

Contents

[Introduction 1](#_heading=h.iwhnlia2udx9)

[Business Understanding 2](#_heading=h.a9ct9i2s5jxw)

[Business Overview and Objective 2](#_heading=h.qo4nyijbs60q)

[Ethical Considerations 3](#_heading=h.rz3t1342gtz6)

[Project Planning 3](#_heading=h.g1w32vx7f3ci)

[Project Scope 4](#_heading=h.ilmrq061ej1i)

[Data Understanding and Exploration 6](#_heading=h.1wrotxgnr9pe)

[Data Preparation 10](#_heading=h.ghddo9dchlqy)

[Data Cleaning and Feature Selection 10](#_heading=h.mju7tvg1fl8a)

[Text Preprocessing 11](#_heading=h.gob9f75jk9tg)

[Feature Engineering 12](#_heading=h.3mvdcr9g6bxa)

[Dataset Balancing 12](#_heading=h.a4uz4t8r96cz)

[Machine Learning Implementation 15](#_heading=h.thyga22tfytf)

[Technologies and Tools Used 15](#_heading=h.9w6g3c4wpd82)

[Queue and Priority Classification 17](#_heading=h.tjs7685zodwg)

[Model Implementations 17](#_heading=h.nxckz65unbeu)

[Hyperparameter Tuning 18](#_heading=h.s00emvg23i08)

[Model Speed and Accuracy Evaluation 19](#_heading=h.hw3nugx0lb4s)

[Cross-Validation Results and Metric Comparison 20](#_heading=h.h2yxut595bpa)

[Confusion Matrix Analysis 22](#_heading=h.h9er7qu4nsqz)

[Learning Curve Analysis 25](#_heading=h.wd653dc6xiap)

[Text Summarization 28](#_heading=h.2urusx1wxsl0)

[Text Preprocessing for Summarization 28](#_heading=h.2tnufbczwwby)

[Extractive Summarization 28](#_heading=h.bjefngozu1u6)

[Abstractive Summarization 28](#_heading=h.rz3llm99c434)

[Challenges Encountered & Solutions 29](#_heading=h.njhct7qfn7mh)

[Conclusion 30](#_heading=h.pqpgvrqmzu0f)

[References 31](#_heading=h.v4ym4uc1nj9b)

[Bibliography 33](#_heading=h.lu8o4ql5g16q)

# **Introduction**

One of the top priorities in CX management research in 2024, as reported by 234 surveyed CX practitioners, is improving both product and customer experience within their organizations. (Scutt and Quaadgras, 2024). For my capstone project, I will use the Customer Support Emails - Ticket System - Helpdesk dataset from Kaggle (Bueck, 2024) to implement a machine learning model aimed at enhancing the customer support experience. I will be achieving this action by implementing the following functionality below:

* Classifying customer support tickets based on its queue category.
* Predicting the prioritization of a ticket from low, medium or high.
* Summarising a customer support ticket to get an overview of the issue.

A common source of customer frustration is the delay in resolving product-related issues. Such delays often stem from agents having missing knowledge or tickets being misrouted multiple times to different teams (Truss and Boehm, 2024). These inefficiencies can significantly prolong issue resolution which can negatively influence the customer experience with the product and their engagement with support services.

According to a research study, automation can play a significant role in accelerating the resolution of customer issues, thereby enhancing customer satisfaction (A Collaborative Research Study, 2021). There are opportunities to implement machine learning models to advance automation by enabling efficient ticket classification for accurate routing, prioritizing tickets based on urgency, and summarizing ticket content to identify the core issue experienced by the customer.

Implementing machine learning to enhance automation in customer support ticketing can improve the consistency of ticket handling and reduce operational errors that could occur from a customer support agent.

# 

# 

# 

# 

# 

# 

# **Business Understanding**

In today’s digital landscape, businesses are increasingly adopting artificial intelligence to enhance operational workflows. This capstone project focuses on applying AI specifically to optimize customer service operations.

## **Business Overview and Objective**

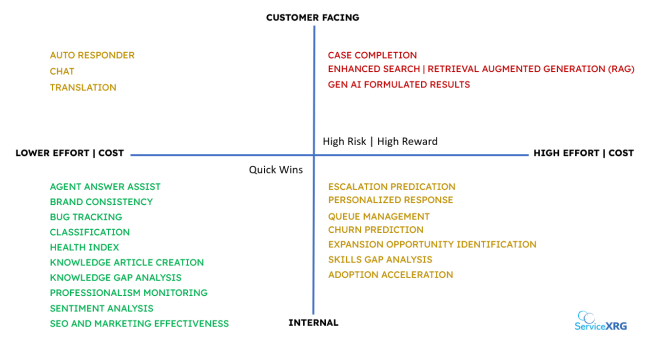
There is research demonstrating the use of artificial intelligence–enabled tools to leverage unstructured digital information in order to improve customer experience and drive business outcomes (Service Excellence Research Group, LLC, 2025). This research defines and presents 20 possible use cases for AI for customer service support.  


Figure 1: 2025 AI for Support: Use Cases, Risks and Quick Wins  
Source: Service Excellence Research Group, LLC. (2025)  
  
These AI-enabled tools can be applied internally for customer support agents or externally for direct customer interactions. My objective is to explore practical application of AI in customer service support, with the specific goal of enhancing customer experience by reducing the resolution time and improving overall service efficiency. I will be developing the following machine learning functionalities to assist internal support agents to resolve tickets more effectively:

* **Queue Management –** Implementation of machine learning to classify tickets and route them to the appropriate team based on the content of the case.
* **Ticket Prioritization –** Implementation of machine learning to predict the urgency of a ticket as low, medium, or high.
* **Ticket Summarization –** Generating a summary of a customer support ticket to provide an overview and capture the core issue reported by the customer.

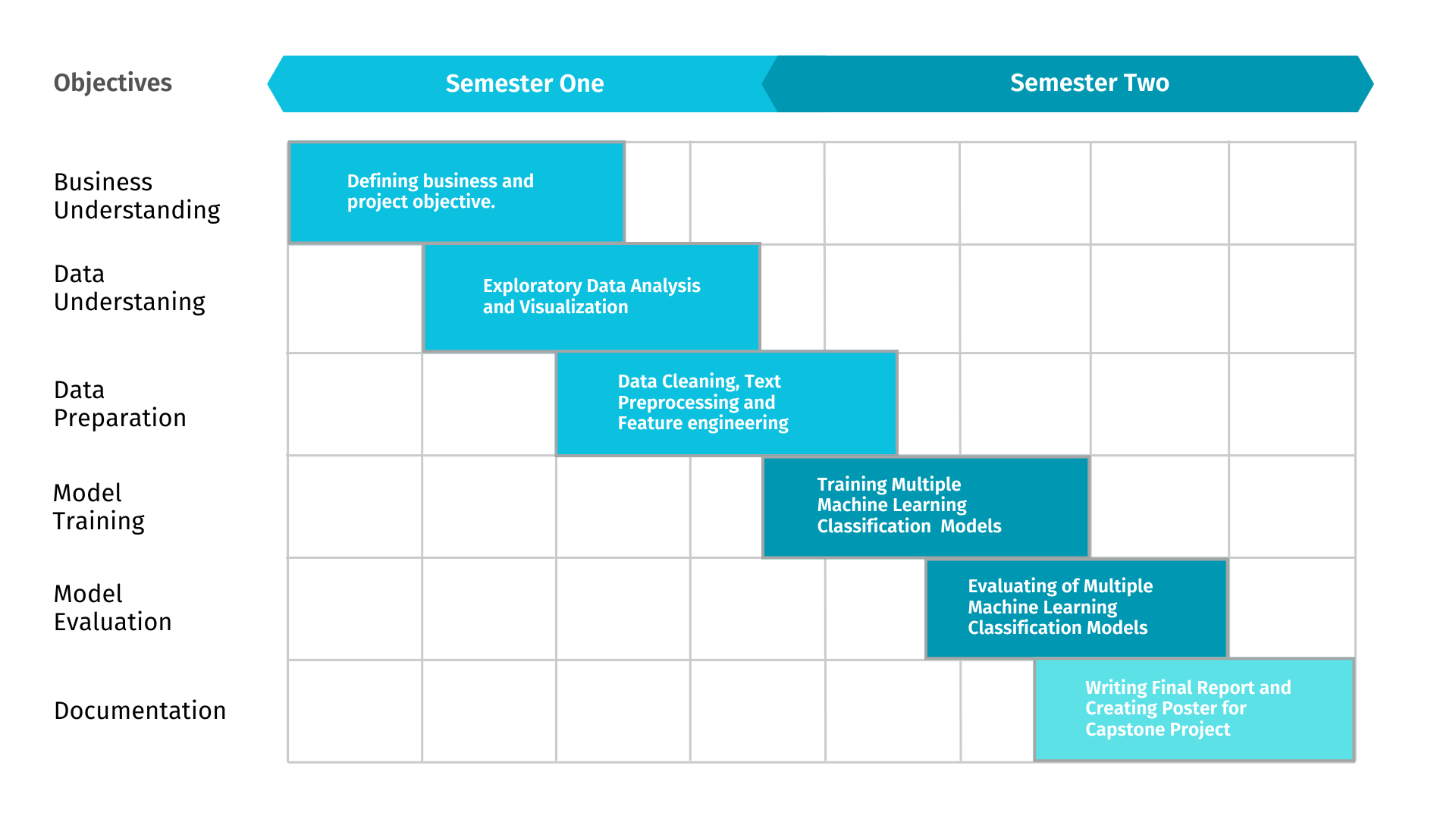
These functionalities can enhance operational efficiency by classifying tickets based on their queue and priority, ensuring they are routed to the appropriate teams and addressed by its urgency. Additionally, ticket summarization provides agents with an overview of the case, helping them quickly understand the customer issue. These capabilities enable customer support agents to work more effectively, reduce resolution times, and minimize customer churn.

## **Ethical Considerations**

According to the Ethics Guidelines for Trustworthy AI, trustworthy AI encompasses three key components: compliance with the law, adherence to ethical principles, and robustness from both technical and social perspectives (European Commission, 2019).

