



ASSURANT®

FINAL REPORT 2024

Aditya Hegde
Muhamad Imannulhakim
Sarthak Khare
Kyrylo Kobzyev
Joseph Lee
Yang Lu
Priyanka Singh



Table of Contents

Introduction	2
Timeline and Tasks	4
Architecture Diagram	5
Data Collection and Preparation	6
Exploratory Data Analysis (EDA)	6
Unsupervised Models	8
Supervised Models	13
Language Models	16
Observed Issues with the Data	18
1. Correlating Failures with other Log Entries	18
2. Identifying SOC Non-Compliance	18
3. Lack of Text Fields for a meaningful NLP/LLM solution	18
4. Truncated Data - Not reliable for Trends	19
Results, Evaluation and Model Selection	19
Implementation	19
Integration of Models into a System	19
API Development and Usage	20
Conclusion and Recommendations	21
Appendix	21

Introduction

Assurant, Inc. is a global provider of risk management solutions, operating in various segments, including housing, lifestyle, and financial services. Founded in 1892 and headquartered in Atlanta, Assurant offers a range of insurance products and services. These include extended service contracts, mobile device protection, vehicle protection, pre-funded funeral insurance, renters insurance, and lender-placed insurance.

Assurant focuses on helping clients manage and protect their investments in properties, appliances, electronics, and other valuable assets. The company operates through a customer-centric approach, leveraging technology and data analytics to deliver innovative and tailored solutions. Assurant has a presence in over 20 countries and partners with leading financial institutions, retailers, and mobile carriers to provide comprehensive risk management services.

Assurant is 365 on the Fortune 500 list of the largest companies in the United States by revenue as of 2024.

Assurant uses Microsoft Azure's Cloud Services for its operations. Since Microsoft manages users' sensitive information (on Assurant's behalf), it must provide structured documentation detailing how it protects that information.

These are done through SOC® examinations. SOC stands for System and Organization Controls. It's an examination geared toward entities that provide services directly related to a user's control systems, like SaaS companies, financial reporting organizations, data centers, and payment processors.

SOC 1 focuses on a service organization's controls related to financial reporting. Entities utilizing these service organizations might request a SOC 1 report to assess how the controls impact their own financial statements. This is crucial for both the entities and the CPAs auditing their financial statements.

SOC 2 evaluates a service organization's controls using five criteria: security, availability, processing integrity, confidentiality, and privacy. A wide range of users might request this report to obtain detailed information and assurance about the service organization's controls concerning 1) the security, availability, and processing integrity of the systems used to process users' data and 2) the confidentiality and privacy of the information processed by these systems.

Since Microsoft has measures in place to ensure SOC compliance, certain actions attempted by Assurant's IT team to Create, Enhance, or Modify Existing Data Pipelines and Assets are rejected because they require compliance.

Assurant, as an Organization, wants to prevent any requests or actions that will be rejected by Microsoft's Data Policies.

Our solution to this problem is to develop a model that can classify proposed actions as potential failures. This will reduce the time needed to identify, diagnose, and repair incidents in IT systems using machine learning.

We are focusing on:

- Developing predictive analytics
- Conducting causal analysis
- Implementing automated remediation
- Establishing a CI/CD pipeline with automated logging and a learning loop

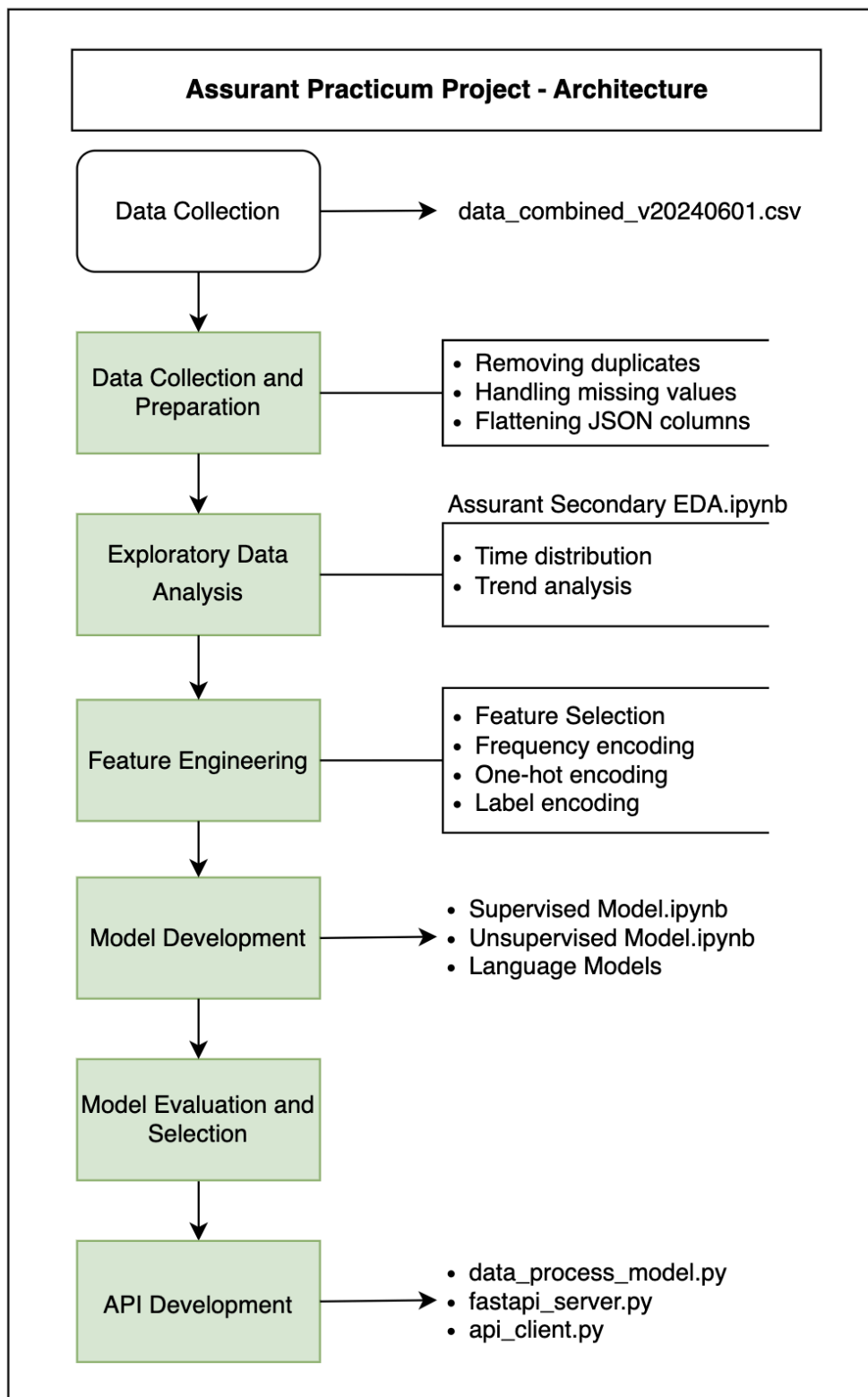
The expected deliverables include:

