1. **DETECTING ANOMALIES: Log Anomaly Classification**

Anomaly Detection System for Communication Channels: Leveraging AI to Identify and Classify Deviations in Email, Voice, and Message Interactions.

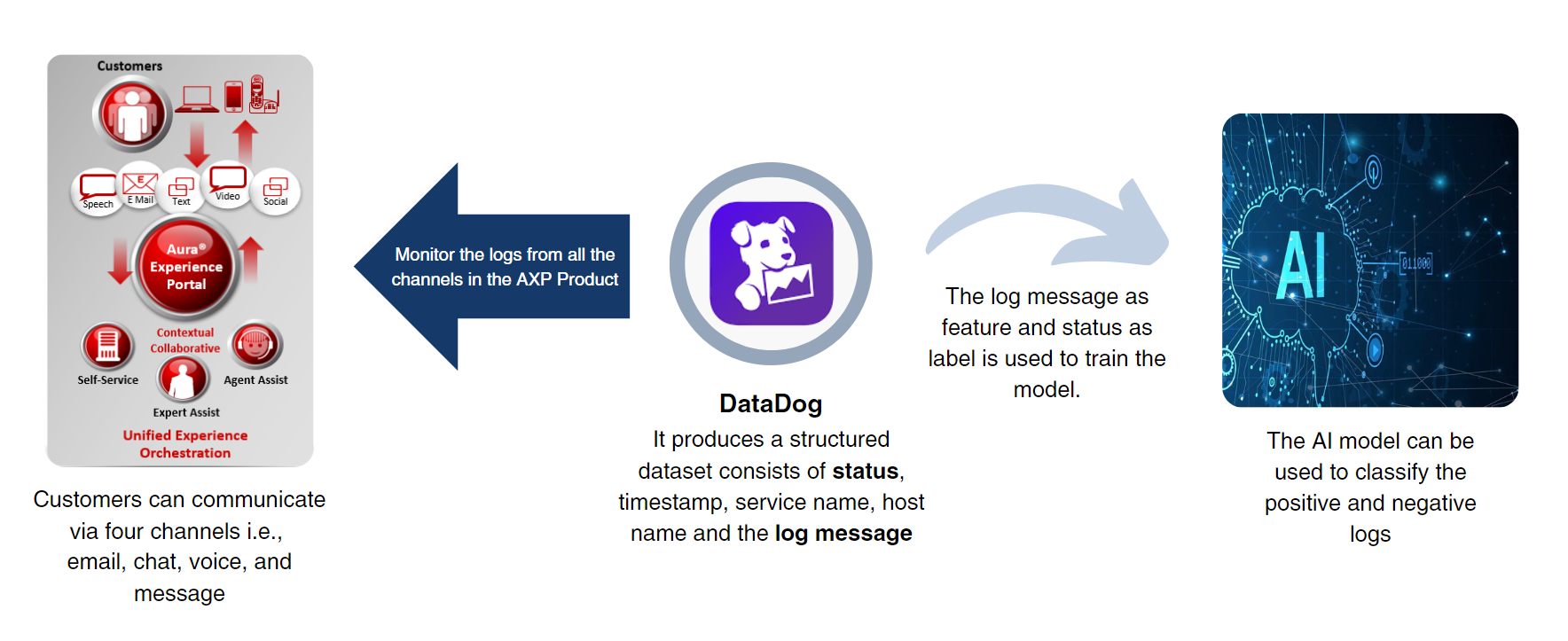
**Problem Statement:** To detect the anomaly in an application using the information provided by logs using an AI model.

**Flow of the Project:**

Anomaly detection is the process of identifying data points, entities, or events that fall outside the normal flow.

The Avaya Workspaces provides four channels: email, chat, voice, and messaging. The system focuses on one channel at a time. This implies that the success and failure scenarios for each channel are analyzed independently. DataDog is used to gather logs related to the success and failure scenarios for each communication channel.

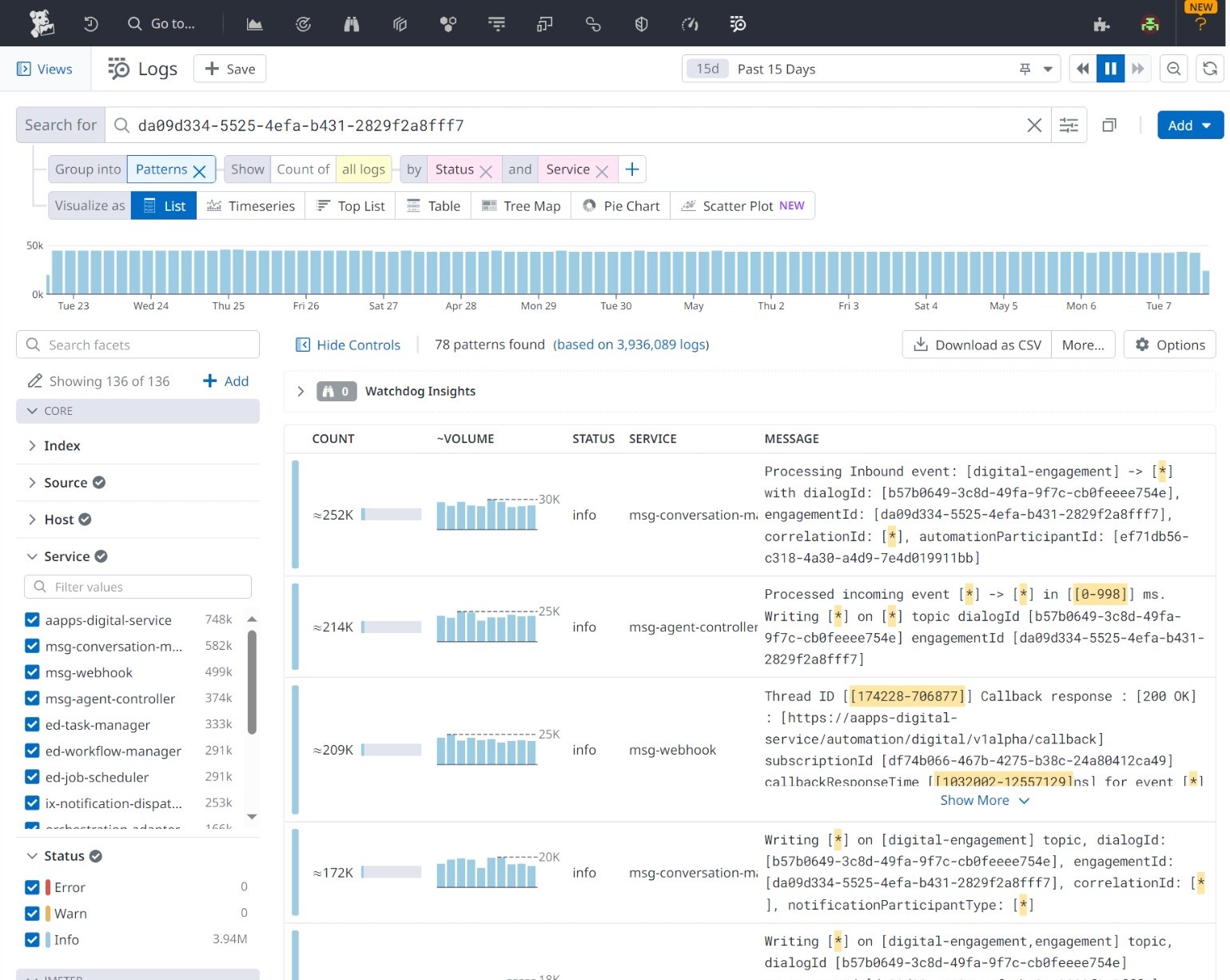
The gathered logs are then fed into an AI model. This model is designed to identify patterns that differentiate successful scenarios from unsuccessful ones. The AI model uses natural language processing techniques to learn from the data and recognize patterns. Once the AI model identifies patterns, the insights gained can be used to improve application failure analysis.



