Table of Contents

AI-Powered Churn Prediction Assistant: A Proof of Concept for Marketing Teams

1. Executive Summary

Customer churn—the rate at which customers discontinue their service with a business—poses a significant threat to revenue and long-term growth. Being able to predict and understand the reasons behind churn is essential for any customer-focused organization.

This Proof of Concept (PoC) presents a solution that empowers the marketing team to predict customer churn using natural language. By leveraging a classical machine learning model and integrating it with a conversational interface powered by an open-source large language model (LLM), we bridge the gap between complex data science tools and business users.

The system allows marketing personnel to interact with a chatbot to inquire about churn risks and receive intuitive responses, without needing any technical expertise. The core components of the solution include:

- A churn classification model trained on customer behavioural and demographic data,
- A FastAPI-based backend to serve predictions,
- An open-source LLM to extract structured features from natural language inputs,
- A chatbot interface for seamless communication.

2. Business Motivation

Reducing customer churn is a top priority for companies that rely on recurring revenue. Losing customers not only impacts immediate revenue but also increases costs associated with acquiring new customers. Therefore, having timely insights into which customers are at risk of leaving enables proactive retention strategies.

Traditional churn prediction tools often require technical knowledge to operate, limiting their accessibility to business teams such as marketing. By integrating an LLM-based assistant, we democratize access to churn insights, allowing marketing professionals to simply ask questions in natural language and receive accurate, actionable responses.

This solution eliminates the dependency on technical intermediaries and enables faster decision-making. It directly supports the client's requirement of using open-source tools while ensuring transparency, interpretability, and ease of use for non-technical stakeholders.

3. Technical Objectives

This project aims to fulfil the following key technical goals, aligning with the client's specifications and practical needs:

1. **Churn Prediction Model**
   Develop a robust classification model that accurately predicts customer churn based on a diverse set of customer attributes. The model should be interpretable, efficient, and suitable for deployment in real-world scenarios.

2. **Natural Language Interface via Open-Source LLM**
   Integrate an open-source large language model (LLM) to serve as a bridge between human input and machine-readable model features. The LLM is responsible for:

   - Understanding user queries in natural language.
   - Extracting structured features required by the churn model.
   - Presenting the prediction and its explanation in a conversational tone.

3. **API and Backend Infrastructure**
   Build an API that manages the interaction between the chatbot and the prediction model. The backend handles session management, data validation, feature extraction, prediction computation, and response generation.

4. **Gradio-based Chatbot Interface**
   Deploy a user-friendly chatbot interface using Gradio that allows marketing team members to interact with the assistant in real-time. The chatbot can prompt users for missing data and iteratively build a complete feature profile.

5. **Missing Data Handling**
   Implement a conversational mechanism to collect missing information. If the user input lacks certain required features, the assistant will ask follow-up questions, merge responses using the LLM, and proceed only when sufficient data is available.

6. **Deployment and Demonstration**
   Provide a deployable version of the entire pipeline in a cloud-accessible environment to facilitate testing and demonstration by the client's team.

4. Dataset Summary

The dataset provided for this project consists of **7,043 observations** and **21 variables**, each representing characteristics of telecom customers and their churn status. Below is a summary of the key features:

- **Demographic and Account Features**
  - CustomerId: Unique identifier for each customer.
  - Gender: Male or Female.
  - Senior_Citizen: Whether the customer is a senior citizen (1 = Yes, 0 = No).
  - Is_Married: Indicates if the customer is married.
  - Dependents: Indicates if the customer has dependents.
- **Service and Usage Features**
  - Tenure: Number of months the customer has stayed with the company.
  - Phone_Service: Whether the customer has phone service.
  - Dual: Indicates dual line usage.
  - Internet_Service: Type of internet service (DSL, Fiber optic, No).
  - Online_Security, Online_Backup, Device_Protection, Tech_Support: Add-on internet services.
  - Streaming_TV, Streaming_Movies: Entertainment-related services.
- **Contract and Billing Information**

- Contract: Contract type (Month-to-month, One year, Two years).
- Paperless_Billing: Whether the customer receives a paperless bill.
- Payment_Method: Method of payment (Electronic check, Postal check, Bank transfer, Credit card).
- Monthly_Charges: The current monthly charge.
- Total_Charges: The total amount billed to the customer.

- **Target Variable**
  - Churn: Whether the customer has discontinued service (Yes or No).

This dataset provides a comprehensive view of customer behaviour and is suitable for training a supervised machine learning model to identify churn patterns.

