**Customer Conversion Analysis for Online Shopping Using Clickstream Data**

**Introduction**

In the fast-growing e-commerce industry, analysing customer behaviour is essential for enhancing engagement, optimizing marketing strategies, and maximizing revenue. Clickstream data, which captures user interactions on a website, provides valuable insights into browsing patterns and purchasing tendencies. By leveraging machine learning techniques, businesses can predict customer conversions, estimate potential revenue, and segment users for targeted marketing.

This project focuses on utilizing **clickstream data** to develop an intelligent and interactive Streamlit web application that enhances decision-making in e-commerce. The application incorporates classification, regression, and clustering models to extract meaningful insights from online shopping behaviour.

**Objective**

The analysis of clickstream data aims to address three key areas:

* **Customer Conversion Prediction (Classification):** Predicting whether a customer will complete a purchase based on browsing behaviour.
* **Revenue Estimation (Regression):** Forecasting potential revenue from individual users to optimize pricing and marketing strategies.
* **Customer Segmentation (Clustering):** Grouping customers based on their online behavior to enable personalized recommendations and targeted advertising.

By implementing data-driven approaches, businesses can improve customer retention, enhance user experience, and increase overall sales.

**Problem Statement**

This project aims to develop a **Streamlit-based web application** that uses machine learning to analyse customer behaviour and predict important outcomes:

1. **Will a customer complete a purchase?** (Classification)
2. **How much revenue can a customer generate?** (Regression)
3. **How can customers be grouped based on their browsing patterns?** (Clustering)

**Dataset:**

The dataset includes the following features:

| **Variable** | **Description** |
| --- | --- |
| YEAR | Year of the recorded session (e.g., 2008) |
| MONTH | Month of the recorded session (from April (4) to August (8)) |
| DAY | Day of the month |
| ORDER | Sequence of clicks during one session |
| COUNTRY | Country of origin of the IP address (e.g., USA, India, Germany) |
| SESSION ID | Unique session identifier |
| PAGE 1 (MAIN CATEGORY) | Main product category (e.g., trousers, skirts, blouses, sale) |
| PAGE 2 (CLOTHING MODEL) | Specific clothing model identifier (217 products) |
| COLOUR | Color of the product (e.g., black, white, blue, etc.) |
| LOCATION | Photo location on the webpage (divided into six parts: top left, top middle, etc.) |
| MODEL PHOTOGRAPHY | Model photography type (1 - en face, 2 - profile) |
| PRICE | Price of the product in US dollars |
| PRICE 2 | Indicates if the product price is higher than the category average (1 - Yes, 2 - No) |
| PAGE | Page number within the e-store website (from 1 to 5) |

**Project Folder Structure & File Purpose**

**Folder: .zen**

The. zen folder is automatically created when initializing ZenML using the “zenml init” command. It contains configuration files, metadata, and database storage for managing ZenML pipelines.

| **File/Folder Name** | **Purpose** |
| --- | --- |
| config.yaml | Stores ZenML configuration settings, including pipeline metadata, storage paths, and environment variables. |
| local\_stores/ | Contains local storage for ZenML metadata and database files. |
| ├── c6b32bb5-3559-4a2c-b63d-9276842853e2/ | A unique identifier for a ZenML local store instance. Used to manage multiple local storage instances. |
| ├── default\_zen\_store/ | The default storage location for ZenML metadata and database files. |
| │ ├── zenml.db | A database file that stores metadata related to ZenML pipeline runs, configurations, and tracking. |

**Folder: data**

The data folder contains datasets used for training, testing, and bulk predictions in the ZenML pipeline. It also includes a data description file explaining the dataset attributes.

| **File Name** | **Purpose** |
| --- | --- |
| bulk\_prediction\_data.csv | Dataset used for making batch predictions on multiple inputs at once. |
| data description.txt | Contains metadata about the dataset, including variable descriptions, column meanings, and citation details. The dataset pertains to an e-shop clothing website in 2008. |
| test\_data.csv | A subset of data used for model evaluation and validation. This ensures that the model performs well on unseen data. |
| train\_data.csv | The primary dataset used for training the machine learning model. This helps the model learn patterns and relationships within the data. |

**Folder: docker**

The docker folder contains Docker configurations for running different components of the project in isolated environments. It enables **containerized deployment** of the ML pipeline, MLflow tracking, and Streamlit web application.

| **File/Folder Name** | **Purpose** |
| --- | --- |
| docker.mainpipeline | Defines the Docker image for running the main ZenML pipeline, installing dependencies from requirements.txt, and setting up an executable entry point. |
| docker.mlflow | Sets up an **MLflow tracking server** inside a container, storing experiment metadata in /app/mlruns and exposing port **5000** for tracking model runs. |
| docker.streamlit | Creates a **Streamlit app container**, installing necessary dependencies and running the UI on **port 8501**. |
| mlruns/ | Stores experiment tracking data generated by MLflow. |
| ├── .trash/ | Stores deleted or outdated MLflow run artifacts for reference or cleanup. |
| ├── models/ | Stores trained models, logs, and artifacts generated during MLflow runs. |