- ML models for predictive analytics and event correlation
- Automated remediation systems
- An integrated CI/CD pipeline
- An automated logging framework
- Comprehensive documentation and training materials
- Performance reports
- Future improvement plans

Timeline and Tasks

Week	Phase	Tasks	Assignees
May 27 - May 31, 2024	Business Understanding	Detailed Requirement Analysis	All
		Start Drafting Business Requirement Document (BRD)	All
		Data Quality Assessment	All
		Exploratory Data Analysis (EDA)	All
June 3 - June 7, 2024	Data Understanding and Preparation	Initial Data Cleaning and Preprocessing	Aditya, Iman, Kyrilo
		Advanced Data Cleaning and Preprocessing	Aditya, Iman, Kyrilo
		Feature Engineering	Sarthak, Priyanka, Lu
		Data Visualization	Sarthak, Lu, Joseph
		Finish Drafting Business Requirement Document (BRD)	All
June 10 - June 28, 2024	Unsupervised Model Development	Model Selection and Initial Training	Aditya, Iman, Joseph
		Model Evaluation and Tuning	Aditya, Iman, Joseph
	Supervised Model Development	Model Selection and Initial Training	Priyanka, Sarthak, Lu, Kyrilo
		Model Evaluation and Tuning	Priyanka, Sarthak, Lu, Kyrilo
	API Development	Initial API Development	Lu
July 1 - July 11, 2024	LLM Model and API Integration	LLM to Predict Success/Fail	Iman, Lu
		RAG Model to Predict SoC Violations	Sarthak, Kyrilo
		Finalize API for End-User Predictions	Sarthak, Lu
Ongoing: Throughout the Project	Documentation and Training Materials	Weekly Updates to Documentation	All
		GitHub Documentation	All
July 8 - July 11, 2024	Performance Reporting and Future Planning	Compile Performance Reports	All
		Develop Future Improvement Plans	All
		Final Review and Handover	All

Architecture Diagram



Data Collection and Preparation

Assurant provided us with the data. The data consisted of the log entries provided by Microsoft to Assurant's IT Team. The data itself consists of ~250K entries for a period spanning 18 months.

The data is in a Standard form that Microsoft provides to all its clients requesting their Logs. The data is relatively clean and standardized since Microsoft is one of the preeminent providers of Cloud Services to organizations worldwide. Additionally, data documentation is readily available online on the Microsoft Azure Help Section.

The data itself had very little missing data and hardly needed any cleaning. All we had to do was remove duplicates and a handful of rows that seemed to be an incorrect copy-paste job.

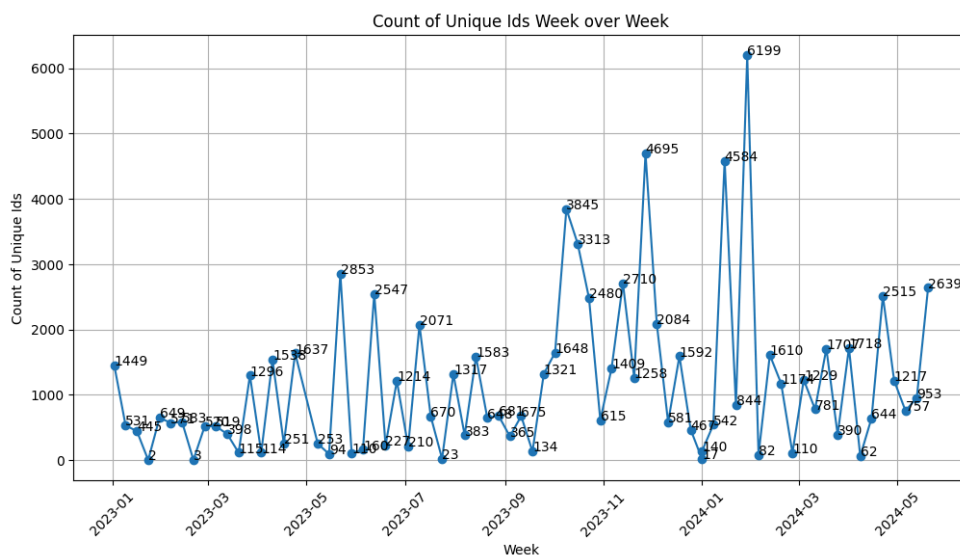
There was a JSON column, which needed to be flattened to access the relevant information, and that meant that it led to upwards of 200 columns with very sparse data due to the sheer variety of actions captured in the logs. Still, these were not missing data but instead were the nature of the dataset itself.

