KERJA PRAKTIK - EF234603

Pengembangan Pipeline Berbasis MLOps untuk Experiment Tracking pada Infrastruktur Data Science

PT. United Tractors Tbk
Jl. Raya Bekasi No.KM.22, RT.7/RW.1, Cakung Bar., Kec. Cakung, Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta 13910
Periode: 20 Agustus 2024 – 20 Januari 2025

Oleh:

Rayhan Arvianta Bayuputra 5025211217

Pembimbing Departemen
Dini Adni Navastara, S.Kom., M.Sc.
Pembimbing Lapangan
Dedi Prananda, S.E.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – EF234603

Pengembangan Pipeline Berbasis MLOps untuk Experiment Tracking pada Infrastruktur Data Science

PT. United Tractors Tbk
Jl. Raya Bekasi No.KM.22, RT.7/RW.1, Cakung Bar., Kec. Cakung, Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta 13910
Periode: 20 Agustus 2024 – 20 Januari 2025

Oleh:

Rayhan Arvianta Bayuputra 5025211217

Pembimbing Departemen
Dini Adni Navastara, S.Kom., M.Sc.

Pembimbing Lapangan
Dedi Prananda, S.E.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

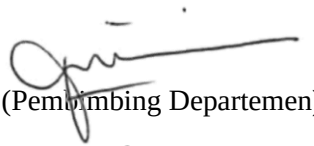
Pengembangan Pipeline Berbasis MLOps untuk Experiment Tracking pada Infrastruktur Data Science

Oleh:

Rayhan Arvianta Bayuputra 5025211217

Disetujui oleh Pembimbing Kerja Praktik:

1. Dini Adni Navastara, S.Kom., M.Sc.
   NIP.



(Pembimbing Departemen)

2. Dedi Prananda, S.E.

(Pembimbing Lapangan)

JAKARTA

5 Februari 2025

[Halaman ini sengaja dikosongkan]

Pengembangan Pipeline Berbasis MLOps untuk Experiment Tracking pada Infrastruktur Data Science

Nama Mahasiswa : Rayhan Arvianta Bayuputra
NRP : 5025211217
Departemen : Teknik Informatika FTEIC-ITS
Pembimbing Departemen : Dini Adni Navastara, S.Kom., M.Sc.
Pembimbing Lapangan : Dedi Prananda, S.E.

ABSTRAK

Kerja praktik ini dilakukan di PT United Tractors Tbk untuk meningkatkan efisiensi siklus pembelajaran mesin melalui penerapan MLOps (*Machine Learning Operations*). Proyek ini difokuskan pada pengembangan *pipeline machine learning* modular yang terdiri dari enam lapisan, dengan implementasi MLOps yang terpusat pada *experiment tracking* menggunakan Databricks. Penerapan ini berhasil meningkatkan keterlacakan eksperimen, efisiensi, dan reproduktibilitas pengembangan model, sekaligus mendukung kolaborasi tim data science. Pipeline yang terstandarisasi ini diharapkan menjadi pedoman dalam pengembangan dan pengoperasian model machine learning di masa depan.

Kata Kunci: MLOps, *machine learning*, *experiment tracking*, Databricks, *pipeline machine learning*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Dengan penuh rasa syukur, penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas rahmat dan bimbingan-Nya, sehingga penulis dapat menyelesaikan Kerja Praktik di PT United Tractors Tbk dengan baik. Penulis memahami bahwa terdapat banyak hal yang masih perlu diperbaiki, baik dalam pelaksanaan kerja praktik maupun dalam penyusunan laporan ini. Namun demikian, penulis berharap laporan ini dapat memberikan manfaat, memperluas wawasan pembaca, dan berkontribusi sebagai bahan referensi yang berguna.

Melalui laporan ini, penulis juga ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan, baik secara langsung maupun tidak langsung, dalam proses penyusunan laporan kerja praktik ini, antara lain:

1. PT United Tractors Tbk selaku mitra industri yang telah memberikan kesempatan bagi penulis untuk melakukan Kerja Praktik di organisasinya.
2. Bapak Ir. Bambang Pramujati, S.T., M.Sc.Eng., Ph.D., selaku Rektor Institut Teknologi Sepuluh Nopember.
3. Bapak Dr. I Ketut Eddy Purnama, S.T., M.T., selaku Dekan Fakultas Teknologi Elektro dan Informatika Cerdas.
4. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D., IPM., selaku Kepala Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember.
5. Ibu Dini Adni Navastara, S.Kom., M.Sc., selaku Dosen Pembimbing Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember.
6. Bapak Dedi Prananda, S.E., selaku Pebimbing Lapangan selama masa Kerja Praktik di PT United Tractors Tbk.

7. Mas Angga Wahyu Pratama, S.Si., M.Kom., selaku *sub-mentor* selama masa Kerja Praktik di PT United Tractors Tbk.
8. Teman-teman serta keluarga penulis yang senantiasa memberikan dukungan dan doa selama penulis menjalankan Kerja Praktik.

Jakarta, 21 Januari 2025

Rayhan Arvianta Bayuputra

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PSEUDOCODE

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sejak diperkenalkannya konsep Industri 4.0, teknologi informasi telah menjadi pendorong utama transformasi di berbagai sektor industri. Perusahaan kini memanfaatkan pengambilan keputusan berbasis data (data-driven decision making) yang didukung oleh kecerdasan buatan (AI) untuk meningkatkan efisiensi dan daya saing. Integrasi teknologi ini memungkinkan analisis data secara real-time, prediksi tren pasar, dan otomatisasi proses bisnis, yang semuanya berkontribusi pada peningkatan produktivitas dan inovasi dalam operasional perusahaan.

PT United Tractors Tbk, didirikan pada tahun 1972, telah berkembang menjadi distributor alat berat terbesar di Indonesia. Perusahaan ini menyediakan produk dari merek-merek ternama dunia seperti Komatsu, UD Trucks, Scania, Bomag, Tadano, dan Komatsu Forest. Selama lebih dari lima dekade, United Tractors berperan signifikan dalam pembangunan Indonesia, khususnya di sektor pertambangan, konstruksi, dan energi. Melalui penyediaan alat berat dan layanan terkait, perusahaan ini mendukung berbagai proyek infrastruktur dan pengembangan industri yang menjadi tulang punggung pertumbuhan ekonomi nasional.

Seiring dengan perkembangan kebutuhan pasar, PT United Tractors Tbk mulai melakukan akselerasi transformasi digital sejak tahun 2017. Langkah ini mencakup berbagai inisiatif digitalisasi untuk meningkatkan efisiensi operasional dan kualitas layanan. Dengan memanfaatkan teknologi terkini, perusahaan mampu menciptakan proses bisnis yang lebih adaptif, terintegrasi, dan responsif terhadap perubahan pasar. Transformasi ini tidak hanya mencakup pengembangan infrastruktur digital, tetapi juga investasi pada pengembangan talenta digital untuk mendukung keberlanjutan inovasi perusahaan. Upaya ini mencerminkan

komitmen PT United Tractors Tbk dalam menghadapi tantangan era digital sekaligus memastikan relevansi peran perusahaan di masa depan.

1.2. Tujuan

Tujuan kerja praktik ini adalah sebagai berikut:

1. Memenuhi kewajiban akademik berupa penyelesaian nilai kerja praktik sebesar 4 SKS.
2. Peningkatan kualitas *Machine Learning Life Cycle* dengan mengimplementasikan prinsip-prinsip *MLOps* dalam siklus data science yang telah ada.

1.3. Manfaat

Manfaat dari kerja praktik ini adalah sebagai berikut:

1. Menyediakan standar baru yang dapat diterapkan pada berbagai *use-case* untuk menjaga kualitas kode yang dihasilkan dalam setiap proyek.
2. Memperkenalkan dan menerapkan prinsip-prinsip *Machine Learning Operations (MLOps)* untuk meningkatkan efisiensi dan kualitas machine learning life cycle yang sudah ada, khususnya pada aspek *experiment tracking*.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana struktur dan mekanisme kerja standarisasi kode yang akan dibuat?
2. Bagaimana prinsip-prinsip *MLOps* dapat diterapkan pada standarisasi kode yang baru?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi	: *Hybrid* (*Work from home* dan United Tractors *Head Office*)
Waktu	: 20 Agustus 2024 – 20 Januari 2025
Hari Kerja	: Senin – Jumat
Jam Kerja	: 07.30 – 16.30 (Senin – Kamis) 07.30 – 17.00 (Jumat)

1.6. Metodologi Kerja Praktik

1.6.1. Perumusan Masalah

Untuk memahami kebutuhan tim data science, struktur tim, serta cara kerjanya, saya mendapat penjelasan langsung dari Bapak Dedi, selaku Head of Data Science and Advanced Analytics Team dari divisi Differentiation and Digitalization PT United Tractors Tbk. Setelah itu, dilakukan pembahasan lebih lanjut terkait kebutuhan dan optimalisasi cara kerja tim bersama Mas Angga selaku Data Scientist Team Lead.

1.6.2. Studi Literatur

Setelah merumuskan masalah terkait sistem yang akan dirancang, saya melakukan studi literatur untuk memahami dasar-dasar dan prinsip-prinsip MLOps, khususnya dalam aspek experiment tracking. Studi ini bertujuan untuk mendalami cara merekam, melacak, dan mengelola eksperimen model pembelajaran mesin secara efisien. Informasi yang dibutuhkan diperoleh melalui berbagai sumber, termasuk dokumentasi daring dan jurnal ilmiah yang relevan dengan implementasi experiment tracking dalam MLOps.

1.6.3. Analisis dan Perancangan

Berdasarkan hasil studi literatur dan analisis kebutuhan, dirancang struktur arsitektur machine learning yang terpisah dan modular pada setiap fasenya. Dalam arsitektur ini, experiment tracking diintegrasikan ke dalam alur kerja pengembangan untuk

memastikan prinsip-prinsip MLOps dapat diterapkan secara optimal. Pendekatan ini bertujuan untuk meningkatkan efisiensi, keterlacakkan, dan kualitas dalam siklus pengembangan *machine learning*.

