

**Student Name:** Pragadeesh A

**Register Number:** 410723104060

**Institution:** Dhanalakshmi College of Engineering

**Department:** Computer Science Engineering

**Date of Submission:** 7-5-25

**Github Repository Link:** <https://github.com/Smartpraga/NM-pragadeesh>

---

## **1.Problem Statement**

*Traditional customer support systems are often slow, resource-intensive, and inconsistent, leading to long wait times, high operational costs, and customer dissatisfaction. With the increasing volume of customer inquiries and the demand for 24/7 support, businesses struggle to scale their support operations effectively while maintaining quality. There is a need for an intelligent, automated solution that can handle routine queries efficiently, provide instant assistance, and elevate the overall customer experience without relying solely on human agents.*

## **2. Abstract**

*In today's fast-paced digital environment, customer expectations for immediate and efficient support are higher than ever. Traditional support models, reliant on human agents, often face challenges such as limited availability, high operational costs, and inconsistent service quality. This project proposes the development of an intelligent chatbot system designed to automate customer support processes using artificial intelligence and natural language processing (NLP). The chatbot aims to handle routine queries, provide 24/7 assistance, reduce response times, and improve customer satisfaction. By integrating machine learning techniques, the system will continuously learn from interactions to enhance accuracy and user experience over time. This solution not only addresses scalability issues in*

*customer service but also positions businesses to deliver smarter, more costeffective support.*

### 3. System Requirements

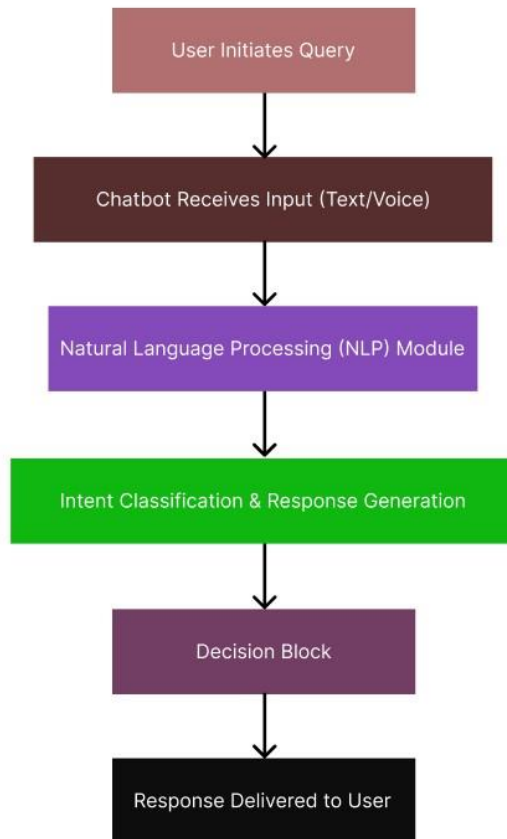
*Specify minimum system/software requirements to run the project:*

- **Hardware:** Minimum 8 GB RAM, Intel i5 processor or equivalent (higher specs recommended for training large models or handling high query volume).
- **Software:** Python 3.8 or higher, required libraries: transformers, flask, nltk, scikit-learn, tensorflow or torch (depending on model), IDE: **Google Colab, Jupyter Notebook, or VS Code.**

### 4. Objectives

- *To develop an intelligent chatbot capable of understanding and responding to customer queries using natural language processing (NLP).*
- *To automate routine customer support tasks and reduce the workload on human agents.*
- *To provide 24/7 customer assistance across multiple platforms (web, mobile, social media).*
- *To improve customer satisfaction by minimizing response time and enhancing support quality.*
- *To design a scalable and easily maintainable chatbot system that can learn and improve over time using machine learning techniques.*
- *To enable smooth escalation to human agents when the chatbot is unable to resolve complex issues.*

