Summary of Phase 1:

In Phase 1 of the project, the primary objective was to analyze customer churn within an e-commerce company. By leveraging a dataset consisting of both personal and transactional information, we aimed to develop insights into the factors contributing to churn behavior. The process began with data preprocessing, which involved tasks such as handling missing values, rounding off numerical columns, removing duplicates, standardizing categorical labels, and addressing outliers. Additionally, we conducted exploratory data analysis (EDA) to gain deeper insights into the dataset's characteristics and relationships between variables. Through techniques like count plots, correlation matrices, pie charts, violin plots, and multivariate analyses, we uncovered patterns and trends related to churn, customer preferences, satisfaction scores, payment methods, and more.

Overall, Phase 1 laid the foundation for subsequent phases, setting the stage for the development of predictive models and retention strategies aimed at improving customer satisfaction and loyalty in the e-commerce domain.

Summary of Phase 2:

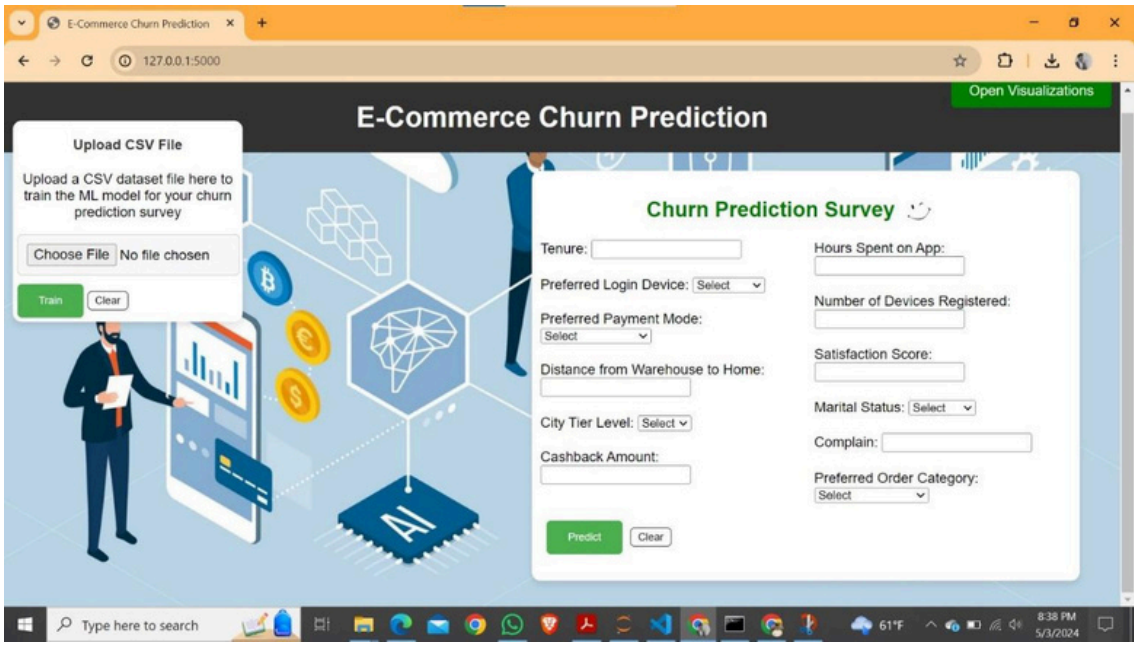
In Phase 2 of the project, we implemented various machine learning models on pre-processed e-commerce data to predict customer churn. After preprocessing steps, including label encoding for categorical variables and scaling for improved algorithm convergence, we selected a range of classification algorithms suited for binary outcome prediction. These included Logistic Regression, Naive Bayes, Decision Tree, Random Forest, K Nearest Neighbors, Support Vector Machine, and Quadratic Discriminant Analysis. By splitting the dataset into training and testing sets with an 80-20 ratio, we ensured robust learning while evaluating each model's generalization performance on unseen data. Through rigorous analysis and visualizations, we examined the predictive capabilities of each model to identify the most effective approaches for churn prediction.

Among the implemented algorithms, Random Forest emerged as the top-performing model with an accuracy of 97.78%, showcasing the effectiveness of ensemble methods for our dataset. Decision Tree also exhibited strong performance with an accuracy of 94.32%, while KNN and SVM demonstrated similar accuracies of around 92%, indicating their capability to capture complex patterns in the data. Naive Bayes and QDA, though showing slightly lower accuracies, provided valuable insights despite their probabilistic assumptions. Logistic Regression, while having the lowest accuracy, still contributed to the understanding of churn prediction. Overall, our comprehensive evaluation of machine learning models equipped us with actionable insights into customer churn behavior in the e-commerce domain, facilitating informed decision-making for retention strategies and business growth.

