

AI-Powered Assistant for Churn Prediction

1. Project Overview

This project implements an AI-powered churn prediction assistant for telecom customers. It combines:

- Data preprocessing & machine learning for churn classification (Logistic Regression).
- FastAPI backend for serving predictions.
- LLM-powered chatbot for natural language input processing.
- Streamlit frontend for interactive user experience.

The assistant can:

- Accept structured inputs directly via form.
- Accept unstructured natural language descriptions via chatbot, convert them to structured features using an LLM, and return churn predictions.

2. Dataset & Preprocessing

2.1 Dataset

The dataset used is Telco Customer Churn containing telecom customer demographics, services, and billing details, along with a churn label.

2.2 Data Cleaning & Preparation

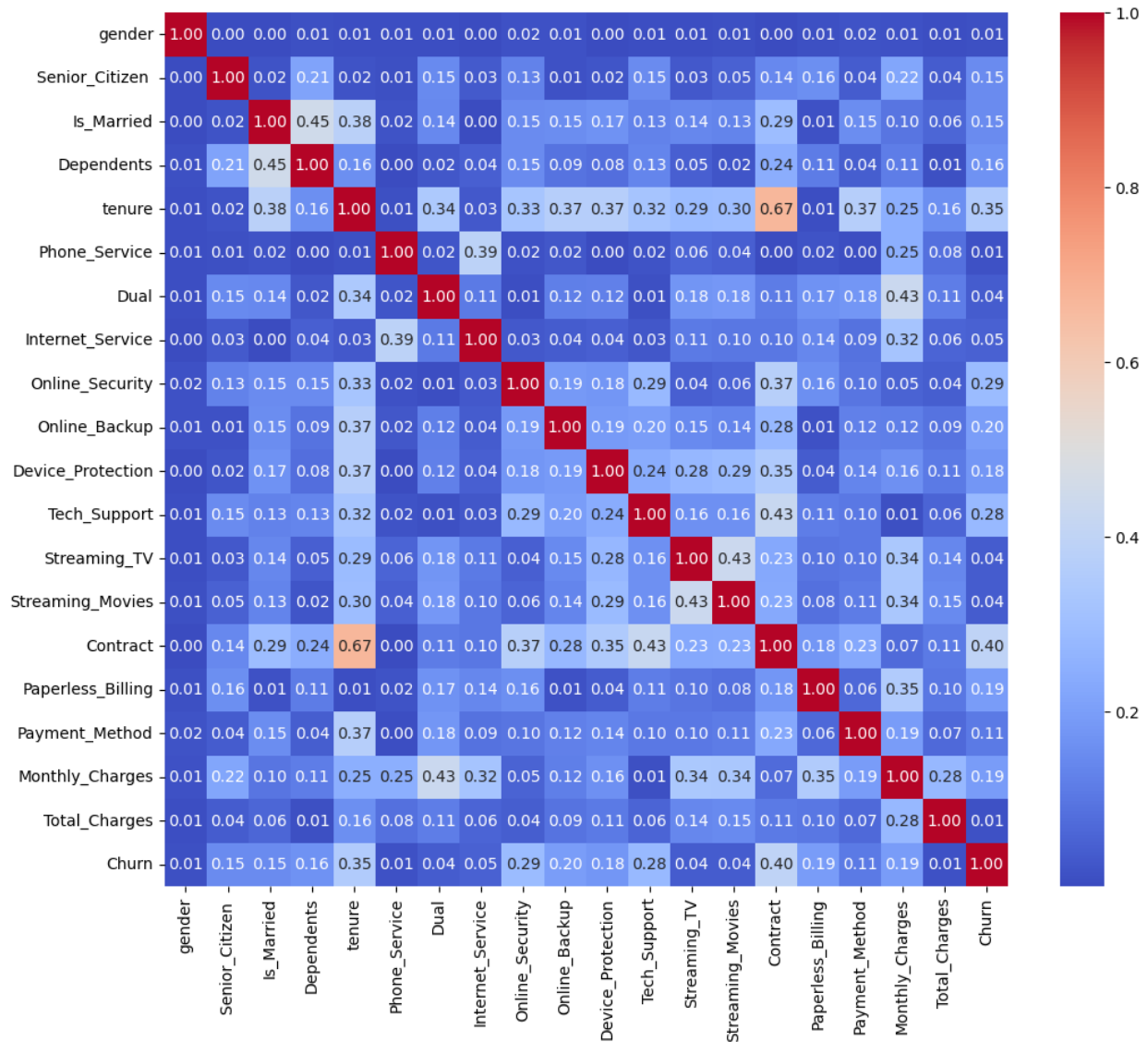
- Removed unnecessary columns (customerID).
- Stripped whitespace from column names.

