

Acknowledgments

I would like to express my sincere gratitude to BITS Pilani and Wipro Technologies for providing me with the opportunity to apply and expand the skills I have acquired throughout my academic journey. Also, My sincere thanks go to my Supervisor and Examiners for their guidance and support throughout this process. Their expertise and insights have greatly contributed to the development of this dissertation.

I believe this experience will help me further explore this field and try new things in the future. I am truly appreciative of the opportunities and learning experiences provided by all those involved.

Abstract

This project develops an innovative ITIL incident ticketing tool enhanced with machine learning (ML) capabilities. ITIL (Information Technology Infrastructure Library) is a set of best practices for IT service management (ITSM) that helps organizations align their IT services with the needs of the business. Incident Management, a crucial ITSM process, manages the lifecycle of incidents to restore normal service operations swiftly. Effective incident management relies on a centralized platform(Ticketing tool) for reporting, documenting, and resolving incidents.

Once incident is logged, delays can occur if an IT agent may lack the necessary knowledge or skill in identifying the appropriate team for incident resolution, often involving time-consuming searches through historical tickets or escalation procedures. This project proposes integrating a machine learning-based feature into the ticketing tool to predict the relevant assignment group based on the incident description. By leveraging historical ticket data, the proposed functionality aims to streamline ticket assignment, allowing agents to quickly direct tickets to the appropriate support teams. This approach is expected to reduce resolution times and minimize operational impact.

Table of Contents

• <u>List of Symbols and Abbreviations</u>	07
• <u>Literature Review</u>	08
• <u>Introduction</u>	09
• <u>Overview of ITIL v4</u>	10
• <u>ITIL Incident Management Process</u>	
• <u>Overview of Ticketing Tool in ServiceNow</u>	11
• <u>Plan Stage in PACE Framework</u>	12
○ <u>Research Objective</u>	
○ <u>Data Collection</u>	
• <u>Analyse Stage in PACE Framework</u>	14
○ <u>Initial Text Preprocessing</u>	
○ <u>Text Cleaning Steps1</u>	
○ <u>Text Cleaning Steps2</u>	
• <u>Let's Understand Topics</u>	21
• <u>Feature Transformation & Initial Model Building</u>	24
○ <u>Bag of Words (BoW)</u>	
○ <u>Handling Imbalance Dataset</u>	
○ <u>Train, Validation and Test Datasets</u>	
○ <u>Traditional Models with TF-IDF</u>	
• <u>Hypothesis Testing on Text Length Column</u>	26
• <u>Construct Stage in PACE Framework</u>	27
○ <u>More on Logistic Regression</u>	
○ <u>Deep Learning Models with Embeddings</u>	
• <u>Comparison of Models Performance</u>	33
• <u>Conclusion</u>	34
• <u>Execute Stage in PACE Framework</u>	35
○ <u>Application Interface and Testing</u>	
○ <u>Constraints and Data Validation</u>	
• <u>References</u>	37
• <u>Appendices & Checklist of Items</u>	38

List of Symbols and Abbreviations

Symbol	Meaning
X	Input features (short description, description, text length)
y	Target variable (assignment group)
ITSM	IT Service Management
ITIL	Information Technology Infrastructure Library
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequency
DT	Decision Tree
RT	Random Forest
SMOTE	Synthetic Minority Oversampling Technique
PCA	Principal Component Analysis
SVC	Support Vector Classifier
OvO	One-vs-One
ANOVA	Analysis of Variance
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers.
GloVe	Global Vectors for Word Representation
Word2Vec	word to vector
H ₀	Null Hypothesis
H _a	Alternative Hypothesis

List of Tables / Figures

Table/Figure	Title
Figure 1	Source: ticket from ServiceNow free Personal Developer Instances (PDI)
Figure 2	PACE Framework
Table 1	Ticketing Data Dictionary with 8500 rows.
Figure 3	Ticket Frequency in Assignment Groups
Figure 4	Top 20 Words in Descriptions
Figure 5	Top 20 Words in each assignment group
Figure 6	Wordclouds for all 16 assignment groups
Table 2	Topics for each group
Figure 7	Comparison of Scores of different ML Models
Figure 8	Median Text Length by Assignment Group
Figure 9	Tukey's HSD Post-Hoc Test for Text Length by Assignment Group
Figure 10	Scores of LR[Model 4;Table 3] and LSTM[Model 8;Table 3] on Unseen Data
Table 3	Performance Comparision of All Models
Figure 11	Front Page of Streamlit Application
Figure 12	Testing the application with an example prediction
Figure 13	Testing the Constraints and Data Validation

Literature Review

As suggested by Revina et al. (2020)[1], simpler approaches can be just as effective as more complex ones by focusing on text preparation and feature engineering (creating additional features). In this project, we focused more on text preparation and feature engineering of data, which helped to gain good results with simpler models.

Also, This paper by Fuchs et al. (2022)[2] conducts a literature review on the application of machine learning (ML) to improve support ticket systems (STS). Their findings highlight the potential of Machine Learning to automate various tasks within STSs, leading to increased efficiency and improved customer service. The review suggests Random Forest (RF) and Support Vector Machines (SVM) as the best performing algorithms for ticket classification.

We tested and compared these models.

In this project, the TF-IDF[3] technique was used for feature transformation, and Machine Learning models[4] such as Logistic Regression (One vs. One), Random Forest, and Multiclass Naive Bayes (Multinomial Naive Bayes) were employed, all of which achieved good results. Since SVM took longer to run, PCA[5] was used to reduce dimensionality (feature count) in our model. Additionally, Bidirectional LSTM[6] for training with Word Embeddings[Word2Vec[7] and Glove[8]] and BERT[9] for text augmentation used. The accuracy and other metrics of all these models were compared.

Hyperparameter tuned - Logistic Regression by adding text length as an additional feature, along with augmented data to address class imbalance, led to improved results.

Introduction

ITIL (Information Technology Infrastructure Library) is a set of best practices for IT service management (ITSM) that helps organizations align their IT services with the needs of the business. Incident Management is a part of the IT Service Management processes that manage the lifecycle of incidents to ensure that normal service operation is restored as quickly as possible.

Effective Incident Management requires a centralized platform for reporting incidents, documentation, and resolution. Traditional ticketing systems address this need, but with the evolution of new technologies like Machine Learning (ML) and AI, we can make these tools even more powerful. This can significantly minimize the time to restore services.

In this project, I will leverage an existing ticketing tool in the market to demonstrate the broader role ticketing systems play in the IT industry. However, my main focus will be developing a more advanced ticketing tool functionality using Machine Learning(Supervised). This will aim to significantly improve efficiency by predicting the most suitable assignment group for incidents based on their descriptions, minimizing impact time for end users, and assisting agents in faster resolution.

Overview of ITIL V4

ITIL(Information Technology Infrastructure Library)[10], is a considered as one of the best frameworks for ITSM(IT Service Management). ITIL v4 provides a comprehensive framework that helps organizations manage IT services more effectively, adapt to new challenges, and deliver greater value to their customers.

The key components of this process:

1. Service Value System (SVS): It ensures that the organization continually co-creates value from its products and services. It includes:

- **Service Value Chain:** It is the central element of Service Value Systems, It contains the key activities to value. It includes Plan, Improve, Engage, Design & Transition, Obtain/Build, and Deliver & Support.
- **Guiding Principles:** There are seven guiding principles presented that an organization follow in any circumstances.
 1. Focus on Value
 2. Start Where You Are
 3. Progress Iteratively with Feedback
 4. Collaborate and Promote Visibility
 5. Think and Work Holistically
 6. Keep It Simple and Practical
 7. Optimize and Automate
- **Governance:** It ensures that policies and decisions align with the organization's objectives and strategy.
- **Practices:** ITIL v4 describes 34 management practices, which are broader than processes and include various activities and resources. Incident, major incident management is also one of the practices.

- **Continual Improvement:** This practice is integral to the SVS and emphasizes the importance of ongoing enhancement of services and processes.

2. ITIL 4 Practices: These 34 practices are grouped into three categories.

- **General Management Practices:** These are applicable across various types of organizations and include practices like Strategic Planning, Risk Management, and Continual Improvement.
- **Service Management Practices:** Directly related to the delivery of IT services, such as Incident Management, Problem Management, and Change Control.
- **Technical Management Practices:** Focus on the technical aspects of IT services, including Deployment Management and Infrastructure and Platform Management.

3. Service Value Chain: It provides flexible operating model for the creation, delivery, and improvement of services. It integrates with the guiding principles to support value co-creation.

ITIL Incident Management Process

ITIL Incident Management is part of Service management Practices of ITIL framework and it is a structured approach to handling IT disruptions or issues within IT services promptly and effectively. It's a reactive process focused on restoring normal service operation as quickly as possible and minimize the customer impact as quickly as possible.

Key Objectives:

- Minimize downtime and service interruptions.
- Ensure rapid restoration of normal service operation.
- Provide timely communication to affected users.

The Incident Management Process

1. **Incident Detection:** Identifying the occurrence of an incident by an individual, team or customer.
2. **Incident Logging:** Recording detailed information about the incident using a central ticketing tool.
3. **Incident Classification and Assignment Group:** Categorizing the incident based on its type (e.g., hardware, software, network) and determine the appropriate team or individual responsible for handling the incident based on the classification.
4. **Incident Prioritization:** Assigning a priority level[p1,p2,p3,p4] to the incident based on its impact and urgency.
5. **Incident Investigation:** Analysing the incident to understand its root cause and potential solutions.
6. **Incident Resolution:** Implementing the chosen solution to restore normal service operation.
7. **Incident Closure:** Verifying that the incident is resolved and closing the incident record.

Integration with Other ITIL Processes: Incident Management is closely linked to other ITIL processes, such as:

- **Problem Management:** Identifying and resolving the underlying causes of incidents.
- **Change Management:** Managing changes to the IT environment to prevent new incidents.
- **Service Request Management:** Handling service requests that do not disrupt service

Overview of Ticketing Tool in ServiceNow

The screenshot shows the ServiceNow interface for an incident ticket. The top navigation bar includes 'All', 'Favorites', 'History', 'Workspaces', and 'Admin'. The incident title is 'Incident - INC0000029'. The form is divided into two main sections: 'Incident' and 'Resolution Information'. The 'Incident' section contains fields for Number (INC0000029), Caller (Charlie Whitherspoon), Category (Inquiry / Help), Subcategory (-- None --), Service, Service offering, Configuration item (WeatherBug), Short description (I can't get my weather report), and Description (WeatherBug icon has disappeared from my desktop. Unable to get my weather report.). The 'Resolution Information' section contains fields for Channel (Phone), State (In Progress), Impact (3 - Low), Urgency (3 - Low), Priority (5 - Planning), Assignment group (Service Desk), and Assigned to (Don Goodliffe). At the bottom, there are tabs for 'Notes', 'Related Records', and 'Resolution Information', and a 'Work notes list' section.