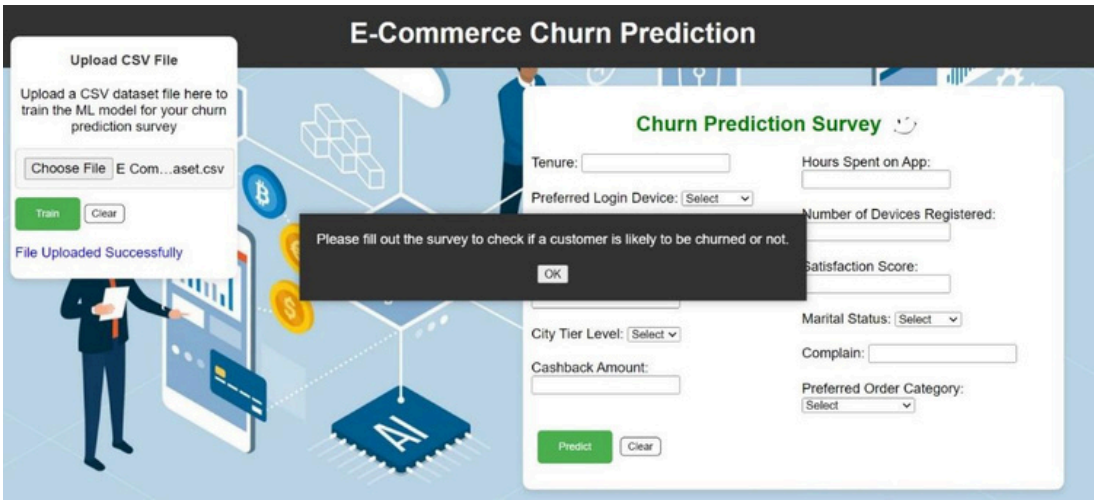
PHASE – 3

We have developed a model for predicting churn in an E-Commerce platform based on various factors in Phase 2. In this phase, we deployed the trained model using the Flask framework, leveraging Python. The outcome of this phase is a web application. We used the Random Forest algorithm for training our model, which achieved an impressive accuracy of 97.78%.

Web Application for Predicting the Churn Status:



This web application allows users to input customer data and receive predictions on whether they are likely to churn from an E-Commerce platform. The home page provides an option to use sample data as input in the form of CSV file. We have carefully selected features from the dataset based on their correlation with churn, ensuring that only the most relevant features are used for prediction. Here, users are prompted to select a file from their local device. Once the dataset has been loaded, we can proceed by clicking the "Train" button. Once the "Train" button is clicked and the file is uploaded successfully, the web page will present
the users with a form or various fields to fill out. These fields include various aspects like user activity, location, preferences, satisfaction, and engagement, all crucial for predicting churn in an e-commerce platform.

