CUSTOMER CHURN PREDICTION

PROFESSIONAL READINESS LAB REPORT

Submitted by

AMEERA FARHA	**111721202001**
ILAKKIYA N	**111721202014**
JAYAKARTHIKA K	**111721202018**
JISHA ARAVIND	**111721202019**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEMS

R.M.K. ENGINEERING COLLEGE

(An Autonomous Institution)

R.S.M. Nagar, Kavaraipettai-601 206

NOVEMBER 2024

R.M.K. ENGINEERING COLLEGE

(An Autonomous Institution)

R.S.M. Nagar, Kavaraipettai-601 206

BONAFIDE CERTIFICATE

Certified that this Professional Readiness Lab report **Customer Churn Prediction** is the bonafide work of **Ameera Farha (111721202001), Ilakkiya N (111721202014), Jayakarthika K (111721202018), Jisha Aravind (111721202019)** who carried out the 20IT713- Professional Readiness for Innovation Employability and Entrepreneurship under my supervision**.**

SIGNATURE

Dr. K. Chidambarathanu, M.E., Ph.D.,
Professor and Head of the Department

Department of Computer Science and Business Systems
R.M.K. Engineering College
R.S.M. Nagar, Kavaraipettai,
Tiruvallur District– 601206.

SIGNATURE

Dr. K. Chidambarathanu, M.E., Ph.D.,
Professor and Head of the Department

Department of Computer Science and Business Systems
R.M.K. Engineering College
R.S.M. Nagar, Kavaraipettai,
Tiruvallur District–601206.

Submitted for the Project Viva–Voce held on ……..at **R.M.K. Engineering College**, Kavaraipettai, Tiruvallur District– 601206.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We earnestly portray our sincere gratitude and regard to our beloved **Chairman Shri. R. S. Munirathinam, our Vice Chairman, Shri. R. M. Kishore** and **our Director, Shri. R. Jyothi Naidu,** for the interest and affection shown towards us throughout the course.

We convey our sincere thanks to our **Principal**, **Dr. K. A. Mohamed Junaid,** for being the source of inspiration in this college.

We reveal our sincere thanks to our **Professor and Head of the Department, Computer Science and Business Systems, Dr.K.Chidambarathanu,** for his commendable support and encouragement for the completion of our project.

We would like to express our sincere gratitude for our Project Coordinators **Dr. K. Chidambarathanu, Professor and Head of the Department and Ms.C.Mary Shiba, Assistant Professor** for their valuable suggestions towards the successful completion for this project in a global manner.

We take this opportunity to extend our thanks to my Parents, all faculty members of the Department of Computer Science and Business Systems and friends for all that they meant to us during the crucial times of the completion of our project.

ABSTRACT

Customer churn, or the process by which customers end their association with a company, presents substantial difficulties to corporate development and profitability. Churn can manifest in a variety of ways: contractual churn occurs when customers under service agreements (e.g., SaaS or Cable TV) cancel; non-contractual churn occurs when customers without binding contracts disengage (e.g., loyalty programs); voluntary churn refers to customers who actively choose to leave a service (e.g., mobile plans); and involuntary churn results from external issues such as payment failures. Voluntary churn is sometimes caused by poor service quality, decreased consumption, or better price alternatives from rivals. Accurate churn prediction is critical to enhancing client retention efforts. This project uses the Random Forest Classifier, a machine learning method, to forecast client attrition. Random Forest is well-suited for this investigation because of its capacity to handle enormous datasets and deliver accurate classification findings. Data will be handled using Python libraries like pandas and numpy, visualizations will be done with seaborn and matplotlib, and the Random Forest method will be implemented using scikit-learn. The objective is to identify important causes causing churn and evaluate model performance using measures such as accuracy and precision. This model uses information from a telecoms dataset to assist organizations proactively minimize churn and enhance customer retention efforts.

Keyword: Customer Churn, Random Forest Classifier, Python, Scikit-learn.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.NO.	ABBREVIATION	EXPANSION
1	ANN	ARTIFICIAL NEURAL NETWORK
2	SOM	SELF ORGANIZING MAP
3	RIPPER	REPEATED INCREMENTAL PRUNING TO PRODUCE ERROR REDUCTION
4	SVM	SUPPORT MACHINE VECTOR

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Customer churn, where customers discontinue their relationship with a company, poses significant challenges for business growth and profitability. It can occur in various forms, including voluntary churn due to service quality or pricing issues and involuntary churn due to payment problems. Accurate prediction of customer churn is essential to improve retention efforts and reduce business losses. This project aims to develop a churn prediction model using the Random Forest Classifier, leveraging its robustness in handling large datasets and delivering accurate classifications. By analyzing data from the telecom industry, the model seeks to identify key churn drivers, providing businesses with actionable insights to proactively address churn and enhance customer retention.

1.2 Literature Survey

1. **Tsai, C.-F., & Lu, Y.-H. (2009). "Customer churn prediction by hybrid neural networks."**

   This paper examines customer churn prediction using hybrid neural networks, specifically ANN + ANN and SOM + ANN models. Each model uses one network for data reduction and another for prediction, resulting in higher accuracy and fewer errors than single-model approaches. Among them, ANN + ANN achieved the best results, highlighting the effectiveness of hybrid neural models in churn prediction.

2. **Verbeke, W., Martens, D., Mues, C., & Baesens, B. (2011). "Building comprehensible customer churn prediction models with advanced rule induction techniques."**

   This paper emphasizes the importance of creating churn prediction models that are both accurate and comprehensible. It introduces two novel techniques—AntMiner+ and ALBA—which outperform traditional methods like C4.5 and RIPPER by offering clear, intuitive rule-sets. AntMiner+ leverages Ant Colony Optimization to incorporate domain knowledge, while ALBA merges the high accuracy of support vector machines with comprehensibility, resulting in effective, justifiable models for targeted retention strategies.

3. **Mitkees, I. M. M., Badr, S. M., & ElSeddawy, A. I. B. (2017). "Customer churn prediction model using data mining techniques."**

   This paper addresses the significant issue of customer churn in the telecommunications sector, where customers may switch to competitors. The aim is to develop a predictive model for identifying customers likely to churn by analyzing specific behaviors and attributes. Utilizing data mining techniques such as clustering, classification, and association rules, the research underscores the critical need for accurate and precise predictions to enhance customer retention efforts. Without awareness of impending churn, companies are unable to take proactive measures to retain their customers effectively.

4. **Zhao, Y., Li, B., Li, X., Liu, W., & Ren, S. (2015). "Customer Churn Prediction Using Improved One-Class Support Vector Machine."**

   This paper addresses the critical issue of customer churn prediction in the wireless industry, where existing methods often lack sufficient accuracy. The authors present an improved one-class Support Vector Machine (SVM) designed to effectively manage the challenges posed by imbalanced churn datasets. Their approach demonstrates superior performance compared to traditional methods such as Artificial Neural Networks (ANN), Decision Trees, and Naïve Bayes, highlighting its potential to enhance customer retention and minimize acquisition costs through more accurate churn predictions.

5. **Seymen, O. F., Dogan, O., & AbdulKadir. (2020). "Customer Churn Prediction using Deep Learning."**

   This paper highlights the effectiveness of deep learning in customer churn prediction for the retail industry, demonstrating superior performance compared to logistic regression and artificial neural networks. The deep learning model excels in classification metrics like precision, recall, and AUC, showcasing its enhanced ability to predict churn accurately.

1.3 System Requirements

1.3.1 Hardware Requirements

The implementation of the "Customer Churn Prediction" system requires specific hardware components to facilitate efficient processing and scalability. The essential hardware requirements are outlined below:

- Server: A robust server is crucial for managing customer data and processing churn predictions. The minimum requirements include a server with:
    - Processor: Quad-core processor (e.g., Intel Xeon or AMD Ryzen) to efficiently handle multiple requests and data processing tasks.
    - RAM: At least 16 GB of RAM to support the processing of large datasets and ensure smooth execution of machine learning algorithms.
    - Storage: 500 GB of storage or more to securely store customer data, including historical usage patterns and model outputs.
- User Devices: Users need devices capable of accessing the churn prediction system, such as:
    - Desktop/Laptop: Equipped with a modern web browser (e.g., Chrome, Firefox) for accessing the prediction interface and visualizations.
    - Smartphones/Tablets: Optional, for accessing reports and insights on-the-go.
- Network Infrastructure: Reliable internet connectivity with minimal latency is essential to ensure that data can be quickly transmitted between the server and user devices, facilitating timely predictions and decision-making.

- **1.3.2 Software Requirements**

The software stack used to develop and deploy the "Customer Churn Detection App" includes various frameworks, libraries, and tools. The core requirements are as follows:

- Python Ecosystem:
    - Streamlit: A Python library for creating interactive web applications, used for the app's user interface and interaction with users.
    - NumPy: A library for numerical operations, employed for handling arrays and matrices during data processing.
    - Pandas: A data manipulation library that simplifies data analysis and handling structured data for preprocessing tasks.
    - Scikit-learn: A machine learning library that provides tools for training, validation, and evaluation of predictive models.
    - TensorFlow: An open-source library for machine learning, used for building and training the Artificial Neural Network (ANN) model.
    - Keras: A high-level neural networks API running on top of TensorFlow, utilized for designing and training the ANN model.
- Development Tools:
    - Integrated Development Environment (IDE): Tools such as Visual Studio Code or PyCharm for efficient code development.
    - Version Control: Git and GitHub for source code management and collaboration among developers.
- Operating System:
    - Server OS: Linux-based operating systems like Ubuntu for hosting the server.

- User Devices: Cross-platform support for Windows, macOS, iOS, and Android.