**Folder: documents**

The documents folder contains reference materials and troubleshooting guides related to the implementation of **ZenML** and other components of the project.

| **File Name** | **Purpose** |
| --- | --- |
| Errors.docx | Documents errors encountered during the ZenML implementation process along with their solutions. This serves as a troubleshooting guide for debugging and resolving common issues. |

**Folder: images**

The images folder contains visual assets used in the **Streamlit web application** (main.py). These images enhance the UI design and branding of the application.

| **File Name** | **Purpose** |
| --- | --- |
| bg\_image.png | Background image used in the Streamlit interface to enhance the user experience. |
| logo.png | Logo displayed in the sidebar or header of the application to maintain branding. |

**Folder: mlruns**

The mlruns folder is used by **MLflow** to store experiment tracking data, including model runs, artifacts, and metadata. It organizes logs for each run and facilitates model evaluation and comparison.

| **File/Folder Name** | **Purpose** |
| --- | --- |
| .trash/ | Stores deleted or outdated MLflow run artifacts for reference or cleanup. |
| 0/ | Represents the default MLflow experiment where all initial runs are logged before custom experiments are set up. |
| 550180154531703585 *(Example Run ID)* | A unique identifier for an MLflow experiment run, containing logs, metrics, parameters, and model artifacts. |
| 630753251537639831 *(Example Run ID)* | Another experiment run with its own logs and stored metadata. Each run ID corresponds to a separate execution of a machine learning pipeline. |
| 800721595944702217 *(Example Run ID)* | Additional run logs for tracking model performance and hyperparameters. |
| 877744070416553456 *(Example Run ID)* | Stores experiment tracking information for a specific model training session. |
| 976576929185389185 *(Example Run ID)* | Logs a different execution with its associated metadata and artifacts. |
| models/ | Stores trained models and associated artifacts registered in MLflow. |

**Folder: models**

The models folder stores all trained machine learning models, categorized into **classification, regression, and clustering models**. It also includes the **best-performing models** identified through evaluation.

| **File/Folder Name** | **Purpose** |
| --- | --- |
| best\_models/ | Stores metadata about the best-selected models for each ML task. |
| ├── best\_models\_file.csv | A CSV file that records the best model for **classification, regression, and clustering**, based on evaluation metrics. |
| classification\_models/ | Contains trained models used for **predicting customer conversion (classification task)**. |
| ├── DecisionTreeClassifier.pkl | A trained **Decision Tree Classifier** model. |
| ├── GaussianNB.pkl | A trained **Naïve Bayes Classifier** model. |
| ├── GradientBoostingClassifier.pkl | A trained **Gradient Boosting Classifier** model. |
| ├── KNeighborsClassifier.pkl | A trained **K-Nearest Neighbors (KNN)** model. |
| ├── LogisticRegression.pkl | A trained **Logistic Regression** model. |
| ├── MLPClassifier.pkl | A trained **Neural Network (MLP) Classifier** model. |
| ├── RandomForestClassifier.pkl | A trained **Random Forest Classifier** model. |
| ├── SVC.pkl | A trained **Support Vector Machine (SVM) Classifier** model. |
| ├── XGBoostClassifier.pkl | A trained **XGBoost Classifier** model. |
| clustering\_models/ | Contains trained models for **customer segmentation (clustering task)**. |
| ├── AgglomerativeClustering.pkl | A trained **Agglomerative Clustering** model. |
| ├── DBSCAN.pkl | A trained **DBSCAN (Density-Based Spatial Clustering)** model. |
| ├── GaussianMixture.pkl | A trained **Gaussian Mixture Model (GMM)**. |
| ├── KMeans.pkl | A trained **K-Means Clustering** model. |
| regression\_models/ | Contains trained models for **predicting potential revenue (regression task)**. |
| ├── DecisionTreeRegressor.pkl | A trained **Decision Tree Regressor** model. |
| ├── ElasticNet.pkl | A trained **ElasticNet Regression** model. |
| ├── GradientBoostingRegressor.pkl | A trained **Gradient Boosting Regressor** model. |
| ├── LassoRegression.pkl | A trained **Lasso Regression** model. |
| ├── LinearRegression.pkl | A trained **Linear Regression** model. |
| ├── MLPRegressor.pkl | A trained **Neural Network (MLP) Regressor** model. |
| ├── PolynomialRegression.pkl | A trained **Polynomial Regression** model. |
| ├── RandomForestRegressor.pkl | A trained **Random Forest Regressor** model, identified as the **best regression model** in best\_models\_file.csv. |
| ├── RidgeRegression.pkl | A trained **Ridge Regression** model. |
| ├── SVR.pkl | A trained **Support Vector Regression (SVR)** model. |
| ├── XGBoostRegressor.pkl | A trained **XGBoost Regressor** model. |