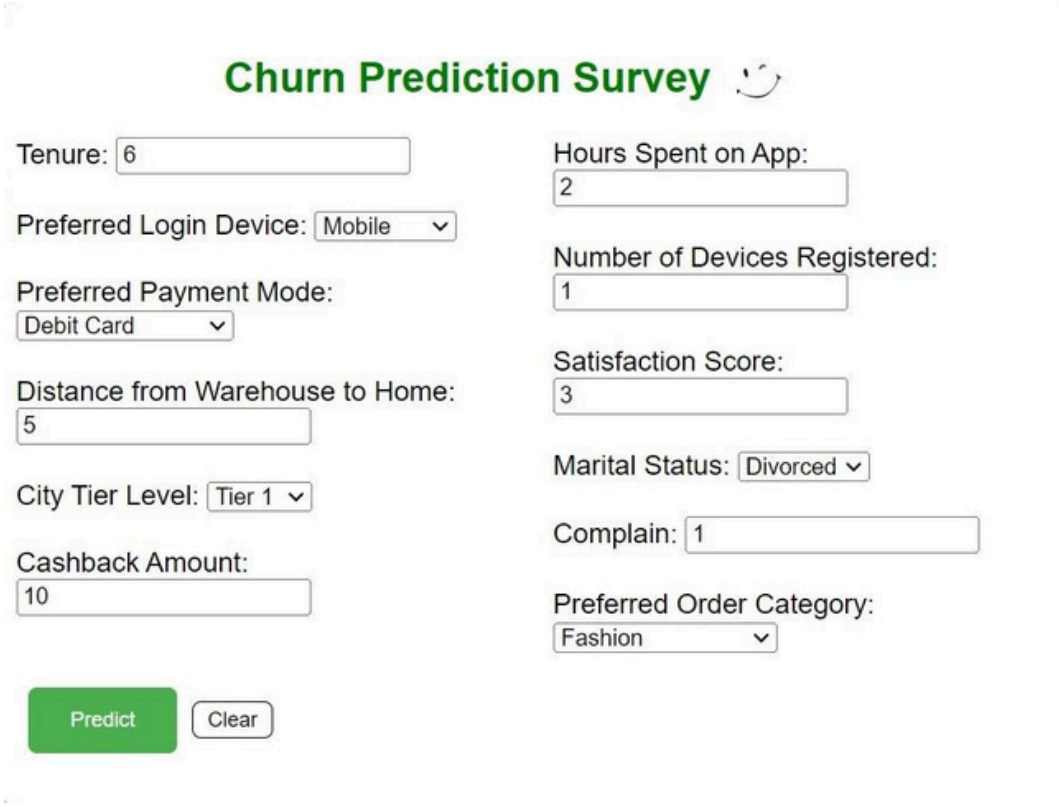


Fields in the web page to fill:

- **Tenure:** Tenure represents the duration for which a customer has been engaged with the e-commerce platform, reflecting their loyalty and commitment.
- **Preferred Login Device:** Preferred login device signifies the device type that a customer predominantly uses to access the e-commerce platform, providing insights into their browsing habits.
- **City Tier**: City tier categorizes the cities based on their population size, infrastructure, and economic development level, influencing customer behavior and purchasing power.
- **Distance From Warehouse to Home:** Warehouse-to-home distance measures the proximity of a customer's residence to the nearest warehouse or distribution center, impacting delivery time and service quality.
- **Preferred Payment Mode:** Preferred payment mode indicates the customer's favored method of making transactions on the e-commerce platform, reflecting their payment preferences and habits.
- **Hours Spend on App**: Hour spent on the app denotes the average time a customer dedicates to using the e-commerce platform's mobile application, reflecting their engagement level and app usage behavior.
- **Number Of Devices Registered**: Number of devices registered represents the count of electronic devices associated with a customer's account, indicating their multi-device usage.
- **Satisfaction Score:** Satisfaction score quantifies the level of satisfaction reported by customers regarding their overall experience with the e-commerce platform, serving as a key indicator of customer loyalty and retention.
- **Marital Status:** Marital status indicates whether a customer is single, married, divorced, or in another marital status category, providing demographic insights that may influence purchasing behavior.

- **Complain:** Complain represents the number of complaints or grievances a customer has lodged with the e-commerce platform, indicating dissatisfaction and potential churn risk.
- **Cashback Amount:** Cashback amount reflects the total monetary value of cashback rewards earned by a customer through transactions on the e-commerce platform, influencing their loyalty and repeat purchases.
- **Preferred Order Category:** Preferred order category signifies the type of products or services that a customer frequently purchases or shows interest in, reflecting their preferences and buying patterns.

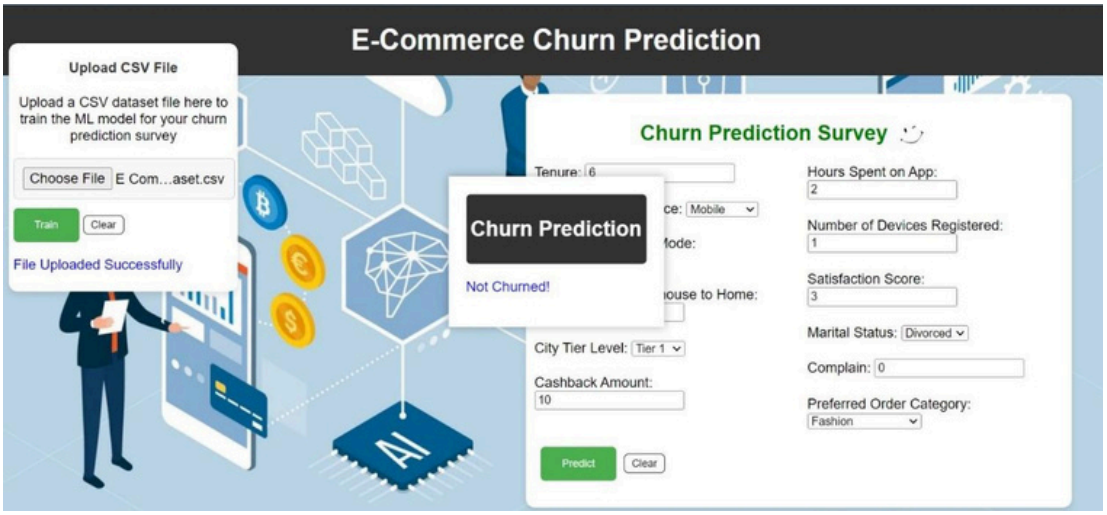
The above fields are filled by the user with their own values and the page is as following:



After clicking on the "Predict" button, a pop-up will appear indicating whether the customer is churned or not.

Churned: Refers to customers who have ceased engaging with the E-Commerce platform, indicating a decline in activity such as making purchases or visiting the site. Churned customers may have discontinued their relationship with the platform for various reasons, including dissatisfaction or lack of interest.

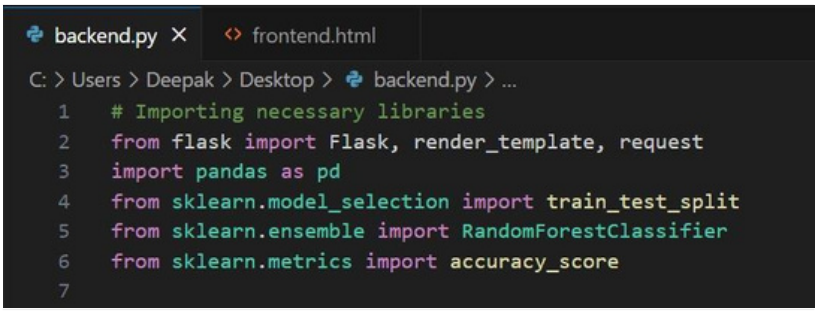
Not Churned: Denotes customers who continue to actively engage with the E-Commerce platform, demonstrating ongoing interaction such as regular purchases or site visits. These customers contribute to the platform's growth and revenue by maintaining their involvement and loyalty over time.



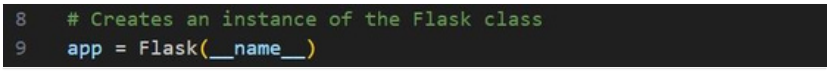
After clicking the "Open Visualizations" button, users are presented with a series of visualization images aimed at simplifying the exploration and comprehension of data pertaining to churn. These visual aids, derived from the exploratory data analysis conducted earlier, offer users an intuitive interface to delve into various aspects of churn prediction, enhancing their understanding of the underlying patterns and trends within the dataset

Explanation of the Application Code:

We chose Flask, a framework based on Python, to build the web app. Here is a snippet of the code where we import the necessary libraries.



This code snippet shows how we create an instance of a Flask web application.



The decorator @app.route('/') links the function below it with the root URL path ("/"). When a request comes in for the base URL of the application, this instructs the server to run the home() function to handle that request. The home() function will then provide a response, like a webpage, which will be sent back to the user's browser in reply to their initial request for the homepage or main application URL.

Upon completion of training, it displays the 'frontend.html' template, signaling that the model is trained, and training has been completed.

```python
# Route for training the model
@app.route('/train', methods=['POST'])
def train_model():
    global preds, acc, trained, minmax, X_train, X_test, y_train, y_test, model
    # Check if a file was uploaded or not
    if 'csv-file' not in request.files:
        message = "No file uploaded!"
        return render_template('frontend.html', message=message)
    # Reads the uploaded CSV file in the label name of csv-file from frontend
    df = pd.read_csv(request.files['csv-file'])
    # Computes min-max normalization ranges for each feature
    for i in df.columns[1:]:
        minmax.append([df[i].min(), df[i].max()])
    # Performs min-max normalization on the features
    for i in df.columns[1:]:
        df[i] = (df[i] - df[i].min()) / (df[i].max() - df[i].min())
    # Splits the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(df.iloc[:, 1:], df.iloc[:, 0], test_size=0.2, random_state=42)
    # Trains the random forest model
    model, preds, acc = randomforest(X_train, y_train)
    # Provides the feedback to the user wheather the file is uploaded or not
    message = "File uploaded and model trained successfully!"
    # Renders the 'frontend.html'
    return render_template('frontend.html', trained=True, model=model, preds=preds, acc=acc, train=True, message=message)