As the capstone project focuses on customer support tickets, it is essential to ensure compliance with GDPR standards when using the dataset for the machine learning model. This includes removing all personally identifiable information (PII) information and using necessary data for the model’s intended functionality. In addition,it will be important that all potential biases are considered. The classification of the ticket queue and priority must be based solely on the ticket issue, rather than on any demographic information about the customer to prevent discrimination. The machine learning model’s performance should also be continuously evaluated to ensure it functions as expected and improves positively both to the customer and support agent experience.

## **Project Planning**

  
Below I have provided a high-level timeline for the capstone project, which spans two semesters. The project is divided into six main objectives: business understanding, data understanding, data preparation, model training, model evaluation, and documentation.

# 

# 

# 

# 

*Figure 2: Project timeline*

## **Project Scope**

In this capstone project, I will provide a Jupyter notebook demonstrating my machine learning implementation, a report paper detailing all aspects of the project, and a poster presentation summarizing the key results. Below, I have included further information on the project's functionality and scope.

**Project Functionality**

|  |  |
| --- | --- |
| Data input | Retrieving customer support ticket information for analysis. |
| **Machine learning model** | Applying natural language processing (NLP) techniques and machine learning models to classify tickets by queue and priority, and to generate summaries of customer issues. |
| **Data output** | The implementation will produce the following outputs:   * A classified ticket queue * A predicted priority level for each ticket * A summarized description of the customer issue |

**Project Boundary - In Scope**

|  |  |
| --- | --- |
| **1.** | **Data Understanding and Exploration -** Conducting an overview and exploratory data analysis (EDA) of the customer support ticket dataset. |
| **2.** | **Data Preparation and Feature Engineering -** Preprocessing text data and applying feature engineering (tokenization, encoding, TF-IDF). Addressing class imbalance using techniques like SMOTE. |
| **3.** | **Queue Classification -** Implementing machine learning models to classify support tickets into predefined queue categories. |
| **4.** | **Priority Classification -** Implementing machine learning models to predict ticket priority (Low, Medium, High). |
| **5.** | **Hyperparameter Optimization -** Performing hyperparameter tuning and cross-validation to improve model performance. |
| **6.** | **Text Summarization -** Implement transformer-based models for ticket summarization:   * Extractive Summarization using BERT * Abstractive Summarization using BART |
| **7.** | **Evaluation -** Assess and compare model performance for both classification and summarization tasks to identify the most effective approaches. |

**Project Boundary - Out of Scope**

|  |  |
| --- | --- |
| **1.** | **System Integration** - Integrating the machine learning models into an existing customer support system. |
| **2.** | **Real-Time Processing -** Processing support tickets in real-time such as handling live incoming tickets continuously |
| **3.** | **Database Management -** Leveraging a database to store and retrieve customer support ticket data for use in a machine learning model. |
| **4.** | **Automated Ticket Assignment -** Developing an end-to-end workflow to automatically assign tickets to specific teams and defining the ticket priority |
| **5.** | **Automated Customer Responses** - Generating and sending automated replies to customers based on ticket content. |
| **6.** | **User Feedback Incorporation -** Collecting and incorporating user feedback (from support agents or customers) into model retraining or evaluation loops. |
| **7.** | **Dashboard Development** - Building a user interface or dashboard for visualizing model outputs or conducting analysis. |

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Data Understanding and Exploration

I conducted an exploratory data analysis (EDA) on the Customer Support Emails - Ticket System - Helpdesk dataset from kaggle (Bueck, 2024). The dataset contains 4,000 rows and 17 columns. The columns of the dataset is the following below:

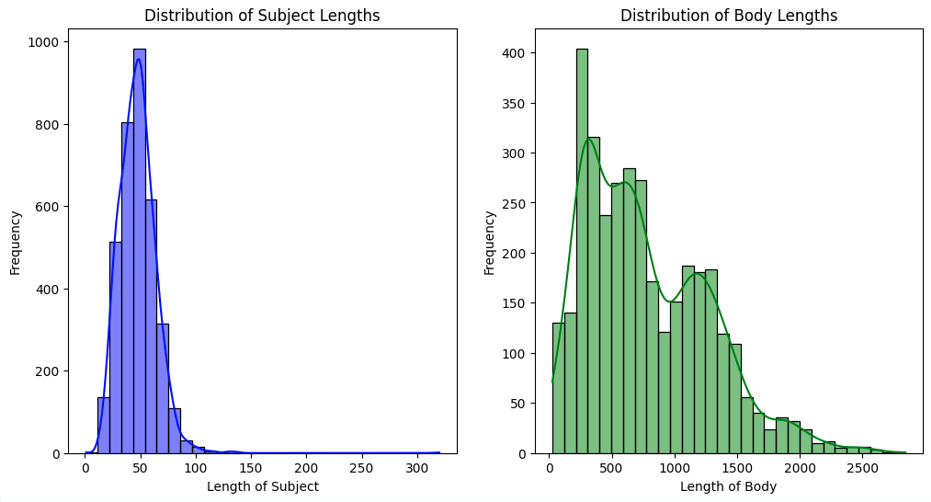
|  |  |
| --- | --- |
| **Columns** | 'subject', 'body', 'answer', 'type', 'queue', 'priority', 'language', 'business\_type', 'tag\_1', 'tag\_2', 'tag\_3', 'tag\_4', 'tag\_5', 'tag\_6','tag\_7', 'tag\_8', 'tag\_9' |

Upon inspection, several missing values were found in the subject, body, tag\_4, tag\_5, tag\_6, tag\_7, tag\_8, and tag\_9 columns. To ensure consistency, I removed rows containing the missing values. Due to a high volume of missing data, I dropped the columns tag\_5, tag\_6, tag\_7, tag\_8 and tag\_9 columns. To simplify the tag representation, tag\_1, tag\_2, tag\_3, and tag\_4 columns were merged into a single tags column. There were no duplicate rows or outliers found during the analysis.

For the scope of this capstone project, which focuses on classifying ticket queues, predicting ticket priority, and summarizing ticket issues, I selected the following relevant columns below:

|  |  |
| --- | --- |
| **Columns** | 'subject', 'body', 'queue', 'priority' ,'tags’ |

I analyzed the distribution of subject and body lengths in the customer support tickets to gain insights into how customers communicate their issues. The subject lines were relatively consistent, with most falling between 40 and 70 characters and peaking around 50 characters, indicating that customers tend to keep subjects concise. In contrast, body lengths showed greater variability, ranging from just a few characters to over 2,500. There is a high concentration between 300 and 800 characters and peaking around 300 characters. This suggests that while many customers provide brief descriptions of their issues, some provide more detailed information on their cases.

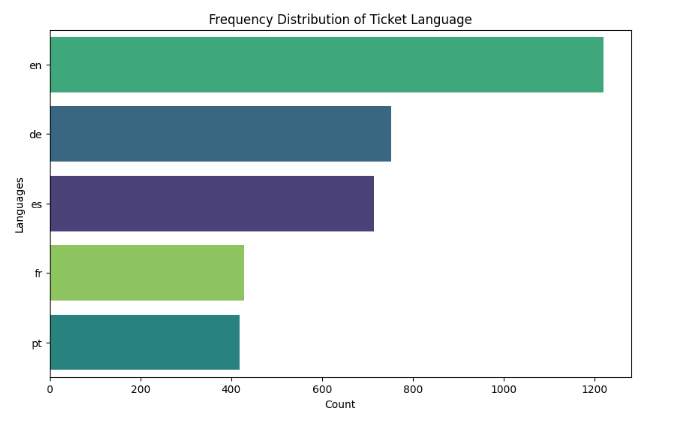


# 

# 

# 

*Figure 3 : Distribution of Subject and Body Lengths*

By analyzing the languages used in customer support tickets, five main languages were identified: English (en), German (de), Spanish (es), French (fr), and Portuguese (pt). Most tickets are in English, with over 1,200—nearly double the count of the next most common language. German and Spanish follow with around 730 tickets each, while French and Portuguese have just over 400. This suggests a strong English-speaking user base, while also highlighting support for other European languages.

# 

# 

# 

# 

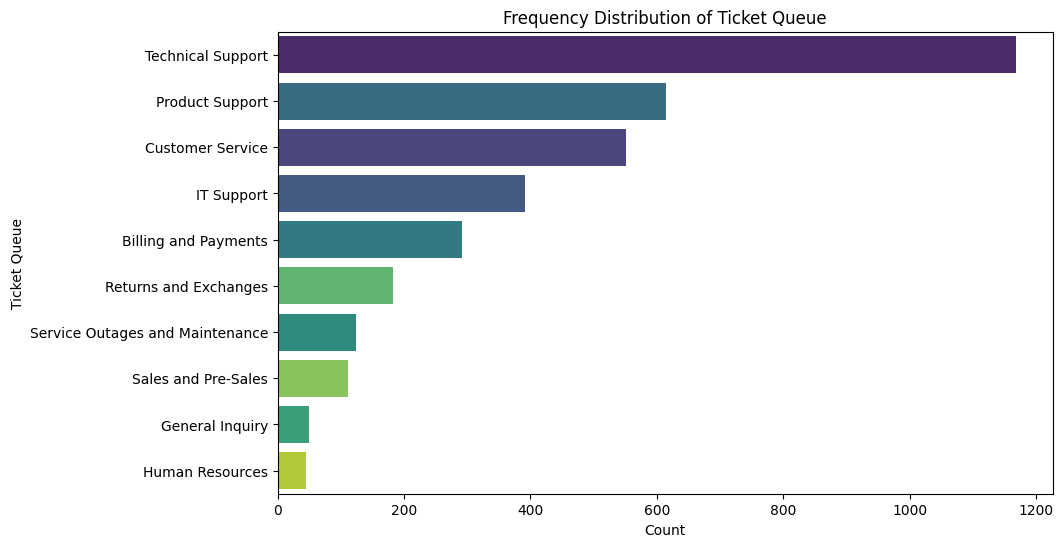
# 

*Figure 4 : Frequency Distribution of Ticket Languages*

Upon reviewing the ticket priority distribution, high and medium priority tickets occur significantly more often than low priority ones. This indicates a class imbalance, which could negatively impact the performance of classification models. Resampling or reweighting methods will need to take place to ensure fair training across the ticket priority.