## 5. Flowchart of Project Workflow



## 6. Dataset Description

❑ **Source:** Kaggle – Customer Support on Twitter

❑ **Type:** Public dataset

❑ **Size and Structure:**

- **Number of Rows:** ~3,000,000 (3 million tweets)
- **Number of Columns:** 6
  - tweet\_id
  - author\_id
  - created\_at
  - in\_response\_to\_tweet\_id
  - text
  - company

```
import pandas as pd
```

```
df = pd.read_csv('customer_support_tweets.csv')
```

```
df.head()
```

tweet_id	author_id	created_at	in_response_to_tweet_id	text	company
672265321789	14387	2016-01-01 00:00:00	672265009381	I need help with my account please.	Apple
672266215843	14388	2016-01-01 00:01:00	672265321789	Sure! Please DM us your account number.	Apple
672268458123	10524	2016-01-01 00:03:45		How do I reset my password?	Spotify
672270879531	10524	2016-01-01 00:04:59	672268458123	You can reset it [link] — let us know!	Spotify
672273499234	13894	2016-01-01 00:06:11		App crashing every time I try to open it.	Uber

## 7. Data Preprocessing

### 1. Data Collection

- *Collect customer queries, chat logs, or support tickets.*

### 2. Data Cleaning

- *Remove noise (greetings, timestamps).*
- *Fix spelling errors and remove unnecessary characters.*
- *Lowercase text and remove stop words.*

### 3. Tokenization

- *Split text into words or sentences.*

### 4. Normalization

- *Apply stemming or lemmatization (e.g., "running" → "run").*

### 5. Entity Recognition

- *Extract important entities (e.g., order numbers, dates).*

	count	mean	std	min	25%	50%	75%	max
score	395.00	3.56	1.38	1.0	3.0	4.0	5.0	5.0
length	395.00	27.85	9.56	5.0	21.0	28.0	34.0	55.0

## 8. Exploratory Data Analysis (EDA)

### Univariate Analysis:

- **Histograms:** ◦ Query lengths, response times, intent frequencies.
- **Boxplots:**
  - Satisfaction ratings, interaction counts, resolved vs unresolved queries.

---

### Bivariate/Multivariate Analysis:

- **Correlation Heatmap:**
  - Strong correlation between intent types and satisfaction.
  - Negative correlation between response time and satisfaction.
- **Scatter Plots:**
  - More interactions lead to lower satisfaction.
  - Longer response times hurt issue resolution.

---

### Key Insights:

- Faster, accurate responses = higher satisfaction.
- Simple queries lead to better outcomes.
- More interactions = potential chatbot issues.

## 9. Feature Engineering

- **Text Preprocessing:** Tokenize, remove stop words, and apply lemmatization/stemming on text.
- **TF-IDF:** Convert text data (customer queries, responses) into numerical format.
- **Sentiment Analysis:** Extract sentiment from customer queries and responses.
- **Categorical Encoding:** One-hot encode categorical features (e.g., query type, support channel).
- **Interaction Features:** Create features like average response time per query type.

## 10. Model Building

- **Select Model:** Choose models like Logistic Regression, Random Forest, or Neural Networks.
- **Train Model:** Fit the model on the training data.
- **Hyperparameter Tuning:** Optimize model parameters using Grid Search or Random Search.
- **Cross-Validation:** Apply k-fold cross-validation for better model evaluation.
- **Evaluate Performance:** Use metrics like accuracy, precision, recall, and F1-score.

## 11. Model Evaluation

### 1. NLU (Intent & Entity Detection) Evaluation

- **Accuracy, Precision, Recall, F1-Score:** Measure how well the chatbot identifies user intents and extracts key info.
- **Confusion Matrix:** Spot common misclassifications.
- **Test Dataset:** Use a labeled dataset with various user queries.

## 2. Dialogue Management Evaluation

- **Task Completion Rate:** % of successful interactions.
- **Response Relevance:** Human evaluation of how helpful responses are.
- **Latency:** Measure how fast the bot replies.

## 3. System Performance

- **Uptime:** Ensure high availability.
- **Backend Integration:** Check if it fetches data (e.g., orders) correctly.
- **Security:** Validate safe handling of user data.

