IF4054 Pengoperasian Perangkat Lunak

CUSTOMER CHURN PREDICTION WORKFLOW

Disusun oleh:

Alexander Jason	**13521100**
Nathania Calista	**13521139**
Haziq Abiyyu Mahdy	**13521170**

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Daftar Isi

Daftar Tabel

Daftar Gambar

A. Pendahuluan

Dataset “Telco Customer Churn” merupakan kumpulan data pelanggan dari sebuah perusahaan telekomunikasi yang bertujuan untuk menganalisis faktor-faktor yang memengaruhi pelanggan berhenti berlangganan layanan mereka (*churn*). Dataset ini dapat digunakan untuk membangun model prediktif yang membantu mengidentifikasi pelanggan yang berpotensi churn. Berikut ini merupakan deskripsi kolom dari dataset:

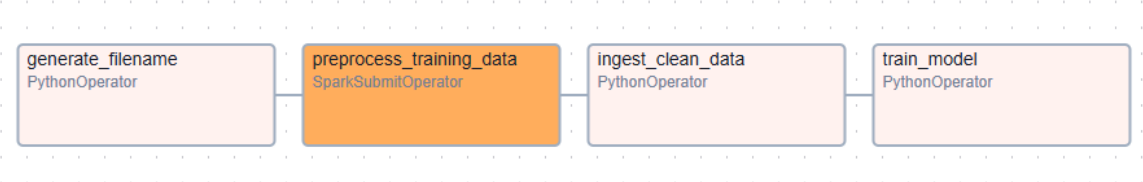
Tabel 1.1. Deskripsi kolom dataset

No.	Nama Kolom	Tipe Data	Deskripsi
1	customerId	object	ID unik untuk setiap pelanggan.
2	gender	object	Jenis kelamin pelanggan (Male/Female)
3	SeniorCitizen	int64	Indikator apakah pelanggan adalah warga senior (1 = Ya, 0 = Tidak)
4	Partner	object	Apakah pelanggan memiliki pasangan (Yes/No)
5	Dependents	object	Apakah pelanggan memiliki tanggungan (Yes/No)
6	tenure	int64	Jumlah bulan pelanggan telah menggunakan layanan
7	PhoneService	object	Apakah pelanggan memiliki layanan telepon (Yes/No)
8	MultipleLines	object	Apakah pelanggan memiliki beberapa jalur telepon (Yes/No/No phone service)
9	InternetService	object	Jenis layanan internet (DSL, Fiber optic, No)
10	OnlineSecurity	object	Apakah pelanggan memiliki layanan keamanan online (Yes/No/No internet service)
11	OnlineBackup	object	Apakah pelanggan memiliki layanan pencadangan online (Yes/No/No internet service)
12	DeviceProtection	object	Apakah pelanggan memiliki layanan perlindungan perangkat (Yes/No/No internet service)
13	TechSupport	object	Apakah pelanggan memiliki layanan dukungan teknis (Yes/No/No internet service)
14	StreamingTV	object	Apakah pelanggan memiliki layanan streaming TV (Yes/No/No internet service)

15	StreamingMovies	object	Apakah pelanggan memiliki layanan streaming film (Yes/No/No internet service)
16	Contract	object	Jenis kontrak pelanggan (Month-to-month, One year, Two year)
17	PaperlessBilling	object	Apakah pelanggan menggunakan penagihan tanpa kertas (Yes/No)
18	PaymentMethod	object	Metode pembayaran (Electronic check, Mailed check, Bank transfer, Credit card)
19	MonthlyCharges	float64	Biaya bulanan yang dibayarkan pelanggan
20	TotalCharges	object	Total biaya yang telah dibayarkan pelanggan selama ini
21	Churn	object	Apakah pelanggan berhenti berlangganan (Yes/No)

B. Machine Learning Workflow

1. Experiment DAG



Gambar 2.1 Experiment DAG

Diagram ini menunjukkan alur kerja yang dilakukan hanya satu kali untuk eksperimen. Berikut penjelasan langkah-langkahnya:

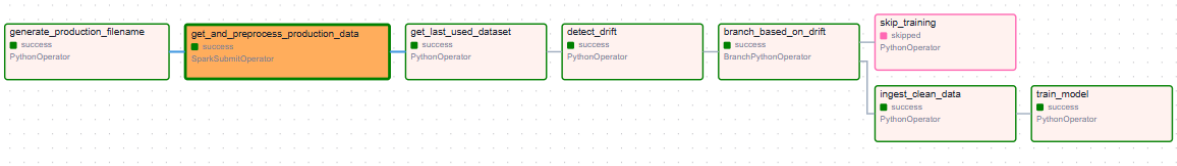
a) Generate Filename
Pada tahap ini, nama file dibangkitkan berdasarkan *timestamp*. Nama file ini berguna sebagai *object key* dari data yang akan dibersihkan pada tahap setelahnya.

b) Preprocess Training Data (Menggunakan Spark)
Pada tahap ini, dataset diambil dari volume dan diproses untuk membersihkan dan mempersiapkannya untuk langkah berikutnya. Proses mencakup:
- Menghapus data duplikat.
- Mengisi atau menghapus nilai yang kosong (N/A).
- Seleksi fitur untuk memilih atribut yang relevan.
- Transformasi untuk menyesuaikan format data (seluruh data kategorikal diubah menjadi numerik)
- Data yang telah diproses akan disimpan ke MinIO menggunakan *object key* dari tahap sebelumnya

c) Ingest Clean Data
Data yang telah dibersihkan diambil dari MinIO dan dikonversi menjadi DataFrame yang akan disimpan pada XCom untuk diakses pada tahap Train Model. Setiap kali dataset di-*ingest* untuk melakukan *training*, dataset yang di-*ingest* akan dicatat sebagai *last_used_dataset* untuk dijadikan referensi pada perhitungan PSI.

d) Train Model
Pada tahap ini, dilakukan eksperimen dengan beberapa algoritma seperti Random Forest dan Logistic Regression. Tiap model akan di-*log* ke MLFlow beserta konfigurasi model seperti algoritma dan parameter serta metrik yang dihitung. Artefak model dengan metrik terbaik akan dicatat sebagai *last_used_artifact_path* agar dapat diakses oleh *service* lain.

2. Production DAG



Gambar 2.2 Production DAG

Diagram ini menunjukkan alur kerja yang dijalankan secara berkala (misalnya, setiap beberapa menit). *Workflow* ini dioptimalkan untuk dalam proses *production*, di mana data selalu dihasilkan secara berkala.

a) Generate Production Filename
   Pada tahap ini, nama file dibangkitkan berdasarkan *timestamp*. Nama file ini berguna sebagai *object key* dari data yang akan dibersihkan pada tahap setelahnya.
b) Get and Preprocess Production Data (Menggunakan Spark)
   Pada tahap ini, dataset diambil dari *script* untuk meng-*generate* data untuk mensimulasikan data *production* dan diproses seperti tahapan Preprocess Training Data pada Experiment DAG. Data yang telah diproses akan disimpan ke MinIO menggunakan *object key* dari tahap sebelumnya.
c) Get Last Used Dataset
   Dataset yang terakhir digunakan untuk *training* (*last_used_dataset*) diambil dari minIO untuk dijadikan referensi pada perhitungan PSI.
d) Detect Drift
   Perhitungan PSI membandingkan antara data *production* yang dihasilkan pada tahap b dengan *last_used_dataset*. Apabila PSI di atas ambang tertentu, maka terdapat *data drift* dan diperlukan *training* ulang dengan *dataset* baru.
e) Branch Based On Drift
   Pada tahap ini, digunakan BranchPythonOperator untuk menentukan *task* selanjutnya yang akan dijalankan berdasarkan *data drift*.
f) Skip Training
   Apabila tidak terdapat *data drift*, maka tidak diperlukan *training* ulang.
g) Ingest Clean Data
   Apabila terdapat data *drift*, maka diperlukan *training* ulang. Oleh karena itu, data *production* yang telah diproses sebelumnya akan diambil dari MinIO.
h) Train Model
   Pada tahap ini, dilakukan eksperimen dengan beberapa algoritma seperti Random Forest dan Logistic Regression. Tiap model akan di-*log* ke MLFlow beserta konfigurasi model seperti algoritma dan parameter serta metrik yang dihitung. Artefak model dengan metrik terbaik akan dicatat sebagai *last_used_artifact_path* agar dapat diakses oleh *service* lain.