# 

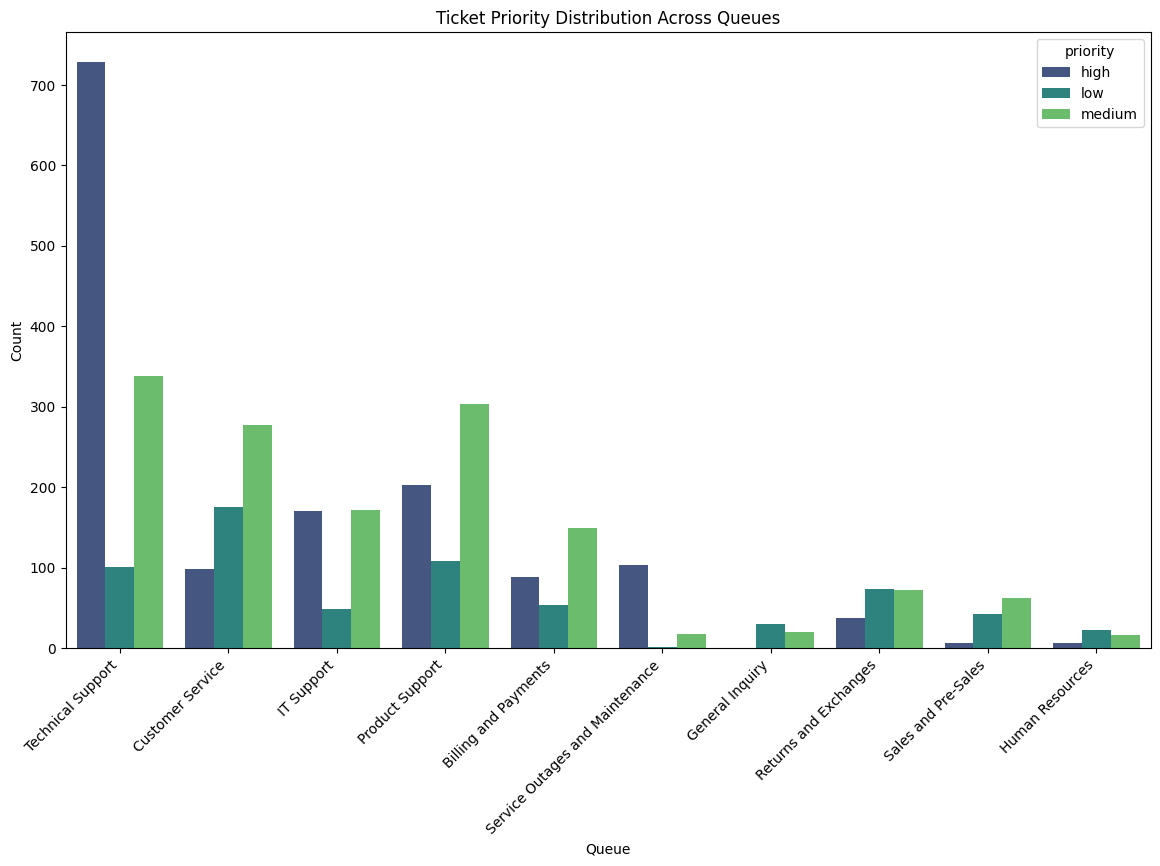
*Figure 5 : Frequency Distribution of Ticket Priority*

The dataset covers ticket queues like Technical Support, Product Support, Customer Service, IT Support, Billing, Returns, Outages, Sales, General Inquiry, and HR. The majority of tickets fall within the top five queues, with Technical Support queue receiving the highest number of cases. There is also a class imbalance which will require resampling or reweighting to improve the performance for the ticket queue classification model. When comparing the ticket priority across queues, the Technical Support queue receives the highest number of high priority tickets, whereas most other queues, such as Customer Service and Product Support, are dominated by medium and low-priority tickets, indicating less critical inquiries.

# 

# 

*Figure 6 : Frequency Distribution of Ticket Queue*



# 

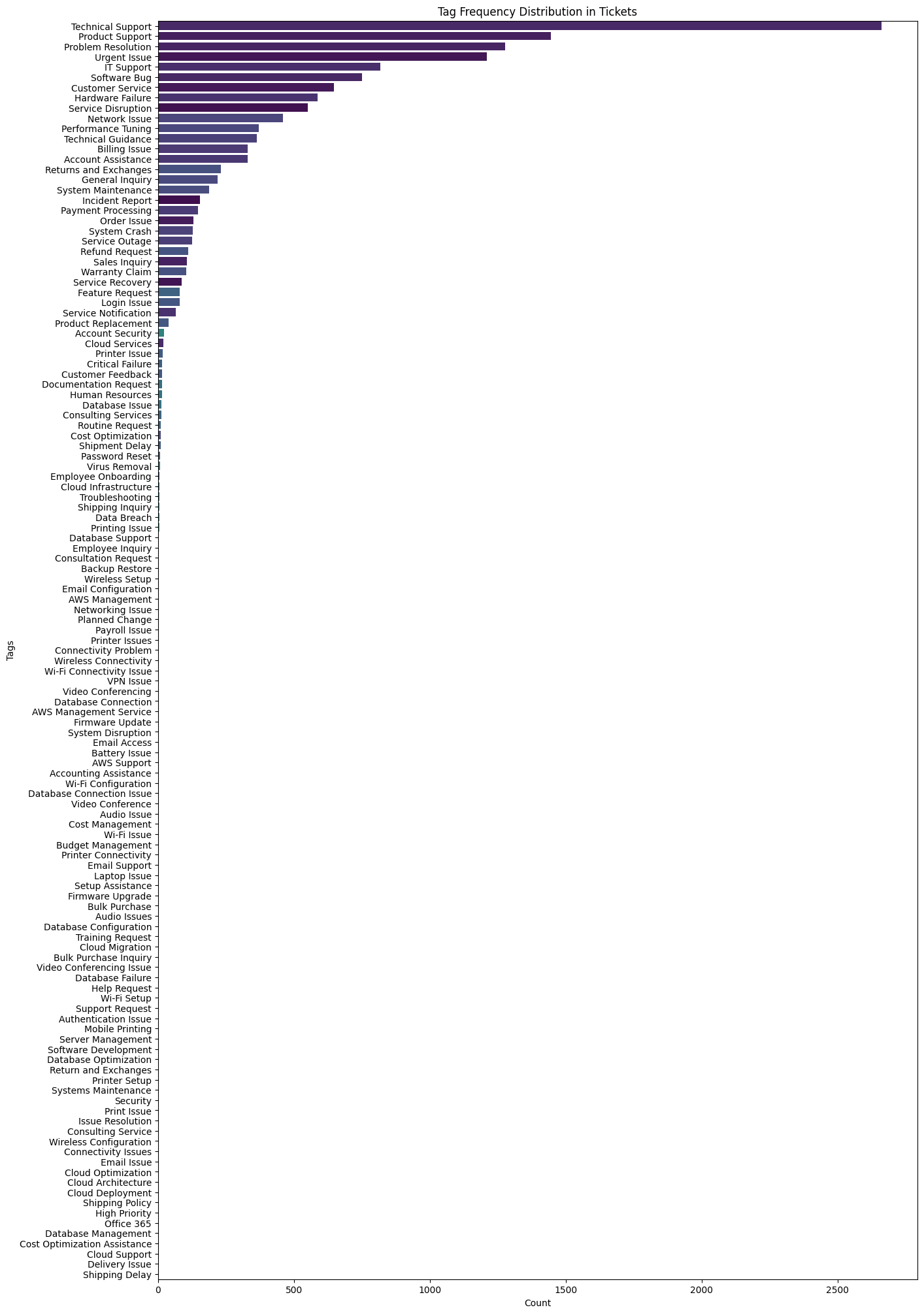
# 

# 

# 

*Figure 7 : Ticket Priority Distribution Across Queues*

From reviewing the distribution for the tags in the customer support tickets, it contains 122 unique tags, with the top five being Technical Support, Product Support, Problem Resolution, Urgent Issue, and IT Support. The chart below shows the overall distribution of tags within the dataset.



# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

*Figure 8 : Tag Frequency Distribution in Tickets*

# **Data Preparation**

Effective data preparation is crucial for ensuring the reliability and performance of machine learning models. It involves transforming raw data into a structured format suitable for machine learning algorithms. (Brownlee, 2020). This section outlines the data preparation steps undertaken to implement both classification and summarization tasks. The process includes data cleaning, feature selection, text preprocessing, feature engineering, and dataset balancing.

## **Data Cleaning and Feature Selection**

To ensure data quality and consistency for both classification and summarization tasks, the following steps were taken during the data cleaning and feature selection process:

**Handling Missing and Incomplete Data:**

All rows with missing values were removed from the dataset. Additionally, entries with empty strings in the subject or body fields were excluded. These steps ensured that only complete rows were retained for modeling, particularly for text-based natural language processing tasks.

**Feature Selection for the Capstone Project**

For the scope of this capstone project, I have selected only the most relevant columns to support ticket queue classification, ticket priority classification and text summarization. The selected features are the following below:

|  |  |
| --- | --- |
| **subject** | The subject line of the support ticket. |
| **body** | The main content of the support ticket. |
| **queue** | The ticket category assigned to a specific team. |
| **priority** | The urgency level of the ticket |
| **tags** | Additional metadata associated with the ticket |

**Queue Selection and Filtering:**

The queue column was filtered to include only the top five categories: **Technical Support, Product Support, Customer Service, IT Support**, and **Billing and Payments**. These queues were selected because they represent the majority of support tickets in the dataset. Less frequent queue types were excluded to reduce noise, address class imbalance, and maintain a clear project focus.

