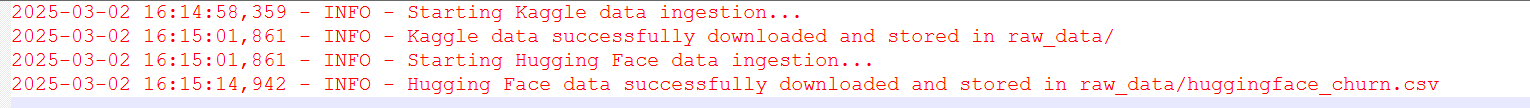
# A screenshot of a computer AI-generated content may be incorrect.

Data Ingestion:



A screenshot of a computer

AI-generated content may be incorrect.

Raw Data Storage:

# Detailed Documentation: End-to-End Data Management Pipeline for Telco Customer Churn Prediction

## Overview

This document outlines the complete data management pipeline implemented for predicting customer churn using the Telco dataset. The pipeline spans from problem formulation to orchestration, ensuring data quality, reproducibility, and model reliability.

## Pipeline Architecture

The pipeline consists of the following 9 modular stages:

### 1. Problem Formulation

Defined the business problem of customer churn and its impact on revenue. Identified objectives such as churn prediction and proactive retention. Used Telco datasets from Kaggle and Hugging Face. Expected outputs include clean datasets, transformed features, and a deployable model.

### 2. Data Ingestion

Automated ingestion scripts were created for Kaggle and Hugging Face datasets. Included logging and error handling. Data was saved in raw format with source-based partitioning.

### 3. Raw Data Storage

Raw data stored in a local data lake structure. Partitioned by source (kaggle/huggingface), type (csv), and timestamp.

### 4. Data Validation

Validation checks implemented using pandas. Checked for missing values, data types, negative values, and duplicates. Generated a CSV report summarizing issues.

### 5. Data Preparation

Handled missing values by dropping rows. Standardized numerical features using StandardScaler. Encoded categorical variables using one-hot encoding. Performed EDA with histograms and box plots.

### 6. Data Transformation and Storage

Engineered features like TotalSpend and TenureCategory. Scaled features using MinMaxScaler. Stored transformed data in SQLite database with schema and sample queries.

### 7. Feature Store

Implemented a custom feature store using SQLite. Defined metadata for each feature in JSON format. Enabled feature retrieval via SQL queries.

### 8. Data Versioning

Used custom versioning strategy with timestamped folders and metadata files. Tracked changes in raw and transformed datasets.

### 9. Model Building

Trained Logistic Regression and Random Forest models using scikit-learn. Evaluated using accuracy, precision, recall, and F1-score. Saved best model (Random Forest) as .pkl file and generated performance report.

### 10. Pipeline Orchestration

Defined DAG using Apache Airflow with PythonOperator tasks. Automated task dependencies from ingestion to model training. Included logging and monitoring setup.

## Challenges Faced and Solutions Implemented

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Solution Implemented |
| Missing Values | Nulls and empty strings in TotalCharges column | Handled using pandas dropna and type conversion |
| Data Type Issues | TotalCharges column was object type due to empty strings | Converted to numeric using pandas with error coercion |
| Duplicate Records | Potential duplicate rows in combined dataset | Checked and removed using pandas duplicated() |
| Categorical Encoding | High cardinality in categorical columns | Used one-hot encoding via pandas get\_dummies |
| Feature Scaling | Numerical features had different scales | Applied StandardScaler and MinMaxScaler from sklearn |
| Feature Engineering | Needed derived features like TotalSpend and TenureCategory | Created using pandas transformations |
| Feature Store | Required centralized feature management | Implemented using SQLite with metadata JSON |
| Versioning | Tracking changes in datasets and features | Used custom versioning with timestamped folders and metadata files |
| Model Training | Selecting best model for churn prediction | Trained Logistic Regression and Random Forest using sklearn |
| Pipeline Orchestration | Automating task dependencies | Defined DAG using Apache Airflow with PythonOperator |