C. Model Deployment

Model di-*deploy* menggunakan FastAPI yang menyediakan API untuk memprediksi *churn* berdasarkan data yang dimasukkan. Terdapat *endpoint* POST /predict yang menerima data pada *request body* dan mengembalikan hasil prediksi (True/False). Dokumentasi API dapat diakses melalui http://localhost:5555/docs. *Endpoint* ini akan mengambil artefak model terbaru dari MinIO, serta memprediksi *churn* berdasarkan data *input*.

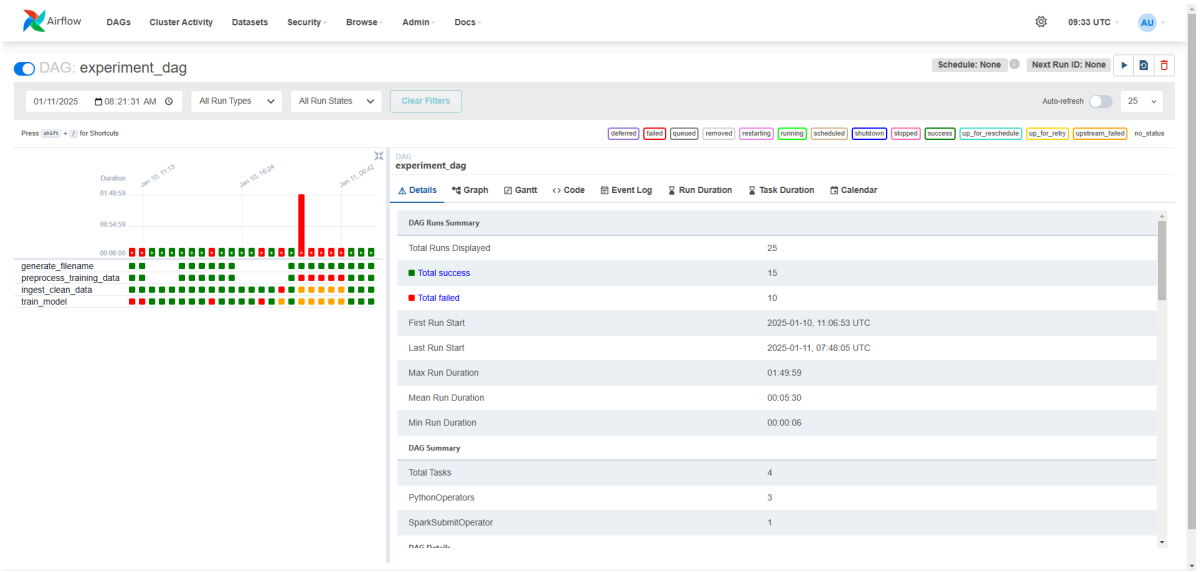
D. Implementasi

Projek ini diimplementasikan dengan Docker pada satu *node*. Berikut merupakan komponen dari projek.

1. Airflow (sleek-airflow)
   Digunakan untuk mengatur dan mengelola ML Workflow.
2. Apache Spark (spark-master, spark-worker-1 dan spark-worker-2)
   Framework untuk pemrosesan data secara terdistribusi. Terdiri dari satu master (spark-master) dan dua worker (spark-worker-1 dan spark-worker-2).
3. MinIO (minio dan minio-client)
   Berfungsi sebagai *S3-compatible object storage* untuk menyimpan artefak model dan dataset
4. MLflow (mlflow dan mlflow-db)
   Digunakan untuk melacak eksperimen, menyimpan model yang dilatih, dan mengelola metadata eksperimen. Database backend (PostgreSQL) digunakan untuk menyimpan metadata eksperimen, sedangkan artifact disimpan di MinIO.
5. Model Deployment (model-deploy)
   Layanan berbasis FastAPI untuk menyediakan layanan prediksi menggunakan model yang telah dilatih

