Churn Predictor Web Application: Software Developer

Sepehr Davanipur
sdavanipur@gmail.com
University of Trieste
Trieste, Friuli Venezia Giulia, Italy

Vishal Nigam
University of Trieste
Trieste, Friuli Venezia Giulia, Italy

Muhammad Mubashar Shahzad
University of Trieste
Trieste, Friuli Venezia Giulia, Italy

Abstract

This report outlines the development of the User Interface (UI) for the Churn Predictor Web Application. As the Software Developer, I was responsible for designing and implementing a clean, interactive front-end using Streamlit in Python [2]. The UI collects customers' input data, communicates with the machine learning backend, and presents churn prediction results clearly. The goal was to ensure usability, responsiveness, and accessibility for non-technical users as well as maintainability and scalability in order leave enough space for future developments and extensions.

Keywords

Web Application, User Interface, Streamlit, UML Diagram

ACM Reference Format:
Sepehr Davanipur, Vishal Nigam, and Muhammad Mubashar Shahzad. 2025. Churn Predictor Web Application: Software Developer. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction and Problem Statement

Managers of bank services usually face the unprecedented problem of the customers leaving their credit card services (also known as churning). In this regard, it is ideal for them to have at their disposal an application that proficiently performs churn prediction so they can take actions proactively. One of the best procedure that can be used to design such application is usage of Machine Learning (ML) models and algorithms and train them on the available datasets in order to use them for future predictions. Furthermore, the application has to be user friendly and accessible for non-technical users so that they can simply and intuitively insert the data they have about their customers and receive relevant predictions either in the form of the likelihood or yes/no output that notifies them about potential churning (in this project, we focused on the likelihood of churning as opposed to have just a yes/no output because it could be more informative). Having all that said, the ideal setting for this approach is assigning the backend of the application or software to a machine learning model and the front-end to the Graphical User Interface (GUI). For machine learning part of the project, we utilized a highly rated open dataset from Kaggle, containing behavioral, transactional, and demographic information of banking clients which is titled "Credit Card customers (Predict Churning customers)" [1]. Therefore, our software development approach can be assigned to three different roles including: Data Engineer, Software Engineer and Software Developer; this setting is the structure that our team has taken for this project and throughout this report, I will be reporting the works that I have done for this project as the Software Developer. Throughout this report, I will refer to our Churn Predictor Web Application as Application (with capital "A"). Altogether, the application that we designed in this project, allows bank managers and other system users to insert their customers' data and get as the output the likelihood of churning. The goal in short was to create a front-end layer that is intuitive, well-structured and seamlessly integrated with the ML kernel running in the backend.

1.1 Functional Requirements

Our Application should be able to take as input the following features which are required by the backend machine learning model. Input features can be divided into three main sub-categories including demographics (such as age), account information (such as card category) and transaction behavior (such as credit limit). There is also the Attrition Flag which shows if the account is still active or the customer has churned. Below is a full list of the features with a brief description for each.

CLIENTNUM: A unique identifier for each customer.

Attrition Flag (Output): Indicates whether the customer's account has been closed (churned) or is still active. This is often the target variable for predictive modeling.

1. Customer Age: The age of the customer in years.
2. Gender: The gender of the customer (M/F).
3. Dependent Count: The number of dependents the customer has.
4. Education Level: The educational qualification of the customer (e.g., High School, Graduate, College, Uneducated, Post-Graduate).
5. Marital Status: The marital status of the customer (e.g., Married, Single, Divorced, Unknown).
6. Income Category: The annual income category of the customer (e.g., Less than $40K, $40K - $60K, $60K - $80K, $80K - $120K, $120K+, Unknown).
7. Card Category: The type of credit card held by the customer (e.g., Blue, Silver, Gold, Platinum).
8. Months on Book: The period, in months, that the customer has been with the bank.
9. Total Relationship Count: The total number of products the customer holds with the bank.
10. Months Inactive: The number of months the customer has been inactive in the last 12 months.
11. Contacts Count: The number of contacts the customer has had in the last 12 months.
12. Credit Limit: The credit limit on the customer's credit card.

