Project Tittle : Revolutionizing customer support with an intelligent chatbot for

automated assisstance

PHASE-2

1. Problem Statement

Traditional customer support systems often struggle with high response times, limited availability, and inconsistent service quality. These issues lead to customer dissatisfaction, increased operational costs, and inefficiencies in handling repetitive queries. As customer expectations for instant, accurate, and 24/7 support continue to rise, there is a pressing need for a scalable and intelligent solution. The challenge is to develop a chatbot system that can provide automated, real-time assistance, understand natural language, learn from interactions, and integrate seamlessly with existing support channels to enhance the overall customer experienc

1. Project Object

**1.Develop an Intelligent Chatbot System:**  
Design and implement a chatbot powered by Natural Language Processing (NLP) and Machine Learning (ML) to understand and respond to customer queries effectively.

**2.Automate Repetitive Customer Queries:**  
Identify and automate responses to common, repetitive customer questions to reduce the workload on human support agents.

**3.Enhance Customer Experience:**  
Provide instant, 24/7 assistance with consistent and accurate responses to improve overall customer satisfaction and engagement.

**4.Integrate with Existing Support Channels:**  
Ensure seamless integration of the chatbot with current customer support platforms such as websites, mobile apps, and social media.

**5.Enable Context-Aware Interactions:**  
Implement contextual understanding to allow the chatbot to maintain conversation history and personalize interactions based on user data.

**6.Monitor and Improve Chatbot Performance:**  
Continuously analyze chatbot interactions to identify areas of improvement and optimize the system using feedback and analytics

1. Flowchart of the Project Workflow

Data Collection

Deployment using Gradio

Visualization of Result

Model Building & Evaluation

Feature Engineering

Exploratory Data Analysis(EDA)

Data preprocessing

1. Data Description

 **Customer Interactions:**

* **Types of Queries:** Data detailing common customer queries (e.g., order status, product inquiries, troubleshooting, billing questions).
* **Response Times:** Tracking the time it takes for the chatbot to respond to customer inquiries, with performance metrics such as average time per response.
* **Satisfaction Scores:** Customer feedback after chatbot interactions, usually measured through satisfaction surveys or post-interaction ratings.
* **Escalation Data:** Instances where the chatbot cannot resolve an issue and hands off the conversation to a human representative.

 **Chatbot Features:**

* **Natural Language Processing (NLP) Metrics:** Data on how well the chatbot understands and processes customer queries. This includes sentiment analysis, keyword recognition, and the bot’s ability to comprehend varying levels of complexity in language.
* **Resolution Rate:** How frequently the chatbot resolves queries independently without requiring human intervention.
* **Learning Curve:** Data on how the chatbot's performance improves over time as it learns from past interactions and is trained with new information.

 **Customer Demographics:**

* **User Information:** Characteristics such as age, region, device type, and language preferences of the customers using the chatbot.
* **Customer History:** Past interaction data that may help the chatbot provide personalized responses based on customer profiles and history.

 **Business Metrics:**

* **Cost Savings:** Data comparing the operational cost of human support agents versus the chatbot’s implementation (e.g., savings on labor, increased efficiency).
* **Efficiency Gains:** Data showing improvements in response time, reduction in wait times, or the volume of handled queries per day.
* **Customer Retention:** Metrics indicating whether chatbot interactions lead to higher customer retention, repeat business, or increased customer loyalty.s

Dataset Link: Bitext\_Sample\_customer\_support\_training\_Dataset\_27K\_responses-v11.csv

5.Data preprocessing

**a. Tokenization:**

* **What:** Tokenization is the process of splitting text into individual words or subword units (tokens).
* **Why:** This helps the model understand and process each word or character in isolation, enabling better NLP processing.
* **Example:** "How can I track my order?" becomes ['How', 'can', 'I', 'track', 'my', 'order', '?'].

**b. Lowercasing:**

* **What:** Convert all text to lowercase to ensure uniformity (since "Order" and "order" should be considered the same).
* **Why:** Helps reduce the complexity of the text by normalizing case differences.

**c. Removing Stop Words:**

* **What:** Remove common words like "the", "is", "and", etc., that do not carry significant meaning.
* **Why:** This reduces the dimensionality and focuses on meaningful words.
* **Tools:** NLTK, SpaCy.

6.Exploratory Data Analysis

**Understanding the Data**

Start by loading the data and understanding its structure. The goal is to get a sense of the number of rows, columns, and data types, as well as the variables that might be included in your dataset.