1.6.4. Implementasi Sistem

Berdasarkan perancangan yang telah disusun, implementasi dan pembangunan arsitektur dilakukan untuk menyelesaikan masalah yang telah diidentifikasi. Proses ini bertujuan memastikan bahwa seluruh komponen arsitektur dapat diimplementasikan dengan baik dan berfungsi secara optimal sesuai dengan kebutuhan yang telah dirumuskan.

1.6.5. Pengujian dan Evaluasi

Setelah arsitektur dan modul-modulnya selesai dibangun, seluruh *pipeline* dari *framework* akan dijalankan untuk memastikan bahwa setiap komponen menghasilkan keluaran sesuai dengan yang diharapkan.

1.6.6. Kesimpulan dan Saran

Berdasarkan seluruh proses yang telah diimplementasikan dalam kerja praktik ini, kesimpulan dapat ditarik mengenai pencapaian yang telah diperoleh. Selain itu, saran-saran juga akan diberikan untuk perbaikan atau pengembangan lebih lanjut, guna meningkatkan efisiensi dan efektivitas sistem yang telah dibangun.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi rincian profil PT United Tractors Tbk, tempat dilaksanakannya kerja praktik saya.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi tinjauan pustaka mengenai teknologi yang digunakan dalam penyelesaian kerja praktik di United Tractors.

1.7.4. Bab IV Analisis dan Perancangan Sistem

Bab ini berisi mengenai tahap analisis arsitektur sistem yang akan dirancang dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi tahap - tahap yang dilakukan untuk proses implementasi arsitektur sistem yang telah dirancang.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari arsitektur yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil United Tractors

PT United Tractors Tbk (UT) adalah anak perusahaan dari PT Astra International Tbk, yang merupakan salah satu grup usaha terbesar di Indonesia. Sejak didirikan pada tahun 1972, UT telah berkembang menjadi distributor alat berat terbesar di Indonesia dan kini memiliki lima pilar bisnis utama: Mesin Konstruksi, Kontraktor Penambangan, Pertambangan, Industri Konstruksi, dan Energi.

Sebagai distributor tunggal alat berat Komatsu di Indonesia, UT memiliki jaringan layanan yang luas, termasuk 20 kantor cabang, 32 site support, 6 kantor perwakilan, dan 54 support point, yang siap memberikan solusi bagi pelanggan di berbagai sektor industri.

Sejak 19 September 1989, UT telah menjadi perusahaan publik dengan mencatatkan sahamnya di Bursa Efek Indonesia. Hingga saat ini, Astra memiliki 59,5% saham UT, dengan sisa saham dimiliki oleh publik.

Dengan komitmen terhadap keberlanjutan, UT secara konsisten melaksanakan kegiatan yang mendukung pencapaian Sasaran Pembangunan Berkelanjutan (SDGs), termasuk melalui program-program seperti UTREES, UTGROWTH, UTFUTURE, UTCARE, dan UTACTION.

2.2. Logo Perusahaan

Gambar 2.1 Logo United Tractors

2.3. Visi Misi Perusahaan

2.3.1. Visi

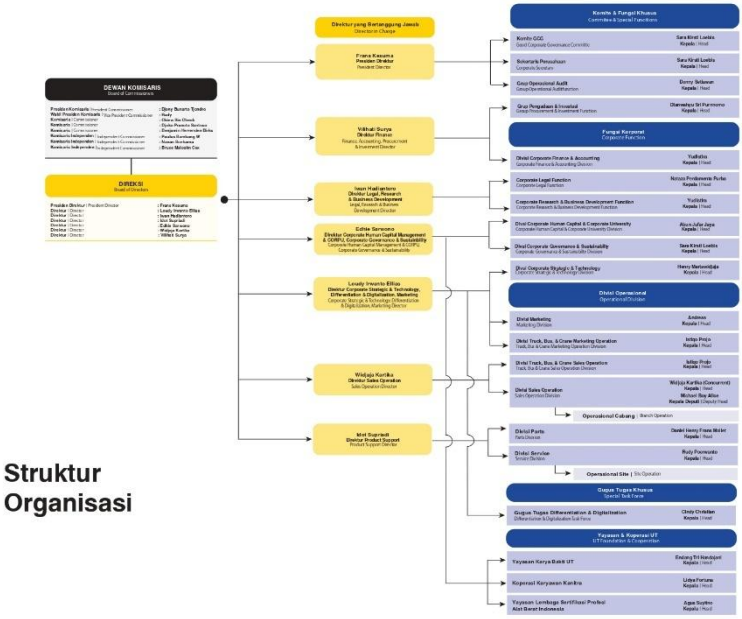
Menjadi perusahaan kelas dunia berbasis solusi di bidang alat berat, pertambangan dan energi, untuk menciptakan manfaat bagi para pemangku kepentingan.

2.3.2. Misi

Menjadi perusahaan yang:

1. Bertekad membantu pelanggan meraih keberhasilan melalui pemahaman usaha yang komprehensif dan interaksi berkelanjutan.
2. Menciptakan peluang bagi insan perusahaan untuk dapat meningkatkan status sosial dan aktualisasi diri melalui kinerjanya.
3. Menghasilkan nilai tambah yang berkelanjutan bagi para pemangku kepentingan melalui tiga aspek berimbang dalam hal ekonomi, sosial dan lingkungan.
4. Memberi sumbangan yang bermakna bagi kesejahteraan bangsa.

2.4. Struktur Organisasi



Gambar 2.2 Struktur Perusahaan

[Halaman ini sengaja dikosongkan]

BAB III

TINJAUAN PUSTAKA

3.1. Pembelajaran Mesin

Pembelajaran mesin (machine learning) adalah cabang dari kecerdasan buatan (artificial intelligence atau AI) yang berfokus pada pengembangan sistem yang dapat belajar dan meningkatkan performa mereka berdasarkan pengalaman atau data tanpa perlu diprogram secara eksplisit. Dalam pembelajaran mesin, algoritma digunakan untuk menganalisis data, mengenali pola, dan membuat keputusan atau prediksi berdasarkan data tersebut. Proses ini seringkali dilakukan dengan memanfaatkan model statistik dan teknik komputasi yang kompleks.

Menurut Microsoft Azure, platform pembelajaran mesin mencakup alat, framework, dan infrastruktur yang memungkinkan para pengembang, data scientist, dan profesional lainnya untuk membangun, melatih, serta menyebarkan model machine learning secara efisien. Platform ini sering kali dirancang untuk mendukung siklus hidup pembelajaran mesin, mulai dari pengumpulan data, pra-pemrosesan, pelatihan model, hingga pengelolaan model yang sudah diterapkan.

Berdasarkan artikel di Dicoding, machine learning dapat diklasifikasikan menjadi tiga jenis utama:

1. Pembelajaran Terawasi

Metode pembelajaran terawasi atau *supervised learning* menggunakan data yang telah diberi label untuk melatih model. Algoritma belajar dari data input dan output yang telah ditentukan untuk memprediksi atau mengklasifikasikan data baru. Contohnya adalah prediksi harga rumah berdasarkan data historis.

2. Pembelajaran Tak Terawasi

Pada metode tak terawasi atau *unsupervised learning*, algoritma tidak diberi label data. Sistem bekerja untuk mengelompokkan data berdasarkan pola atau hubungan yang tidak diketahui sebelumnya. Contohnya adalah segmentasi pelanggan berdasarkan pola pembelian mereka.

3. Pembelajaran Penguatan

Metode pembelajaran penguatan atau *reinforcement learning* melibatkan agen (*agent*) yang berinteraksi dengan lingkungannya untuk mengambil tindakan tertentu. Agen ini menerima imbalan (*reward*) atau hukuman (*penalty*) berdasarkan tindakan yang dilakukan. Contohnya adalah robot yang belajar bergerak melalui serangkaian percobaan dan umpan balik.

Dalam perkembangannya, pembelajaran mesin telah diterapkan pada berbagai bidang seperti pengenalan suara, analisis gambar, sistem rekomendasi, deteksi anomali, dan prediksi bisnis. Dengan semakin banyaknya data yang tersedia dan peningkatan daya komputasi, pembelajaran mesin menjadi salah satu teknologi inti yang mendorong transformasi digital di era modern.

3.2. Python

Python adalah bahasa pemrograman tingkat tinggi yang dikembangkan oleh Guido van Rossum dan pertama kali dirilis pada 20 Februari 1991. Nama "Python" terinspirasi dari acara komedi Inggris "Monty Python's Flying Circus".

Python dikenal karena sintaksisnya yang sederhana dan mudah dipahami, menjadikannya pilihan populer untuk pemrograman baik bagi pemula maupun profesional. Bahasa ini mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek dan pemrograman prosedural. Selain itu, Python bersifat open-source, memungkinkan komunitas global untuk berkontribusi dalam pengembangannya.

Sejak peluncuran awalnya, Python telah mengalami beberapa evolusi signifikan, dengan rilis utama seperti Python 1.0 pada

tahun 1994, Python 2.0 pada tahun 2000, dan Python 3.0 pada tahun 2008. Perlu dicatat bahwa Python 3.0 tidak kompatibel dengan versi sebelumnya, yang menandai perubahan besar dalam desain bahasa ini.

Saat ini, Python digunakan secara luas dalam berbagai aplikasi, termasuk pengembangan web, analisis data, kecerdasan buatan, dan komputasi ilmiah, mencerminkan fleksibilitas dan kekuatan bahasa ini dalam berbagai domain.