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

13. Total Revolving Bal: The total revolving balance on the customer's credit card.

14. Avg Open To Buy: The open to buy credit line (average of the last 12 months).

15. Total Amount of Change (Q4 to Q1): The change in transaction amount from Q4 to Q1.

16. Total Transaction Amount: The total transaction amount in the last 12 months.

17. Total Transaction Count: The total transaction count in the last 12 months.

18. Total Count Change (Q4 to Q1): The change in transaction count from Q4 to Q1. 19. Average Utilization Ratio: The average card utilization ratio.

The Application is in such a way that after launching, it starts loading the model, preprocessor and metadata and then, user can select the input mode to choose between entering customer details or loading sample input values. It goes through some procedures and in the end, shows as the output whether the customer is likely to churn or will stay active in the future.

1.2 Non-functional Requirements

The Application must be user-friendly, intuitive to use, seamlessly integrating and functioning.

The Application code base should be maintainable, scalable in order to allow data and software engineers to update their software according to the upcoming needs or novel machine learning approaches, etc. Furthermore, considering the fact that the customers' data are of high privacy value and highly confidential, the Application has to be of high dependability and security in order to prevent any attacks.

2 Background

2.1 Technology Stack

For this project, we used Streamlit as a Python framework for rapid development of web applications particularly used for machine learning. Streamlit's simplicity and maintainability makes it an ideal option for development of these sorts of applications and makes the development procedure much easier for potential improvements in the future. Below is a brief description of the technology stack used in this project which are Streamlit and Universal Modeling Language (UML) diagrams:

- Streamlit: Used for rapid UI prototyping and real-time interaction.
- UML Diagrams: Created Use Case Flow Chart and Sequence Diagrams to define application behavior.

2.2 Unified Modeling Language (UML) Diagram

UML diagrams are used in the project to model workflow, architecture and interactions of the Application and provide a visual representation of them [3]. Two main sorts of UML diagrams have been used:

- Use Case Flow Chart Diagram: It provides a high-level view of functionality and potential usages of the procedures involved in the project.
- Sequence Diagram: It provides a high-level view of the system's behavior and the chain and sequence of the interactions between the user and the system at different layers.

3 Methodology and approach

3.1 UML Use Case Flow Chart Diagram

The use case flow chart diagram shows the flow of work of the Application from the backend to the front-end and how users fits in accordingly. Below is a visual representation of use case flow chart diagram:

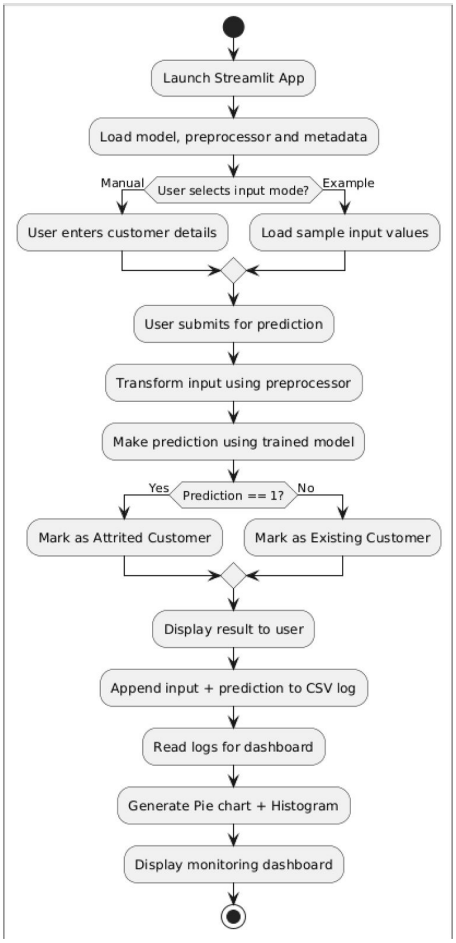

Figure 1: UML Use Case Flow Chart Diagram

3.2 UML Sequence Diagram

This diagram provides a clear representation of the chain and sequence of interactions between the user and the system. It includes how user interacts with the interface in order to use the Application and how the Application conveys user's commands and options to the backend and how the backend responds to the request and sends back the results to the front-end interface. Therefore altogether, it represent Application's flow of work.