5. Churn Prediction Model Development

5.1. Preprocessing

A comprehensive preprocessing pipeline was applied to prepare the data for modelling:

- **Handling Missing Values**:

  The total_charges column had missing values, which were imputed with **0**. This was a reasonable assumption since customers with missing total_charges typically had zero tenure, meaning they were new customers with no charges yet.

- **Box-Cox Transformation**:

  The total_charges feature was transformed using a **Box-Cox transformation** (with a +1 offset) to reduce skewness and approximate normality, which can help certain models converge faster and perform better.

- **Data Splitting**:

  A **stratified train-test split** was performed before any transformation or resampling to prevent data leakage and maintain class distribution. The test size was set to 20%.

- **Categorical Encoding**
  - **Ordinal Encoding** was used for binary and ordinal categorical variables (e.g., gender, senior_citizen, phone_service, etc.).
  - **One-Hot Encoding** was applied to nominal categorical variables like internet_service, contract, and payment_method.

- **Feature Scaling**:

  Numerical variables (tenure, monthly_charges, total_charges) were standardized using StandardScaler.

- **Pipeline Integration**:

  All transformations were integrated using a ColumnTransformer, allowing streamlined and reproducible preprocessing.

5.2. Feature Engineering

- **Class Imbalance Handling**:
  - The dataset exhibited class imbalance in the target variable (churn).
  - **SMOTEENN** (a combination of SMOTE and Edited Nearest Neighbors) was used to both oversample the minority class and clean noisy examples from the majority class, enhancing class balance and model generalization.

5.3. Model Choice

A wide range of machine learning models were evaluated to identify the most effective churn predictor:

- **Logistic Regression**
  - Baseline linear model.
  - Chosen for its simplicity, speed, and interpretability.
- **Random Forest**
  - Ensemble of decision trees.
  - Handles nonlinear relationships and interactions well.
- **Balanced Random Forest**
  - Modified version of Random Forest using internal resampling.
  - Better suited for imbalanced datasets.
- **XGBoost**
  - Gradient boosting with high performance.
  - Fast, scalable, and supports GPU acceleration.
  - Excellent for handling structured/tabular data.
- **LightGBM**
  - Similar to XGBoost but often faster with large datasets.

    - Uses leaf-wise tree growth for improved accuracy.
- **CatBoost**
    - Efficient gradient boosting with built-in categorical handling.
    - Fast and accurate with minimal tuning.
- **K-Nearest Neighbors (KNN)**
    - Simple and intuitive.
    - Acts as a non-parametric benchmark.
- **MLP (Multi-Layer Perceptron) using TensorFlow**
    - A deep learning approach with dense layers and dropout.
    - Tested for its capacity to capture complex patterns in data.

Each model was configured with class balancing techniques where applicable (e.g., class_weight='balanced' or scale_pos_weight). The use of multiple models allows comparing different trade-offs between performance, interpretability, and computational efficiency.

5.4. Training & Evaluation

- **Data Splitting**:

  An 80/20 train-test split was performed using stratification to maintain the original class distribution of the churn variable across both sets.

- **Resampling Strategy**:

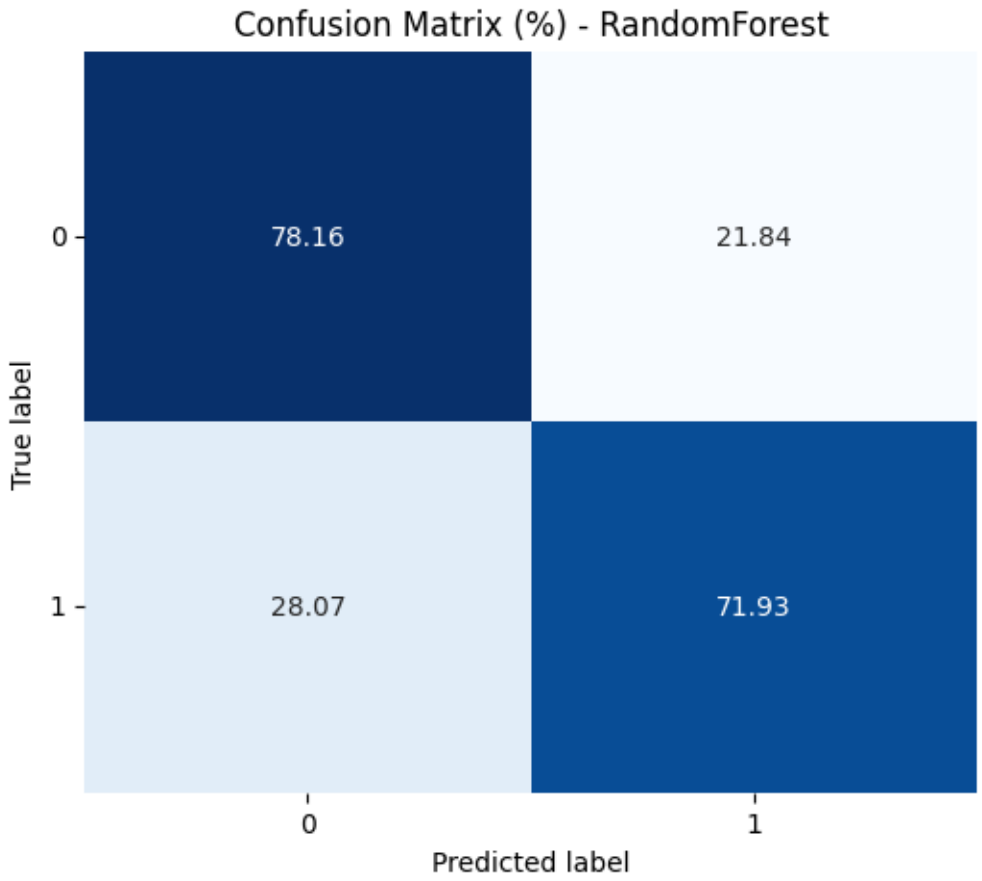
  All models were trained and evaluated twice:

    - **With SMOTEENN**: Applied to the training data to address class imbalance using oversampling and cleaning techniques.
    - **Without SMOTEENN**: Models were trained on the original, imbalanced dataset.

  After comparison, it was found that **models trained without SMOTEENN consistently outperformed** their counterparts in terms of accuracy and weighted F1 score. Therefore, the final evaluation and model selection were based on results **without applying SMOTEENN**.

- **Evaluation Metrics**:
  - **Accuracy**: Used as a general performance indicator.
  - **Precision, Recall, F1 Score**: Critical due to the imbalanced nature of the dataset.
  - **Weighted F1 Score**: Used as the primary criterion to account for label imbalance.
  - **Confusion Matrix (in %)**: Used to visualize the class-wise prediction performance of each model.
- **Model Selection**:

  Based on the evaluation results, the **Random Forest** classifier trained on the original (non-resampled) data provided the best performance, achieving a **weighted F1 score of 0.7741**.
  This model was selected as the final model to be downloaded and deployed.



```
Training RandomForest...
RandomForest - Test Accuracy: 0.7651
              precision    recall  f1-score   support

           0       0.89      0.78      0.83      1035
           1       0.54      0.72      0.62       374

    accuracy                           0.77      1409
   macro avg       0.71      0.75      0.72      1409
weighted avg       0.79      0.77      0.77      1409
```

5.5. Threshold Selection Summary for RandomForest Model

After evaluating different classification thresholds, **0.5** and **0.6** stand out as the best choices based on overall performance metrics:

Key Points:

- **Threshold 0.6** offers the highest weighted F1 score and a balanced trade-off between precision and recall for both classes.
- **Threshold 0.5** provides slightly better recall for the minority class (class 1), with very competitive overall performance.
- Both thresholds ensure good accuracy and balanced class-wise performance, making them preferable over others.

6. LLM Integration

A few open-source LLMs were tested; however, due to limited computational resources, only models that could be run on Kaggle were used, as local execution was not feasible. Among the tested options, **Mistral 7B** provided the best performance and was selected for integration.

6.1. Feature Extraction via LLM

- The LLM interprets natural language inputs describing customer characteristics.
- It maps this unstructured input to a predefined JSON schema used by the churn prediction model.
- The extraction logic is robust to varied phrasing, enabling flexibility in user input.

6.2. Explanation Generation via LLM

- After prediction, the raw output (Yes/No + probability) is translated into plain English using the LLM.
- Explanations aim to communicate why a customer is likely or unlikely to churn in a way that is digestible to non-technical users.

6.3. Handling Missing Information

- If any required features are missing from the initial input, the assistant:
    - Detects the missing fields.
    - Prompts the user for additional details.
    - Merges new replies with previous session data using the LLM.
    - Re-runs extraction and prediction once complete.