**Language Standardization:**

The text in the subject and body columns was translated into English to standardize the language used throughout the dataset. This step was essential for ensuring consistency across all text entries and for applying natural language processing techniques effectively. The presence of multiple languages could also introduce inconsistencies during the text preprocessing phase.

## **Text Preprocessing**

Applying text preprocessing techniques to textual data plays a crucial role in the performance of text classification models. As noted by Uysal and Günal (2014), the quality and type of preprocessing applied can significantly influence the accuracy and efficiency of machine learning algorithms. I applied the following text preprocessing techniques to the subject, body, and tags columns.

**Text Normalization**

I have applied text normalization to make the text data consistent. Text normalization is a process in which different variations of text are converted into a standard form (Ghosh and Gunning, 2019). I have applied the following text normalization techniques below:

|  |  |
| --- | --- |
| **Lowercasing** | All text was converted to lowercase to standardize word representation and ensure consistency. |
| **Whitespace Trimming** | Leading and trailing whitespace characters were removed to eliminate formatting inconsistencies. |
| **Contraction Expansion** | Common contractions were expanded to their full forms to enhance clarity and improve tokenization matching. |
| **Spell Contractions** | Applying spell contractions before text classification improves feature quality by reducing noise and ensuring words are written correctly. |
| **URL and Email Removal** | URLs and email addresses were removed from the text, as they do not provide meaningful semantic information. |

**Tokenization**

While working with natural language processing, tokenization is considered a crucial step that involves splitting the text into smaller parts called tokens. (Chopra, Joshi and Mathur, 2016) It breaks down raw text into smaller, manageable units for easier analysis and processing. I have applied tokenization to the subject, body, and tags columns.

**Stopword and Punctuation Removal**

Another action I have taken is removing stop words and punctuation from the text data, as they don’t contribute much to the overall meaning of the sentence (Ghosh and Gunning, 2019). These words have limited semantic value for text classification. This preprocessing step was applied to the tokenization of the subject, body, and tags.

**Stemming and Lemmatization**

To reduce the vocabulary size of the textual data, stemming and lemmatization can be applied, which are popular natural language processing techniques (Kasliwal, 2018). Stemming was applied to the subject and tags fields to reduce words to their root forms, while lemmatization was used for the body field to convert words to their dictionary base forms.I applied lemmatization to the body field to preserve semantic accuracy, as opposed to stemming.

## **Feature Engineering**

When working with text data the format is mostly unstructured. Therefore, it is essential to convert it into numeric features because most machine learning algorithms are capable of dealing only with numbers (Ghosh and Gunning, 2019). Since the capstone project primarily involves text data, I have converted the text data to numeric values by taken the following steps:

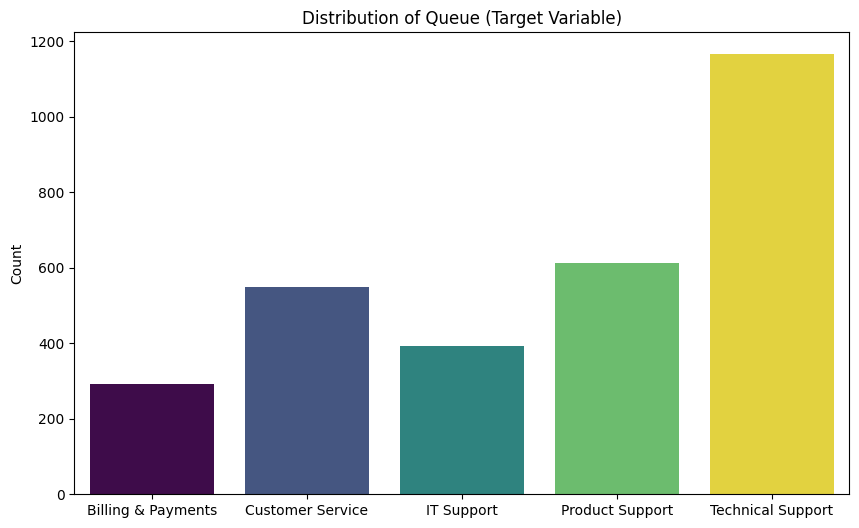
|  |  |
| --- | --- |
| **Label Encoding** | The queue column was label-encoded, assigning a unique integer to each queue type based on alphabetical order, as the categories have no inherent ranking. |
| **Ordinal Encoding** | The priority column was encoded using a custom ordinal mapping: {'low': 0, 'medium': 1, 'high': 2}. This approach preserves the natural order of the priority levels. |
| **Bag-of-Words (BoW)** | The tags column was vectorized using the Bag-of-Words model. This model uses word frequencies to construct a vector, providing the count of each word in the text (Srinivasa-Desikan, 2018). I chose this approach because the tags do not carry any semantic meaning. |
| **Term Frequency-Inverse Document Frequency (TF-IDF)** | The subject and body columns were vectorized using TF-IDF. It calculates how frequently a word appears and adjusts for how common the word is across all text, which helps determine its importance (Srinivasa-Desikan, 2018). I chose this approach because TF-IDF can highlight distinctive words in the subject and body columns. |

## **Dataset Balancing**

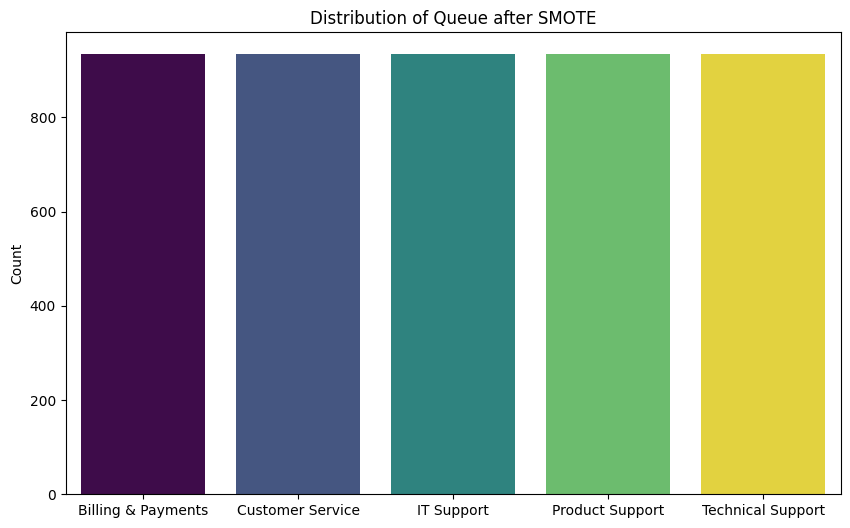
From conducting exploratory data analysis, I found that there is a class imbalance in the queue and priority columns, which can affect the performance of the multi-class classification task. An issue that can arise from multi-class classification with an imbalanced dataset is that there may be too few examples in the minority classes, leading to poor model performance (Brownlee, 2021). To address this issue, I have applied the synthetic minority oversampling technique to balance the dataset for the queue and priority target variables.

On the following page, I provided visualizations of the distribution of the queue and priority target variables, both before and after applying SMOTE.

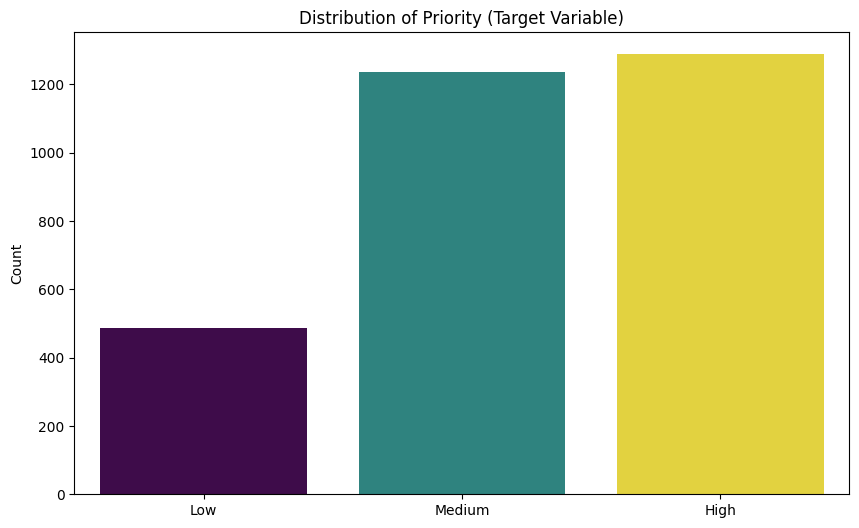
# 



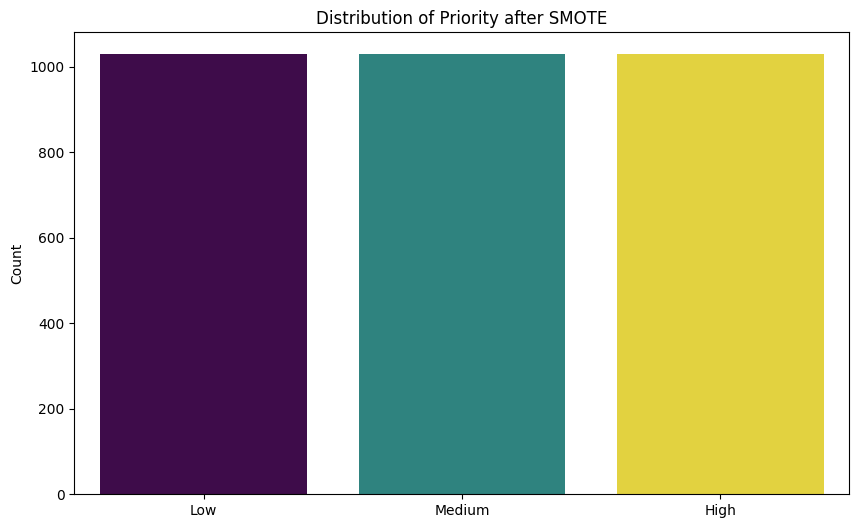
*Figure 9 : Distribution of Queue (Target Variable)*



*Figure 10 : Distribution of Queue after SMOTE*



*Figure 11 : Distribution of Priority (Target Variable)*



*Figure 12 : Distribution of Priority after SMOTE*

# 

# **Machine Learning Implementation**

This section outlines the machine learning implementation for classifying support tickets by queue and priority level, and summarizing ticket content. The objective is to develop accurate and efficient models capable of categorizing customer support tickets by queue and priority level, while also providing text summarization. In the following sections, I will provide detailed information on each stage of the model development and evaluation process.

