**Towards Efficient Automation of Expert Allocation - Approach for SME Triaging**

**Abstract**

Subject Matter Experts (SMEs) are individuals possessing deep knowledge and expertise in a specific field or area. In many organisations, effective placement of these experts into appropriate departments is critical for optimising productivity and driving innovation. This research paper delves into the challenge of triaging SMEs to suitable departments using advanced machine learning techniques. The term 'triage' refers to the process of assessing and prioritising cases based on their severity and urgency to ensure that resources are allocated efficiently and where they are needed most. 'SMEs triaging' typically refers to a process in which experts in a particular field or subject evaluate, prioritise, and assign resources or responses to specific issues, challenges, or inquiries based on their knowledge and expertise.The primary objective of this work is to develop a model that offers better accuracy and efficiency in SME allocation tasks. Five different models were employed - Q-Learning, DQN Classifier, Artificial Neural Network, Support Vector Classifier (SVC) and K-Nearest Neighbour. Of these, SVC exhibited the most promising results. Leveraging this insight, a novel hybrid model was proposed, combining the foundational principles of wrong prediction single batch feedback learningwith Support Vector Classifier. This model showcased improved prediction capabilities, presenting a potentially effective approach for organisations to streamline their SME allocation processes.

**Keywords** - Subject Matter Experts (SMEs), triage, machine learning techniques, Q-Learning, DQN Classifier, Artificial Neural Network, Support Vector Classifier (SVC), K-Nearest Neighbour

**1.) Introduction**

In the current era, characterised by rapid technological advancements and an ever-expanding pool of knowledge, the effective harnessing of expertise is pivotal for organisational success and innovation. Subject Matter Experts (SMEs) are the individuals who possess profound knowledge and insights, mastered within specific domains. Typically they serve as invaluable reservoirs of specialised information. The strategic placement of these experts within an organisation, aligning their capabilities with suitable departments or tasks, is a critical determinant of productivity and innovation.

When a person of contact outside an organisation raises a concern (question or problem ticket) to the Customer Success Manager (CSMs), the CSMs need to correctly search for the correct department or the correct SME. We try to automate this process where users don’t know who is the right person of contact having domain specific knowledge.

The tribal knowledge of an organisation or a company refers to unwritten information that is not commonly known by others within a company. Tribal knowledge refers to the collective knowledge and skills that are unique to an organisation and its employees. It is undocumented and often unshared, allowing it to be easily lost when employees leave. It is valuable information that has accumulated through informal channels that remains undocumented and isolated from the rest of the organisation. Tribal knowledge can be converted into company property, but it may also be completely incorrect. It is a problem that many companies are either unaware of or unconcerned about.

This research paper embarks on a significant exploration: the challenge of triaging SMEs of appropriate departments facilitated by the application of advanced machine learning techniques. The primary objective of this endeavour is to craft a model that not only enhances the accuracy but also augments the efficiency of SME allocation tasks. In this pursuit, five distinct models have been employed and assessed - Q-Learning, Deep Q-Network (DQN) Classifier, Artificial Neural Network (ANN), Support Vector Classifier (SVC) and K-Nearest Neighbours.

Our fundamental concept centres on the principle of avoiding redundant problem-solving efforts. Instead, our approach hinges on harnessing the information supplied by Subject Matter Experts (SMEs) and using it to create a generalised codebase. This accumulation of knowledge represents a vital resource, one originating from a non-human entity. In this research paper, our primary aim is to explore strategies for addressing this issue. In our specific case, we have generated a sample dataset using GPT4 and conducted our experiments and analyses based on this dataset.

Building upon this insight, we introduce a novel hybrid model, a synthesis of foundational self learning principles of wrong prediction single batch learning on the Support Vector Classifier based upon feedback. This hybrid model exhibits superior predictive capabilities, offering a potential paradigm shift for organisations aiming to streamline their SME allocation procedures. By integrating machine learning prowess with the intrinsic human expertise represented by SMEs, this research bridges the gap between cutting-edge technology and domain-specific knowledge, fostering a more agile and innovative organisational landscape.

The main aim of this paper is to create an architecture which can learn itself in case of wrong prediction. At first a base model was trained (among the 5 models, SVC performed the best). The input to the model are problem tickets which are further used for SME triaging purposes. In case the model fails to provide a correct output, the user has to provide feedback based on which the model re-trains itself on a single batch.

In the following pages, we will delve into a detailed examination of each model, elucidating their methodologies, and highlighting their transformative potential in reshaping the importance of the deployment of Subject Matter Experts (SMEs) triaging automation solution within organisational frameworks. Additionally, we explore the significance of real-time feedback mechanisms, temporal considerations, and adaptive learning in refining and optimising the allocation process. In case of wrong output, the model would learn through the feedback. Through this research, we endeavour to empower organisations with the tools and insights necessary to harness the full potential of their subject matter expertise, thereby propelling them towards new frontiers of productivity and innovation.

This paper takes a utility-driven perspective toward Subject Matter Experts (SMEs), aiming to extend its applicability beyond mere triaging tasks. It thoroughly investigates the information supplied by SMEs and its seamless integration into our model.

From the perspective of a customer service landscape, many individuals face the frustration of navigating complex automated phone systems when they contact companies for assistance. This issue spans various sectors, including e-commerce, banking, fintech, technology, healthcare, and more. Customers are often required to select from a multitude of options or press numerous buttons to reach the correct department, leading to dissatisfaction and potential customer churn. However, our innovative approach offers a solution to this common problem. By implementing our method, customers no longer need to endure the lengthy and vexing process of department selection. Instead, their complaints are seamlessly and accurately directed to the appropriate department, saving time and reducing frustration. This approach holds great promise in enhancing the customer experience across a multitude of industries.

We introduce an encoder-decoder based methodology, wherein the encoder handles text embedding, and the decoder tackles the classification task. Mainly a discriminator network has been used as a decoder. Extensive experimentation with various encoders and discriminative decoders was conducted, leading to the selection of the most effective combinations. The encoders employed in our testing encompassed a range of options, including Distilbert base nli mean token [12], Sentence Encoder - all-mpnet-base-v2 [13], Sentence Encoder - Multi-qa-mpnet-base-dot-v1 [14], Distilbert Bert Base Uncased [15], Bert Base Uncase [16], Distilroberta-v1 [17], Roberta Model [18], and Universal Sentence Encoder [19]. Complementing these encoders, our decoder choices included Q-Learning, DQN Classifier, Artificial Neural Network, Support Vector Classifier (SVC) and KNN. Notably, the Universal Sentence Encoder paired with the Support Vector Classifier yielded highly promising results.

One common challenge encountered is the distinction between support-related and product-related tickets. The inherent overlap in ticket types poses difficulties in establishing ideal decision boundaries for classification models. To address this challenge, we introduce the concept of instance learning, which is elucidated in the later sections of this paper. In case of wrong triage, based on feedback, the model tends to learn from its mistakes.

In every corporate sector, the role of a Customer Support Manager (CSM) is pivotal, serving as a conduit between various departments and teams. Our research is geared toward empowering CSMs in the realms of customer support, product and finance related issues. We propose an automated triaging method aimed at reducing the workload of CSMs, streamlining the process of selecting the appropriate SME without delay and eliminating human errors.

In summary, this paper extends beyond traditional SME triage, offering a comprehensive approach that leverages advanced encoding and decoding techniques to enhance decision-making in organisations, particularly benefiting the critical role of Customer Support Managers.

**2.) About the Dataset**

For this study, we generated a proprietary sample dataset utilising the GPT 4 model developed by OpenAI. The methodology for formulating prompts was inspired by [11]. The prompt used in dataset creation was systematically structured, comprising 4 sequential formats - i) Context, ii) Query, iii) Instructions, and iv) Output Format.

The dataset's outcomes were stored in a JSON format for ease of access and analysis. To ensure unbiased training, an even distribution was maintained, with each class represented by 500 samples, specifically curated for Subject Matter Expert Triaging.

The subsequent image below provides an illustrative example of the interaction with OpenAI's latest Conversational AI model, GPT 4 web framework.