7. System Architecture and Deployment

This section summarizes the complete pipeline for the churn prediction assistant, including architecture design, model orchestration, and deployment strategy.

7.1. Architecture Overview

The system is composed of the following modular components:

- **Pre-trained Churn Prediction Model:**

  A Random Forest model trained locally on engineered features from the customer dataset. It was selected based on superior performance (Weighted F1 Score: 0.7741) without SMOTEENN resampling.

- **Open-Source LLM (Mistral 7B):**

  Used for natural language processing tasks including feature extraction and explanation generation. Due to local resource limitations, the model is executed on Kaggle, which provides the necessary GPU infrastructure.

- **FastAPI Backend:**

  Acts as the central controller:

  - Manages user sessions and coordinates feature extraction, prediction, and explanation tasks.
  - Communicates with the remote LLM and local model.

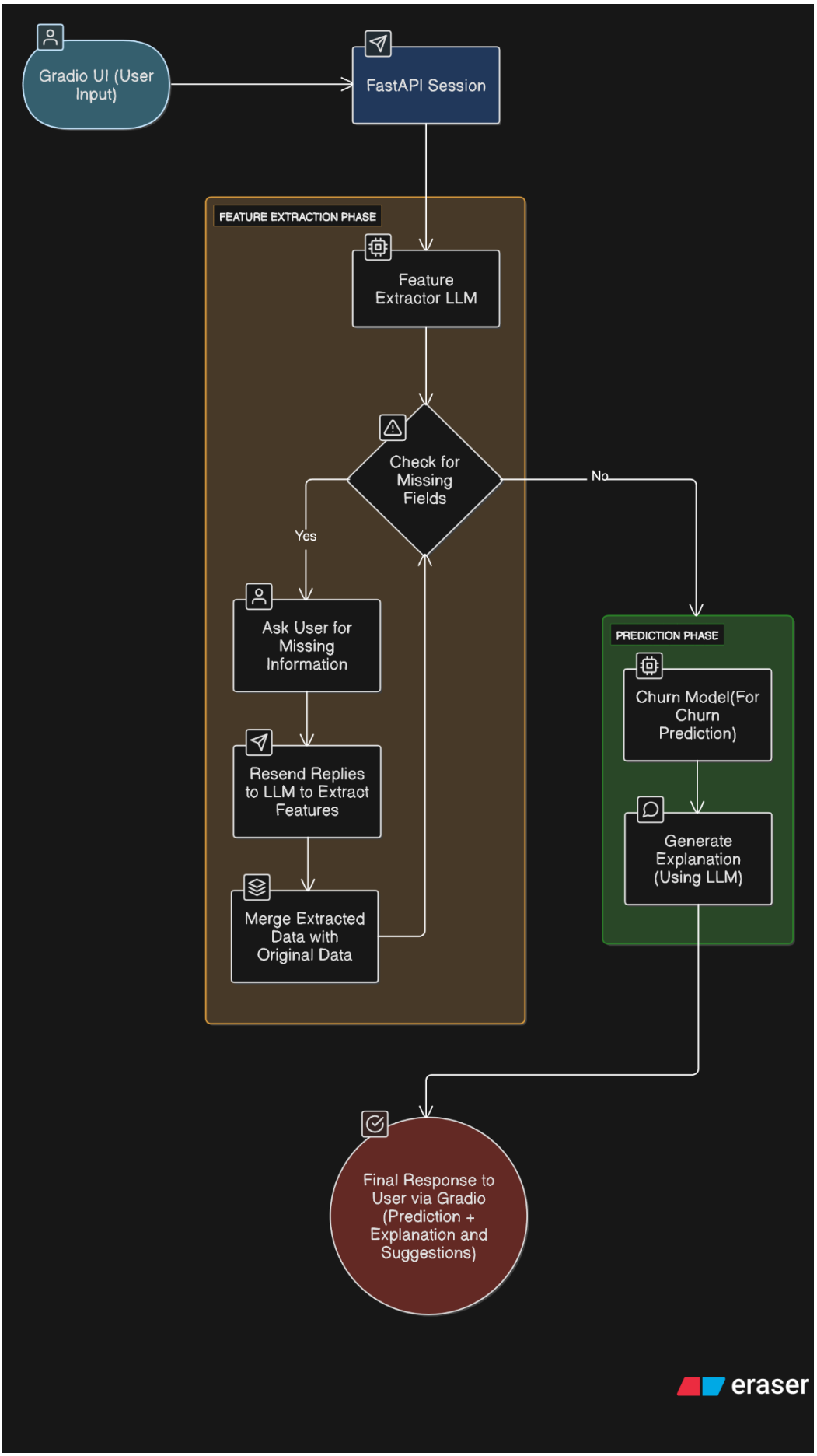
- **Gradio Chat Interface:**

  Provides a user-friendly conversational environment for the marketing team to:

  - Enter customer descriptions in free-form language.
  - Receive predictions and interpretive responses.
  - Supply missing details through natural follow-ups.