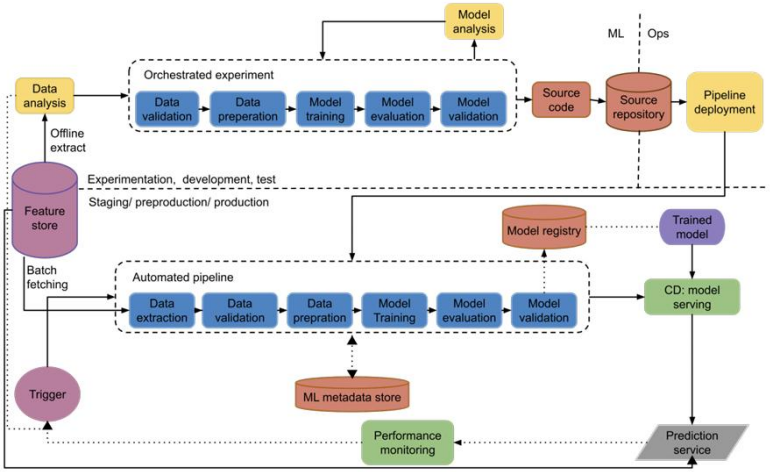
3.3. MLOps

Machine Learning Operations (MLOps) adalah pendekatan sistematis untuk mengotomatisasi dan mengoperasikan produk pembelajaran mesin (ML) secara *end-to-end*. MLOps mengintegrasikan prinsip-prinsip dari pembelajaran mesin, rekayasa perangkat lunak, dan praktik DevOps untuk menjembatani kesenjangan antara pengembangan (Dev) dan operasionalisasi (Ops). Tujuan utama MLOps adalah memastikan model pembelajaran mesin dapat diterapkan di lingkungan produksi secara andal dan berkelanjutan, memungkinkan siklus hidup produk ML yang lebih efisien dan terukur.

MLOps didasarkan pada sembilan prinsip utama: otomatisasi CI/CD (*Continuous Integration and Continuous Delivery/Deployment),* orkestrasi alur kerja, reproducibility, *versioning,* kolaborasi, pelatihan dan evaluasi ML berkelanjutan, pelacakan dan pencatatan metadata ML, pemantauan berkelanjutan, serta *loop* umpan balik. Prinsip-prinsip ini dirancang untuk meningkatkan efisiensi dan kualitas siklus hidup pembelajaran mesin, dari pengembangan hingga deployment dan pemantauan model di lingkungan *production.*

Selain prinsip, MLOps melibatkan beberapa komponen teknis utama seperti repositori kode sumber, infrastruktur pelatihan model, *feature store*, *registry* model, dan komponen pemantauan. MLOps juga mengutamakan pengelolaan metadata dan versi model untuk memastikan *traceability* dan akurasi eksperimen di

seluruh iterasi. Komponen-komponen ini diintegrasikan dalam arsitektur *end-to-end* yang menggabungkan otomasi CI/CD dan orkestrasi *pipeline* pembelajaran mesin untuk mendukung proses yang kompleks dan saling terkait.



Gambar 3.1 Alur Kerja Penerapan MLOps

Implementasi MLOps membutuhkan keterlibatan peran multidisiplin seperti *data scientist*, *data engineer*, *software engineer*, dan *DevOps engineer*. Kerja sama antar peran ini sangat penting untuk merancang, mengelola, dan mengoperasikan sistem ML yang efektif di lingkungan produksi. Dengan pendekatan MLOps, organisasi dapat mengatasi tantangan dalam membawa *proof-of-concept* ke *production*, meningkatkan keandalan model, serta mendukung inovasi berbasis data yang lebih cepat dan efisien

3.4. MLflow

Mlflow adalah platform *open-source* yang dirancang untuk mengelola siklus hidup pembelajaran mesin (ML) secara menyeluruh. Platform ini menyediakan alat untuk memfasilitasi

proses persiapan data, pelatihan model, dan deployment dalam skala besar.

Mlflow terdiri dari empat komponen utama:

1. Tracking

   Menyediakan API dan antarmuka pengguna (UI) untuk mencatat parameter, versi kode, metrik, dan artefak selama proses pelatihan model. Hal ini memungkinkan pengguna untuk membandingkan berbagai eksperimen dan melacak hasilnya.

2. Projects

   Menyediakan format standar untuk mengemas kode data sains agar dapat digunakan kembali dan direproduksi dengan mudah. Setiap proyek dikemas dalam direktori dengan kode atau repositori Git, menggunakan file deskriptor untuk menentukan dependensi dan cara menjalankan kode tersebut.

3. Models

   Menawarkan format standar untuk mengemas model ML sehingga dapat digunakan dalam berbagai alat hilir, seperti penyajian real-time melalui REST API atau inferensi batch di Apache Spark. Format ini mendefinisikan konvensi yang memungkinkan penyimpanan model dalam berbagai "flavor" yang dapat dipahami oleh alat yang berbeda.

4. Model Registry

   Menyediakan penyimpanan terpusat untuk mengelola model ML, termasuk versi, tahap pengembangan, dan anotasinya. Fitur ini memfasilitasi kolaborasi tim dalam mengelola siklus hidup model, mulai dari *development* hingga *production*.

Dengan mengintegrasikan komponen-komponen ini, MLflow membantu praktisi ML dalam mengatasi tantangan yang terkait dengan reproduksibilitas eksperimen, pengelolaan dependensi, dan deployment model. Platform ini dirancang untuk bekerja dengan berbagai pustaka ML dan alat yang ada, sehingga memudahkan integrasi ke dalam alur kerja yang sudah berjalan.

3.5. Databricks

Databricks adalah platform data dan kecerdasan artifisial yang menggabungkan kemampuan data lake dan data warehouse dalam arsitektur yang disebut "lakehouse". Konsep lakehouse memungkinkan penyimpanan data terstruktur dan tidak terstruktur dalam satu sistem terpadu, sehingga mendukung analitik bisnis dan beban kerja pembelajaran mesin secara efisien.

Platform Databricks dirancang untuk berjalan di berbagai layanan cloud terkemuka, termasuk Amazon Web Services (AWS), Microsoft Azure, dan Google Cloud Platform (GCP). Integrasi ini memungkinkan organisasi memanfaatkan infrastruktur cloud pilihan mereka sambil tetap menggunakan fitur-fitur canggih yang ditawarkan oleh Databricks.

Beberapa fitur utama yang tersedia dalam platform Databricks meliputi:

1. Delta Lake

Proyek *open-source* yang meningkatkan keandalan data lakes dengan menyediakan transaksi ACID, penanganan skema, dan manajemen data yang lebih baik.

2. Apache Spark

Framework komputasi terdistribusi yang memungkinkan pemrosesan data dalam skala besar dengan performa tinggi.

3. *Managed* Mlflow

Alat *open-source* untuk mengelola siklus hidup pembelajaran mesin, termasuk pelacakan eksperimen, pengemasan model, dan deployment.

4. Unity Catalog

Unity Catalog adalah sistem manajemen data terpusat untuk tata kelola data yang lebih aman dan terstruktur. Konsep-konsep utama dalam Unity Catalog meliputi:

- Catalogs: Entitas tingkat atas untuk mengelompokkan sumber data berdasarkan domain atau departemen
- Schemas: Wadah di dalam catalog untuk mengatur tabel, tampilan (*views*), dan fungsi terkait.
- Tables: Representasi data terstruktur dalam bentuk baris dan kolom. Tabel ini dapat berupa managed table (data dikelola oleh Unity Catalog) atau external table (data disimpan di lokasi eksternal seperti Amazon S3).
- Volumes: Penyimpanan untuk file tidak terstruktur (misalnya, JSON, Parquet). Volume memungkinkan analisis data tidak terstruktur dengan standar tata kelola yang sama.
- Models: Unity Catalog mendukung pengelolaan model pembelajaran mesin yang terintegrasi dengan Managed MLflow. Model dapat disimpan, dipantau, dan dikelola bersama data sumbernya.

5. Workflows, Jobs, dan Scheduling

Fitur ini mendukung automasi dan orkestrasi proses data engineering, analitik, dan pembelajaran mesin. Workflows adalah alat orkestrasi untuk mengelola pipeline data atau alur kerja pembelajaran mesin. Jobs memungkinkan pengguna menjalankan task atau notebook secara terjadwal atau berdasarkan permintaan. Scheduling mendukung automasi penuh dengan penjadwalan berbasis waktu atau *event*.

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem

Pada bab ini akan dijelaskan analisis kebutuhan dan evaluasi terhadap sistem machine learning yang ada saat ini, termasuk tantangan yang dihadapi dalam pengelolaan siklus hidup machine learning (*Machine Learning Life Cycle*). Analisis ini mencakup pemetaan kebutuhan untuk membangun workflow dan arsitektur pipeline standarisasi yang dapat digunakan pada berbagai proyek machine learning.

4.1.1. Definisi Umum

Secara umum, sistem yang dirancang merupakan template standar untuk proyek machine learning yang mencakup seluruh siklus hidup machine learning. Template ini bertujuan untuk meningkatkan efisiensi, reproduktibilitas, dan skalabilitas melalui penerapan prinsip MLOps. Sistem ini dirancang untuk mendukung proses mulai dari pengumpulan data, preprocessing, hingga pelatihan model, dengan fokus pada integrasi *experiment tracking* dalam alur kerja pengembangan.

4.1.2. Analisis Kebutuhan

Dalam membangun pipeline standarisasi berbasis MLOps, terdapat beberapa kebutuhan yang diidentifikasi berdasarkan evaluasi sistem yang ada dan tantangan yang dihadapi oleh tim data science:

1. Reproduksibilitas Eksperimen

Sistem harus mendukung pencatatan detail setiap eksperimen, termasuk parameter model, versi data, hasil metrik evaluasi, dan kode yang digunakan. Hal ini penting untuk memastikan eksperimen dapat diulang dengan hasil yang konsisten.

2. Manajemen Versi

Setiap elemen dalam siklus machine learning, seperti data, model, dan kode, harus memiliki kontrol versi yang terintegrasi untuk mendukung audit dan pengembangan yang terkoordinasi.

3. Kolaborasi Tim

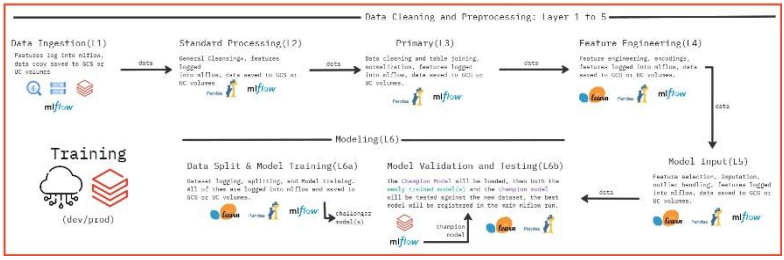
Sistem harus memungkinkan kolaborasi antara anggota tim dengan `menyediakan akses yang mudah ke eksperimen, data, dan model. Selain itu, komunikasi antar peran seperti data scientist, data engineer, dan DevOps engineer harus difasilitasi.