**Folder: pipelines**

The pipelines folder contains **ZenML pipeline scripts** for automating the training, evaluation, and selection of models for **classification, regression, and clustering tasks**. Each pipeline is structured to handle **data ingestion, preprocessing, feature selection, model training, evaluation, and best model selection**.

| **File Name** | **Purpose** |
| --- | --- |
| classification\_pipeline.py | Defines the pipeline for **customer conversion prediction (classification task)**. It preprocesses data, applies Recursive Feature Elimination (RFE), trains multiple classification models, evaluates them, and selects the best-performing model. |
| clustering\_pipeline.py | Defines the pipeline for **customer segmentation (clustering task)**. It processes the dataset, selects relevant features, applies scaling, trains clustering models, evaluates them, and identifies the best segmentation model. |
| regression\_pipeline.py | Defines the pipeline for **revenue estimation (regression task)**. It preprocesses the dataset, selects relevant features using RFE, trains regression models, evaluates them, and picks the best-performing model. |
| \_\_pycache\_\_/ | Stores compiled Python files (.pyc) for optimized execution of the pipeline scripts. |

**Pipeline Execution Flow**

Each pipeline follows this structured workflow:

1. **Data Ingestion** – Loads training and test datasets.
2. **Data Preprocessing** – Handles missing values, encodes categorical data, and scales numerical features.
3. **Feature Engineering & Selection** – Creates new meaningful features and applies **Recursive Feature Elimination (RFE)** for optimal feature selection.
4. **Model Training** – Trains multiple models using various machine learning algorithms.
5. **Model Evaluation** – Assesses model performance using appropriate metrics.
6. **Best Model Selection** – Identifies the best-performing model and logs metrics using **MLflow**.
7. **Final Model Output** – Returns the best model for deployment.

**Folder: requirements doc**

The requirements doc folder contains documentation related to the **Clickstream Customer Conversion Analysis** project. It includes essential details about project requirements, objectives, methodology, and implementation.

| **File Name** | **Purpose** |
| --- | --- |
| Clickstream-customer conversion.docx | A Microsoft Word document outlining the project scope, requirements, and detailed technical specifications. |
| Clickstream-customer conversion.pdf | A PDF version of the requirements document for easy sharing and reference. |

**Folder: src**

The src folder contains core functions for **data preprocessing, feature selection, model training, evaluation, and best model selection**. These scripts are used by the ZenML Steps.

| **File Name** | **Purpose** |
| --- | --- |
| data\_feature\_engineering.py | Creates new features from existing data, such as session-based and seasonal indicators, to enhance model performance. |
| data\_feature\_selection.py | Selects the most relevant features using **Chi-Square, ANOVA, Pearson Correlation, and Variance Threshold** methods. |
| data\_loading.py | Handles **loading of training and testing datasets** from CSV files for processing. |
| data\_preprocessing.py | Cleans data by **removing duplicates, handling missing values, encoding categorical variables, and detecting outliers**. |
| find\_best\_model.py | Identifies and logs the **best-performing model** based on evaluation metrics and stores it in models/best\_models/. |
| model\_evaluation.py | Evaluates trained models using **classification (Accuracy, Precision, Recall), regression (MSE, RMSE, R² Score), and clustering (Silhouette Score, Davies-Bouldin Index) metrics**. |
| model\_training.py | Trains multiple models for **classification, regression, and clustering**, using algorithms like Random Forest, XGBoost, Decision Trees, and Neural Networks. |

**Folder: steps**

The steps folder contains **ZenML step functions** used in the pipeline for data processing, feature engineering, model training, evaluation, and selection. Each step executes a specific task within the **classification, regression, and clustering workflows**.

| **File Name** | **Purpose** |
| --- | --- |
| best\_model.py | Selects the **best-performing model** for classification, regression, and clustering using evaluation metrics. |
| clean\_data\_steps.py | Cleans data by **handling missing values, removing duplicates, and checking for outliers**. Encodes categorical features and stores preprocessed data. |
| clean\_data.py | Integrates data cleaning, encoding, and outlier detection to provide a **fully preprocessed dataset** for model training. |
| evaluate\_model.py | Evaluates models using task-specific metrics: **Accuracy, Precision, Recall, F1-score (classification)**, **MSE, RMSE, R² (regression)**, and **Silhouette Score, Davies-Bouldin Index (clustering)**. |
| feature\_engineering.py | Generates new **session-based, time-based, and purchase-related** features to improve model performance. |
| feature\_selection.py | Selects the most relevant features using **Chi-Square, ANOVA, Pearson Correlation, and Variance Threshold** techniques for different ML tasks. |
| ingest\_data.py | Loads **training and test datasets** from CSV files for processing. |
| rfe\_feature\_selection.py | Uses **Recursive Feature Elimination (RFE)** to optimize model input features, reducing dimensionality while retaining important predictors. |
| train\_model.py | Trains models for **classification, regression, and clustering** using various ML algorithms (Decision Trees, Random Forest, XGBoost, SVM, etc.). |
| transforming\_data.py | Applies **Standard Scaling** for numerical features and **SMOTE/Undersampling** to handle class imbalance in training data. |