Flow of the Project

**Dataset Preparation:**

A successful flow or scenario of an Email is executed and its interaction details are recorded using a unique Engagement ID. When we apply a filter of Engagement ID on DataDog, it provides a sequence of logs that are involved in the routing of an Email.

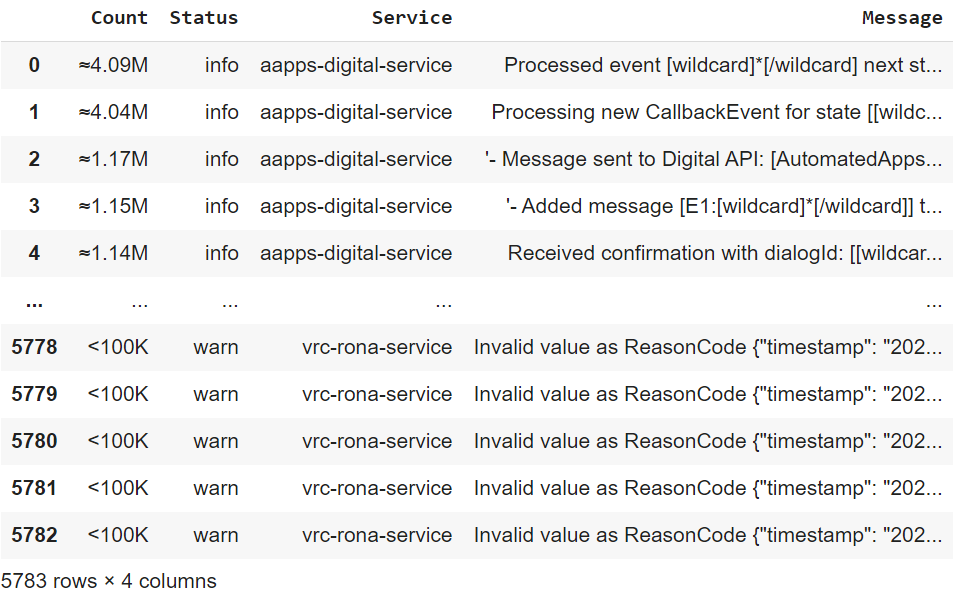


The dataset is downloaded using “Download as CSV” button after applying the filters i.e. Engagement ID, Service and Status.

The entire Email flow consists of **33 services**.

**Services**: ['aapps-digital-service', 'analytics-cdr-consolidator','analytics-cdr-data-adaptor', 'analytics-dsi','analytics-duration-proc', 'analytics-event-filter','analytics-event-normalizer', 'analytics-passthrough-proc','async-gateway', 'customer-journey-etl', 'ed-job-scheduler','ed-task-manager', 'ed-workflow-historical-collector','ed-workflow-manager', 'ix-dsk-connector','ix-dsk-media-connector', 'ix-dsk-request-service','ix-notification-dispatchers', 'journey-service','msg-agent-controller', 'msg-conversation-manager','msg-web-gateway', 'msg-webhook', 'orchestration-adapter','pre-routing', 'smtp-connector', 'uc3-connector','vrc-availability-service', 'vrc-matching-engagement','vrc-matching-service-rest', 'vrc-matching-service','vrc-metrics-service', 'vrc-rona-service']

Using the above information about the services involved in the Email flow. A dataset is prepared by filtering the Engagement ID and Services. Each Service log is then labeled as “positive” (if the status is “info” or “warn”) and “negative” (if the status is “error”).



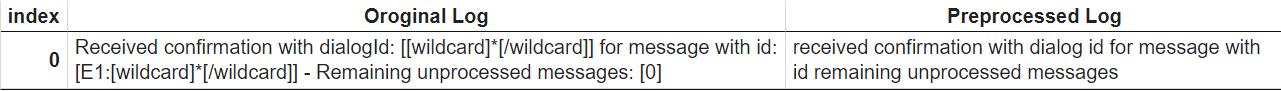
**Dataset Preprocessing:**

The dataset prepared must be processed before feeding to Machine Learning or Deep Learning Algorithms.

The data preprocessing includes:

1. Replacing special characters with spaces.
2. Lowercasing uppercase words.
3. Removing words containing digits.
4. Removing punctuation.
5. Removing specific words like 'wildcard', 'yyyy', 'mmdd', 'hhmmss', 'sss' etc.

