MLOPS PROJECT REPORT

CUSTOMER CHURN PREDICTION

Group Members : Abhishek Sahoo,Ayush Chakroborty,Prajwal Wagh,Vikram Pathak,Yash Joglekar

**Introduction:**

Customer churn is a critical issue for businesses, as retaining existing customers is more cost-effective than acquiring new ones. This project focuses on building an **end-to-end MLOps pipeline** to predict customer churn with high accuracy. The pipeline ensures **automated data processing, model training, deployment, and continuous monitoring**, following industry-standard **MLOps best practices**.

Our approach integrates **machine learning, software engineering, and DevOps** to build a scalable and reliable system. The project leverages tools like **MLflow, FastAPI, Streamlit, and Alibi-Detect** to create an efficient and maintainable pipeline.

This report summarizes the entire pipeline, from **data ingestion to monitoring**, with key artifacts, snapshots, and insights.

**Objective of the Project**

The primary goal of this project is to **develop a robust and automated MLOps pipeline** that:  
✅ **Accurately predicts customer churn** using historical data.  
✅ **Automates the ML lifecycle** – from data preparation to model deployment.  
✅ **Tracks experiments and models** for reproducibility using MLflow.  
✅ **Deploys the model as an API** using FastAPI for real-time predictions.  
✅ **Develops a user-friendly interface** via Streamlit to interact with predictions.  
✅ **Monitors data drift** in production to ensure model stability.

**Key Components & Approach**

**1.Dataset Preparation & Storage**

Defined **dataset schema** (numerical/categorical columns).  
Stored in **Parquet format** for efficient processing.  
Split into **training (60%), testing (20%), and production (20%) datasets**.

**2.ML Pipeline & Experiment Tracking**

Built a **data preprocessing pipeline** (scaling, encoding, imputation).  
Ran multiple **ML experiments** (Random Forest, Logistic Regression, Decision Trees).  
Used **MLflow** to log models, hyperparameters, and evaluation metrics.

**3.Model Deployment as API (FastAPI)**

The **best model** was selected and deployed using **FastAPI**.  
Exposed a **RESTful API** for real-time predictions.

**4.User Interface (Streamlit)**

Built a **web UI** where users can **input data** and **receive predictions**.

**5.Model Monitoring & Drift Detection**

Implemented **data drift analysis** using **Alibi-Detect**.  
Monitored **statistical changes in production data** to trigger retraining if needed.

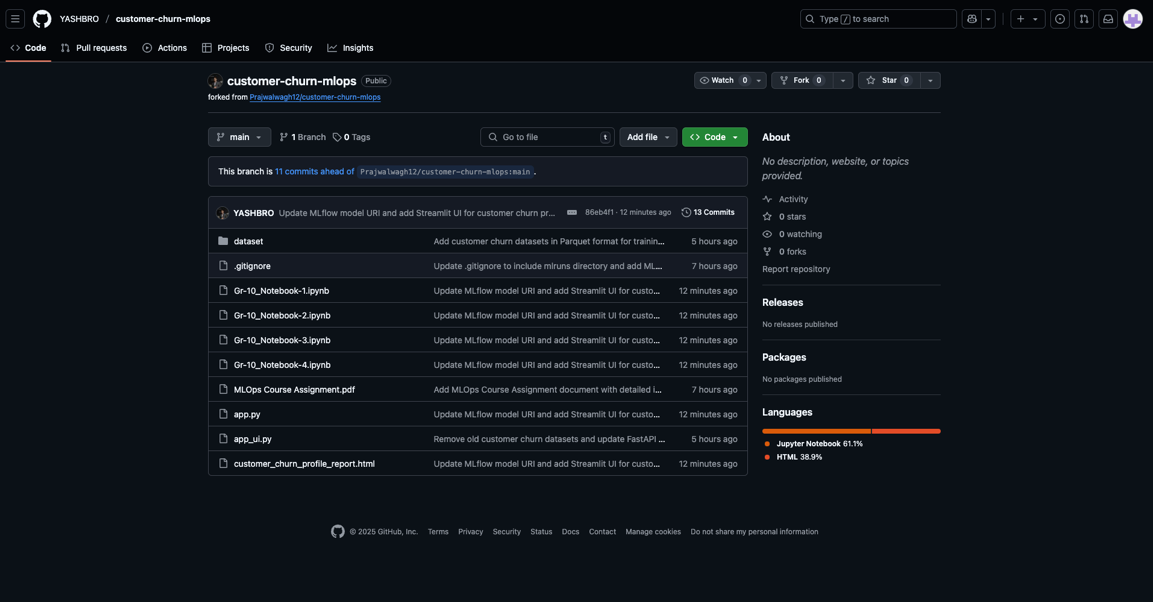
**Technology Stack Used**

**Data Handling:** Pandas, Parquet  
**ML Models:** Scikit-learn (Random Forest, Logistic Regression, Decision Trees)  
**MLOps Tools:** MLflow (Experiment Tracking)  
**Deployment:** FastAPI (Model Serving as API)  
**UI Development:** Streamlit (User Interface)  
**Monitoring:** Alibi-Detect (Data Drift Detection)