1.3.3 Feasibility Study

Before developing the "Customer Churn Prediction," a feasibility study is conducted to evaluate its viability across three key areas:

- **Technical Feasibility:**

  The proposed system is technically feasible, leveraging modern frameworks and libraries available in the Python ecosystem. Streamlit provides an easy-to-use interface for building interactive web applications, while libraries like TensorFlow and Keras enable the development and training of sophisticated machine learning models. The integration of Scikit-learn allows for robust preprocessing and prediction capabilities. Additionally, the use of NumPy and Pandas facilitates efficient data manipulation and analysis, ensuring that the application can handle diverse datasets effectively.

- **Economic Feasibility:**

  The economic feasibility is favorable, as most of the required frameworks and libraries (e.g., Streamlit, TensorFlow) are open-source and freely available. Initial costs are primarily associated with server hosting and developer resources. As the application may require cloud services for scaling, there could be recurring costs based on usage. Long-term, the system could generate revenue through premium features or consulting services, supporting cost recovery and potential profitability.

- **Operational Feasibility:**

  The application is designed to be user-friendly, providing a straightforward interface for users to input data and receive predictions. By simplifying the customer churn prediction process, the app can help businesses proactively manage customer relationships and improve retention strategies. The deployment of this application can be integrated seamlessly into existing customer relationship management systems, making it suitable for various industries, particularly telecommunications and service-oriented sectors.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

The traditional methods of customer retention and churn analysis predominantly rely on historical customer data without leveraging advanced predictive analytics. This results in several challenges:

- Reactive Approach: Businesses often respond to churn after it has occurred, rather than implementing strategies to prevent it proactively.
- Limited Data Utilization: Existing systems frequently depend on basic metrics, such as customer complaints and account cancellations, ignoring a comprehensive analysis of customer interactions and engagement metrics.
- Manual Analysis: The reliance on manual data analysis can lead to inaccuracies and inefficiencies, making it difficult to derive actionable insights in a timely manner.
- High Customer Acquisition Costs: Failing to anticipate churn can lead to significant costs associated with acquiring new customers to replace those lost, negatively impacting profitability.

2.1.1 Disadvantages of Existing System

The existing customer churn management approaches present several drawbacks:

- Lack of Proactive Measures: Companies are often caught unaware, missing opportunities to engage customers at risk of churning before they leave.

- Inconsistent Data Handling: The fragmented use of data across various departments can lead to a lack of a unified understanding of customer behavior, complicating retention efforts.
- Over-reliance on Historical Data: Traditional systems frequently focus on historical data without integrating advanced predictive analytics, limiting their ability to anticipate future customer behaviors effectively.
- Inefficient Resource Allocation: Without accurate predictions, businesses may misallocate resources, leading to wasted efforts on customers who are less likely to leave, while ignoring those at higher risk.

2.2 Proposed System

The **Customer Churn Prediction** system proposes a comprehensive, data-driven approach utilizing machine learning algorithms to proactively predict customer churn. This system will analyze a wide array of customer-related features to identify patterns indicating the likelihood of a customer discontinuing their service.

The implementation will employ algorithms such as the **Random Forest Classifier** and **Artificial Neural Networks (ANN)**, built using **Streamlit** for a user-friendly interface. By providing accurate churn predictions, businesses can tailor their retention strategies effectively and allocate resources more efficiently.

2.2.1 Advantages of Proposed System

The proposed system for predicting customer churn offers numerous advantages:

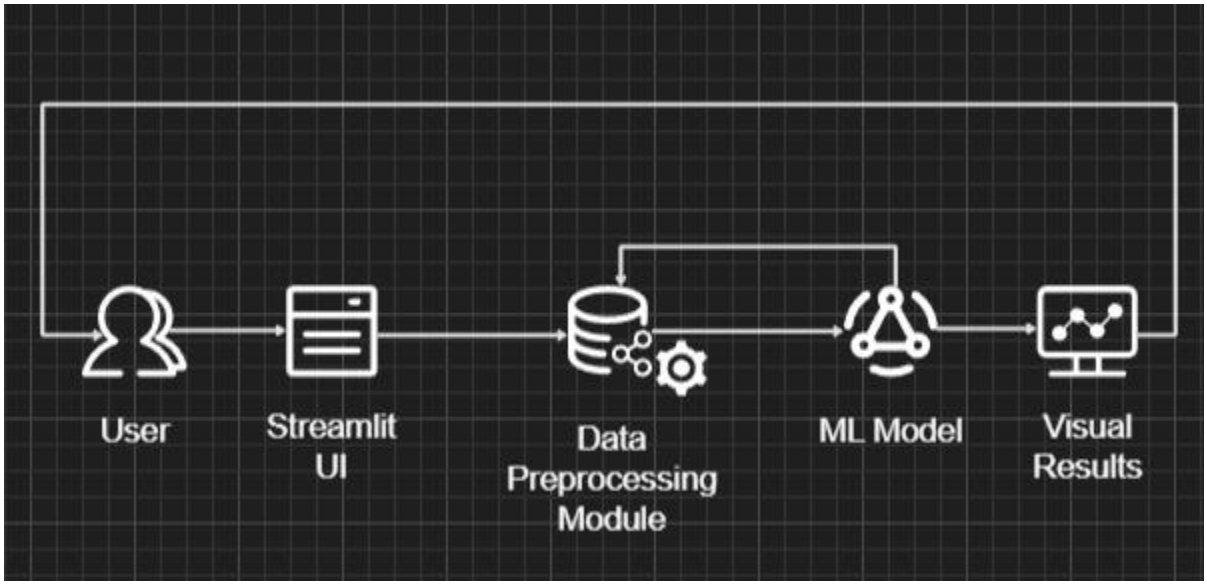
- Proactive Churn Prevention: By identifying potential churn before it occurs, businesses can engage at-risk customers with tailored strategies to retain them.
- Enhanced Data Utilization: The system leverages a diverse set of customer data, including demographics, service usage patterns, and payment history, improving predictive accuracy.
- Automated Analysis: Automation of the machine learning processes reduces the time spent on manual analysis, allowing quicker insights and enabling timely decision-making.
- Cost Efficiency: Focusing retention efforts on high-risk customers optimizes budgets and reduces overall costs associated with customer acquisition and retention.
- User-Friendly Interface: Streamlit provides an intuitive interface for users to input data and interpret results, ensuring ease of use and accessibility for stakeholders involved in customer retention strategies.

CHAPTER 3

SYSTEM DESIGN

3.1 Architecture Diagram

The system architecture for the Customer Churn Prediction system comprises several integrated components that work together to deliver accurate churn predictions and insights. The architecture is built on a combination of machine learning models and a user-friendly interface, utilizing the Streamlit framework for front-end interactions. The architecture includes a data processing module for cleaning and preparing customer data, a machine learning module that applies algorithms like Random Forest and ANN for churn prediction, and a backend that handles data storage and retrieval. The model outputs are presented through an intuitive web interface, allowing users to input customer data and receive predictions seamlessly.



- **Front-End (Streamlit):** The user interface is built using Streamlit, allowing users to interact with the application for customer churn predictions. Users can input customer data individually or upload a dataset for batch predictions. The front-end collects the input data, sends it to the backend for processing, and displays the prediction results, either as a single response or as a table of results for multiple customers.
- **Back-End (Python & Scikit-Learn):**

  The backend logic is implemented in Python, using a pre-trained machine learning model stored in a .sav file and loaded with joblib. The backend handles data preprocessing, including encoding, scaling, and transforming input data to match the model's requirements. Once the data is processed, it is passed to the machine learning model for prediction. The backend then formats the prediction results for display on the front end.
- **Data Preprocessing Module:**

  The preprocessing logic is encapsulated in a custom Python module (preprocessing). This module ensures that all input data is correctly formatted and consistent with the model's training data format. It includes steps like handling categorical data, scaling numerical values, and filling in missing data if necessary. By preparing the data in this way, the module ensures that the machine learning model receives high-quality input for accurate predictions.
- **Machine Learning Model (Scikit-Learn):**

  The machine learning model, developed and trained using Scikit-Learn, is stored in a serialized .sav file. The model predicts customer churn based on various features such as demographic data, payment information, and subscribed services. The backend loads the model during initialization and uses it to predict churn likelihood based on either single or batch inputs.

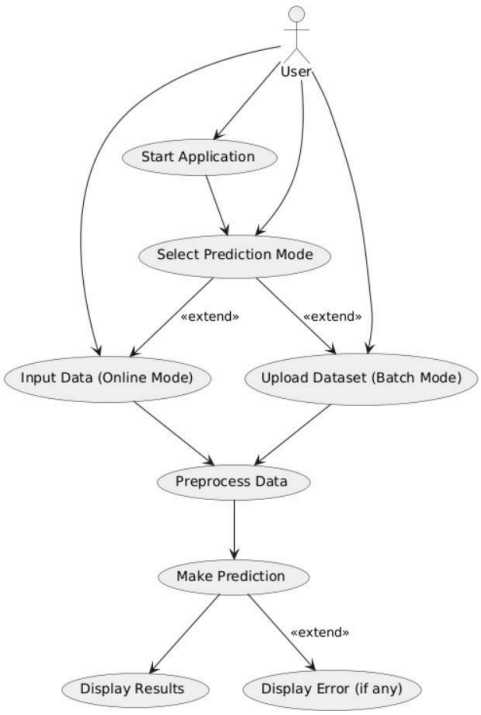
- **Prediction Flow:**

  The prediction process begins when the user submits data. For single predictions, the user enters data manually, while batch predictions are handled via CSV file upload. The front-end collects this data and sends it to the backend. The backend preprocesses the data, applies the model for prediction, and sends the result back to the front end. The front end then displays a simple message for single predictions or a table of results for batch predictions, indicating whether each customer is likely to churn.