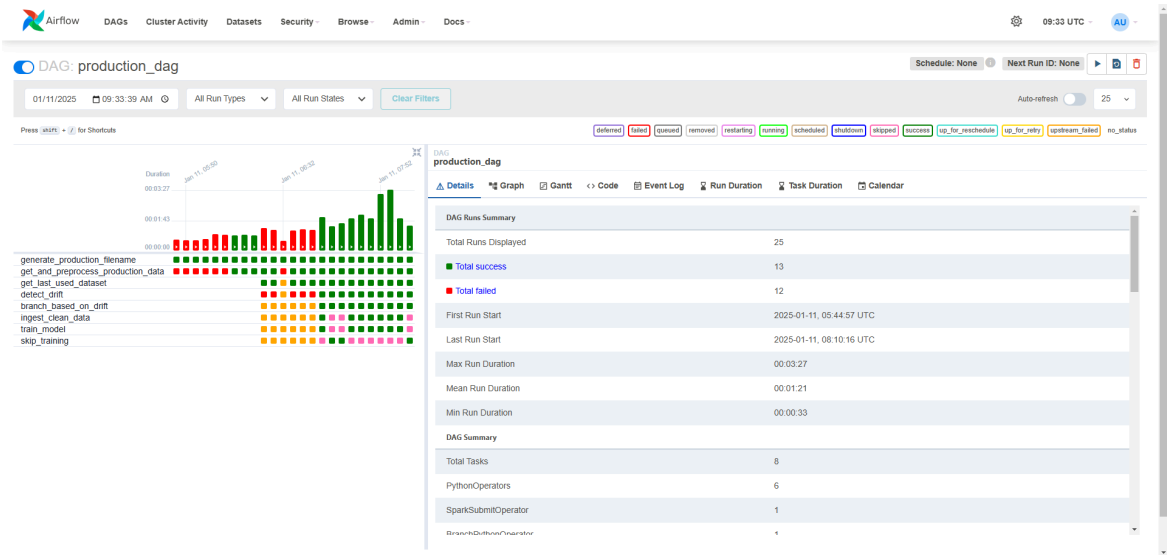
E. Hasil Implementasi

Berikut merupakan status eksekusi DAG (Directed Acyclic Graph) bernama experiment_dag di Apache Airflow. DAG ini digunakan untuk mengatur alur eksperimen, termasuk preprocessing data, pelatihan model, dan logging ke MLflow. Dari grafik dan statistik, terlihat jumlah eksekusi berhasil (hijau), gagal (merah), dan total waktu eksekusi.



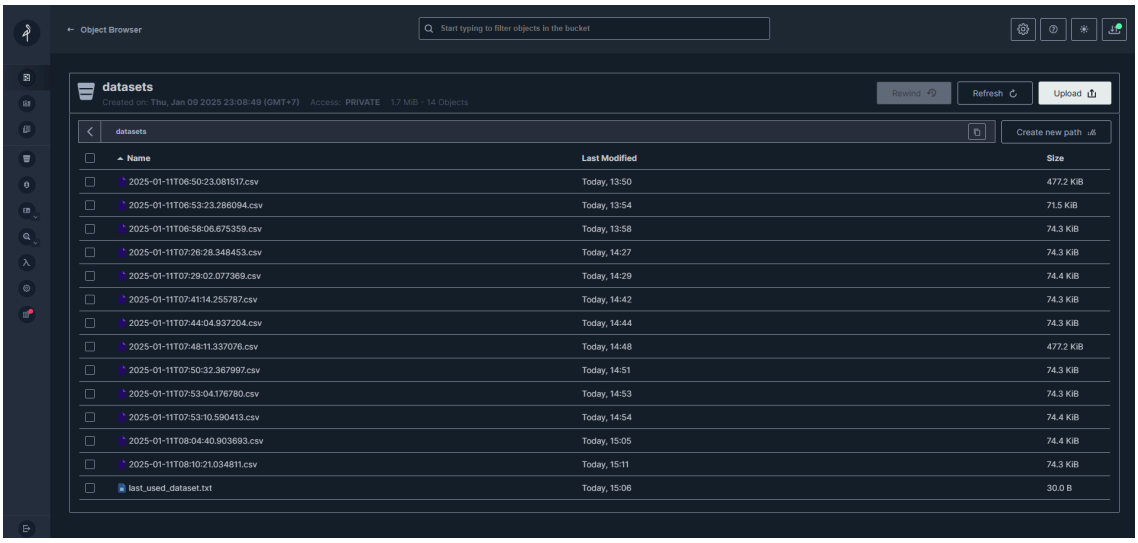
Gambar 5.1 Experiment DAG Runs

Berikut merupakan status eksekusi DAG production_dag yang dikhususkan untuk kondisi *production*, seperti pendeteksian data drift, pelatihan ulang model, dan pembaruan model di MLflow. Grafik menyoroti pola eksekusi, dengan informasi tentang task yang berhasil dan gagal, memastikan stabilitas pipeline produksi.



