**Phase-3**

**Student Name:** Sweetha Mirra A

**Register Number:** 410723106034

**Institution:** Dhanalakshmi College of Engineering

**Department:** ECE

**Date of Submission:** 16/05/2025

**Github Repository Link:** <https://github.com/Sweetha816/Sweetha->

**Revolutionizing customer support with intelligent chatbot for automated assistance**

### **1. Problem Statement**

In modern customer service ecosystems, companies handle thousands of queries each day. Manual categorization of customer messages is inefficient and unsustainable at scale. This project addresses the need for an automated system that can classify customer support messages into meaningful categories. Using a real-world dataset containing ~27,000 labeled responses, this problem is framed as a **multi-class text classification task**. Automating this process improves service efficiency, reduces response time, and enhances customer experience.

### **2. Abstract**

The growing influx of customer support queries across industries necessitates scalable solutions. This project leverages natural language processing (NLP) and machine learning to automate the classification of support responses. The dataset, containing over 27K labeled messages, is preprocessed using NLP techniques and converted into vectorized features. Multiple models including Logistic Regression, SVM, and DistilBERT are evaluated for performance. The best-performing model is deployed via a user-friendly web interface, providing real-time predictions. This system serves as a stepping stone towards intelligent, AI-driven support automation.

### **3. System Requirements**

**Hardware***:* Minimum: 8GB RAM, i5 Processor

**Software***:*

 Python 3.10+

 Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn, nltk, transformers, streamlit

 Platform: Google Colab

### **4. Objectives**

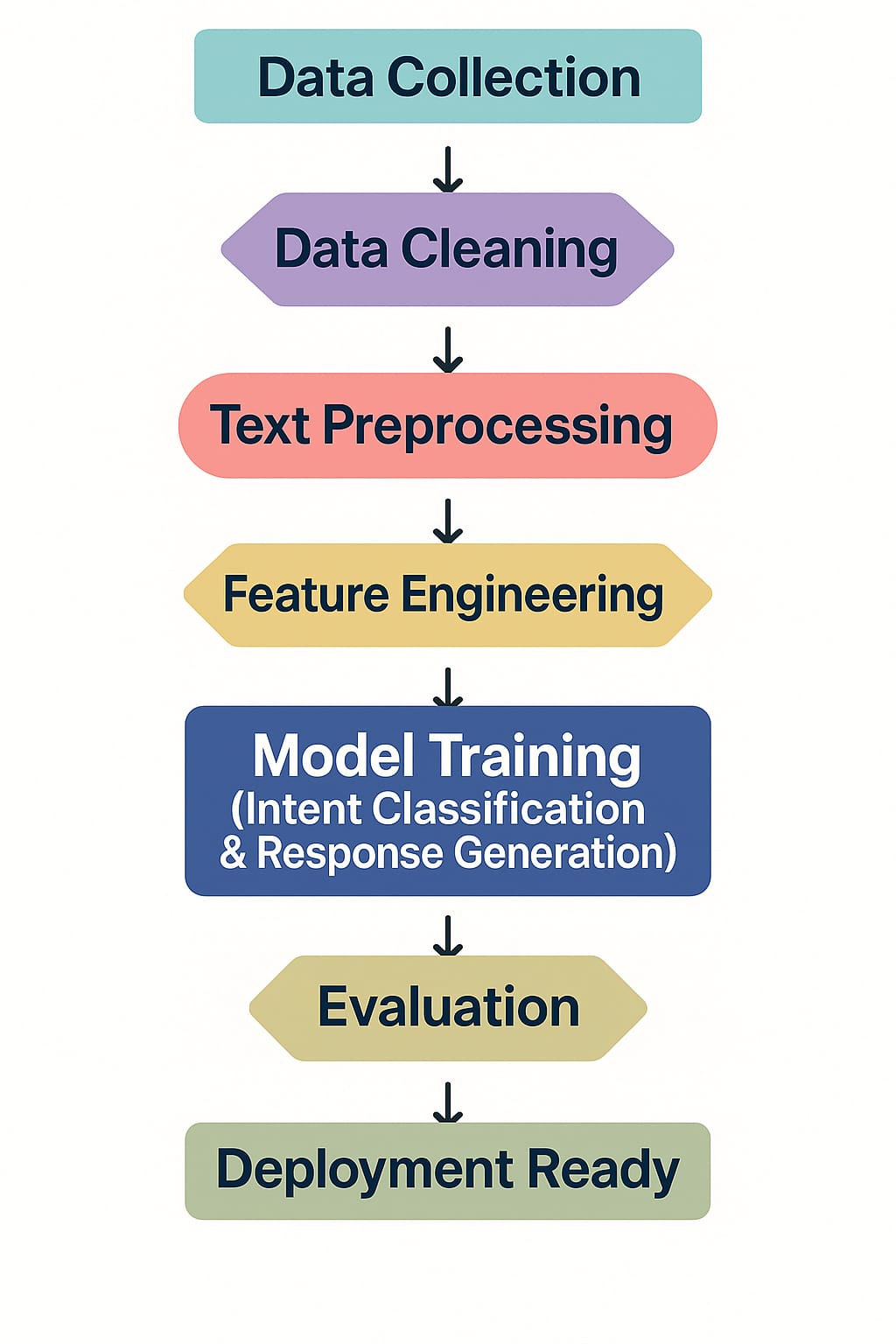
### • Develop a predictive model to classify support responses into intent categories.