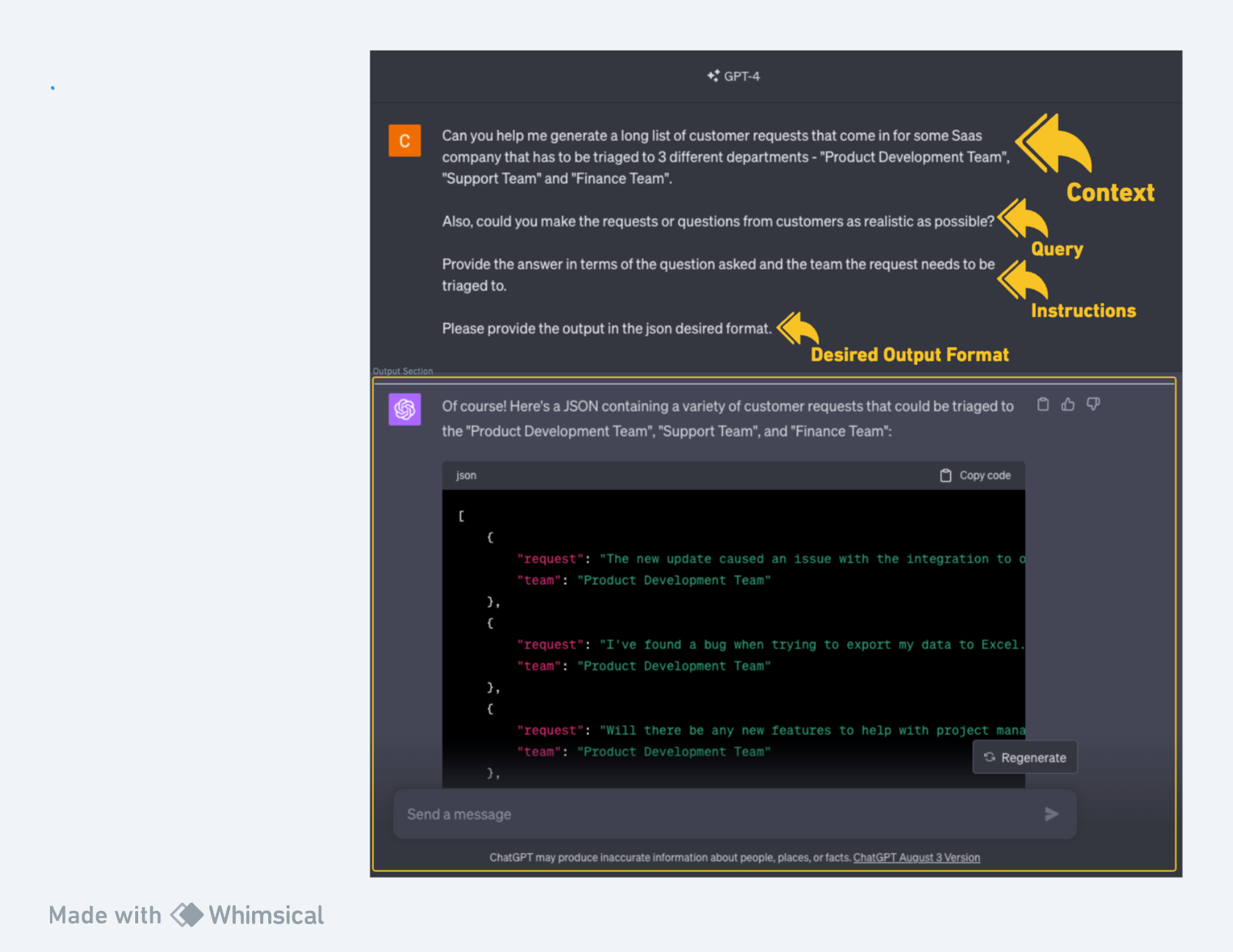


Figure 1 - Prompt used in GPT 4

**3.) Related Works**

Advancements in Artificial Intelligence (AI) and Machine Learning (ML) have significantly impacted various industries, necessitating the incorporation of novel technologies to sustain competitiveness and revenue generation. This section synthesises related studies, illuminating the intersection of AI, ML, and the role of Subject Matter Experts (SMEs) in varied domains.

Bauer et al. [1] explored ML adaptation in small and medium level industries, revealing a conspicuous lack of robust Data Science management and interdisciplinary collaboration. The study underscores the requisite for ML frameworks that enhance the business success factor by strategically navigating technological enablers and mitigating associated challenges.

Research reveals a nuanced relationship between SMEs and their impact on both training and predictive evaluations. Williams [2] delves into the efficacy of SMEs in technical training, exploring how their expertise influences learner outcomes. The authors had explained the effectiveness of assigning the technical professionals (Subject Matter Experts) for training their area of expertise through their skill and knowledge. Through paper, authors tried to explain the influence of tutors on student's effort. They examine problem based learning for academic achievement.

In a focused study at the University of Limburg Faculty of Health Sciences, Schmidt et al. [3] investigated the impact of tutors' subject-matter expertise on student achievement and effort in a problem-based curriculum. The findings reveal a nuanced role of content-expert tutors in boosting student achievement and self-directed study investment, particularly in initial curriculum years.

Another study by Bejar [4] examined SMEs' predictive capabilities concerning the difficulty and determination of items on a Standard English Written Test, revealing that even seasoned experts might fall short of replicating item statistics’ ratings after extensive training. The study examined up to which level the subject matter experts could predict the difficulty and determination of the items on an Standard English Written Test. On a human level the conducted study on 4 subject matter experts suggested that even after an extended period of training and practice does not reach up to a level that would be required to substitute the ratings of item statistics.

An innovative application of SME in health monitoring is delineated by Dang [5], wherein a deep learning model, enhanced with SME, was developed for equipment health monitoring systems. Utilizing a sliding window strategy, autocorrelation, and partial autocorrelation functions with elbow method, the study successfully identified an optimal window size, contributing to real-time health monitoring. The deep learning model has been trained using the hyperparameter tuning and an optimal set of parameters has been chosen.

Conversely, Robinson [6] presented an application of subject matter expertise in conjunction with Natural Language Processing (NLP) in the aviation safety reporting system database. The topic terms generated from the database are presented to the SME experts to judge the qualification and factors.

Kostas [7] navigates the rapidly growing esport industry, identifying requisite knowledge, skills, and abilities for aspiring professionals. Engaging subject matter experts through interviews, the study aimed to map essential competencies, guiding career trajectory in esports by aligning potential industry professionals with emerging career opportunities. The overarching objective was to delineate the requisite expertise and capabilities that potential esport industry professionals should possess to align with the career opportunities in the field.

Teferra et al. [8] pioneered a hybrid model validation approach, weaving quantitative validation metrics (consisting of scalar values) with qualitative subject matter expert (SME) evaluations via a Probabilistic Neural Network (PNN). The model effectively formalised relationships between input validation metrics and output SME scores as a classification problem using PNN classifiers and various validation metrics to quantitatively represent SME judgement, demonstrating its applicability in varied model validation contexts such as real-world shock qualification testing.

Christianto et al. [9] introduced the Smart Interpretable Model (SIM) framework, which champions transparency in AI technologies by formulating comprehensible fuzzy IF-THEN rules and membership functions, without necessitating advanced AI knowledge from users. SIM allows users to embed their prior knowledge throughout its application, facilitating nuanced insights into anomalous rules and feature contributions while ensuring trust and understanding in its predictive capabilities. Utilising various datasets, SIM’s performance and generated rules are validated against contemporary rule-based models, demonstrating its efficacy and versatility.

Lastly, Tecuci et al. [10] explored a knowledge-acquisition experiment where subject matter experts, devoid of prior knowledge-engineering experience, successfully instruct the Disciple-COA agent in critiquing courses of action - a challenge identified by DARPA's High-Performance Knowledge Bases program. It elucidates the COA problem, Disciple-COA architecture, expert-agent interaction, and automated knowledge-base development. The experiment’s success substantiates the efficacy of the Disciple approach in addressing knowledge acquisition bottlenecks.

**Table 1 -** Comparison and gaps between other related works and our proposal.

| **Related Research Papers** | **Comparison with our work** |
| --- | --- |
| Bauer et al. [1] delineates the indispensable role of AI in tackling market-related issues in organisations, the need of adopting Artificial Intelligence today in the organisations by analysing the problems related to market issues.  While it underscores the exigencies of adopting AI, our work further augments this by offering a pragmatic approach to implementing AI in SME fields. | Our paper also presents a way to implement AI in the field of SME. While the work in [1] underscores the exigencies of adopting AI, our work further augments this by offering a pragmatic approach to implementing AI in SME fields. |
| Williams [2], the pivotal role of SMEs in enriching knowledge delivery is highlighted. The authors advocate for the necessity of SMEs had explained the need of Subject Matter Experts for excellent delivery of knowledge and skill set. | Our work extends this discussion by presenting an automated mechanism for efficient Subject Expert allocation automation. |
| Schmidt et al. [3] underscore the significance of problem-based learning, employing subject-matter knowledge to shepherd academic students through their learning journey. The study is an analysis on problem based learning that has been done to use their subject-matter knowledge to guide academic students. | While our research orbits a similar thematic nucleus of subject-matter knowledge, it distinctively pivots toward an intensive exploration of triaging automation |
| Bejar [4] offers insight into establishing a threshold for pretesting item statistics at a human level. | In contrast, our paper introduces a methodology featuring a machine learning model designed to incrementally learn and adapt over time. |
| While [5] articulates a deep learning framework tailored for real-time health monitoring ML model by integrating subject matter expertise. | Our research pivots towards addressing issues pertinent to finance, support, and product domains, employing akin architectural strategy. Furthermore, our work introduces a novel methodology, underscoring a dynamic approach to hypothesis modification. |
| In the methodology articulated by Robinson [6], subject identification unfolds through a two-stage process: initially, topics are generated from samples within the aviation database, subsequently, conceptual themes are identified by Subject Matter Experts (SMEs). | Contrarily, our paper proposes a streamlined, singular step approach. By introducing a single sample, our method autonomously suggests the pertinent Subject Expert Department. |
| Kostas' investigation into esports, as documented in study [7], underscores the imperative of identifying and illuminating the pivotal knowledge, skills, and abilities (KSAs) indispensable for pursuing careers within the burgeoning esport industry. This is achieved by harnessing expert insights to delineate vital expertise and capacities, ensuring alignment with the ever-evolving demands of the industry. | In contrast, our work pivots towards training the model to mature into a subject matter expert, encapsulating a distinct, yet complementary, avenue of exploration in the area of expertise towards finance, support and product. |
| Teferra et al. [8] concentrated their efforts on developing an efficient Subject Matter Expert (SME) score, which serves as an evaluative mapping for model appraisal, that is, an efficient SME score as a scoring map for model evaluation. | Contrarily, our research pivots toward identifying the apt Subject Matter Expert, and not generating predicted scores on the extent and level of subject knowledge. |
| The methodology employed by Christianto et al. [9] centres on the utilisation of IF-THEN rules within a fuzzy logic framework. | Instead our research adopts a machine learning-oriented approach. We explore various models with a keen emphasis on enabling the devised model to proficiently operate across diverse environments and accommodate assorted test cases. |
| Tecuci et al. [10] explore the realm of interaction-based knowledge base development, with a distinct emphasis on facilitating subject matter experts in instructing a learning agent. This process is pivotal for the effective acquisition of a knowledge base. | Contrasting this, our research steers its focus toward the execution of knowledge transfer from a technical vantage point, specifically employing machine learning and deep learning methodologies. |