Figure 1. Source: ticket from ServiceNow free Personal Developer Instances (PDI)

In the context of IT service management, ticketing tools like ServiceNow play a crucial role in handling and resolving user issues efficiently. This overview outlines the process of creating and managing a ticket using a ServiceNow developer instance.

A typical ticket in ServiceNow includes several key fields (Figure 1):

Number: A unique identifier for each ticket, automatically generated by the system.

Caller: The individual reporting the issue.

Category/Subcategory: Classifications that help in organizing the issue by type.

Service and Configuration Item: These fields identify the affected service and specific system component.

Short Description and Description: Provide a brief summary and detailed explanation of the issue.

Priority, Impact, and Urgency: These fields help in determining the ticket's importance and the order in which it should be addressed.

Assignment Group and Assigned to: Define who is responsible for resolving the ticket.

State and Work Notes: Track the progress and internal updates on the issue.

This structured approach ensures that issues are logged, prioritized, and managed systematically, enhancing the efficiency of IT service delivery.

Problem Statement:

Manual assignment of IT support tickets to the appropriate support groups is a time-consuming and error-prone process. This often leads to delayed resolution times and decreased customer satisfaction. To address this issue, we aim to develop a machine learning model that can automatically classify IT tickets based on their textual content, such as the ticket description and subject line. By accurately predicting the appropriate support group for each ticket, we can significantly improve the efficiency and effectiveness of the IT support process.

This project follows PACE Framework[11] from building machine learning to deployment.

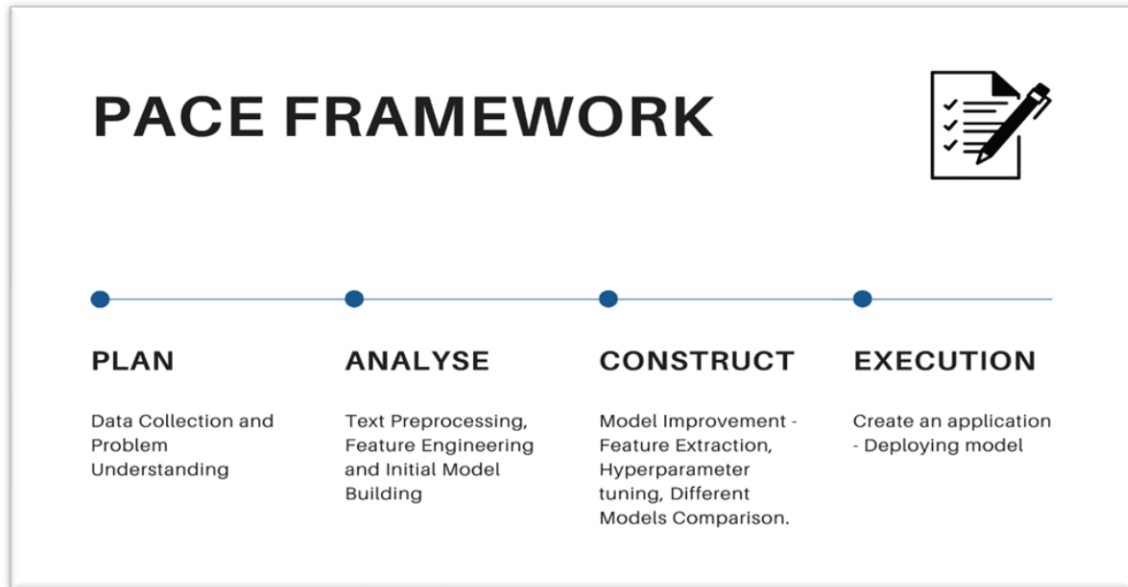


Figure 2. PACE Framework

Plan Stage in PACE Framework

Research Objective

This project aims to develop a novel ITIL incident ticketing tool functionality integrated with machine learning (ML).

Expected Outcome: The developed tool will utilize ML to Predict the most suitable assignment group for incident tickets based on their incident descriptions. This will enhance efficiency and accuracy in incident resolution.

System Requirements:

Hardware Requirements	Software Requirements
Computer/laptop: Operating System: Windows 10, macOS, or Linux. Processor: Intel Core i5 or equivalent. Memory (RAM): At least 8GB. Internet: A stable internet connection	Google Colab: Access through a Google account. Free access to basic GPU resources. Python: Version 3.7 or higher. Open source libraries are used. Required libraries: Scikit-learn[12] - Pandas, NumPy, nltk, TensorFlow[13], Hugging Face Transformers – BERT pretrained, nlpaug[14], Gensim etc. GitHub: A GitHub account for version control. Git installed locally (Version 2.20 or higher). Text Editor/IDE: Visual Studio Code or similar Streamlit account: Access through GitHub Dataset Requirements: Kaggle account

Data Collection

The dataset[15] used for this project was taken from Google datasets, specifically from Kaggle, that provides a variety of public datasets. The dataset includes detailed descriptions of incidents and their corresponding resolver groups. It is important to note that we made an assumption that the dataset contains information on closed or resolved tickets. Our task is text classification[16], here we have to classify assignment group based on the input given.

Due to the limitation that the available data includes only incident descriptions, I cannot leverage other important data from real-world tickets, such as category, sub-category, application name, and resolution notes, which could help build a more robust model.

Each row in this dataset represents a ticket of specific issue or request that has been reported.

Table 1. Ticketing Data Dictionary with 8500 rows.

Field Name	Description	Data Type
Short description	A brief summary or title of the issue or request	String
Description	Detailed information about the issue or request	String
Caller	The name or identifier of the person who reported the issue or made the request.	String
Assignment group	The team or group that is responsible for handling the issue or request.	String

Analyse Stage in PACE Framework

Initial Text Preprocessing

Observations from the data:

1. Dataset contains total of four columns[Short description, Description, Caller, Assignment group] with 8500 tickets data.
2. **Removing Duplicate:** Identified 83 duplicate ticket entries and removed duplicate texts to avoid redundancy in the dataset.
3. **Assignment Group Imbalance and Skew Analysis:** As shown in Figure 3, 74 Unique assignment groups were observed and most of the tickets(~46%) are belonged to Group_0, Data is imbalance and skewed towards Group0. Also, There are few groups having only one ticket assigned.

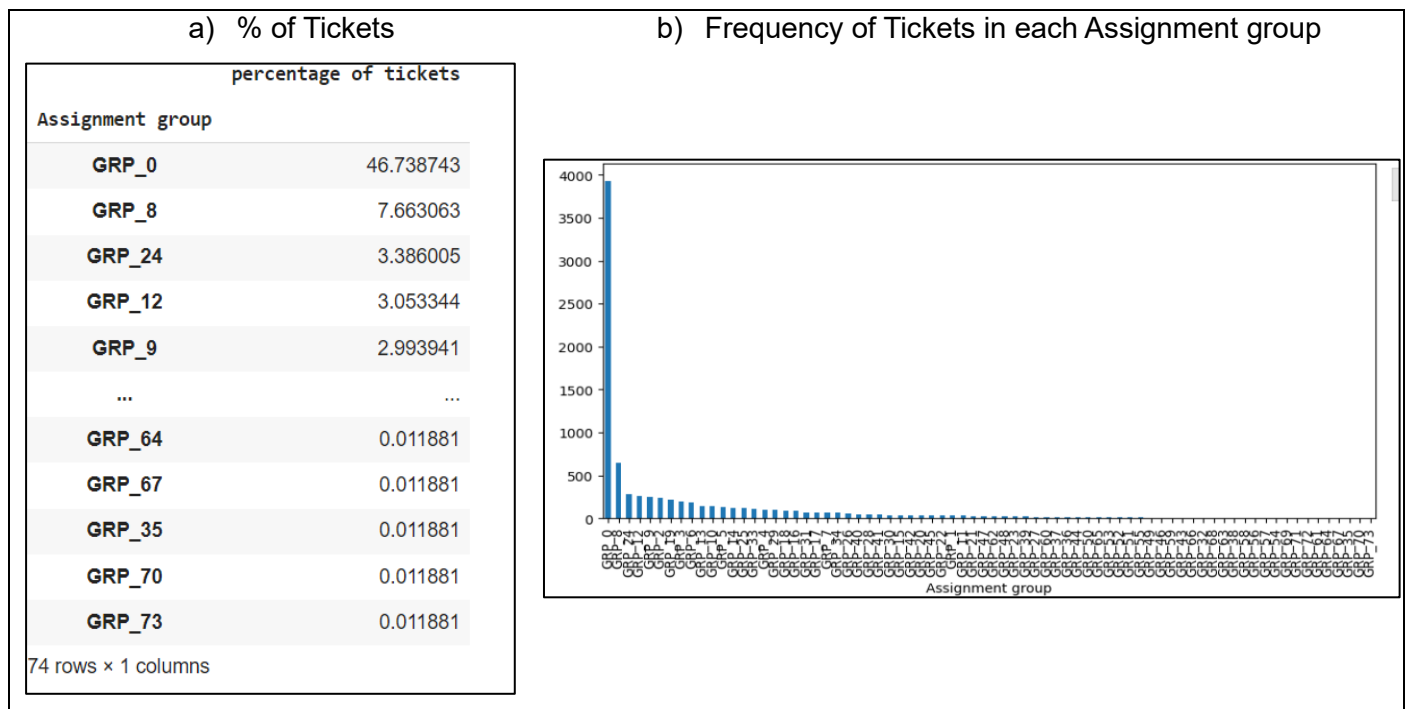


Figure 3. Ticket Frequency in Assignment Groups

- **Solution:** Assignment groups with less than 100 tickets will be removed to improve model performance and avoid over-fitting. With too few examples, the model might not generalize well and could lead to inaccurate predictions for underrepresented groups, also when a group has very few samples, the model might overfit to those specific examples, learning noise rather than generalizable patterns. This can lead to poor performance on unseen data. After compressing assignment group to less, final dataset1 contains 16 assignment groups with total of 7066 rows(tickets):
4. **Column Names Standardization & Pruning:** Standardized the column names by converting them to lowercase and combining words with underscores. Additionally, removed the "Caller" column as it does not provide any useful information for modelling.
 5. **Duplicate Description Rows Identification and filtering:** ~37% of the tickets contain the same content in both their short and long descriptions. To avoid duplicate and redundant data in modelling, short descriptions were replaced with a space when they match the long descriptions.