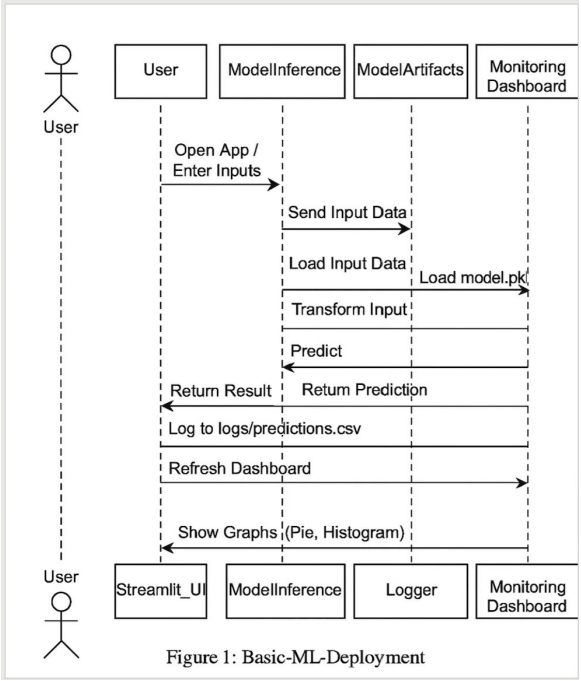

Figure 2: UML Sequence Diagram

3.3 Front-end Implementation

In this part, I sought to design a light, intuitive and user friendly front-end UI in order for users to utilize the Application without facing any difficulties. As mentioned before, I used Streamlit in order to design the GUI, its simplicity and capability for rapid prototyping and development, made it a great choice for us in this regard.

- For implementation of the front-end using UI Controls, input form with labeled fields for all required features were used to get the input data from the user. The UI was built in such a way that it could interact with the backend model efficiently in order to be adaptive in the sense that it has the possibility to incorporate the most relevant data features in the interface, that is to say that in the data engineering section, during the preprocessing and feature engineering, features that were not contributing to the final output were removed for example; therefore, the front-end UI has this possibility to adapt its input fields using some metadata.
- After the user inputs the relevant feature values, the front-end connect with the backend model and shows to the user whether the user is likely to churn in the future or going to continue using the services.
- As another part, the possibility of using advanced features for advanced users has been designed in order for them to interact with the backend machine learning model using more advanced relevant feature.
- As the last part to mention regarding the capabilities of the UI, it has the ability to maintain some logs of the analyzes it has performed and show to user plots including pie charts to represent the ratio of churning.

Below you can see different parts and aspects of our UI:

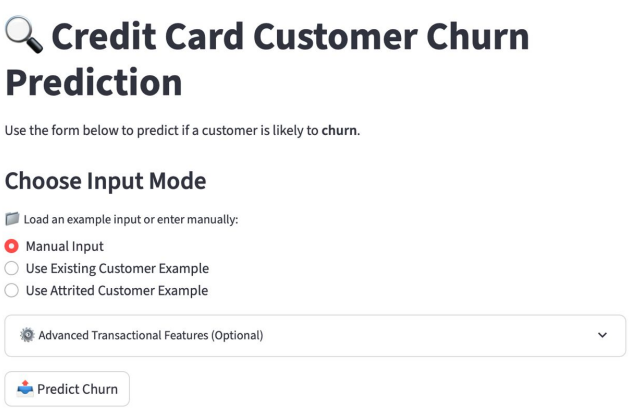

Figure 3: Different options for the user

Users choose among manual insertion, seeing active customer's example and seeing churned customer's example.