4.2. Perancangan Sistem

Pada bagian ini, perancangan arsitektur pipeline dan workflow MLOps yang terstandar akan dijelaskan. Perancangan ini bertujuan untuk menyediakan kerangka kerja yang dapat digunakan oleh tim data science dalam berbagai proyek machine learning.

4.2.1. Desain Alur Kerja *Pipeline Machine Learning*

Desain alur kerja *pipeline machine learning* yang diusulkan berfokus pada pipeline standarisasi untuk siklus hidup pembelajaran mesin yang terdiri dari beberapa lapisan (*layers*) yang dirancang secara modular.



Gambar 4.1 Alur Kerja *Pipeline Machine Learning*

Setiap lapisan memiliki fungsi spesifik untuk memastikan proses berjalan secara sistematis dan dapat diulang. Adapun alur dari pipeline yang telah dirancang adalah sebagai berikut:

1. *Layer* 0: Preraw

Lapisan ini digunakan untuk memastikan bahwa data berada dalam format tabular yang sesuai. Proses ini opsional, tergantung pada format awal data yang tersedia.

2. *Layer* 1: Raw

Pada lapisan ini, data mentah dimuat dan diproses untuk memastikan dapat digunakan pada tahap berikutnya.

3. *Layer* 2: Intermediate

Data dibersihkan dan diproses secara standar, termasuk normalisasi tipe data, penghapusan nilai kosong, dan transformasi awal lainnya.

4. *Layer* 3: Primary

Tahap ini berfokus pada penggabungan (*linking*) data dari berbagai sumber dan normalisasi lebih lanjut untuk menghasilkan dataset yang lebih terstruktur.

5. *Layer* 4: Feature Engineering

Pada lapisan ini, dilakukan rekayasa fitur untuk menghasilkan atribut yang relevan bagi model machine learning.

6. *Layer* 5: Model Input

Data dipersiapkan untuk menjadi input model, termasuk proses pembagian data untuk *training*, validasi, dan *testing*.

7. *Layer* 6: Models

Model dilatih, di-tuning, dan disimpan pada lapisan ini. Selain itu, fungsi untuk pembagian dataset dan evaluasi skor model juga dikelola di sini.

Dari konsep layering yang telah dirancang, *pipeline* berjalan dalam satu mode utama, yaitu mode training, yang difokuskan pada proses pelatihan model.

1. Mode Training

Dalam mode ini, pipeline berjalan dari Layer 1 hingga Layer 6, mencakup seluruh proses mulai dari pemrosesan data mentah hingga pelatihan dan penyimpanan model. Selama mode ini, *experiment tracking* diintegrasikan untuk mencatat seluruh informasi eksperimen, termasuk parameter model, versi data, hasil evaluasi, dan kode yang digunakan. Hal ini memungkinkan data scientist untuk mengevaluasi performa model secara komprehensif dan memilih model terbaik berdasarkan metrik yang dicatat.

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang kami buat. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi pipeline *machine learning* dan implementasi *workflow CI/CD* MLOps.

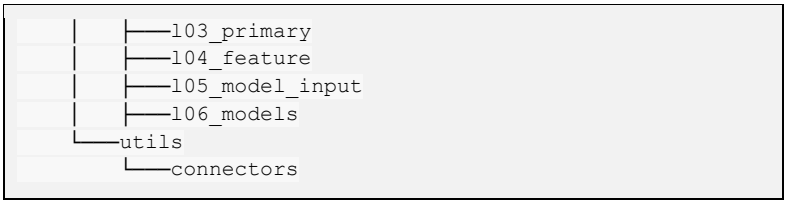
5.1. Implementasi Pipeline Machine Learning

Implementasi pipeline machine learning dilakukan dengan end-to-end, mulai dari pembangunan struktur proyek secara keseluruhan, hingga implementasi dari setiap *layer pipeline*.

5.1.1. Struktur Proyek

Proyek ini dibangun dengan pendekatan modular, di mana setiap komponen dikelompokkan berdasarkan fungsi dan fase siklus hidup data. Struktur direktori terdiri dari beberapa elemen utama sebagai berikut:

```
Project:.
├───config
│   └───keys
├───data
│   ├───1_raw
│   ├───2_intermediate
│   ├───3_primary
│   ├───4_feature
│   ├───5_model_input
│   ├───6_model
│   └───tmp
├───log
├───notebooks
├───resources
└───src
    ├───library
    ├───pipeline
    │   ├───100_preraw
    │   ├───101_raw
    │   ├───102_intermediate
```



Gambar 5.1 Pohon Direktori Proyek

Struktur direktori pada gambar 5.1 terdiri dari beberapa elemen utama sebagai berikut:

1. Direktori Utama

Berisi berkas *requirements* untuk mengelola *dependency* pengembangan.

2. Direktori *config*

Menyimpan berkas konfigurasi utama dari proyek yang disimpan dalam format *YAML*, berkas konfigurasi data yang berisi informasi lokasi data atau *query* data yang hendak digunakan.

3. Direktori *data*

Berisi delapan lapisan data, mulai dari *raw* (lapisan pertama) hingga *models* (lapisan keenam). Data pada setiap lapisan diproses sesuai dengan fungsinya, misalnya *raw data* disimpan di *1_raw*, sementara data yang telah siap untuk pelatihan model disimpan di *5_model_input*.

4. Direktori *src*

Menyediakan skrip utama yaitu *main_train.py*, serta modul-modul pendukung dalam subdirektori library, pipeline, dan utils. Subdirektori pipeline terbagi lagi menjadi enam lapisan (L01 hingga L06), yang masing-masing mewakili satu tahap dalam pipeline *machine learning*.

5. Direktori *log*

Berisi *log* (catatan) untuk setiap lapisan dalam pipeline, membantu dalam melacak proses dan keluaran pada setiap tahap.

6. Direktori *notebooks*

Menyimpan *notebook* Jupyter untuk analisis eksplorasi, pengujian, atau pengembangan prototipe model.

Struktur ini dirancang untuk memfasilitasi pemisahan tugas, meningkatkan keterbacaan kode, dan mendukung penerapan prinsip MLOps secara sistematis.

5.1.2. *Script* Utama dalam Implementasi Pipeline

Terdapat satu *script* utama yang akan menjalankan seluruh proyek machine learning dari awal hingga akhir, yaitu *main_train.py*. Script ini akan memanggil fungsi-fungsi dari *script pipeline* utama yang tersimpan pada *pipeline.py*.

A. Script *main_train.py* dan *main_inference.py*

Script main_train.py memiliki struktur logika yang dirancang untuk menginisialisasi konfigurasi, membuat koneksi, mengatur MLflow, dan menjalankan pipeline yang telah dirancang dalam mode *training*. Script ini memanggil fungsi run_pipeline dari pipeline.py, yang akan mengeksekusi alur kerja mulai dari pemrosesan data hingga pelatihan dan penyimpanan model.

```
1  load_project_to_root_path()
2
3  def main():
4      # Load configuration
5      config = load_config()
6
7      # Determine ingestion method and initialize required
connection(s)
8      connections = create_connection(config)
9
10     # Configure MLflow
11     config_mlflow(config)
12
```

```
    # Run the pipeline, passing in config and connections
    run_pipeline(config, "training", **connections)
```

Pseudocode 5.1 *Script Utama*

Pada pseudocode 5.1, langkah-langkah utamanya meliputi:

1. Inisiasi Project: Fungsi *load_project_to_rooth_path* menambahkan *root path* proyek ke variabel *sys.path* agar seluruh modul pada proyek dapat diakses dengan mudah.
2. Inisiasi Konfigurasi: Fungsi *load_config* memuat parameter dari *file* konfigurasi *YAML* untuk digunakan di seluruh pipeline.
3. Pembuatan Koneksi: Fungsi *create_connection* menginisiasi koneksi ke layanan luar seperti Google Cloud Storage dan Google BigQuery berdasarkan konfigurasi yang telah dimuat.
4. Konfigurasi MLflow: Fungsi *config_mlflow* memastikan MLflow siap untuk mencatat eksperimen dan model, baik dalam mode lokal maupun pada *server* terpisah seperti Databricks.
5. Eksekusi Pipeline: Fungsi *run_pipeline* menjalankan *pipeline* sesuai dengan mode yang ditentukan (*"training"* atau *"inference"*).

Berikut adalah fungsi-fungsi pendukung yang digunakan pada *script* utama

```
def load_project_to_root_path():
```

```
    current_dir = os.getcwd()
    project_root = os.path.abspath(os.path.join(current_dir,
"../"))
    sys.path.append(project_root)
```

Pseudocode 5.2 Fungsi *load_project_to_root_path*

```
def load_config(config_path: str = "../config/config.yaml"):
    # Baca file konfigurasi YAML
    with open(config_path, 'r') as file:
        config = yaml.safe_load(file)

    return config
```

Pseudocode 5.3 Fungsi *load_config*

```
def create_connection(config):
    # Inisialisasi dict untuk koneksi
    connections = {}

    # Koneksi ke GCS jika diaktifkan
    if config.get("connections", {}).get("gcs",
{}).get("enabled", False):
        gcs_config = config["connections"]["gcs"]
        if not gcs_config.get("service_account_key_path"):
            raise ValueError("GCS service account key path
is required.")
        connections["gcs_client"] =
create_gcs_client(gcs_config)
        print("GCS connection established.")

    # Koneksi ke BigQuery jika diaktifkan
    if config.get("connections", {}).get("bq",
{}).get("enabled", False):
        bigquery_config = config["connections"]["bq"]
        if not
bigquery_config.get("service_account_key_path"):
            raise ValueError("BigQuery service account key
path is required.")
        connections["bq_client"] =
create_bq_client(bigquery_config)
        print("BigQuery connection established.")

    return connections
```