**4.) Proposed Methodology**

In this section, we provide a comprehensive overview of the suggested model's architecture. Our approach centres around training a robust base model capable of accurately triaging ticket requests. We then employ a single batch learning strategy to continually enhance the model's performance by correcting any erroneous predictions. Additionally, we emphasise the importance of periodic model retraining to maintain its effectiveness.

This section is further divided into 3 other subsections. In Section 4.1, we delve into the five distinct classification algorithms that play a pivotal role in our methodology. These algorithms serve as the foundation for our base model, enabling it to efficiently categorise ticket requests. Section 4.2 outlines the integration of feedback learning principles into our approach, coupled with the utilisation of pre-trained weights from the Support Vector Classifier. We elucidate how our model undergoes updates through a mini-batch strategy, ensuring its adaptability and accuracy. In Section 4.3, we emphasise on the necessity of timestamp-based retraining for the base model. We elucidate the rationale behind periodically retraining the model from scratch, ensuring its continued relevance and effectiveness over time.

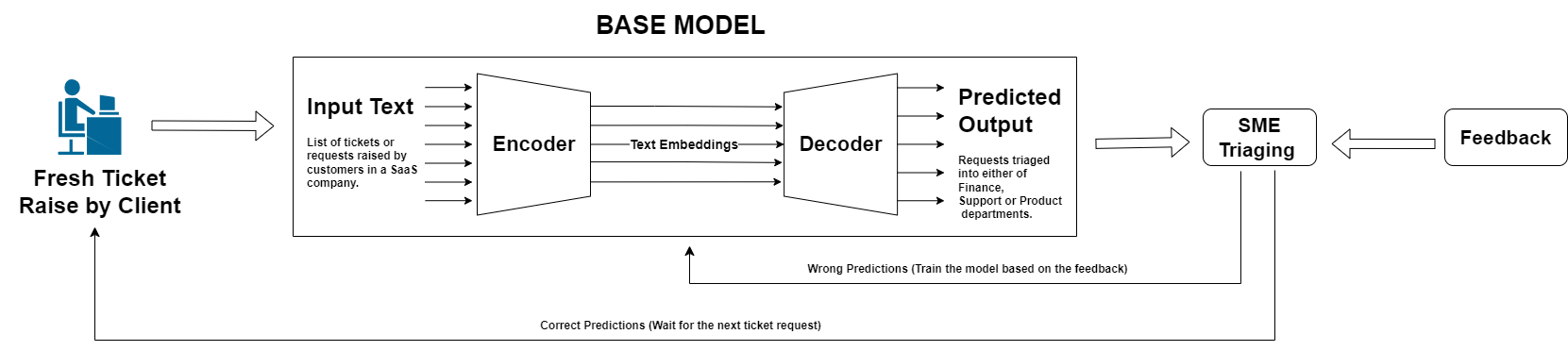
By structuring our proposed methodology in these three subsections, we provide a clear and organised framework for the readers to comprehend and appreciate the intricacies of our research approach.

**4.1) Classification Model**

A thorough analysis of categorization algorithms was conducted for the job of request triaging procedures inside a corporate ecosystem. The Q-learning method, the DQN Classifier in reinforcement learning, the neural network, Support Vector Classifier (SVC) and the K-Nearest Neighbours were all taken into consideration. These models are developed and tested to see how well they distribute incoming tickets to the appropriate departments. The goal was to make it easier for subject matter experts to efficiently resolve client inquiries. The study revealed that the Support Vector Classifier (SVC) produced the best outcomes in terms of the relevant evaluation measures. The next part consists of an examination of each algorithm, including its special qualities, advantages, and applicability for request triaging in a corporate environment.

The diagram presented below illustrates the foundational base model for training at a specific timestamp. The core objective of this model is to perform classification tasks, specifically categorising requests into various departments within the organisation. The model architecture consists of an encoder and a decoder. The encoder performs the operation of creating a text embedding and the decoder is a classifier model which discriminates among which of the following classes (Finance, Product and Support) it belongs.

The basic model is used by the SME for triaging new tickets that are raised by clients. After making incorrect predictions, the Customer Success Manager may offer input that helps the model improve and make more accurate classifications in the future.



**Figure 1 : Basic Model Architecture**

**4.1.1) Q - Learning for Classification**

Q-learning is a type of model-free reinforcement learning algorithm used to find the optimal action-selection policy for a given finite Markov decision process. Since our proposal is to make the model learn based on user feedback in case of wrong prediction, reinforcement learning has been used. Q-learning isn't typically used for classification problems directly but in order to frame the classification task in the reinforcement learning context, an environment needs to be created where the agent can interact, receive rewards based on its actions (predictions), and update its policy (Q-values).

Implementing Q-learning for text classification is non-trivial, but here's a high-level overview:

1. **State:** Each text sample can be a state.
2. **Action:** Predicting a class (Support Team, Finance Team, Product Development Team) can be an action.
3. **Reward:** If the prediction is correct, give a positive reward. If it's wrong, give a negative reward.

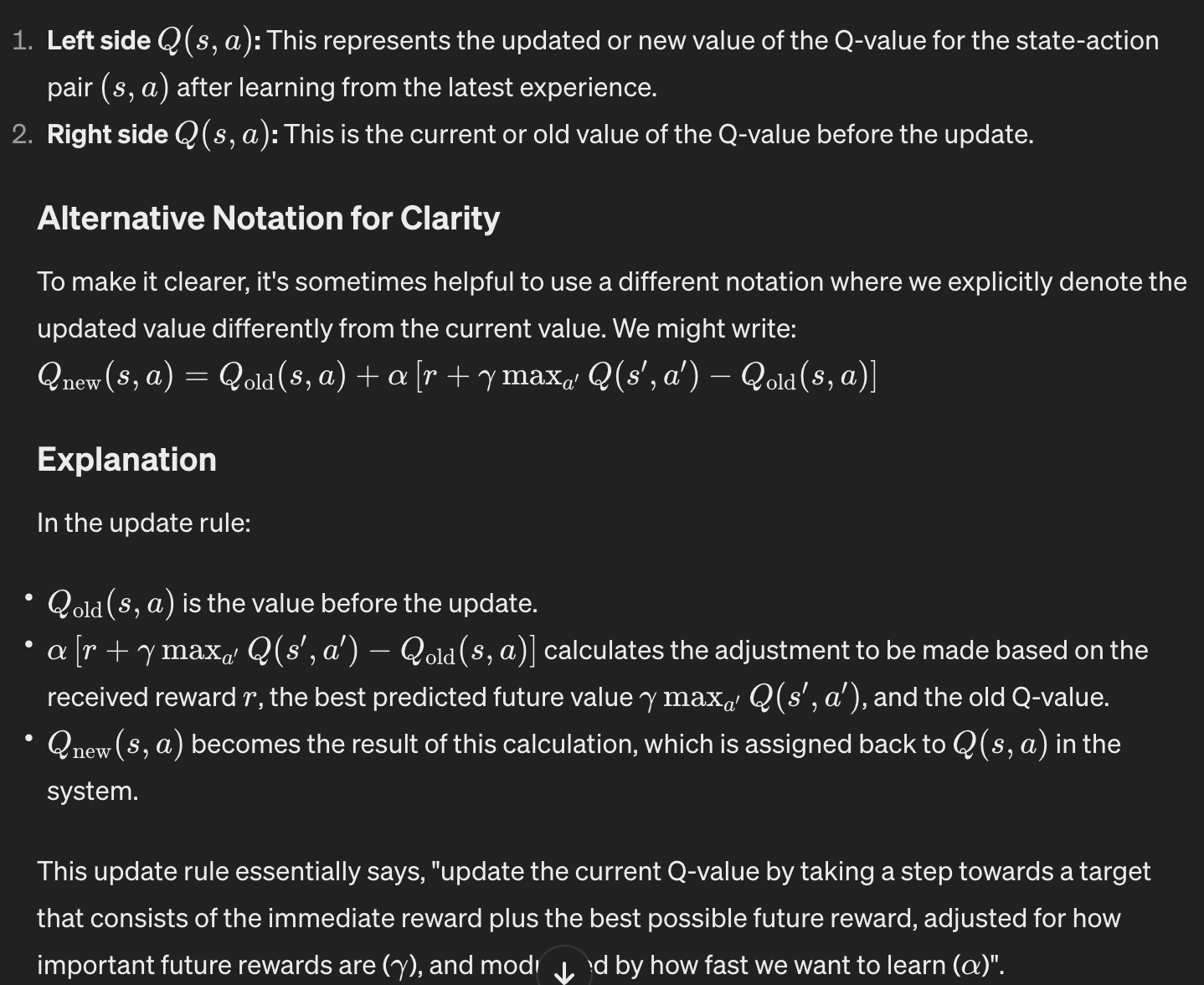
In order to select the optimal number of epochs, the Early Stopping method has been used.