However, we came to understand other issues with the data after exploring it in depth.

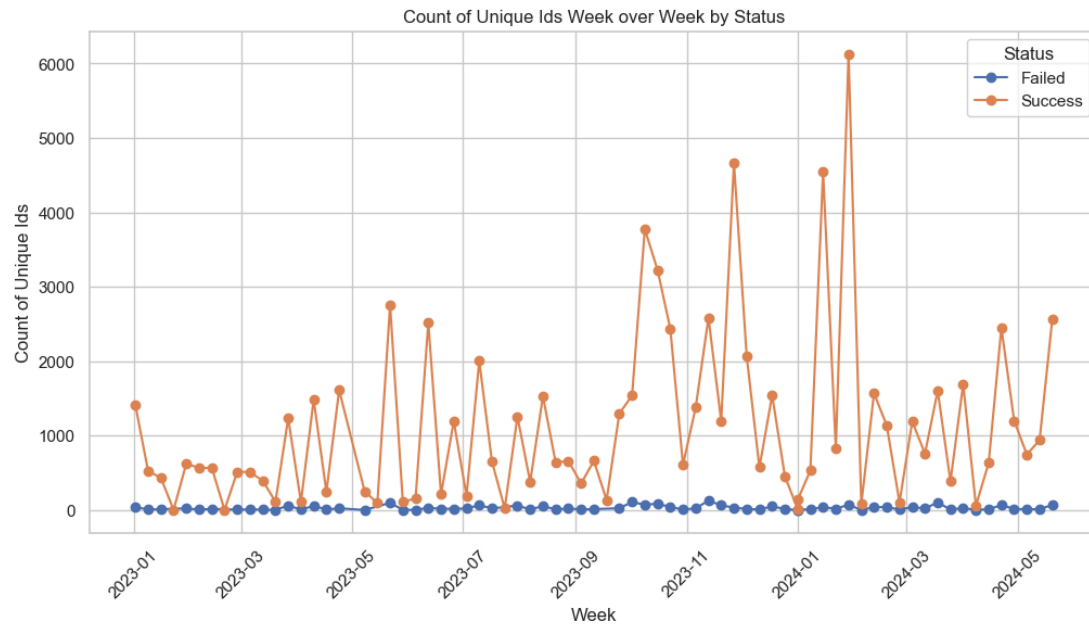
Exploratory Data Analysis (EDA)

We started with an EDA to understand the data. Here are some significant findings:

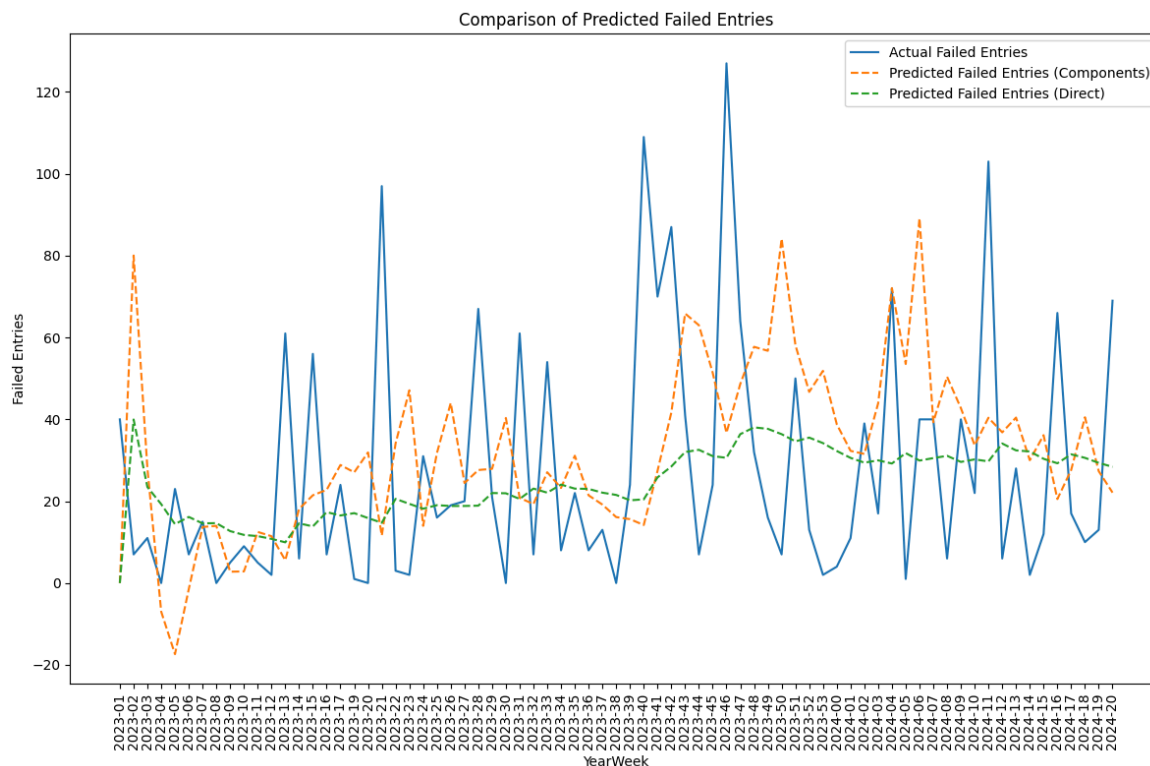
- About 20% of the data provided was duplicated, and the duplicate rows were removed from the data before further analysis.
- More than 50% of the data is from May 2024 alone, specifically between the 26th and 31st of May. To facilitate our analysis, we have divided the data into two distinct periods - before and after the 25th of May.
- Approximately 4.8% of "Execute" log entries are failures in the prior timeframe.
- All Failures are tagged as either "Pipelines" or "Release".