**Step Execution in Pipelines**

1. **Data Ingestion** → ingest\_data.py
2. **Data Preprocessing** → clean\_data\_steps.py, clean\_data.py
3. **Feature Engineering & Selection** → feature\_engineering.py, feature\_selection.py, rfe\_feature\_selection.py
4. **Model Training** → train\_model.py
5. **Model Evaluation** → evaluate\_model.py
6. **Best Model Selection** → best\_model.py

**Folder: support**

The support folder contains **pre-processed mappings, encoded values, and scaling models** that support classification, regression, and clustering pipelines. These files are essential for ensuring consistency in data transformations.

| **File/Folder Name** | **Purpose** |
| --- | --- |
| add\_ons/ | Stores individual mappings for categorical variables used in  classification, regression, and clustering models. |
| ├── classification\_page2\_clothing\_model\_mapping.pkl | Mapping of clothing models for the classification pipeline. |
| ├── classification\_purchase\_completed\_mapping.pkl | Mapping of the purchase\_completed feature for classification. |
| ├── classification\_season\_mapping.pkl | Mapping of the season feature for classification. |
| ├── clustering\_page2\_clothing\_model\_mapping.pkl | Mapping of clothing models for the clustering pipeline. |
| ├── clustering\_purchase\_completed\_mapping.pkl | Mapping of the purchase\_completed feature for clustering. |
| ├── clustering\_season\_mapping.pkl | Mapping of the season feature for clustering. |
| ├── regression\_page2\_clothing\_model\_mapping.pkl | Mapping of clothing models for the regression pipeline. |
| ├── regression\_purchase\_completed\_mapping.pkl | Mapping of the purchase\_completed feature for regression. |
| ├── regression\_season\_mapping.pkl | Mapping of the season feature for regression. |
| classification\_encoded\_mappings.pkl | Stores all **encoded categorical mappings** for the classification pipeline. |
| clustering\_encoded\_mappings.pkl | Stores all **encoded categorical mappings** for the clustering pipeline. |
| regression\_encoded\_mappings.pkl | Stores all **encoded categorical mappings** for the regression pipeline. |
| classification\_standard\_scaler.pkl | Pre-trained **Standard Scaler** for numerical features in classification. |
| clustering\_standard\_scaler.pkl | Pre-trained **Standard Scaler** for numerical features in clustering. |
| regression\_standard\_scaler.pkl | Pre-trained **Standard Scaler** for numerical features in regression. |

**Role of the support Folder in Pipelines**

* **Encodes categorical features** (e.g., season, clothing model, purchase status) to maintain consistency.
* **Applies Standard Scaling** to normalize numerical features for better model performance.
* **Ensures mapping consistency** across training and inference phases.

**Folder: utils**

The utils folder contains utility functions essential for data processing, feature engineering, model training, and evaluation. These functions serve as helper modules to streamline pipeline execution.

| **File Name** | **Purpose** |
| --- | --- |
| data\_transformation.py | Handles **standard scaling** and **data balancing** using **SMOTE** and **undersampling** to ensure model consistency. |
| encoding\_values.py | Stores **predefined categorical mappings** and manages encoding processes for categorical variables. |
| get\_result.py | Retrieves the **best model outputs** and evaluation metrics from the latest ZenML pipeline run. |
| helper\_func.py | Provides **model training functions**, allowing training and saving of regression, classification, and clustering models with **MLflow tracking**. |
| rfe\_selection.py | Implements **Recursive Feature Elimination (RFE)** to select the most relevant features for **classification and regression** tasks. |

**Role of the utils Folder in the Project**

* **Data Preprocessing**: Handles **scaling, encoding, and balancing** for effective training.
* **Feature Selection**: Implements **RFE** for identifying the most impactful features.
* **Model Training & Evaluation**: Streamlines **model training**, **tracking**, and **retrieval** of the best models.
* **Pipeline Optimization**: Supports **automated selection and retrieval** of optimal configurations.

**File: main\_pipeline.py**

The main\_pipeline.py script is responsible for orchestrating the **classification, regression, and clustering pipelines** using **ZenML** and **MLflow tracking**.

| **File Name** | **Purpose** |
| --- | --- |
| main\_pipeline.py | Executes **classification, regression, and clustering pipelines** sequentially, tracks experiments using **MLflow**, and logs execution time. |

**Key Functionalities of main\_pipeline.py**

1. **Pipeline Execution**:
   * Runs **classification, regression, and clustering pipelines** in sequence.
   * Ensures seamless integration of all ML tasks.
2. **MLflow Tracking**:
   * Sets up **MLflow experiments** for tracking **model training and evaluation**.
   * Logs execution details for **experiment reproducibility**.
3. **Error Handling & Logging**:
   * Catches errors during execution and logs full **exception traceback** for debugging.
   * Ensures robust execution flow with structured exception handling.
4. **Execution Time Logging**:
   * Calculates and logs **pipeline execution time** for performance monitoring.
5. **Standalone Execution**:
   * When executed as \_\_main\_\_, it runs the pipeline using **predefined training and test data files**.