6. **Handling Missing Data:** In our dataset, null values(blanks) were found in both the short and long descriptions, but they were minimal. Instead of removing those rows completely, imputed the null values with a space in our ticket data.
7. **Merging Descriptions into a Single Field:** Both short and long descriptions were combined into a single column as 'description'. This is to simplify text preprocessing and can apply text cleaning techniques once, rather than separately for each description type going forward.

```
def merge_descriptions(df):  
  
    df['description'] = df['short_description'] + ' ' + df['long_description']  
    df = df.drop(['short_description', 'long_description'], axis=1)  
    return df  
  
df1 = merge_descriptions(df1)
```

Text Cleaning Steps1

Libraries & Modules Used for cleaning the text data:

```
# Kaggle API  
!pip install Kaggle -q  
  
# File handling and data manipulation  
import zipfile  
import pandas as pd  
from google.colab import files  
import pickle  
import warnings  
warnings.filterwarnings('ignore')  
  
# Numerical computation  
import numpy as np  
  
# Visualization  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Text processing and NLP  
import re  
import nltk  
from nltk.corpus import words  
from nltk.corpus import stopwords  
from nltk.tokenize import word_tokenize  
from collections import Counter  
from collections import defaultdict  
  
nltk.download('punkt')  
nltk.download('stopwords')  
nltk.download('wordnet')  
nltk.download('words')  
  
from tqdm import tqdm #to show progress bar  
  
!pip install googletrans==3.1.0a0 -q #for translation  
from googletrans import Translator  
  
!pip install pyspellchecker -q #for spelling check and correction  
from spellchecker import SpellChecker  
  
!pip install spacy -q #spacy for lemmatization  
import spacy  
!python -m spacy download en_core_web_sm -q
```

Observations from the Text Data and corrections:

1. **Case Sensitivity:** The data contains both uppercase and lowercase letters. To ensure consistent analysis, all text has been converted to lowercase.

2. **Line Breaks:** Descriptions frequently include carriage return (\x000d) and newline (\n) characters, which can disrupt text flow. These characters have been replaced with spaces to create a more coherent structure.
3. **Language Diversity:** While the majority of descriptions are in English, some contain text in other languages, such as German. To maintain consistency and facilitate analysis, these non-English descriptions have been translated into English.
4. **Special Characters:** The data contains a variety of special characters, including non-ASCII characters, punctuation marks, and symbols. These characters can interfere with text processing and analysis. To simplify the data, they have been removed using regular expressions[17].
5. **Masked User Information:** Some descriptions include masked email addresses and user names, such as masked_id@company.com, masked_id@gmail.com, [firstname.lastname@gmail.com](#), were found in the data. Replaced text with space.
6. **Image References:** A small number of descriptions contain references to image attachments. Since image data is not directly relevant to text analysis, these references have been removed.
7. **Numeric and Temporal Data:** Some of the descriptions include numbers, dates, and times. While this information can be valuable in certain contexts, it may not be directly relevant to the specific analysis goals. Therefore, these numeric and temporal elements have been removed.
8. **Common Words:** Certain words, such as greetings, 'received,' 'from,' 'hi,' 'hello,' 'please,' 'emailto,' 'kind,' and 'kindly' has found and need to be removed from descriptions to simplify the text and eliminate unnecessary complexity.

Created Python function called 'text_cleaning_steps_1' to perform above cleaning steps and created a new column as 'description_cleaned' for cleaned descriptions text in the dataframe.

```
def text_cleaning_steps_1(text):
    text = text.lower()
    translated = translator.translate(text, src='de', dest='en')#translating from German to English
    text = translated.text
    text = text.replace('\x000d',' ')
    text = text.replace('\n',' ')
    text = text.replace('â€', ' ')
    text = re.sub(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b', '',text) #General pattern to remove any email address
    text = re.sub(r'\S+\.S+@gmail\.com', '', text) # Specific pattern to remove emails like "something.something@gmail.com"
    text = re.sub(r'\[s*cid:[^\]]+\]', '',text) #images pattern
    text = re.sub(r'\d+', '', text) # removing numbers
    text = re.sub(r"['^w\s]", '', text) # removing special characters except '(apostrophe, to not to breaks words don't, won't etc
meaning)
    text = text.encode("ascii", "ignore").decode() # removing non-ascii characters
    text = ''.join([word for word in text.split() if word not in words_to_remove]) #removing few words
    text = re.sub(r"['^a-zA-Z0-9\s]", '', text) #removing punctuations
    text = text.strip()

    return text
```

```
for i in tqdm(range(len(df1))):
    df1.loc[i, 'description_cleaned'] = text_cleaning_steps_1(df1['description'].iloc[i])

100%|██████████| 7066/7066 [14:58<00:00, 7.86it/s]
```

Text Cleaning Steps2

Lemmatization

Lemmatization[18] is the process of reducing a word to its base or root form, known as its "lemma." Stemming process simply chops off affixes without considering the context but whereas in lemmatization it considers the context and converts words to their meaningful base form, ensuring that the resulting lemma is a valid word. It is one of the effective ways to reduce the dimensionality of text data.

Example: Consider the following words: “resolve” “resolving” “resolves”

- Without Lemmatization: These words would be treated as distinct entities in a text analysis.
- With Lemmatization: All these forms would be reduced to the lemma "resolve."
- **Solution:** Created python function called ‘*lemmatize_text*’ and performed lemmatization on the ‘*description_cleaned*’ column and added the result to the ‘*description_lemmatized*’ column.

```
nlp = spacy.load('en_core_web_sm')

def lemmatize_text(text):
    lemmas = []
    for token in nlp(text):
        if token.text.strip():
            lemmas.append(token.lemma_.lower())
    return lemmas

test_data = lemmatize_text('resolve resolving resolves')
print(test_data)

output: ['resolve', 'resolve', 'resolve']

df1['description_lemmatized'] = df1['description_cleaned'].apply(lemmatize_text)
```

Stop Words Removal

Stop words are commonly used words in a language that are often filtered out during text preprocessing in Natural Language Processing (NLP) tasks. These words include articles, prepositions, conjunctions, and other words that don't carry significant meaning and don't contribute much to the overall context of the text.

By removing stop words, noise will be reduced from the data and improve model performance.

- **Example:** Consider this description: “*unable to access hr tool page*”
 - If we remove stop word like “to” the sentence simplifies to: “*unable access hr tool page*”
 - This version of the sentence retains the core meaning but is now more concise, with fewer words for a machine learning model to process.
- *description_lemmatized* Column contains total of 127 unique stop words, which were removed.
- **Solution:** Created python function called ‘*remove_stopwords*’ and performed lemmatization on the ‘*description_lemmatized*’ column and added the result to the ‘*description_lemmatized_with_no_stop_words*’ column.

```
def remove_stopwords(tokens):
    return [token for token in tokens if token not in stop_words]

df1['description_lemmatized_with_no_stop_words'] = df1['description_lemmatized'].apply(remove_stopwords)
```

Removing Less Informative Words | Reducing Noise

Removing less informative words or tokens is a critical preprocessing step. This helps to focus the model on the more significant and distinctive features of the text, thereby improving the performance, accuracy of the classification and could be useful to mitigate the curse of dimensionality, improving the performance and efficiency classifier. Below noise words were found in descriptions and removed them.

- **Single-Letter Tokens:** There are some single words presented in the descriptions these words with just one letter don't add much meaning to the text. Removed them to help the model focus on more important words.
- **Less informative Tokens with low frequency:** Two and three letters words with low frequency(<10) and less informative are removed. Words and phrases like "request", "thank you", or "regards" are often too generic to be useful in classification, so removed them.
- **Yes/No/NA Responses:** If the text includes a lot of yes/no/NA answers, they might not help much in classifying the text into groups. Removed these responses to make the model focus on more meaningful content.
- **Solution:** Created python function called 'remove_less_informative_words' and applied the function on the 'description_lemmatized_with_no_stop_words' column and added the result to the 'description_lemmatized_with_no_stop_words_no_less_informative_words' column.

```
def remove_less_informative_words(tokens):
    return [token for token in tokens if len(token) > 1 and token not in less_informative_words]

df1['description_lemmatized_with_no_stop_words_no_less_informative_words'] =
df1['description_lemmatized_with_no_stop_words'].apply(remove_less_informative_words)

# Checking if single-letter tokens still exist
single_letter_tokens = []
for tokens in df1['description_lemmatized_with_no_stop_words_no_less_informative_words']:
    for token in tokens:
        if len(token) == 1:
            single_letter_tokens.append(token)

if single_letter_tokens:
    print("Single letter tokens still exist:", Counter(single_letter_tokens))
else:
    print("No single-letter tokens found")
```

Created new dataframe called df1_cleaned by keeping only three columns.

```
df1_cleaned =
df1[['description', 'description_lemmatized_with_no_stop_words_no_less_informative_words', 'assignment_group']].copy()
df1_cleaned.head()
```

Output:

	description	description_lemmatized_with_no_stop_words_no_less_informative_words	assignment_group
0	login issue -verified user details.(employee# ...	[login, issue, verify, detail, employee, manag...	GRP_0
1	outlook_x000D_\n_x000D_\nreceived from: hmjdr...	[outlook, team, meeting, skype, meeting, appea...	GRP_0
2	cant log in to vpn_x000D_\n_x000D_\nreceived ...	[log, vpn, log, vpn, good]	GRP_0
3	unable to access hr_tool page	[unable, access, hr, tool, page]	GRP_0
4	skype error	[skype, error]	GRP_0

Spelling Check

There are total of 7952 unique words in the ticket corpus, To efficiently perform spell-check we can exclude English words and focusing only on non-English words.

As Misspelled words can lead to incorrect interpretation of information, we need to ensure that text is free from spelling errors which helps maintain the integrity and accuracy of the content.

