**Phase-3 Submission Template**

**Student Name:** MOHAMMED FAROOQ H

**Register Number:** 410723104045

**Institution:** Dhanalakshmi college of engineering

**Department:** Computer Science & Engineering

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

**Github Repository Link:** [**https://github.com/Xjod01/NM\_FAROOQ**](https://github.com/Xjod01/NM_FAROOQ)

Revolutionizing customer support with an intelligent chatbot for automated assistance

### **1. Problem Statement**

Traditional customer support systems often struggle with long response times, inconsistent service quality, and high operational costs due to their reliance on human agents for handling repetitive and routine queries. This leads to customer dissatisfaction, decreased efficiency, and scalability challenges. There is a growing need for an intelligent, always-available solution that can automate common support tasks, provide instant responses, and seamlessly escalate complex issues—ensuring improved customer experience and reduced workload on support teams*.*

### **2. Abstract**

This project aims to revolutionize customer support by developing an intelligent chatbot that provides automated, real-time assistance to users. Traditional support systems face challenges such as delayed response times, inconsistent service, and high operational costs due to heavy dependence on human agents. The objective is to create a smart, scalable solution that can handle frequently asked questions, troubleshoot common issues, and route complex queries to human representatives when needed. The chatbot leverages natural language processing (NLP) and machine learning (ML) to understand user intent, respond accurately, and continuously improve from interactions.

### **3. System Requirements**

Specify minimum system/software requirements to run the project:

Hardware:

Intel Core i5 or Higher

Minimum 8Gb RAM

Software:

- python 3.8+

- jupyter Notebook / Google Colab

- Libraries: pandas , numpy ,Scikit – learn , tensorflow , spaCY

### **4. Objectives**

* Enhance Customer Experience
* Provide 24/7 instant responses to common customer queries.
* Reduce customer wait times by at least 60% within the first 3 months of deployment.
* Automate Repetitive Support Tasks
* Automate handling of Tier 1 support tasks (FAQs, order tracking, password resets) to reduce manual workload by 50%.
* Increase Support Team Efficiency
* Enable live agents to focus on complex issues by routing routine queries to the chatbot.
* Improve agent response times and satisfaction ratings by at least 20%.
* Personalize Customer Interactions
* Use NLP and user data to deliver tailored responses that match the customer’s history and behavior.
* Achieve a 25% increase in customer satisfaction scores (CSAT).

**5. Flowchart of Project Workflow**

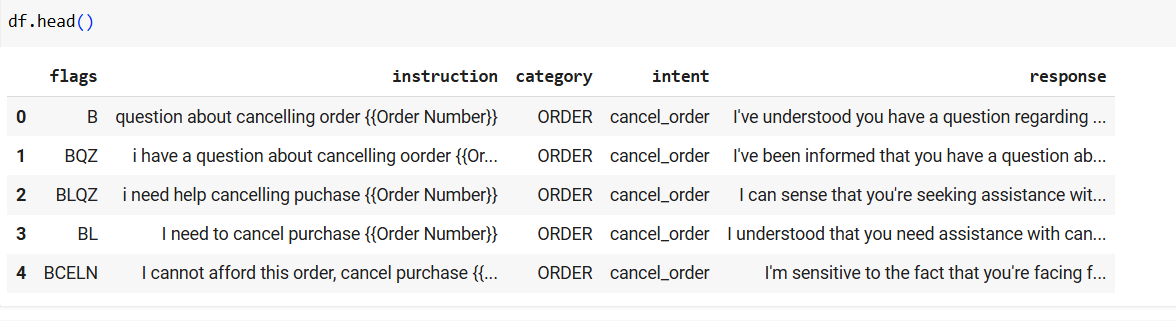
A diagram of data analysis

AI-generated content may be incorrect.