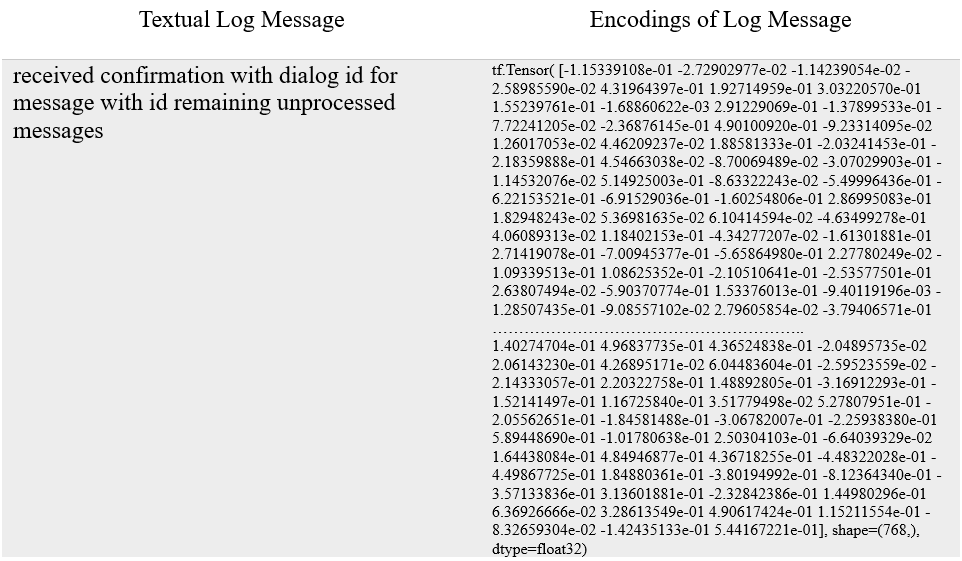
For Example:



The difference between an Original Log and Preprocessed Log

**Feature Engineering:**

Textual data, being **unstructured**, needs to be transformed into a numerical representation before it can be used as input for machine learning algorithms. This process of converting text into numerical vectors is known as text vectorization.



Conversion of Textual Log Message into Vector Encodings

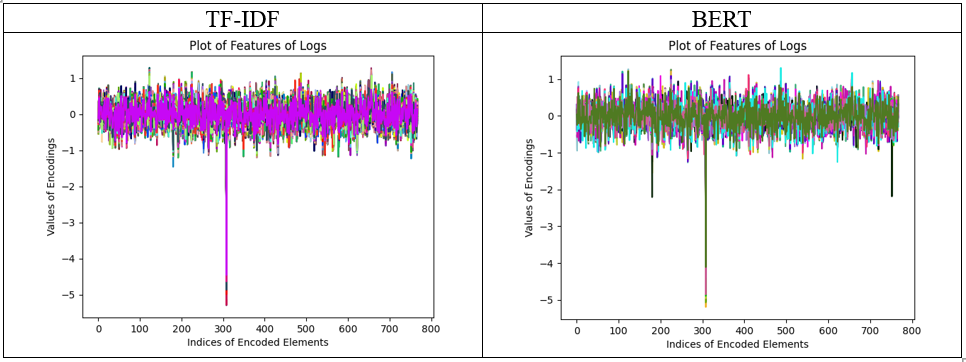
In this project two techniques are used to convert text into numerical encodings:

1. TF-IDF (Term Frequency-Inverse Document Frequency): Reflects the importance of a word in a document relative to a collection of documents.
2. BERT (Bidirectional Encoder Representations from Transformers): Utilizes transformers to capture contextual information of words in a sentence.

**Dataset Visualization:**

1. Simple Visualization:

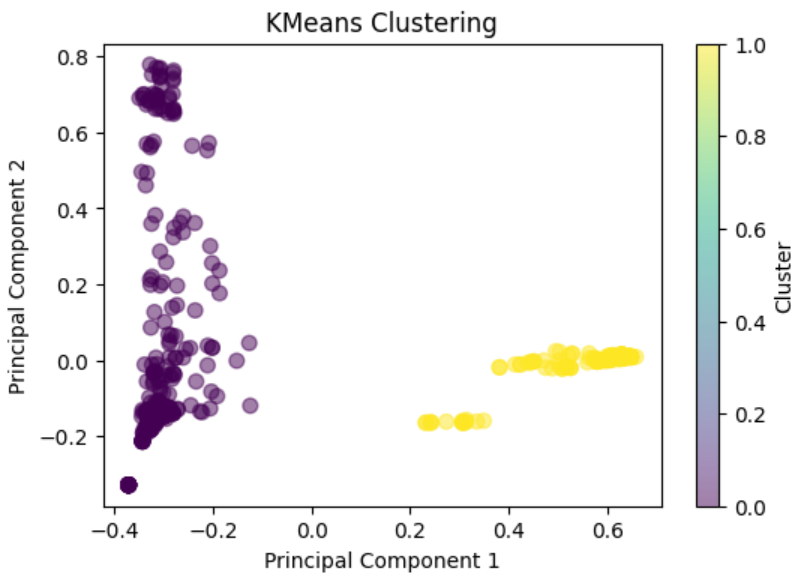
The encoded vectors of the logs are plotted with the X-axis as indices of the vector and the Y-axis as the encoding values. The aim was to identify some patterns that differentiate one log encoding from the other. With this simple approach, the plots of the logs appear to be overlapping.



Visualize the encodings of the logs

1. KMeans Clustering Algorithm:

It is an unsupervised algorithm. In KMeans, K stands for the number of clusters. K=2 is used because two clusters are required, one represents the positive logs whereas the other represents the negative logs. From the figure, it is clear that two discrete clusters are formed. And the set of logs in each cluster belong to the same label i.e. purple belongs to positive and yellow belongs to negative. So, it can be concluded that the log data is sufficient to provide the status and can be used to train an ML(Machine Learning) or DL(Deep Learning) model.



Visualize the dataset using KMeans divided into two discrete clusters.

**Training an AI Model:**

1. **Machine Learning Approach:**

In this approach, the vector encodings of the textual data are fed into an ML(Machine Learning) model. ML is best for well-defined tasks with structured and labeled data. The dataset consists of logs of 33 services that are involved in the entire flow of Email Communication.

No. of samples: 546

Positive Samples: 273 (with status as info or warn)

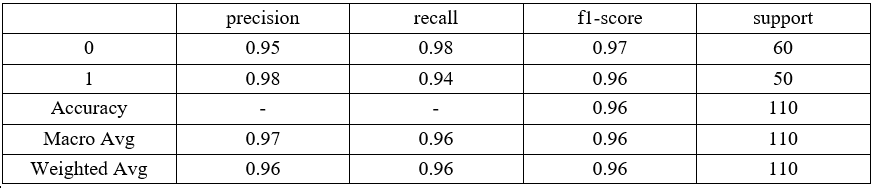
Negative Samples: 273 (with status as error)

For these types of problems where anomaly classification is the prime goal, classifiers such as Support Vector Machine, Random Forest Classifier & Decision Trees are mostly used.

