**End-to-End Machine Learning Pipeline Documentation**

**1. Pipeline Design**

The customer churn prediction pipeline is designed to automate the entire data processing, model training, and deployment workflow. The pipeline follows a modular structure using Apache Airflow for orchestration, DVC for dataset versioning, and MLflow for model tracking.

**Pipeline Components:**

1. **Data Ingestion:**
   * Retrieves data from multiple sources: local CSV and Kaggle API.
   * Stores raw data in data\_lake/raw/.
   * Logs ingestion status for monitoring.
2. **Data Validation:**
   * Checks for missing values, incorrect data types, and outliers.
   * Generates a data quality report.
   * Stores validated data in data\_lake/validated/.
3. **Data Preparation & Feature Engineering:**
   * Handles missing values and encodes categorical variables.
   * Standardizes numerical features using StandardScaler.
   * Stores transformed datasets in data\_lake/processed/.
4. **Feature Store & Data Versioning:**
   * Stores and manages engineered features using Feast.
   * Versions raw and processed datasets using DVC.
   * Ensures reproducibility across pipeline runs.
5. **Model Training & Evaluation:**
   * Trains multiple models (Random Forest, Logistic Regression) using scikit-learn.
   * Evaluates models using accuracy, precision, recall, and F1 score.
   * Saves trained models using MLflow.
6. **Model Deployment:**
   * Serves the trained model using a Flask API.
   * Provides an endpoint for real-time churn predictions.
   * Enables batch inference on new customer data.
7. **Pipeline Orchestration:**
   * Uses Apache Airflow DAG to automate data flow.
   * Defines task dependencies for end-to-end execution.
   * Logs pipeline execution for monitoring.

**2. Challenges & Solutions**

**1. Large Dataset Processing in Google Colab**

* **Challenge:** Google Colab's memory limitations caused crashes during data preprocessing.
* **Solution:**
  + Implemented **chunk-based reading** to load data in smaller parts.
  + Reduced sample size for visualization (2000 rows instead of full dataset).
  + Used **Dask/Pandas optimizations** for handling large files efficiently.

**2. Handling Data Format Differences (Local vs. Kaggle Dataset)**

* **Challenge:** Different column names and categorical encodings between datasets.
* **Solution:**
  + Created **dataset-specific preprocessing** to align features.
  + Used dynamic **column mapping** to handle schema variations.
  + Standardized categorical encodings across both datasets.

**3. Outliers and Data Quality Issues**

* **Challenge:** Some features contained extreme values (e.g., age > 1000).
* **Solution:**
  + Applied **Z-score-based outlier detection**.
  + Used median imputation for replacing extreme values.
  + Logged detected anomalies for manual review.

**4. Model Performance & Overfitting**

* **Challenge:** Initial models showed overfitting on training data.
* **Solution:**
  + Used **cross-validation** and **hyperparameter tuning**.
  + Implemented feature selection to reduce complexity.
  + Evaluated multiple models to choose the best performer.

**5. Deployment & API Integration Issues**

* **Challenge:** Integrating the trained model into an API with real-time inference.
* **Solution:**
  + Used **Flask for API deployment** with proper request handling.
  + Provided **sample requests** for testing API functionality.
  + Implemented logging and monitoring for model predictions.

**3. Conclusion**

The end-to-end pipeline successfully automates customer churn prediction while ensuring data quality, scalability, and reproducibility. The implemented solutions address key challenges, making the pipeline robust and deployment-ready.

**Next Steps:**

* Extend support for **real-time data streaming**.
* Implement **AutoML for hyperparameter tuning**.
* Deploy the model using a **cloud-based service (AWS/GCP/Azure)**.

**Deliverables:**

📌 **Pipeline documentation (this document)** 📌 **Execution logs and monitoring reports** 📌 **Final trained model & API deployment guide**