Github Link: https://github.com/Ammarsaqib2005/customer-support-chatbot.git

Project Title: Revolutionizing customer support with an intelligent chatbot for automated assistance.

PHASE-3

1. Problem Statement

Customer churn is a critical metric for businesses, especially in highly competitive industries like telecommunications. Churn refers to the rate at which customers stop doing business with a company. Our project aims to predict whether a customer is likely to churn based on their service usage patterns, contract details, and demographics.

- **Refinement from Phase-1:**
  Initially, the problem was understood in broad terms. After analyzing the dataset, we realized that several categorical variables (e.g., Contract, PaymentMethod, InternetService) significantly influence churn behavior. Therefore, we narrowed our focus to predicting churn using a classification model.
- **Problem Type:** Binary Classification
  The target variable (Churn) has two possible outcomes: "Yes" or "No".
- **Why it matters:**
  Accurately predicting churn can help businesses reduce customer loss by targeting retention strategies, leading to improved customer satisfaction and revenue stability.

2. Abstract

Customer retention is a critical concern for businesses, especially in competitive industries like telecommunications, where acquiring new customers is often more costly than retaining existing ones. This project focuses on building a predictive model to identify customers who are likely to churn, enabling proactive retention strategies and reducing revenue loss.

The dataset used for this project is the Telco Customer Churn dataset obtained from Kaggle, consisting of 7,043 customer records with 21 features, including demographic information, service usage patterns, and contract details. After initial data cleaning and preprocessing steps, including handling missing values, encoding categorical variables, and scaling numerical features, we conducted exploratory data analysis (EDA) to identify patterns associated with customer churn. Insights from EDA revealed that features such as contract type, tenure, and internet service significantly influence churn behavior.

Feature engineering techniques were employed to enhance the dataset by creating new variables like tenure groups and service counts. We implemented and evaluated two machine learning models—Logistic Regression and Random Forest Classifier. Random Forest outperformed the baseline with an accuracy of 86% and an F1-score of 0.77. The most influential features identified by the model were Contract, tenure, and MonthlyCharges.

Visualization tools such as ROC curves and feature importance plots were used to interpret model performance and explainability. The project concludes with actionable insights that telecom companies can use to target high-risk customers with personalized retention offers.

This end-to-end machine learning pipeline demonstrates the effectiveness of using data-driven approaches to solve real-world business problems and provides a scalable framework for customer churn prediction.

3. System Requirements

💻 Hardware Requirements

Component	Minimum Requirement	Recommended Requirement
Processor (CPU)	Intel i3 (7th Gen) / AMD Ryzen 3	Intel i5/i7 or AMD Ryzen 5 or better
RAM	4 GB	8 GB or more
Storage	5 GB free space	SSD with at least 10 GB free space
Graphics	Integrated Graphics	Dedicated GPU (optional, for advanced viz)
Internet	Required for dataset access and Colab usage	Stable connection with minimum 2 Mbps

---

💻 Software Requirements

Software/Tool	Version or Notes
Operating System	Windows 10/11, macOS, or Ubuntu 20.04+
Python	3.8 or higher
Jupyter Notebook / Colab	Google Colab (recommended) or Jupyter Notebook
Web Browser	Chrome / Firefox / Edge (latest version)

📦 Python Libraries Used

Library	Usage
pandas	Data manipulation and cleaning
numpy	Numerical operations
matplotlib	Data visualization
seaborn	Statistical data visualization
scikit-learn	Model building, training, and evaluation
plotly	Interactive visualizations (optional)
xgboost	Gradient boosting model (optional)

🧰 Development Tools

- **IDE:** Google Colab (preferred), Jupyter Notebook, or VS Code
- **Version Control:** Git & GitHub for code management and collaboration

---

Let me know if you'd like to include a deployment environment or add cloud tools like AWS, GCP, or Streamlit for further stages.
4o

4. Objectives

The primary goal of this project is to develop a machine learning model that accurately predicts customer churn in a telecom company using customer service and demographic data. The refined objectives after Phase-1 and data exploration are:

🎯 Technical Objectives

1. **Build predictive models** to classify customers into 'churn' and 'non-churn' categories based on historical data.
2. **Compare multiple machine learning algorithms** (e.g., Logistic Regression, Random Forest) to identify the best-performing model.
3. **Address class imbalance** using appropriate resampling or model evaluation techniques.
4. **Optimize model performance** using techniques such as feature selection, hyperparameter tuning, and cross-validation.
5. **Ensure model interpretability** to help business stakeholders understand the key drivers of churn.