However there is no spike in Failures during the same time, and they remain relatively stable over time.



Trying to analyze the Failures in either absolute terms or by analyzing the total number of entries.



The count of Log entries week over week shows spikes October 2024 to January 2024.

Unsupervised Models

We initially attempted to use Unsupervised Models to cluster together similar log entries. The models that we used to predict Status are:

1. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN is a density-based clustering algorithm that identifies clusters based on the density of data points in a region. It groups points that are closely packed together, marking points in low-density regions as outliers. DBSCAN requires two parameters: eps (the maximum distance between two points to be considered neighbors) and minPts (the minimum number of points required to form a dense region). Unlike K-Means, DBSCAN can find clusters of arbitrary shape and is robust to noise, making it suitable for datasets with varying density and noise.
2. **K-Means Clustering:** K-Means Clustering is a popular partitioning method used to divide a dataset into K distinct, non-overlapping subsets or clusters. The algorithm works by initializing K centroids, then iteratively assigning each data point to the nearest centroid and updating the centroids to the mean of the assigned points. This process repeats until convergence, usually when assignments no longer change. K-Means is simple and efficient for large datasets but assumes spherical clusters and equal

variance, which may not always be the case. It is sensitive to the initial placement of centroids and may converge to a local optimum.

The Target Category for the Clustering Operations were **Data_DeploymentResult**, which had a granular view of the task statuses and **Status**, which had a binary Succeeded/Failed category.

The features were selected by inspecting and choosing only those features which seemed to have meaningful information, they are:

1. UserAgent
2. AuthenticationMechanism
3. ProjectId
4. ActorUserId
5. Data_RequesterId
6. IpAddress
7. TenantId
8. ScopeId
9. Data_DeploymentResult
10. Data_CallerProcedure
11. Data_Result
12. OperationName
13. Status
14. ActorClientId

We also used Transformation on certain columns to make them usable:

1. **One Hot Encoding:** One Hot Encoding is a method used to convert categorical variables into a binary matrix representation. Each category is represented by a binary vector where only one element is "1" (hot) and all others are "0". This technique is widely used because it does not assume any ordinal relationship between categories and is simple to implement. However, it can lead to high-dimensional data, especially if the categorical variable has many unique values, which can be computationally expensive and may cause issues like multicollinearity in some machine learning algorithms. Two Columns in our dataset were One Hot Encoded - **OperationName** and **Data_CallerProcedure**.
2. **Frequency Encoding:** Frequency Encoding is a technique used to convert categorical variables into numerical values based on the frequency of each category in the dataset. This method assigns each category a value corresponding to the number of times it appears in the dataset. It is particularly useful when the categorical variable has a high cardinality (i.e., many unique categories), as it reduces the dimensionality compared to other encoding techniques like One Hot Encoding. However, it may not capture the intrinsic relationships between categories as effectively as other methods. All Columns other than the Target and those that were One Hot Encoded were Frequency Encoded.

The results from both DBSCAN and K-Means Clustering were disappointing, since misclassifications were very high, this is not surprising since there are no good signals that can be used for prediction or clustering.

Apriori: Another model we tried implementing was the Apriori algorithm, an association rule learning method, aimed at identifying association rules indicating failures, characterized by high lift and confidence within our dataset. This approach was chosen for its ability to explore identifier associations in large datasets, aiming to detect frequent itemsets and association rules that might predict failures. High lift values suggest a positive association between the antecedent and consequent, while high confidence values denote a strong likelihood that the consequent (failure) occurs when the antecedent conditions are met. Identifying such rules was expected to provide actionable insights into potential predictive factors for failures.

Our exploratory data analysis (EDA) determined that failures occurred exclusively in operations named "Pipelines.DeploymentJobCompleted" and "Release.DeploymentCompleted", so we focused our analysis on the subset of data that included these specific operation names. To preprocess the data into a format suitable for the Apriori model, we first expanded the JSON contained in the data column into a DataFrame. Next, we added the columns 'ScopeDisplayName', 'ProjectName', 'IpAddress', 'UserAgent', and 'OperationName' to this DataFrame. After incorporating these details, we converted the data into a list of transactions, then one-hot encoded the transactions to create a DataFrame suitable for modeling. After running the model on this one-hot encoded DataFrame, we obtained the rules relevant to our analysis by filtering for rules where the consequent was 'Failed'.

