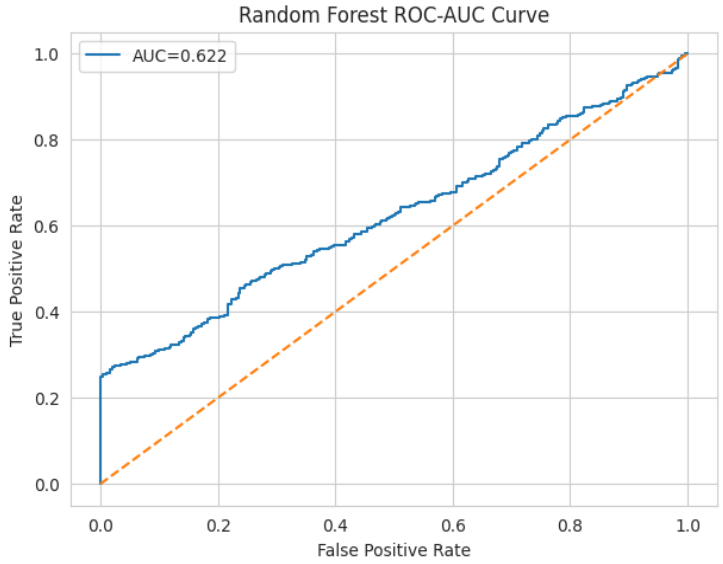


Fig 23. Random Forest ROC-AUC Curve

ROC-AUC Curve of the Random Forest measure's ability to discriminate among failure as well as recovery results. The particular model was associated with a moderate predictive ability depicted by an AUC value of 0.622. The respective curve shows that Random Forest has an ability to reveal resilience trends within random classification requiring optimizations to enhance resilience failure detection.

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=True, eval_metric=None, feature_types=None,
              feature_weights=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.05, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=200,
              n_jobs=None, num_parallel_tree=None, ...)
```

Fig 24. XGBoost Model Training Configuration

The Training Configuration of the XGBoost approach shows applied parameters of boosting classifiers applied in prediction of pipeline recovery. The model utilizes 200 estimators, learning rate 0.05 as well as depth 5 to learn intricate links among features of failure, retries as well as resilience components on performance of high-throughput pipelines.

	precision	recall	f1-score	support
0	0.39	0.07	0.12	290
1	0.72	0.95	0.82	710
accuracy			0.70	1000
macro avg	0.55	0.51	0.47	1000
weighted avg	0.62	0.70	0.62	1000

Fig 25. XGBoost Classification Report

The Classification Report of the XGBoost indicates that there is better recognition of failure in comparison to Random Forest. The particular model was found in having 70% accuracy, recovery classification obtaining 0.72 along with recall 0.95 as well as F1-score 0.82. Recall was low 0.07, which suggests that it is not an easy task in detecting unsuccessful recovery contexts.

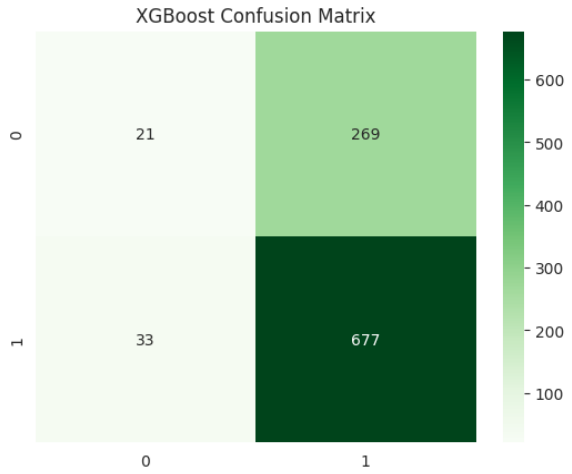


Fig 26. XGBoost Confusion Matrix

The Confusion Matrix of the XGBoost demonstrates the results of prediction of the state of the pipeline recovery. The model forecasted correctly 677 recovered cases, as well as 21 lost cases, along with provided 269 false recovery predictions and 33 lost recoveries. This shows superior failure identification capability over Random Forest while showing classification imbalance problems.