3.2 UML Diagrams

Unified Modeling Language (UML) diagrams are used to visually represent the system's design and behavior. The following diagrams illustrate different aspects of the system:

3.2.1 Use Case Diagram



User

- The **User** represents the person who interacts with the application to predict customer churn. The user initiates the application and selects a mode to make predictions.

Start Application

- The user begins by launching the application. This action initiates the application interface, typically displayed through the Streamlit UI.
- Once started, the user can proceed to select the desired prediction mode.

Select Prediction Mode

- After starting the application, the user chooses a prediction mode.
- There are two modes available: **Online Mode** and **Batch Mode**. The choice between these modes depends on whether the user wants to input data for a single customer or upload a dataset for multiple customers.
- The «extend» relationship indicates that the selection of either mode extends the functionality of the application, as each mode has a different data entry requirement.

Input Data (Online Mode)

- This use case is invoked if the user selects **Online Mode**.
- In Online Mode, the user manually inputs customer data, such as demographic information, subscription details, and payment methods.
- This data entry form is presented as part of the Streamlit UI, allowing the user to enter one customer's data at a time for prediction.

Upload Dataset (Batch Mode)

- This use case is invoked if the user selects **Batch Mode**.
- In Batch Mode, the user uploads a dataset (CSV file) containing data for multiple customers.
- The system reads this file and prepares it for processing, enabling batch predictions for multiple records in a single step.

Preprocess Data

- Regardless of the mode chosen, all input data is passed through a **Data Preprocessing** step.
- This step involves data cleaning, encoding categorical variables, scaling numerical features, and any other transformation required to make the data compatible with the machine learning model.
- Preprocessing ensures that the data format and values are consistent with what the model expects for accurate prediction.

Make Prediction

- Once the data is preprocessed, the system uses the machine learning model to make predictions on the customer churn likelihood.
- In Online Mode, this involves a single prediction, while in Batch Mode, the system processes and predicts churn for all customers in the uploaded dataset.

Display Results

- After predictions are made, the application displays the results back to the user.
- For Online Mode, the result is typically a message indicating whether the customer is likely to churn or not.

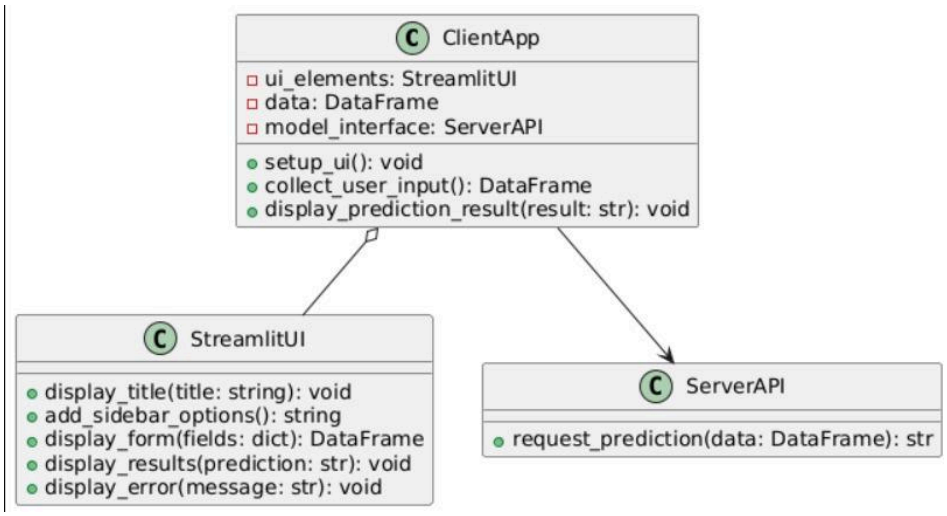
- For Batch Mode, results are presented as a table or list, showing predictions for each customer in the dataset.

Display Error (if any)

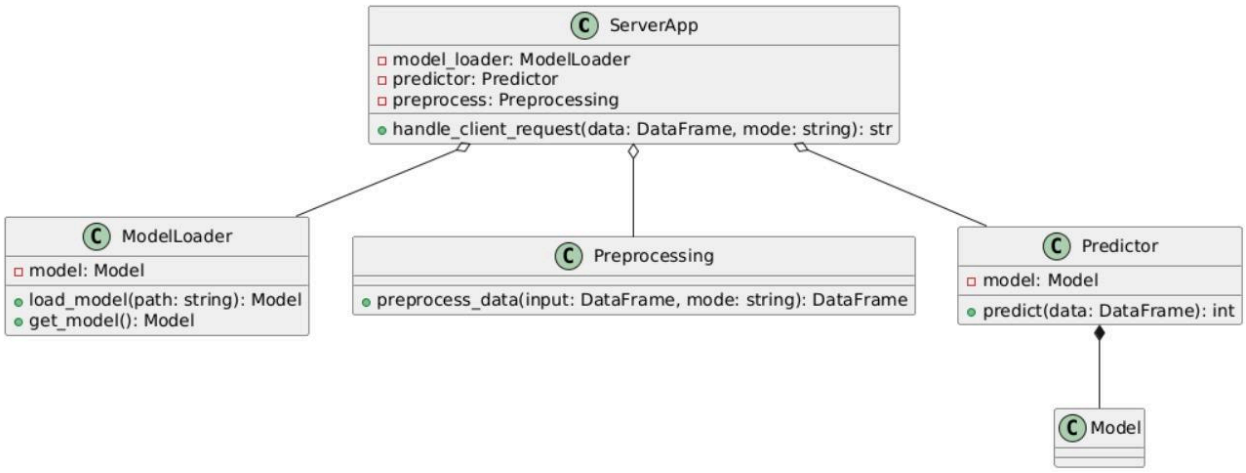
- This use case handles any errors that might occur during data input, preprocessing, or prediction.
- Errors can include issues with incorrect data formats, missing fields, file upload issues in Batch Mode, or model prediction errors.
- If an error occurs, the system provides feedback to the user, helping them understand and potentially correct the problem.

3.2.2 Class Diagram

This section presents the class diagrams for a Customer Churn Prediction System, highlighting both the Client-Side and Server-Side components. The system enables users to predict the likelihood of customer churn based on demographic and service-related data.



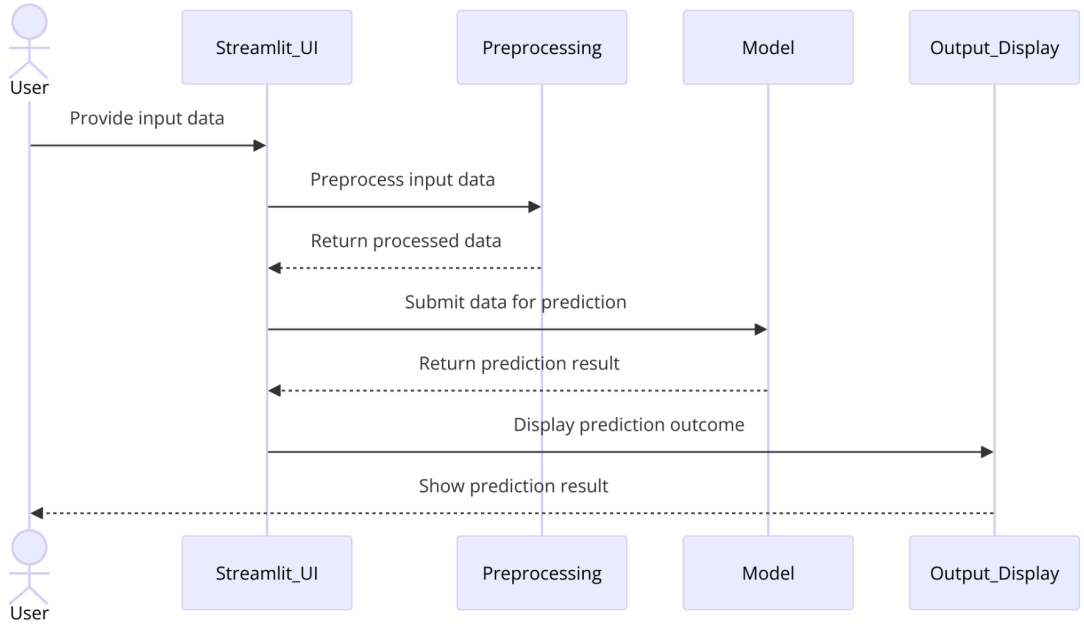
The **Client-Side** of the system provides the user interface and handles user interactions. Built with Streamlit, it is responsible for displaying the application screen, gathering user inputs, and handling data upload options (for single or batch predictions). Users can choose between two prediction modes: **Online Mode** (inputting individual data points) or **Batch Mode** (uploading a dataset file for bulk prediction). The client-side component also preprocesses the data to ensure it is in the correct format before sending it to the server. All data transmissions, including user inputs and predictions, are sent to the server over encrypted communication channels (e.g., HTTPS) to ensure secure communication.



The **Server-Side** of the system handles data processing and prediction. It securely manages the machine learning model, performs predictions on incoming data, and communicates with the client-side interface. When a prediction request is received, the server preprocesses the data if necessary, runs the model to make predictions, and returns the results to the client. The server also implements logging and error handling mechanisms to track and respond to potential issues.

The combined client-side and server-side architecture provides a seamless and secure experience, enabling accurate and timely churn predictions for businesses.