Pseudocode 5.4 Fungsi *create_connection*

```
def config_mlflow(config):
```

```
    # Ambil konfigurasi MLflow
    mlflow_config = config["mlflow"]
    mode = mlflow_config["mode"]

    if mode == "local":
mlflow.set_tracking_uri(mlflow_config["local"]["mlflow_tracking
_uri"])
mlflow.set_experiment(mlflow_config["local"]["experiment_name"]
)

    elif mode == "databricks":
        os.environ["MLFLOW_TRACKING_URI"] =
mlflow_config["databricks"]["mlflow_tracking_uri"]
        os.environ["DATABRICKS_HOST"] =
mlflow_config["databricks"]["databricks_host"]
        os.environ["DATABRICKS_TOKEN"] =
mlflow_config["databricks"]["databricks_token"]
        mlflow.set_tracking_uri("databricks")
mlflow.set_experiment(mlflow_config["databricks"]["experiment_p
ath"])

    else:
        raise ValueError("Unsupported MLflow mode")

    print(f"MLflow configured in {mode} mode.")
```

Pseudocode 5.5 Fungsi *config_mlflow*

B. Script *pipeline.py*

Script *pipeline.py* berfungsi sebagai pusat koordinasi dalam menjalankan seluruh *pipeline*. Script ini memanggil *layer-layer pipeline* secara berurutan, memastikan alur data berjalan dengan baik dari awal hingga akhir.

```
def run_pipeline(config, mode, **connections):
    # Mulai run utama MLflow untuk melacak seluruh pipeline
    with start_mlflow_run("main"):
        # Bersihkan file YAML untuk memulai dari awal
        clear_yaml_files(mode)

        # Jalankan proses data (Layer 1 hingga 5)
        raw = run_layer_01(config, mode, **connections)
        general = run_layer_02(raw, mode)
        primary = run_layer_03(general, mode)
```

```
        features = run_layer_04(primary, mode)
        model_input = run_layer_05(features, mode)

        if mode == "training":
            print("Running in training mode...")

            # Latih model Challenger
            trained_models, run_ids, params, input_example,
x_test, y_test = run_layer_06(model_input)

            # Coba muat model Champion
            champion_model = None
            try:
                champion_model =
load_model_from_mlflow_alias(config["champion_model"],
alias="champion")
                print("Champion model loaded.")
            except Exception as e:
                print(f"Champion model not found. Proceeding
with Challenger models only. Error: {e}")

            # Skor model Challenger (dan Champion jika ada)
            run_layer_06b(trained_models, run_ids, params,
input_example, x_test, y_test, champion_model=champion_model)

        else:
            raise ValueError("Invalid mode. Choose
'training'.")
```

Pseudocode 5.6 Script Pipeline Utama

Langkah utama dalam pseudocode 5.6 ini adalah:

1. Inisialisasi MLflow: Dengan *start_mlflow_run*, setiap pipeline dilacak dalam satu run utama di MLflow.
2. Pemrosesan Data: Dimulai dari *layer* pertama (*run_layer_01*) hingga *layer* kelima (*run_layer_05*), *pipeline* memproses data secara bertahap.
3. Mode Pelatihan: Jika mode *"training"* dipilih, *pipeline* melatih model dan mencatat model yang dihasilkan serta metrik terkait.

4. Model Champion: Setelah pelatihan, *pipeline* mencoba memuat model champion dari MLflow, jika tersedia, untuk digunakan dalam evaluasi.
5. Penyimpanan dan Evaluasi: Model yang dilatih dievaluasi, dan hasilnya dicatat dalam MLflow, bersama dengan parameter dan metrik evaluasi dari eksperimen.

5.1.3. Implementasi *Layer* 1: *Raw* Data *Ingestion*

Layer 1 dalam *pipeline machine learning* bertujuan untuk melakukan proses ingestion data mentah dari berbagai sumber data eksternal seperti Databricks, Google Cloud Storage (GCS), dan BigQuery (BQ). Data yang diingest akan disimpan dalam format parquet pada direktori */data/1_raw* untuk memastikan integritas dan kompatibilitas dengan *layer* berikutnya. *Layer* ini menjadi fondasi bagi seluruh *pipeline*, data mentah yang telah dingiest akan diproses lebih lanjut dalam tahap berikutnya.

Alur kerja *layer* 1 mencakup langkah-langkah berikut:

1. Inisiasi Volume dan Konfigurasi

Menggunakan fungsi *load_data_config* untuk memuat konfigurasi data dari file *YAML*, yang mencakup informasi tentang sumber data, query, dan detail koneksi.

2. Ingestion Data dari Sumber Eksternal

a. Databricks: Menggunakan SparkSession untuk menjalankan query SQL pada tabel Databricks dan menyimpan hasilnya dalam format parquet.

b. Google Cloud Storage: Mengunduh file dari bucket GCS, mengonversinya ke format parquet, dan menyimpannya di direktori */data/1_raw*.

c. BigQuery: Menjalankan query pada tabel BigQuery, mengambil hasil dalam bentuk DataFrame, dan menyimpannya sebagai file parquet.

3. Logging dan Dokumentasi

File yang diingest dicatat ke MLflow menggunakan fungsi log_features dan log_artifact. Hal ini memungkinkan pelacakan sumber dan atribut data mentah.

Berikut adalah implementasi pseudocode beserta fungsi-fungsi pendukungnya pada *layer* 1.

A. Fungsi Utama *Layer* 1

```
def run_layer_01(config, mode, **connections):
    # Muat konfigurasi data dan tentukan volume berdasarkan mode
    data = load_data_config(mode)
    volume = get_volume(mode)

    # Lakukan ingest data dari Databricks jika diaktifkan
    if config["connections"].get("databricks", {}).get("ingest", False):
        spark = get_spark_session()
        databricks_ingest(spark, data["databricks"], volume)

    # Lakukan ingest data dari GCS jika diaktifkan dan koneksi tersedia
    if config["connections"].get("gcs", {}).get("ingest", False) and "gcs_client" in connections:
        gcs_ingest(connections["gcs_client"], data["gcs"], volume)

    # Lakukan ingest data dari BigQuery jika diaktifkan dan koneksi tersedia
    if config["connections"].get("bq", {}).get("ingest", False) and "bq_client" in connections:
        bq_ingest(connections["bq_client"], data["bq"], volume)

    # Muat semua file mentah ke dalam dataframe dan log ke MLflow
    raw_file_paths = get_all_raw_files(volume)
    raw_dataframes = load_files_to_dict(raw_file_paths)
    logged_features = log_features(raw_dataframes, "1_raw", mode)
    log_artifact(logged_features[0], logged_features[1])

    return raw_file_paths
```

Pseudocode 5.7 Fungsi Utama *Layer* 1

Fungsi *run_layer_01* bertanggung jawab untuk mengatur seluruh proses ingestion dalam *layer* 1.

B. Fungsi Ingestion Data

```
1  def databricks_ingest(spark, databricks_data, volume):
2      # Tentukan direktori untuk menyimpan data
3      data_dir = os.path.join(volume, "data/1_raw")
4      os.makedirs(data_dir, exist_ok=True)
5
6      for table_name, config in databricks_data.items():
7          query = config.get("query")
8          if not query:
9              print(f"No query for table `{table_name}`.
Skipping.")
10             continue
11
12         try:
13             # Eksekusi query dan load data ke dalam
dataframe
14             df = spark.sql(query)
15
16             # Periksa apakah data tersedia
17             if df.isEmpty():
18                 raise ValueError(f"No data returned for
`{table_name}`.")
19
20             # Tentukan path file untuk menyimpan data
21             raw_data_path = os.path.join(data_dir,
f"{table_name}.parquet")
22             os.makedirs(os.path.dirname(raw_data_path),
exist_ok=True)
23
24             # Simpan data sebagai file Parquet
25             df = df.toPandas()
26             df.to_parquet(raw_data_path, index=False)
27             print(f"Data for `{table_name}` saved to
{raw_data_path}")
28
29         except Exception as e:
30             print(f"Error during Databricks ingestion for
`{table_name}`: {e}")
```

Pseudocode 5.8 Fungsi Ingestion Databricks

Fungsi *databricks_ingest* menggunakan Spark untuk menjalankan *query SQL* pada tabel Databricks. Hasilnya disimpan dalam direktori */data/1_raw* sebagai *file* parquet. Fungsi ini hanya dapat digunakan apabila kode berjalan pada lingkungan Databricks, karena *query* Spark *SQL* terhadap tabel yang tersimpan di Databricks tidak bisa dilakukan dari luar lingkungan Databricks.

```
1  def gcs_ingest(client, files, volume):
2      # Tentukan direktori data
3      data_dir = os.path.join(volume, "data")
4
5      for label, (bucket_name, source_blob_name) in
files.items():
6          bucket = client.bucket(bucket_name)
7          blob = bucket.blob(source_blob_name)
8
9          # Tentukan nama file dan path penyimpanan
10         file_name = os.path.basename(source_blob_name)
11         file_name = camel_to_snake_case(file_name)
12         temp_file_path = os.path.join(data_dir, "tmp",
file_name)
13         raw_data_path = os.path.join(data_dir, "1_raw",
f"{os.path.splitext(file_name)[0]}.parquet")
14
15         # Pastikan direktori ada
16         os.makedirs(os.path.dirname(temp_file_path),
exist_ok=True)
17         os.makedirs(os.path.dirname(raw_data_path),
exist_ok=True)
18
19         # Download file sementara
20         blob.download_to_filename(temp_file_path)
21
22         # Tentukan format file dan baca data
23         file_extension =
os.path.splitext(file_name)[1].lower()
24         if file_extension == '.csv':
25             df = pd.read_csv(temp_file_path)
26         elif file_extension in ['.xls', '.xlsx']:
27             df = pd.read_excel(temp_file_path)
28         elif file_extension == '.parquet':
29             df = pd.read_parquet(temp_file_path)
30         else:
31             print(f"Unsupported file format for {file_name}.
Skipping.")
32             os.remove(temp_file_path)
```

```
            continue

        # Simpan data sebagai file parquet
        df.to_parquet(raw_data_path, index=False)

        # Hapus file sementara
        os.remove(temp_file_path)
        print(f"File {file_name} from GCS saved to
{raw_data_path}")
```