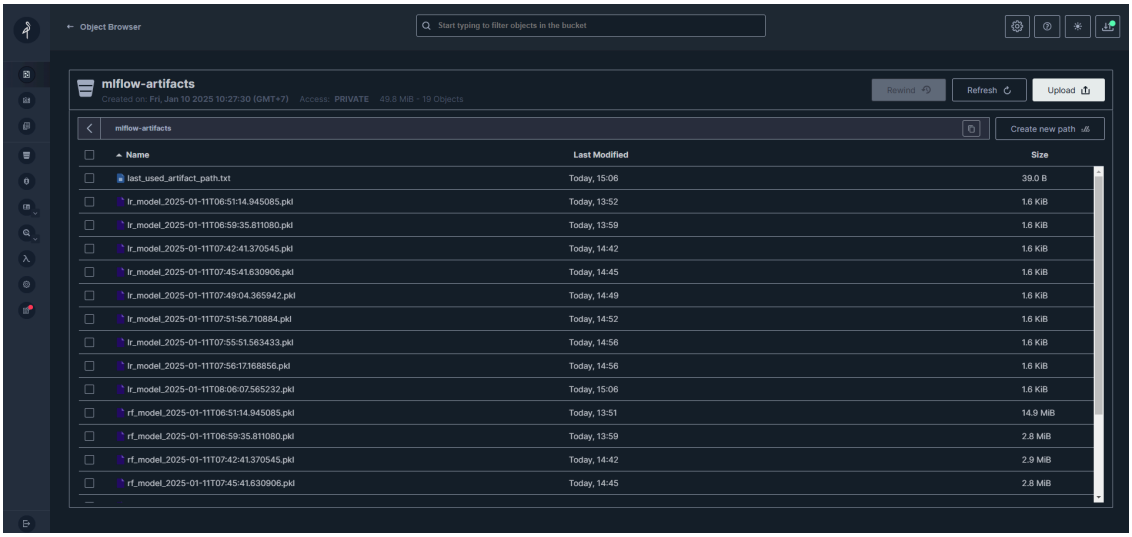
Gambar 5.2 Production DAG Runs

Kemudian, daftar dataset yang disimpan di MinIO, yang digunakan sebagai penyimpanan objek untuk pipeline. Dataset ini mencakup file CSV hasil preprocessing serta dataset terbaru yang digunakan untuk pelatihan atau validasi model. Struktur penyimpanan ini memastikan aksesibilitas dataset dalam alur kerja.