## **Technologies and Tools Used**

This capstone project leverages a diverse set of Python libraries and tools to handle various tasks, including data understanding, data preprocessing, visualization, natural language processing, and machine learning. Below is a categorized breakdown of the main libraries and frameworks used, along with their core purposes in the project.

**Data Handling & Visualization**

|  |  |
| --- | --- |
| **pandas (pd)** | For structured data manipulation and analysis. |
| **numpy (np)** | For numerical computations and array operations. |
| **seaborn (sns)** | For high level statistical data visualization. |
| **matplotlib.pyplot (plt)** | Core plotting library for basic charts and graphs. |

**Text Processing & Natural Language Processing (NLP)**

|  |  |
| --- | --- |
| **re** | Regular expressions for pattern matching and text cleaning. |
| **BeautifulSoup** | Removing HTML tags in the text |
| **contractions** | Expands contracted words to their full forms. |
| **langdetect** | Detects the language of given text snippets. |
| **TextBlob** | To perform basic spell check |
| **nltk** | **- Tokenization (word\_tokenize, sent\_tokenize) -** Splits text into individual words and sentences for easier analysis.  **- Stemming (PorterStemmer) -** Simplifies words to their root form.  **- Lemmatization (WordNetLemmatizer)** - Converts words to their base or dictionary form.  **- Stopwords Removal (stopwords)** - Provides a list of common words to remove from text during preprocessing. |

**Transformers & Summarization**

|  |  |
| --- | --- |
| **summarizer.Summarizer** | Uses BERT for extractive text summarization. |
| **transformers.pipeline** | Used to load a BART-based model (facebook/bart-large-cnn) for abstractive summarization |

**Feature Extraction & Encoding**

|  |  |
| --- | --- |
| **LabelEncoder** | Converts categorical labels) into numerical format for encoding target variables in classification tasks. |
| **CountVectorizer** | Implements the Bag-of-Words (BoW) model by converting text into a matrix of token counts. |
| **TfidfVectorizer** | Converts text into numerical features using TF-IDF (Term Frequency–Inverse Document Frequency). |

**Data Preprocessing & Class Balancing**

|  |  |
| --- | --- |
| **StandardScaler** | Standardizes features by removing the mean and scaling to unit variance. |
| **SMOTE (from imblearn)** | Balances class distribution using synthetic oversampling. |

**Machine Learning Models & Evaluation**

|  |  |
| --- | --- |
| **Classification Algorithms** | DecisionTreeClassifier, KNeighborsClassifier, LinearSVC, RandomForestClassifier, and GaussianNB |
| **Model Training & Validation:** | StratifiedKFold, train\_test\_split, cross\_val\_score, learning\_curve, and, RandomizedSearchCV. |
| **Evaluation Metrics:** | accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report, and, make\_scorer |

**Utilities**

|  |  |
| --- | --- |
| **math** | For mathematical operations. |
| **time** | Used for timing and performance measurement. |

## **Queue and Priority Classification**

I will be implementing five different machine learning models and evaluating their performance in classifying the ticket queue and priority level. Each model will undergo hyperparameter tuning, cross-validation, performance metric analysis, and learning curve analysis. The final step will be selecting the most suitable model for each task.

### **Model Implementations**

This capstone project uses five classification algorithms: Decision Tree, K-Nearest Neighbors (KNN), LinearSVC (a linear Support Vector Machine), Naive Bayes, and Random Forest. The model selection was based on a comparative study on multi-class text classification for categorizing customer issues (Biju et al., 2018). The study found that Support Vector Machine (SVM) outperformed the other classifiers, achieving the highest accuracy on their dataset.

I applied and evaluated the same five classifiers, using LinearSVC to represent the SVM approach, and assessed their performance in classifying customer support tickets by queue and priority level. Below I have provided further information on the overview of the model classifiers:

|  |  |
| --- | --- |
| **Decision Tree** | A supervised machine learning method that splits data step-by-step using specific features to make predictions. For text classification, it uses terms as internal nodes and follows decision paths to determine the appropriate category. (Biju et al., 2018). |
| **K-Nearest Neighbors** | K-Nearest Neighbors is a simple, non-parametric classification algorithm that uses similarity or distance measures such as cosine similarity or Euclidean distance to classify data points based on their neighbors. (Biju et al., 2018). |
| **LinearSVC** | LinearSVC is a type of Support Vector Machine (SVM) used for classification tasks. It tries to find the best straight-line (or hyperplane) to separate different classes by maximizing the margin between them. |
| **Naive Bayes** | A simple and effective classifier that uses Bayes' Theorem. It's commonly used for tasks like spam detection, organizing emails, document classification, sentiment analysis, and language recognition (Biju et al., 2018). |
| **Random Forest** | Random Forest is a supervised learning method that builds multiple decision trees using random samples of data and averages their results for classification. |

## 

### 

### **Hyperparameter Tuning**

Hyperparameter tuning is a critical component in optimizing the performance and generalization of machine learning models (Ilemobayo et al., 2024). It involves identifying the optimal set of hyperparameters to achieve the best possible model performance. Each machine learning algorithm has its own unique set of hyperparameters, which can be adjusted based on the dataset and the specific task. I have applied commonly used hyperparameters appropriate to the machine learning algorithms listed below:

|  |  |
| --- | --- |
|  | Hyperparameters |
| Decision Tree | 'max\_depth', 'min\_samples\_split', 'min\_samples\_leaf' ,'criterion' |
| K-Nearest Neighbors | 'n\_neighbors', 'weights', 'metric' |
| LinearSVC | 'C', 'tol', 'max\_iter' |
| Naive Bayes | 'var\_smoothing' |
| Random Forest | 'n\_estimators’, 'max\_depth', 'min\_samples\_split', 'min\_samples\_leaf, 'max\_features' |

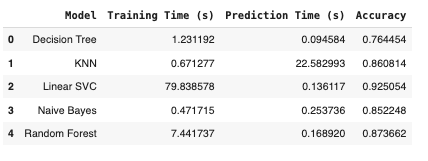
When tuning hyperparameters in machine learning models, two common techniques are GridSearchCV and RandomizedSearchCV. GridSearchCV is used to identify the optimal set of hyperparameters by going through all possible hyperparameter combinations (Agrawal, 2020). While this approach can find the most optimal set of hyperparameter values, it is often computationally expensive and time-consuming. An alternative approach is to implement RandomizedSearchCV, which randomly samples parameter combinations from specified distributions of hyperparameter values. It typically performs as well as grid search but is significantly more cost and time-efficient (Raschka and Mirjalili, 2020).

For this capstone project, I chose to implement RandomizedSearchCV due to its speed and efficiency in hyperparameter tuning. Using this method, I optimized the hyperparameters of the machine learning model to classify the queue and priority level of customer tickets. The table below summarizes the optimal hyperparameter values that I have identified using RandomizedSearchCV.

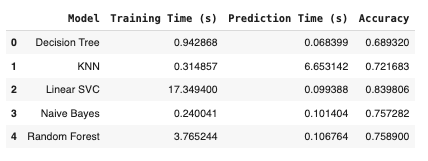
|  |  |
| --- | --- |
|  | **Ticket Queue Classification - Optimal Hyperparameters Values** |
| **Decision Tree** | {'min\_samples\_split': 10, 'min\_samples\_leaf': 2, 'max\_depth': 30, 'criterion': 'gini'} |
| **K-Nearest Neighbors** | {'weights': 'distance', 'n\_neighbors': 5, 'metric': 'manhattan'} |
| **LinearSVC** | {'tol': 0.001, 'max\_iter': 2000, 'C': 0.01} |
| **Naive Bayes** | {'var\_smoothing': np.float64(7.543120063354607e-07)} |
| **Random Forest** | {'n\_estimators': 300, 'min\_samples\_split': 15, 'min\_samples\_leaf': 5, 'max\_features': 'sqrt', 'max\_depth': 50} |

|  |  |
| --- | --- |
|  | **Ticket Priority Classification - Optimal Hyperparameters Values** |
| **Decision Tree** | {'min\_samples\_split': 10, 'min\_samples\_leaf': 2, 'max\_depth': 30, 'criterion': 'gini'} |
| **K-Nearest Neighbors** | {'weights': 'distance', 'n\_neighbors': 5, 'metric': 'manhattan'} |
| **LinearSVC** | {'tol': 0.001, 'max\_iter': 2000, 'C': 0.01} |
| **Naive Bayes** | {'var\_smoothing': np.float64(7.543120063354607e-07)} |
| **Random Forest** | {'n\_estimators': 300, 'min\_samples\_split': 15, 'min\_samples\_leaf': 5, 'max\_features': 'sqrt', 'max\_depth': 50} |

### **Model Speed and Accuracy Evaluation**

After identifying the optimal hyperparameter values for the models based on the ticket queue and priority classification task, I evaluated the efficiency of each model. I evaluated its training and prediction time alongside its classification accuracy.   
  
**Model Speed and Accuracy for Ticket Queue Classification  
**

**Model Speed and Accuracy for Ticket Priority Classification**



Across both speed and accuracy evaluation, Linear SVC achieved the highest accuracy. Naive Bayes consistently had the fastest training time, while Decision Tree delivered the quickest prediction time. For practical applications balancing speed and accuracy, Naive Bayes stands out as a top choice. However Linear SVC is the best option when accuracy is the top priority and training time is less critical.