Pseudocode 5.9 Fungsi Ingestion Google Cloud Storage

Fungsi *gcs_ingest* mengunduh *file* dari Google Cloud Storage, mengkonversinya ke bentuk parquet, dan menyimpannya di direktori */data/1_raw*.

```
def bq_ingest(bq_client, bq_data, volume):
    data_dir = os.path.join(volume, "data/1_raw")

    for table_name, config in bq_data.items():
        query = config.get("query")
        if not query:
            print(f"No query found for table `{table_name}`.
Skipping.")
            continue

        try:
            # Jalankan query dan ambil data ke dalam
DataFrame
            print(f"Ingesting data for BigQuery table:
{table_name}")
            df = bq_client.query(query).to_dataframe()

            # Cek jika data kosong
            if df.empty:
                raise ValueError(f"The query for
`{table_name}` did not return any data.")

            # Tentukan path file untuk data parquet
            raw_data_path = os.path.join(data_dir,
f"{table_name}.parquet")
            os.makedirs(os.path.dirname(raw_data_path),
exist_ok=True)

            # Simpan data sebagai file parquet
            df.to_parquet(raw_data_path, index=False)
            print(f"Data for `{table_name}` saved to
{raw_data_path}")
```

```
26
27          except Exception as e:
28              print(f"Error during BigQuery ingestion for
`{table_name}`: {e}")
29              raise
```

Pseudocode 5.10 Fungsi Ingestion BigQuery

Fungsi *bq_ingest* mengeksekusi query pada tabel BigQuery, menyimpan hasilnya sebagai file parquet, dan mencatat error jika terjadi kegagalan.

C. Fungsi Pendukung *Layer* 1

```
1  def load_data_config(mode,
config_path="../config/data.yaml"):
2      with open(config_path, 'r') as file:
3          config = yaml.safe_load(file)
4      return config[mode]
5
6  def get_all_raw_files(volume):
7      raw_dir = os.path.join(volume, 'data/1_raw')
8      if not os.path.isdir(raw_dir):
9          raise FileNotFoundError(f"The directory {raw_dir}
does not exist.")
10     return [os.path.join(raw_dir, f) for f in
os.listdir(raw_dir) if f.endswith('.parquet')]
11
12 def get_spark_session(app_name="DatabricksIngestion"):
13     try:
14         spark = SparkSession.builder \
15             .appName(app_name) \
16             .getOrCreate()
17         print(f"Using SparkSession with app name:
{app_name}")
18         return spark
19     except Exception as e:
20         print(f"Error initializing SparkSession: {e}")
21         raise
```

Pseudocode 5.11 Fungsi Pendukung *Layer* 1

Fungsi *load_data_config* memuat konfigurasi YAML berdasarkan mode operasi. Fungsi *get_all_raw_files* mengambil daftar semua *file* parquet dari direktori */data/1_raw*. Fungsi *get_spark_session* membuat atau mengakses SparkSession aktif untuk mendukung *ingestion* dari Databricks.

5.1.4. Implementasi *Layer* 2: *General Data Cleansing*

Layer 2 dalam *pipeline machine learning* bertujuan untuk melakukan pembersihan data secara umum (general data cleansing) pada semua tabel yang diingest dari *layer* 1. Proses ini dirancang untuk memastikan data memiliki kualitas yang konsisten sebelum diproses lebih spesifik pada *layer* 3 dan seterusnya. Fungsi utama yang digunakan dalam *layer* 2 adalah *standard_processing*, yang dirancang agar dapat diterapkan ke semua tabel secara fleksibel dan seragam (*fit-to-all*).

Alur kerja *layer* 2 mencakup langkah-langkah berikut:

1. Memuat Data Mentah

Data mentah yang telah diingest pada *layer* 1 dimuat menggunakan fungsi *load_files_to_dict*, yang mengonversi file Parquet ke dalam bentuk dictionary berisi DataFrame.

2. Pembersihan Data

Data dari setiap tabel diproses menggunakan fungsi *standard_processing*. Tahapannya meliputi:

- Penyesuaian nama kolom agar seragam.
- Penghapusan nilai kosong sepenuhnya atau duplikat.
- Penanganan nilai string seperti mengubah huruf menjadi kecil dan menghapus spasi tambahan.
- Konversi tipe data sesuai dengan atribut yang disarankan.

3. Saran Atribut untuk Pemrosesan

Terdapat fungsi *suggest_attributes_for_processing* yang digunakan untuk menentukan atribut penting seperti:

- Tipe data setiap kolom
- Kolom *primary key*
- Kolom yang perlu diurutkan (*order_by*).

- Format tanggal untuk kolom bertipe datetime.

4. Penyimpanan dan Logging

Data yang telah diproses disimpan ke direktori */data/2_intermediate* menggunakan fungsi *save_dict_to_files*. Hasil pembersihan data juga dicatat ke MLflow menggunakan fungsi *log_features* dan *log_artifact*, sehingga setiap langkah dalam pipeline dapat dilacak.

Berikut adalah implementasi pseudocode beserta fungsi-fungsi pendukungnya pada *layer* 2.

A. Fungsi Utama *Layer* 2

```
1  def run_layer_02(raw_paths, mode):
2      # Load raw data files into a dictionary of DataFrames
3      data = load_files_to_dict(raw_paths)
4
5      # Apply specific processing logic to customer master
data
6      data["customer_master_mask"] =
cust_master_split(data["customer_master_mask"])
7
8      # Suggest attributes for processing
9      attributes_to_process =
suggest_attributes_for_processing(data)
10
11     # Initialize a dictionary for processed DataFrames
12     processed_data = {}
13     for table_name, df in data.items():
14         processed_data[table_name] = standard_processing(df,
attributes_to_process)
15
16     # Save processed data to the intermediate directory
17     save_dict_to_files(processed_data,
"data/2_intermediate", mode)
18
19     # Log processed data and features to MLflow
20     logged_features = log_features(processed_data,
"2_intermediate", mode)
21     log_artifact(*logged_features)
22
23     return processed_data
```

Pseudocode 5.12 Fungsi Utama *Layer* 2

B. Fungsi *Standard Processing*

```
1  def standard_processing(
2      data,
3      dtype=None,
4      col_names=None,
5      primary_keys=None,
6      order_by=None,
7      partition_key=None,
8      usecols=None,
9      dt_format=None,
10 ):
11     """Process raw data by applying basic cleaning steps."""
12
13     def _pk_should_have_null(df):
14         if primary_keys:
15             return df.dropna(subset=primary_keys,
how="any").reset_index(drop=True)
16         return df
17
18     if isinstance(data, dict):
19         data = pd.concat([df for _, df in
sorted(data.items())], axis=0, ignore_index=True)
20
21     partition_key = partition_key or ""
22
23     renamed_data = standardize_column_names(
24         data.drop(columns=[META_COL],
errors="ignore").drop_duplicates()
25     )
26     if col_names:
27         renamed_data = renamed_data.rename(
28             columns={str(k).lower(): v for k, v in
col_names.items()}
29         )
30
31     data_cols = [col for col in renamed_data.columns if col
not in {partition_key, META_COL}]
32     not_null_data = renamed_data.dropna(subset=data_cols,
how="all")
33
34     str_types = ["string", "object"]
35     trim_data = not_null_data.copy()
36     all_cols = [col for col in
trim_data.select_dtypes(include=str_types).columns if col !=
META_COL]
37     trim_data[all_cols] = (
38         trim_data[all_cols]
```

```
        .astype(str)
        .applymap(lambda x: x.strip().strip("'").strip('"'),
na_action="ignore")
        .applymap(lambda x: np.nan if x in ["", "nan"] else
x, na_action="ignore")
    )

    if usecols:
        trim_data = trim_data[usecols]
        all_cols = usecols

    typed_data = _pk_should_have_null(trim_data)

    if dtype:
        for col, d in dtype.items():
            if isinstance(d, str) and d in
{"datetime64[ns]"}:
                typed_data[col] = pd.to_datetime(
                    typed_data[col], errors="coerce",
format=dt_format.get(col) if dt_format else None
                )
            elif isinstance(d, str) and d.lower() in
{"float64", "int64"}:
                typed_data[col] =
pd.to_numeric(typed_data[col], errors="coerce")

        valid_dtype = {col: dtype[col] for col in dtype if
col in typed_data.columns}
        if valid_dtype:
            typed_data = typed_data.astype(valid_dtype)

        str_columns =
typed_data[all_cols].select_dtypes(include=str_types).columns
        typed_data[str_columns] =
typed_data[str_columns].applymap(lambda x: x.lower() if
pd.notna(x) else x)

    unique_data =
deduplicate(_pk_should_have_null(typed_data), primary_keys,
order_by)

    return unique_data
```

Pseudocode 5.13 Fungsi *standard_processing*

C. Fungsi Pendukung *Layer* 2

```
def _get_column_order(primary_keys, desc_by, all_columns):
    """Returns column order for sorting."""
```

```
3       other_columns = sorted(set(all_columns) -
set(primary_keys + desc_by))
4       return primary_keys + desc_by + other_columns
5
6   def deduplicate(data, primary_keys=None, desc_by=None):
7       """Deduplicate data based on primary keys and optional
descending columns."""
8       if primary_keys:
9           if not desc_by:
10              desc_by = []
11          all_columns = data.columns.to_list()
12          column_order = _get_column_order(primary_keys,
desc_by, all_columns)
13          not_null_data = data.dropna(subset=primary_keys)
14          unique_data =
not_null_data.sort_values(column_order,
ascending=False).drop_duplicates(subset=primary_keys,
keep="first")
15      else:
16          unique_data = data.drop_duplicates()
17
18      return unique_data.reset_index(drop=True)
19
20 def suggest_attributes_for_processing(data):
21      """
22      Suggest attributes (dtype, column names, etc.) based on
common patterns in the data.
23      """
24      if isinstance(data, dict):
25          data = pd.concat(data.values(), axis=0,
ignore_index=True)
26
27      attributes = {}
28
29      # Suggest dtype based on column data types
30      dtype = {col: ("float64" if
pd.api.types.is_float_dtype(data[col]) else "int64" if
pd.api.types.is_integer_dtype(data[col]) else "datetime64[ns]"
if pd.api.types.is_datetime64_any_dtype(data[col]) else
"string") for col in data.columns}
31      attributes["dtype"] = dtype
32
33      # Suggest column names by replacing spaces and capital
letters
34      col_names = {col: col.strip().lower().replace(" ", "_")
for col in data.columns if " " in col or col.isupper()}
35      attributes["col_names"] = col_names if col_names else
None
```

```
36
37      # Suggest primary keys (columns with 'id')
38      attributes["primary_keys"] = [col for col in
data.columns if "id" in col.lower()]
39
40      # Suggest order by columns (columns with 'date' or
'time')
41      attributes["order_by"] = [col for col in data.columns if
"date" in col.lower() or "time" in col.lower()]
42
43      # Suggest partition key (e.g., columns containing
'partition')
44      attributes["partition_key"] = next((col for col in
data.columns if "partition" in col.lower()), None)
45
46      # Keep all columns initially for usecols
47      attributes["usecols"] = list(data.columns)
48
49      # Suggest date formats for date-related columns
50      attributes["dt_format"] = {col: "%Y%m%d" for col in
data.columns if "date" in col.lower() and
pd.api.types.is_string_dtype(data[col])}
51
52      return attributes
```