	antecedents	consequents	support	confidence	lift
	(Pipelines.DeploymentJobCompleted)	(Failed)	0.116991	0.180026	0.99171
(Task.prc_UpdateEnvironmentDeploymentRequest)		(Failed)	0.116991	0.180026	0.99171
	(nan)	(Failed)	0.181530	0.181530	1.00000
(Task.prc_UpdateEnvironmentDeploymentRequest, ...		(Failed)	0.116991	0.180026	0.99171
(Pipelines.DeploymentJobCompleted, nan)		(Failed)	0.116991	0.180026	0.99171
(Task.prc_UpdateEnvironmentDeploymentRequest, ...		(Failed)	0.116991	0.180026	0.99171
(Task.prc_UpdateEnvironmentDeploymentRequest, ...		(Failed)	0.116991	0.180026	0.99171

The antecedents are features that precede and potentially influence the outcome. The consequents are the outcomes associated with the antecedents. A support of 0.116991 means that about 11.7% of the transactions (or data records) in the dataset contain this combination of antecedents and consequents. A confidence of 0.180026 means there is an 18% probability that the consequent occurs when the antecedent is present. From the rules generated, we observed that the lift values for all rules with failure as a consequence are 1 or very close to 1. This indicates that there is no meaningful association between the antecedents and the consequent, suggesting that the occurrence of failures is independent of the conditions captured by these rules. Essentially, the action resulting in failure happens just as frequently as would be expected by chance, implying no predictive value in the associations. These results highlight the challenges in using association rule learning for predicting failures in this specific context and

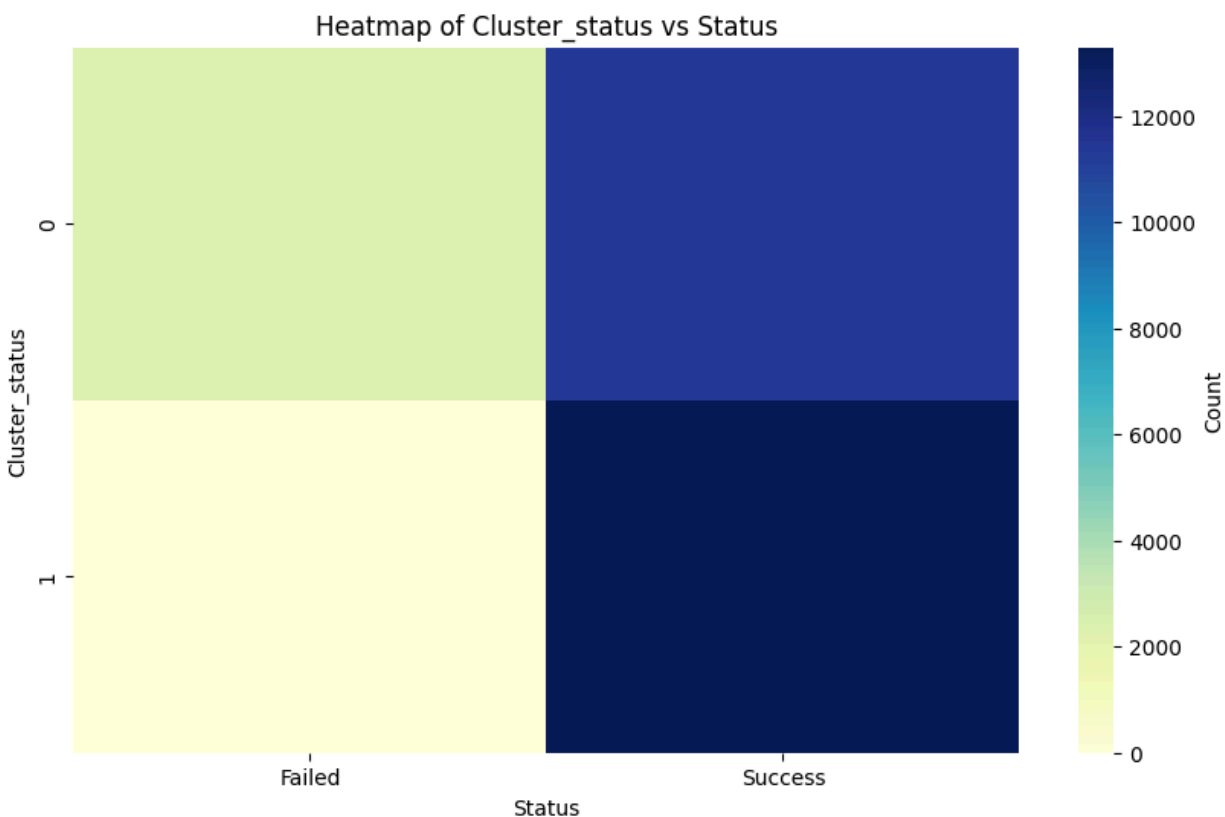
underscore the need for alternative approaches or additional data to better understand the underlying factors.

Performance Metrics for DBSCAN:

```
Adjusted Rand Index (ARI): 0.012299465429271348
Normalized Mutual Information (NMI): 0.054110431540893865
Homogeneity: 0.22280010063518466
Completeness: 0.030794698929310835
V-Measure: 0.054110431540893865
Fowlkes-Mallows Index: 0.39032536860182976
```

- **Adjusted Rand Index (ARI):** The ARI score ranges from -1 to 1, with 1 indicating perfect clustering and 0 indicating random clustering. A score of 0.0123 is very close to 0, suggesting that the clustering is not much better than random assignment.
- **Normalized Mutual Information (NMI):** The NMI score ranges from 0 to 1, with 1 indicating perfect correlation between the true labels and the predicted clusters. A score of 0.0541 is quite low, suggesting that the clusters do not share much information with the true labels.
- **Homogeneity:** Homogeneity measures if each cluster contains only members of a single class. A score of 0.2228 indicates that the clusters are not very homogeneous, meaning that clusters contain mixed classes.
- **Completeness:** Completeness measures if all members of a given class are assigned to the same cluster. A score of 0.0308 is very low, indicating that members of the same class are spread across multiple clusters.
- **V-Measure:** V-Measure is the harmonic mean of homogeneity and completeness. A score of 0.0541 is low, reaffirming that both homogeneity and completeness are poor.
- **Fowlkes-Mallows Index:** The Fowlkes-Mallows Index ranges from 0 to 1, with 1 indicating perfect clustering. A score of 0.3903 is moderate but not particularly strong, indicating some, but not strong, similarity between the predicted clusters and the true labels.