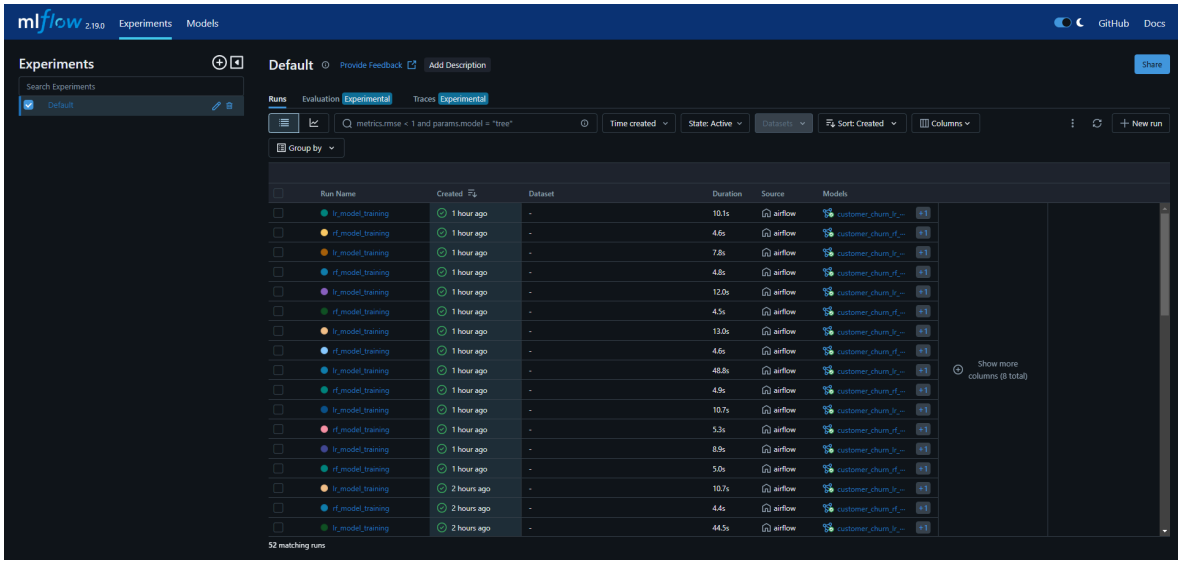
Gambar 5.3 Datasets di MinIO

Lalu artefak model disimpan di MinIO, seperti file model .pkl dan jalur artefak terakhir. Setiap artefak terkait dengan eksperimen atau versi model tertentu, memastikan replikasi dan manajemen artefak yang efisien dalam siklus hidup MLflow.



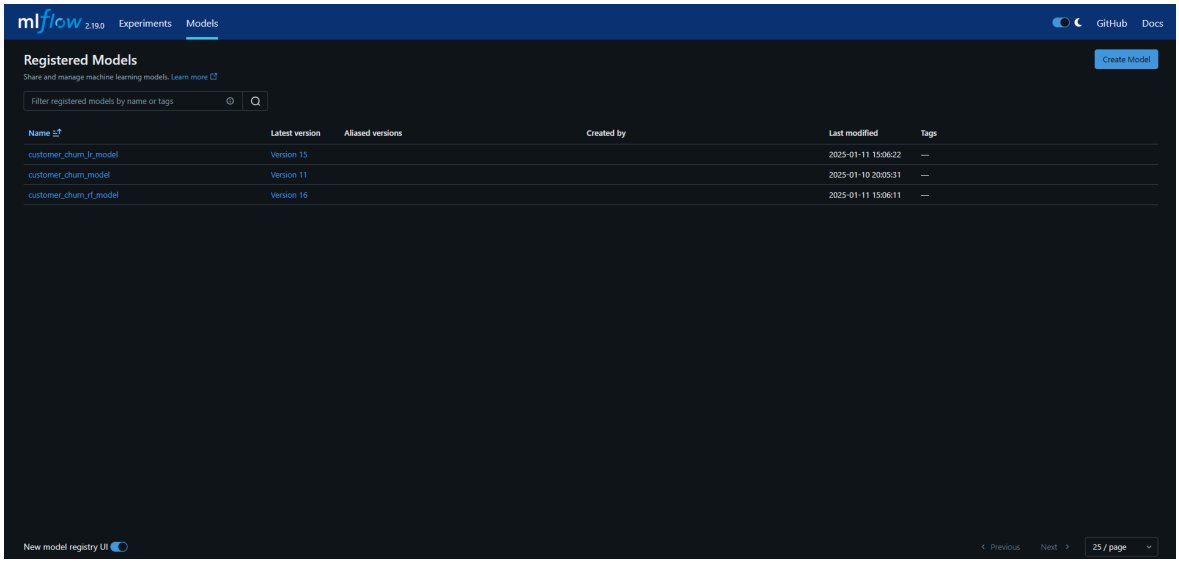
Gambar 5.4 Artefak di MinIO

Daftar eksperimen yang dilakukan akan dilog ke MLflow. Informasi seperti durasi eksekusi, sumber pipeline (Airflow), dan versi model terlihat di sini. Fitur ini membantu dalam memantau dan membandingkan performa berbagai eksperimen secara terpusat.