Pseudocode 5.14 Fungsi Pendukung *Layer* 2

5.1.5. Implementasi Layer 3: *Primary Layer*

Layer 3 dalam *pipeline machine learning* bertujuan untuk menghasilkan tabel utama (*final table*) melalui proses pembersihan data (*data cleaning*), penggabungan tabel (*joining and linking*), serta transformasi data. Proses ini menjadi bagian penting dalam pipeline, di mana data dari *layer* 2 diproses lebih lanjut agar siap digunakan pada *layer* berikutnya atau langsung untuk analisis.

Layer 3 mulai bersifat *case-specific*, artinya operasi yang dilakukan tergantung pada kebutuhan dari setiap use-case. Dalam contoh yang diberikan, *layer* 3 difokuskan pada penggabungan dua tabel, yaitu *customer_master_mask* dan *prospect_base_table*, untuk menghasilkan tabel utama. Implementasi ini mencerminkan kebutuhan spesifik dari suatu studi kasus tertentu.

Alur kerja *Layer* 3 mencakup langkah-langkah berikut:

1. *Data Cleaning*

Melakukan pembersihan awal, termasuk penyesuaian nama kolom untuk menghindari konflik. Sebagai contoh, kolom *created_date* pada tabel *customer_master_mask* akan diubah menjadi *created_date_customer*, dan pada tabel *prospect_base_table* diubah menjadi *created_date_prospect*.

2. *Joining and Linking*

Menggabungkan tabel *customer_master_mask* dan *prospect_base_table* menggunakan kolom kunci *account_no* dari tabel pertama dan *account_number* dari tabel kedua. Proses penggabungan dilakukan dengan metode *inner join* untuk memastikan hanya data yang cocok yang dipertahankan.

3. *Data Transformation*

Tabel hasil penggabungan diproses lebih lanjut menggunakan fungsi *clean_transform*. Transformasi meliputi konversi tipe data, normalisasi nilai numerik, dan pembersihan nilai yang tidak valid.

4. *Output Data*

Data yang telah diproses disimpan dalam direktori */data/3_primary* untuk digunakan pada *layer* berikutnya. Hasil proses juga dicatat ke MLflow untuk memastikan pelacakan yang lengkap.

Berikut adalah implementasi pseudocode beserta fungsi-fungsi pendukungnya pada *layer* 3.

A. Fungsi Utama *Layer* 3

```
1  def run_layer_03(tables, mode):
2      if "customer_master_mask" not in tables or
"prospect_base_table" not in tables:
3          raise KeyError("Required tables are missing.")
4
5
tables["customer_master_mask"].rename(columns={'created_date':
'created_date_customer'}, inplace=True)
6
```

```
tables["prospect_base_table"].rename(columns={'created_date':
'created_date_prospect'}, inplace=True)
7
8       tables["final"] =
join_table(tables["customer_master_mask"],
tables["prospect_base_table"])
9       tables["final"] = clean_transform(tables["final"])
10
11      tables = {"final": tables["final"]}
12
13      save_dict_to_files(tables, "data/3_primary", mode)
14      logged_features = log_features(tables, "3_primary",
mode)
15      log_artifact(*logged_features)
16
17      return tables
```

Pseudocode 5. 15 Fungsi Utama Layer 3

B. Fungsi *Clean and Transform*

```
1   def clean_transform(train):
2       train = standardize_column_names(train)
3
4       for col in ['created_date_customer',
'created_date_prospect', 'estimate_deal_date',
'req_delivery_date', 'valid_from', 'valid_to']:
5           train[col] = pd.to_datetime(train[col],
dayfirst=True, errors='coerce')
6
7       for col in ['amount', 'amount_dp']:
8           train[col] = pd.to_numeric(train[col].replace({',':
''}, regex=True), errors='coerce')
9
10      return train
```

Pseudocode 5.16 Fungsi *Clean and Transform Layer* 3

C. Fungsi *Join Table*

```
1   def join_table(df1, df2):
2       return pd.merge(df1, df2, left_on="account_no",
right_on="account_number", how='inner')
```

Pseudocode 5. 17 Fungsi *Join Table Layer* 3

5.1.6. Implementasi *Layer* 4: *Feature* Engineering

Layer 4 dalam *pipeline machine learning* bertujuan untuk melakukan rekayasa fitur (*feature engineering*) yang memperkaya

data agar lebih informatif dan relevan untuk digunakan dalam pelatihan atau inferensi model. Proses rekayasa fitur meliputi pembuatan fitur baru, transformasi kolom, dan encoding variabel kategori.

Pada *layer* ini, fitur-fitur baru dihasilkan berdasarkan logika domain, seperti menghitung selisih waktu atau nilai tertentu, serta memastikan data terorganisir dengan baik untuk mendukung model. Selain itu, *layer* 4 bersifat *case-specific*, yang artinya implementasi disesuaikan dengan kebutuhan *use-case* tertentu.

Alur kerja *layer* 4 pada contoh studi kasus proyek ini mencakup langkah-langkah berikut:

1. Pembuatan Fitur Baru

   Terdapat fungsi *differences_features* yang digunakan untuk menambahkan kolom baru, seperti:

- *date_difference*: selisih hari antara *created_date_prospect* dan *created_date_customer*.
- *delivery_time_difference*: selisih hari antara *req_delivery_date* dan *estimate_deal_date*.
- *remaining_amount*: selisih antara kolom *amount* dan *amount_dp*.

2. Reorganisi Kolom

Kolom target, yaitu *prospect_status*, ditempatkan sebagai kolom terakhir untuk memastikan konsistensi format data.

3. Encoding Variabel Kategori

Terdapat fungsi *encode_categorical_columns* yang digunakan untuk mengubah variabel kategori menjadi angka melalui label *encoding*. Kolom-kolom seperti *prospect_owner*, *brand_temp*, dan *sector_mapping* diubah menjadi format numerik. Encoding juga dilakukan pada kolom target *prospect_status* untuk klasifikasi

biner. Nilai *"billing"* dikonversi menjadi 1 dan nilai lainnya menjadi 0.

4. Output Data

Data hasil rekayasa fitur disimpan di direktori */data/4_feature* dengan format parquet. Proses rekayasa fitur juga dilacak ke MLflow untuk pemantauan.

Berikut adalah implementasi *source code* beserta fungsi-fungsi pendukungnya pada *layer* 4.

A. Fungsi Utama *Layer* 4

```
def run_layer_04(table, mode):
    table["final"] = differences_features(table["final"])
    cols = [col for col in table["final"].columns if col != 'prospect_status'] + ['prospect_status']
    table["final"] = table["final"][cols]

    categorical_cols = ['prospect_owner', 'prospect_product_plant', 'user_detail_area', 'brand_temp',
                        'prospect_product_confidence_level', 'top', 'model', 'sector_mapping']
    table["final"] = encode_categorical_columns(table["final"], categorical_cols)

    table["final"]['prospect_status'] = table["final"]['prospect_status'].apply(lambda x: 1 if x == 'billing' else 0)

    save_dict_to_files(table, "data/4_feature", mode)
    logged_features = log_features(table, "4_feature", mode)
    log_artifact(*logged_features)

    return table
```

Pseudocode 5.18 Fungsi Utama Layer 4

B. Fungsi *differences_features*

```
def differences_features(df):
    df['date_difference'] = (df['created_date_prospect'] - df['created_date_customer']).dt.days
    df['delivery_time_difference'] = (df['req_delivery_date'] - df['estimate_deal_date']).dt.days
    df['remaining_amount'] = df['amount'] - df['amount_dp']
```

```
5       return df
```

Pseudocode 5. 19 Fungsi *difference_features Layer* 4

C. Fungsi *encoding*

```
1  def encode_categorical_columns(df, categorical_cols):
2      for col in categorical_cols:
3          le = LabelEncoder()
4          df[col] = le.fit_transform(df[col].astype(str))
5          print(f"Mapping untuk kolom '{col}':")
6          print(pd.DataFrame(list(zip(le.classes_,
le.transform(le.classes_))), columns=[col, 'Encoded_Value']))
7          print("\n" + "="*50 + "\n")
8      return df
```

Pseudocode 5.20 Fungsi *encoding Layer 4*

5.1.7. Implementasi *Layer* 5: *Model Input*

Layer 5 bertujuan untuk mempersiapkan data yang telah melalui proses feature engineering pada *layer* 4 agar siap digunakan sebagai input dalam pelatihan atau inferensi model machine learning.

Alur kerja *Layer* 5 pada contoh studi kasus proyek ini mencakup langkah-langkah berikut:

1. *Feature Selection*

Menghapus kolom yang tidak relevan atau redundan untuk meningkatkan interpretabilitas dan performa model.

2. *Imputation*

Mengisi nilai-nilai yang hilang, dalam contoh implementasinya nanti akan menggunakan metode *K-Nearest Neighbors (KNN)*.

3. *Outlier Handling*

Mengelola nilai-nilai ekstrem agar tidak mengganggu proses pelatihan model.