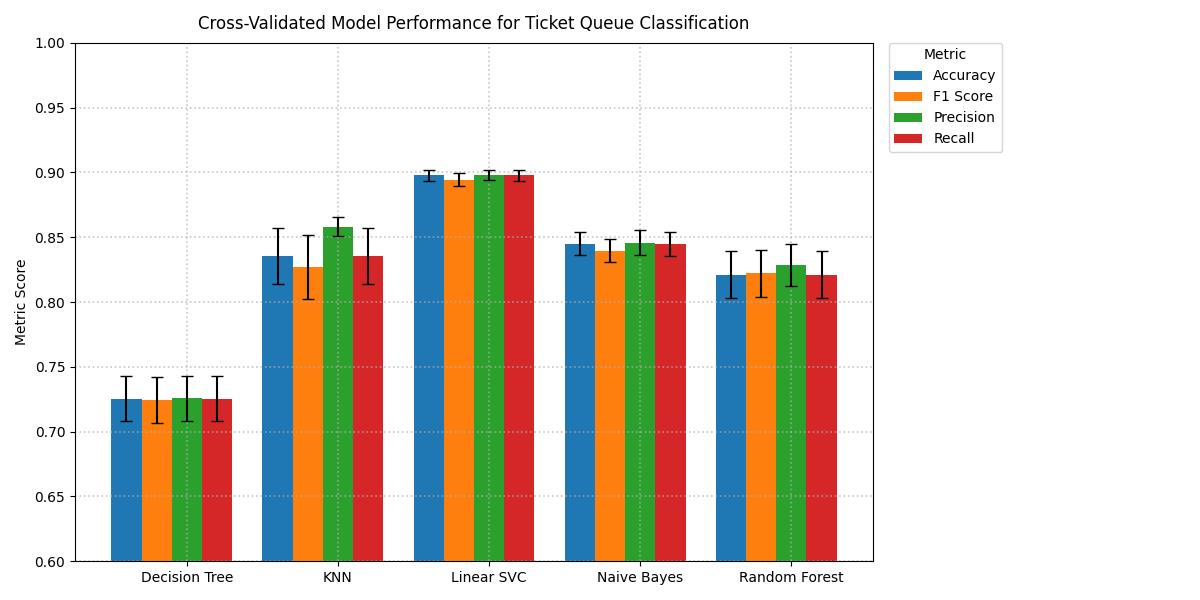
### **Cross-Validation Results and Metric Comparison**

Cross-validation is essential for evaluating the performance and generalization ability of machine learning models (Bhagat and Bakariya, 2025). It helps in model assessment, hyperparameter tuning, and identifying potential issues. To ensure fair performance comparisons across models, I applied 5-fold stratified cross-validation to both the queue classification and priority classification tasks. Stratified k-fold was specifically chosen to preserve the class distribution within each fold, allowing for more reliable and precise performance evaluation (Bhagat and Bakariya, 2025).  
  
Each model was evaluated using four key performance metrics:

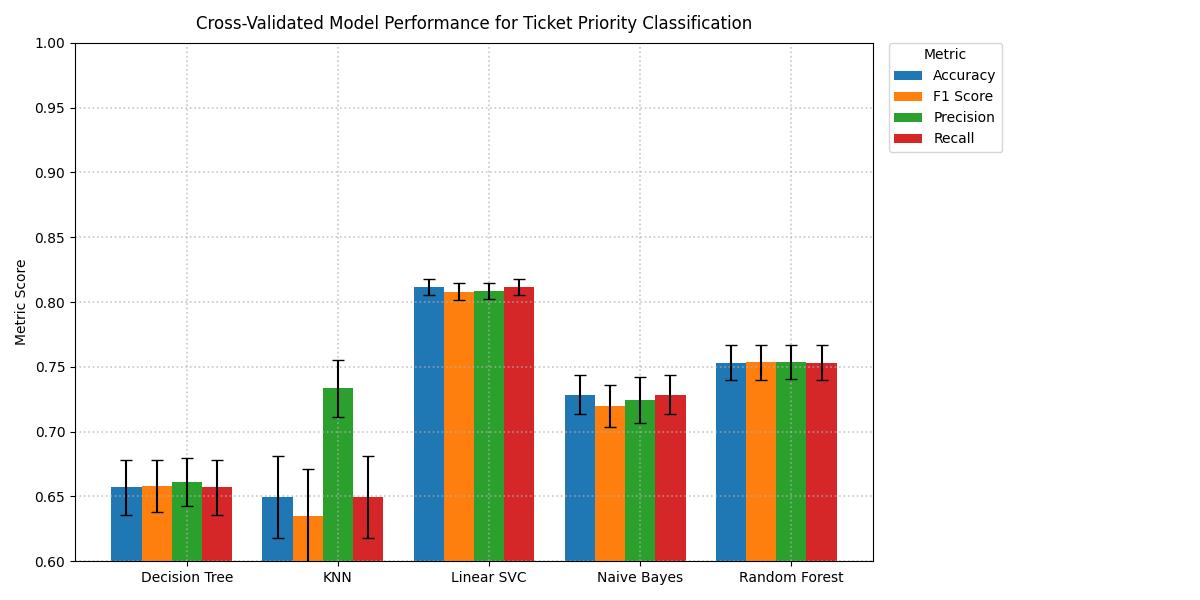
|  |  |
| --- | --- |
| **Accuracy** | The ratio of correct predictions to total predictions, indicating how often the model is correct overall. |
| **Macro Precision** | Calculates precision for each class individually and then averages the results. It shows how accurate the model’s positive predictions are across all classes, giving equal importance to each one. |
| **Macro Recall** | Measures recall for each class and averages the scores. It indicates how well the model identifies all true instances in each class, regardless of class size. |
| **Macro F1 Score** | Computes the F1 score for each class and then averages them. It reflects the model’s overall balance between precision and recall, treating every class equally. |

Each metric was computed using cross\_val\_score() in scikit-learn, with the mean and standard deviation calculated across five cross-validation folds. This approach allows for evaluating not only a model's overall performance but also its stability and variance.

In both the queue and priority classification tasks, Linear SVC was the top-performing model. It achieved the highest accuracy, macro precision, macro recall, and macro F1 scores within each task. The small standard deviations indicate consistent performance across folds. Additionally, its balanced metric scores suggest strong generalization capabilities and fairness across classes.



*Figure 13 : Cross-validated Model Performance for Ticket Queue Classification*



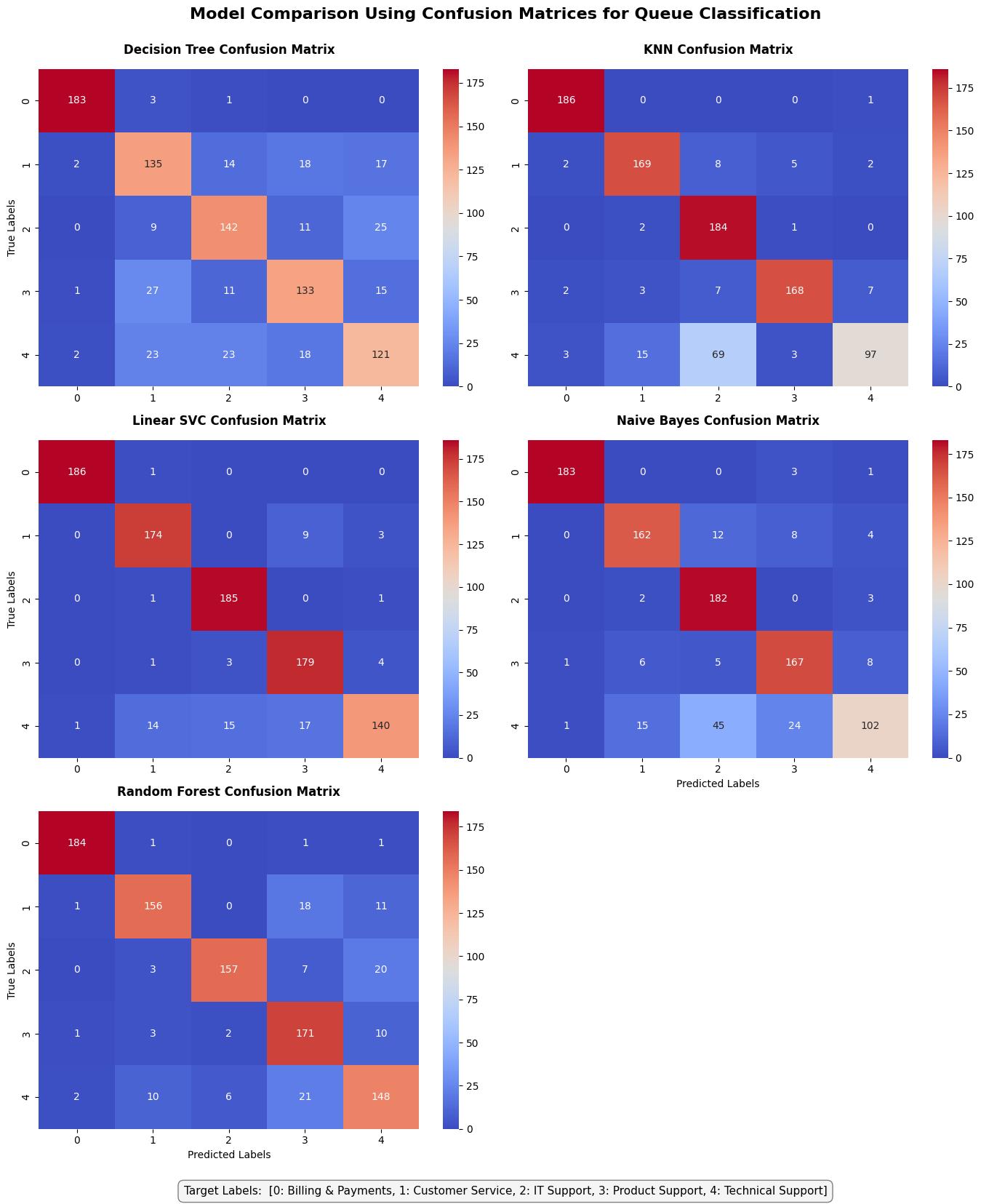
*Figure 14 : Model Performance Comparison for Ticket Priority Classification (Cross-Validation Results)*

### **Confusion Matrix Analysis**

While cross-validation metrics provide a general overview of model performance, confusion matrices offer a more detailed perspective on the classification results. Confusion matrices can be a powerful method for analyzing multi-class classifiers to show prediction distribution in one view (Heydarian, Doyle and Samavi, 2022). This is important for multi-class classification tasks like ticket queue and priority classification, as it provides insights into the model's performance on specific classes.