2.3 Encoding Categorical Features

Label encoding was applied to categorical columns where all encoders were stored for use during prediction to ensure consistent transformation.

2.4 Feature Selection

- Correlation analysis was performed to drop features with correlation < 0.04 with churn.
- Dropped: (gender, Phone_Service, Dual, Streaming_TV, Streaming_Movies, Total_Charges)



3. Model Development

The churn dataset used in this project is imbalanced, meaning the number of “No Churn” customers greatly outnumbers “Churn” customers.

- **No Churn (0)** → Majority class (73%)
- **Churn (1)** → Minority class (27%)

3.1 Impact of Class Imbalance

In classification, an imbalanced dataset can cause the model to:

- Favor predicting the majority class.
- Have misleading accuracy (e.g., always predicting "No Churn" could give 73% accuracy without learning anything useful).
- Reduce performance on the minority class, which is actually the class of interest here.

For churn prediction, we care about identifying customers likely to churn, so detecting the minority class accurately is crucial.

Thus, balancing techniques were considered.

3.2 Train-Test Split & Scaling

3.3 Model Tested

- Multiple models were tested (SVM, Naive Bayes, Logistic Regression).

A. Support Vector Machine (SVM)

- **Purpose:** Find a hyperplane that best separates churners from non-churners in high-dimensional space.
- **Strengths:**
 - Works well for small-to-medium datasets.
 - Can handle non-linear decision boundaries.
- **Weaknesses for this problem:**
 - Sensitive to imbalanced data tends to bias toward the majority class.
 - Higher training time.
 - Model outputs are not probabilistic by default (less interpretable for business decisions like churn risk).

Performance: Achieved moderate accuracy, but recall for the minority class was low, meaning many churners were missed.

B. Naive Bayes (GaussianNB)

- **Purpose:** Probabilistic classifier based on Bayes' theorem with the assumption of feature independence.
- **Strengths:**
 - Computationally efficient.
 - Produces probabilistic outputs.

- **Weaknesses for this problem:**

- Independence assumption rarely holds for customer churn data (features like Contract length and Tenure are correlated).
- Tends to underperform when decision boundaries are complex.

Performance:

Fast training, but lower overall accuracy and especially weaker AUC compared to Logistic Regression.

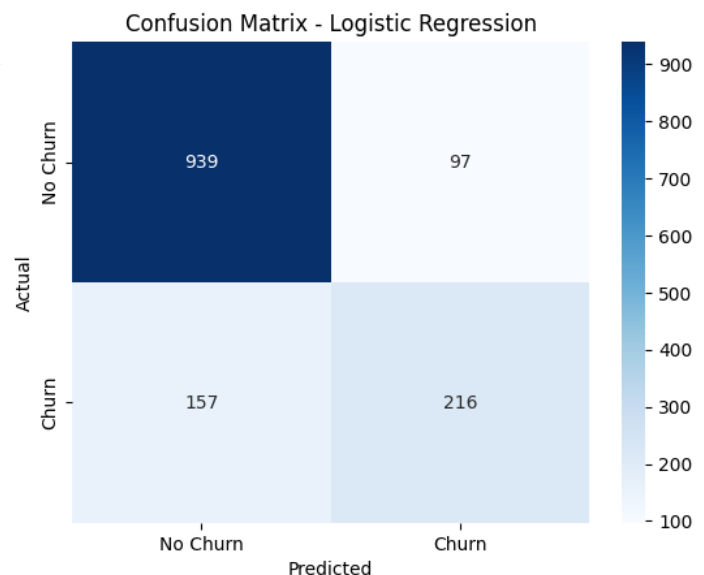
C. Logistic Regression (Final Choice)