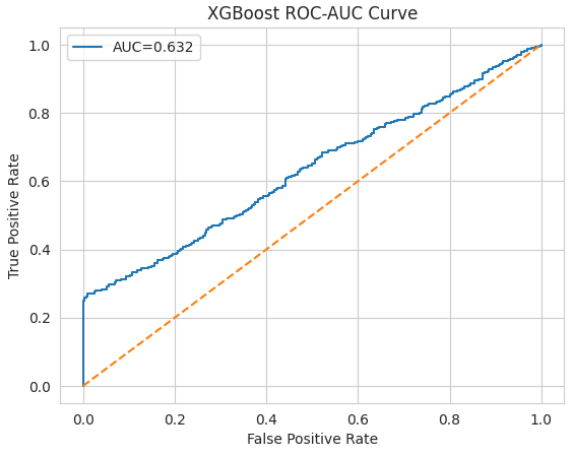
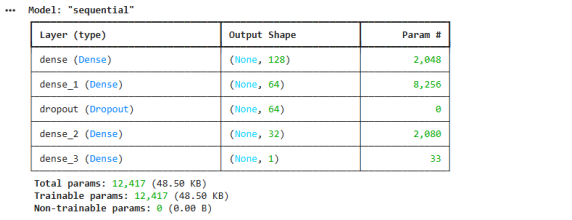


Fig 27. XGBoost ROC-AUC Curve

The ROC-AUC Curve of XGBoost shows that the model can differentiate among the unsuccessful as well as successful pipeline recovery. The obtained AUC about 0.632 is a better predictive power than the Random Forest.



Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	2,048
dense_1 (Dense)	(None, 64)	8,256
dropout (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

Total params: 12,417 (48.50 KB)
Trainable params: 12,417 (48.50 KB)
Non-trainable params: 0 (0.00 B)

Fig 28. Deep Neural Network Architecture Summary

Architecture Summary of the Deep Neural Network shows utilized DNN architecture in prediction of failures recovery. The layers have 128, 64 as well as 32 neurons within the network as well as a sigmoid output layer.

```
Epoch 1/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.7078 - loss: 0.6052 - val_accuracy: 0.7000 - val_loss: 0.5985
Epoch 2/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7134 - loss: 0.5767 - val_accuracy: 0.7000 - val_loss: 0.5888
Epoch 3/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7119 - loss: 0.5682 - val_accuracy: 0.7000 - val_loss: 0.5799
Epoch 4/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7119 - loss: 0.5594 - val_accuracy: 0.7000 - val_loss: 0.5761
Epoch 5/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7147 - loss: 0.5515 - val_accuracy: 0.6988 - val_loss: 0.5755
Epoch 6/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7144 - loss: 0.5410 - val_accuracy: 0.6988 - val_loss: 0.5788
Epoch 7/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7125 - loss: 0.5410 - val_accuracy: 0.7025 - val_loss: 0.5748
Epoch 8/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7197 - loss: 0.5311 - val_accuracy: 0.7050 - val_loss: 0.5750
Epoch 9/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7178 - loss: 0.5267 - val_accuracy: 0.7013 - val_loss: 0.5821
Epoch 10/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7266 - loss: 0.5161 - val_accuracy: 0.7013 - val_loss: 0.5810
Epoch 11/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7350 - loss: 0.5067 - val_accuracy: 0.6963 - val_loss: 0.5861
Epoch 12/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7356 - loss: 0.5044 - val_accuracy: 0.6888 - val_loss: 0.5948
Epoch 13/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7347 - loss: 0.5012 - val_accuracy: 0.6888 - val_loss: 0.6014
Epoch 14/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7434 - loss: 0.4930 - val_accuracy: 0.6787 - val_loss: 0.5951
Epoch 15/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7434 - loss: 0.4812 - val_accuracy: 0.6800 - val_loss: 0.5970
Epoch 16/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7563 - loss: 0.4817 - val_accuracy: 0.6775 - val_loss: 0.6111
Epoch 17/60
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7634 - loss: 0.4662 - val_accuracy: 0.6475 - val_loss: 0.6205
Epoch 18/60
```

Fig 29. Deep Neural Network Training Performance

Training Performance of the "Deep Neural Network" shows model learning behaviors within 60 epochs. There was an enhancement within training accuracy, having no change in validation, with a validation accuracy about 70% being obtained throughout training.