### **6. Dataset Description**

### **Source**: Kaggle-Telco Coustomer Churn.

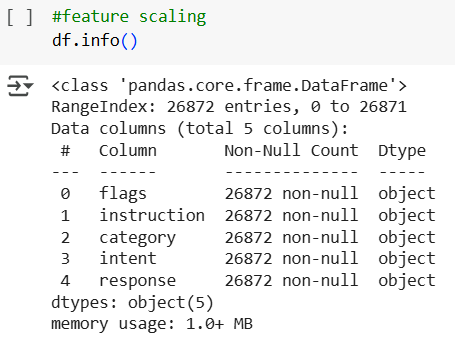
**Type**: Public



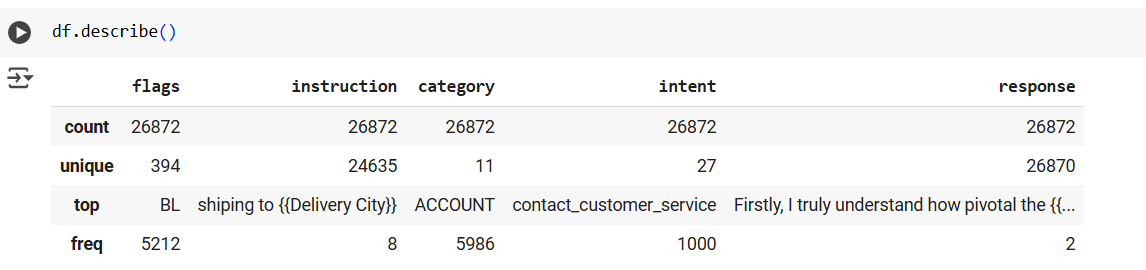
### 7. **Data** **Preprocessing**

* Handle missing values, duplicates, outliers
* Feature encoding and scaling
* Show before/after transformation screenshots

### 



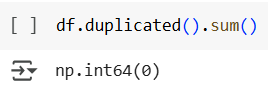
### df.describe()



### df.isnull().sum()

### 

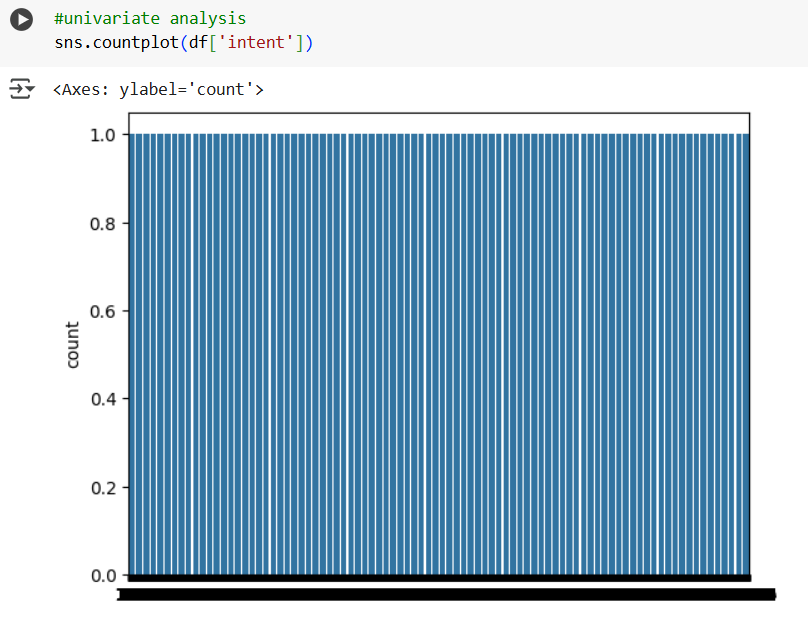
### df.duplicated().sum()



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

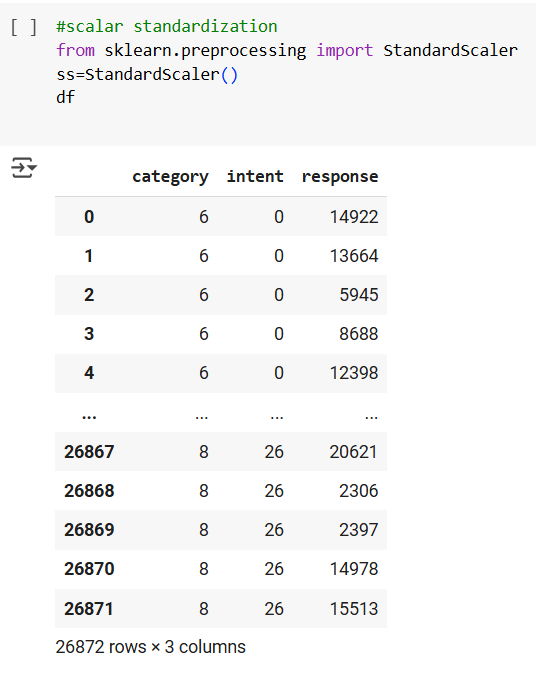
* Use visual tools like histograms, boxplots, heatmaps
* Reveal correlations, trends, patterns
* Write down key takeaways and insights
* Include screenshots of visualizations