- **Purpose:** Linear model that predicts probabilities using the logistic function.
- **Why Selected:**
 - Well-suited for binary classification with interpretability.
 - Probabilistic output (useful for ranking customers by churn risk).
 - Coefficients provide insight into feature importance and impact direction.
- **Strengths:**
 - Easy to interpret.
 - Works well on linearly separable or near-linearly separable problems.
 - Fast to train and predict.
- **Weaknesses:**
 - Assumes a linear relationship between features and log-odds of the outcome.
 - Without balancing, suffers from the same imbalance bias.

➤ **Before upsampling**

↔ Accuracy: 0.7459190915542938

Classification Report:					
	precision	recall	f1-score	support	
0	0.92	0.72	0.81	1036	
1	0.51	0.82	0.63	373	
accuracy			0.75	1409	
macro avg	0.72	0.77	0.72	1409	
weighted avg	0.81	0.75	0.76	1409	

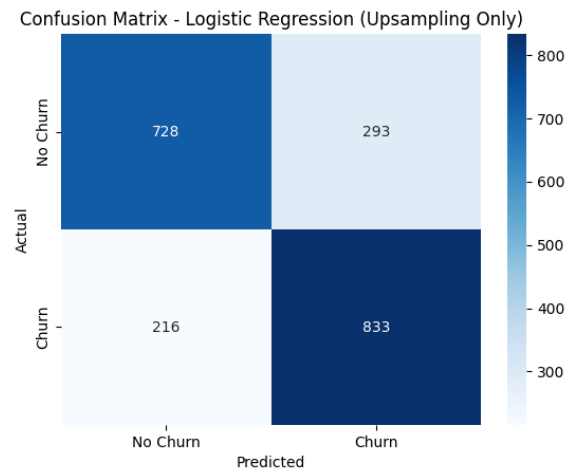


➤ After upsampling

➡ Accuracy: 0.7459190915542938

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.72	0.81	1036
1	0.51	0.82	0.63	373
accuracy			0.75	1409
macro avg	0.72	0.77	0.72	1409
weighted avg	0.81	0.75	0.76	1409



3.4 Model Saving

After training the final Logistic Regression model on the upsampled dataset, it was saved to a file for later use in the API and Streamlit application.

In addition to the model itself, two other important components were saved:

1. **Encoders** – These contain the mapping from categorical feature values to numerical values, ensuring that any new data provided to the model during prediction is transformed in the same way as the training data.
2. **Feature Order** – The exact sequence of features used when the model was trained. Maintaining this order is critical so that the model receives inputs in the correct structure during inference.

The three components trained model, label encoders, and feature order were packaged together and stored in a serialized .pkl file named log_reg_with_encoders.pkl using the Joblib library.

4. Backend API (FastAPI)

4.1 Purpose

- Serve predictions via /predict endpoint for structured inputs.
- Handle /chat endpoint for natural language queries using LLM.

4.2 Preprocessing for API

Ensures:

- Missing features are filled with defaults.
- Consistent label encoding using saved encoders.