**Role of main\_pipeline.py in the Project**

* **Automates the execution of all ML pipelines** in a single script.
* **Facilitates experiment tracking** with **MLflow**.
* **Ensures smooth integration** between different machine learning tasks.

**File: main.py**

The main.py script serves as the **interactive web interface** for the Clickstream Customer Conversion Analysis. It is built using **Streamlit** and allows users to interact with machine learning models for **classification, regression, and clustering** predictions.

| **File Name** | **Purpose** |
| --- | --- |
| main.py | Streamlit-based web application for customer conversion analysis using **classification, regression, and clustering** models. |

**Key Functionalities of main.py**

1. **Loads Best Performing Models Dynamically**
   * Reads the **best models** from models/best\_models/best\_models\_file.csv.
   * Loads the corresponding .pkl model files for **classification, regression, and clustering**.
2. **Loads Encoding Mappings**
   * Loads categorical **label encoders** for classification and regression from support/.
   * Ensures categorical data is **correctly transformed** before making predictions.
3. **Streamlit UI with Interactive Input Modes**
   * **Manual Input Mode**: Users can enter **browsing session details** to predict **purchase conversion** or **revenue estimation**.
   * **CSV Upload Mode**: Users can upload a dataset for **bulk predictions** and view results in a table/graph.
4. **Machine Learning Predictions**
   * **Classification**: Predicts whether a customer will **purchase or not**.
   * **Regression**: Estimates **expected revenue** from customer behaviour.
   * **Clustering**: Groups customers into **behavioural segments** using unsupervised learning.
5. **Visualization & Reporting**
   * Displays **real-time predictions** for individual inputs.
   * Supports **bulk CSV processing** and generates **scatter plots** for clustering results.

**Role of main.py in the Project**

* **Acts as the user interface** for the **customer conversion analysis system**.
* **Integrates ML models** dynamically without requiring manual intervention.
* **Facilitates real-time and batch predictions**, making it suitable for both **individual** and **business-level analysis**.
* **Provides a seamless experience** with intuitive UI components and **automated feature encoding**.

**File: flow.py**

This file contains the folder structure of the **Clickstream Customer Conversion** project. It provides an organized overview of directories and files, categorizing datasets, source code, pipelines, models, utilities, and documentation to ensure clarity in project workflow and navigation.

**File: requirements.txt**

This file lists the essential Python libraries required to run the Clickstream Customer Conversion project. These dependencies enable efficient data processing, machine learning model training, and deployment.

* **pandas** - Data manipulation and analysis library.
* **numpy** - Supports numerical computations and array operations.
* **scikit-learn** - Provides machine learning algorithms for classification, regression, and clustering.
* **zenml** - A pipeline framework for reproducible machine learning workflows.
* **zenml[server]** - Enables ZenML server-based tracking and pipeline management.
* **mlflow** - Manages and tracks machine learning experiments and models.
* **zenml[mlflow]** - Integrates MLflow with ZenML for streamlined tracking.
* **imbalanced-learn** - Provides techniques to handle imbalanced datasets, such as SMOTE.
* **xgboost** - Optimized gradient boosting library for high-performance modeling.
* **scipy** - Supports scientific computations and statistical functions.
* **streamlit** - A framework for deploying interactive machine learning applications.

These libraries collectively facilitate data preprocessing, feature engineering, model training, evaluation, and visualization within the project.

**File: docker-compose.yml**

This file defines the containerized environment for the **Clickstream Customer Conversion** project, ensuring seamless integration of MLflow, pipeline execution, and the Streamlit UI.

* **MLflow Service (mlflow)**
  + Runs the MLflow tracking server on port **5000**.
  + Stores experiment logs and artifacts in **mlruns/**.
* **Main Pipeline Service (mainpipeline)**
  + Executes the machine learning pipeline using ZenML.
  + Connects to MLflow for tracking experiments.
  + Utilizes a predefined **ZenML stack (mlflow\_stack)**.
* **Streamlit UI Service (streamlit)**
  + Deploys the interactive web application on port **8501**.
  + Loads models from **models/** and data from **data/**.
  + Depends on the **main pipeline** for inference.

This configuration ensures a modular, scalable, and reproducible setup for running machine learning workflows in a **containerized environment**.