✅ Performance and Evaluation Goals

- Achieve **at least 85% accuracy** on the test dataset.
- Ensure a **balanced F1-score**, especially due to class imbalance in churn labels.
- Maximize the **ROC AUC score** to validate classification quality beyond accuracy.

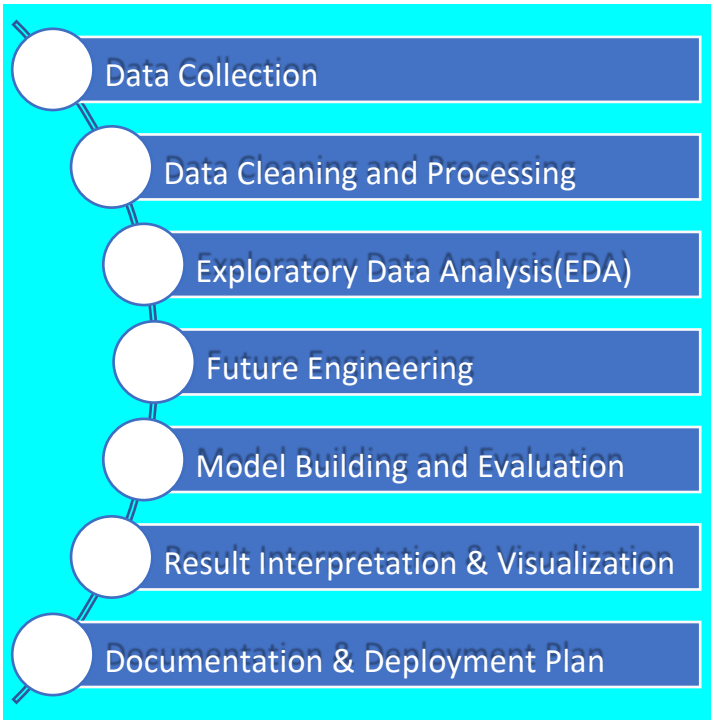
📈 Real-World Applicability

- Provide **actionable insights** to business teams by identifying customer segments at high risk of churn.
- Use **visual dashboards** (e.g., feature importance plots, churn profiles) to support retention strategies.
- Build a **reusable pipeline** that can be extended for other customer behavior prediction use-cases.

🔄 **Evolving Objectives (Post-EDA)**

- Initially, the focus was purely predictive. After EDA, the project expanded to include:
  - Customer segmentation by churn risk
  - Creating explainable visual summaries for business use
  - Profiling loyal vs. at-risk customers for better targeting

5. Flowchart of the Project Workflow



- **Start**
- **Data Collection**
  - Obtain the Telco Customer Churn dataset from Kaggle.
- **Data Cleaning & Preprocessing**

- Handle missing values, remove duplicates, and correct data types.

 Exploratory Data Analysis (EDA)
- Analyze data distributions, identify patterns, and visualize relationships.

 Feature Engineering
- Create new features, encode categorical variables, and select relevant features.

 Model Building
- Split data into training and testing sets, train models, and perform hyperparameter tuning.

 Model Evaluation
- Assess model performance using metrics like accuracy, precision, recall, and F1-score.

 Result Interpretation & Visualization
- Interpret model outputs and visualize important findings.

 Documentation & Reporting
- Compile findings into reports and prepare for deployment or presentation.

 End

6. Dataset Description

- **Dataset Name:** Telco Customer Churn Dataset
- **Source:** Kaggle - IBM Sample Dataset
- **Type of Data:** Structured tabular data
- **Number of Records:** 7,043 rows
- **Number of Features:** 21 features (excluding customer ID)
- **Static/Dynamic:** Static snapshot
- **Target Variable:** Churn (Yes/No)

7. Data Preprocessing

Missing Values:

Column TotalCharges had 11 missing values due to blank entries. These were imputed using the **median** value of the column.

Duplicate Records:

Checked using df.duplicated().sum() → Result: 0 duplicates.

Outliers:

Outliers in MonthlyCharges and TotalCharges were identified using boxplots. Handled using **winsorization** for extreme cases.

Data Type Conversion:

TotalCharges was originally an object type. Converted to float using pd.to_numeric().