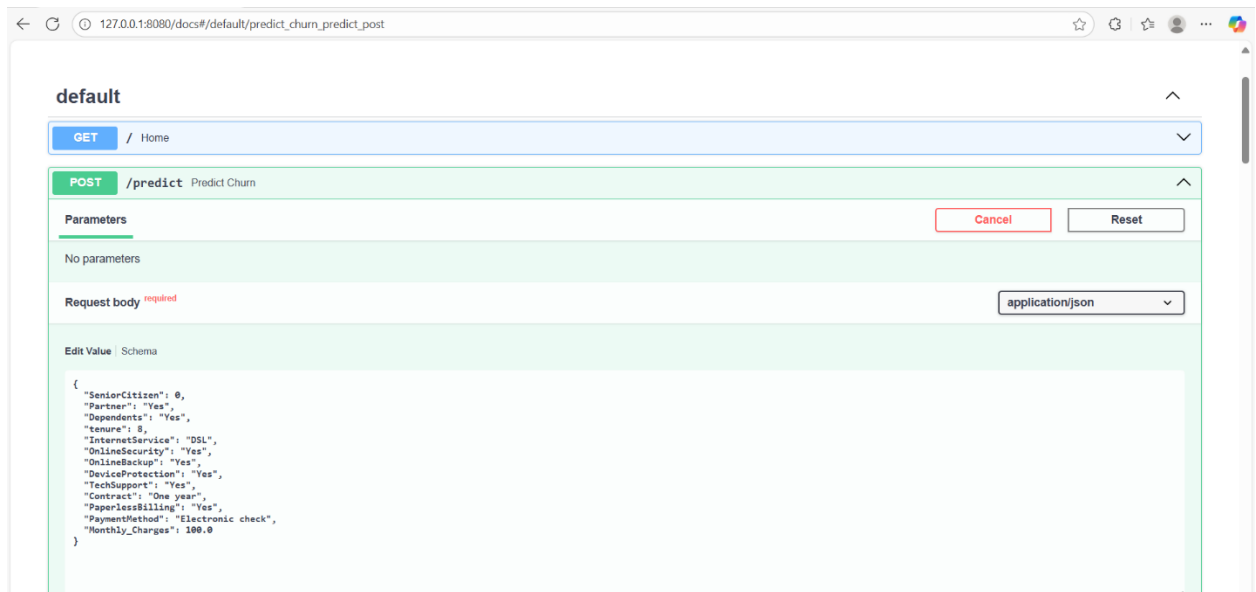
4.3 Insights Generation

Generates recommendations based on customer profile & prediction result.

4.4 Structured Input Test – */predict* Endpoint

It accepts a JSON payload containing structured feature values.

➤ Example request:

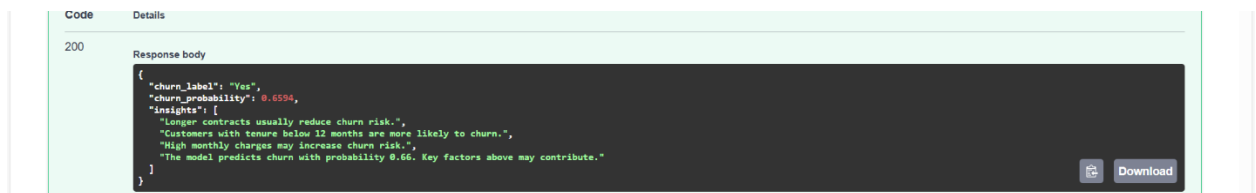


The screenshot shows a REST client interface with the following details:

- URL:** 127.0.0.1:8080/docs#/default/predict_churn_predict_post
- Method:** POST
- Path:** /predict (Predict Churn)
- Parameters:** No parameters
- Request body:** application/json
- Request body content:**

```
{
  "SeniorCitizen": 0,
  "Partner": "Yes",
  "Dependents": "Yes",
  "tenure": 8,
  "InternetService": "DSL",
  "OnlineSecurity": "Yes",
  "OnlineBackup": "Yes",
  "DeviceProtection": "Yes",
  "TechSupport": "Yes",
  "Contract": "One year",
  "PaperlessBilling": "Yes",
  "PaymentMethod": "Electronic check",
  "Monthly_Charges": 100.0
}
```