**Key Steps:**

* **Shape of the Data:** How many rows and columns are in the dataset?
* **Data Types:** Check the data types of each column (numerical, categorical, text, datetime).
* **Summary Statistics:** Use functions like describe() to get the statistical summary (mean, median, std, min, max, etc.) for numerical variables.
* **Missing Data:** Identify any missing values in the dataset.

python

CopyEdit

# Sample code to explore the data using Pandas

import pandas as pd

# Load the dataset

df = pd.read\_csv("customer\_support\_chatbot\_data.csv")

# Shape of the dataset

print(df.shape)

# Data types and missing values

print(df.info())

# Statistical summary of numerical features

print(df.describe())

# Check for missing values

print(df.isnull().sum())

**2. Analyzing the Distribution of Numerical Variables**

Next, focus on the distribution of numerical features such as response time, satisfaction score, and interaction duration. This helps identify trends, distributions, and outliers.

**Key Steps:**

* **Histograms:** To visualize the frequency distribution of numerical features.
* **Boxplots:** To detect outliers in numerical features.
* **Correlation Matrix:** To understand the relationship between numerical variables.

python

CopyEdit

import matplotlib.pyplot as plt

import seaborn as sns

# Plot histograms for numerical features

df['response\_time'].hist(bins=20)

plt.title('Distribution of Response Time')

plt.xlabel('Response Time (in seconds)')

plt.ylabel('Frequency')

plt.show()

# Plotting boxplot to detect outliers

sns.boxplot(x=df['satisfaction\_score'])

plt.title('Boxplot of Satisfaction Score')

plt.show()

# Correlation matrix to explore relationships

corr\_matrix = df.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

7. Feature Engineering

**Text-Based Features (from Customer Queries and Responses)**

Since you have text data (customer queries and chatbot responses), it’s important to extract meaningful features from this unstructured text. Here are some techniques for feature engineering from text:

**a. Query Length (Word or Character Count)**

* **What:** The length of the customer’s query, measured in words or characters.
* **Why:** Longer queries might indicate more complex issues or confusion, which could correlate with longer response times or lower satisfaction.
* **How:** Calculate the number of words or characters in each query.

python

CopyEdit

df['query\_word\_count'] = df['query\_text'].apply(lambda x: len(x.split()))

df['query\_char\_count'] = df['query\_text'].apply(lambda x: len(x))

**b. Sentiment of the Query**

* **What:** Analyze the sentiment of the customer’s query to determine if it’s positive, negative, or neutral.
* **Why:** The sentiment of the query can influence the chatbot’s response and impact customer satisfaction.
* **How:** Use sentiment analysis (e.g., TextBlob, VADER, or deep learning models) to classify the sentiment of the query.

python

CopyEdit

from textblob import TextBlob

df['query\_sentiment'] = df['query\_text'].apply(lambda x: TextBlob(x).sentiment.polarity)

8. Model Building

**efine the Problem**

Before you start building the model, it’s important to define the problem you want to solve. Based on the features and the business requirements, the possible objectives can include:

* **Classification Problem:** Predicting categorical outcomes like whether the customer will be satisfied (binary: satisfied vs. unsatisfied), or predicting the category of the query (e.g., "Billing," "Technical Support").
* **Regression Problem:** Predicting continuous outcomes like customer satisfaction scores, response times, or interaction duration.

For this example, let's assume you are building a **classification model** to predict customer satisfaction or whether the query will be resolved (binary classification).

**2. Prepare Data for Modeling**

**Step 1: Splitting the Data**

You need to split the dataset into training and testing sets to evaluate the model’s performance.

* **Training Set:** Used to train the model.
* **Testing Set:** Used to evaluate the performance of the model.

Typically, a 70:30 or 80:20 split is used.

python

CopyEdit

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

# Select features (X) and target variable (y)

X = df.drop(columns=['satisfaction\_score', 'resolved']) # Drop target column

y = df['satisfaction\_score'] # Target variable for classification

# Split the data into training and testing sets (80% training, 20% testing)

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

**Step 2: Scaling the Data (If Needed)**

Some models, especially those based on distance (like k-NN or SVM), require the features to be scaled. You can use standard scaling to ensure the features have a mean of 0 and a standard deviation of 1.

python

CopyEdit

from sklearn.preprocessing import StandardScaler

# Apply standard scaling to the features (exclude the target variable)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

9.Visualization of Result & Model Insights

## ****1. Confusion Matrix****

* It shows the number of correct and incorrect predictions.
* Helps identify where the model is making mistakes.

## ****2. ROC Curve & AUC Score****

* Measures the model’s ability to distinguish between classes.
* AUC closer to 1 means better performance.

