2. Data Ingestion	File folder
3. Raw Data Storage	File folder
4. Data Validation	File folder
5. Data Preparation	File folder
6. Data Transformation and Stor...	File folder
7. Feature Store	File folder
8. Data Versioning	File folder
9. Model Building	File folder
10. Orchestrate	File folder
.DS_Store	DS_STORE File
DMML Assignment - Group 38	Microsoft Word Document

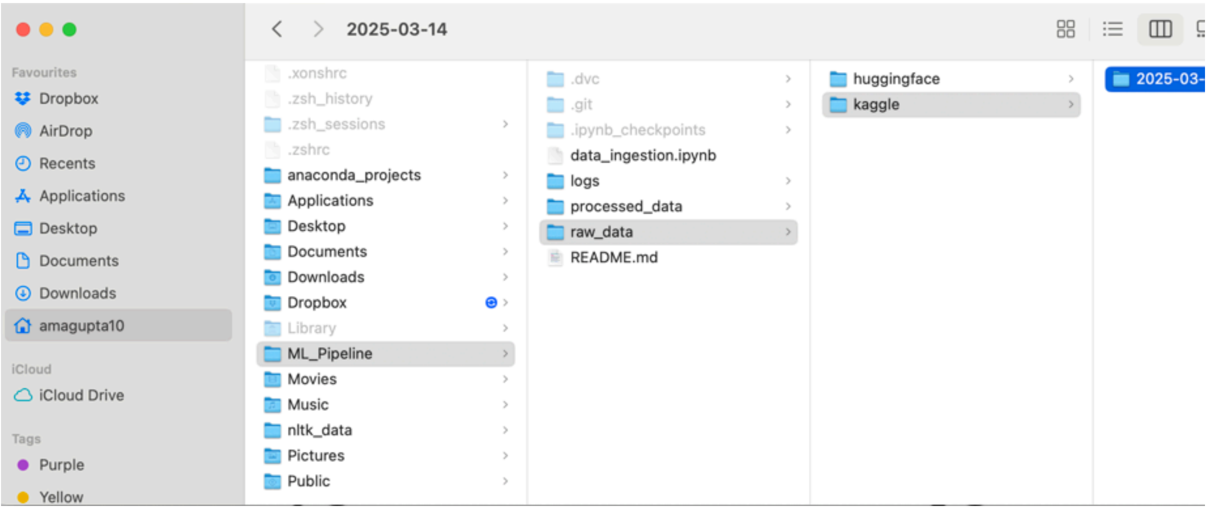
Step1 (Directory 2):

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



data_storage_huggingface	PNG File	294 KB	No	323 KB	10%	14-03-2025 21:14
data_storage_kaggle	PNG File	306 KB	No	337 KB	10%	14-03-2025 21:15

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

missing_va	duplicate_	data_types	negative_values	
	0			
0		object		
0		object		
0		object		
0		object		
0		object		
0		float64		
0		object		
0		object		
0		object		
0		object		
0		object		
0		object		
0		object		
0		int64		
0		object		
0		object		
0		object		
0		object		
0		object		
0		object		
0		int64		

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.

.DS_Store	DS_STORE File
cleaned_churn_data	Microsoft Excel Comma S...
Data_Process.ipynb	IPYNB File
data_processing	Text Document

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.

.DS_Store	DS_STORE File
churn_model_training	Text Document
LogisticRegression_churn_model...	PKL File
model_performance_report	Text Document
Model_train.ipynb	IPYNB File

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.

```
"def data_ingestion():\n",
"    logging.info(\"Data ingestion started.\")\n",
"    logging.info(\"Data ingestion completed.\")\n",
"\n",
"def raw_data_storage():\n",
"    logging.info(\"Raw data storage started.\")\n",
"    logging.info(\"Raw data storage completed.\")\n",
"\n",
"def data_validation():\n",
"    logging.info(\"Data validation started.\")\n",
"    logging.info(\"Data validation completed.\")\n",
"\n",
"def data_preparation():\n",
"    logging.info(\"Data preparation started.\")\n",
"    logging.info(\"Data preparation completed.\")\n",
"\n",
"def data_transformation():\n",
"    logging.info(\"Data transformation started.\")\n",
"    logging.info(\"Data transformation completed.\")\n",
"\n",
"def feature_store():\n",
"    logging.info(\"Feature store tasks started.\")\n",
"    logging.info(\"Feature store tasks completed.\")\n",
"\n",
"def data_versioning():\n",
"    logging.info(\"Data versioning started.\")\n",
"    logging.info(\"Data versioning completed.\")\n",
"\n",
"def model_building():\n",
"    logging.info(\"Model building started.\")\n",
"    logging.info(\"Model building completed.\")\n",
"default_args = {\n",
"    'owner': 'airflow',\n",
"    'start_date': days_ago(1),\n",
"    'retries': 1,\n",
"}\n",
"with DAG(\n",
"    'dm4ml_data_pipeline',\n",
"    default_args=default_args,\n",
"    description='Orchestrates the end-to-end data pipeline for customer churn prediction',\n",
"    schedule_interval='*/10 * * * *',\n",
"    catchup=False,\n",
") as dag:\n",
"    \n",
"    task_ingestion = PythonOperator(\n",
"        task_id='data_ingestion',\n",
"        python_callable=data_ingestion\n",
"    )\n",
"    \n",
"    task_storage = PythonOperator(\n",
"        task_id='raw_data_storage',\n",
"        python_callable=raw_data_storage\n",
"    )\n",
"    \n",
```

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.

Challenge	Description	Solution Implemented
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