➤ Example Response:



The screenshot shows the response details in a REST client:

- Status:** 200
- Response body:**

```
{
  "churn_label": "Yes",
  "churn_probability": 0.6594,
  "insights": [
    "Longer contracts usually reduce churn risk.",
    "Customers with tenure below 12 months are more likely to churn.",
    "High monthly charges may increase churn risk.",
    "The model predicts churn with probability 0.66. Key factors above may contribute."
  ]
}
```
- Buttons:** Download

4.5 Natural Language Test – */chat* Endpoint

The */chat* endpoint takes free-text customer descriptions, processes them through the LLM to extract structured features, and then returns predictions.

➤ Example request:

POST /chat Chat With Model

Parameters Cancel Reset

No parameters

Request body *required* application/json

Edit Value Schema

```
{ "message": "The customer is a 65-year-old senior citizen, has fiber optic internet, no tech support, monthly charge is 90 dollars, tenure is 8 months." }
```

➤ Example Response:

200

Response body

```
{  "churn_label": "Yes",  "churn_probability": 0.812,  "extracted_features": {    "SeniorCitizen": 0,    "Partner": "No",    "Dependents": "No",    "tenure": 8,    "InternetService": "Fiber optic",    "OnlineSecurity": "No",    "OnlineBackup": "No",    "DeviceProtection": "No",    "TechSupport": "No",    "Contract": "Month-to-month",    "PaperlessBilling": "No",    "PaymentMethod": "Credit card (automatic)",    "MonthlyCharges": 90  },  "insights": [    "Month-to-month contracts are associated with higher churn risk.",    "Customers with tenure below 12 months are more likely to churn.",    "High monthly charges may increase churn risk.",    "The model predicts churn with probability 0.81. Key factors above may contribute."  ]}
```

Download

5. Chatbot with LLM (Groq API)

5.1 Purpose

- Convert unstructured text to structured JSON features.
- Enforce strict format for ML model input.

5.2 Feature Extraction

- A predefined list of expected features is used as a template. These include all the key attributes required by the machine learning model.
- When a user provides a free-text description (for example, "This customer has been with us for 8 months, uses fiber optic internet, pays \$90 per month, and has no tech support"), the LLM is instructed through a prompt to:
 1. Identify the relevant details in the message.
 2. Match those details to the expected feature names.
 3. Output the result as a strictly formatted JSON object containing all expected features, with missing values replaced by sensible defaults.

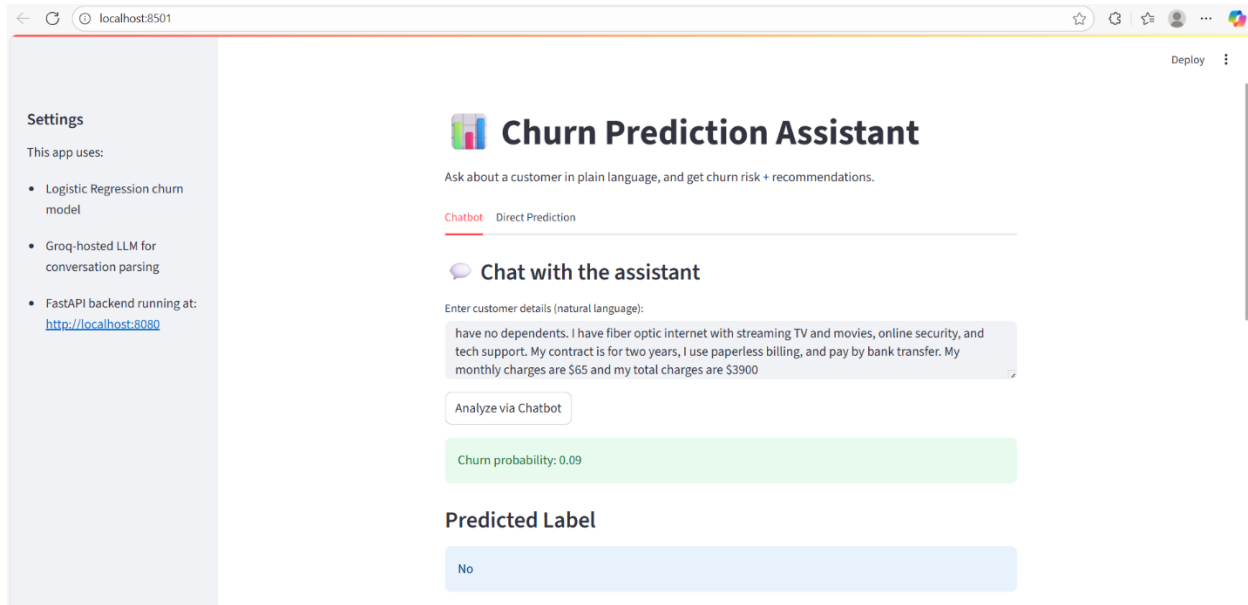
This ensures that even if the user's message is unstructured or incomplete, the backend still receives all required fields in the correct format for prediction.