Performance Metrics for K-Means Clustering:



```
Adjusted Rand Index (ARI): 0.024813837099302762
Normalized Mutual Information (NMI): 0.12499734925028282
Homogeneity: 0.21214273991282134
Completeness: 0.08860117539849403
V-Measure: 0.12499734925028283
Fowlkes-Mallows Index: 0.6597162675460182
```

- **Adjusted Rand Index (ARI):** The ARI score is still close to 0, indicating that the clustering is only slightly better than random assignment. This is a slight improvement from the previous score of 0.0123 but still not strong.
- **Normalized Mutual Information (NMI):** The NMI score has improved from 0.0541 to 0.1250, which indicates a better correlation between the true labels and the predicted clusters. However, the score is still relatively low, suggesting that the clusters do not share a significant amount of information with the true labels.
- **Homogeneity:** The homogeneity score has slightly decreased from 0.2228 to 0.2121. This indicates that the clusters are still not very homogeneous, meaning that clusters contain mixed classes.

- **Completeness:** The completeness score has improved from 0.0308 to 0.0886. This indicates that more members of the same class are now being assigned to the same cluster, but the score is still quite low.
- **V-Measure:** The V-Measure, which is the harmonic mean of homogeneity and completeness, has improved from 0.0541 to 0.1250. This improvement shows better overall clustering performance, but it is still not very high.
- **Fowlkes-Mallows Index:** The Fowlkes-Mallows Index has significantly improved from 0.3903 to 0.6597. This indicates a better similarity between the predicted clusters and the true labels, suggesting that the clustering algorithm is doing a better job at grouping similar items together.

Supervised Models

We attempted to use supervised learning models to predict log entries that are likely to fail. The target variable indicating deployment result was obtained from the data column. Since deployment results are applicable for only the “Execute” category of logs, the model was trained on such entries. The models which were used for training are:

1. Random Forest Classifier:

Random forest classifier is an ensemble machine learning technique. The model consists of multiple decision trees each trained on randomly sampled subset of data and features. The model combines the prediction of individual decision trees in order to make a final prediction.

2. Gradient Boosting Classifier:

Gradient boosting classifier is also an ensemble machine learning model. In this model, decision trees are trained sequentially where each tree is trained to predict the residuals of the previous tree, thereby reducing error and enhancing the model accuracy.

For feature selection and engineering, first the “data” column containing the json dictionary was processed to result in more individual columns. Columns with all NaN values were dropped. For dealing with the missing values in columns, chi square test was used to arrive at the conclusion that absence of a feature was significantly related to the target variable. The missing values were replaced with the keyword “missing”.

For columns representing the same identifier, (such as ProjectName and ProjectId and ActorDisplayName and ActorUserId etc.), only one column from such pairs was selected for model training. Since there were a lot of columns with categorical variables, columns with high cardinality were subjected to frequency encoding and one hot encoding was applied to columns with low cardinality. The dataset was then divided into training and test sets.

The features used in the final model are:

ActorDisplayName	ProjectName_GL.Legacy-AFAS
Data_PipelineName_encoded	ProjectName_GT.CORPSYS-FIN

Data_ReleaseName_encoded	ProjectName_GT.EDS-DataWarehouse
Data_StageName_encoded	ProjectName_GT.ICS-EAI
Data_EnvironmentName_encoded	ProjectName_GT.ICS-EUE
Data_JobName_encoded	ProjectName_GlobalLifestyle-ConnectedLiving
ProjectName_AS.CL-ReleaseManagement	ProjectName_Missing
ProjectName_AS.Wireless-Aspire	ProjectName_RIM.GL.FS-Portal Dev - DD 10272028
ProjectName_ASP.DTE-DraftTracEnterprise	ProjectName_RIM.GT.Digital-Service-Run - DD 11182028
ProjectName_ASP.PMSClaims-Projects	Area_Pipelines
ProjectName_ASP.SmartFlow-Hazard	Area_Release
ProjectName_EBS	ScopeDisplayName_AIZ-GH (Organization)
ProjectName_GA-APAC	ScopeDisplayName_AIZ-GL (Organization)
ProjectName_GH-Shared Services	ScopeDisplayName_AIZ-GT (Organization)
ProjectName_GH.AppDev-AgilSource	ScopeDisplayName_AIZ-Global (Organization)
ProjectName_GH.AppDev-SSP	ScopeDisplayName_AIZ-Hyla (Organization)
ProjectName_GH.Lending-Digital	ScopeDisplayName_AIZ-SSP (Organization)
ProjectName_GH.Lending-TrackAll	ScopeDisplayName_aiz-alm (Organization)
ProjectName_GL.CL-Elita	Data_CallerProcedure_Release.prc_CancelDeployment
ProjectName_GL.CL-GlobalModules	Data_CallerProcedure_Release.prc_OnDeploymentCompleted
ProjectName_GL.CL-ePrism	Data_CallerProcedure_Release.prc_RejectMultipleReleaseEnvironment
ProjectName_GL.IT-Argentina Applications	Data_CallerProcedure_Task.prc_UpdateEnvironmentDeploymentReques

For the initial RF classifier model, 100 estimators were considered for training. The accuracy obtained for this model was 89%. Further hyperparameter tuning was carried out using GridSearchCV along with Cross Validation. The best model after hyperparameter tuning achieved an accuracy of 89%. Here is the confusion matrix for the model performance on the test dataset.



