A screenshot of a computer

Description automatically generated

**Step1 (Directory 2):**

To download two datasets from 2 website using API

Example 1) Kaggle 2) Hugging Face

2025-03-02 16:14:58,359 - INFO - Starting Kaggle data ingestion...

2025-03-02 16:15:01,861 - INFO - Kaggle data successfully downloaded and stored in raw\_data/

2025-03-02 16:15:01,861 - INFO - Starting Hugging Face data ingestion...

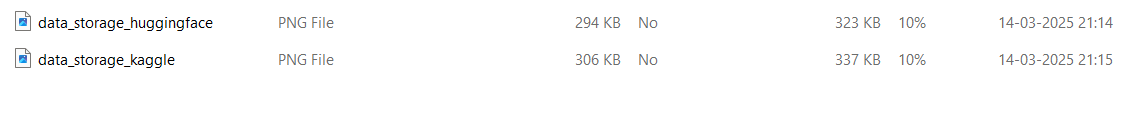
2025-03-02 16:15:14,942 - INFO - Hugging Face data successfully downloaded and stored in raw\_data/huggingface\_churn.csv

**Step 2 : (Directory 3):**

Storing the downloaded data into your local machine

A screenshot of a computer

Description automatically generated



**Step 3 : (Directory 4):**

**Data Validation.**

**Data validation** is the process of ensuring that data is **accurate, clean, and useful** before it is processed or stored. It checks that the data entered into a system meets certain **rules or constraints**, which are defined based on the type of data and the business logic involved

An excel file indicating what are the missing, duplicate, data\_types and negative values

A screenshot of a computer

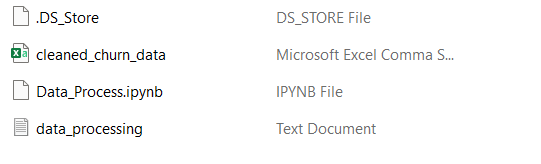
Description automatically generated

**Step 4 : (Directory 5):**

**Data preparation** is the process of collecting, cleaning, organizing, and transforming raw data into a format that is ready for analysis, reporting, or machine learning.

It’s a critical step in the **data lifecycle** because high-quality decisions and models require high-quality data.

In this case a single code for both Kaggle and hugging face.



**Step 5 : (Directory 6)**

**Data transformation** is the process of converting data from its original format into a new structure or format that is more appropriate for analysis, reporting, or machine learning.

Example

\"\"\"Perform feature engineering and transformations.\"\"\"\n",

### ✨ Common Data Transformation Tasks

| **Task** | **Description** | **Example** |
| --- | --- | --- |
| **Normalization** | Scaling numeric data to a standard range | Scale income to a 0–1 range |
| **Standardization** | Shifting and scaling data to have mean = 0, std dev = 1 | Standardize test scores |
| **Encoding** | Converting categorical data into numerical form | "Yes"/"No" → 1/0 |
| **Aggregation** | Summarizing data | Total sales per month |
| **Pivoting/Unpivoting** | Restructuring data tables | Rows to columns and vice versa |
| **Filtering** | Removing irrelevant data | Only keep rows where status = "active" |
| **Date-Time Conversion** | Changing date formats or extracting components | "2025-08-18" → year: 2025 |

**Step 6 : (Directory 7)**

A **Feature Store** is a centralized system or platform used to **store, manage, and serve features** for machine learning models—both during training and in production.

Sample Code

"# Create a table to store engineered features\n",

"cursor.execute('''\n",

" CREATE TABLE IF NOT EXISTS feature\_store (\n",

" customerID TEXT PRIMARY KEY,\n",

" tenure INTEGER,\n",

" MonthlyCharges REAL,\n",

" TotalCharges REAL,\n",

" Contract\_OneYear INTEGER,\n",

" Contract\_TwoYear INTEGER,\n",

" PaymentMethod\_CreditCard INTEGER,\n",

" PaymentMethod\_ElectronicCheck INTEGER,\n",

" PaymentMethod\_MailedCheck INTEGER,\n",

" Churn INTEGER\n",

**Step 7 : (Directory 8)**

Data **versioning** refers to the process of tracking, managing, and controlling changes made to datasets over time. Just like software version control (e.g., Git for code), data versioning ensures that every modification, addition, or deletion in a dataset is recorded, allowing users to reproduce experiments, roll back to previous states, and maintain consistency across projects.