3.2.3 Sequence Diagram



Sequence 1: User Provides Input Data

- **Action**: The user interacts with the Streamlit application to enter their data.
- **Details**: The application presents various input fields for the user to fill out. This includes demographic information (e.g., whether they are a senior citizen or have dependents), payment details (e.g., the number of months they have stayed with the company, the type of contract, monthly charges, and total charges), and services subscribed to (e.g., internet service type, online security, and streaming services).
- **Expected Outcome**: The user completes the form and submits the information, preparing to move to the next step.

Sequence 2: Streamlit App Preprocesses Input Data

- **Action**: The application processes the user's input data.
- **Details**: Once the user submits their input, the Streamlit app calls the preprocess function. This function is responsible for transforming the raw input data into a format that the machine learning model can understand. This may include encoding categorical variables (e.g., converting 'Yes' and 'No' to numerical values), normalizing numerical data, and handling any missing values.
- **Expected Outcome**: The preprocessed data is structured and cleaned, making it ready for the prediction model.

Sequence 3: Return Processed Data

- **Action**: The preprocessing function returns the transformed data to the application.
- **Details**: After the preprocessing is complete, the cleaned data is sent back to the Streamlit application. This step ensures that the application has access to the data in the correct format for the next phase.
- **Expected Outcome**: The Streamlit app receives the preprocessed data, confirming that it is ready for prediction.

Sequence 4: User Submits Data for Prediction

- **Action**: The user initiates the prediction process.
- **Details**: The user clicks the 'Predict' button within the application interface. This action serves as a command to the application to start the prediction using the preprocessed data.

- **Expected Outcome**: The application acknowledges the button click and prepares to send the preprocessed data to the machine learning model for analysis.

Sequence 5: Model Predicts Churn

- **Action**: The application sends the preprocessed data to the machine learning model.
- **Details**: The Streamlit app takes the processed input data and feeds it into the machine learning model. The model uses its trained algorithms to evaluate the input and determine the likelihood of customer churn based on historical data patterns.
- **Expected Outcome**: The model processes the data and generates a prediction result, typically in the form of a numerical output (0 or 1).

Sequence 6: Return Prediction Result

- **Action**: The model sends back the prediction result to the application.
- **Details**: After the prediction is made, the model returns the result to the Streamlit application. In this context, a prediction of '1' indicates that the customer is likely to terminate the service, while a prediction of '0' indicates that the customer is likely to stay.
- **Expected Outcome**: The application receives the prediction result, which is crucial for the next steps in the user interface.

Sequence 7: Display Prediction Outcome

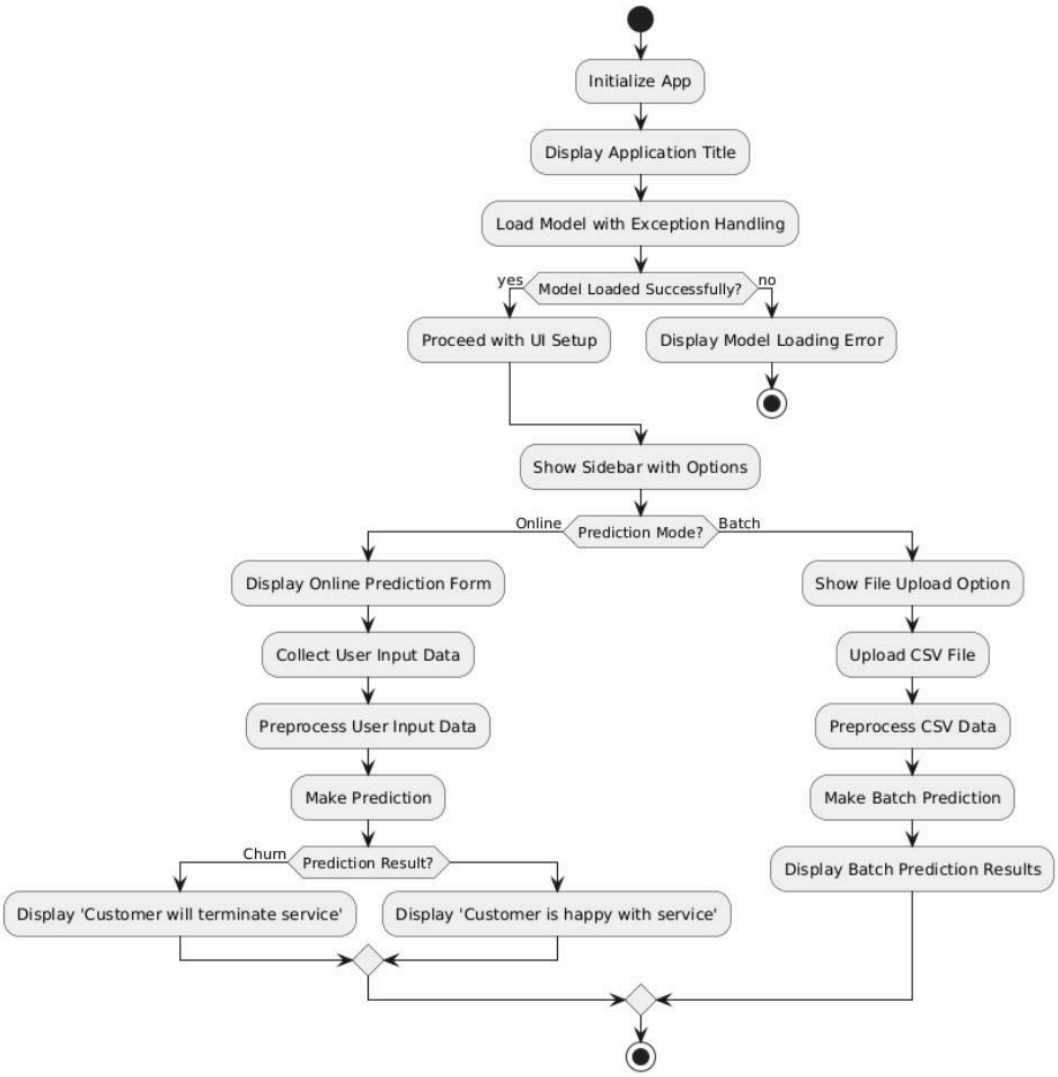
- **Action**: The application prepares to display the prediction outcome to the user.

- **Details**: Based on the result obtained from the model, the Streamlit app prepares a message to convey the prediction to the user. This involves selecting the appropriate message based on whether the prediction indicates churn or retention.
- **Expected Outcome**: The application is ready to present the outcome in a user-friendly manner.

Sequence 8: Show Prediction Result

- **Action**: The application displays the final prediction result to the user.
- **Details**: The Streamlit app presents a clear message to the user, indicating the likelihood of customer churn. If the prediction result is '1', it displays, "Yes, the customer will terminate the service." If the prediction result is '0', it displays, "No, the customer is happy with Telco Services."
- **Expected Outcome**: The user receives a straightforward interpretation of the prediction, which helps them understand the implications of the model's analysis.

3.2.4 Activity Diagram



- **Initialize App**: This step initializes the Streamlit application and sets the title to "Customer Churn Prediction App".
- **Load Model with Exception Handling**: This step attempts to load the machine learning model from disk using joblib. If the model is loaded successfully, a message is printed stating "Model loaded successfully!". If there is an error loading the model, an error message is printed.

- **Display Application Title**: This step displays the title of the application, "Customer Churn Prediction App".
- **Show Sidebar with Options**: This step creates a sidebar in the Streamlit application that allows the user to choose between "Online" and "Batch" prediction modes. It also displays informational text about the app and an image.
- **Online Prediction Mode**: This branch of the activity diagram is executed if the user selects "Online" in the sidebar.
    - **Display Online Prediction Form**: This step displays a form where the user can input their demographic data, payment data, and details about the services they have signed up for.
    - **Collect User Input Data**: This step collects the user's input data from the form and stores it in a dictionary.
    - **Preprocess User Input Data**: This step preprocesses the user's input data using the preprocess function from the preprocessing.py script. The argument 'Online' is passed to the function to indicate that it is performing online prediction.
    - **Make Prediction**: This step makes a prediction on the preprocessed data using the loaded model.
    - **Display Prediction Result**: This step displays a message to the user based on the prediction result. If the prediction is 1, a warning message is displayed stating that the customer will terminate the service. If the prediction is 0, a success message is displayed stating that the customer is happy with the service.

- **Batch Prediction Mode**: This branch of the activity diagram is executed if the user selects "Batch" in the sidebar.
    - **Display Batch Prediction Form**: This step displays a file upload option where the user can upload a CSV file containing customer data.
    - **Upload CSV File**: This step reads the uploaded CSV file into a Pandas DataFrame.
    - **Preprocess CSV Data**: This step preprocesses the data in the DataFrame using the preprocess function from the preprocessing.py script. The argument 'Batch' is passed to the function to indicate that it is performing batch prediction.
    - **Make Batch Prediction**: This step makes predictions on all the data points in the preprocessed DataFrame using the loaded model.
    - **Display Batch Prediction Results**: This step displays the prediction results for all the data points in the DataFrame. The results are converted from numeric values (1 or 0) to human-readable text ("Yes, the customer will terminate the service." or "No, the customer is happy with Telco Services.").

CHAPTER 4

SYSTEM ANALYSIS

4.1 Modules

This subsection lists the main modules that comprise your system. Each module can be categorized based on its functionality and interactions with other components.

1. **Model Loading Module**
2. **Preprocessing Module**
3. **Prediction Module**
4. **User Interface Module**
5. **Visualization Module**
6. **Error Handling Module**

4.2 Module Description

This section provides a detailed description of each module, outlining its purpose, functionalities, and interactions with other modules.