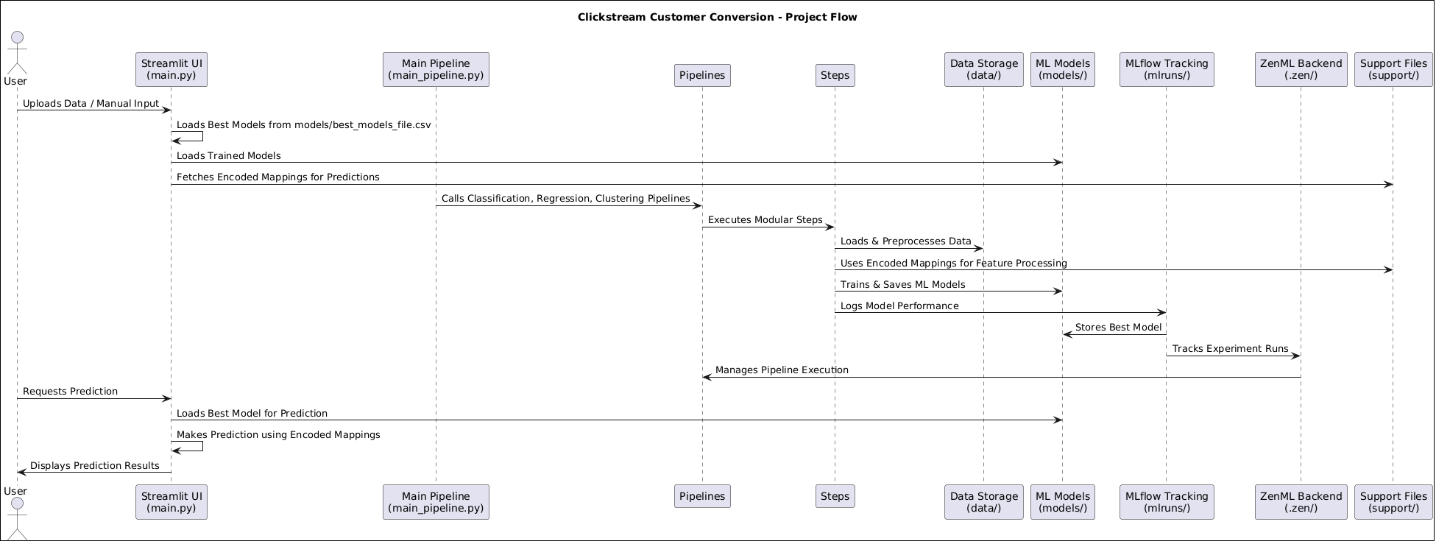
**File: entrypoint.sh**

The **entrypoint.sh** script serves as the entry point for the containerized execution of the project. It ensures the proper initialization and execution of services within the **Dockerized environment**.

* **Configures Environment Variables:** Sets up necessary configurations such as MLflow tracking URI and ZenML stack.
* **Initializes ZenML:** Ensures that the ZenML stack and configurations are correctly initialized.
* **Executes the Main Pipeline:** Triggers the machine learning pipeline, handling data ingestion, preprocessing, model training, and evaluation.
* **Ensures Container Readiness:** Ensures that all dependencies are met before starting the application.

This script is critical for automating workflow execution within **Docker containers**, ensuring a smooth and reproducible deployment process.