The classification report of the model is as follows:

	precision	recall	f1-score	support
Canceled	0.50	0.04	0.08	45
Failed	0.76	0.66	0.70	454
Not deployed	0.98	0.93	0.96	91
Succeeded	0.91	0.96	0.94	2106
Succeeded with issues	1.00	0.45	0.62	11
partially succeeded	0.89	0.67	0.76	12
accuracy			0.89	2719
macro avg	0.84	0.62	0.68	2719
weighted avg	0.88	0.89	0.88	2719

The top 10 variables as per the random forest classifier are as follows:

	Feature	Importance
1	Data_PipelineName_encoded	0.185128
42	Data_CallerProcedure_Release.prc_RejectMultipl...	0.162793
3	Data_StageName_encoded	0.096096
41	Data_CallerProcedure_Release.prc_OnDeploymentC...	0.087179
5	Data_JobName_encoded	0.087006
4	Data_EnvironmentName_encoded	0.080153
40	Data_CallerProcedure_Release.prc_CancelDeployment	0.074623
2	Data_ReleaseName_encoded	0.041095
0	ActorDisplayName	0.026196
32	Area_Release	0.025170

For the gradient boosting classifier, an accuracy of 88.6% was obtained. Since the Random Forest Classifier has better performance, it is recommended that this model be used with the API.

Language Models

To categorize the data into success or failures, we experimented with a few LLM models. These are the result:

1. Falcon

Falcon is a Transformer-based language model fine-tuned for generating text, answering questions, and performing a variety of NLP tasks with high accuracy and contextual relevance. By using this model, we were able to categorize the data based on the Details column into failure and success. However, the prediction for the data where it doesn't contain the word "Succeed" and "Failed" is not very good, as the result often changes with each run. We also tried to use the model to determine the SOC violations. The model was able to tell us the violation based on the Details column, as shown in the picture below:

index	Details	SOC_violation
0	Service Connection "OctopusDeployWebHosting" of type OctopusEndpoint executed in project GL.CL-Elita.	This operation violates the Security Operations Center (SOC) standard. It is important to ensure that all operations are following the standard to maintain the security and safety of the company. Please follow the standard to avoid any violations.
1	Service Connection "OctopusDeployWebHosting" of type OctopusEndpoint executed in project GL.CL-Elita.	The operation violates the SOC 1-404 Security Control Standard. The security principle 'Access Control: Restricting Data and System Access Based on Role' is violated as the endpoint executed in the project 'gl.cl-elita' does not follow the established role access control mechanisms defined in the organization's security policy.
2	Service Connection "SE-Assurant Spoke MAXVALUE NonProd" of type azurearm executed in project GL.CL-TradelnUpgrade.	The operation violates Azure Service Configuration standard - the maxvalue nonprod parameter in the service configuration is not supported. The non-supported parameter causes the operation to fail.
3	Service Connection "SE-Assurant Spoke MAXVALUE Model" of type azurearm executed in project GL.CL-TradelnUpgrade.	This command is violating the service connection limit of 2500 maximum values.

However, calculating its accuracy is difficult, as we don't have the reference labels. If we base the violation on the existence of the word "failed" or "succeed" or its variants, the model often incorrectly claims violations for data that are successful and asserts no violations for data that are failures. However, it is impossible to get an exact figure due to the lack of context in the data.

2. Distilbert

DistilBERT is a Transformer-based model fine-tuned for sentiment analysis, capable of classifying text into positive or negative sentiment. We used this model in the hopes of categorizing the data based on the sentiment in the Detail column. However, the results of this model are more inaccurate than those of the Falcon model, as it often categorizes successful rows (Positive Sentiment) as failures (Negative Sentiment).

Based on those results, it is evident that the text field in the data is not sufficient to build a meaningful LLM model that can accurately predict failure and the success of each row, or even to determine the kind of SOC standard that it violates. Thus, we decided to use basic regular expressions to categorize the data, based on the existence of the word "failed" and "succeed" or its variation.

SOC analysis using a RAG Model

Similar issues to the categorization were encountered. Feature engineering with just a single text field hasn't yielded any meaningful result. Producing reliable results with a Retrieval-Augmented Generation (RAG) model for detecting SOC violations proved challenging due to several limitations inherent in the data and methodology.

- Firstly, without ground truth labels, it was difficult to validate the accuracy of the model's predictions. For instance, sentences in the details column such as "System executed unexpected operation in module" and "Unauthorized access attempt detected" give no information on SOC violations and are context-dependent, making it hard for the model to consistently interpret whether they constituted a violation.

- The reliance on a single text field limited the contextual understanding required to identify nuanced violations accurately.
- We tried providing context to the LLM by feeding it definitions of SOC1, SOC2 violations to get some result however it doesn't meaningfully discern or identify given the lack of context in the data.

These factors collectively hindered the effectiveness of using a RAG model for this purpose, reiterating the need for more context from an auditor or a subject expert to provide a meaningful result.