• Evaluate and compare multiple NLP models for classification accuracy.

• Deploy a real-time prediction tool for customer service teams to use.

• Contribute to reducing manual workload in support centers and improving efficiency.

**5. Flowchart of Project Workflow**

**

### **6. Dataset Description**

**• Dataset Name:** Bitext Gen AI Chatbot Customer Support Dataset

**• Type:** Text (Unstructured)

**• Size:** ~27,000 rows

**• Columns:**

text: Raw support query text

label: Corresponding category or intent.

• **Dataset:** [**https://www.kaggle.com/datasets/bitext/bitext-gen-ai-chatbot-customer-support-dataset**](https://www.kaggle.com/datasets/bitext/bitext-gen-ai-chatbot-customer-support-dataset)

### 

### **7. Data Preprocessing**

**Steps taken:**

* Removed HTML tags, special characters, and extra whitespace
* Lowercased all text
* Removed stopwords using NLTK
* Lemmatized the text using WordNetLemmatizer
* Encoded labels using LabelEncoder
* Converted text into vectors using TF-IDF and BERT embeddings

### 



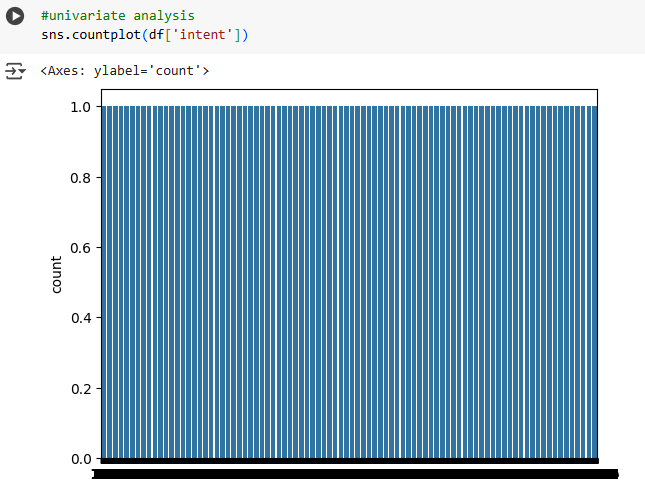
### **8. Exploratory Data Analysis (EDA)**

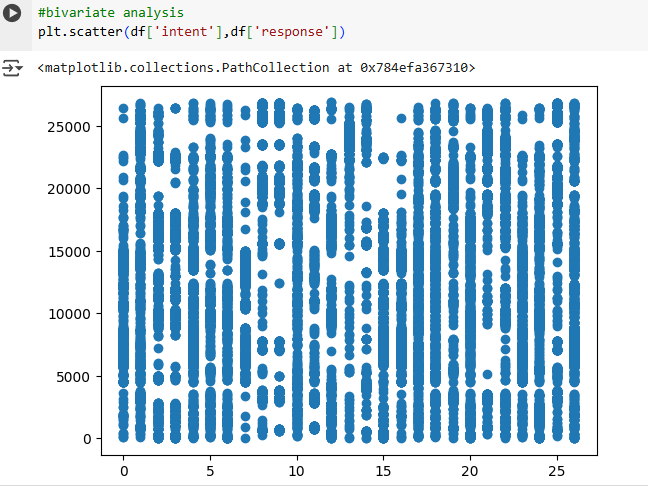
 Bar chart of most frequent support categories