1. First, we will identify English words from ticket corpus using nltk word english word list.
2. then we will filter Non-English words from those list of 7,952 unique words.
3. With the filtered list, we will perform spell-check only on the non-English words. This will reduce the computational load and speed up the process.

```
spell = SpellChecker()

corrected_words = {}
for word in tqdm(non_english_words_df['word']):
    corrected_word = spell.correction(word)
    corrected_words[word] = corrected_word

non_english_words_df['corrected_word'] = non_english_words_df['word'].map(corrected_words)
non_english_words_df['corrected_status'] = non_english_words_df.apply(lambda row: 'not changed' if row['word'] == row['corrected_word'] else ('changed' if row['corrected_word'] else 'undefined'), axis=1)
```

- After performing spell check, 1,687 words were corrected, 246 had no corrections found, and spell check couldn't identify relevant words for 3,357, so they are listed as None.
- These words were not modified during spell check, but we can perform lemmatization when adding them back to the list.
- The changed words are mostly domain-specific, so the spell checker did not work well in this case. We can remove words with low frequency (≤ 2) to reduce noise.
- Similarly, from the undefined words, we can remove those with low frequency (≤ 3) to reduce noise from the data.

Overall, Spell check couldn't help much on this dataset and not effective.

Problem of Repeated Words in Each Ticket

There are some repeated words in each ticket. e.g.: [log, vpn, log, vpn, good], these words appeared twice in a single ticket description. Repeated words can create noise, also they can artificially inflate the importance of certain words in the document, leading to an imbalance in the term frequency, which might bias the model.

So those words were removed and kept the words of single occurrences of them in each ticket.

```
def find_repeated_words(text):
    word_counts = Counter(text.split())
    repeated_words = [word for word, count in word_counts.items() if count > 1]
    return repeated_words

df1_final['repeated_words'] = df1_final['description'].apply(find_repeated_words)

df1_final[df1_final['repeated_words'].apply(lambda x: len(x) > 0)]

def remove_repeated_words(text):
    words = text.split()
    unique_words = []
    for word in words:
        if word not in unique_words:
```



```

unique_words.append(word)
return ''.join(unique_words)

df1_final['description'] = df1_final['description'].apply(remove_repeated_words)

```

Dataset: Before Data Cleaning Process:

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pilcoqds	GRP_0
1	outlook	_x000D_\n_x000D_\nreceived from: hmjdrvpb.komu...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	_x000D_\n_x000D_\nreceived from: eylqgodm.ybqk...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

Dataset: After Data Cleaning Process:

	assignment_group	description
0	GRP_0	login issue verify detail employee manager che...
1	GRP_0	outlook team meeting skype appear calendar som...
2	GRP_0	log vpn good
3	GRP_0	unable access hr tool page
4	GRP_0	skype error

The final cleaned dataset contains:

Total number of words: 80298 and Number of unique words: 3804

Let's Understand Topics

TOP 20 words from all groups

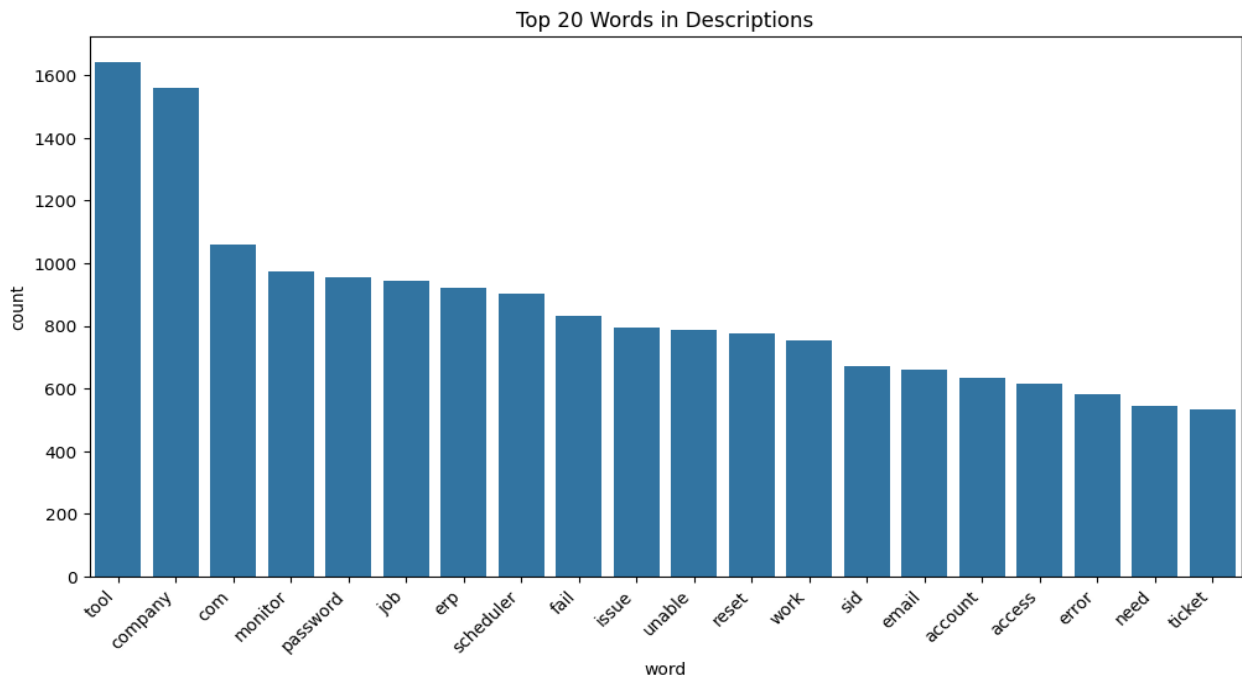


Figure 4. Top 20 Words in Descriptions

The following are the top words from each assignment group:

assignment_group	top_words
GRP_0	password,erp,unable,reset,account,issue,tool,email,outlook,sid
GRP_10	tool,company,job,com,scheduler,monitor,fail,hr,payroll,erp
GRP_12	hostname,server,access,drive,disk,file,space,available,company,folder
GRP_13	tool,order,inwarehouse,erp,customer,issue,sale,error,see,item
GRP_14	hostname,erp,server,sid,error,issue,production,system,exe,work
GRP_19	laptop,work,issue,unable,system,need,pc,connect,dear,new
GRP_2	access,sid,system,erp,need,ticket,tool,company,account,code
GRP_24	problem,setup,new,ws,ewew,computer,defective,install,tool,eu
GRP_25	tool,engineering,eu,error,system,work,issue,customer,new,problem
GRP_3	need,pc,issue,work,laptop,computer,connect,monitor,print,error
GRP_33	printer,pc,defective,work,phone,germany,long,check,error,message
GRP_4	company,network,usa,access,work,contact,sw,since,issue,outage
GRP_5	job,tool,scheduler,monitor,company,com,fail,sid,cold,abended
GRP_6	company,tool,com,job,scheduler,monitor,fail,create,abended,delivery
GRP_8	company,com,tool,job,monitor,scheduler,fail,since,backup,power
GRP_9	tool,company,job,com,scheduler,monitor,fail,report,abended,sale

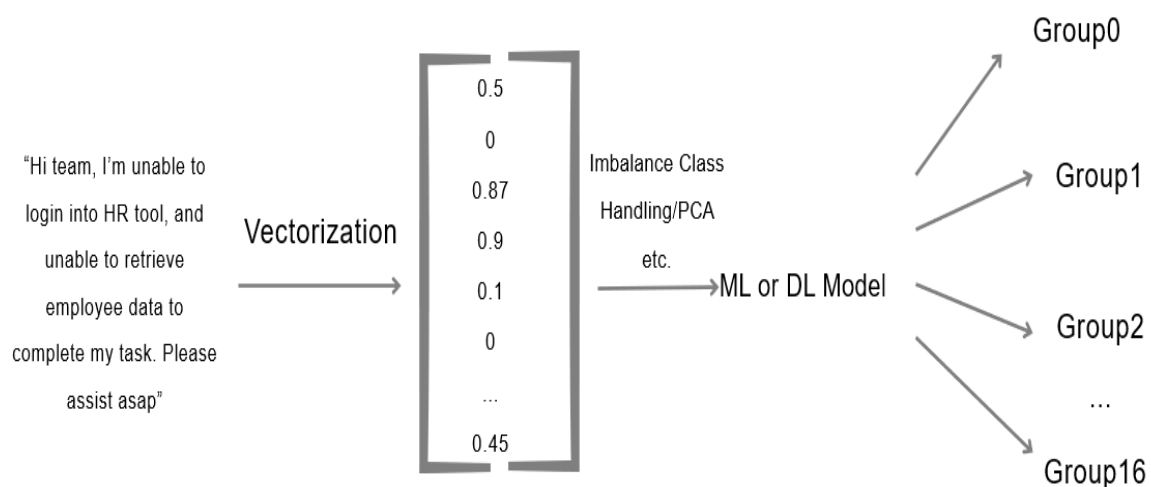
Figure 5. Top 20 Words in each assignment group

Let's create wordclouds for all 16 assignment groups. A word cloud[19] is a graphical representation where words are displayed in varying sizes based on their frequency or importance in a given text, it produces an image where the most frequent words appear larger and bolder. This visualization helps to quickly identify key themes or prominent terms in the text.

Table 2. Topics for each group

Assignment Group	Group Name Assumptions	Related Topics
GRP_0	Password and Account Management	Password management, account recovery, ERP systems, technical support
GRP_10	HR and Payroll Systems	Human resources, payroll systems
GRP_12	Server and Network Infrastructure	Network infrastructure, server management, data storage, file systems
GRP_13	Order Fulfilment and Customer Service	Order fulfilment, inventory management, customer service
GRP_14	Production and System Issues	System administration, error troubleshooting, production issues
GRP_19	Laptop and Computer Issues	Computer hardware, technical support, laptop issues, network connectivity
GRP_2	System Access and Security	Access control, system administration, ticketing systems
GRP_24	Computer Setup and Troubleshooting	Computer setup, hardware troubleshooting, installation issues, software problems
GRP_25	Engineering and System Errors	Engineering tools, system errors, customer support, problem-solving
GRP_3	Computer Hardware and Network Issues	Computer hardware, technical support, network issues, printer problems
GRP_33	Printer and Communication Issues	Printer troubleshooting, technical support, phone systems, communication issues
GRP_4	Network Infrastructure and Access Control	Network infrastructure, access control, outage management
GRP_5	Job Scheduling and Monitoring	Job scheduling, monitoring tools, system failures
GRP_6	Delivery and Scheduling Issues	Job scheduling, monitoring tools, system failures, delivery processes
GRP_8	Backup and Power Issues	Network infrastructure, backup systems, power outages, system failures
GRP_9	Sales and Reporting	Job scheduling, monitoring, sales reporting

The NLP Pipeline as follows:



Feature Transformation & Initial Model Building

Bag of Words (BoW)

The Bag of Words model is a simple method for text representation in NLP tasks. It is a process of converting a text into a numerical format that machine learning models can understand.

TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF is an extension of the BoW model that adjusts word frequencies based on their importance. It calculates the importance of a word by considering both how often it appears in a particular document (Term Frequency) and how rare it is across a collection of documents (Inverse Document Frequency). The idea is to reduce the weight of common words like "the" and "is," which appear frequently across many documents, and increase the weight of more unique words that might carry more meaningful information. This helps improve the quality of the text representation by focusing on more informative words.

- Performed TF- IDF on the descriptions data.

Handling Imbalance Dataset

As our data is highly imbalanced where over 50% of the data is from group-0 assignment group, this will be treated as majority class and rest of them as minority class, RandomOverSample technique was used to deal with this problem.

RandomOverSampler is a technique used to address class imbalance in machine learning datasets. It works by randomly duplicating examples from the minority class until the class distribution becomes more balanced.

This helps prevent the model from being biased towards the majority class and improves overall performance.

Train, Validation and Test Datasets

When working with ticket data, it's essential to divide the data into three distinct sets:

1. Train Dataset: This dataset is used to train the machine learning model.
2. Validation Dataset: This dataset is used to evaluate the model's performance during the training process.
3. Test Dataset: This dataset is used to evaluate the model's final performance after it has been trained and tuned.

Why is this division important?

- **Preventing overfitting:** Overfitting occurs when a model becomes too specialized to the training data and performs poorly on new data. By using a separate validation set, we can identify and address overfitting.
- **Evaluating generalization:** The test dataset provides an unbiased evaluation of the model's ability to generalize to new data, which is crucial for real-world applications.

Traditional Models with TF- IDF

Multinomial Naive Bayes: A probabilistic classifier based on Bayes' theorem. It assumes that features are independent given the class. It is suitable for text classification and other tasks where features are discrete.

OneVsOne Logistic Regression: A binary classifier that trains a separate logistic regression model for each pair of classes. Its predictions are made by majority vote among the individual binary classifiers. It is very effective for multi-class classification problems.

Random Forest Classifier: An ensemble learning method that combines multiple decision trees. Each decision tree is trained on a random subset of the data and features. It reduces overfitting and improves generalization performance.

These models were chosen as they are well-established and have been successfully applied in various natural language processing tasks.

The metrics used to assess performance are as follows:

- **Accuracy:** The overall proportion of correct predictions.
- **Precision:** The proportion of positive predictions that were actually correct.
- **Recall:** The proportion of actual positive cases that were correctly predicted.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of performance.
- **Training Time:** The time taken to train each model.

X: Input features (short description, description)

y: Target variable (assignment group)

Scores are as follows:

	Model	Accuracy	Precision	Recall	F1_Score	Training Time(in seconds)
0	MultinomialNB	0.664308	0.774553	0.664308	0.688984	1.547835
1	OneVsOne_LogisticRegression	0.721698	0.777125	0.721698	0.734628	147.889016
2	RandomForestClassifier	0.732704	0.735899	0.732704	0.711346	121.481577

	Model	Accuracy	Precision	Recall	F1_Score	Training Time(in seconds)
0	SVC_OVO	0.713836	0.70889	0.713836	0.679455	965.138031

Figure 7. Comparison of Scores of different ML Models.

Since the SVC (Support Vector Classifier – part of Support Vector Machines) is taking a longer time (more than 30 minutes) to run with the existing features, dimensionality reduction technique called PCA (Principal Component Analysis) was performed to reduce the number of features while retaining the important ones that can explain 95% of the variance in the data. As SVC is primarily suitable for binary classification, we

used the OVO (One vs One) model on top of it to handle the multiclass problem more effectively. The Results as follows:

Feature count reduces from 3804 to 913 after performing PCA(these features can explain 95% of the variance in the data)

feature count after performing PCA (45312, 913)

Key Findings

- **Accuracy:** The Random Forest Classifier achieved the highest accuracy of 0.732704.
- **Precision:** The OneVsOne Logistic Regression and Multinomial Naive Bayes models had similar precision scores, with the Random Forest Classifier slightly lower.
- **Recall:** All models achieved the similar recall score.
- **F1-Score:** The OneVsOne Logistic Regression model had the highest F1-score of 0.734628, indicating a good balance between precision and recall.
- **Training Time:** The Multinomial Naive Bayes model was the fastest to train, followed by the Random Forest Classifier. The OneVsOne Logistic Regression model took significantly longer to train.
- **SVC with PCA:** The training time of SVC was reduced after applying the PCA technique to the data. It also produced results similar to other models, with fewer features being trained on, and reduced the training time. However, the training time is still relatively high (~17 minutes) compared to other models.

Based on the evaluation metrics[Figure 7], the OneVsOne Logistic Regression model appears performing good so far, It offers a good balance between accuracy, precision, recall, and F1-score.

Hypothesis Testing on Text Length Column

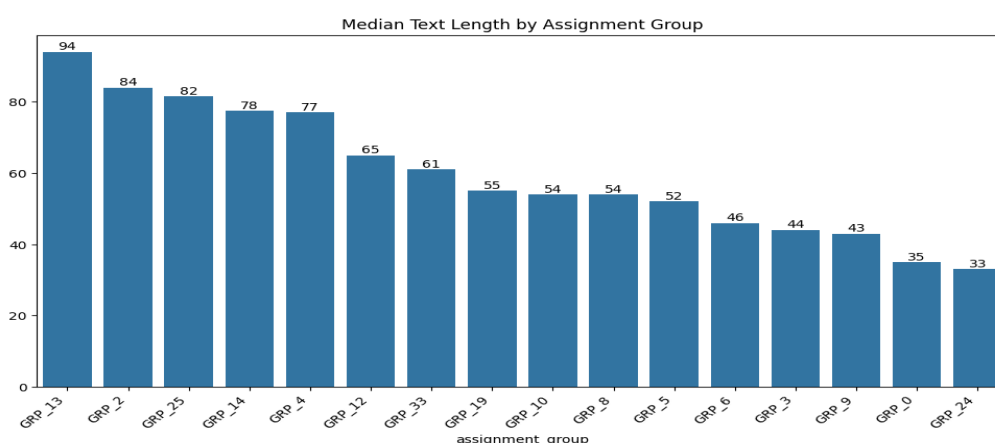


Figure 8. Median Text Length by Assignment Group

As shown in the figure 8, we can observe that the mean and median text lengths vary among the assignment groups. To verify if this variation is statistically significant, we can perform hypothesis testing on the text length data. If the difference is found to be significant, we can consider including this as an additional feature in our model.

Null Hypothesis (H_0):

The mean text length is the same across all assignment groups. There is no statistically significant difference in the mean text lengths.

Alternative Hypothesis (H_a):

At least one assignment group has a mean text length that is significantly different from the others. This implies that there is a statistically significant variation in mean text lengths among the assignment groups

F-scores:

```
F-statistic: 42.61793276910973
P-value: 7.885097740072651e-121
There is a significant difference in text length between assignment groups.
```

Plot: Tukey's HSD Post-Hoc Test:

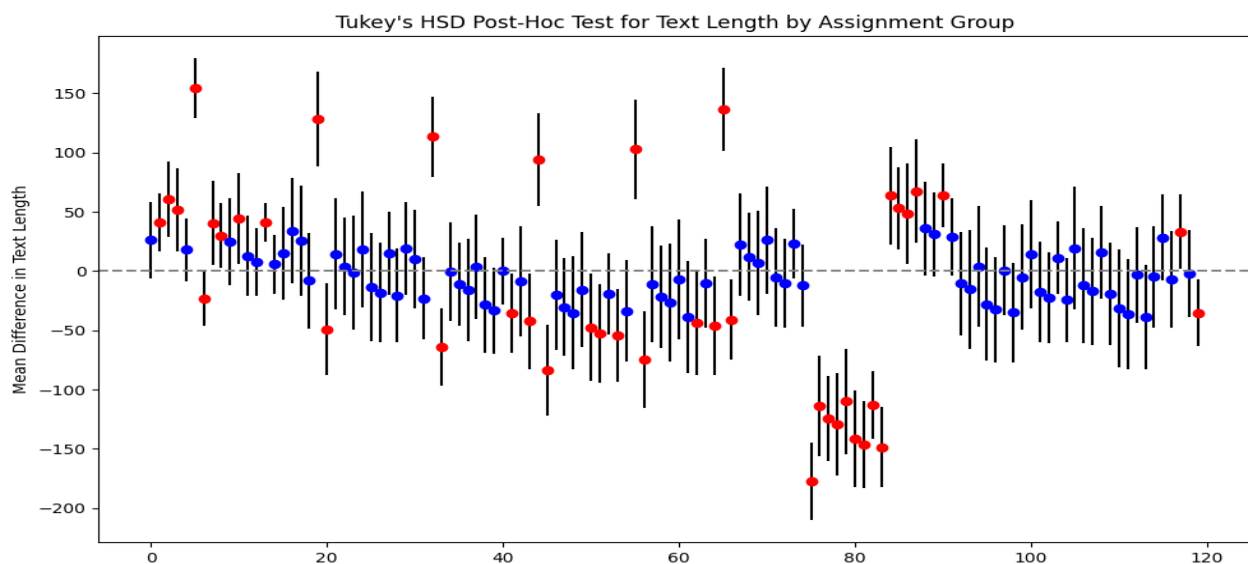


Figure 9. Tukey's HSD Post-Hoc Test for Text Length by Assignment Group

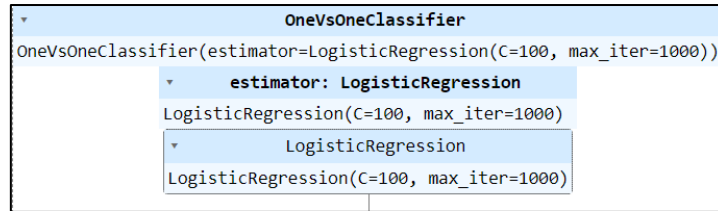
There appear to be several significant differences in text length between different assignment groups, as evidenced by the numerous blue dots[Figure 9].