For implementing Early Stopping, we keep a track of the validation loss and stop training until it reaches a state where it doesn't improve for a certain number of epochs. In this case, the loss is defined by the Q-learning update formula, and validation loss can be considered as the total negative rewards on the validation set.

**Here's how you can implement early stopping:**

1. Split the data into training, validation, and test sets.
2. During training, evaluate on the validation set every epoch.
3. If the validation loss doesn't improve for a certain number of epochs (patience), stop training.

The Q-learning update rule is given by the equation:   
**Q(s, a) = Q(s, a) + alpha \* [r + gamma \* max\_a' Q(s', a') - Q(s, a)]**, where Q(s, a) is the current Q-value of the state-action pair, alpha is the learning rate, r is the reward for taking, action a in state s, gamma is the discount factor, max\_a' Q(s', a') is the maximum Q-value for the next state s' over all possible actions a'.



**The Pseudo Code for the following Code is given below:-**

1. PREPARE THE DATA:

- Load the Encoder

- Encode X (data) using loaded Encoder and Encode y (labels) using LabelEncoder

- Split data into Train, Validation, and Test sets in the ratio of 70:15:15

1. Declare the following:

* n\_actions = 3 for 3 classes - Support Team, Finance Team, Product Development Team
* alpha = 0.01 # learning rate
* gamma = 0.5 # discount factor
* epsilon = 0.5 # exploration rate
* Initialize Q-table with zeros

1. DEFINE FUNCTION get\_reward(true\_label, predicted\_label):

IF true\_label EQUALS predicted\_label THEN

RETURN 1

ELSE

RETURN -1

END IF

1. SET patience FOR early stopping to a certain number (e.g., 12 as the number of epochs with no improvement after which training will be stopped).
2. SET best\_val\_loss to infinity
3. SET no\_improvement\_epochs to 0
4. SET maximum\_epochs to 200.
5. FOR epoch IN maximum\_epochs:

# Training Phase

FOR each state, action pair in Training set:

IF random\_number BETWEEN 0 and 1 is LESS THAN epsilon THEN

SELECT a random action (Exploration)

ELSE

SELECT the action with the highest Q-value (Exploitation)

END IF

Compute reward using get\_reward function

Update Q-value using the Q-learning update rule

END FOR

# Validation Phase

SET val\_loss to 0

FOR each state in Validation set:

SELECT the action with the highest Q-value

Compute reward using get\_reward function

UPDATE val\_loss using negative reward

END FOR

IF val\_loss is LESS THAN best\_val\_loss THEN

UPDATE best\_val\_loss with val\_loss

RESET no\_improvement\_epochs to 0

ELSE

INCREMENT no\_improvement\_epochs by 1

END IF

IF no\_improvement\_epochs is GREATER THAN OR EQUAL TO patience THEN

PRINT "Early stopping"

BREAK

END IF

END FOR

1. # Testing Phase
2. SET correct\_predictions to 0
3. FOR each state in Test set:

SELECT the action with the highest Q-value

IF selected action EQUALS true test label THEN

INCREMENT correct\_predictions

END IF

1. END FOR
2. Compute and Print classification report

**4.1.2) DQN Classifier**

DQN, which stands for Deep Q-Network, is a fundamental approach in reinforcement learning (RL) that combines deep learning techniques with the Q-learning algorithm. Q-learning is a model-free RL algorithm used to learn an optimal policy for an agent interacting with an environment to maximise its cumulative reward over time. The Q-value represents the expected cumulative reward that an agent can achieve by taking a particular action from a specific state and then following the optimal policy thereafter.

The DQN approach enhances Q-learning by using deep neural networks to approximate the Q-values. Traditional Q-learning methods often use tabular representations to store Q-values for each state-action pair, but this approach becomes impractical for large state or action spaces. Deep Q-Networks use neural networks to approximate the Q-values, allowing them to handle high-dimensional and continuous state spaces more effectively.

Here's a step-by-step explanation of the DQN approach:

1. **Experience Replay:** DQN employs an experience replay buffer to store a history of agent experiences (state, action, reward, next state). This replay buffer breaks the temporal correlation between consecutive experiences, making the learning process more stable.
2. **Q-Network Architecture:** DQN uses a deep neural network as its Q-network. The input to the network is the current state, and the output is a vector of Q-values for all possible actions in that state.
3. **Target Q-Network:** To stabilise the learning process and address the issues of moving targets in traditional Q-learning, DQN introduces a target Q-network. This network is a copy of the Q-network but with frozen weights that are updated periodically. The target Q-network is used to compute the target Q-values for the Bellman update.
4. **Loss Function and Training:** The loss function for DQN is typically the mean squared error between the predicted Q-values by the Q-network and the target Q-values generated using the Bellman equation. The Bellman equation calculates the expected Q-value considering the immediate reward and the maximum Q-value from the next state according to the target Q-network.
5. **Exploration Strategy:** DQN employs an exploration strategy, often epsilon-greedy, to balance the exploration of new actions and exploiting the current knowledge.
6. **Training Process:** During training, the DQN algorithm samples batches of experiences from the replay buffer. For each batch, it calculates the loss and updates the Q-network's weights using gradient descent. The target Q-network's weights are updated less frequently to stabilise the learning process.

Deep Q-Networks (DQN) are a type of reinforcement learning (RL) algorithm, originally designed for learning optimal policies in environments, such as games, where an agent interacts with an environment to maximise some notion of cumulative reward. DQN is not typically used for classification tasks as in an RL scenario, the agent's actions influence the future states it encounters, and the learning process involves finding a policy that maximizes the expected cumulative reward. However, in an attempt to adapt DQN for classification, the classification task has been treated as an Reinforcement Learning problem where the agent's actions are the class labels and the reward is a measure of how correct the agent's decision was. The implementation is a very basic adaptation of DQN to the classification problem. In a typical RL scenario, actions taken by the agent affect the future states it experiences, which isn't the case in a classification setting.

When it comes to classification tasks, the goal is different. In classification, the objective is to correctly assign labels to input data, and the actions taken by an agent (in this case, the model) do not influence the future states or inputs. The decision-making process in classification is more about mapping inputs to outputs rather than interacting with an environment over time. This is why using DQN in this manner can be seen as a force-fit, and there are better-suited algorithms for classification tasks.

Here's a conceptual outline for how we can use DQN for classification:

1. **State:** A given input data point (e.g., a sentence in the case of our data).
2. **Actions:** Possible class labels.
3. **Reward:** +1 if the agent correctly classifies the input data point, -1 otherwise.

**The Pseudo Code for the following Code is given below:-**

1. PREPARE THE DATA:

- Load the Encoder

- Encode X (data) using loaded Encoder and Encode y (labels) using LabelEncoder

- Split data into Train, Validation, and Test sets in the ratio of 70:15:15

1. DEFINE THE DQNCLASSIFIER CLASS:

CLASS DQNClassifier:

FUNCTION initialise(input\_dim, num\_classes, gamma, alpha):

Set input\_dim, num\_classes, gamma, alpha

Build the neural network model

FUNCTION build\_model:

Create a sequential neural network:

- Layer 1: Dense with 128 neurons, relu activation

- Layer 2: Dense with 64 neurons, relu activation

- Layer 3: Dense with 'num\_classes' neurons, linear activation

Compile the model using Adam optimizer and Mean Squared Error loss

Return the model

FUNCTION compute\_val\_loss(X\_val, y\_val):

Initialise 'val\_loss' to 0

For each state in X\_val:

Calculate the model's prediction for that state and update 'val\_loss'

Return average 'val\_loss'

FUNCTION train(X\_train, y\_train, X\_val, y\_val, epochs, batch\_size, patience):

Initialise best\_val\_loss to infinity and no\_improvement\_epochs to 0

FOR each epoch up to 'epochs':

FOR each state in X\_train:

Use the model to predict Q-values for the state

Update the Q-value for the action taken using Q-learning formula

Train the model on this state and its updated Q-values

Calculate validation loss using compute\_val\_loss

If validation loss has improved:

Update best\_val\_loss and reset no\_improvement\_epochs

Else:

Increment no\_improvement\_epochs

If no\_improvement\_epochs equals patience:

Break training loop (Early Stopping)

FUNCTION predict(X):

Return action with highest Q-value for each sample in X using the model

1. TRAIN AND EVALUATE THE MODEL:

- Instantiate DQNClassifier with appropriate input dimensions and number of classes.

- Train the model using train data and validate using test data.