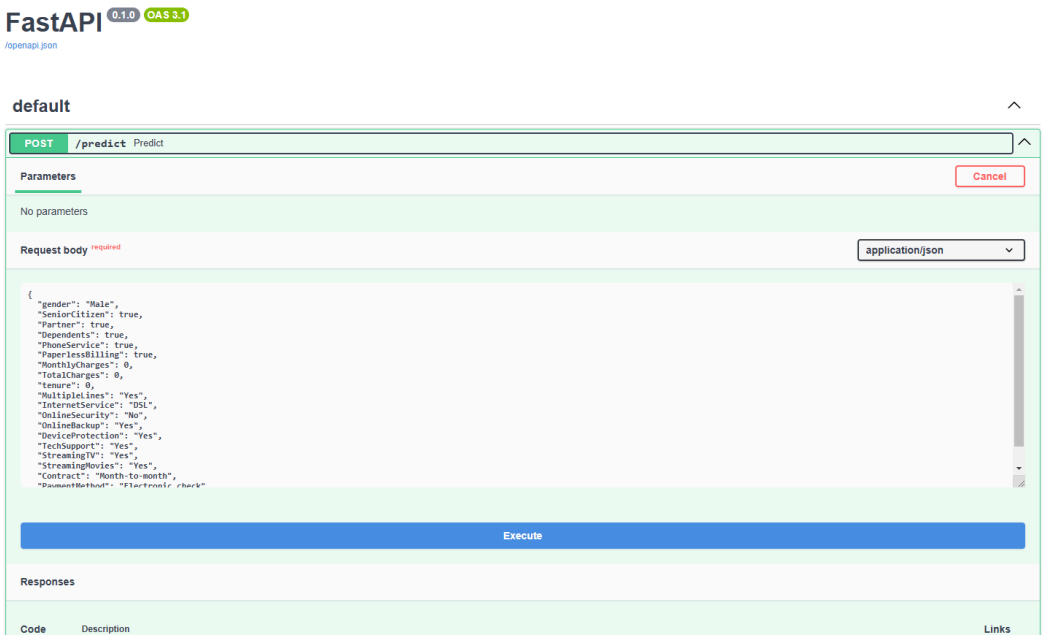
Gambar 5.5 MLFlow Experiment

Model yang sudah di-*train* akan diregistrasi di MLflow Model Registry. Setiap model memiliki versi unik (e.g., Version 10, Version 11), sehingga mempermudah pelacakan, promosi ke lingkungan produksi, atau rollback ke versi sebelumnya jika diperlukan.