Observation: It seems that the text length across the assignment groups is not uniformly distributed. This is suggested by the observed by **one-way ANOVA test/ F-test** and differences in error bars, indicating that the variability in text length between groups is statistically significant. Therefore, we can consider incorporating text length as a potential feature in our model to account for this variation.

Construct Stage in PACE Framework: More on Logistic Regression

Hyperparameter Tuning

As we have achieved good results on the Logistic Regression earlier, we will perform Hyperparameter tuning to optimize the performance of the model. Proper tuning helps avoid underfitting or overfitting, improving accuracy and efficiency.



	Model	Accuracy	Precision	Recall	F1_Score
0	OneVsOne_LogisticRegression_Tuned+ROS	0.72956	0.749118	0.72956	0.727823

To address memory constraints, we employed Principal Component Analysis (PCA) on the dataset prior to conducting grid search cross-validation for logistic regression. While fine-tuning did yield some improvement, the overall accuracy gain was marginal.

Adding Text Length as New Feature on Tuned Model.

We scaled the text length column to ensure numerical stability and prevent potential issues during model training and evaluation. By scaling the text length column, we ensured that all values are within a similar range, which can help the model converge more quickly and accurately.

	Model	Accuracy	Precision	Recall	F1_Score
	OneVsOne_LogisticRegression_Tuned+New_Feature+ROS	0.743711	0.764584	0.743711	0.742221

After fine-tuning the model's parameters and incorporating the additional feature of text length(scaled), we observed a significant improvement in all key metrics (precision, recall, F1-score, accuracy) of the OVO-logistic regression model.

Handling Class Imbalance using SMOTE Technique

So far, we have used RandomOverSampling to address the class imbalance issue. Now, we are utilizing the SMOTE[20] (Synthetic Minority Over-sampling Technique) technique. Unlike simple oversampling, where instances of the minority class are duplicated, SMOTE creates synthetic data points for the minority class. This is done by generating new, synthetic samples based on the feature space of existing instances, which helps the model learn from the minority class without overfitting to the majority class.

	Model	Accuracy	Precision	Recall	F1_Score
	OneVsOne_LogisticRegression_Tuned+New_Feature+...	0.75	0.767796	0.75	0.748193

After applying the SMOTE technique, we observed a good improvement across all key metrics.

Adding Augmented Data to address Class Imbalance

Using NLPAug Package

SMOTE works best with numerical data, and since we are dealing with text data, we will use another technique called text augmentation using the nlpaug package, specifically synonym augmentation with an augmentation factor of 3. Thus, nlpaug will generate 3 additional examples for each row (sample) across all

classes by replacing words in a sentence with their synonyms, thus generating variations of the original text. This will help create more samples for the minority class, leading to improved model performance.

Model	Accuracy	Precision	Recall	F1_Score
OneVsOne_LogisticRegression+New_Feature+NLPAug	0.772799	0.763446	0.772799	0.759359

NLP augmentation outperformed SMOTE, particularly on unseen data, improving the model's performance.

Performance on test data: Accuracy 0.7652050919377652, Precision 0.8156244905582924, Recall 0.7652050919377652, F1-Score 0.7818480267125477

BERT based Augmentation:

A BERT-based data augmentation approach was applied using the Bert Tokenizer and TFBertForMaskedLM from the Hugging face Transformers library to generate additional samples for minority classes. This method involves masking certain words in a given sample and predicting replacements using the pre-trained BERT language model. By doing this, the dataset was augmented to increase the representation of minority classes by a factor of 3.

Model	OneVsOne_LogisticRegression+New_Feature+BERT_Aug
Accuracy	0.761006
Precision	0.748762
Recall	0.761006
F1_Score	0.744705

Performance on test data: Accuracy 0.7581329561527581, Precision 0.7852512683792344, Recall 0.7581329561527581, F1-Score 0.766824696353548

BERT-based augmentation yielded lower results compared to NLP augmentation.

Deep Learning Models[Birectional LSTM] with Embeddings

Deep Learning models automatically learn the most relevant features from raw text and can model complex, non-linear relationships in data. Recurrent Neural Networks (RNNs) can extract hierarchical and high-level features without needing manual intervention. Neural networks RNNs (like LSTMs and GRUs) are adept at handling sequential data and can learn long-term dependencies in text.

Word Embeddings:

While TF-IDF is a valuable technique, it has limitations. It does not capture sentence context and structure and also its high-dimensional, sparse data representations, which can significantly increase computational costs. To overcome these limitations of TFIDF, more sophisticated techniques are employed to transform text into word embeddings. These techniques effectively capture the semantic meaning of words, reduce dimensionality, and better handle syntactic relationships.

Semantics vs. Syntax

- Semantics: Refers to the meaning or content of a sentence.
- Syntax: Relates to the grammatical structure or arrangement of words in a sentence.

Popular Pre-trained Word Embedding Models:

Word2Vec: Looks at local words around a target word and learns from predicting them. Like a word guessing game focused on immediate neighbours.

GloVe: Analyses the entire text to see how often words appear together and learns from those overall patterns. Like looking at the big picture of word relationships in the whole text.

These models have been trained on massive datasets and can be used directly in various NLP applications.

In our project we used Word2Vec and GloVe models.

Bidirectional LSTM:

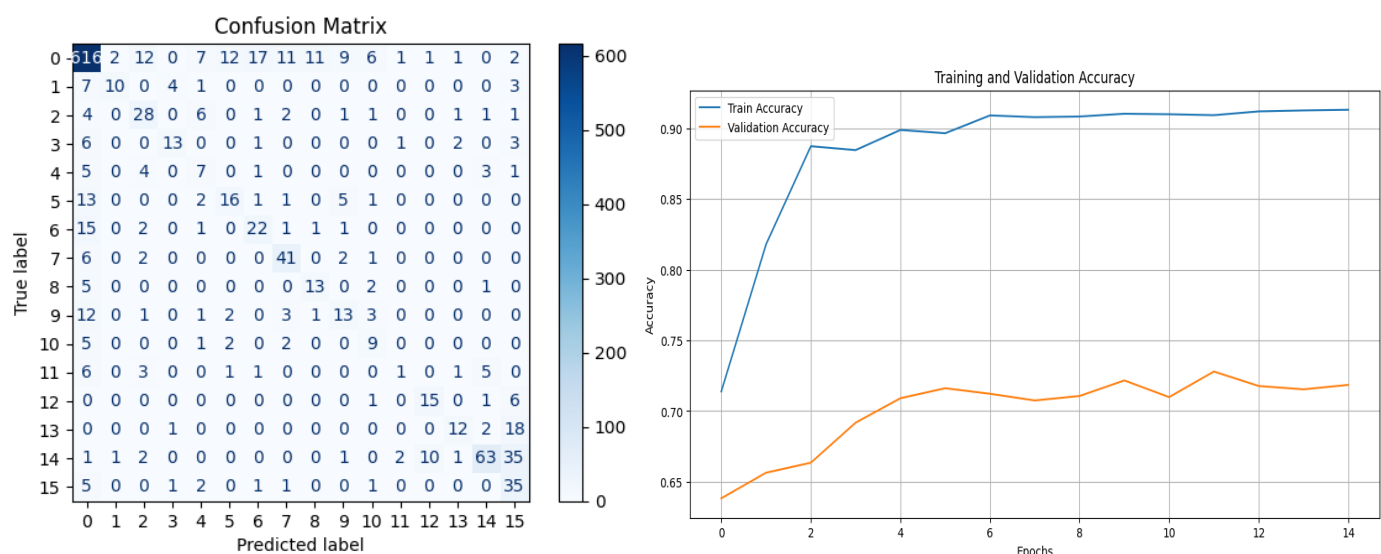
Bidirectional Long Short-Term Memory (BiLSTM) is a type of recurrent neural network (RNN) architecture that enhances the learning capability of standard LSTM networks by processing data in both forward and backward directions.

Since we're dealing with text sequences, using a Bidirectional(LSTM()) layer could improve performance by capturing dependencies in both directions.

Bidirectional LSTM with Word2Vec Embeddings and Random Oversampling

Converted text to embeddings of 300 dimension using a pre-trained[word2vec-google-news-300] Gensim Word2Vec model and performed random oversampling to address class imbalance in these embeddings. Subsequently, trained this data using a Bidirectional LSTM.

Model's Performance:



Performance on validation set:

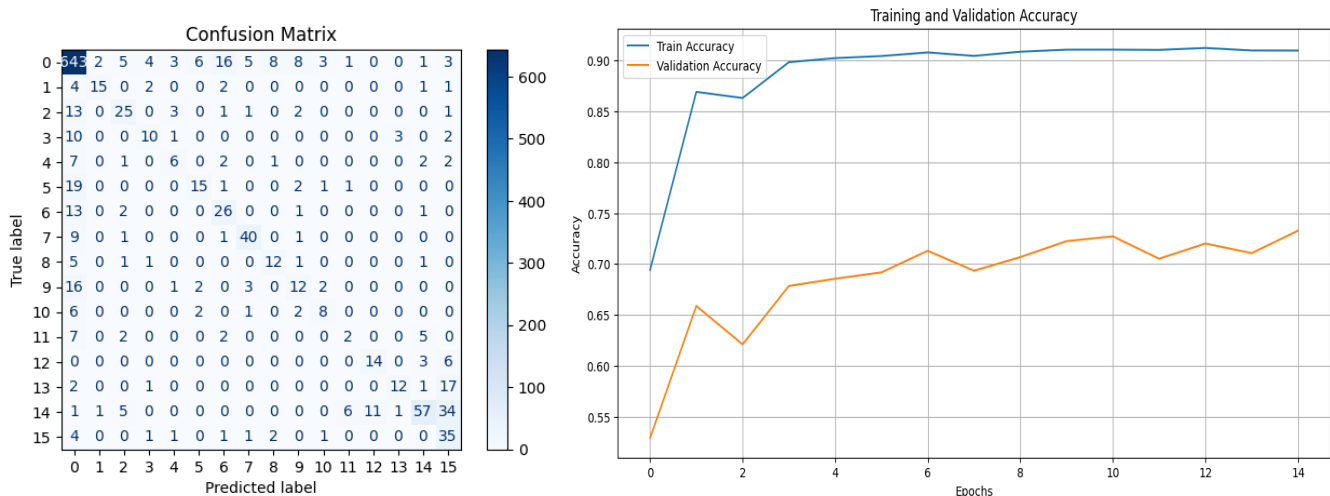
	Model	Accuracy	Precision	Recall	F1_Score
	OneVsOne_LogisticRegression_Tuned+ROS	0.729560	0.749118	0.729560	0.727823
	OneVsOne_LogisticRegression_Tuned+New_Feature+ROS	0.743711	0.764584	0.743711	0.742221
	OneVsOne_LogisticRegression_Tuned+New_Feature+...	0.750000	0.767796	0.750000	0.748193
	OneVsOne_LogisticRegression+New_Feature+NLPAug	0.772799	0.763446	0.772799	0.759359
	OneVsOne_LogisticRegression+New_Feature+BERT_Aug	0.761006	0.748762	0.761006	0.744705
	LSTM(Bi-directional)+Word2Vec(Pre-trained)+ROS	0.718553	0.739511	0.718553	0.719317

The LSTM with pre-trained embeddings underperformed compared to traditional models, this is due to the small dataset size and overfitting on the training data.