Write a code in python get upload the files in GIT

<https://github.com/n1000/ml_pipeline/tree/main>

**Step 8 : (Directory 9)**

**Model Building** is the process of developing a machine learning (ML) or statistical model that can learn from data and make predictions or decisions.

It involves preparing the data, choosing the right algorithm, training the model, evaluating its performance, and tuning it for accuracy.

A screenshot of a computer

Description automatically generated

**Step 9 : (Directory 10)**

Sample Python code

The word **“Orchestrate”** in data science / ML / cloud computing contexts means:

**Coordinating and automating multiple tasks, processes, or services so they work together smoothly as one system.**

It’s like being a **conductor of an orchestra** — ensuring each instrument (data pipeline, model training, deployment, monitoring) plays at the right time in harmony.

A screenshot of a computer program

Description automatically generated

**Step 10 : (A word document)**

**Content as follows**

### **Detailed Documentation: End-to-End Data Management Pipeline for Machine Learning**

## **Explanation of the Pipeline Design**

### **Overview**

The objective of this pipeline is to design, implement, and orchestrate a complete data management pipeline for customer churn prediction. The pipeline encompasses the full lifecycle of data management, from ingestion to orchestration, ensuring data quality and model reliability.

### **Pipeline Architecture**

The pipeline follows a modular architecture with the following stages:

1. **Problem Formulation**
   * Define the business problem and objectives.
   * Identify key data sources (transaction logs, web interactions, APIs).
   * Establish expected outputs (clean datasets, transformed features, deployable model).
2. **Data Ingestion**
   * Automated fetching of data from sources (e.g., databases, APIs).
   * Implementation of error handling mechanisms for failed ingestions.
   * Logging mechanisms for tracking ingestion status.
3. **Raw Data Storage**
   * Data stored in a data lake (AWS S3, Google Cloud Storage).
   * Partitioning by source, type, and timestamp for efficient retrieval.
4. **Data Validation**
   * Implementation of validation checks for missing values, data types, and anomalies.
   * Use of tools like Great Expectations or PyDeequ to generate data quality reports.
5. **Data Preparation**
   * Handling missing values via imputation or removal.
   * Standardization of numerical attributes.
   * Encoding of categorical variables.
6. **Feature Engineering & Transformation**
   * Aggregation of customer behavior features.
   * Generation of derived features such as tenure and frequency.
   * Storage in a feature store for retrieval.
7. **Data Versioning**
   * Use of DVC (Data Version Control) for dataset tracking.
   * Maintenance of metadata and version history.
8. **Model Training**
   * Experimentation with ML algorithms (Logistic Regression, Random Forest).
   * Evaluation using accuracy, precision, recall, and F1-score.
   * Model tracking with MLflow.
9. **Pipeline Orchestration**
   * Automation via Apache Airflow to ensure task dependencies.
   * Monitoring and logging of failures.
   * Visualization of DAGs for task execution.

## **Challenges Faced and Solutions Implemented**

| **Challenge** | **Description** | **Solution Implemented** |
| --- | --- | --- |
| **Data Collection** | Inconsistent data sources, missing labels, and imbalanced datasets. | Implemented active learning and semi-supervised learning for labeling; used data augmentation to balance classes. |
| **Data Granularity** | Some sources lacked fine-grained timestamped data. | Applied lossless data aggregation techniques to retain necessary details. |
| **Data Quality Issues** | Missing values, duplicate records, and incorrect formats. | Used pandas for handling missing values and Great Expectations for automated data validation. |
| **Handling High Cardinality Categorical Data** | Customer categories and identifiers introduced feature explosion. | Used **embedding techniques** instead of one-hot encoding to reduce dimensionality. |
| **Feature Drift and Data Drift** | Changing customer behaviour affected model performance over time. | Implemented **monitoring scripts** to track drift and retrain the model when performance drops. |
| **Error Handling in Data Ingestion** | API failures and incomplete logs disrupted ingestion. | Added **error logging and retry mechanisms** to ensure robust data fetching. |
| **Pipeline Failures & Dependency Management** | Interdependent tasks failed due to cascading failures in ingestion or validation. | Defined DAG dependencies in **Apache Airflow** and added alerting mechanisms. |
| **Versioning and Reproducibility Issues** | Dataset changes impacted model consistency. | Used **DVC** to version control both raw and transformed datasets. |
| **Model Overfitting** | Model performed well on training but failed on new data. | Implemented **regularization, dropout techniques, and cross-validation.** |

**Finally Group Recording of all executives**

**Ex. https://drive.google.com**