Observed Issues with the Data

1. Correlating Failures with other Log Entries

- a. Our inability to connect a log entry to any other entry, indicating a failure, severely hampers our ability to build a journey or identify the series of steps taken before a failure. This significantly impairs our failure prediction capabilities.
- b. Without the ability to do this, we could not spot any pattern of preceding steps before a failure, and this would prevent us from predicting failures before they occur.
- c. According to Microsoft Azure Documentation, the field CorrelationId links entries together, but the CorrelationId for all log entries is unique and cannot be linked to any other entry. This means we could only look at the failure entry in isolation, which did not help with insights.

2. Identifying SOC Non-Compliance

- a. While we had a high-level understanding of the tenets of SOC compliance, we could not identify the exact policy breaches that an action might cause that might lead to a rejection or a failure.
- b. Analysis of trends and spikes has not yielded any meaningful information.
- c. The direct error codes and more data/information on the underlying system might provide insights and detect violations.

3. Lack of Text Fields for a meaningful NLP/LLM solution

- a. Most data fields in the dataset are system-generated identifiers with no inherent or semantic meaning; they cannot be used for NLP or LLM-based models to garner insights.
- b. For association rule learning to be fully effective, we need additional fields to identify and understand repetitive patterns.
- c. The only text field in the dataset, "Details," describes only the step/outcome itself and is unusable for deriving insights.

4. Truncated Data - Not reliable for Trends

- a. If we look at the data shared by the ScopeName, we see that most ScopeNames have a maximum of 30K rows. This means the data provided has a limit imposed on it and was incomplete.
- b. Additionally, the data was only until 31st May, and 52% of the data is only from the last five days of May(26th May - 31st May)
- c. These facts point to the data being incomplete/truncated, and we cannot be sure to draw reliable/meaningful conclusions about the trends of successes and failures.

Results, Evaluation and Model Selection

The only models that yielded any meaningful results were the Supervised Models, all attempts at using Unsupervised models like K-Means Clustering, DBSCAN and Apriori, and advanced Language Models Distilbert, Falcon and Retrieval-Augmented Generation have not yielded meaningful results.

So we must proceed with the Supervised Models, specifically the Random Forest Classifier, since it has more promising results.

Implementation

Integration of Models into a System

In our solution, we integrate machine learning models into the system with key steps including data processing and model prediction. In order to maintain the consistency of prediction results and model after integration, the same data processing process as model is adopted, and the combination of original data and new data is included to ensure the consistency of data.

1. **Data Processing:** Use pandas to load data from CSV files and concatenate data received by API services. To ensure the accuracy of the forecast, unnecessary columns are removed to reduce noise and potential impact on prediction accuracy. Specifically, we filter the 'Area' column values to retain only 'Pipelines' or 'Release', and set 'Category' to 'Execute'. The classification features are encoded by one hot encoder and transform, and the missing features of default values are added to ensure that the processed data is consistent with the expected features of the model.
2. **Model Predict:** The pre-trained model, saved using joblib, is loaded for making predictions. By using joblib's saved model to predict the processed data, we only need to predict the newly added data. The predicted result and the 'id' of the data itself correspond to the composition of the dictionary as the overall return result output.

This approach ensures that the integration of the model into the system is seamless, maintaining consistency and accuracy in predictions by strictly following the data processing pipeline used during model training.


API Development and Usage

The development and usage of an API are crucial for enabling external systems to interact with our machine learning model. We have implemented both server-side and client-side deployment code for the API service. Here is how we developed and utilized the API for prediction:

1. API Development:

- a. **Framework:** We used FastAPI for its high performance and ease of use. It also can generate a readable Swagger UI.
- b. **Endpoints:** We developed two main endpoints:
 - i. GET/: A simple health check to ensure the server is running.

Server response

Code	Details
200	<div>Response body</div> <div> <pre>"Server is running...."</pre> <div>  <div>Download</div> </div> </div>

- ii. POST/uploadfile/: Accepts a CSV file, processes the data, runs it through the model, and returns the prediction results.

2. API Usage:

- a. **Client Request:** The client can send a POST request to the /uploadfile/ endpoint, providing a CSV file. All data formats must be consistent with the training data file.
- b. **Data Processing:** The server automatically reads and processes the CSV file to match the expected format.
- c. **Model Prediction:** After the data process finishes it will be passed to the model for predictions.
- d. **Response:** The server responds with a result that provides the log data IDs and their corresponding predictions.

Server response

Code	Details
200	<div>Response body</div> <div> <pre>{ "filename": "log_data_request.csv", "predictions": "{\\"3c4bdc31-56bb-45db-8d9a-22ce554f0988\\": \\"Succeeded\\", \\"893dc326-1297-4489-bc50-8201610ee1a2\\": \\"Failed\\", \\"9c4a6628-81b9-49e3-8262-907b9abef684\\": \\"Succeeded\\"}"</pre> <div>  <div>Download</div> </div> </div>

By integrating the machine learning model into the system and developing the API, we enable the system to make predictions on new data. The system processes incoming data, uses the trained model to make predictions, and provides the results through a well-defined API, ensuring ease of use and scalability.

Conclusion and Recommendations

In conclusion, while we have developed a model that can be used to predict Failures and will contribute to utility with the data that we have access to currently, there are many avenues that remain open for exploration or incremental improvement.

The following avenues can be used to enhance model performance or to identify potential failures using other more reliable signals. Some potential avenues are:

1. The ability to connect log entries to build a sequential journey of user actions, this may enhance our understanding of potential failures and improve our ability to predict failures.
2. Adding detailed Text Fields would allow for the use of Advanced Language Models and by extension make explaining the issue to an end user easier.

Appendix

- [Github Link](#)