**Classification Reports of the Models:**

Classifier 1: Support Vector Machine:

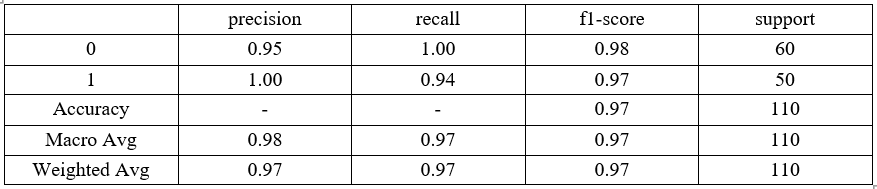
Accuracy: **0.9636363636363636**



Classification Report of SVC

Classifier 2: RandomForestClassifier:

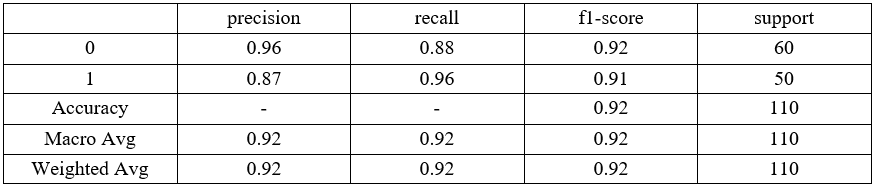
Accuracy: **0.9727272727272728**



Classification Report of RandomForestClassifier

Classifier 3: Decision Tree:

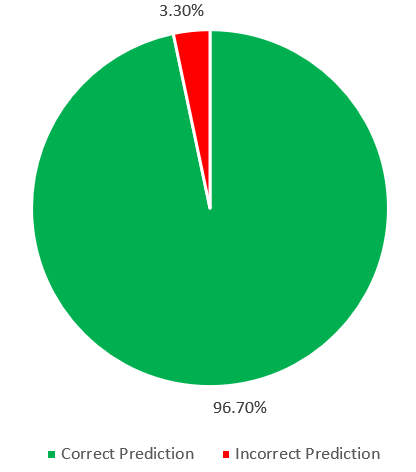
Accuracy: **0.9181818181818182**



Classification Report of Decision Tree

Accuracy is **96.36%** with **SVM** and **97.27%** with **RandomForestClassifier.**

Inference of Model on the **real-time data** downloaded from the Datadog.



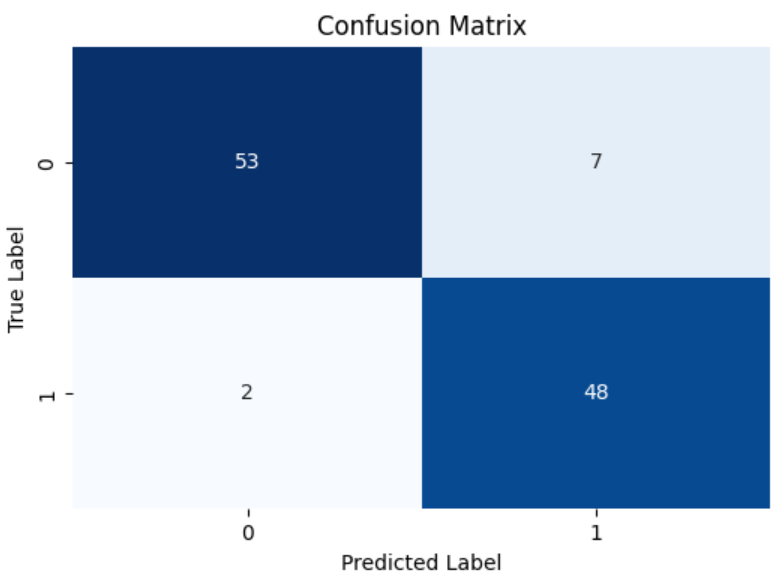
Inference of ML model on Unseen data

**Performance analysis using Confusion Matrix:**

**A confusion matrix provides a summary of the performance of a classification model. It's particularly useful in evaluating the accuracy of a model's predictions, especially for binary classification problems (such as ‘positive’ = ‘1’ and ‘negative’ = ‘0’) where there are only two possible outcomes.**

**It displays the following four important metrics:**

1. **True Positives (TP): These are the cases where the model correctly predicts the positive class.**
2. **True Negatives (TN): These are the cases where the model correctly predicts the negative class.**
3. **False Positives (FP): These are the cases where the model incorrectly predicts the positive class when it's actually negative. Also known as Type I error.**
4. **False Negatives (FN): These are the cases where the model incorrectly predicts the negative class when it's actually positive. Also known as Type II error.**



Confusion Matrix of the predictions with RandomForestClassifier

1. **Deep Learning Approach:**

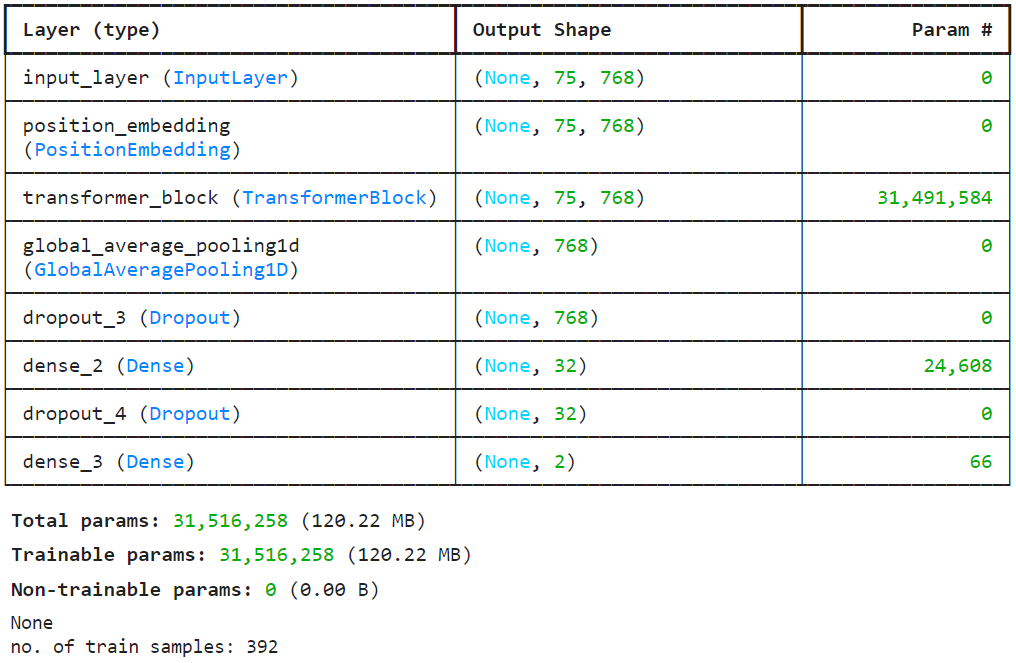
In this approach, the vector encodings of the textual data are fed into a DL(Deep Learning) model. Deep learning is best for complex tasks that require machines to make sense of unstructured data such as textual data.