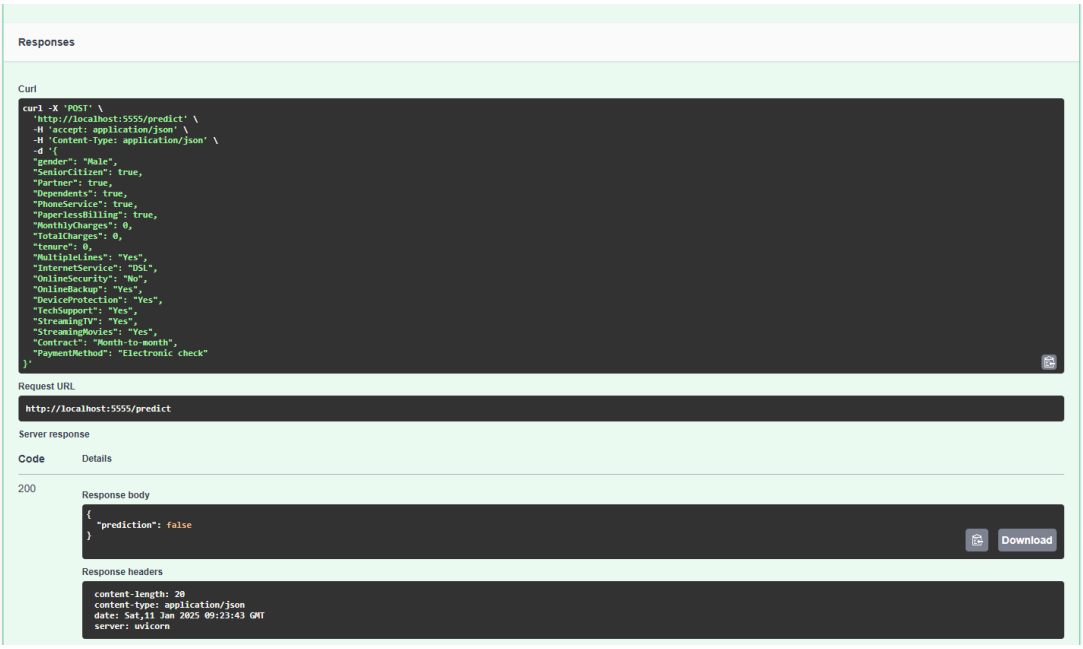
Gambar 5.6 MLFlow Model

Sebagai pengujian model, kami mengembangkan API yang dibangun menggunakan FastAPI. Endpoint /predict menerima input data dalam format JSON, yang mencakup fitur-fitur seperti gender, Partner, InternetService, dan lainnya, yang digunakan oleh model untuk menghasilkan prediksi. Formulir ini memungkinkan pengguna untuk melakukan pengujian interaktif terhadap API langsung melalui dokumentasi yang dihasilkan secara otomatis oleh FastAPI.



Gambar 5.7 [Bonus] Prediction API Request

Lalu respons API dari endpoint /predict akan ditampilkan. API mengembalikan hasil prediksi dalam format JSON, seperti { "prediction": false }, yang menunjukkan bahwa model memprediksi pelanggan tidak akan berhenti berlangganan. Selain itu, detail header respons, payload JSON, dan curl command untuk menjalankan request juga ditampilkan, mempermudah debugging dan integrasi dengan sistem lain.



Gambar 5.8 [Bonus] Prediction API Response

F. Kesimpulan

Proyek ini mengintegrasikan MLflow, Airflow, Spark, dan Docker untuk membangun *workflow* data dan machine learning yang efisien, otomatis, dan skalabel.

- MLflow mengelola siklus hidup model
- Airflow mengorkestrasi setiap *task* pada *workflow*
- Spark menangani pemrosesan *big data* secara paralel
- Docker menyediakan lingkungan konsisten untuk pengembangan dan produksi.

Kombinasi teknologi ini menciptakan *workflow* yang dapat diandalkan untuk proses *machine learning* dari *data preprocessing* hingga *deployment*. *Lesson Learned* dari tugas ini:

- Orkestrasi dengan Airflow
  Airflow mempermudah pengelolaan alur kerja kompleks, tetapi membutuhkan pemahaman mendalam untuk konfigurasi dan debugging.
- Pengelolaan Model dengan MLFlow
  Logging, registrasi, dan *versioning* model dengan MLFlow meningkatkan reprodusibilitas dan transparansi eksperimen *machine learning*.
- Skalabilitas Spark
  Spark memungkinkan *preprocessing* data dengan cepat, tetapi memerlukan ketelitian ekstra dalam *debugging*.
- Pentingnya Logging dan Error Handling
  Logging sangat diperlukan untuk melihat status program serta *debugging*.

G. Referensi

https://mlflow.org/docs/latest/index.html
https://spark.apache.org/docs/latest/
https://airflow.apache.org/docs/

H. Pembagian Kerja

Tabel 8.1. Pembagian Kerja

Nama Anggota	**NIM Anggota**	**Pembagian Kerja**
Haziq Abiyyu Mahdy	13521070	• Setup environment dan **Docker** • Data Pre-processing menggunakan **Apache Spark** • Experiment DAG (**Airflow**) • Model Deployment • Laporan • Video
Alexander Jason	13521100	• Setup environment dan **Docker** • Model Training & Retraining menggunakan **MLflow** • Data Drift Detection • Laporan • Video
Nathania Calista	13521139	• Setup environment dan **Docker** • Production DAG (**Airflow**) • Integrasi Airflow dan MLflow • Laporan • Video

I. Lampiran

Berikut adalah daftar laman yang digunakan selama pengerjaan tugas besar ini:

- *Repository* Gitlab
  https://gitlab.informatika.org/nathaniaclista/if4054-xops
  https://github.com/nathaniacalista01/XOps (repo ini digunakan karena gitlab down pada saat mendekati pengumpulan)
- Video : https://youtu.be/_EBamoXhTyw