In the queue and priority classification tasks, Linear SVC was the top-performing model based on the confusion matrix analysis. It demonstrated consistently high accuracy across all classes with minimal confusion between them. Random Forest ranked second in both tasks, showing strong class-level accuracy and relatively low confusion, though it exhibited slightly more misclassifications than Linear SVC. Decision Tree ranked third, offering moderate accuracy with some class confusion, while KNN and Naive Bayes performed the worst overall, showing high misclassification rates and significant difficulty distinguishing between closely related classes.

On the following page, I provided visualizations of confusion matrix analysis comparison of the models for both queue and priority classification tasks.



*Figure 14 : Model Performance Comparison for Ticket Priority Classification (Cross-Validation Results)*

*Figure 15 : Model Comparison Using Confusion Matrices for Ticket Queue Classification*

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

*Figure 16 : Model Comparison Using Confusion Matrices for Ticket Priority Classification*

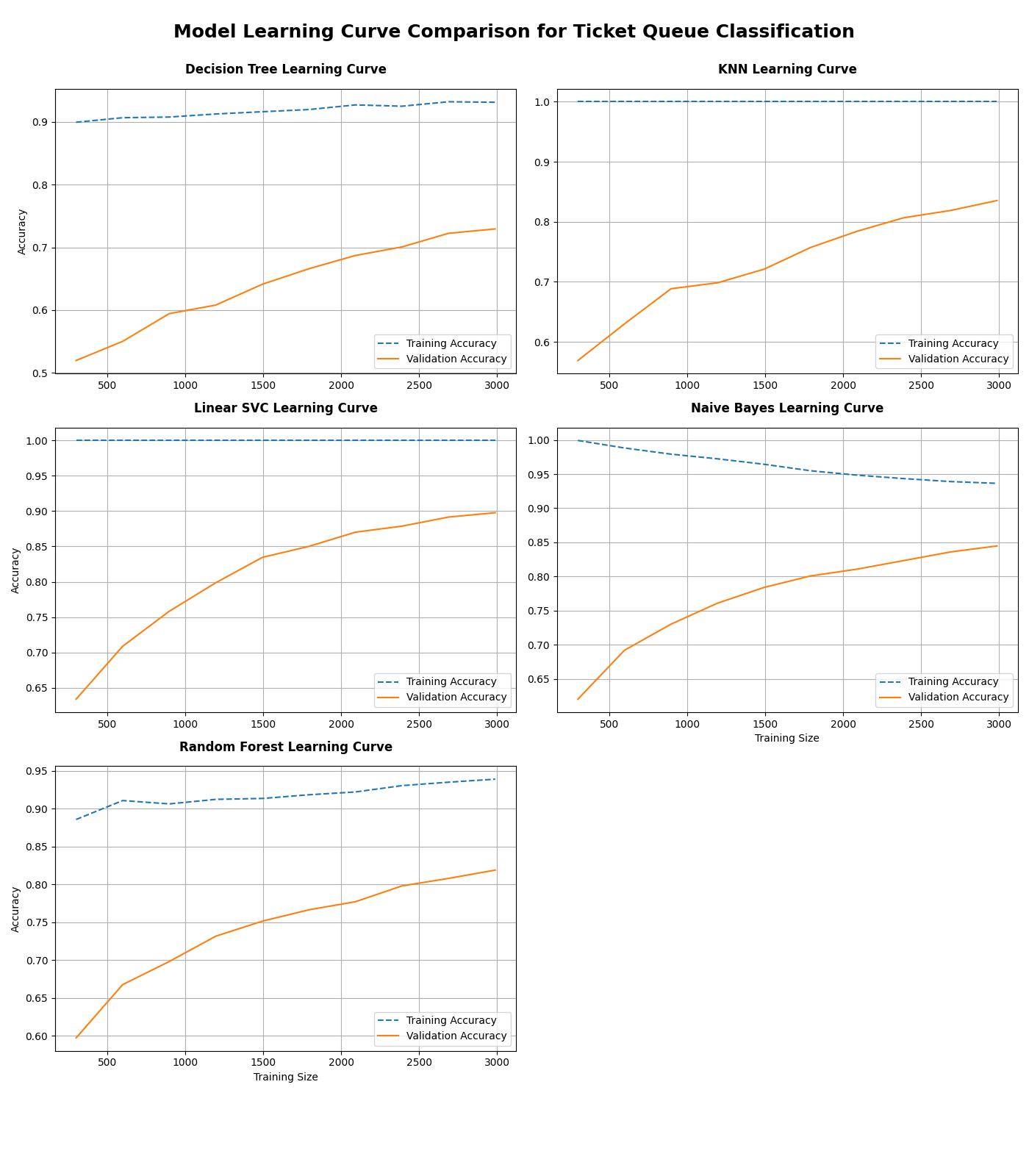
### **Learning Curve Analysis**

I have conducted a learning curve analysis on the models to evaluate whether their learning performance indicates overfitting, underfitting, or good generalization. I have chosen to implement a learning curve analysis as it can provide insight into such learning behavior by plotting generalization performance against the number of training examples (Viering and Loog, 2021). They help determine whether a model is suffering from underfitting, overfitting, or whether it would benefit from additional training data.

Across both the ticket queue and priority classification tasks, Linear SVC consistently demonstrated the strongest learning performance. It improved steadily with more data, generalized well, and achieved the highest validation accuracy. The narrow and decreasing gap between the training and validation curves indicates strong generalization, suggesting that the model continues to benefit from additional data.

Random Forest also performed well on both tasks, demonstrating strong validation accuracy and consistent improvement as the training size increased. However, it showed a slightly larger gap between training and validation accuracy, indicating mild overfitting compared to Linear SVC. Other models, such as Decision Tree, KNN, and Naive Bayes, exhibited weaker generalization and lower overall validation performance.

On the following page, I have provided visualizations comparing the learning curve analyses of the models for both the queue and priority classification tasks.



*Figure 17 : Model Learning Curve Comparison for Ticket Queue Classification*

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

*Figure 18 : Model Learning Curve Comparison for Ticket Priority Classification*

### 

## **Text Summarization**

In this section, I applied text summarization techniques on the ticket body column. The objective is to assist support agents by providing concise summaries on tickets, enabling them to quickly understand the core issue that the customer is experiencing. I implemented both extractive and abstractive summarization methods to evaluate their performance and suitability.

### **Text Preprocessing for Summarization**

The text preprocessing approach applied to text summarization was different from the one used for the ticket queue and ticket classification. For text summarization, the focus was on cleaning the text while preserving its meaning and structure. I took the following actions:

|  |  |
| --- | --- |
| **Contraction Expansion** | Common contractions were expanded to their full forms to enhance clarity for summarization. |
| **HTML tags Removal** | Removed as they do not contribute to the meaning of the content and can confuse language models, leading to lower-quality summaries. |
| **URL and Email Removal** | URLs and email addresses were removed from the text, as they do not provide information about the issue. |
| **Whitespace Trimming** | Leading and trailing whitespace characters were removed to eliminate formatting inconsistencies. |
| **Removed greetings and sign-off lines** | Common greetings and sign-off lines were removed, as they did not contribute to the core issue. |

### **Extractive Summarization**

Extractive summarization examines the original text and extracts the sentences that best convey the ideas without losing too much meaning. Its core principle is to rank the sentences by importance (Ghosh and Gunning, 2019). The advantage of extractive summarization is that it preserves the original meaning and context of the text. However, it can lack flexibility in language, as it is limited to the source text. For this capstone project I implemented BERT (Bidirectional Encoder Representations from Transformers) for extractive summarization for the ticket body column.

### **Abstractive Summarization**

Abstractive summarization is a method of creating an abstract summary that does not rely on using the same words from the original document. Instead, it rewrites the content in a more concise form (Ghosh and Gunning, 2019). The advantage of abstractive summarization is that it produces more fluent and human-like summaries. However, it can sometimes generate inaccurate or misleading information. For this capstone project, I implemented BART (Bidirectional and Auto-Regressive Transformers) for abstractive summarization of the ticket body column.

# **Challenges Encountered & Solutions**

Throughout the capstone project, I encountered several challenges that emerged during both data preparation and model development. These obstacles required careful, strategic decisions to ensure that the machine learning implementation remained effective, scalable, and aligned with the project's goals and objectives. The table below highlights key challenges and the solutions implemented to address them.

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Description** | **Solution** |
| Imbalanced Classes | Ticket queues and priority levels were unevenly distributed, risking biased model predictions. | Applied SMOTE to balance the dataset for both queue and priority classification tasks. |
| Multilingual Dataset | Tickets were written in five different languages, complicating consistent NLP preprocessing. | Translated all ticket content into English to enable uniform and effective text processing workflows. |
| Model Selection | Selecting the most appropriate machine learning model for ticket queue and priority classification. | Evaluated and compared multiple models using cross-validation, confusion matrix analysis, learning curve analysis, and assessments of model speed and accuracy. |

# 

# Conclusion

This project focused on the classification and summarization of customer support tickets to improve the efficiency of ticket routing and enhance customer support agent productivity. After evaluating multiple machine learning models, the Linear Support Vector Classifier (Linear SVC) was the most effective model for both ticket priority classification and ticket queue classification.   
  
For both tasks, Linear SVC consistently achieved the highest performance across all key evaluation metrics. Its confusion matrices indicate minimal misclassifications, and the learning curves demonstrated strong generalization and improved performance as more training data was introduced. Additionally, it offers fast prediction times, making it highly suitable for real-time production environments.