The dataset consists of logs of 33 services that are involved in the entire flow of Email Communication.

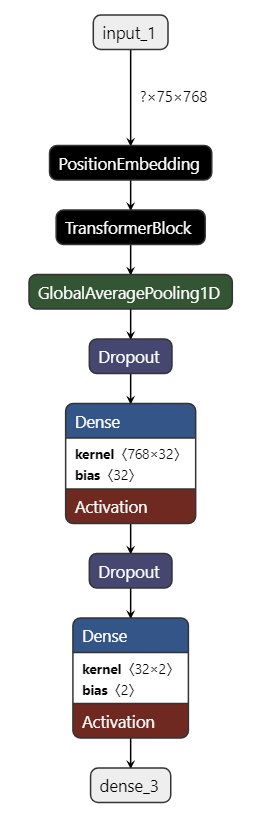
No. of samples: 546

Positive Samples: 273 (with status as info or warn)

Negative Samples: 273 (with status as error)



Layers of Model & Architecture



Architecture of the Model

1. Input Layer: the encoded data is provided to the model at this layer.
2. Transformer Layer: It is a custom layer that has transformer architecture implemented in it that consists of some layers:
   1. [Multi-Head Attention](https://paperswithcode.com/method/multi-head-attention) layer: It is a module for attention mechanisms that run through an attention mechanism several times in parallel.
   2. Dense Layers: In a dense layer, each neuron is connected to every neuron in the previous layer. Dense layers perform a linear transformation followed by an activation function, allowing the network to learn complex nonlinear relationships between the input and output data.
   3. Normalization Layers: Normalization layers are used to standardize the inputs of a layer by scaling them to have zero mean and unit variance.
   4. Dropout Layers: Dropout is a regularization technique used to prevent overfitting in neural networks. Dropout layers randomly "drop out" a fraction of the units (neurons), meaning they are temporarily removed from the network along with all their incoming and outgoing connections.
3. Embedding Layer: [Position Embedding](https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/) layer helps in understanding the position or order of the tokens in the input sequence.
4. Output Layer: This layer gives the final result i.e. class in which the inputted data belongs.

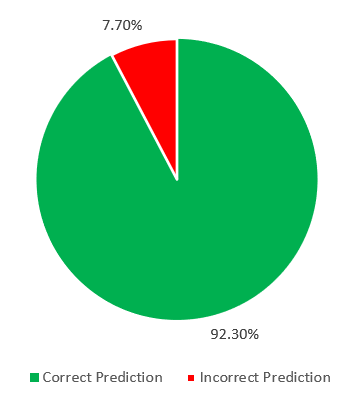
**Model Evolution:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model description** | **Accuracy Curve** | **Loss Curve** | **Conclusion** |
| **Model1:**  Learning rate: 3e-4  Epochs: 50 |  |  | The validation loss increases while the training loss decreases. It represents that the model is **overfitting** on the training dataset. |
| **Model2:**  Learning rate: 3e-5  Epochs: 50 |  |  | By decreasing the learning rate from **3e-4 to 3e-5**, the **overfitting is eliminated** (since both the training and validation losses are decreasing at the same rate). But the accuracy of the model reduced to 70% from 89%. |
| **Model3:**  Learning rate: 3e-5  Epochs: 100 |  |  | After increasing epochs to 100 from 50, the accuracy is also increasing gradually. We must check for higher epochs (i.e., >100). Since the **learning rate is decreased** the minima might be at the **higher epoch**. |
| **Model4:**  Learning rate: 3e-5  Epochs: 300 |  |  | The model overfits for the larger epochs (after 150 epochs). Training to 150 epochs will give optimal training without overfitting and with almost 90% accuracy and ~0.25 loss. |
| **Model5:**  Learning rate: 3e-4  Epochs: 100  Dropout layer:0.3. |  |  | After increasing the dropout rate (i.e., from 0.1 to 0.3) in transformer\_classifier the model still **overfits** at 40 epochs onwards after achieving almost 85% accuracy. |
| **Model6:**  Learning rate: 3e-5  Epochs: 100 (early stopping at 41 epochs) |  |  | The model achieves almost 75% accuracy and stopped at 40th epoch considering the relative increase in the values of the training and validation losses. |
| **Model7:**  Learning rate: 3e-4  Epochs: 30 |  |  | The learning rate is increased from **3e-5** to **3e-4**. But due to this the after 7th epoch there is a major hike in accuracy and a drop in loss which specifies that the model **overshoots** the global minimum because of the fast learning rate. |
| **Model8:**  Learning rate: 3e-5  Epochs: 30 (early stopping at 19 epochs) |  |  | This is the **best model** as it produces good results with a very small number of epochs i.e., **19,** and a considerable accuracy of about **89%** on the training dataset. |

**Model8** is the most optimized and the **best model**.

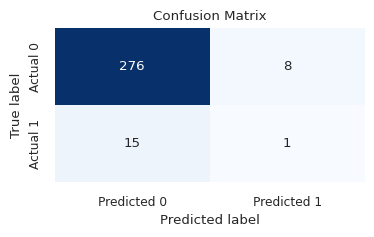
Accuracy is **89%** with **Transformer Deep Learning Model (i.e. Model8)**

Inference of model on the **real-time data** downloaded from the Datadog.