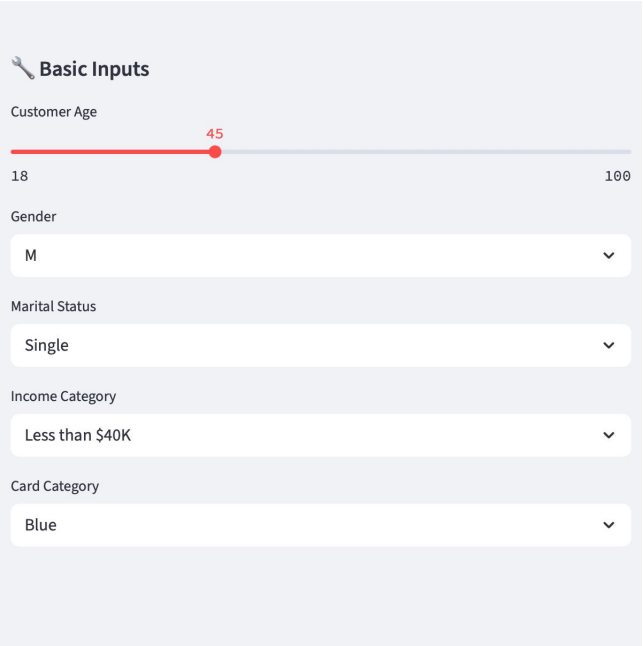

Figure 4: Basic features data insertion

Beginner users can use basic features in order to make predictions.

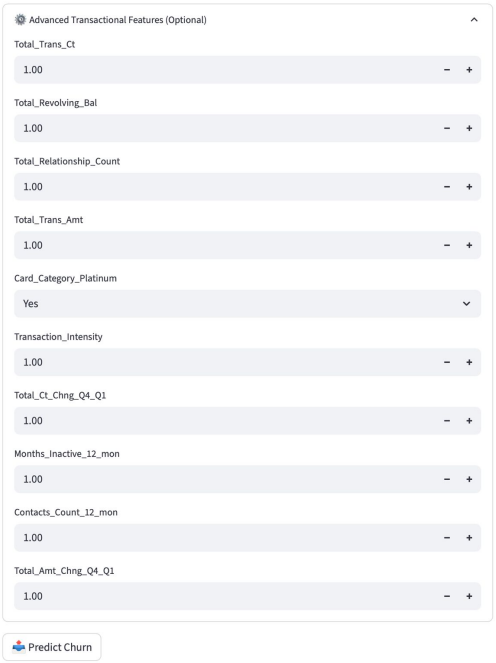

Figure 5: Advanced features data insertion

Advanced users can use advanced features in order to make predictions.

4 Discussion and conclusion

In this project, I sought to design a GUI to be used as an interface to deploy a machine learning model in order to predict if the customers are likely to churn. the churning likelihood of customers based on their data. The Application is supposed to be used mainly by bank managers. The interface had to match the front-end user inputs with the backend model expectations. I worked closely with the Data Engineer to ensure correct orders of the features and with Software Engineer in order to design an interface which is piped properly with all layers of the Application and to test the API endpoints and error handling. The Application should be utilized by users and can be improved based on real user suggestions. Furthermore, my own suggestions for further development in the future are adding a use case in order to save the newly inserted users' data to the database to obtain a more enriched and updated dataset to fine-tune the backend machine learning model. In the end, the application can be improved to be used for a better experience on mobile devices.

5 Team Structure

We carried out the project in a team consisting of three members with clearly defined roles for each one of us. Vishal Nigam served the role of the Data Engineer and performed data preprocessing, model development, evaluation and tuning to build the backend machine learning model, I served the role of Software Developer and worked on the frontend user interface and all the other details that I mentioned in this report and Muhammad Mubashar Shahzad served the role of Software Engineer focusing on backend integration and API management. Our team work enables us to perform different parts of the project effectively and build an application that works smoothly on different layers.

6 Citations and Bibliographies

References

[1] Kaggle. 2020. Credit Card Customer Churn Prediction Dataset. https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers/ Accessed: 2025-06-27.
[2] Streamlit. 2024. Streamlit Documentation. https://docs.streamlit.io/ Accessed: 2025-06-27.
[3] Visual Paradigm. 2024. UML Use Case Guidelines. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/use-case-diagram/ Accessed: 2025-06-27.