- Predict the classes for test data using the trained model.

- Compute accuracy and classification report by comparing predicted labels with true test labels.

- Print accuracy and classification report.

**4.1.3) Artificial Neural Network**

An Artificial Neural Network (ANN), simply termed a "Neural Network", is a computational model inspired by the biological neural structures present in the human brain. These networks are foundational in the domain of machine learning, especially deep learning. An ANN consists of interconnected units called neurons, organised into layers: an input layer, several hidden layers, and an output layer. During the learning process, data is fed into the network, and the network produces an output. This output is compared to the desired outcome, and the connection strengths, or weights, between neurons are iteratively adjusted using algorithms like backpropagation to reduce the error. The more layers or neurons a network has, the more complex patterns it can recognize, but this also raises challenges like the need for more data and computational resources, and the potential for overfitting.

The Neural Network is a type of ANN designed for multi-class classification on the SME dataset. The network structure consists of multiple layers, with each layer containing a set number of neurons. Between layers, operations like BatchNormalization and Dropout are employed. BatchNormalization helps in achieving a stable learning process by normalising the activations of the neurons, while Dropout prevents overfitting by randomly setting a fraction of input units to 0 during training. The activation functions like 'relu' and 'softmax' determine the output of each neuron. The model is trained using the 'adam' optimizer and 'categorical\_crossentropy' as the loss function, which is standard for multi-class classification tasks. After training, the model's performance is evaluated based on its accuracy on a test dataset, and a detailed classification report is generated. In essence, this Neural Network is a multi-layered computational model trained to classify data points in the SME dataset into one of three categories.

**The Pseudo Code for the Neural Network model training is given below:-**

1. PREPARE THE DATA:

- Load Universal Sentence Encoder from TensorFlow Hub

- Encode X (data) using Universal Sentence Encoder to get 'X\_encoded'

- Encode y (labels) using LabelEncoder to get 'y\_encoded'

- One-hot encode 'y\_encoded' to get a categorical matrix representation

- Split data into Train, Validation, and Test sets in the ratio of 70:15:15

1. SET UP THE NEURAL NETWORK:

- Initialise a Sequential model

- Add a Dense layer with 512 units, ReLU activation, and input dimension of 512

- Add a BatchNormalization layer

- Add a Dropout layer with 0.6 dropout rate

- Add another Dense layer with 512 units and ReLU activation

- Add another BatchNormalization layer

- Add another Dropout layer with 0.5 dropout rate

- Add a Dense layer with 64 units and ReLU activation

- Add another BatchNormalization layer

- Add another Dropout layer with 0.5 dropout rate

- Add a final Dense layer with a number of units equal to the number of categories in 'y\_train', using softmax activation

1. COMPILE AND TRAIN THE MODEL:

- Compile the model with 'categorical\_crossentropy' loss, 'adam' optimizer, and track 'accuracy' as a metric

- Train the model on 'X\_train' and 'y\_train' with a specified number of epochs and batch size, displaying progress for each epoch

1. EVALUATE THE MODEL:

- Evaluate the model on 'X\_test' and 'y\_test', storing the resulting loss and accuracy

- Print the loss and accuracy

1. MAKE PREDICTIONS:

- Predict class probabilities for 'X\_test' using the trained model

- Convert the predicted class probabilities to class labels by finding the index with the maximum value for each sample

- Convert the one-hot encoded 'y\_test' back to class labels in the same manner

1. GENERATE AND DISPLAY REPORT:

- Create a classification report comparing the predicted class labels to the true class labels

- Display the classification report

**4.1.4) K-Nearest Neighbours**

For the classification purpose, KNN algorithm is one of the most efficient algorithm. KNN classifies the data based on the nearest similar point. To efficiently manage and retrieve embeddings, a FAISS vector store is utilized as its very efficient in similarity searches. FAISS allows the creation of indexes to organize and search through vectors efficiently and its faster vectors in a collection that are most similar to a query vector. The FAISS index is created to store the embeddings, providing a mechanism for fast and accurate k-nearest neighbor searches. The KNN search is employed for text classification. Given an input sentence, the text encoder generates an embedding, which is then used to query the FAISS index. The KNN search identifies the nearest neighbors, allowing for the classification of the input text based on the metadata associated with these neighbors.

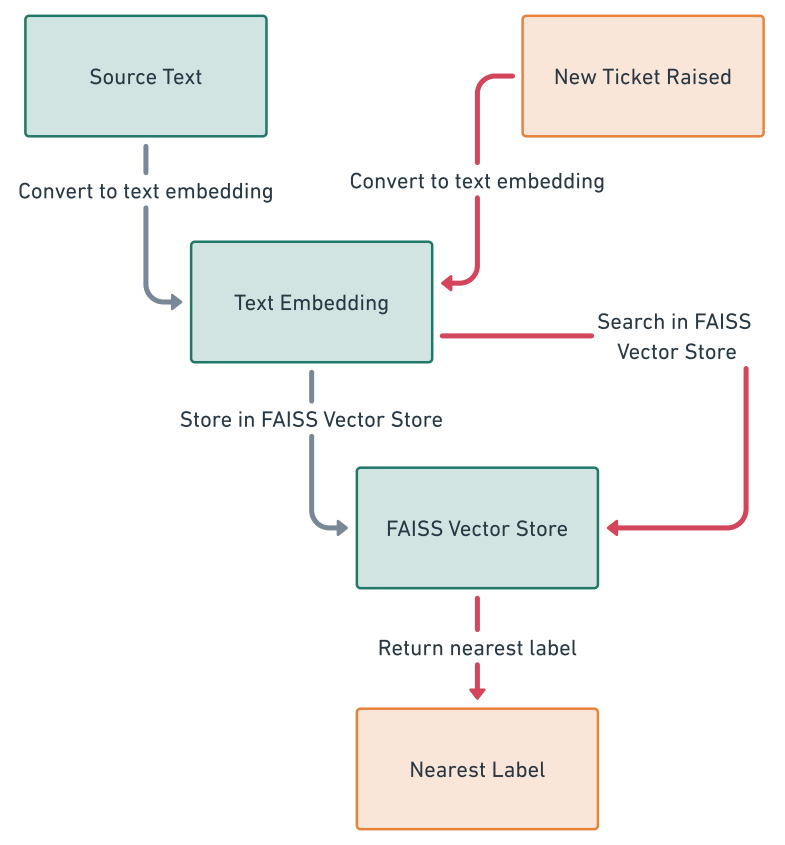


Figure 2 : Implementation of KNN using FAISS vector store

**The Pseudo Code for the KNN model training is given below:-**

1. PREPARE THE DATA:

- Load the Encoder

- Split data into Train, Validation, and Test sets in the ratio of 70:15:15

- Import the Langchain[20] package and then store Train dataset as documents, text as page\_content and labels as metadata

1. CREATE A FAISS INDEX:

- Convert the content of each Document to embeddings

- Save the text embeddings into the FAISS vectorstore.

1. MODEL EVALUATION:

- Predict the class labels for 'X\_test' using KNN Algorithm (k=1)

- Generate a detailed classification report.

**4.1.5) SVC**

The Support Vector Classifier (SVC) is a supervised machine learning algorithm used primarily for classification tasks. It operates on the idea of finding a hyperplane that best divides a dataset into classes. The optimal hyperplane is the one that maximises the margin between the two classes. The instances that are closest to this hyperplane are known as "support vectors", and they play a critical role in determining the position and orientation of the hyperplane. Mathematically, SVC tries to solve an optimization problem to find the hyperplane which maximises this margin while incurring minimal classification error. In cases where the data is not linearly separable, the SVC employs the kernel trick, mapping the input data into higher-dimensional spaces where a separating hyperplane can be found. By doing so, SVC is able to classify non-linear data efficiently.

In this study, we present an innovative approach to enhance the performance of Support Vector Classifier (SVC) models by integrating principles reminiscent of reinforcement learning. Our methodology seeks to harness real-time feedback mechanisms, ensuring iterative refinement and model adaptability. In our methodology, we employed a Support Vector Classifier (SVC) and refined its performance via hyperparameter optimization. Subsequent to the training phase, we tested the model's robustness and adaptability by introducing it to single-sentence predictions. Upon receiving an input sentence, the SVC predicts an output; if this output is judged incorrect by the user, immediate feedback is solicited regarding the correct classification. In cases where the model's prediction was incorrect, we instituted an interactive feedback mechanism: the user was prompted to provide the correct output. The model was then fine-tuned further by training it on this single-instance feedback batch. When the model undergoes a rapid retraining phase using this singular data instance, thereby incorporating the newfound knowledge. This iterative approach is designed to incrementally enhance the model's precision and adaptability to evolving data patterns. This continuous feedback loop allows the SVC to dynamically adjust its decision boundaries, optimising its performance and demonstrating an ability to learn and adapt in real-time.

**The Pseudo Code for the SVC model training is given below:-**

1. PREPARE THE DATA:

- Load the Encoder

- Encode X (data) using Encoder and Encode y (labels) using LabelEncoder

- Split data into Train, Validation, and Test sets in the ratio of 70:15:15

1. SET UP PARAMETERS FOR GRID SEARCH:

- Define a parameter grid with values for:

- 'C': a list of regularisation strengths [5,6,7,8,9]

- 'kernel': a list of kernel types ['linear', 'rbf', 'sigmoid']

- 'gamma': a list with values ['scale', 'auto']

1. GRID SEARCH SETUP:

- Initialise an instance of the Support Vector Classifier as 'classifier'.