```

The following function returns the trained Random Forest model, predictions, and accuracy. We utilized 100 estimators for the Random Forest model.

```python
# Function to train the random forest model
def randomforest(X_train, y_train):
    global preds, acc, trained, minmax, X_test, y_test, model
    # Initializes and train the random forest classifier
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)
    # Makes the predictions on the test set
    rf_preds = rf_model.predict(X_test)
    # Computes accuracy score
    rf_acc = accuracy_score(y_test, rf_preds)
    return rf_model, rf_preds, rf_acc
```

It gathers form data submitted by the user, scales the features via the Min-Max scaler, and readies them for model prediction. The features are structured into a list to align with the model's requirement for a list of samples, enabling the model to predict the outcome based on the given features. The predicted result is then passed to the frontend, where it is retrieved in the predict function and it is then displayed on Web Page.

```python
# Route for predicting churn
@app.route('/predict', methods=['POST'])
def predict():
    global model, minmax
    if request.method == 'POST':
        # Extracting the feature values from the form and perform min-max normalization
        features = []
        features.append((float(request.form['E-Comm-Tenure']) - minmax[0][0]) / (minmax[0][1] - minmax[0][0]))  # Normalizing '
        features.append(float(request.form['Preferred-Login-Device']))  # No normalization needed for 'PreferredLoginDevice'
        features.append((float(request.form['Warehouse-To-Home']) - minmax[3][0]) / (minmax[3][1] - minmax[3][0]))  # Normalizi
        features.append(float(request.form['Preferred-Payment-Mode']))  # No normalization needed for 'PreferredPaymentMode'
        features.append((float(request.form['Hour-Spend-On-App']) - minmax[4][0]) / (minmax[4][1] - minmax[4][0]))  # Normalizi
        features.append((float(request.form['Number-Of-Device-Registered']) - minmax[5][0]) / (minmax[5][1] - minmax[5][0]))  #
        features.append((float(request.form['Satisfaction-Score']) - minmax[6][0]) / (minmax[6][1] - minmax[6][0]))  # Normaliz
        features.append(float(request.form['City-Tier-Level']))  # No normalization needed for 'CityTier'
        features.append(float(request.form['CashbackAmount']))  # No normalization needed for 'CashbackAmount'
        features.append((float(request.form['Marital-Status']) - minmax[7][0]) / (minmax[7][1] - minmax[7][0]))  # Normalizing
        features.append(float(request.form['Complain']))  # No normalization needed for 'Complain'
        features.append(float(request.form['Preferred-Order-Category']))  # No normalization needed for 'PreferedOrderCategory'
        # Reshapes the features
        features = [features]
        # Makes prediction using the trained model
        prediction = model.predict(features)[0]
        # Returns the prediction to frontend to display the results
        return str(prediction)
```

This code segment verifies whether the file is executed directly or imported, ensuring that the Flask app runs only when directly executed. Enabling debug mode with debug=True activates debugging functionalities such as auto-reloading upon code modifications, facilitating interactive development without the need for restarts. This approach prevents unintended executions when imported as a module.


```python
# Route for predicting churn
@app.route('/predict', methods=['POST'])
def predict():
    global model, minmax
    if request.method == 'POST':
        # Extracting the feature values from the form and perform min-max normalization
        features = []
        features.append((float(request.form['E-Comm-Tenure']) - minmax[0][0]) / (minmax[0][1] - minmax[0][0]))  # Normalizing '
        features.append(float(request.form['Preferred-Login-Device']))  # No normalization needed for 'PreferredLoginDevice'
        features.append((float(request.form['Warehouse-To-Home']) - minmax[3][0]) / (minmax[3][1] - minmax[3][0]))  # Normalizi
        features.append(float(request.form['Preferred-Payment-Mode']))  # No normalization needed for 'PreferredPaymentMode'
        features.append((float(request.form['Hour-Spend-On-App']) - minmax[4][0]) / (minmax[4][1] - minmax[4][0]))  # Normalizi
        features.append((float(request.form['Number-Of-Device-Registered']) - minmax[5][0]) / (minmax[5][1] - minmax[5][0]))  #
        features.append((float(request.form['Satisfaction-Score']) - minmax[6][0]) / (minmax[6][1] - minmax[6][0]))  # Normaliz
        features.append(float(request.form['City-Tier-Level']))  # No normalization needed for 'CityTier'
        features.append(float(request.form['CashbackAmount']))  # No normalization needed for 'CashbackAmount'
        features.append((float(request.form['Marital-Status']) - minmax[7][0]) / (minmax[7][1] - minmax[7][0]))  # Normalizing
        features.append(float(request.form['Complain']))  # No normalization needed for 'Complain'
        features.append(float(request.form['Preferred-Order-Category']))  # No normalization needed for 'PreferedOrderCategory'
        # Reshapes the features
        features = [features]
        # Makes prediction using the trained model
        prediction = model.predict(features)[0]
        # Returns the prediction to frontend to display the results
        return str(prediction)
```