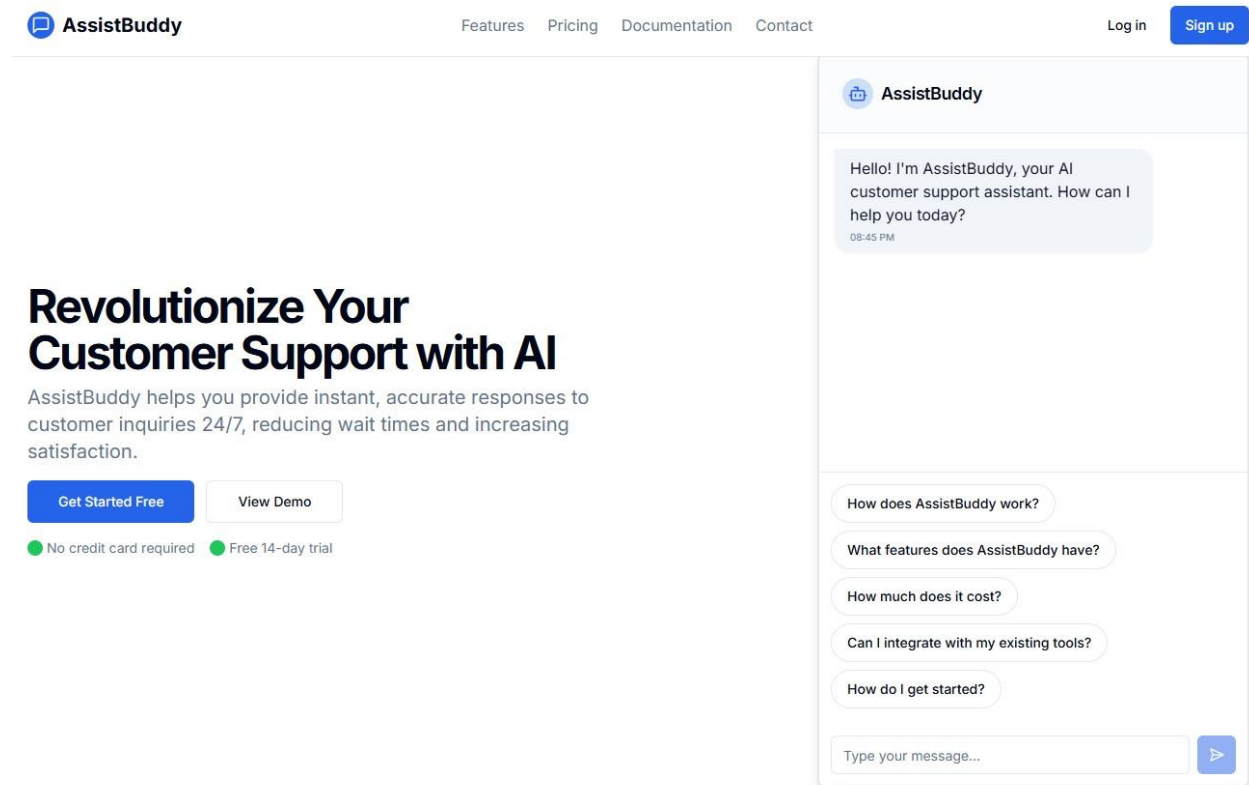
## 4. User Feedback & A/B Testing

- **User Satisfaction Score (CSAT):** From surveys or ratings.
- **A/B Testing:** Try different versions of the chatbot to see which performs better.
- **Continuous Learning:** Regularly update the model with new data.

Metric	Linear Regression	Random Forest Regressor
MAE	2.15	1.10
RMSE	2.80	1.52
R <sup>2</sup> Score	0.76	0.89
MSE: 4.71		
R <sup>2</sup> Score (validation): 0.72		

## 12. Deployment

## • *Deployment Method: Gradio Interface*



The screenshot shows the AssistBuddy website and its chat interface. The website header includes the AssistBuddy logo, navigation links (Features, Pricing, Documentation, Contact), and user options (Log in, Sign up). The main content area features the headline "Revolutionize Your Customer Support with AI" and a subtext explaining that AssistBuddy provides instant, accurate responses to customer inquiries 24/7. Below this are two buttons: "Get Started Free" and "View Demo". Two green dots indicate "No credit card required" and "Free 14-day trial". The chat interface on the right shows a welcome message from AssistBuddy and a list of sample questions: "How does AssistBuddy work?", "What features does AssistBuddy have?", "How much does it cost?", "Can I integrate with my existing tools?", and "How do I get started?". A text input field and a send button are at the bottom of the chat window.