- Set up StratifiedKFold cross-validation with 5 splits, shuffling enabled and a set random state.

- Initialize GridSearchCV with the classifier, parameter grid, the cross-validation strategy, and 'accuracy' as the scoring metric.

1. PERFORM GRID SEARCH:

- Fit the GridSearchCV instance on the training data 'X\_train' and 'y\_train'.

- Retrieve the best parameters and the best estimator from the grid search.

1. EVALUATE THE BEST MODEL:

- Predict the class labels for 'X\_test' using the best estimator.

- Calculate the accuracy by comparing the predicted labels to the true test labels.

- Generate a detailed classification report.

1. DISPLAY RESULTS:

- Print the best parameters, accuracy, and the classification report.

**4.2) Using the Pre-training model for Single Sentence Prediction**

In Section 4.1, we illustrate the process of training a classification model for triage tasks. An evaluation of five distinct models is presented in Section 5, with the Support Vector Machine (SVM) emerging as the top-performing choice when paired with the Universal Sentence Encoder. Consequently, we have preserved this well-performing SVM model for future use in classification of the corporate requests raised by the customers. Additionally, we implemented a feature that prompts users for immediate feedback regarding the model's prediction accuracy. The user (CSMs) is prompted to provide a brief response about the accuracy of the forecast. To help the algorithm learn from incorrect predictions, we ask the user for the proper label output.

In our study, we didn’t use any reinforcement learning algorithm for the model refinement, rather we introduced the idea of using feedback based learning for training on a specific batch of data. This batch comprises 10 random samples from finance team, product development team, and support team labels, in addition to a single wrong predicted output, totalling a batch size of 31. This strategy aids in mitigating the risk of model overfitting, as elucidated in Table 1.

To the best of our knowledge, no human made machine learning model is ideal in this world till date with 100% perfection. Given this, our SVC model, like others, is prone to occasional inaccuracies and is likely to give a wrong output. Rather than accumulating these inaccuracies over a period for a comprehensive re-training session, we advocate for an immediate model update mechanism. We tend to save such ticket requests in case of wrong answer predictions. Instead of storing the wrongly predicted data till a certain period of time and then re-training the model to capture the latest information and correct itself on wrong predictions, we here propose a method to re-train the model with a batch having that single instance so that the model gets updated instantly. Any delay in incorporating user feedback or waiting too long between updates, the model might provide many wrong predictions until the next update.

Our recommendation for prompt model updates stems from the understanding that without timely corrections, the model is predisposed to give analogous requests triaging or wrong predictions. The reason being we suggest a quick re-training of the model is because the model will continue to give a wrong predicted output on such similar tickets where the model failed to give a correct triaged output earlier. This could result in frequent type-1 or type-2 errors until the model is re-trained later. In order to save this issue and prevent the retraining of the model everytime, we had saved the pre-trained SVC model. The main purpose of this pre-trained model is to predict a sentence and then re-train the model based on feedback. For wrongly predicted data, a batch update process has been adopted to prevent the issue of overfitting to individual feedback instances. It's computationally more efficient than retraining the model from scratch every time there's new feedback. This approach leverages the Universal Sentence Encoder to transform the feedback sentence along with 30 other random sentences into embeddings and then retrain the model using this data on the same parameters used earlier while training the SVC model . Once training is done, the model is saved back to the directory.

The following instructions followed by the model is as follows:-

1. Loads the pre-trained SVC model
2. Accepts a sentence (ticket request) from the user.
3. Predicts the label for that sentence.
4. If the prediction is wrong, it asks for feedback (correct label) from the user.
5. Fine-tune the model on that sentence with the provided label along with the rest of the data having a batch size of 31. The whole data is fit into the pre-trained model using the hyperparameters.
6. Saves the retrained model back.

**Table1** to Justify the reason behind choosing a batch size to re-train a pre-trained model and to analyse different strategies for retraining a model based on new or corrected data:

| **Different Strategies for re-training the model** | **Advantage** | **Disadvantage** |
| --- | --- | --- |
| Strategy 1 - Retraining on the Whole Dataset | * The model gets refreshed with the entire dataset, which ensures that it retains the knowledge it has previously gained. * Useful if the dataset isn't large, as training might not take a significant amount of time. | * Can be computationally expensive and time-consuming as the dataset grows. * Redundant computations for the instances that the model has already seen. |
| Strategy 2 - Retraining Only on the New Data Instance | * Very fast, as there exists only one instance for training. | * The model might forget the previous data. * If the new data is noisy or an outlier, the model might become biased towards it. |
| Strategy 3 - Retraining on a Batch (a mix of old and new data) - current approach | * A balance between retraining on the whole dataset and just the new instance. * Can incorporate recent feedback while also revisiting older data to prevent forgetting. | * The choice of batch size and which old data to include might need experimentation. * Some computational overhead, though not as much as retraining on the entire dataset. |

These strategies are primarily contingent upon the dataset's size and the available computational resources, serving as essential constraints. When dealing with a smaller dataset, employing the retraining approach on the entire dataset (strategy 1) can yield satisfactory results. Conversely, when dealing with a large dataset but facing computational limitations, one might opt for batch retraining (strategy 3). In this approach, a recent data batch, inclusive of new feedback, is combined with randomly selected older data. This approach guarantees a seamless integration of recent feedback with established knowledge. In practical scenarios, especially those involving substantial datasets, a batch-wise or mini-batch update is often favoured. This strategy strikes a balance between infusing the model with new data and preserving the foundational knowledge derived from the initial, larger dataset.

**4.3) Time-Stamp Consideration**

Over time, there is a significant likelihood that our underlying hypothesis may evolve. This evolution can be driven by various factors, including:

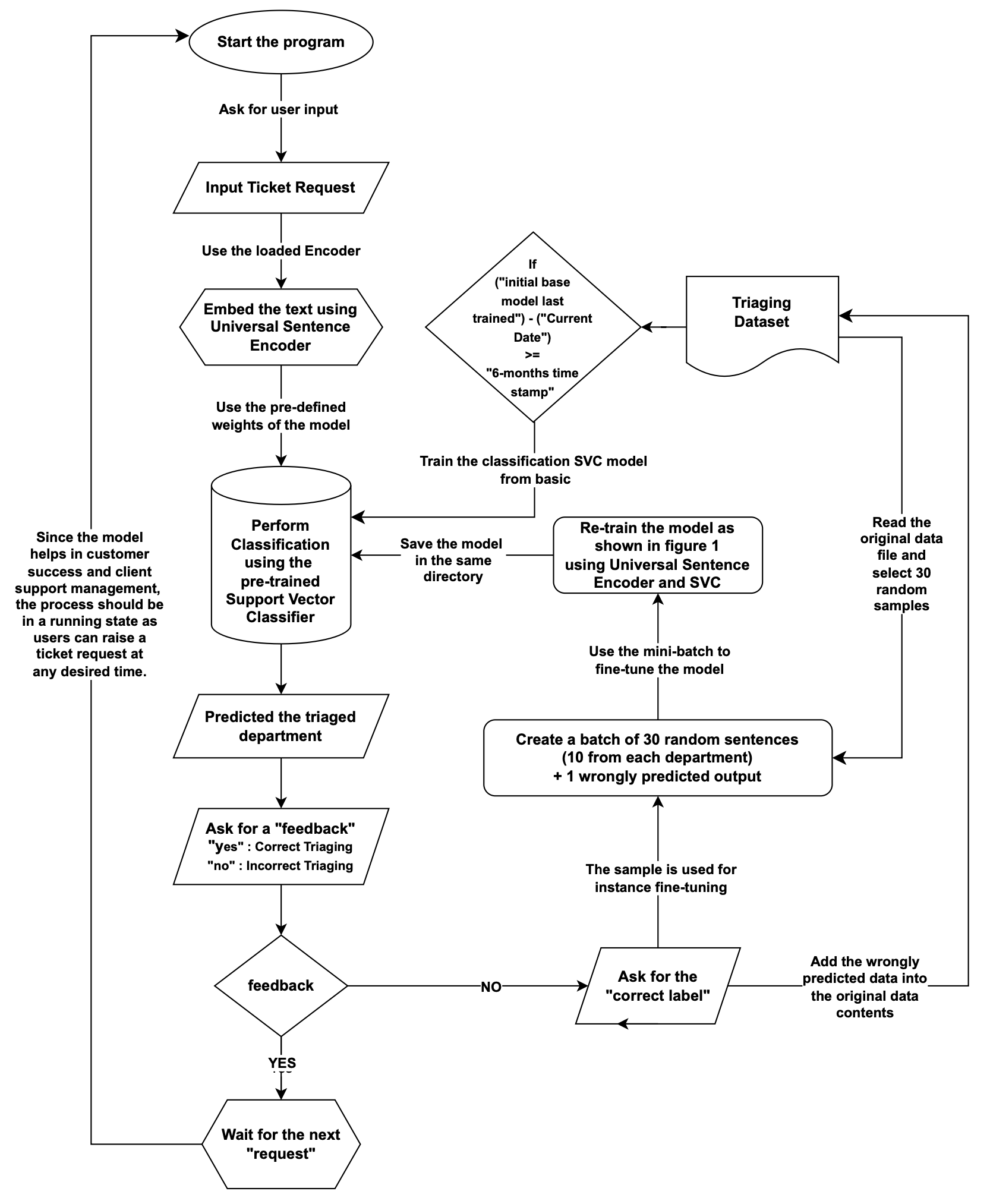
1. **Shifts in Product Design** - Changes in the fundamental structure, features, or aesthetics of the product can lead to shifts in user expectations and experiences.
2. **Terminological Updates** - Alterations in the language or terminology used within the product, such as updates to user interface labels or the introduction of new industry-specific terms, can impact user interactions and feedback.
3. **Ideological Adjustments** - Evolving company philosophies, mission statements, or core values can influence the way the product is perceived and used by both customers and the organisation itself.
4. **Functional Modifications** - Adjustments to the product's functionalities, such as the addition of new features or improvements to existing ones, can change the user interaction patterns and feedback generated.
5. **Market Dynamics** - External factors like shifts in market trends, competitive landscape, or emerging technologies can necessitate changes in the product to remain relevant and competitive.
6. **User Feedback and Behavior** - Ongoing user feedback and behaviour patterns can provide valuable insights into areas where the product may need adjustments or enhancements.