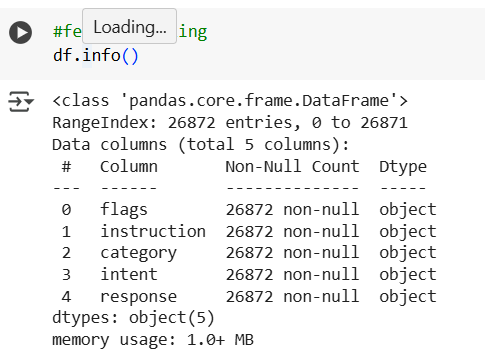
### 



### **9. Feature Engineering**

* New feature creation
* Feature selection
* Transformation techniques
* Explain why and how features impact your model



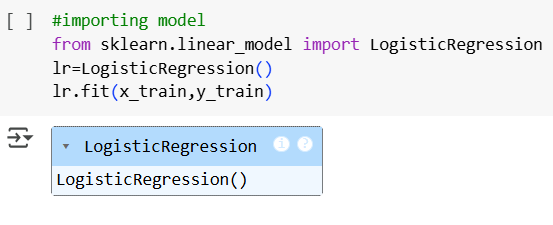
**

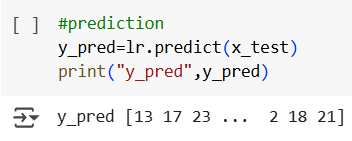
### 

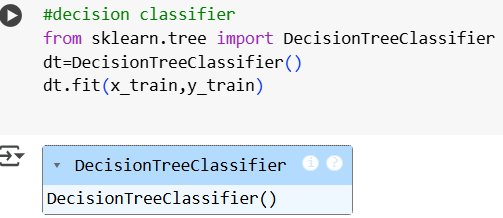
### **10. Model Building**

* Try multiple models (baseline and advanced)
* Explain why those models were chosen
* Include screenshots of model training outputs