	precision	recall	f1-score	support
0	0.34	0.24	0.28	290
1	0.72	0.81	0.76	710
accuracy			0.64	1000
macro avg	0.53	0.52	0.52	1000
weighted avg	0.61	0.64	0.62	1000

Fig 30. Deep Neural Network Classification Report

The Classification Report of the "Deep Neural Network" is a measure of performance of the recovery prediction. The model had accuracy of 64% having an accuracy rate of 0.72 within the class 1 recovery prediction with a recall of 0.81 along with F1-score of 0.76. DNN shows reduced usefulness within classification, compared to ensemble models, due to lack of feature complexity as well as characteristics of the dataset

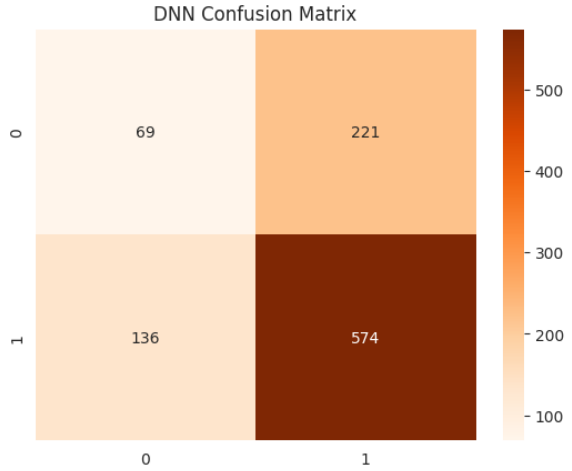


Fig 31. Deep Neural Network Confusion Matrix

Confusion Matrix of the Deep Neural Network indicates outcomes of prediction in classes of recovery procedure. The model was capable of correctly classifying 574 recovered as well as 69 failed cases and misclassified 221 failures as recoveries.

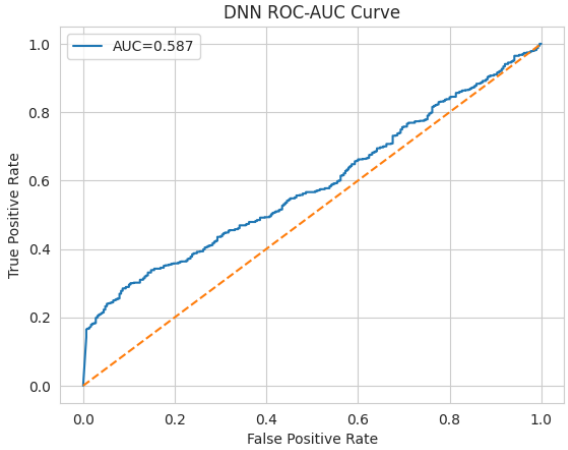


Fig 32. Deep Neural Network ROC-AUC Curve

The ROC-AUC Curve of the "Deep Neural Network" is utilized to determine measurement of capability of classification separation between pipeline recovery states. The AUC of the model was 0.587, which is not as high as that of the XGBoost as well as Random Forest.

	Model	Accuracy	ROC-AUC
0	Random Forest	0.710	0.622069
1	XGBoost	0.698	0.631782
2	DNN	0.643	0.586940

Fig 33. Model Comparison Results

The comparison outcomes table shows performance of implemented approaches according to overall prediction. Random Forest had the maximum accuracy of 0.710 as well as XGBoost had the maximum ROC-AUC 0.632. Accuracy as well as ROC-AUC of DNN were 0.643 as well as 0.587, suggesting that ensemble models give better predictions within the analysis of pipeline failure recovery.

Discussion

The findings indicate that ensemble learning approaches are able to deliver the best analysis of pipeline recovery trends, XGBoost has the highest ROC-AUC along with Random Forest is the most accurate. There was an absence of balance of classes in failure detection. The results of studies underscore significance of adaptive resilience measures in high throughput management of pipelines within a dependable manner.

RESEARCH LIMITATIONS AND FUTURE DIRECTIONS

Research Limitation

The research is restricted through a simulated dataset utilized as opposed to actual pipeline logs inhibiting a realistic generalization. Failure detectors were sensitive to class imbalance, and inability to grasp more composite operational resilience behaviors was due to lack of diverse features [29].