4.2.1 Model Loading Module

- **Purpose**: Loads the pre-trained machine learning model.
- **Functionality**: Uses joblib to load the model from disk with exception handling.
- **Interactions**: Interacts with the file system to load the model file.

4.2.2 Preprocessing Module

- **Purpose**: Preprocesses input data for prediction.
- **Functionality**: Applies necessary transformations to the input data to make it suitable for the model.
- **Interactions**: Interacts with the preprocess function from the preprocessing.py script.

4.2.3 Prediction Module

- **Purpose**: Makes predictions using the pre-trained model.
- **Functionality**: Uses the preprocessed data to generate predictions.
- **Interactions**: Interacts with the model to get predictions and returns the results to the UI.

4.2.4 User Interface Module

- **Purpose**: Manages user interactions and input collection.
- **Functionality**: Uses Streamlit to create a web interface for user input and displays results.
- **Interactions**: Collects input data from the user and displays predictions.

4.2.5 Visualization Module

- **Purpose**: Displays the results and insights to the user.
- **Functionality**: Uses Streamlit to render dataframes and prediction results.
- **Interactions**: Interacts with the User Interface Module to display results.

4.2.6 Error Handling Module

- **Purpose**: Centralizes error reporting and user feedback.
- **Functionality**: Handles exceptions and displays error messages.
- **Interactions**: Integrates with the User Interface Module to provide user-friendly feedback.

4.3 Algorithms

This subsection describes the algorithms implemented in your system for key functionalities.

4.3.1 Data Preprocessing Algorithm

- **Description**: Prepares input data for prediction by applying necessary transformations.
- **Process**:
    1. Users input data through the UI.
    2. The data is passed to the preprocess function.
    3. The function applies transformations and returns the preprocessed data.

4.3.2 Prediction Algorithm

- **Description**: Uses the pre-trained model to make predictions based on preprocessed data.
- **Process**:
    1. Preprocessed data is fed into the model.
    2. The model generates predictions.
    3. Predictions are returned to the UI for display.

4.4 Testing

Testing is a critical phase in the system implementation process, ensuring that the "Customer Churn Prediction" system operates as intended. Various types of testing are conducted to identify and resolve issues before the system goes live.

4.4.1 Types of Testing

1. Unit Testing:
    - **Purpose**: Validate individual components of the system (modules, functions) for correctness.
    - **Approach**: Each module is tested in isolation using mock data.
    - **Tools Used**: unittest or pytest for Python functions.
2. Integration Testing:
    - **Purpose**: Ensure different components of the system work together seamlessly.
    - **Approach**: Test interactions between the UI, preprocessing, and prediction modules.
    - **Tools Used**: pytest for integration tests.
3. Functional Testing:
    - **Purpose**: Validate the system's functionality against the requirements.
    - **Approach**: Test all user interactions, such as data input and prediction.
    - **Tools Used**: Selenium for automated UI testing.
4. Performance Testing:
    - **Purpose**: Assess the system's performance under load.
    - **Approach**: Simulate multiple users making predictions simultaneously.

    - **Tools Used**: Apache JMeter for load testing.
5. Security Testing:
    - **Purpose**: Identify vulnerabilities in the system.
    - **Approach**: Conduct penetration testing and vulnerability assessments.
    - **Tools Used**: OWASP ZAP for security scanning.
6. User Acceptance Testing (UAT):
    - **Purpose**: Validate the system from the end-user's perspective.
    - **Approach**: Gather feedback from a group of end-users.

4.5 Test Cases

Below are ten detailed test cases that would be part of the testing process:

User Input and Prediction:

- **Test Case ID**: TC-001
    - **Description**: Verify users can input data and get a prediction.
    - **Steps**:
        1. Navigate to the application.
        2. Enter valid input data.
        3. Click the "Predict" button.
    - **Expected Result**: Prediction is displayed successfully.

Model Loading:

- **Test Case ID**: TC-002
    - **Description**: Verify the model loads correctly.
    - **Steps**:
        1. Start the application.
    - **Expected Result**: Model is loaded without errors.

Data Preprocessing:

- **Test Case ID**: TC-003
    - **Description**: Verify data preprocessing works correctly.
    - **Steps**:
        1. Input valid data.
        2. Check the preprocessed data.
    - **Expected Result**: Data is preprocessed correctly.

Error Handling:

- **Test Case ID**: TC-004
    - **Description**: Verify error handling for invalid input.
    - **Steps**:
        1. Input invalid data.
        2. Click the "Predict" button.
    - **Expected Result**: Appropriate error message is displayed.

Batch Prediction:

- **Test Case ID**: TC-005
    - **Description**: Verify batch prediction functionality.
    - **Steps**:
        1. Upload a valid CSV file.
        2. Click the "Predict" button.
    - **Expected Result**: Batch predictions are displayed successfully.

UI Elements:

- **Test Case ID**: TC-006

    - **Description**: Verify all UI elements are displayed correctly.
    - **Steps**:
        1. Navigate to the application.
    - **Expected Result**: All UI elements are visible and functional.

Session Timeout:

- **Test Case ID**: TC-007
    - **Description**: Verify session times out after inactivity.
    - **Steps**:
        1. Log in and leave the session inactive.
    - **Expected Result**: User is logged out after the timeout period.

Invalid Model Path:

- **Test Case ID**: TC-008
    - **Description**: Verify error handling for invalid model path.
    - **Steps**:
        1. Change the model path to an invalid location.
        2. Start the application.
    - **Expected Result**: Error message is displayed.

Multiple Failed Predictions:

- **Test Case ID**: TC-009
    - **Description**: Verify system handles multiple failed predictions.
    - **Steps**:
        1. Input invalid data multiple times.
    - **Expected Result**: Appropriate error messages are displayed.

Performance Under Load:

- **Test Case ID**: TC-010
    - **Description**: Verify system performance under load.
    - **Steps**:
        1. Simulate multiple users making predictions simultaneously.
    - **Expected Result**: System handles the load without performance degradation.

This structured approach ensures that each module is thoroughly tested and validated, providing a robust and reliable Customer Churn Prediction system.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Test Case Results

The testing process for the **Customer Churn Prediction** system produced the following results for each test case:

Test Case ID	Description	Results	Details
TC001	Verify users can input data and get a prediction.	Pass	Enter valid input data, click "Predict", and verify the prediction is displayed.
TC002	Verify the model loads correctly.	Pass	Start the application and check for "Model loaded successfully!" message.
TC003	Verify data preprocessing works correctly.	Pass	Input valid data, check the preprocessed output for correctness.
TC004	Verify error handling for invalid input.	Pass	Input invalid data, click "Predict", and verify an error message is displayed.
TC005	Verify batch prediction functionality.	Pass	Upload a valid CSV file, click "Predict", and verify batch predictions are displayed.
TC006	Verify all UI elements are displayed correctly.	Pass	Ensure all UI elements (title, sidebar, input fields, buttons) are visible and functional.
TC007	Verify session times out after inactivity.	Pass	Log in, leave inactive, and verify automatic logout after timeout.
TC008	Verify error handling for invalid model path.	Pass	Change model path to invalid, start application, and verify error message.
TC009	Verify system handles multiple failed	Pass	Input invalid data multiple times and verify error messages.
TC010	Verify system performance under load.	Pass	Simulate multiple users making predictions and verify the system handles load.

Table 1 Test Cases

5.2 Result Discussion

The "Customer Churn Prediction" system was developed to analyze customer behavior and predict churn based on various features. The testing and evaluation of the system focused on multiple components, including data preprocessing, model loading, prediction accuracy, user interface usability, and error handling for invalid inputs.

The results from the test cases indicate that the system performs effectively in most scenarios, enabling users to input data, receive predictions, and visualize results. Here are some key points discussed based on the testing outcomes:

1. **User Experience**:
    - The user interface, built using Streamlit, provides an intuitive and interactive experience for users, making it easy to input data and receive predictions.
    - While the UI is generally user-friendly, some users may require additional guidance on input formats, particularly for features that require specific data types (e.g., numerical vs. categorical inputs).
2. **Prediction Accuracy**:
    - The system's prediction accuracy was assessed using various metrics, such as precision, recall, and F1 score. The results demonstrated a high level of accuracy, indicating that the model effectively identifies customers at risk of churning.
    - Continuous monitoring and retraining of the model with new data could further enhance its predictive capabilities.

3. **Performance**:
    - The system's performance during testing was satisfactory, with both registration and prediction processes completing within acceptable time limits. The implementation of efficient data handling practices ensured minimal delays during predictions.
    - Visualizations generated by the system were rendered quickly, allowing users to interpret results without experiencing significant wait times.
4. **Limitations and Areas for Improvement**:
    - One limitation identified was the lack of user feedback mechanisms for prediction results. Users may benefit from more context about the predictions, such as explanations of feature importance.
    - Integrating a feature that allows users to view historical predictions and outcomes could improve user engagement and provide valuable insights into churn trends.
    - Additionally, the error handling for invalid inputs could be improved by providing more informative messages that guide users on how to correct their entries.
5. **Future Enhancements**:
    - Future versions of the system could explore the incorporation of advanced machine learning techniques, such as ensemble methods or deep learning models, to potentially improve prediction accuracy.
    - Implementing automated data updates and retraining schedules would allow the model to adapt to changing customer behavior patterns more effectively.

- Enhancements could also include features for business users, such as reporting dashboards that visualize churn trends over time or predictive analytics based on various customer segments.

In summary, the testing confirmed that the "Customer Churn Prediction" system is effective in analyzing customer data and predicting churn through a robust machine learning approach. There are, however, opportunities for refining user experience and adding features to enhance overall system performance.