5.3 Output Handling

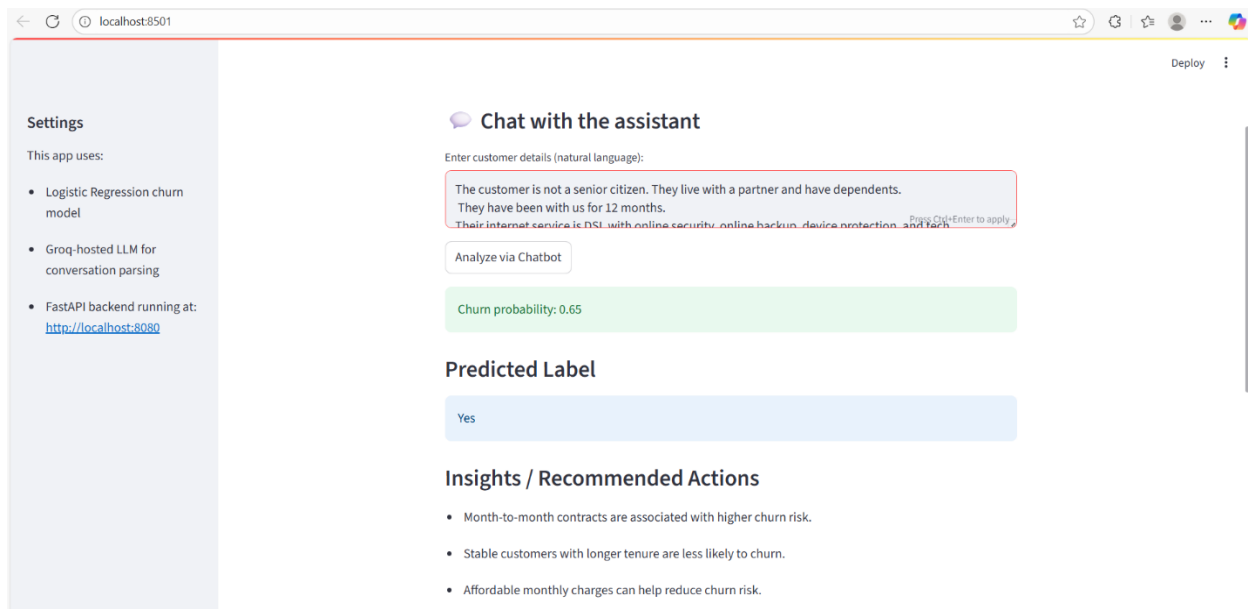
- If model misses features, defaults are added.
- Returns structured dictionary for API processing.