Future Direction

The research can be enhanced in future studies to involve real-time industrial pipeline data as well as group adaptive learning approaches utilized to enhance prediction of future outcomes. Hybrid models, explainable artificial intelligence as well as reinforcement learning techniques can be possible to design intelligent resilience mechanisms within a dynamic distributed high-throughput contexts [30].

CONCLUSION

The study examined resilience engineering techniques to enhance recovery of failure following high-throughput pipeline frameworks utilizing retry, fault management and predictive modelling approaches. The experimental findings have been that accuracy of Random Forest was the most accurate 71% whereas XGBoost had the greatest ROC-AUC about 0.632. The outcomes have shown that resilience trends that are intelligent can assist improve reliability of pipelines; more optimizations need to be done to assist predict failure more precisely.

REFERENCES

[1] Li, T., Chandramouli, B., Faleiro, J.M., Madden, S. and Kossmann, D., 2021, June. Asynchronous prefix recoverability for fast distributed stores. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 1090-1102).

[2] Taft, R., Sharif, I., Matei, A., VanBenschoten, N., Lewis, J., Grieger, T., Niemi, K., Woods, A., Birzin, A., Poss, R. and Bardea, P., 2020, June. Cockroachdb: The resilient geo-distributed sql database. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data* (pp. 1493-1509).

[3] O'Keeffe, D., Salonidis, T. and Pietzuch, P., 2018. Frontier: Resilient edge processing for the internet of things. *Proceedings of the VLDB Endowment*, *11*(10), pp.1178-1191.

[4] Goldstein, J., Abdelhamid, A., Barnett, M., Burckhardt, S., Chandramouli, B., Gehring, D., Lebeck, N., Meiklejohn, C., Minhas, U.F., Newton, R. and Peshawaria, R.G., 2020. Ambrosia: Providing performant virtual resiliency for distributed applications. *Proceedings of the VLDB Endowment*, *13*(5), pp.588-601.

[5] Benacchio, T., Bonaventura, L., Altenbernd, M., Cantwell, C.D., Düben, P.D., Gillard, M., Giraud, L., Göddeke, D., Raffin, E., Teranishi, K. and Wedi, N., 2021. Resilience and fault tolerance in high-performance computing for numerical weather and climate prediction. *The International Journal of High Performance Computing Applications*, *35*(4), pp.285-311.

[6] Distler, T., 2021. Byzantine fault-tolerant state-machine replication from a systems perspective. *ACM Computing Surveys (CSUR)*, *54*(1), pp.1-38.

[7] Kim, D., Nelson, J., Ports, D.R., Sekar, V. and Seshan, S., 2021, August. Redplane: Enabling fault-tolerant stateful in-switch applications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (pp. 223-244).

[8] Correia, A., Felber, P. and Ramalhete, P., 2020, April. Persistent memory and the rise of universal constructions. In *Proceedings of the Fifteenth European Conference on Computer Systems* (pp. 1-15).

[9] Gog, I., Isard, M. and Abadi, M., 2021, November. Falkirk wheel: Rollback recovery for dataflow systems. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 373-387).

[10] Suri-Payer, F., Burke, M., Wang, Z., Zhang, Y., Alvisi, L. and Crooks, N., 2021, October. Basil: Breaking up BFT with ACID (transactions). In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (pp. 1-17).

[11] Mesbahi, M.R., Rahmani, A.M. and Hosseinzadeh, M., 2018. Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Computing and Information Sciences*, *8*(1), p.20.

[12] Kumari, P. and Kaur, P., 2021. A survey of fault tolerance in cloud computing. *Journal of King Saud University Computer and Information Sciences*, *33*(10), pp.1159-1176.

[13] Singhvi, A., Akella, A., Anderson, M., Cauble, R., Deshmukh, H., Gibson, D., Martin, M.M., Strominger, A., Wenisch, T.F. and Vahdat, A., 2021, August. CliqueMap: productionizing an RMA-based distributed caching system. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (pp. 93-105).

[14] Manchana, R., 2021. Event-Driven Architecture: Building Responsive and Scalable Systems for Modern Industries. *International Journal of Science and Research (IJSR)*, *10*(1), pp.1706-1716.

[15] Fu, Y. and Soman, C., 2021, June. Real-time data infrastructure at uber. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2503-2516).