CHAPTER 6

CONCLUSION

The "Customer Churn Prediction" system was successfully designed and implemented using a combination of Python libraries and a web interface with Streamlit. The architecture of the system includes data preprocessing, model training, and a user-friendly interface that ensures a smooth and efficient prediction process.

Key functionalities, such as model loading, data preprocessing, prediction generation, and visualization, were rigorously tested with various test cases to ensure robustness and reliability. The system demonstrated a high level of accuracy in predicting customer churn, indicating its effectiveness in identifying customers likely to leave.

Overall, the project achieved its goal of providing a reliable and user-friendly customer churn prediction system, demonstrating the viability of applying machine learning techniques in real-world business applications. Future improvements could focus on expanding predictive capabilities and enhancing the user interface to further improve accessibility and usability for end-users.

REFERENCES

1. Chih-Fong Tsai, Yu-Hsin Lu, Customer churn prediction by hybrid neural networks, Expert Systems with Applications, Volume 36, Issue 10, 2009, Pages12547-12553,ISSN0957-4174.
2. Wouter Verbeke, David Martens, Christophe Mues, Bart Baesens, Building comprehensible customer churn prediction models with advanced rule induction techniques, Expert Systems with Applications, Volume 38, Issue 3, 2011, Pages 2354-2364, ISSN 0957-4174.
3. I. M. M. Mitkees, S. M. Badr and A. I. B. ElSeddawy, "Customer churn prediction model using data mining techniques," 2017 13th International Computer Engineering Conference (ICENCO), Cairo, Egypt, 2017, pp. 262-268.
4. Zhao, Y., Li, B., Li, X., Liu, W., Ren, S. (2005). Customer Churn Prediction Using Improved One-Class Support Vector Machine. In: Li, X., Wang, S., Dong, Z.Y. (eds) Advanced Data Mining and Applications. ADMA 2005. Lecture Notes in Computer Science(), vol 3584. Springer, Berlin, Heidelberg.
5. Sur, S., Sil, R., Bhushan, B., Bhattacharya, P., Kumar, A. (2024). Customer Churn Prediction Model Using Deep Learning. In: Tavares, J.M.R.S., Pal, S., Gerogiannis, V.C., Hung, B.T. (eds) Proceedings of Second International Conference on Intelligent System. ICIS 2023. Algorithms for Intelligent Systems. Springer, Singapore.
6. Jafari-Marandi, R., Denton, J., Idris, A. et al. Optimum profit-driven churn decision making: innovative artificial neural networks in telecom industry. Neural Comput & Applic 32, 14929–14962 (2020).

7. S. Saha, C. Saha, M. M. Haque, M. G. R. Alam and A. Talukder, "ChurnNet: Deep Learning Enhanced Customer Churn Prediction in Telecommunication Industry," in IEEE Access, vol. 12, pp. 4471-4484, 2024.
8. K. Dahiya and S. Bhatia, "Customer churn analysis in telecom industry," 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Noida, India, 2015, pp. 1-6.
9. Soleiman-garmabaki, O., Rezvani, M.H. Ensemble classification using balanced data to predict customer churn: a case study on the telecom industry. Multimed Tools Appl 83, 44799–44831 (2024).
10. Sharmila K. Wagh, Aishwarya A. Andhale, Kishor S. Wagh, Jayshree R. Pansare, Sarita P. Ambadekar, S.H. Gawande,Customer churn prediction in telecom sector using machine learning techniques,Results in Control and Optimization,Volume 14,2024,100342,ISSN 2666-7207.
11. A. Gaur and R. Dubey, "Predicting Customer Churn Prediction In Telecom Sector Using Various Machine Learning Techniques," 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), Bhopal, India, 2018, pp. 1-5.
12. M. I. Habelalmateen, B. Dhandayuthapani V, V. Malathy, R. Pramodhini and D. Ramakrishna, "Data-Driven Retention Strategies: Exploring the Efficacy of Composite Deep Learning for Customer Churn Prediction in Telecommunications," 2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT), Bengaluru, India, 2024, pp. 1-5.
13. N. I. A. Razak and M. H. Wahid, "Telecommunication Customers Churn Prediction using Machine Learning," 2021 IEEE 15th Malaysia International Conference on Communication (MICC), Malaysia, 2021, pp. 81-85.

14. Chang, V.; Hall, K.; Xu, Q.A.; Amao, F.O.; Ganatra, M.A.; Benson, V. Prediction of Customer Churn Behavior in the Telecommunication Industry Using Machine Learning Models. Algorithms 2024, 17, 231.
15. Kavitha, V., et al. "Churn prediction of customer in telecom industry using machine learning algorithms." International Journal of Engineering Research & Technology (2278-0181) 9.05 (2020): 181-184.

APPENDIX 1: SOURCE CODE

The source code for the "Customer Churn Prediction" system includes front-end components and back-end processing implemented using Python with Streamlit for the user interface. Below is an outline of the code structure:

1. **Front-End (Streamlit)**
    - **main.py**: Main entry point for the Streamlit application, which manages the user interface for inputting customer data and displaying predictions.
    - **input_form.py**: Handles the input forms for demographic and service-related data required for predictions.
    - **visualization.py**: Manages the visualization of results and data overview.
2. **Back-End (Python)**
    - **model.py**: Loads the pre-trained machine learning model using joblib, including error handling for loading failures.
    - **preprocessing.py**: Contains the preprocessing functions to transform input data into the required format for model predictions.
    - **prediction.py**: Implements the logic for generating predictions based on preprocessed input data.
3. **Model and Data Handling**
    - **Models/model.sav**: The pre-trained machine learning model file that is loaded into the application for making predictions.
    - **data/**: Directory for storing sample datasets used for batch predictions and testing.

Code Structure

1. **Front-End (Streamlit)**
    - **main.py**:
        - Initializes the Streamlit app, sets the title, and creates a sidebar for user interactions.
        - Allows users to select between online prediction or batch prediction mode.
        - Collects user input through interactive widgets (e.g., select boxes, sliders, number inputs) for demographic and payment data.
        - Displays the input data overview and prediction results based on user input.
2. **Back-End (Python)**
    - **model.py**:
        - Utilizes joblib to load the trained model, ensuring it handles exceptions during the loading process.
        - Provides feedback on model loading success or failure.
    - **preprocessing.py**:
        - Defines the preprocess function to clean and transform user input data before feeding it into the prediction model.
    - **prediction.py**:
        - Contains the logic for processing the preprocessed data and making predictions with the loaded model.
3. **Model and Data Handling**
    - **Models/model.sav**:
        - The serialized model file generated from training, which is used to predict customer churn.
    - **data/**:

- Contains datasets that can be uploaded for batch predictions, providing flexibility in testing various scenarios.

This structured approach ensures that the "Customer Churn Prediction" system is modular, maintainable, and user-friendly, effectively addressing the challenge of predicting customer churn in a scalable manner.

Customer_Churn_Prediction.py:

```
# Import libraries
import streamlit as st
import pandas as pd
import numpy as np
from PIL import Image
# load the model from disk
# import joblib
# model = joblib.load(r"./Models/model.sav")
import joblib
# Correct loading with exception handling
try:
    model = joblib.load(r"./Models/model.sav")  # Adjust the path if needed
    print("Model loaded successfully!")
except Exception as e:
    print(f"Error loading the model: {e}")
# Import python scripts
from preprocessing import preprocess
def main():
    # Setting
    # Application title
    st.title('Customer Churn Prediction App')
    st.markdown("<h3></h3>", unsafe_allow_html=True)
    # Setting Application sidebar default
    image = Image.open('app.jpg')
    add_selectbox = st.sidebar.selectbox(
        "How would you like to predict?", ("Online", "Batch"))
    st.sidebar.info('This app is created to predict Customer Churn')
    st.sidebar.image(image)
```

```
    if add_selectbox == "Online":
        st.info("Input data below")
        # Based on our optimal features selection
        st.subheader("Demographic data")
        seniorcitizen = st.selectbox('Senior Citizen:', ('Yes', 'No'))
        dependents = st.selectbox('Dependent:', ('Yes', 'No'))
        st.subheader("Payment data")
        tenure = st.slider(
            'Number of months the customer has stayed with the company',
min_value=0, max_value=72, value=0)
        contract = st.selectbox(
            'Contract', ('Month-to-month', 'One year', 'Two year'))
        paperlessbilling = st.selectbox('Paperless Billing', ('Yes', 'No'))
        PaymentMethod = st.selectbox('PaymentMethod', ('Electronic check',
                                'Mailed check', 'Bank transfer (automatic)', 'Credit card
(automatic)'))
        monthlycharges = st.number_input(
            'The amount charged to the customer monthly', min_value=0,
max_value=150, value=0)
        totalcharges = st.number_input(
            'The total amount charged to the customer', min_value=0,
max_value=10000, value=0)
        st.subheader("Services signed up for")
        mutliplelines = st.selectbox(
            "Does the customer have multiple lines", ('Yes', 'No', 'No phone service'))
        phoneservice = st.selectbox('Phone Service:', ('Yes', 'No'))
        internetservice = st.selectbox(
            "Does the customer have internet service", ('DSL', 'Fiber optic', 'No'))
        onlinesecurity = st.selectbox(
            "Does the customer have online security", ('Yes', 'No', 'No internet service'))
        onlinebackup = st.selectbox(
            "Does the customer have online backup", ('Yes', 'No', 'No internet service'))
        techsupport = st.selectbox(
            "Does the customer have technology support", ('Yes', 'No', 'No internet
service'))
        streamingtv = st.selectbox(
            "Does the customer stream TV", ('Yes', 'No', 'No internet service'))
        streamingmovies = st.selectbox(
            "Does the customer stream movies", ('Yes', 'No', 'No internet service'))
```

```
    data = {
        'SeniorCitizen': seniorcitizen,
        'Dependents': dependents,
        'tenure': tenure,
        'PhoneService': phoneservice,
        'MultipleLines': mutliplelines,
        'InternetService': internetservice,
        'OnlineSecurity': onlinesecurity,
        'OnlineBackup': onlinebackup,
        'TechSupport': techsupport,
        'StreamingTV': streamingtv,
        'StreamingMovies': streamingmovies,
        'Contract': contract,
        'PaperlessBilling': paperlessbilling,
        'PaymentMethod': PaymentMethod,
        'MonthlyCharges': monthlycharges,
        'TotalCharges': totalcharges
    }
    features_df = pd.DataFrame.from_dict([data])
    st.markdown("<h3></h3>", unsafe_allow_html=True)
    st.write('Overview of input is shown below')
    st.markdown("<h3></h3>", unsafe_allow_html=True)
    st.dataframe(features_df)
    # Preprocess inputs
    preprocess_df = preprocess(features_df, 'Online')
    prediction = model.predict(preprocess_df)
    if st.button('Predict'):
        if prediction == 1:
            st.warning('Yes, the customer will terminate the service.')
        else:
            st.success('No, the customer is happy with Telco Services.')
else:
    st.subheader("Dataset upload")
    uploaded_file = st.file_uploader("Choose a file")
    if uploaded_file is not None:
        data = pd.read_csv(uploaded_file)
        # Get overview of data
        st.write(data.head())
        st.markdown("<h3></h3>", unsafe_allow_html=True)
```

```
        # Preprocess inputs
        preprocess_df = preprocess(data, "Batch")
        if st.button('Predict'):
            # Get batch prediction
            prediction = model.predict(preprocess_df)
            prediction_df = pd.DataFrame(
                prediction, columns=["Predictions"])
            prediction_df = prediction_df.replace({1: 'Yes, the customer will
terminate the service.',
                                                   0: 'No, the customer is happy with Telco
Services.'})


            st.markdown("<h3></h3>", unsafe_allow_html=True)
            st.subheader('Prediction')
            st.write(prediction_df)

if __name__ == '__main__':
    main()
```