**Project Flow:**

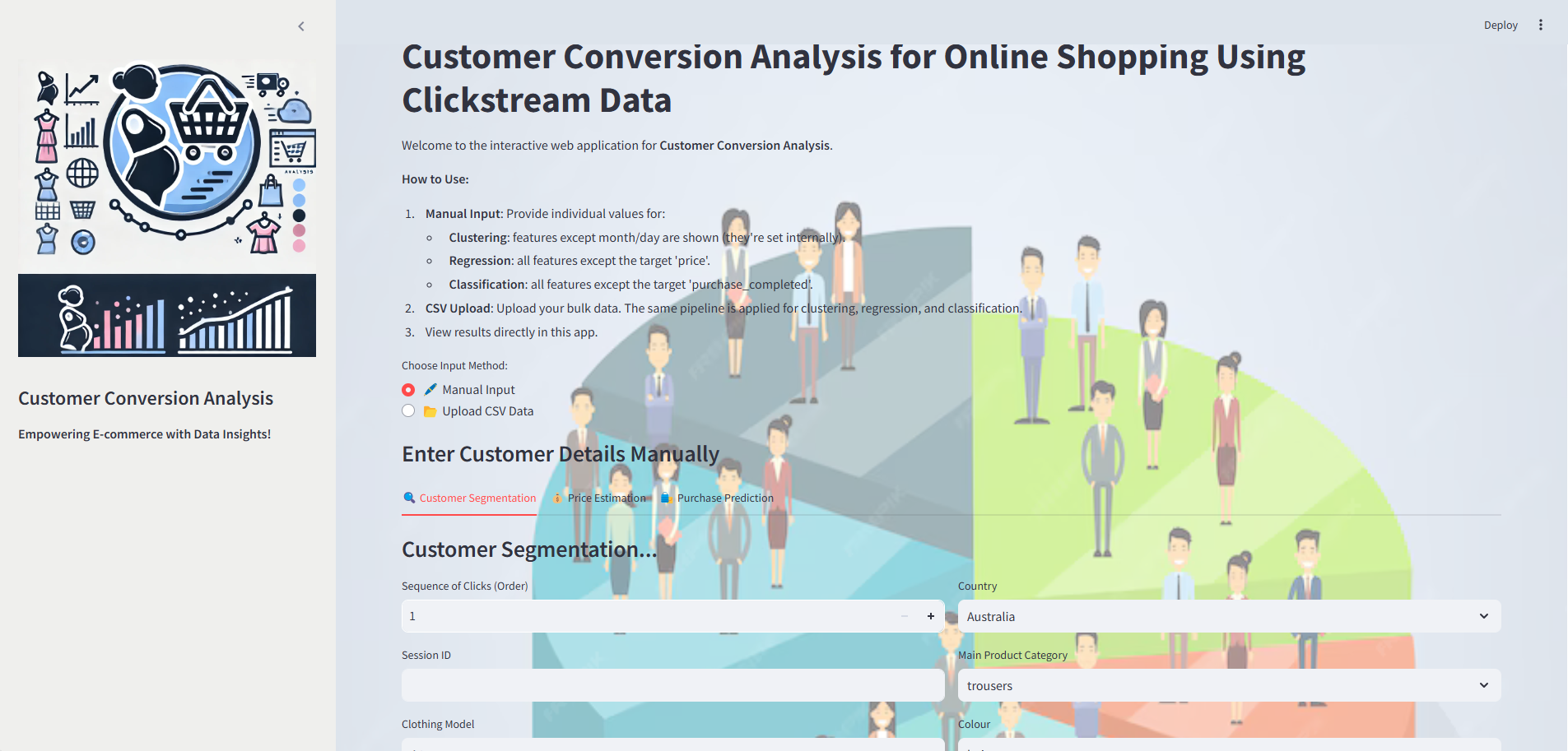


**Project Folder Structure:**

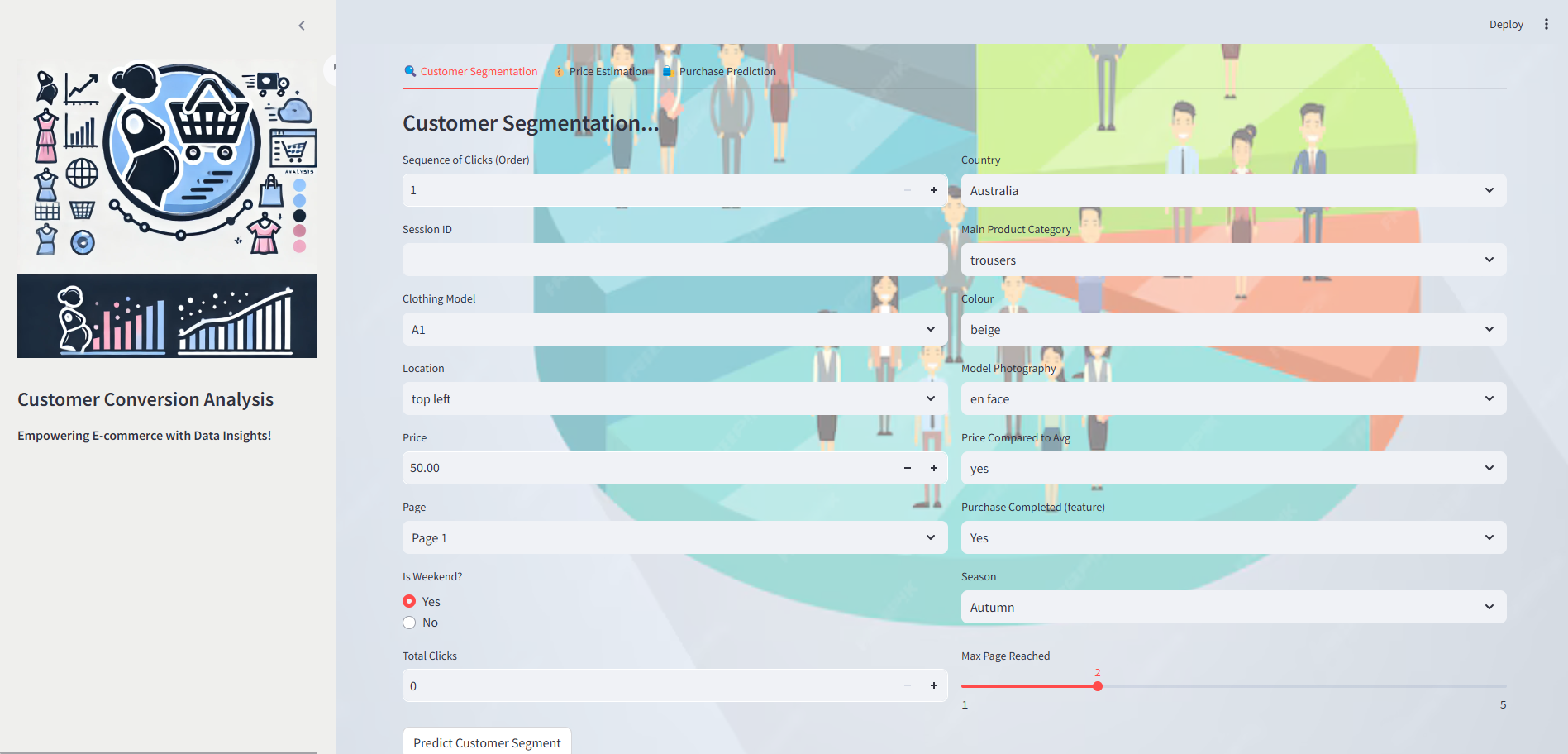
To know more about the project’s folder structure, refer “flow.py” file