[16] Posey, B., Ngo, L.B., Chowdhury, M. and Apon, A., 2018. Infrastructure for transportation cyber-physical systems. In *Transportation Cyber-Physical Systems* (pp. 153-171). Elsevier.

[17] Taleb, M.Y., 2018. *Optimizing Distributed In-memory Storage Systems: Fault-tolerance, Performance, Energy Efficiency* (Doctoral dissertation, École normale supérieure de Rennes).

[18] Jensen, C., Howard, H. and Mortier, R., 2021, April. Examining Raft's behaviour during partial network failures. In *Proceedings of the 1st Workshop on High Availability and Observability of Cloud Systems* (pp. 11-17).

[19] Zhou, J., Xu, M., Shraer, A., Namasivayam, B., Miller, A., Tschannen, E., Atherton, S., Beamon, A.J., Sears, R., Leach, J. and Rosenthal, D., 2021, June. Foundationdb: A distributed unbundled transactional key value store. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2653-2666).

[20] Li, B., Zuo, G., Bai, W. and Zhang, L., 2021, August. 1pipe: Scalable total order communication in data center networks. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (pp. 78-92).

[21] Arasu, A., Chandramouli, B., Gehrke, J., Ghosh, E., Kossmann, D., Protzenko, J., Ramamurthy, R., Ramananandro, T., Rastogi, A., Setty, S. and Swamy, N., 2021, June. Fastver: Making data integrity a commodity. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 89-101).

[22] Wei, X., Chen, R., Chen, H. and Zang, B., 2021. Xstore: Fast rdma-based ordered key-value store using remote learned cache. *ACM Transactions on Storage (TOS)*, *17*(3), pp.1-32.

[23] Park, S.Y., Faraci, G., Murphy, G., Pilcher, C., Busch, M.P. and Lee, H.Y., 2021. Microdrop human immunodeficiency virus sequencing for incidence and drug resistance surveillance. *The Journal of infectious diseases*, *224*(6), pp.1048-1059.

[24] Cao, W., Zhang, Y., Yang, X., Li, F., Wang, S., Hu, Q., Cheng, X., Chen, Z., Liu, Z., Fang, J. and Wang, B., 2021, June. Polardb serverless: A cloud native database for disaggregated data centers. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 2477-2489).

[25] Arasu, A., Chandramouli, B., Gehrke, J., Ghosh, E., Kossmann, D., Protzenko, J., Ramamurthy, R., Ramananandro, T., Rastogi, A., Setty, S. and Swamy, N., 2021, June. Fastver: Making data integrity a commodity. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 89-101).

[26] Abbo, L., Shukla, B.S., Giles, A., Aragon, L., Jimenez, A., Camargo, J.F., Simkins, J., Sposato, K., Tran, T.T., Diaz, L. and Reyes, J., 2019. Linezolid-and vancomycin-resistant Enterococcus faecium in solid organ transplant recipients: infection control and antimicrobial stewardship using whole genome sequencing. *Clinical Infectious Diseases*, *69*(2), pp.259-265.

[27] Garg, M., Peluso, S., Arun, B. and Ravindran, B., 2019, December. Generalized consensus for practical fault tolerance. In *Proceedings of the 20th International Middleware Conference* (pp. 55-67).

[28] Willsey, M., Stephenson, A.P., Takahashi, C., Vaid, P., Nguyen, B.H., Piszczek, M., Betts, C., Newman, S., Joshi, S., Strauss, K. and Ceze, L., 2019, April. Puddle: A dynamic, error-correcting, full-stack microfluidics platform. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 183-197).

[29] Saha, S.K., Aggarwal, S., Pathak, R., Koutsonikolas, D. and Widmer, J., 2019, October. MuSher: An agile multipath-TCP scheduler for dual-band 802.11 ad/AC wireless LANs. In *The 25th Annual International Conference on Mobile Computing and Networking* (pp. 1-16).

[30] Liu, H., Ding, W., Chen, Y., Guo, W., Liu, S., Li, T., Zhang, M., Zhao, J., Zhu, H. and Zhu, Z., 2019, June. Cfs: A distributed file system for large scale container platforms. In *Proceedings of the 2019 International Conference on Management of Data* (pp. 1729-1742).