7.2. Data and Interaction Flow

1. The user inputs a customer description via the Gradio interface.
2. The backend sends this to the LLM (hosted on Kaggle) for feature extraction.
3. If required fields are missing, the system prompts the user for clarification and merges new responses.
4. Once all features are available, the local churn prediction model makes a prediction.
5. The result is sent back to the LLM for explanation generation.
6. The Gradio chatbot displays the final output to the user.


Gradio UI (User Input)
FastAPI Session
FEATURE EXTRACTION PHASE
Feature Extractor LLM
Check for Missing Fields
No
Yes
Ask User for Missing Information
Resend Replies to LLM to Extract Features
Merge Extracted Data with Original Data
PREDICTION PHASE
Churn Model(For Churn Prediction)
Generate Explanation (Using LLM)
Final Response to User via Gradio (Prediction + Explanation and Suggestions)
eraser

8. Business and Technical Justification

This section outlines how the proposed solution meets the client's needs both from a business standpoint and a technical perspective.

8.1. Business Motivation

- **Natural Language Accessibility**:
  Marketing teams can interact with the system using plain English, reducing the barrier to using machine learning models.
- **Decision Support**:
  Clear churn explanations enable marketing teams to tailor interventions and reduce customer attrition proactively.
- **Proof of Concept Demonstration**:
  The system demonstrates technical feasibility and user experience flow for a more comprehensive product.

8.2. Technical Justification

- **Modular Pipeline**:
  Each component—LLM, backend API, model, and UI—is modular, making maintenance, upgrades, and scaling easier.
- **Use of Open-Source Tools**:
  All models and dependencies comply with the client's request to avoid closed-source or proprietary systems.
- **Robust Feature Extraction**:
  The LLM is capable of understanding a wide variety of user inputs and accurately mapping them to model features.
- **Prompt-based Missing Data Resolution**:
  The assistant intelligently asks users to fill in missing fields and continues the flow seamlessly.
- **Reusability and Extensibility**:
  The codebase and architecture are designed to be extended in future deployments for other business cases or domains.

9. Conclusion & Future Work

9.1. Conclusion

This project successfully delivers a Proof of Concept (PoC) for a customer churn prediction assistant powered by a classical machine learning model and enhanced with a natural language interface. By combining the accuracy of structured churn prediction with the flexibility of a conversational AI layer, the system enables marketing teams to engage directly with analytical insights without technical barriers.

Key achievements include:

- Development of a reliable churn classification model.
- Seamless integration between an open-source LLM and a predictive backend.
- A Gradio-based user interface, supported by FastAPI and exposed via Ngrok.
- Smart feature extraction and real-time data gathering through a chatbot dialogue.
- Clear, understandable prediction outputs with personalized explanations.

The solution demonstrates the feasibility of deploying ML-driven insights in an accessible, user-friendly manner and sets the foundation for broader adoption within customer-facing business units.

9.2. Future Work

Several opportunities exist to further enhance this prototype:

1. **Session Management and Multi-User Support**:
    - Implement persistent session IDs and dynamic session handling using a database or cache layer (e.g., Redis) to support multiple concurrent users.
2. **Authentication and Security**:

- Add user authentication and permission handling to protect endpoints and manage usage securely.

3. **Model Improvement**:
    - Explore advanced algorithms or deep learning models to increase prediction performance.
    - Incorporate time-based or behavior-based features for improved accuracy.
4. **Feedback Loop and Model Retraining**:
    - Integrate feedback from user actions (e.g., intervention success or churn confirmation) to create a continuous learning loop.
5. **Full Cloud Deployment**:
    - Containerize the backend using Docker and deploy the entire system on cloud platforms like AWS, GCP, or Azure for production-level scalability.
6. **Enhanced NLP Handling**:
    - Fine-tune the LLM on domain-specific dialogues to improve feature extraction and conversation flow.
7. **Dashboard Integration**:
    - Add a dashboard for marketing managers to view aggregate churn trends, generate reports, and track system usage.