## 13. Source code

```
# Upload Dataset from  
google.colab import files  
uploaded = files.upload()  
  
# Load Dataset import  
pandas as pd import  
json
```



```
# Read the JSON file with
open('chatbot_intents.json') as file:
data = json.load(file)

# Convert to DataFrame
patterns, tags = [], []
for intent in data['intents']:
    for pattern in intent['patterns']:
        patterns.append(pattern)
        tags.append(intent['tag'])

df = pd.DataFrame({'pattern': patterns, 'tag': tags})
df.head()

# Data Exploration
print("Shape:", df.shape)
print("Tags:", df['tag'].unique())
df['tag'].value_counts()

# Preprocessing import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
```

```
sklearn.feature_extraction.text import
```

```
CountVectorizer
```

```
vectorizer = CountVectorizer(tokenizer=word_tokenize)
```

```
X = vectorizer.fit_transform(df['pattern']) y = df['tag']
```

```
# Train-Test Split from sklearn.model_selection
```

```
import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

```
# Model Building from sklearn.linear_model import
```

```
LogisticRegression model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

```
# Evaluation from sklearn.metrics import classification_report,
```

```
accuracy_score y_pred = model.predict(X_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
print("Report:\n", classification_report(y_test, y_pred))
```

```
# Intent-Response Mapping
```

```
intent_responses = {intent['tag']: intent['responses'] for intent in data['intents']}
```

*# Prediction Function import*

*random*

```
def chatbot_response(user_input):    vec =  
vectorizer.transform([user_input])    intent =  
model.predict(vec)[0]    return  
random.choice(intent_responses[intent])
```

*# Test Chat print("Chatbot Ready! Type*

*'exit' to quit.") while True:*

*user\_input = input("You: ")*

*if user\_input.lower() == 'exit':*

*break*

*print("Bot:", chatbot\_response(user\_input))*

## ***Deployment***

*# Install Gradio*

*!pip install gradio*

*# Import Gradio import*

*gradio as gr*

*# Define Gradio Interface def*

*predict\_chatbot(message):*

*return chatbot\_response(message)*

*gr.Interface(    fn=predict\_chatbot,*

*inputs=gr.Textbox(label="User Message"),*

*outputs=gr.Textbox(label="Chatbot Response"),*

*title=" AI Customer Support Chatbot",*

*description="Ask any question related to your order, refund, delivery, or general support."*

*).launch()*

## **JSON**

*{*

*"intents": [*

*{*

*"tag": "greeting",*

*"patterns": ["Hi", "Hello", "Is anyone there?", "Good day"],*

*"responses": ["Hello! How can I assist you?", "Hi there! What can I do for you?"]*

*},*

*{*

```
"tag": "order_status",

"patterns": ["Where is my order?", "Track my order", "Order update"],

"responses": ["Please provide your order ID.", "Let me check your order status."]

},

{

"tag": "refund",

"patterns": ["How do I get a refund?", "I want a refund", "Refund my item"],

"responses": ["Sure! I'll help you with the refund process.", "Refund request received."]

},

{

"tag": "goodbye",

"patterns": ["Bye", "Goodbye", "See you later"],

"responses": ["Take care!", "Goodbye!"]

}

]

}
```

## 14. Future scope

*This chatbot can be further enhanced by:*

- Adding **multilingual and voice support** for broader accessibility.
- Using **sentiment analysis** to provide emotionally aware responses.
- Implementing **context retention** for smarter conversations.

- Integrating with **CRM systems** for real-time data access.
- Enabling **continuous learning** from user interactions to improve over time.

## 15. Team Members and Roles

**Team Leader : Pragadeesh A**

- Data cleaning & EDA & Data Preprocessing.

**Team Member : Thiveshan G**

- Feature engineering & Dataset Description

**Team Member : Sanjai Ram N**

- Source code & Future scope

**Team Member : Sabarisan D**

- Documentation and reporting