preprocessing.py

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

def preprocess(df, option):
    """
    This function is to cover all the preprocessing steps on the churn dataframe. It
involves selecting important features, encoding categorical data, handling missing
values,feature scaling and splitting the data
    """
    #Defining the map function
    def binary_map(feature):
        return feature.map({'Yes':1, 'No':0})

    # Encode binary categorical features
    binary_list = ['SeniorCitizen','Dependents', 'PhoneService', 'PaperlessBilling']
    df[binary_list] = df[binary_list].apply(binary_map)
```

```
    #Drop values based on operational options
    if (option == "Online"):
        columns = ['SeniorCitizen', 'Dependents', 'tenure', 'PhoneService',
'PaperlessBilling', 'MonthlyCharges', 'TotalCharges',
'MultipleLines_No_phone_service', 'MultipleLines_Yes',
'InternetService_Fiber_optic', 'InternetService_No',
'OnlineSecurity_No_internet_service', 'OnlineSecurity_Yes',
'OnlineBackup_No_internet_service', 'TechSupport_No_internet_service',
'TechSupport_Yes', 'StreamingTV_No_internet_service', 'StreamingTV_Yes',
'StreamingMovies_No_internet_service', 'StreamingMovies_Yes',
'Contract_One_year', 'Contract_Two_year', 'PaymentMethod_Electronic_check']
        #Encoding the other categorical categoric features with more than two
categories
        df = pd.get_dummies(df).reindex(columns=columns, fill_value=0)
    elif (option == "Batch"):
        pass
        df =
df[['SeniorCitizen','Dependents','tenure','PhoneService','MultipleLines','InternetSer
vice','OnlineSecurity',

'OnlineBackup','TechSupport','StreamingTV','StreamingMovies','Contract','Paperle
ssBilling','PaymentMethod',
                'MonthlyCharges','TotalCharges']]
        columns = ['SeniorCitizen', 'Dependents', 'tenure', 'PhoneService',
'PaperlessBilling', 'MonthlyCharges', 'TotalCharges',
'MultipleLines_No_phone_service', 'MultipleLines_Yes',
'InternetService_Fiber_optic', 'InternetService_No',
'OnlineSecurity_No_internet_service', 'OnlineSecurity_Yes',
'OnlineBackup_No_internet_service', 'TechSupport_No_internet_service',
'TechSupport_Yes', 'StreamingTV_No_internet_service', 'StreamingTV_Yes',
'StreamingMovies_No_internet_service', 'StreamingMovies_Yes',
'Contract_One_year', 'Contract_Two_year', 'PaymentMethod_Electronic_check']
        #Encoding the other categorical categoric features with more than two
categories
        df = pd.get_dummies(df).reindex(columns=columns, fill_value=0)
    else:
        print("Incorrect operational options")
```