### 

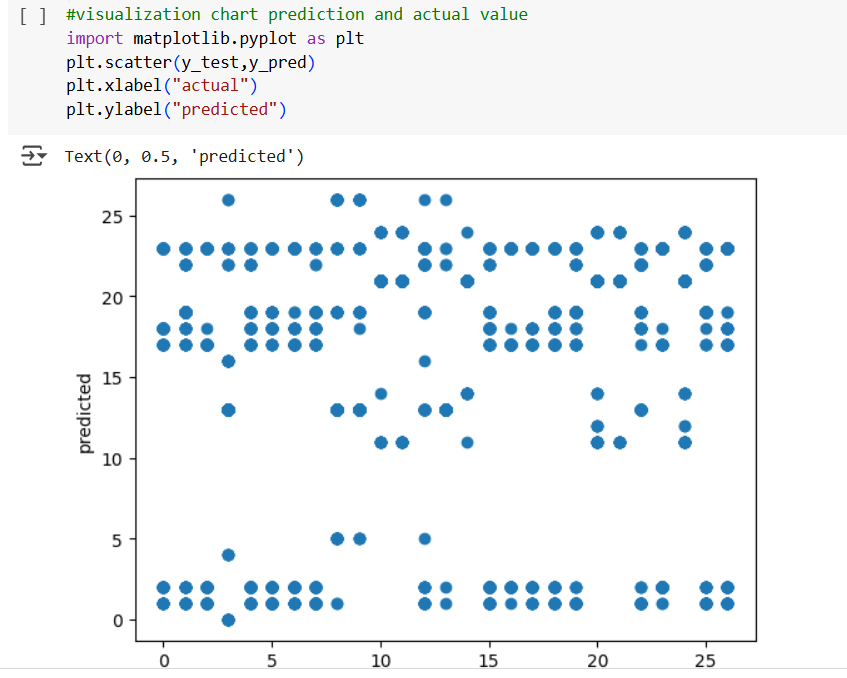


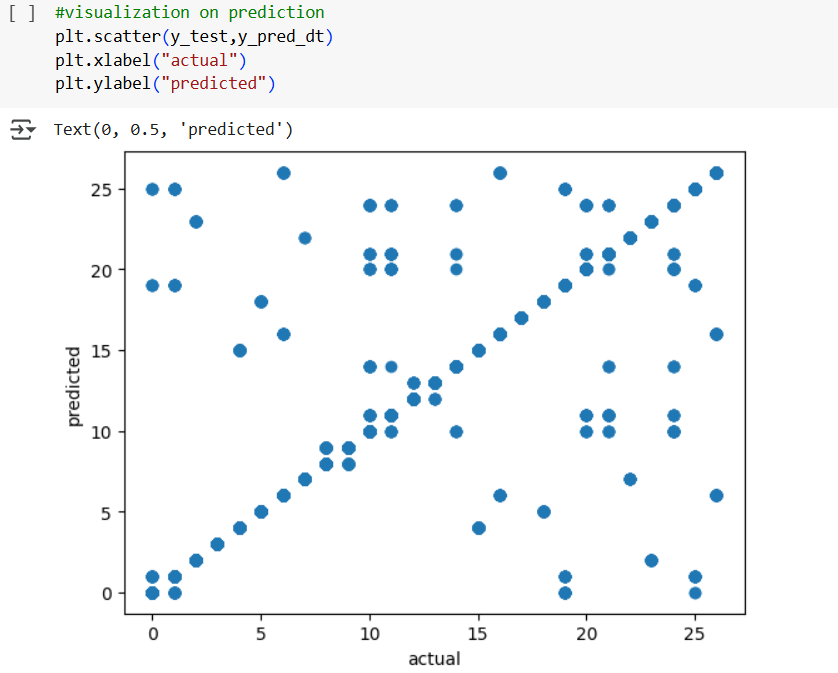


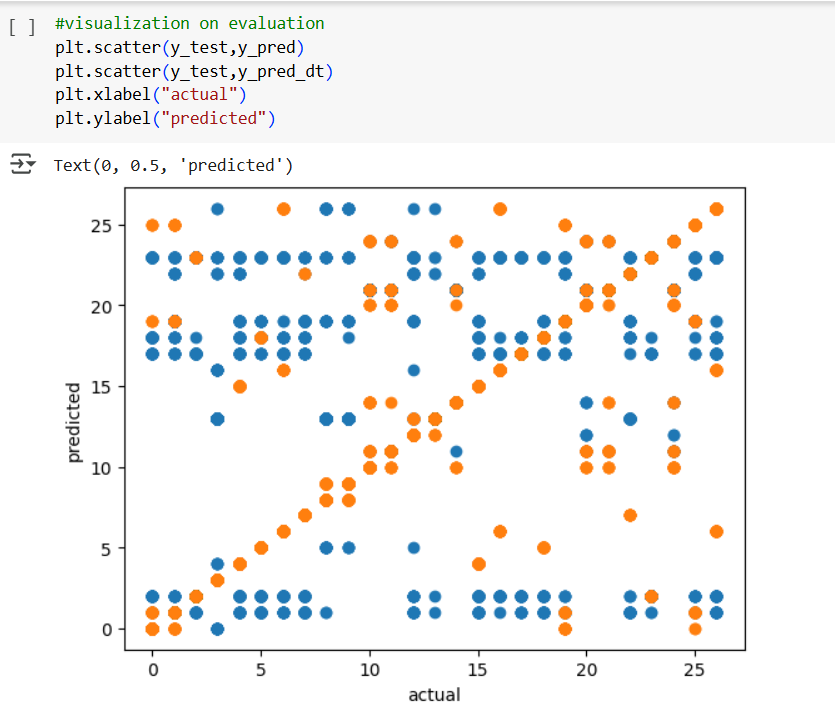


### **11. Model Evaluation**

* Show evaluation metrics: accuracy, F1-score, ROC, RMSE, etc.
* Visuals: Confusion matrix, ROC curve, etc.
* Error analysis or model comparison table
* Include all screenshots of outputs