To effectively adapt to these evolving factors, users can set a time period after which the model should undergo a complete retraining with hyperparameter tuning. Training from scratch within the specified timeframe ensures that the model fully integrates all available data and helps mitigate the potential compounding of errors that can result from frequent updates.

In our specific context, we have chosen a 6-month time duration for model retraining. This interval allows for a comprehensive grasp of evolving product dynamics and user feedback. However, the optimal time period can vary depending on the organisation's nature. For instance, in a dynamic startup environment where product changes occur frequently, a shorter 2-month time period may be more suitable. Conversely, in a larger and more stable MNC setting, a 1-year time period might be appropriate, allowing ample time to observe structural, use case, and feature changes along with corresponding shifts in our underlying hypothesis.

Neglecting to set a time period carries the risk of overfitting the model to recent data. This can lead to potential biases, especially when users repeatedly input similar feedback, potentially skewing the model's performance. Hence our suggested proposal is to update the initial model at a particular timestamp following the architecture shown in figure1.

**Figure 2: Overall Architectural Diagram**

****

**5.) Results**

The suggestive encoder and decoder method has shown promising results. These following models have been trained on the cloud computing hardware setup platform (AWS Amazon SageMaker), instance type ml.m5d.8xlarge and available resources vCPU 32, 128GB memory RAM and 2 x 600 GB NVMe SSD storage. The shorthand notations for evaluation metrics are - Precision (P), Recall (R), F1-score (F), Accuracy, and FT(Finance Team), PDT(Product Development Team), ST(Support Team)

**Table 2 -**  Demonstration of classification reports generated for the various encoders and decoders (discriminative classifiers) used in the paper.

|  | **Q-Learning** | **DQN Classifier** | **Neural Network** | **SVC** | **KNN using FAISS** |
| --- | --- | --- | --- | --- | --- |
| **Sentence Encoder -**  **Distilbert base nli mean token [12]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.30 | 0.38 | 0.33 | | PDT | 0.27 | 0.21 | 0.23 | | ST | 0.26 | 0.25 | 0.25 |   Accuracy = 0.28 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.09 | 0.12 | 0.10 | | PDT | 0.26 | 0.52 | 0.35 | | ST | 0.22 | 0.03 | 0.05 |   Accuracy = 0.23 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.88 | 0.90 | 0.89 | | PDT | 0.83 | 0.76 | 0.79 | | ST | 0.66 | 0.75 | 0.70 |   Accuracy = 0.78 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.93 | 0.93 | 0.93 | | PDT | 0.81 | 0.86 | 0.83 | | ST | 0.82 | 0.77 | 0.79 |     Accuracy = 0.85 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.97 | 0.90 | 0.94 | | PDT | 0.77 | 0.88 | 0.82 | | ST | 0.81 | 0.74 | 0.78 |     Accuracy = 0.84 |
| **Sentence Encoder - all-mpnet-base-v2 [13]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.33 | 0.32 | 0.32 | | PDT | 0.32 | 0.27 | 0.29 | | ST | 0.36 | 0.24 | 0.29 |   Accuracy = 0.30 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.11 | 0.12 | 0.11 | | PDT | 0.30 | 0.58 | 0.40 | | ST | 0.27 | 0.02 | 0.04 |   Accuracy = 0.26 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.96 | 0.97 | 0.96 | | PDT | 0.85 | 0.81 | 0.83 | | ST | 0.72 | 0.79 | 0.75 |   Accuracy = 0.85 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.95 | 0.96 | | PDT | 0.83 | 0.90 | 0.86 | | ST | 0.86 | 0.81 | 0.84 |   Accuracy = 0.88 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.98 | 0.98 | | PDT | 0.82 | 0.86 | 0.84 | | ST | 0.82 | 0.79 | 0.80 |   Accuracy = 0.87 |
| **Sentence Encoder - Multi-qa-mpnet-base-dot-v1 [14]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.32 | 0.40 | 0.26 | | PDT | 0.33 | 0.28 | 0.30 | | ST | 0.35 | 0.25 | 0.29 |   Accuracy = 0.32 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.14 | 0.14 | 0.14 | | PDT | 0.30 | 0.56 | 0.39 | | ST | 0.27 | 0.02 | 0.04 |   Accuracy = 0.27 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.96 | 0.97 | 0.96 | | PDT | 0.86 | 0.79 | 0.83 | | ST | 0.70 | 0.82 | 0.76 |   Accuracy = 0.84 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.98 | 0.98 | | PDT | 0.86 | 0.90 | 0.88 | | ST | 0.87 | 0.83 | 0.85 |   Accuracy = 0.90 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.95 | 0.96 | | PDT | 0.85 | 0.90 | 0.87 | | ST | 0.84 | 0.81 | 0.83 |   Accuracy = 0.88 |
| **Distilbert Bert Base Uncased [15]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.33 | 0.42 | 0.37 | | PDT | 0.35 | 0.32 | 0.33 | | ST | 0.35 | 0.26 | 0.30 |   Accuracy = 0.35 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.15 | 0.14 | 0.14 | | PDT | 0.31 | 0.62 | 0.41 | | ST | 0.29 | 0.04 | 0.07 |   Accuracy = 0.28 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.97 | 0.97 | 0.97 | | PDT | 0.89 | 0.77 | 0.83 | | ST | 0.70 | 0.81 | 0.75 |   Accuracy = 0.85 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.98 | 0.98 | | PDT | 0.88 | 0.88 | 0.88 | | ST | 0.85 | 0.85 | 0.85 |   Accuracy = 0.90 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.97 | 0.93 | 0.95 | | PDT | 0.80 | 0.84 | 0.82 | | ST | 0.79 | 0.79 | 0.79 |   Accuracy = 0.85 |
| **Bert Base Uncase [16]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.32 | 0.40 | 0.36 | | PDT | 0.33 | 0.29 | 0.31 | | ST | 0.31 | 0.24 | 0.27 |   Accuracy = 0.32 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.13 | 0.10 | 0.11 | | PDT | 0.30 | 0.60 | 0.40 | | ST | 0.29 | 0.03 | 0.05 |   Accuracy = 0.25 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.91 | 0.92 | 0.91 | | PDT | 0.85 | 0.82 | 0.83 | | ST | 0.69 | 0.74 | 0.71 |   Accuracy = 0.83 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.95 | 0.93 | 0.94 | | PDT | 0.84 | 0.94 | 0.88 | | ST | 0.86 | 0.77 | 0.81 |   Accuracy = 0.88 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.97 | 0.93 | 0.95 | | PDT | 0.78 | 0.88 | 0.83 | | ST | 0.84 | 0.77 | 0.80 |   Accuracy = 0.86 |
| **Distilroberta-v1 [17]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.35 | 0.41 | 0.38 | | PDT | 0.32 | 0.25 | 0.28 | | ST | 0.31 | 0.33 | 0.32 |   Accuracy = 0.31 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.11 | 0.08 | 0.09 | | PDT | 0.32 | 0.56 | 0.41 | | ST | 0.28 | 0.07 | 0.11 |   Accuracy = 0.26 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.96 | 0.94 | 0.95 | | PDT | 0.88 | 0.73 | 0.82 | | ST | 0.71 | 0.82 | 0.76 |   Accuracy = 0.85 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.98 | 0.98 | | PDT | 0.86 | 0.90 | 0.88 | | ST | 0.87 | 0.83 | 0.85 |   Accuracy = 0.90 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.95 | 0.98 | 0.96 | | PDT | 0.85 | 0.84 | 0.85 | | ST | 0.83 | 0.83 | 0.83 |   Accuracy = 0.88 |
| **Roberta Model [18]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.37 | 0.39 | 0.38 | | PDT | 0.26 | 0.20 | 0.23 | | ST | 0.26 | 0.25 | 0.25 |   Accuracy = 0.29 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.09 | 0.08 | 0.08 | | PDT | 0.24 | 0.56 | 0.33 | | ST | 0.29 | 0.02 | 0.04 |   Accuracy = 0.23 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.95 | 0.93 | 0.94 | | PDT | 0.89 | 0.75 | 0.81 | | ST | 0.62 | 0.83 | 0.71 |   Accuracy = 0.83 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.93 | 0.95 | | PDT | 0.88 | 0.84 | 0.86 | | ST | 0.72 | 0.83 | 0.77 |   Accuracy = 0.87 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.95 | 0.90 | 0.93 | | PDT | 0.76 | 0.84 | 0.80 | | ST | 0.80 | 0.74 | 0.77 |   Accuracy = 0.83 |
| **Universal Sentence Encoder [19]** | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.33 | 0.38 | 0.36 | | PDT | 0.35 | 0.31 | 0.33 | | ST | 0.32 | 0.32 | 0.32 |   Accuracy = 0.34 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.11 | 0.10 | 0.10 | | PDT | 0.34 | 0.61 | 0.43 | | ST | 0.31 | 0.09 | 0.13 |   Accuracy = 0.28 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.95 | 0.93 | 0.94 | | PDT | 0.90 | 0.84 | 0.88 | | ST | 0.74 | 0.83 | 0.78 |   Accuracy = 0.87 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.97 | 0.93 | 0.95 | | PDT | 0.89 | 0.96 | 0.92 | | ST | 0.89 | 0.85 | 0.87 |   Accuracy = 0.91 | |  | P | R | F | | --- | --- | --- | --- | | FT | 0.98 | 0.95 | 0.96 | | PDT | 0.84 | 0.88 | 0.86 | | ST | 0.83 | 0.81 | 0.82 |   Accuracy = 0.88 |