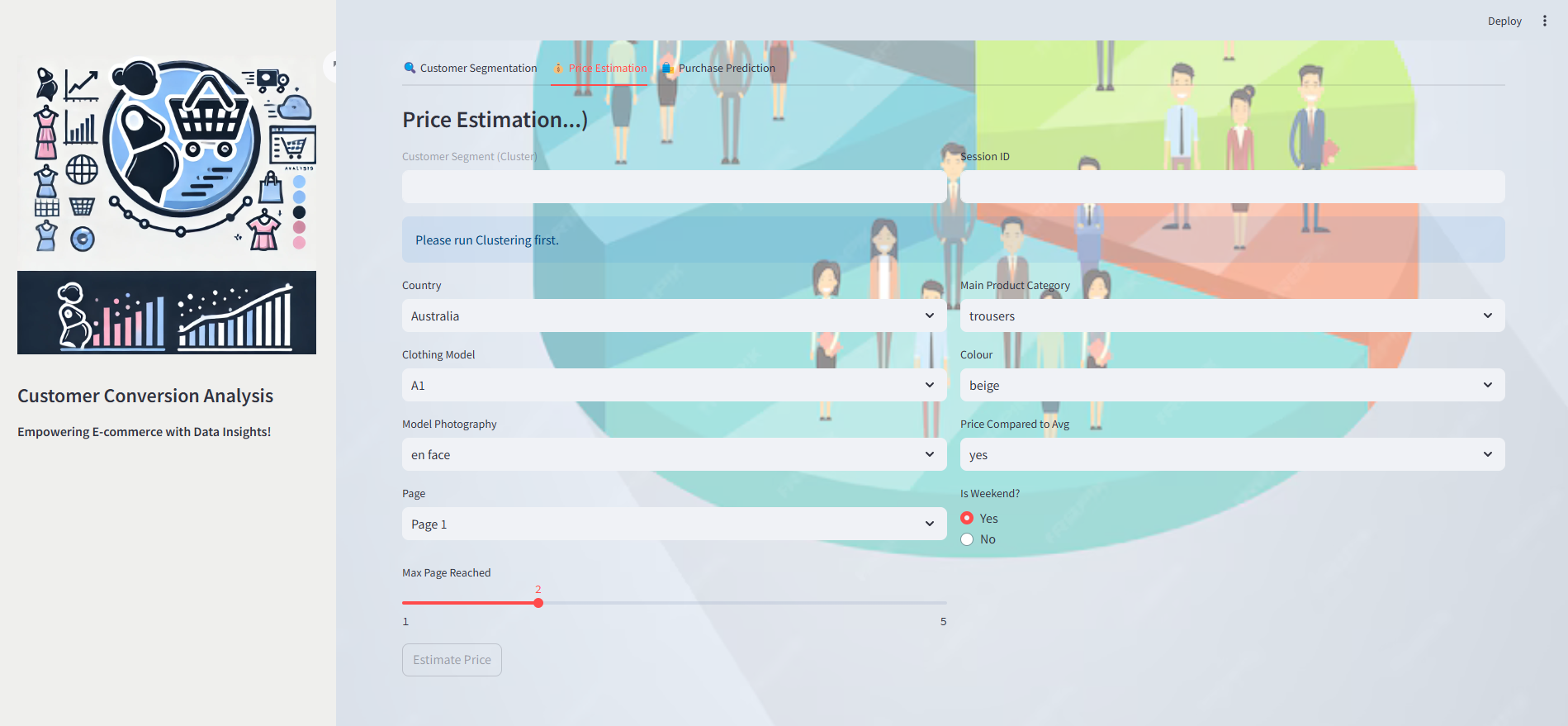
**Streamlit UI:**

***User Interface***



***Manual Input Tabs***







***Upload CSV Data Tabs***