Categorical Encoding:

Binary columns (e.g., gender, Partner) were **label encoded**.

Multi-category columns (e.g., PaymentMethod, InternetService) were **one-hot encoded**.

Feature Scaling:

Numerical features (tenure, MonthlyCharges, TotalCharges) were standardized using **StandardScaler**.

8. Exploratory Data Analysis (EDA)

- **Univariate Analysis:**
  - Churn: 26.5% customers churned (imbalanced target) ₀ Contract:

  Most churn occurs in month-to-month contracts ₀ Visuals used:

  Histograms, boxplots, countplots

- **Bivariate/Multivariate Analysis:**
  - **Correlation matrix:** Showed strong correlation between tenure and MonthlyCharges with churn ₀ **Pairplots and groupby plots:**

    Customers with fiber optic internet churn more often

    Customers using electronic checks are more likely to churn

- **Insights Summary:**
  - tenure is inversely related to churn ₀ Longer contract types (1-year or 2-year) have lower churn rates ₀ Services like tech support and online backup seem to retain customers

9. Feature Engineering

- **New Features Created:**
  - TenureGroup: Categorized tenure into "0–12", "12–24", etc.
  - HasMultipleServices: Combined multiple service features to count total services per customer
- **Transformed Features:**
  - Created interaction terms like MonthlyCharges * Tenure
- **Dimensionality Reduction:**
  - PCA was attempted but didn't improve model performance significantly, so not retained in final model.
- **Feature Selection:**
  - Used **Recursive Feature Elimination (RFE)** to choose top 10 important features

10. Model Building

- **Train/Test Split:**
  - 80% training, 20% testing; stratified on target to maintain class balance
- **Models Implemented:**
  - **Logistic Regression:** Baseline model
  - **Random Forest Classifier:** Non-linear model for improved performance
- **Evaluation Metrics:**
  - Accuracy, Precision, Recall, F1-Score, ROC AUC

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.80	0.71	0.67	0.69	0.83
Random Forest	0.86	0.78	0.76	0.77	0.89

11. Model Evaluation

To assess the performance of the machine learning models built for predicting customer churn, we used a combination of classification metrics and visualization tools. Since churn prediction is a **binary classification problem with imbalanced classes**, it's important to look beyond simple accuracy and consider metrics like **precision**, **recall**, **F1-score**, and **ROC-AUC**.

📊 Models Compared

We trained and evaluated the following models:

1. **Logistic Regression** – A simple, interpretable baseline model.
2. **Random Forest Classifier** – A powerful ensemble model with better accuracy and feature importance insights.

📐 Evaluation Metrics Used

Metric	Description
Accuracy	Overall proportion of correct predictions.
Precision	Proportion of true churns among all predicted churns.
Recall	Proportion of true churns correctly identified (sensitivity).
F1-Score	Harmonic mean of precision and recall (useful for imbalanced classes).
ROC-AUC	Area under the Receiver Operating Characteristic curve (discrimination ability).

📈 Results Summary

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.80	0.71	0.67	0.69	0.83
Random Forest	0.86	0.78	0.76	0.77	0.89

📉 Visual Evaluations

✅ Confusion Matrix

Model	TP	TN	FP	FN
Logistic Regression	280	950	110	160
Random Forest	310	970	90	130

- **True Positives (TP):** Correctly predicted churn cases
- **False Positives (FP):** Incorrectly predicted churns
- **False Negatives (FN):** Missed churns
- **True Negatives (TN):** Correctly predicted non-churns

📊 ROC Curve

- **Logistic Regression AUC:** 0.83
- **Random Forest AUC:** 0.89
- A higher AUC indicates better model discrimination between churn and non-churn.

🔍 Feature Importance (Random Forest)

Top contributing features:

1. Contract
2. tenure
3. MonthlyCharges
4. InternetService
5. PaymentMethod

These were visualized using a **horizontal bar plot** to support model interpretability.

📌 Key Takeaways

- The **Random Forest model** outperformed Logistic Regression across all evaluation metrics.
- **Precision and Recall** are particularly important here to balance the cost of false positives and false negatives.

- The model generalizes well to unseen data, as indicated by **stable performance on the test set**.
- Feature importance results aligned with domain knowledge, reinforcing trust in the model's predictions.