Bidirectional LSTM with GloVe Embeddings and Random Oversampling

Converted text to embeddings of 100 dimension using a pre-trained[glove-wiki-gigaword-100] Gensim GloVe model and performed random oversampling to address class imbalance in these embeddings. Subsequently, trained this data using a Bidirectional LSTM.

Model's Performance:



Performance on validation set:

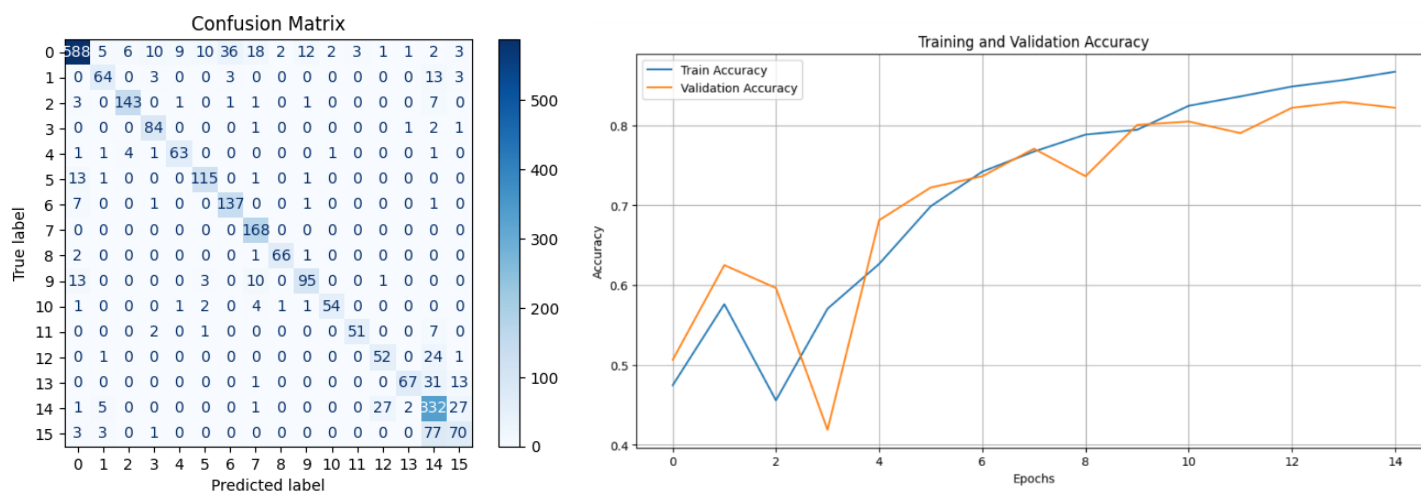
LSTM(Bi-directional)+Word2Vec(Pre-trained)+ROS	0.718553	0.739511	0.718553	0.719317
LSTM(Bi-directional)+Glove(Pre-trained)+ROS	0.732704	0.739597	0.732704	0.725685

GloVe embeddings produced similar results to Word2Vec and underperforms compared to Logistic Regression with TF-IDF.

GloVe Embeddings and BERT-Aug Data for Bidirectional LSTM

This is similar to above model but Instead of random oversampling technique, BERT-Augmented data was used for training.

Model's Performance:



Performance on validation set	Accuracy 0.8221117061973986 Precision 0.831787639021385 Recall 0.8221117061973986 F1-Score 0.8210658049450847
Performance on Test/unseen set	Accuracy 0.685997171145686 Precision 0.7054641464592714 Recall 0.685997171145686 F1-Score 0.6865107580501806

A bidirectional LSTM using GloVe embeddings trained on BERT-augmented data performed well on the training and validation sets but underperformed on the unseen test data, indicating a lack of generalization to new data.

Comparison of Models Performance

Table 3. Performance Comparison of All Models

Traditional Models + Feature Engineering	Accuracy	Precision	Recall	F1_Score
MultinomialNB	0.6643	0.7746	0.6643	0.6890
OneVsOne_LogisticRegression	0.7217	0.7771	0.7217	0.7346
RandomForestClassifier	0.7327	0.7359	0.7327	0.7113
SVC_OVO	0.7138	0.7089	0.7138	0.6795
OneVsOne_LogisticRegression_Tuned+ROS	0.7296	0.7491	0.7296	0.7278
OneVsOne_LogisticRegression_Tuned+New_Feature+ROS	0.7437	0.7646	0.7437	0.7422
OneVsOne_LogisticRegression_Tuned+New_Feature+SMOTE	0.7500	0.7678	0.7500	0.7482
OneVsOne_LogisticRegression+New_Feature+NLPAug	0.7728	0.7634	0.7728	0.7594
OneVsOne_LogisticRegression+New_Feature+BERT_Aug	0.7610	0.7488	0.7610	0.7447
Average	0.7322	0.7545	0.7322	0.7263

Deep Learning Models + Embeddings	Accuracy	Precision	Recall	F1_Score
LSTM(Bi-directional)+Word2Vec(Pre-trained)+ROS	0.7186	0.7395	0.7186	0.7193
LSTM(Bi-directional)+Glove(Pre-trained)+ROS	0.7327	0.7396	0.7327	0.7257
LSTM(Bi-directional)+Glove(Pre-trained)+BERT_Aug	0.8221	0.8318	0.8221	0.8211
Average	0.7578	0.7703	0.7578	0.7554

Scores on Test/Unseen Data: Final Evaluation

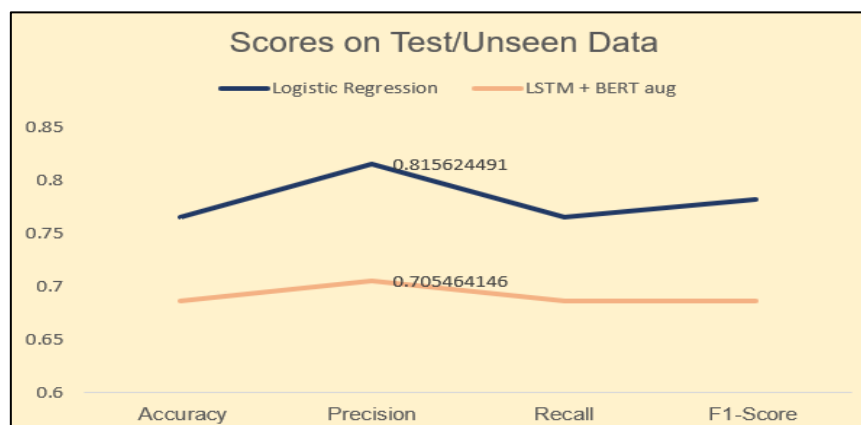


Figure 10. Scores of LR[Model 4;Table 3] and LSTM[Model 8;Table 3] on Unseen Data

Conclusion

The chosen model[OneVsOne_LogisticRegression+New_Feature+NLPAug] demonstrates better generalization to unseen data, prioritizing robustness over purely optimizing for validation accuracy. While several models were evaluated, it was observed that some achieved high accuracy on the validation set but struggled to maintain similar performance when tested on unseen data. This indicated overfitting, where the model memorized the validation data patterns without learning underlying generalizable features. To mitigate this, a model with slightly lower validation accuracy but significantly better performance on unseen test data was selected. This approach ensures the model can generalize more effectively to real-world scenarios, enhancing its predictive reliability and applicability in practical deployments. By prioritizing generalization over mere validation performance, the selected model strikes an optimal balance between complexity and robustness.

Future Scope of the Work:

While this solution is a good improvement, we can make it even better in the future. Here are some ideas:

Smarter Models:

- Using advanced AI models such as Large Language models to understand the context of tickets more deeply.
- Making sure the AI is fair and doesn't discriminate.

Privacy and Security:

- Protecting sensitive information in tickets.
- Following data privacy rules.

Execute Stage in PACE Framework: Building an Application using Streamlit

The final model was selected and deployed using Streamlit[21], which is an open-source framework designed for building interactive web applications specifically for data science and machine learning projects. It enables developers to create visually appealing and user-friendly applications with minimal effort, allowing data scientists and machine learning engineers to transform their scripts into shareable web applications effortlessly.

The Source code and other files required for the application were deployed in GitHub repository and The Git Repository was lined with Streamlit account to deploy the model.