6. Frontend (Streamlit)

6.1 Chat Mode



a



6.2 Form Mode

Settings

This app uses:

•

Logistic Regression churn model

•

Groq-hosted LLM for conversation parsing

•

FastAPI backend running at: <http://localhost:8080>

Settings

This app uses:

•

Logistic Regression churn model

•

Groq-hosted LLM for conversation parsing

•

FastAPI backend running at: <http://localhost:8080>

Churn Prediction Assistant

Ask about a customer in plain language, and get churn risk + recommendations.

Chatbot

Direct Prediction

Direct Form Input

Senior Citizen

0

Partner

Yes

Dependents

Yes

Tenure (months)

12

Internet Service

DSL

Online Security

Yes

Online Backup

Yes

Device Protection

Yes

Tech Support

Yes

Contract

Month-to-month

Paperless Billing

Yes

Payment Method

Electronic check

Monthly Charges

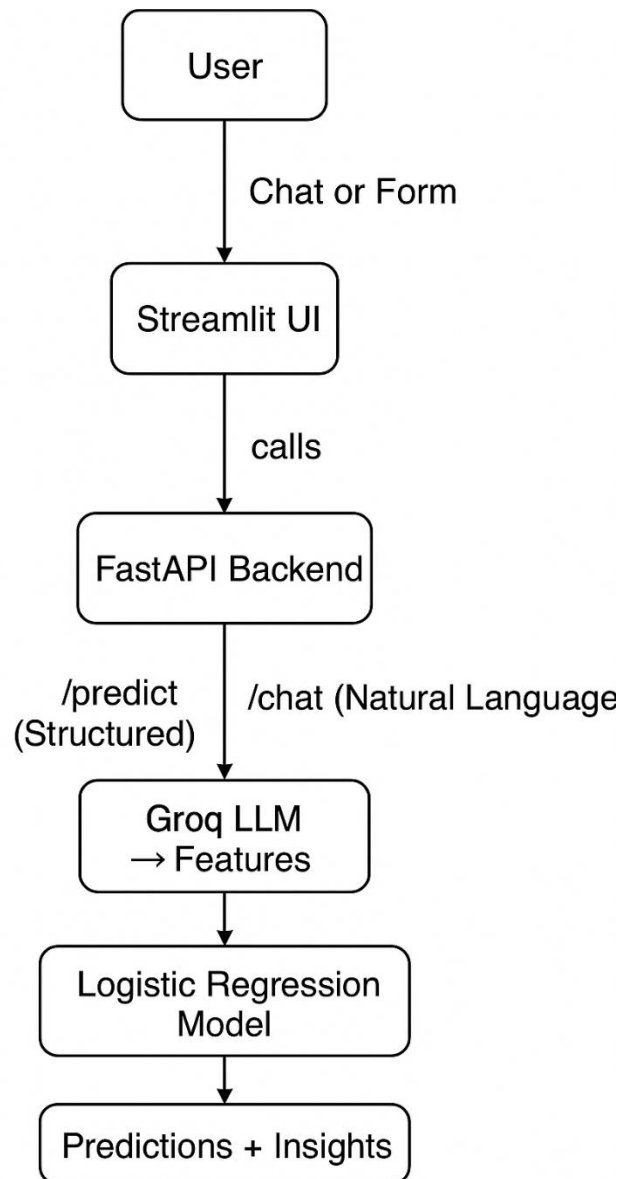
70.00

Predict

Churn probability: 0.65

Predicted Label: Yes

7. System Architecture Diagram



8. Final Deliverables

1. **Classification Notebook** – Data cleaning, model training, evaluation, and saving model.
2. **FastAPI Backend** – Endpoints for predictions and chatbot.
3. **Chatbot Module** – LLM-powered natural language parser.
4. **Streamlit Frontend** – Interactive web application.
5. **Screenshots of API and chatbot outputs.**