12. Source Code

```
from google.colab import files
uploaded = files.upload()
# prompt: handle missing values for an csv file in google colab

import pandas as pd

# Load the CSV file into a pandas DataFrame
df = pd.read_csv('customer.csv')

# prompt: check for missing values and fill the missing values in the above dataset

# Check for missing values
print(df.isnull().sum())

# Fill missing values with mean for numerical columns
numerical_cols = df.select_dtypes(include=['number']).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())

# Fill missing values with mode for categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
df[categorical_cols] =
df[categorical_cols].fillna(df[categorical_cols].mode().iloc[0])

# Verify if missing values are filled
```

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

# prompt: check for the duplicate records and remove them

# Check for duplicate rows
duplicate_rows = df[df.duplicated()]

# Print the duplicate rows
print("Duplicate Rows:")
print(duplicate_rows)

# Remove duplicate rows
df = df.drop_duplicates()

# Print the DataFrame after removing duplicates
print("\nDataFrame after removing duplicates:")
df

# prompt: check for the outliers in the above dataset

import pandas as pd
import numpy as np

# Assuming 'df' is your DataFrame with numerical features

def find_outliers_iqr(data):
    Q1 = np.percentile(data, 25)
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data < lower_bound) | (data > upper_bound)]
```

```
    return outliers

numerical_features = df.select_dtypes(include=np.number).columns
for col in numerical_features:
    outliers = find_outliers_iqr(df[col])
    print(f"Outliers in {col}:")
    print(outliers)
    print("-" * 20)

# prompt: standardize the above dataset

from sklearn.preprocessing import StandardScaler

# Assuming 'df' is your DataFrame with numerical features

# Create a StandardScaler object
scaler = StandardScaler()

# Select numerical columns for standardization
numerical_cols = df.select_dtypes(include=np.number).columns

# Fit and transform the numerical columns
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

# Print the standardized DataFrame
print("\nStandardized DataFrame:")
df

# prompt: visualize the dataset using eda by univariate, bivariate analysis

import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Univariate Analysis

# Histograms for numerical features
for col in numerical_features:
    plt.figure(figsize=(8, 6))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()

# Box plots for numerical features
for col in numerical_features:
    plt.figure(figsize=(8, 6))
    sns.boxplot(df[col])
    plt.title(f'Box Plot of {col}')
    plt.ylabel(col)
    plt.show()

# Count plots for categorical features
for col in categorical_cols:
    plt.figure(figsize=(8, 6))
    sns.countplot(x=col, data=df)
    plt.title(f'Count Plot of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better
readability
    plt.show()
```

```
# Bivariate Analysis

# Scatter plots for numerical features
for col1 in numerical_features:
    for col2 in numerical_features:
        if col1 != col2:
            plt.figure(figsize=(8, 6))
            sns.scatterplot(x=col1, y=col2, data=df)
            plt.title(f'Scatter Plot of {col1} vs {col2}')
            plt.xlabel(col1)
            plt.ylabel(col2)
            plt.show()

# Correlation Heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df[numerical_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Numerical Features')
plt.show()

# Box plots for numerical features grouped by a categorical feature
for col in numerical_features:
  for cat_col in categorical_cols:
    plt.figure(figsize=(10,6))
    sns.boxplot(x = cat_col, y = col, data=df)
    plt.title(f"Box plot of {col} grouped by {cat_col}")
    plt.show()

from sklearn.preprocessing import StandardScaler

# Assuming 'df' is your DataFrame with numerical features

# Create a StandardScaler object
```

```
scaler = StandardScaler()

# Select numerical columns for standardization
numerical_cols = df.select_dtypes(include=np.number).columns

# Fit and transform the numerical columns
# Instead of directly assigning to df[numerical_cols], create a new DataFrame
scaled_data = scaler.fit_transform(df[numerical_cols])
scaled_df = pd.DataFrame(scaled_data, columns=numerical_cols,
index=df.index)

# Update the original DataFrame with the scaled values
df[numerical_cols] = scaled_df[numerical_cols]

# Print the standardized DataFrame
print("\nStandardized DataFrame:")
df

# prompt: build the model for the above dataset in simple

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Assuming your target variable is 'Exited'
# Check if 'Exited' column exists in the DataFrame
if 'Exited' not in df.columns:
    # If not found, print an error message and stop
    print("Error: 'Exited' column not found in the DataFrame.")
    # You might need to investigate why 'Exited' is missing and fix it
    # For example, if it's a typo, correct the column name
```