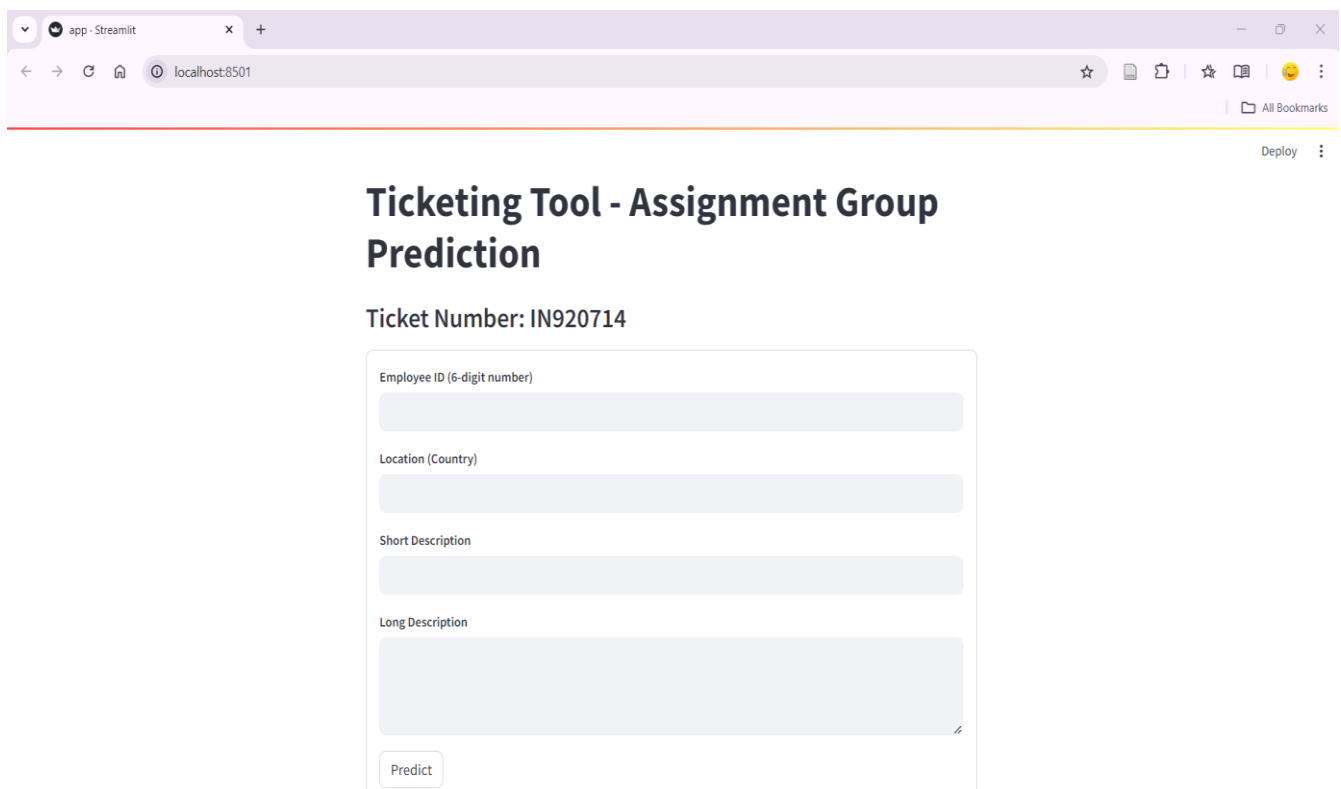
Project Files:

```
LabelEncoder.pkl
MinMaxScaler.pkl
OVO_LR_model.pkl
TfidfVectorizer.pkl
app.py
final_words_to_remove_updated.csv
requirements.txt
results_on_validation_data.csv
sample-test.ipynb
```

Application Interface and Testing

Long description Issue: “Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you”

Each time user opens this page, it will generate new ticket number.



The screenshot shows a web browser window with the address bar displaying 'localhost:8501'. The page title is 'app - Streamlit'. The main content area has a heading 'Ticketing Tool - Assignment Group Prediction' and a sub-heading 'Ticket Number: IN920714'. Below this is a form with four input fields: 'Employee ID (6-digit number)', 'Location (Country)', 'Short Description', and 'Long Description'. A 'Predict' button is located at the bottom of the form.

Figure 11. Front Page of Streamlit Application

Short + Long description Issue: "Login Issue", "Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you"

Ticketing Tool - Assignment Group Prediction

Ticket Number: IN631277

Employee ID (6-digit number)

123456

Location (Country)

India

Short Description

Login Issue

Long Description

Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you

Predict

Long Description

Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you

Predict

Predicted Assignment group for ticket IN631277: GRP_0

Ticket Number	Employee ID	Location	Short Description	Long Description	Predicted Assignment Group
0 IN631277	123456	India	Login Issue	Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you	GRP_0

Figure 12. Testing the application with an example prediction

It was predicted as Group_0, which is correct.

Constraints and Data Validation

The user needs to provide all the required fields, including a six-digit employee ID. Failing to provide these details will result in an error.

Ticketing Tool - Assignment Group Prediction

Ticket Number: IN631277

Employee ID (6-digit number)

123456

Location (Country)

India

Short Description

Login Issue

Long Description

Predict

Please fill all the required fields.

Ticketing Tool - Assignment Group Prediction

Ticket Number: IN631277

Employee ID (6-digit number)

1234

Location (Country)

India

Short Description

Login Issue

Long Description

Hello Team, Engineering tool says not connected and I'm unable to submit reports, Can anyone please help on this issue. Thank you

Predict

Please enter a valid 6-digit numerical Employee ID.

Figure 13. Testing the Constraints and Data Validation

References

- [1] A. Revina, K. Buza, and V. G. Meister, "IT Ticket Classification: The Simpler, the Better," *IEEE Access*, vol. 8, pp. 193380–193395, 2020, doi: 10.1109/ACCESS.2020.3032840.
- [2] S. Fuchs, C. Drieschner, and H. Wittges, *Improving Support Ticket Systems Using Machine Learning: A Literature Review*. 2022. Accessed: Oct. 05, 2024. [Online]. Available: <http://hdl.handle.net/10125/79570>
- [3] P. Bafna, D. Pramod, and A. Vaidya, "Document clustering: TF-IDF approach," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Mar. 2016, pp. 61–66. doi: 10.1109/ICEEOT.2016.7754750.
- [4] B. Mahesh, *Machine Learning Algorithms -A Review*, vol. 9. 2019. doi: 10.21275/ART20203995.
- [5] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Comput. Stat.*, vol. 2, no. 4, pp. 433–459, 2010, doi: 10.1002/wics.101.
- [6] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," arXiv.org. Accessed: Oct. 05, 2024. [Online]. Available: <https://arxiv.org/abs/1508.01991v1>
- [7] K. W. Church, "Word2Vec," *Nat. Lang. Eng.*, vol. 23, no. 1, pp. 155–162, Jan. 2017, doi: 10.1017/S1351324916000334.
- [8] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
- [9] M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding," arXiv.org. Accessed: Oct. 05, 2024. [Online]. Available: <https://arxiv.org/abs/2103.11943v1>
- [10] J. Iden and T. R. Eikebrokk, "Implementing IT Service Management: A systematic literature review," *Int. J. Inf. Manag.*, vol. 33, no. 3, pp. 512–523, Jun. 2013, doi: 10.1016/j.ijinfomgt.2013.01.004.
- [11] "Turn Data into Decisions with the PACE Framework," <https://www.michelebedin.com/en/>. Accessed: Oct. 05, 2024. [Online]. Available: <https://www.michelebedin.com/en/turn-data-into-decisions-with-pace-framework/>
- [12] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Mach. Learn. PYTHON*.
- [13] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep Learning With TensorFlow: A Review," *J. Educ. Behav. Stat.*, vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.
- [14] J. Praveen Gujjar, H. R. Prasanna Kumar, and M. S. Guru Prasad, "Advanced NLP Framework for Text Processing," in *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, Mar. 2023, pp. 1–3. doi: 10.1109/ISCON57294.2023.10112058.
- [15] Siva, "Automatic Ticket Assignment using NLP." 2020. [IEEE]. Available: <https://www.kaggle.com/aviskumar/automatic-ticket-assignment-using-nlp>
- [16] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text Classification Algorithms: A Survey," *Information*, vol. 10, no. 4, Art. no. 4, Apr. 2019, doi: 10.3390/info10040150.
- [17] J. E. F. Friedl, *Mastering Regular Expressions*. O'Reilly Media, Inc., 2006.
- [18] J. Plisson, N. Lavrac, and D. Mladenec, "A Rule based Approach to Word Lemmatization".
- [19] F. Heimerl, S. Lohmann, S. Lange, and T. Ertl, "Word Cloud Explorer: Text Analytics Based on Word Clouds," in *2014 47th Hawaii International Conference on System Sciences*, Jan. 2014, pp. 1833–1842. doi: 10.1109/HICSS.2014.231.
- [20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.
- [21] *streamlit/streamlit*. (Oct. 05, 2024). Python. Streamlit. Accessed: Oct. 05, 2024. [Online]. Available: <https://github.com/streamlit/streamlit>

Appendices

Appendix 1: Automatic Ticket Classification Dataset

This appendix provides a description of the Automatic Ticket Classification Dataset, which was used in this study. The dataset is available from Kaggle (<https://www.kaggle.com/datasets/aviskumar/automatic-ticket-assignment-using-nlp/data>).

The dataset contains the requests/issues of customers reported to IT team. The tickets are labeled with the assignment group that is responsible for resolving the complaint.

Appendix 2: Project Code

This appendix provides a link to the code repository for the project described in this dissertation. The code utilizes Python Open Source libraries and implements the methodologies discussed in PACE Framework.

Project Code: https://github.com/swarna987456/MTech_Final_Project.git

Appendix 3: External Certifications from Google:

- Python: <https://coursera.org/share/b7bde199ee42091fe80fdaecb4a31bc9>
- Google Advance Data analytics: <https://coursera.org/share/e1563fd2967b3a70d656b6f69e3a7c77>
- Google GenAI courses: https://partner.cloudskillsboost.google/public_profiles/85cfd38a-c578-45a4-992e-9dc1a21f9c3d

Checklist of items for final report (with Yes or No marked, as applicable)

- a) Is the Cover page in proper format? Y / N
- b) Is the Title page in proper format? Y / N
- c) Is the Certificate from the Supervisor in proper format? Has it been signed? Y / N
- d) Is Abstract included in the Report? Is it properly written? Y / N
- e) Does the Table of Contents' page include chapter page numbers? Y / N
- f) Is Introduction included in the report? Is it properly written? Y/N
- g) Are the Pages numbered properly? Y / N
- h) Are the Figures numbered properly? Y/ N
- i) Are the Tables numbered properly? Y / N
- j) Are the Captions for the Figures and Tables proper? Y / N
- k) Are the Appendices numbered? Y / N
- l) Does the Report have Conclusions/ Recommendations of the work? Y / N
- m) Are References/ Bibliography given in the Report? Y / N
- n) Have the References been cited in the Report? Y / N
- o) Is the citation of References/ Bibliography in proper format? Y / N