**Github Link :** <https://github.com/YASHBRO/customer-churn-mlops>

Screenshots:

1.Github Repo

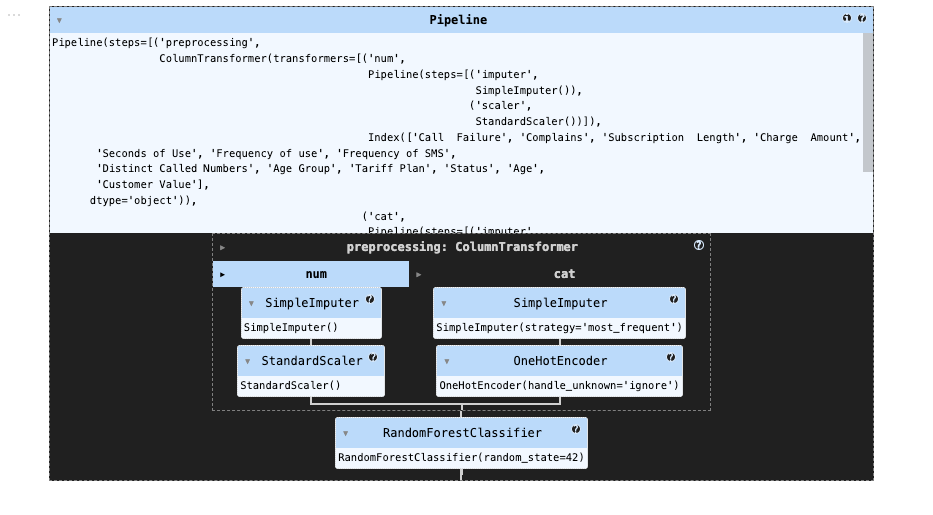


2.Mlflow

A screenshot of a computer

Description automatically generated

3.Pipeline



**Results & UI Predictions**

Best Model: RandomForestClassifier  
Test Accuracy: 94.29%  
Cross-validation Accuracy: 93.75%  
Comparison with other models: The Random Forest model outperformed all others, showing the highest accuracy

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**DATA DRIFT ANALYSIS & MODEL MONITORING**

**Objective:**  
Monitoring model performance over time is crucial to ensure its accuracy in a real-world production environment. This project implemented **data drift detection using Alibi-Detect** to analyze whether the production data distribution significantly deviates from the training data. If significant drift is detected, it may indicate the need for model retraining.

**Step 1: Load Training and Production Data** – The training dataset serves as the baseline, while the production dataset is collected from real-world usage post-deployment.

**Step 2: Apply Drift Detection Algorithm** – The **Kolmogorov-Smirnov (KS) Test** compares feature distributions, with a **p-value threshold of 0.05** to detect significant drift.

**Step 3: Interpret the Results** – **If p-value < 0.05**, drift is detected; otherwise, the feature distribution remains stable.

**Data Drift Table :**

**A screenshot of a computer

Description automatically generated**

**Conclusion & Lessons Learned**

Key Takeaways from the Project

Building an MLOps pipeline ensures automation and efficiency.  
Experiment tracking with MLflow helped in selecting the best model.  
Model deployment with FastAPI made real-time predictions accessible via API.  
Monitoring data drift is crucial to maintain model reliability over time.

Challenges Faced & How We Overcame Them

* Challenge: Data drift detection required handling missing values.  
  Solution: Implemented imputation techniques to clean data before drift analysis.
* Challenge: Model versioning required consistent tracking.  
  Solution: Used MLflow to log model artifacts, parameters, and performance metrics.
* Challenge: Ensuring API and UI work seamlessly.  
  Solution: Used Postman for API testing and Streamlit for an interactive UI.[

**Future Scope :**

**Automate Model Retraining** – Set up an **automated pipeline** to trigger retraining when drift exceeds a threshold, using **CI/CD** for seamless deployment.

**Improve Feature Engineering** – Conduct **feature importance analysis** and experiment with **new transformations** to enhance predictive power.

**Scale API Deployment** – Deploy FastAPI with **Docker & Kubernetes**, implementing **load balancing** for real-time request handling.