## ****3. Feature Importance Plot****

* Shows which features contributed most to the model’s predictions.

## ****4. Classification Report Visualization (Optional)****

Use yellowbrick or a table format to visualize Precision, Recall, F1-Score.

## ****5. Distribution of Predictions****

* Helps see class imbalance or prediction skew.
* Useful for evaluating model fairness or bias.

**Overall Insights You Can Derive**

* **Key Drivers of Satisfaction:** Feature importance reveals what matters most (e.g., fast response time, polite sentiment).
* **Where the Chatbot Fails:** Confusion matrix and classification report show if the model misses unhappy users or complex queries.
* **Prediction Biases:** Class distribution plots and AUC show if the model is biased or generalizes well.

10.Tools and Technologies Used

## ****1. Programming Language****

### ▸ ****Python****

* Widely used in data science and machine learning.
* Large ecosystem of libraries for NLP, EDA, modeling, and deployment.

## 📦 ****2. Libraries and Frameworks****

### 🔹 ****Data Processing & Analysis****

* **Pandas** – For data manipulation and analysis.
* **NumPy** – For numerical operations.

### 🔹 ****Visualization****

* **Matplotlib** – For creating static, animated, and interactive plots.
* **Seaborn** – For statistical data visualization (built on top of Matplotlib).
* **Yellowbrick** – For model performance visualization.

### 🔹 ****Machine Learning & Modeling****

* **Scikit-learn** – For classification, regression, model selection, and preprocessing.
* **XGBoost** – For high-performance gradient boosting models.
* **RandomForest, SVM, LogisticRegression** – From Scikit-learn for diverse ML modeling.

### 🔹 ****Natural Language Processing (NLP)****

* **TextBlob** – For basic sentiment analysis and NLP.
* **SpaCy** – For advanced NLP tasks like tokenization and entity recognition.
* **NLTK** – For classic NLP utilities like stopword removal, stemming, etc.

## 🧪 ****3. Model Evaluation Tools****

* **Scikit-learn metrics module** – For confusion matrix, accuracy, F1-score, ROC-AUC, etc.
* **Yellowbrick** – For visual diagnostics of machine learning models.

## 🧼 ****4. Data Preprocessing Tools****

* **Scikit-learn's StandardScaler, LabelEncoder, OneHotEncoder** – For scaling and encoding data.
* **TF-IDF Vectorizer** – For converting text to numerical format using term frequency.

## ☁️ ****5. Deployment & Integration (Optional/Advanced)****

If integrating the model with a chatbot in production:

### ▸ ****Chatbot Frameworks****

* **Rasa** – Open-source conversational AI framework.
* **Dialogflow** – Google’s NLP engine for chatbots.
* **Microsoft Bot Framework** – For enterprise-grade bots.

### ▸ ****Model Serving****

* **Flask or FastAPI** – To deploy ML models as REST APIs.
* **Docker** – To containerize and deploy models.
* **Cloud Platforms (AWS, GCP, Azure)** – For scalable deployment.

## 💾 ****6. Development Environment****

* **Jupyter Notebook** – For interactive coding and visualization.
* **VS Code / PyCharm** – For development and debugging.
* **Google Colab** – For free cloud-based notebook execution with GPU support.

## 🗃️ ****7. Version Control & Project Management****

* **Git & GitHub/GitLab** – For code versioning and collaboration.
* **Trello / Jira** – For task tracking and project management (especially in teams).

11.Team Members and Contributions

**1. D.Gopika – Data Analyst**

* **Responsibilities:**
  + Performed **data cleaning**: handled missing values, standardized formats, and removed outliers.
  + Conducted **exploratory data analysis (EDA)**: visualized data distributions, correlations, and trends using Seaborn and Matplotlib.

**2. A.Dhivya – Machine Learning Engineer**

* **Responsibilities:**
  + Executed **feature engineering**: implemented text vectorization (TF-IDF), sentiment extraction, and categorical encoding.
  + Led **model development**: built, trained, and evaluated classification models (e.g., Logistic Regression, Random Forest, XGBoost).

**3.R.Elakkiya – Documentation Lead**

* **Responsibilities:**
  + Handled **documentation and reporting**: compiled methodology, model performance, visualizations, and project insights into a final report.
  + Created presentation slides and summaries for stakeholder communication.

**Task Summary**

| **Task** | **Assigned Member** |
| --- | --- |
| **Data Cleaning** | D.Gopika |
| **Exploratory Data Analysis (EDA)** | D.Gopika |
| **Feature Engineering** | A.Dhivya |
| **Model Development** | A.Dhivya |
| **Documentation & Reporting** | R.Elakkiya |