**Cross-validated Model Performance for Ticket Queue Classification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Linear SVC** | 0.898 | 0.898 | 0.898 | 0.894 |

**Cross-validated Model Performance for Ticket Priority Classification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Linear SVC** | 0.812 | 0.809 | 0.812 | 0.808 |

Regarding **summarization**, extractive summarization was chosen as the most suitable approach. Since support agents require accurate and comprehensive overview of customer issues, extractive summarization preserves the original wording and ensures that no critical information is lost. This is especially important in technical or sensitive cases where precision is crucial and paraphrasing could lead to misinterpretation.

# 

# References

* A Collaborative Research Study (2021) Support Transformation 2021: ServiceXRG. Available at:<https://f.hubspotusercontent10.net/hubfs/7830314/SupportTransformation%202021_ServiceXRG.pdf> (Accessed: 13 May 2025).
* Agrawal, T. (2020) Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient. 1st edn. New York: Apress.
* Bhargav Srinivasa-Desikan (2018) Natural Language Processing and Computational Linguistics : A Practical Guide to Text Analysis with Python, Gensim, SpaCy, and Keras. Birmingham, UK: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=a6c95e9a-7760-30c7-98f2-096e5a66549c (Accessed: 17 May 2025).
* Brownlee, J. (2020) *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery.
* Brownlee, J. (2021). SMOTE oversampling for imbalanced classification. Machine Learning Mastery. Available at: <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/> (Accessed: 17 May 2025).
* Bueck, T. (2024). Customer Support Emails - Ticket System - Helpdesk [Dataset]. Available at: <https://www.kaggle.com/datasets/tobiasbueck/multilingual-customer-support-tickets> (Accessed: 13 May 2025). Licensed under: Creative Commons Attribution 4.0
* Chopra, D., Joshi, N. and Mathur, I.(2016). *Mastering Natural Language Processing with Python*. Birmingham: Packt Publishing.
* European Commission. (2019). *Ethics guidelines for trustworthy AI*. High-Level Expert Group on Artificial Intelligence. Available at:<https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=60419> (Accessed: 15 May 2025).
* Ilemobayo, J.A., Durodola, O.I., Adewumi, T.O. and Awotunde, O.J. (2024). *Hyperparameter tuning in machine learning: A comprehensive review*. Journal of Engineering Research and Reports, 26(6), pp.388–395.
* Meenu Bhagat and Dr. Brijesh Bakariya (2025) ‘A Comprehensive Review of Cross-Validation Techniques in Machine Learning’, International Journal on Science and Technology, 16. doi:10.71097/ijsat.v16.i1.1305.
* Mohammadreza Heydarian, Thomas E. Doyle and Reza Samavi (2022) ‘MLCM: Multi-Label Confusion Matrix’, IEEE Access, 10, pp. 19083–19095. doi:10.1109/ACCESS.2022.3151048.
* Nirant Kasliwal (2018) Natural Language Processing with Python Quick Start Guide : Going From a Python Developer to an Effective Natural Language Processing Engineer. Birmingham: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=c26cdc97-b963-3350-bfd2-f23a7aea9169 (Accessed: 16 May 2025).
* P K Biju et al. (2018) ‘Multiclass Text Classification and Analytics for Improving Customer Support Response through different Classifiers’, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 538–542. doi:10.1109/icacci.2018.8554881.
* Raschka, S. and Mirjalili, V., 2020. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. 3rd ed. Birmingham: Packt Publishing.
* Service Excellence Research Group, LLC. (2025) *2025 AI for support: Use cases, risks and quick wins*. Available at:<https://7830314.fs1.hubspotusercontent-na1.net/hubfs/7830314/AI%20for%20Support.pdf> [Accessed 15 May 2025].
* Scutt, J. and Quaadgras, T. (2024) The state of customer experience management, 2024. Qualtrics. Available at: <https://www.qualtrics.com/m/www.xminstitute.com/wp-content/uploads/2024/05/XMI_RR_DS_StateCXManagement-2024.pdf?ty=mktocd-thank-you> (Accessed: 13 May 2025).
* Sohom Ghosh and Dwight Gunning (2019) *Natural Language Processing Fundamentals : Build Intelligent Applications That Can Interpret the Human Language to Deliver Impactful Results*. Birmingham: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=b57f566d-8d1f-3caa-8a30-2fd48d76f960 (Accessed: 16 May 2025).
* Truss, M. and Boehm, S. (2024) ‘AI-based Classification of Customer Support Tickets: State of the Art and Implementation with AutoML’, Proceedings of the IWEMB 2021/2022: Fifth and Sixth International Workshop on Entrepreneurship, Electronic and Mobile Business [Preprint]. Available at: https://research.ebsco.com/linkprocessor/plink?id=54e9ee0f-a797-3f1f-b754-ba84ad754cbe (Accessed: 13 May 2025).
* Uysal, A.K. and Günal, S. (2014) ‘The impact of preprocessing on text classification’, *Information Processing & Management*, 50, pp. 104–112. doi:10.1016/j.ipm.2013.08.006.
* Viering, T. and Loog, M. (2021) ‘The Shape of Learning Curves: a Review’. Available at: https://research.ebsco.com/linkprocessor/plink?id=a4842dab-a070-3d8b-82ef-5dcc208969ff

# 

# 

# Bibliography

* A Collaborative Research Study (2021) Support Transformation 2021: ServiceXRG. Available at:<https://f.hubspotusercontent10.net/hubfs/7830314/SupportTransformation%202021_ServiceXRG.pdf> (Accessed: 13 May 2025).
* Agrawal, T. (2020) Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient. 1st edn. New York: Apress.
* Bhargav Srinivasa-Desikan (2018) Natural Language Processing and Computational Linguistics : A Practical Guide to Text Analysis with Python, Gensim, SpaCy, and Keras. Birmingham, UK: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=a6c95e9a-7760-30c7-98f2-096e5a66549c (Accessed: 17 May 2025).
* Brownlee, J. (2020) *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery.
* Brownlee, J. (2021). SMOTE oversampling for imbalanced classification. Machine Learning Mastery. Available at: <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/> (Accessed: 17 May 2025).
* Bueck, T. (2024). Customer Support Emails - Ticket System - Helpdesk [Dataset]. Available at: <https://www.kaggle.com/datasets/tobiasbueck/multilingual-customer-support-tickets> (Accessed: 13 May 2025). Licensed under: Creative Commons Attribution 4.0
* Chopra, D., Joshi, N. and Mathur, I.(2016). *Mastering Natural Language Processing with Python*. Birmingham: Packt Publishing.
* European Commission. (2019). *Ethics guidelines for trustworthy AI*. High-Level Expert Group on Artificial Intelligence. Available at:<https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=60419> (Accessed: 15 May 2025).
* Ilemobayo, J.A., Durodola, O.I., Adewumi, T.O. and Awotunde, O.J. (2024). *Hyperparameter tuning in machine learning: A comprehensive review*. Journal of Engineering Research and Reports, 26(6), pp.388–395.
* Meenu Bhagat and Dr. Brijesh Bakariya (2025) ‘A Comprehensive Review of Cross-Validation Techniques in Machine Learning’, International Journal on Science and Technology, 16. doi:10.71097/ijsat.v16.i1.1305.
* Mohammadreza Heydarian, Thomas E. Doyle and Reza Samavi (2022) ‘MLCM: Multi-Label Confusion Matrix’, IEEE Access, 10, pp. 19083–19095. doi:10.1109/ACCESS.2022.3151048.
* Nirant Kasliwal (2018) Natural Language Processing with Python Quick Start Guide : Going From a Python Developer to an Effective Natural Language Processing Engineer. Birmingham: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=c26cdc97-b963-3350-bfd2-f23a7aea9169 (Accessed: 16 May 2025).
* P K Biju et al. (2018) ‘Multiclass Text Classification and Analytics for Improving Customer Support Response through different Classifiers’, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 538–542. doi:10.1109/icacci.2018.8554881.
* Raschka, S. and Mirjalili, V., 2020. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. 3rd ed. Birmingham: Packt Publishing.
* Rauf, O. (2023) *Exploring traditional evaluation plots for multiclass classification models*. [online] *Medium*. Available at:<https://medium.com/@owaisr814/exploring-traditional-evaluation-plots-for-multiclass-classification-models-5bd8ec80d3da> [Accessed 18 May 2025].
* Service Excellence Research Group, LLC. (2025) *2025 AI for support: Use cases, risks and quick wins*. Available at:<https://7830314.fs1.hubspotusercontent-na1.net/hubfs/7830314/AI%20for%20Support.pdf> [Accessed 15 May 2025].
* Scutt, J. and Quaadgras, T. (2024) The state of customer experience management, 2024. Qualtrics. Available at: <https://www.qualtrics.com/m/www.xminstitute.com/wp-content/uploads/2024/05/XMI_RR_DS_StateCXManagement-2024.pdf?ty=mktocd-thank-you> (Accessed: 13 May 2025).
* Sohom Ghosh and Dwight Gunning (2019) *Natural Language Processing Fundamentals : Build Intelligent Applications That Can Interpret the Human Language to Deliver Impactful Results*. Birmingham: Packt Publishing. Available at: https://research.ebsco.com/linkprocessor/plink?id=b57f566d-8d1f-3caa-8a30-2fd48d76f960 (Accessed: 16 May 2025).
* Truss, M. and Boehm, S. (2024) ‘AI-based Classification of Customer Support Tickets: State of the Art and Implementation with AutoML’, Proceedings of the IWEMB 2021/2022: Fifth and Sixth International Workshop on Entrepreneurship, Electronic and Mobile Business [Preprint]. Available at: https://research.ebsco.com/linkprocessor/plink?id=54e9ee0f-a797-3f1f-b754-ba84ad754cbe (Accessed: 13 May 2025).
* Uysal, A.K. and Günal, S. (2014) ‘The impact of preprocessing on text classification’, *Information Processing & Management*, 50, pp. 104–112. doi:10.1016/j.ipm.2013.08.006.
* Viering, T. and Loog, M. (2021) ‘The Shape of Learning Curves: a Review’. Available at: https://research.ebsco.com/linkprocessor/plink?id=a4842dab-a070-3d8b-82ef-5dcc208969ff (Accessed: 18 May 2025).

# 