Inference of DL Model on Unseen Data

**Performance analysis using Confusion Matrix:**



Confusion Matrix of the prdictions with Transformer-based model

1. **DETECTING ANOMALIES: Sequence Deviation Analysis in Log Data**

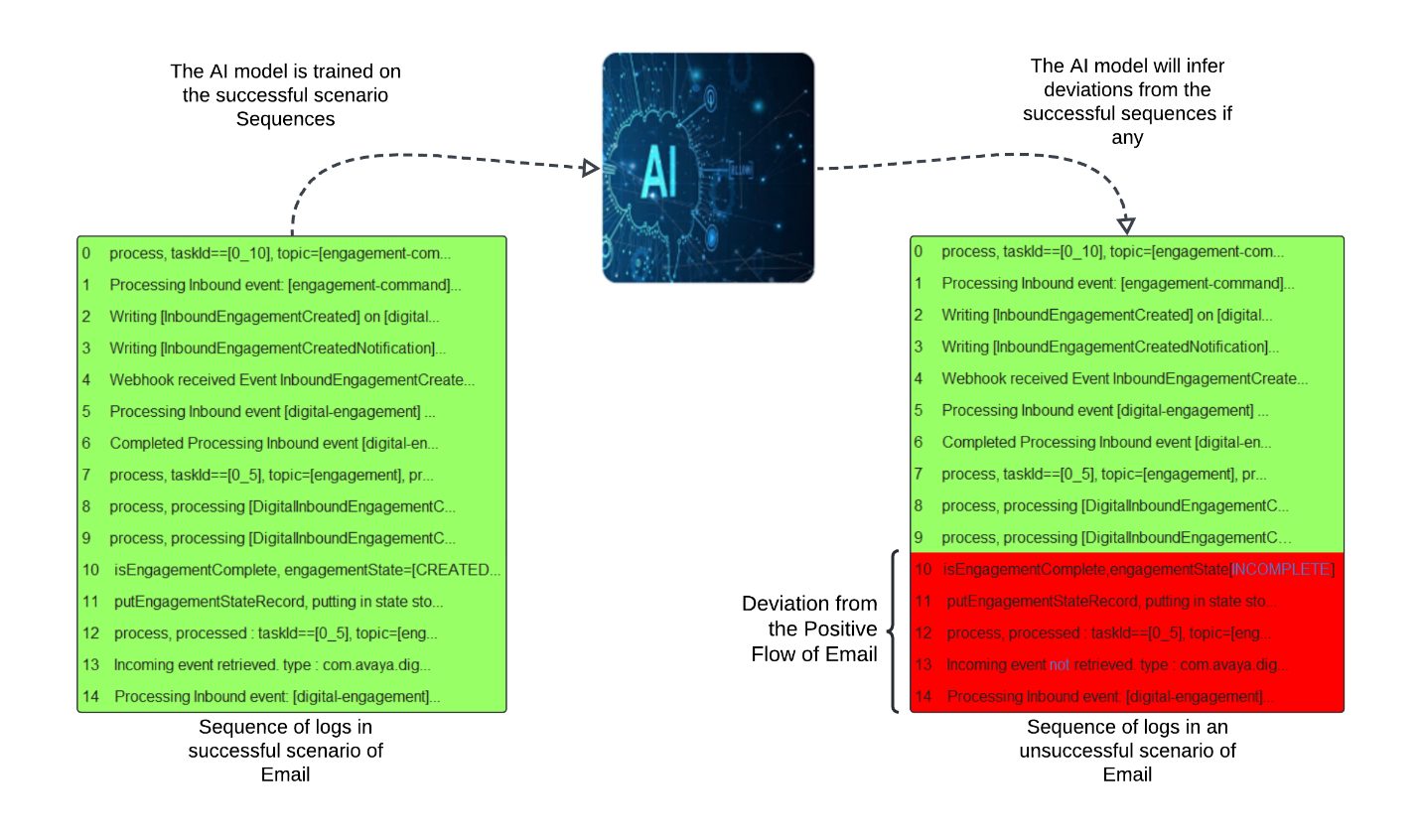
**Problem Statement:** To detect the anomaly in the sequence of services and their logs involved in the communication channels.

**Flow of the Project:**

The objective is to identify deviations or abnormalities from the expected behavior within the sequence of services and their associated logs.

The Avaya Workspaces provides four channels to interact with the customers: email, chat, voice, and messaging. The success and failure scenarios for each channel are analyzed independently. DataDog is used to gather logs related to the success and failure scenarios for each communication channel.

The gathered logs are then fed into an AI model. This model is trained on the positive scenarios of the channel's entire journey. For any real-time interaction if logs are provided to the model, then the model predicts the next logs of the interaction. If these **predicted logs match the following logs** by the system then the flow of the interaction is said to be **positive** but if the **predicted logs and following logs don’t match** then the flow of the interaction is **deviated from the positive flow**. The AI model uses natural language processing techniques to learn from the data and recognize patterns. Once the AI model identifies patterns, the insights gained can be used to improve application failure analysis.