**

**

**

### **12. Deployment**

* Deploy using a free platform:
* Streamlit Cloud
* Gradio + Hugging Face Spaces
* Flask API on Render or Deta
* Include:
* Deployment method
* Public link
* UI Screenshot
* Sample prediction output

**13. Source code**

* import pandas as pd
* from sklearn.model\_selection import train\_test\_split
* from sklearn.preprocessing import LabelEncoder, StandardScaler
* from sklearn.ensemble import RandomForestClassifier
* from sklearn.metrics import classification\_report, confusion\_matrix
* import matplotlib.pyplot as plt
* import seaborn as sns
* df = pd.read\_csv('/content/Customer-Churn-Records.csv')
* df.head( )
* df.info( )
* df.describe( )
* df.isnull( ).sum( )
* df.drop\_duplicates( )
* df.drop\_duplicates( ).sum( )
* df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)
* #Histogram chart
* df.hist(figsize=(10,10))
* plt.show()
* #Bivariate analysis
* sns.pairplot(df)
* plt.show()
* #Feature engineering
* for col in ['Geography', 'Gender', 'Card Type']:
* le = LabelEncoder()
* df[col] = le.fit\_transform(df[col])
* df
* #Scalar standardization
* scaler = StandardScaler()
* df\_scaled = scaler.fit\_transform(df)
* df
* #Label encoding and onehot encoding
* df\_encoded = pd.get\_dummies(df, columns=['Geography', 'Gender', 'Card Type'])
* df
* #Model building
* X = df.drop('Exited', axis=1)
* y = df['Exited']
* #import model
* from sklearn.model\_selection import train\_test\_split
* from sklearn.ensemble import RandomForestClassifier
* from sklearn.metrics import classification\_report, confusion\_matrix
* x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,
* random\_state=42)
* from sklearn.linear\_model import LogisticRegression
* model = LogisticRegression()
* model.fit(x\_train, y\_train)
* #Prediction
* y\_pred = model.predict(x\_test)
* print("y\_prediction", y\_pred)
* #Random forest classifier
* model = RandomForestClassifier(n\_estimators=100, random\_state=42)
* model.fit(x\_train, y\_train)
* y\_random\_prediction = model.predict(x\_test)
* print("y\_prediction", y\_random\_prediction)
* # Evaluate
* y\_pred = model.predict(x\_test)
* print("Classification Report:\n", classification\_report(y\_test, y\_pred))
* print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))
* y\_random\_prediction = model.predict(x\_test)
* print("Classification Report:\n", classification\_report(y\_test,
* y\_random\_prediction))
* print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_random\_prediction))
* #Visualize prediction and actual value
* plt.figure(figsize=(10, 6))
* plt.scatter(y\_test, y\_pred, alpha=0.5, label='Predicted')
* plt.scatter(y\_test, y\_random\_prediction, alpha=0.5, label='Random Predicted')
* plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], linestyle='--',
* color='red', label='Perfect Prediction')
* plt.xlabel('Actual Values')
* plt.ylabel('Predicted Values')
* plt.title('Actual vs. Predicted Values')
* plt.legend()
* plt.show()
* #Histogram chart random forest and logistic regression
* plt.figure(figsize=(10, 6))
* plt.hist(y\_pred, bins=20, alpha=0.5, label='Logistic Regression')
* plt.hist(y\_random\_prediction, bins=20, alpha=0.5, label='Random Forest')
* plt.xlabel('Predicted Values')
* plt.ylabel('Frequency')
* plt.title('Histogram of Predicted Values')
* plt.legend()
* plt.show()

**14. Future scope**

* Integrate with live CRM systems for real-time predictions
* Add NLP to analyze customer feedback sentiment
* Implement customer segmentation for targeted retention strategies

**13. Team Members and Roles**

|  |  |  |
| --- | --- | --- |
| **NAME** | **ROLE** | **RESPONSIBILITY** |
| HARIHARAN K | LEADER | Data Collection and Cleaning |
| MAGESH L | MEMBER | Data Visualization and Interpretation |
| MOHAMMED FAROOQ H | MEMBER | Exploratory Data Analysis |
| HARISH JAYARAJ R | MEMBER | Model Evaluation |
| ABDUL AZEEZ M | MEMBER | Model Building |

**GITHUB SCREENSHOT**