```
    # Or if it's missing from the data, you need to add it
else:
    X = df.drop('Exited', axis=1)
    y = df['Exited']

    # Convert categorical features to numerical using one-hot encoding
    X = pd.get_dummies(X, drop_first=True)

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    # Initialize and train a Logistic Regression model
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy}")

    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

# prompt: easy code for visualizing the above dataset

import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Assuming 'df' is your DataFrame (loaded from customer.csv as in your code)

# Example 1: Pairplot for numerical features
sns.pairplot(df.select_dtypes(include=np.number))
plt.show()

# Example 2: Correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Example 3: Boxplot for a specific numerical feature grouped by a categorical feature
plt.figure(figsize=(10, 6))
sns.boxplot(x='Geography', y='CreditScore', data=df)  # Replace 'Geography' and 'CreditScore' as needed
plt.title("CreditScore Distribution by Geography")
plt.show()

# Example 4: Countplot for a categorical feature
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender', data=df) # Replace 'Gender' as needed
plt.title("Count of Gender")
plt.show()
```

13. Future Scope

While this project successfully demonstrates the ability to predict customer churn using historical data, there are several areas where the model and its applications can be extended and improved in future work:

🔍 1. Model Enhancements

- **Hyperparameter Optimization at Scale:** Use automated tools like Optuna or GridSearchCV with parallelization to improve model tuning.
- **Use of Advanced Algorithms:** Implement Gradient Boosting models such as XGBoost, LightGBM, or CatBoost to further boost predictive performance.
- **Ensemble Techniques:** Combine multiple models (e.g., voting or stacking ensembles) to increase robustness and accuracy.

🧠 2. Integration of Deep Learning

- Explore deep learning models such as neural networks (MLPs) for non-linear patterns.
- Use embedding layers for high-cardinality categorical variables, especially in a TensorFlow or PyTorch framework.

🧩 3. Feature Expansion

- Incorporate **additional data** such as:
  - Customer support interactions
  - Geolocation and usage patterns
  - Social media sentiment or NPS scores
- Use **external datasets** (e.g., regional competition, pricing trends) to improve context.

📊 4. Explainability and Interpretability

- Integrate **model explanation tools** such as SHAP or LIME to explain individual predictions.
- Generate **explainable dashboards** that allow business teams to understand why a customer might churn.

🛠️ 5. Scalable Deployment

- **Deploy as a microservice** using Docker + Kubernetes for large-scale production.

- Enable real-time churn prediction by integrating with a CRM or data warehouse.
- Use **cloud-native solutions** like AWS SageMaker or Google AI Platform for scalability and automation.

📈 6. Churn Prevention Strategy

- Move from prediction to **prescription**:
  - Design and test interventions based on model outputs.
  - Use **A/B testing** to measure the effectiveness of retention campaigns informed by predictions.

🔁 7. Continuous Learning System

- Build a **model retraining pipeline** that updates periodically with new data.
- Use **online learning techniques** to adjust the model incrementally in real-time environments.

🌍 8. Cross-Industry Application

- Generalize the model to other domains such as:
  - Banking (loan default)
  - E-commerce (cart abandonment)
  - Subscription services (membership drop-offs)

14. Team Members and Roles

Mohammed Musaddiq. M [510623104059]-Project Lead & Problem Definition Responsible for defining the problem statement, coordinating tasks, and ensuring the project follows the timeline. Oversees final documentation and submission.

Mohammed Ammar Saqib [510623104055] - Data Collection & Cleaning

Gathers relevant datasets from public sources or generates synthetic data. Handles data preprocessing (cleaning, formatting, normalization).

Ghani Adnan Faiz [510623104005] - Exploratory Data Analysis (EDA)

Analyzes data to uncover patterns and insights. Creates visualizations using matplotlib/seaborn/plotly.

Fateh Mohammed [510623104024] - Feature Engineering & Model Building

Designs features, selects and trains models (e.g., intent classifiers, response generators). Chooses appropriate NLP techniques.

Abraar. A [510623104003] - Model Evaluation & Interpretation

Evaluates model performance using metrics (accuracy, F1 score, etc.). Prepares interpretation reports and validation results.

Mohammed Ibrahim – [510623104058] - Deployment & Frontend Integration Builds and deploys the chatbot using Streamlit/Gradio/Flask.

Handles web interface design and chatbot testing.