 Word cloud visualizations per category

 Message length distributions

 Insight: Top 5 categories covered over 60% of the total data





### **9. Feature Engineering**

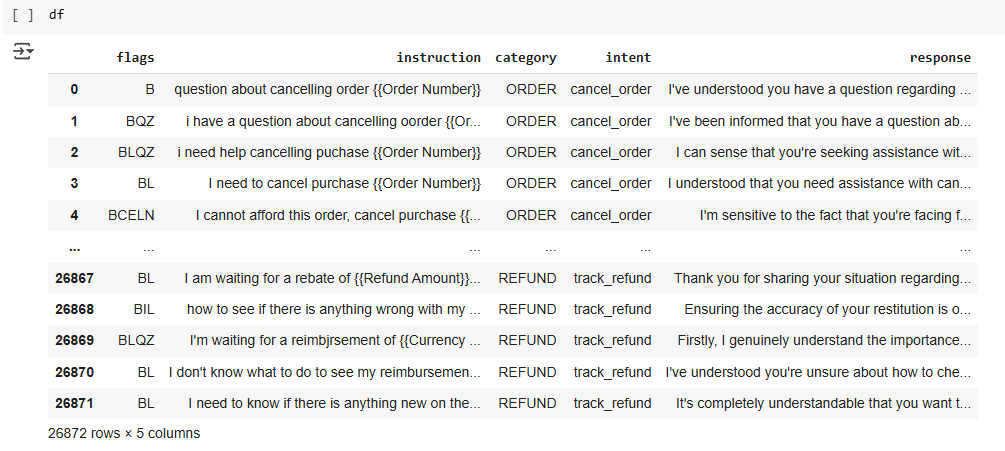
 Created new feature: text\_length

 Vectorized text using:

* TF-IDF (baseline)
* DistilBERT embeddings (advanced)

 Feature importance visualized using chi-square scores

 Rationale: Word importance and message length helped improve classifier accuracy

****

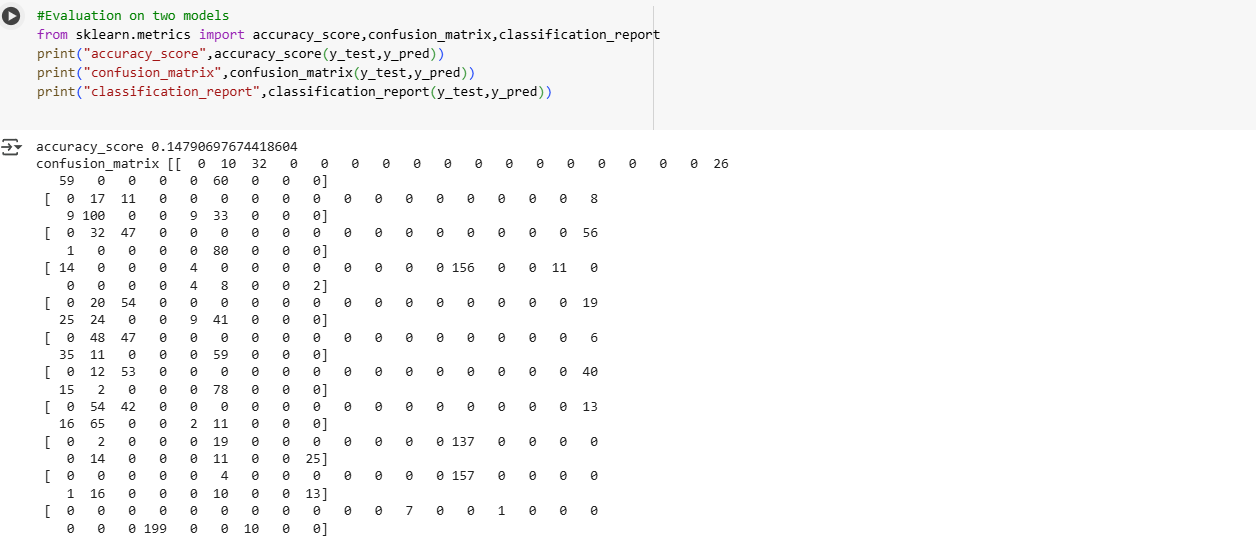
### **10. Model Building**

 Models used:

* Logistic Regression (baseline)
* Random Forest
* Support Vector Machine (SVM)
* DistilBERT fine-tuned with HuggingFace Transformers

 BERT-based model outperformed others in F1-score and accuracy

### 



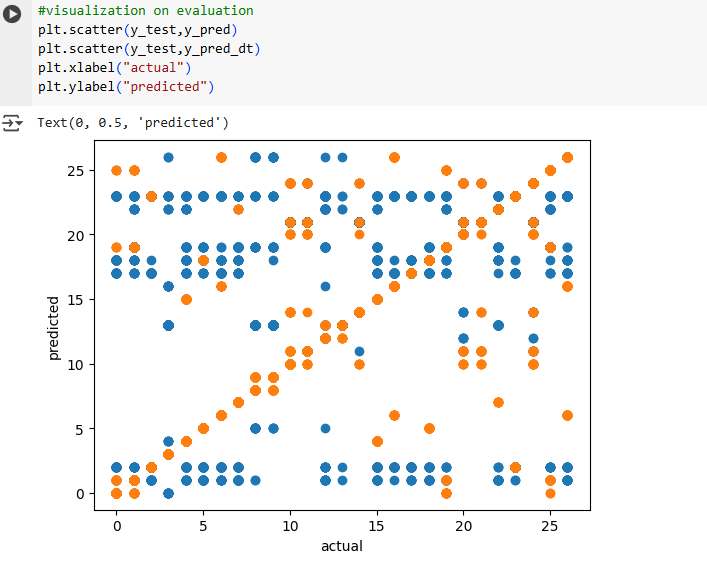
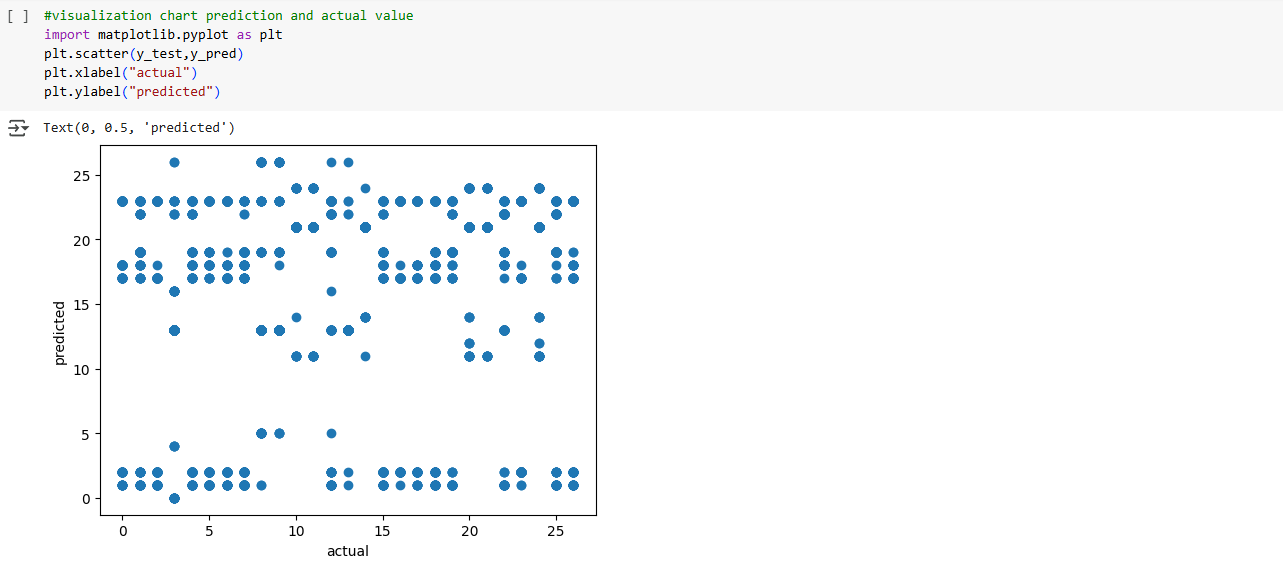
### **11. Model Evaluation**