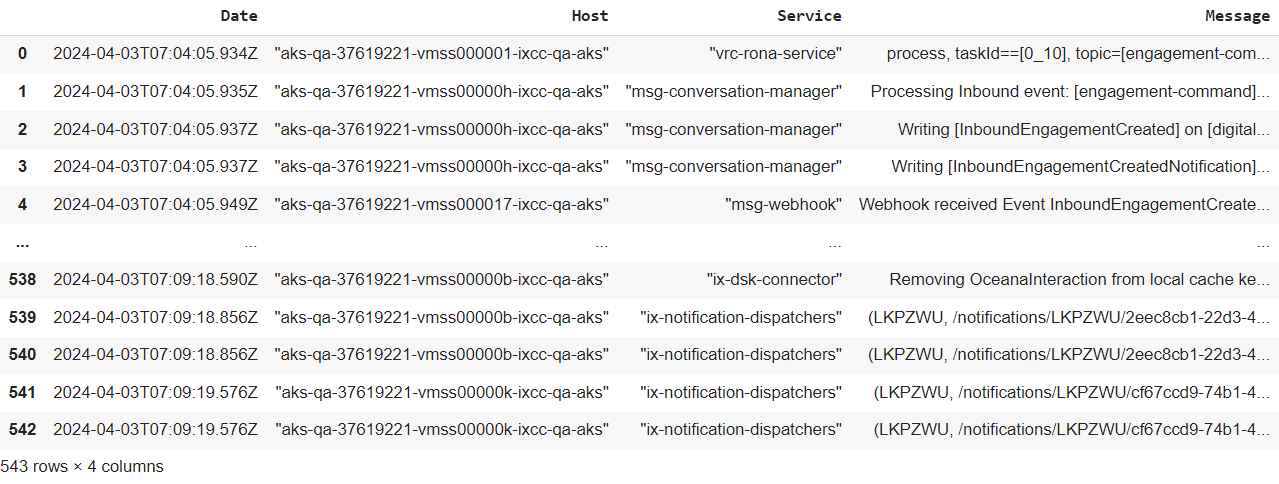


Flow of the Project

**Dataset Preparation:**

A successful scenario of an Email is executed and its interaction details are recorded using a unique Engagement ID. When we apply a filter of Engagement ID on DataDog, it provides a sequence of logs that are involved in the routing of an Email. For a particular Engagement ID, we then sort the logs w.r.t time in ascending order to download the data in sequential order.

This sequential data consists of positive scenarios and is used to train an AI model on the occurrence of logs using the concept of **Sequence Learning**.



Dataset download from the Datadog filtered with engagement ID and sorting w.r.t time in ascending order

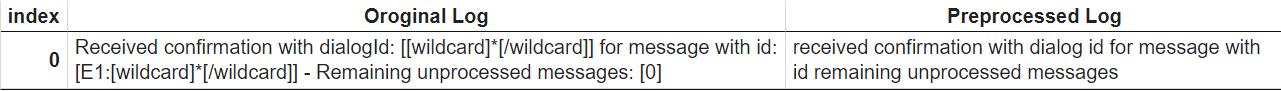
**Dataset Preprocessing:**

The dataset prepared must be processed before feeding to Machine Learning or Deep Learning Algorithms.

The data preprocessing includes:

1. Replacing special characters with spaces.
2. Lowercasing uppercase words.
3. Removing words containing digits.
4. Removing punctuation.
5. Removing specific words like 'wildcard', 'yyyy', 'mmdd', 'hhmmss', 'sss' etc.

For Example:

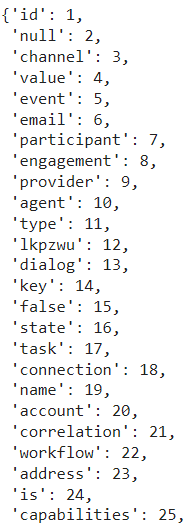
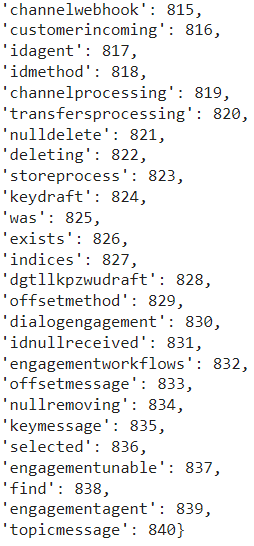


The difference between an Original Log and Preprocessed Log

**Feature Engineering:**

The entire dataset is fed into a Tokenizer which gives the term frequency and Inverse document frequency to the words such that the most frequent word is mapped to a small numerical value.

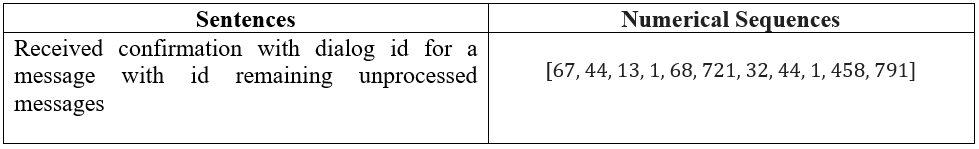
A mapping is created to display the words in a numerical form.

 ………………

Dictionary with key as words and values as numerical

Each word is associated with some numerical value and after applying it to any sentence the words are converted into numerical values mapped to words in the corpus.

For Example:



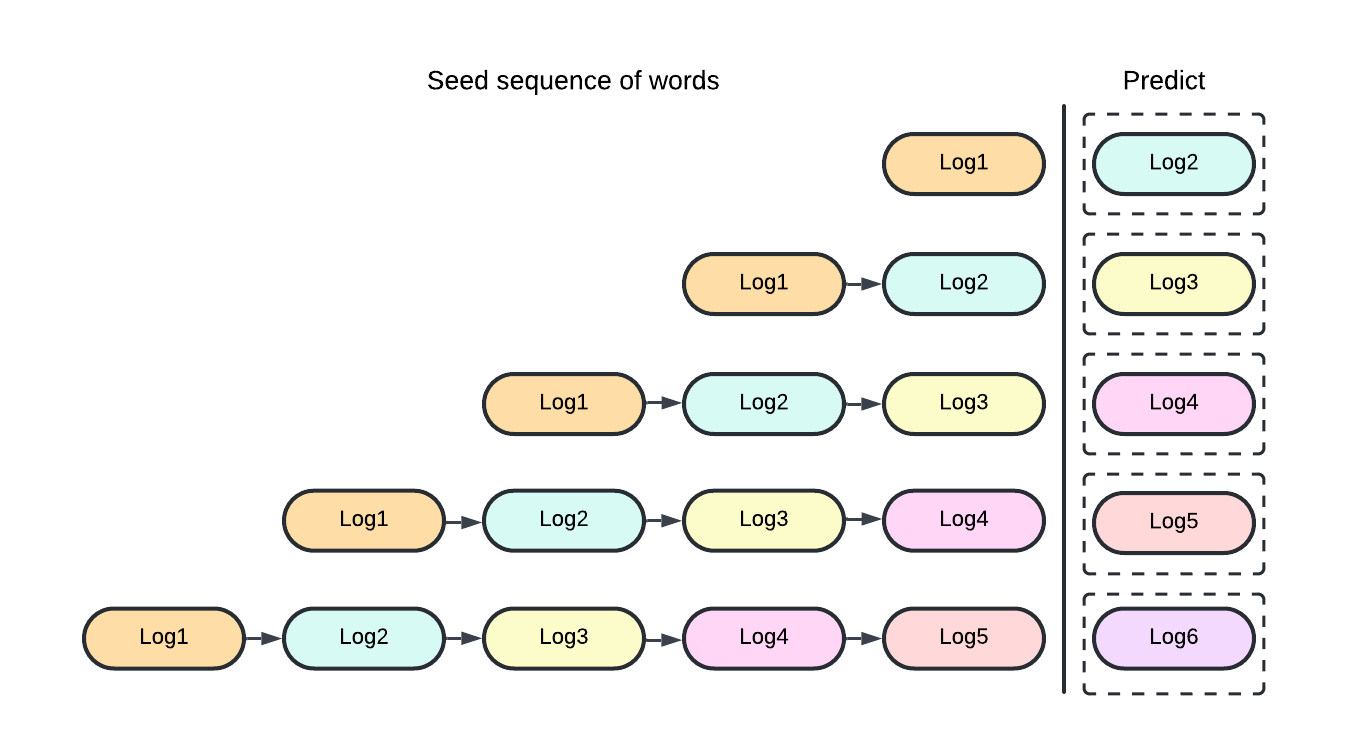
Numerical Encoding for sequence analysis