**Table 3 -** Approx. Time Taken by Decoder to train after Universal Sentence Encoder [19] text embeddings

| **Discriminator Network** | **Approx. Average hours taken to train the model** |
| --- | --- |
| Q-Learning | 1.13 |
| DQN Classifier | 4.68 |
| Neural Network | 0.18 |
| K-Nearest Neighbour | 0.001 |
| Support Vector Classifier | 0.11 |

There could be several reasons why the Support Vector Classifier, Neural Network and KNN performed better in terms of precision, recall, and F1-score compared to Q-learning and DQN Classifier:

1. Q-learning and a DQN (Deep Q-Network) Classifier were considered for a classification problem, as these are techniques generally associated with reinforcement learning, which is typically used for decision-making in sequential tasks. On the other hand, SVMs, NN and KNN are supervised learning models used for classification and regression analysis.
2. SVM is better suited for classification as it is inherently a classification algorithm, and it might be better suited for the classification task at hand compared to Q-learning and DQN, which are more tailored for reinforcement learning tasks. Q-learning and DQN have different learning paradigms which are based on learning from interaction with an environment and trying to maximise a cumulative reward, which might not align well with the objectives of a typical classification problem.
3. SVM Handles high-dimensional data distribution well. SVM might perform better especially if it’s linearly separable or close to linearly separable in the high-dimensional space. However, they are generally not affected by non-stationarity in the same way as Q-learning and DQN because they are trained on a fixed dataset and do not rely on sequential or temporal information.

Whereas Q-learning and DQN might struggle if the data distribution is non-stationary, which is often the case in real-world classification tasks. The term "non-stationary" in this context refers to data distributions that change over time or are not constant. In other words, the statistical properties of the process generating the data (such as the mean, variance, or probability distribution) are not constant but change over time.

1. For Q-learning and Deep Q Networks (DQN), which are reinforcement learning algorithms, non-stationarity can be particularly challenging. This is because these algorithms try to learn an optimal policy based on past experiences, and if the environment is changing, the past experiences may no longer be relevant or accurate for making future decisions.
2. SVMs can perform well even with a smaller amount of data also it has built-in regularisation parameters, which might have helped in preventing overfitting, resulting in better generalisation to the test data, whereas deep learning-based methods like DQN usually require large amounts of data to perform well.
3. Q-learning and DQN require a carefully designed reward structure, which might not map well to the objectives of a classification task. SVM, on the other hand, directly optimises for classification accuracy (or other related metrics).

In summary, the Support Vector Classifier has performed better because it is inherently suited for classification tasks and might have been better configured for the problem at hand. Q-learning and DQN, being reinforcement learning algorithms, have different objectives and might not have aligned well with the classification task, resulting in lower performance metrics.

**6.) Conclusion**

This paper deals with the dynamic realm of harnessing Subject Matter Experts (SMEs) for organisational success and innovation in the age of rapid technological advancements. We embarked on a significant exploration by developing and assessing five distinct models - Q-Learning, Deep Q-Network (DQN) Classifier, Artificial Neural Network (ANN), Support Vector Classifier (SVC) and KNN - to efficiently allocate SMEs to suitable departments. Our hybrid model, blending feedback learning principles with SVC, emerged as a potential paradigm shift, bridging cutting-edge technology with domain-specific expertise.

Beyond traditional SME triage, our approach extends to revolutionising customer service experiences across various industries. By seamlessly directing customer inquiries to the right department, we mitigate frustration and enhance overall satisfaction. Furthermore, our encoder-discriminator based methodology, combining encoders like the Universal Sentence Encoder with classifiers, showcased promising results.

We introduced instance learning to address the challenge of classifying support-related and product-related tickets, empowering Customer Support Managers (CSMs) in their pivotal roles. Moreover, we underscored the significance of timely model updates, which serve to prevent inaccuracies and enhance decision-making. Our research remains adaptable to evolving factors such as product design, terminology, ideology, functionality, market dynamics, and user feedback. Setting a timestamp for periodic model retraining ensures continued relevance and effectiveness.

This comprehensive framework leverages advanced machine learning techniques for optimising SME allocation, benefiting not only triaging tasks but also enhancing organisational decision-making processes. It lays the groundwork for a more agile, innovative, and customer-centric future across diverse industries.

**7.) Limitation of our work**

A drift detection mechanism which employs statistical tests and or algorithm to identify the concept drift problems automatically along with a framework that triggers alerts when a significant drift is detected can be helpful. In such a case a manual time period need not be invoked. Hence a full fledged framework of adaptive learning based data monitoring system could be achieved for subject matter expert triaging.

**References**

1. Bauer, Markus, Clemens van Dinther, and Daniel Kiefer. "Machine learning in SME: an empirical study on enablers and success factors." (2020).
2. Williams, Saundra Wall. "The effectiveness of subject matter experts as technical trainers." Human Resource Development Quarterly 12.1 (2001): 91.
3. Schmidt, Henk G., et al. "Influence of tutors' subject-matter expertise on student effort and achievement in problem-based learning." Academic medicine 68.10 (1993): 784-91.
4. Bejar, Isaac I. "Subject matter experts' assessment of item statistics." Applied Psychological Measurement 7.3 (1983): 303-310.
5. Dang, Jr-Fong. "The deep learning-based equipment health monitoring model adopting subject matter expert." International Journal of Computer Integrated Manufacturing (2023): 1-16.
6. Robinson, S. D. "Temporal topic modeling applied to aviation safety reports: A subject matter expert review." Safety science 116 (2019): 275-286.
7. Karadakis, Kostas, and Maria Manus Painchaud. "Esport Knowledge, Skills, and Abilities: Perspectives from Subject Matter Experts." Athens Journal of Sports 9.2 (2022): 79-100.
8. Teferra, Kirubel, et al. "Mapping model validation metrics to subject matter expert scores for model adequacy assessment." Reliability Engineering & System Safety 132 (2014): 9-19.
9. Christianto, Hotman, et al. "Smart interpretable model (SIM) enabling subject matter experts in rule generation." Expert Systems with Applications 207 (2022): 117945.
10. Tecuci, Gheorghe, et al. "An experiment in agent teaching by subject matter experts." International Journal of Human-Computer Studies 53.4 (2000): 583-610.
11. White, Jules, et al. "A prompt pattern catalog to enhance prompt engineering with chatgpt." arXiv preprint arXiv:2302.11382 (2023).
12. Hugging face [online]. <https://huggingface.co/sentence-transformers/distilbert-base-nli-mean-tokens> [Accessed: 15/10/2023]
13. Hugging face [online]. <https://huggingface.co/sentence-transformers/all-mpnet-base-v2>. [Accessed: 15/10/2023]
14. Hugging face [online]. <https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1>. [Accessed: 15/10/2023]
15. Hugging face [online]. <https://huggingface.co/distilbert-base-uncased>. [Accessed: 15/10/2023]
16. Hugging face [online]. <https://huggingface.co/bert-base-uncased>. [Accessed: 15/10/2023]
17. Hugging face [online]. <https://huggingface.co/sentence-transformers/all-distilroberta-v1>. [Accessed: 15/10/2023]
18. Hugging face [online]. <https://huggingface.co/roberta-base>. [Accessed: 15/10/2023]
19. TensorFlow [online]. <https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder>. [Accessed: 15/10/2023]
20. Langchain [online]. <https://python.langchain.com/docs/get_started/introduction>. [Accessed: 06/01/2024]

# 

| Model | Classification Report |
| --- | --- |
| [12] |  |
| [13] |  |
| [14] |  |
| [15] |  |
| [16] |  |
| [17] |  |
| [18] |  |
| [19] |  |