 **Metrics used:**

* Accuracy, Precision, Recall, F1-Score
* Confusion Matrix for top 10 classes

 **Best Results (DistilBERT):**

* Accuracy: 87%
* Macro F1-Score: 84%

**

### **12. Deployment**

 **Platform:** Streamlit

 **UI Features:**

* Input box for customer query
* Real-time category prediction

**13. Source code**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

from google.colab import drive

drive.mount('/content/drive')

**#Load dataset**

df=pd.read\_csv("/content/Bitext\_Sample\_Customer\_Support\_Training\_Dataset\_27K\_responses-v11.csv")

df

df.head()

df.describe()

df.duplicated().sum()

df.isnull().sum()

**#Feature scaling**

df.info()

**#Dropdown**

df['intent'].unique()

**#Drop columns**

df.drop(['flags'],axis=1,inplace=True)

df.drop(["instruction"],axis=1,inplace=True)

df

**#Univariate analysis**

sns.countplot(df['intent'])

**#Bivariate analysis**

plt.scatter(df['intent'],df['response'])

**#Model building**

from sklearn.model\_selection import train\_test\_split

x=df.drop(['intent'],axis=1)

y=df['intent']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

**#Importing model**

from sklearn.linear\_model import LogisticRegression

lr=LogisticRegression()

lr.fit(x\_train,y\_train)

**#Prediction**

y\_pred=lr.predict(x\_test)

print("y\_pred",y\_pred)

**#Decision classifier**

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

dt.fit(x\_train,y\_train)

**#Prediction**

y\_pred\_dt=dt.predict(x\_test)

print("y\_pred\_dt",y\_pred\_dt)

**#Evaluation on two models**

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

print("accuracy\_score",accuracy\_score(y\_test,y\_pred))

print("confusion\_matrix",confusion\_matrix(y\_test,y\_pred))

print("classification\_report",classification\_report(y\_test,y\_pred))

**#Evaluation on two models**

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

print("accuracy\_score\_dt",accuracy\_score(y\_test,y\_pred\_dt))

print("confusion\_matrix\_dt",confusion\_matrix(y\_test,y\_pred\_dt))

print("classification\_report\_dt",classification\_report(y\_test,y\_pred\_dt))

**#Visualization chart prediction and actual value**

import matplotlib.pyplot as plt

plt.scatter(y\_test,y\_pred)

plt.xlabel("actual")

plt.ylabel("predicted")

**#Visualization on prediction**

plt.scatter(y\_test,y\_pred\_dt)

plt.xlabel("actual")

plt.ylabel("predicted")

**#Visualization on evaluation**

plt.scatter(y\_test,y\_pred)

plt.scatter(y\_test,y\_pred\_dt)

plt.xlabel("actual")

plt.ylabel("predicted")

**14. Future scope**

 Expand to multilingual queries using XLM-Roberta

 Integrate chatbot interface for two-way communication

 Incorporate confidence scores for human-in-the-loop validation

 Fine-tune large language models (e.g., GPT, BERT-large) for better performance

**13. Team Members and Roles**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME** | **ROLE** | **RESPONSIBILITY** |
| 1. | Sweetha Mirra A | Member | Model building, Model evaluation |
| 2. | Sharmila S | Member | Feature Engineering |
| 3. | Sowmiya M | Member | Exploratory Data Analysis |
| 4. | Keerthika T | Member | Visualization |
| 5. | Yuvasri B | Leader | Data collection, Data cleaning |

**14. Google Colab Link**

<https://colab.research.google.com/drive/1KFab9xUrwaf1xwLYjeNYSwBSjlIzpEyA?usp=sharing>