**Using the approach of Next Word Prediction:**

The model is trained to predict the next word in the sequence given the previous words. This is typically done using a loss function like cross-entropy loss, which measures the difference between the predicted probabilities of each word and the actual next word.

A similar concept is applied here,

The model will be trained to predict the next log in the sequence given the previous log messages. After the entire dataset is trained then it can be assumed that the model can understand the sequences of logs and can predict the deviation from the natural flow of logs.



Next Log Prediction: The model will be trained on the positive scenario of Email sequences

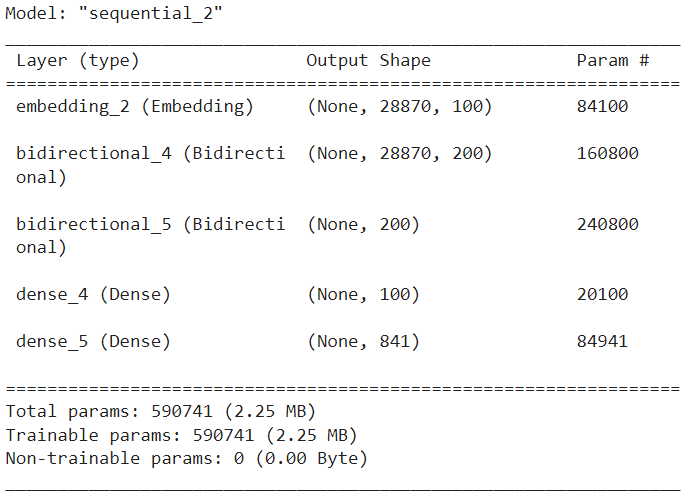
**Training an AI Model:**

1. **Deep Learning Approach:**

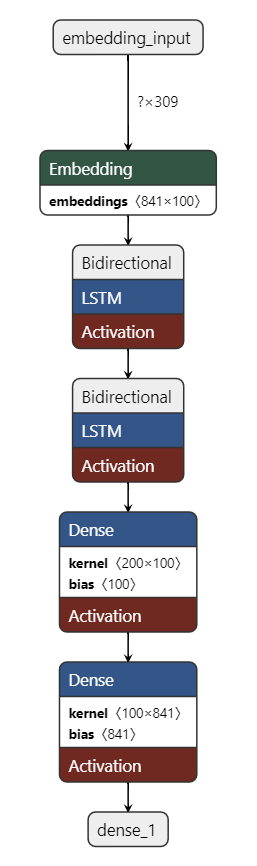
The vector sequential numerical encodings of the textual data are fed into a DL(Deep Learning) model. Deep learning is best for complex tasks that require machines to make sense of unstructured data such as textual data.

The dataset consists of a sequence of logs involved in the Email flow i.e. A sequence of approximately (542-548) logs. A total of 28,870 words or tokens are there which makes the dataset a matrix of (28,870 x 28,870) dimensions.

The matrix cannot be stored in a GPU of 15GB memory. So, the model is trained on the concept of **transfer learning** i.e. model is trained on smaller pieces of the larger dataset. However, this approach works fine with a small set of sentences.



Layers of Model & Architecture



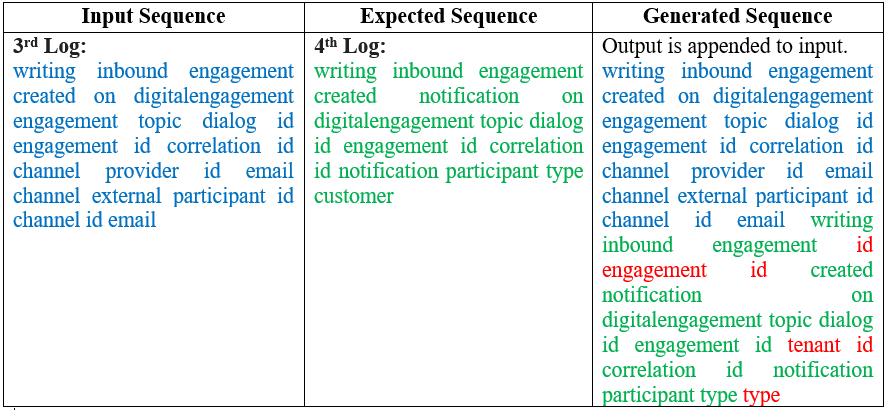
Architecture of the Model

1. Embedding Layer: This layer converts input sequences (sequences of integers) into fixed-size dense vectors of configurable size.
2. Bidirectional layer: Bidirectional layers process the input sequence both forwards and backward, combining the information from both directions. This can be useful for capturing contextual information from both past and future states in a sequence.
3. Dense layer: Dense layers are fully connected layers, meaning each neuron in this layer receives input from every neuron in the previous layer.

**Performance of Model on Training Dataset:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model description** | **Accuracy Curve** | **Loss Curve** | **Conclusion** |
| **Model:**  Learning rate: 0.01  Epochs: 50 |  |  | This is the **best model** as it produces good results with a smaller number of logs in the sequences. However, the results are fine in that it predict the next words in the log with an accuracy of **80%**. |

Inference on the **real-time** data :



If the generated log matches greater than 80% of the expected then it is considered **positive flow** else the Normal (Positive) flow of the interaction is **deviated** after this log message.

**Conclusion:**

Since the log generated will be almost 80% accurate of the actual log. So, to deal with the positive flow we can have a threshold of 80% match. If the predicted log and the actual log have **80% of the content common sequentially** then it will be considered as matching to **positive flow** else **deviation**.