Instructions to run our web application:

1. Run the backend.py file to launch the Flask application in the terminal with the command, python backend.py. This command launches the Flask development server, and you'll see a message confirming that the server is running. You can then access the application through your web browser at http://127.0.0.1:5000/.

2. Open your browser and go to http://127.0.0.1:5000/. You will see the home page of the E Commerce Churn Prediction application.

3. On the home page, upload the sample dataset. The model will take some time to train based on this dataset.

4. Once the training is complete, you can enter user details. Carefully fill out the various fields provided.

5. Click the predict button to receive the prediction whether the customer is churned or not.

6. On the same page, you will find a pop up containing the prediction result.

7. To stop the Flask development server, return to the terminal and press Ctrl + C.

With our application, users can easily navigate to understand the churn status and explore trends and patterns in the data that determine the churn status.

JavaScript Functions Overview:

uploadFile ():

- Handles CSV file upload for training the ML model.
- Sends the file to the server for training.
- Displays success and instructions popups.

predict ():

- Processes user input for churn prediction.
- Sends data to the server for prediction.
- Displays churn prediction result in a popup

Incorporating of the best Machine Learning Model from Phase 2:

In the previous phase of our project, we explored various machine learning models to predict churn in an e-commerce dataset. Despite experimenting with different models, we found that the Random Forest model consistently outperformed the others in terms of accuracy. Therefore, for this phase of the project, we decided to integrate the Random Forest model into our Flask application. The same code used to implement the Random Forest model in phase two has been incorporated into our Python code for the Flask application. When users upload an e-commerce dataset from the application's homepage, it is used to train the Random Forest model. Subsequently, user-provided input data is normalized and sent to the model for prediction, benefiting from the insights gained during the training phase.

Normalization Using Min-Max Scaling on User's Input:

Ensuring consistent and effective model training was a key focus during Phase 3 of our project, particularly regarding the normalization of input features.

we applied Min-Max scaling to normalize user-provided input data before making predictions in our Flask application. These scaling parameters, including min_values and max_values, were specifically obtained during Phase 3 and tailored to our Random Forest model, which was chosen for its accuracy with our e-commerce dataset.

Upon initializing the application, the pre-trained Random Forest model along with its corresponding Min-Max scaling parameters were loaded into the Flask application's backend.py file. When users submit data for churn prediction, these input features are normalized using the Min-Max scaling parameters, ensuring alignment with the expectations of the Random Forest model for precise churn predictions.

The project holds promise for advancing churn prediction in the e-commerce sector:

The project has a lot of potential to improve how we predict if customers will stop using an e-commerce platform. By using advanced techniques in machine learning and optimization, the app aims to make these predictions more accurate and reliable. It fine-tunes its settings and combines different methods to be a useful tool for businesses in the e-commerce industry.

It also makes it easy for people to use through a simple web interface, making it accessible to anyone in the industry. Plus, it pays attention to details like how it scales data, which helps make its predictions more accurate. Overall, it is not just about predicting churn—it is about offering a flexible tool to improve how e-commerce companies make decisions.

Suggestions and Observations:

Our churn prediction website helps users understand the factors influencing churn and provides quick and accurate assessments based on input data, benefiting both businesses and customers. We are considering adding a feature that highlights the most significant factors affecting churn, empowering users to make informed decisions. Users can experiment with different values to see how they impact churn predictions, enhancing their understanding and decision-making. In the future, we might introduce a recommendation feature suggesting strategies based on users' profiles and offering a choice of machine learning models to explore diverse prediction approaches.

Benefits of Our Churn Prediction Application:

Utilizing advanced machine learning techniques, our application delivers precise predictions based on relevant factors in the e-commerce domain. This empowers users with valuable insights into customer churn and aids in strategic decision-making.

Gain a deeper understanding of customer behavior by exploring the fundamental factors that drive churn predictions. Our application serves as an educational resource, offering users comprehensive analyses of churn-related data to enhance their understanding of customer retention dynamics.