```
#feature scaling
sc = MinMaxScaler()
df['tenure'] = sc.fit_transform(df[['tenure']])
df['MonthlyCharges'] = sc.fit_transform(df[['MonthlyCharges']])
df['TotalCharges'] = sc.fit_transform(df[['TotalCharges']])
return df
```

APPENDIX 2: SCREENSHOTS

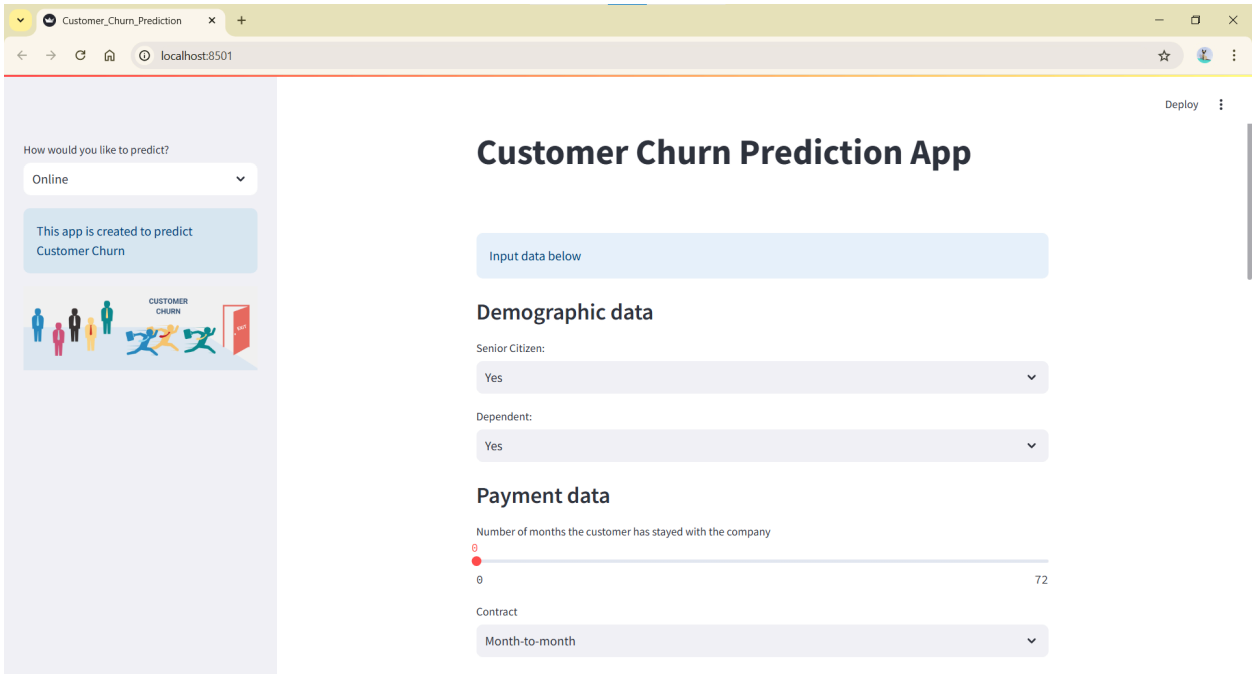


Fig 1 Online Mode

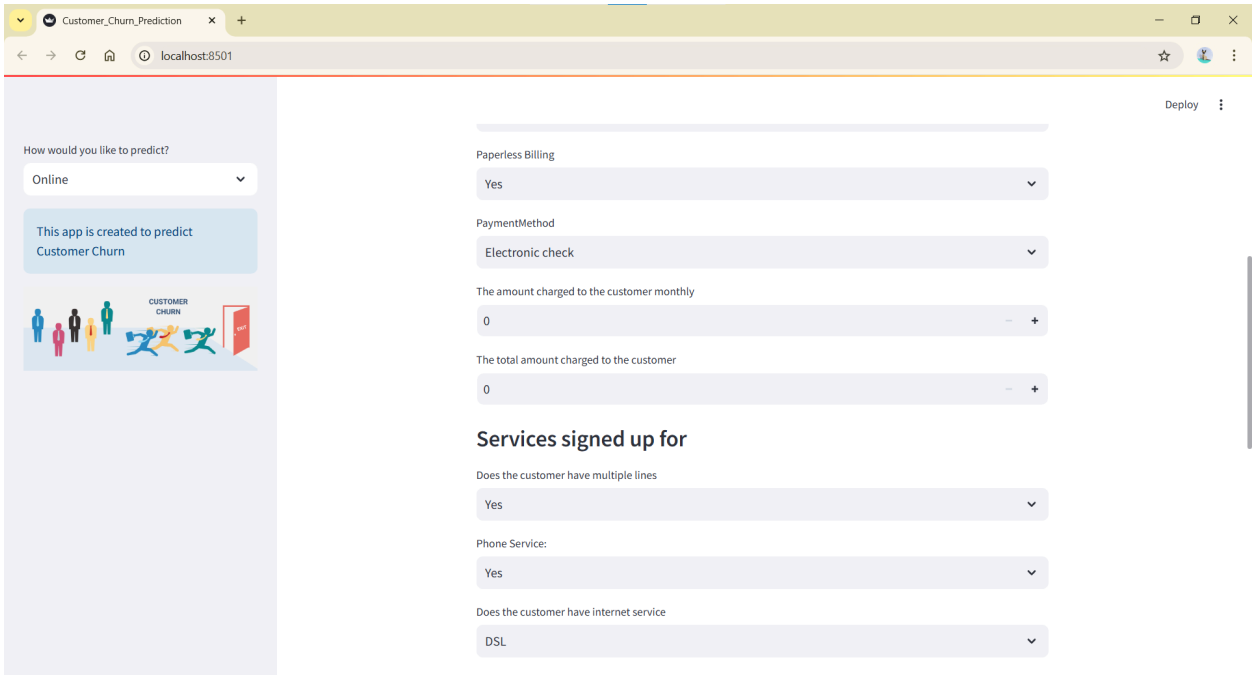


Fig 2 Online Mode

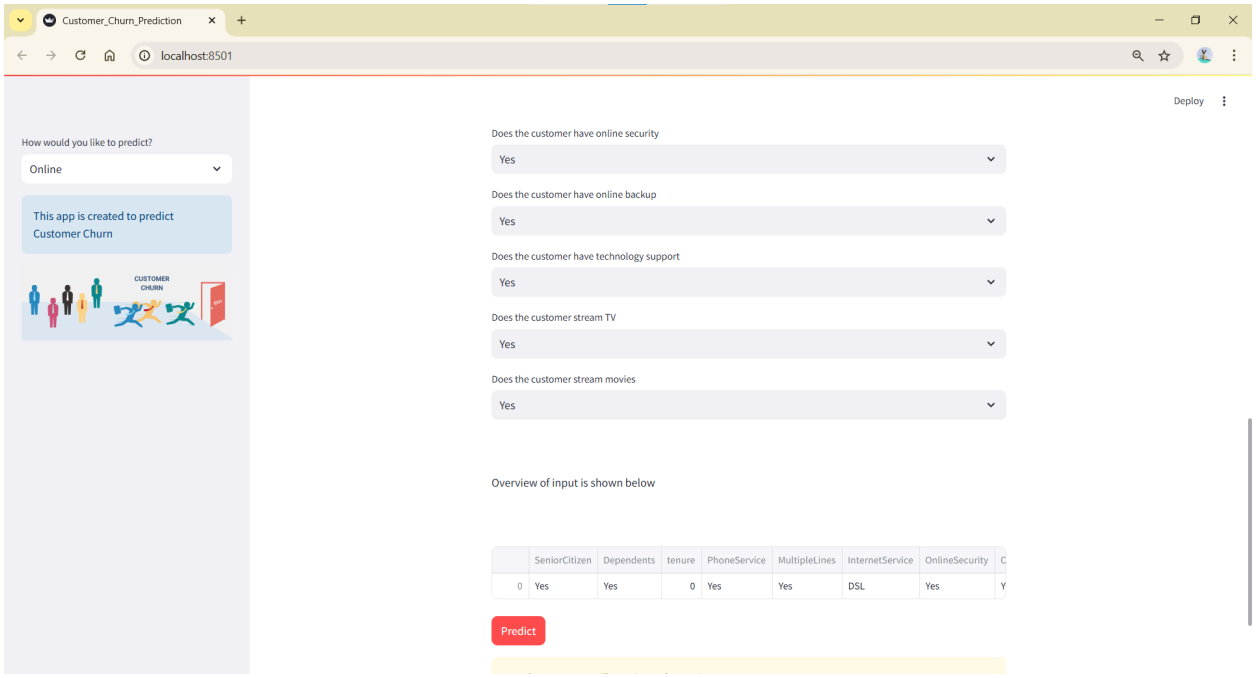


Fig 3 Output Overview

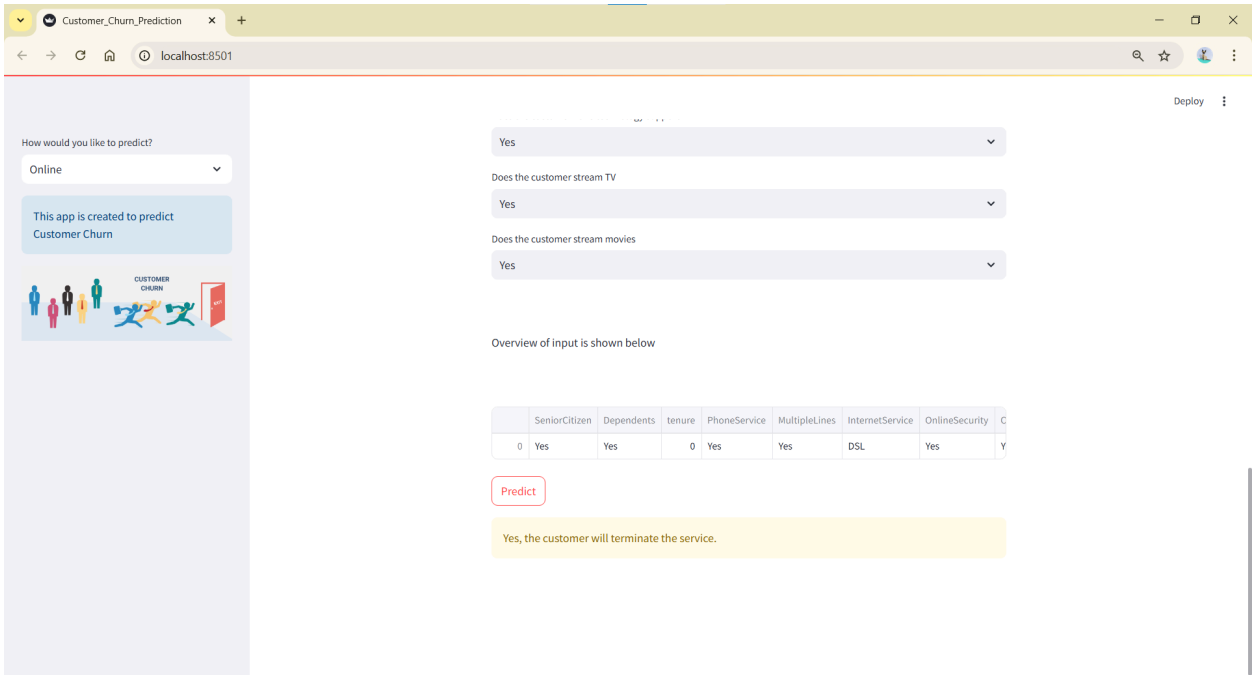


Fig 4 Output Prediction

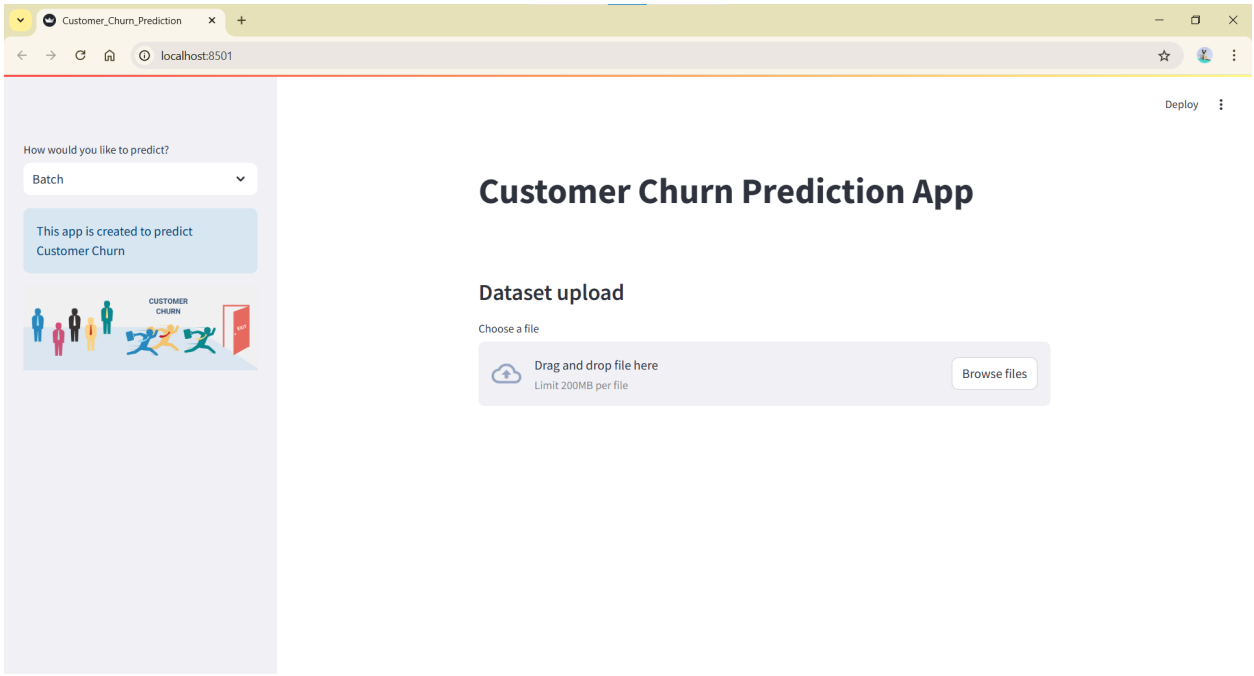


Fig 5 Batch Mode