4. Penyimpanan dan Logging

Data hasil transformasi disimpan di direktori */data/5_model_input* dalam format Parquet. Data yang dihasilkan juga dilacak ke MLflow menggunakan fungsi *log_features* dan *log_artifact* untuk memastikan transparansi proses dan reproduksi hasil.

Berikut adalah implementasi *source code* beserta fungsi-fungsi pendukungnya pada *layer* 5.

A. Fungsi Utama *Layer* 5

```
def run_layer_05(table: Dict[str, pd.DataFrame], mode: str)
-> Dict[str, pd.DataFrame]:
    if "final" not in table:
        raise KeyError("The input table must contain a key
named 'final'.")

    df = table["final"]

    # Drop irrelevant columns
    df = drop_features(df, [
        'sector_usage', 'branch_division',
'last_modified_date',
        'incoterm', 'created_date_extract', 'account_no',
        'account_number', 'created_date_customer',
'created_date_prospect'
    ])

    # Drop datetime columns
    df = drop_features(df,
df.select_dtypes(include=['datetime64[ns]']).columns)

    # KNN imputation
    df = knn_impute(df)

    table["final"] = df

    # Save and log
    save_dict_to_files(table, "data/5_model_input", mode)
    log_artifact(*log_features(table, "5_model_input",
mode))

    return table
```

Pseudocode 5.21 Fungsi Utama *Layer* 5

B. Fungsi *Feature Selection*

```
1  def drop_features(df: pd.DataFrame, columns_to_drop:
List[str]) -> pd.DataFrame:
2      df.drop(columns=columns_to_drop, inplace=True,
errors='ignore')
3
df.drop(columns=df.select_dtypes(include=['datetime64[ns]']).co
lumns, inplace=True)
4      return df
```

Pseudocode 5.22 Fungsi *Feature Selection Layer* 5

C. Fungsi *Imputation*

```
1  def knn_impute_column(df, column, n_neighbors=5):
2      df[[column]] =
KNNImputer(n_neighbors=n_neighbors).fit_transform(df[[column]])
3      return df

4  def knn_impute(df: pd.DataFrame) -> pd.DataFrame:
5      for col in ['amount', 'amount_dp',
'delivery_time_difference', 'remaining_amount']:
6          df = knn_impute_column(df, col)
7      return df
```

Pseudocode 5.23 Fungsi *Imputation KNN Layer* 5

5.1.8. Implementasi Layer 6: Modelling

Layer 6 bertujuan untuk melakukan pelatihan, tuning, dan evaluasi model machine learning. Layer ini dirancang untuk mendukung pendekatan Champion-Challenger, di mana model baru yang dilatih (*Challenger*) dibandingkan dengan model yang sudah ada (*Champion*). Layer ini dibagi menjadi dua bagian utama:

1. Layer 6: Fokus pada pelatihan model, mencakup pembagian data, pelatihan model, dan logging ke MLflow.
2. Layer 6b: Fokus pada evaluasi model, pemilihan model terbaik, dan logging model terbaik untuk keperluan *deployment*.

Alur kerja *layer* 6 pada contoh studi kasus proyek ini mencakup langkah-langkah berikut:

1. *Data Splitting*

Data dibagi menjadi set pelatihan dan pengujian menggunakan sebuah fungsi Bernama *split_data*. Kolom target (*prospect_status*) dan kolom identifier *prospect_product_delivery_prospect_product_delivery_no* digunakan untuk membuat *truth table* dengan fungsi *generate_truth_table*.

2. *Model Training*

Tiga model akan dilatih, yaitu model Random Forest, XGBoost, dan LightGBM. *Hyperparameter* model di-*tuning* menggunakan GridSearchCV yang terintegrasi dengan Mlflow untuk pencatatan parameter dan metrik setiap eksperimen.

3. Data *Logging* dan Penyimpanan

Dataset pelatihan, pengujian, dan *truth table* disimpan dalam direktori */data/6_model*. Dataset juga dicatat ke MLflow menggunakan sebuah fungsi Bernama *log_dataset* untuk memastikan keterlacakan dan reproduksi eksperimen.

Sedangkan untuk alur kerja *layer* 6b pada contoh studi kasus proyek ini mencakup langkah-langkah berikut:

1. Model Evaluation

Model *Challenger* dievaluasi menggunakan fungsi score_model, yang menghitung metrik seperti *accuracy* dan *F1 score*. Jika tersedia, model *Champion* juga dievaluasi menggunakan data pengujian yang sama.

2. Model Selection

Model terbaik dipilih berdasarkan metrik F1 score. Model ini ditetapkan sebagai *Champion baru* jika memiliki kinerja yang lebih baik daripada *Champion* sebelumnya.

3. Model Logging dan *Saving*

Model terbaik dicatat ke MLflow menggunakan fungsi log_model, termasuk metrik, parameter, dan contoh input. Model

disimpan di direktori */data/6_model* dengan nama file yang mencerminkan jenis model dan statusnya sebagai model terbaik.

Berikut adalah implementasi source code beserta fungsi-fungsi pendukungnya pada *layer* 6.

A. Fungsi Utama *Layer* 6

```
1  def run_layer_06(table: Dict[str, pd.DataFrame]) ->
Tuple[Dict[str, BaseEstimator], Dict[str, str], Dict[str,
Dict[str, Any]], pd.DataFrame, pd.DataFrame, pd.Series]:
2      if "final" not in table:
3          raise KeyError("Missing 'final' key in input
table.")

4      df = table["final"]
5      target_column, identifier_column = "prospect_status",
"prospect_product_delivery_prospect_product_delivery_no"

6      # Log dataset and split data
7      log_dataset(df, get_volume("training") +
"/data/5_model_input/final.parquet", "final", target_column,
identifier_column)
8      x_train, x_test, y_train, y_test = split_data(df,
target_column)

9      # Generate truth table and save train/test data
10     save_dict_to_files({"test_truth_table":
generate_truth_table(x_test, y_test, identifier_column)},
"data/6_model", "training")
11     save_dict_to_files({"x_train":
x_train.drop(columns=identifier_column), "y_train": y_train,
"x_test": x_test.drop(columns=identifier_column), "y_test":
y_test}, "data/6_model", "training")

12     # Log datasets to MLflow
13     log_artifact(*log_features({"x_train": x_train,
"y_train": y_train, "x_test": x_test, "y_test": y_test},
"6_model", "training"))

14     # Define hyperparameters and train models
15     param_grids = {
16         "rf": {'n_estimators': [100, 200], 'max_depth': [10,
20], 'random_state': [42]},
17         "xgb": {'n_estimators': [100, 200], 'learning_rate':
[0.01, 0.1], 'max_depth': [6, 9], 'random_state': [42]},
18         "lgbm": {'n_estimators': [100, 200],
```

```
'learning_rate': [0.01, 0.1], 'max_depth': [6, 9, -1],
'random_state': [42]}
    }
    models, run_ids, params = {}, {}, {}
    for model_name, param_grid in param_grids.items():
        model, param, run_id =
train_model(eval(model_name.capitalize() + "Classifier()"),
model_name, param_grid, x_train, y_train)
        models[model_name], params[model_name],
run_ids[model_name] = model, param, run_id

    # Return results
    return models, run_ids, params, x_train.iloc[[0]],
x_test, y_test
```

Pseudocode 5.24 Fungsi Utama *Layer* 6

B. Fungsi Utama *Layer* 6b

```
def run_layer_06b(trained_models, run_ids, model_params,
input_example, x_test, y_test, champion_model=None):
    if x_test.empty or y_test.empty or len(x_test) !=
len(y_test):
        raise ValueError("Test features and labels must be
non-empty and of the same length.")

    metrics = {name: score_model(model, x_test, y_test,
run_ids[name]) for name, model in trained_models.items()}

    if champion_model:
        with mlflow.start_run(run_name="Champion_Scoring",
nested=True) as champion_run:
            metrics["champion"] =
score_model(champion_model, x_test, y_test,
champion_run.info.run_id)

    best_model_name = max(metrics, key=lambda name:
metrics[name]["f1_score"])
    best_model = champion_model if best_model_name ==
"champion" else trained_models[best_model_name]

    log_model(best_model, input_example if
best_model_name != "champion" else None,
metrics[best_model_name], model_params.get(best_model_name))
    save_model(best_model, best_model_name)

    return metrics
```

Pseudocode 5.25 Fungsi Utama *Layer* 6b

C. Fungsi *Train Model*

```
1  def train_model(estimator, model_name, param_grid, X_train,
y_train):
2      with
mlflow.start_run(run_name=f"{model_name}_GridSearch",
nested=True) as run:
3          grid_search = GridSearchCV(estimator, param_grid,
scoring='f1_weighted', cv=3, n_jobs=-1)
4          grid_search.fit(X_train, y_train)

5          for i, params in
enumerate(grid_search.cv_results_["params"]):
6              with
mlflow.start_run(run_name=f"{model_name}_Trial_{i+1}",
nested=True):
7                  mlflow.log_params(params)
8                  mlflow.log_metric("mean_test_score",
grid_search.cv_results_["mean_test_score"][i])
9                  mlflow.log_metric("std_test_score",
grid_search.cv_results_["std_test_score"][i])

10         best_model = grid_search.best_estimator_
11         best_params = grid_search.best_params_

12         mlflow.log_params(best_params)

13     return best_model, best_params, run.info.run_id
```

Pseudocode 5.26 Fungsi *Train Model*

D. Fungsi Logging, Scoring, dan Pendukung Lainnya

```
1  def log_dataset(df, source, name, target, identifier):
2      mlflow.log_input(
3          mlflow.data.from_pandas(df.drop(columns=identifier),
source=source, name=name, targets=target)
4      )
```

Pseudocode 5.27 Fungsi *log_dataset Layer* 6

```
1  def split_data(df, target, test_size=0.2, random_seed=42):
2      x = df.drop(columns=target)
3      y = df[target]
4      return train_test_split(x, y, test_size=test_size,
stratify=y, random_state=random_seed)
```

Pseudocode 5.28 Fungsi *split_data Layer* 6

```
1  def save_model(model, model_name, sub_dir="data/6_model"):
2      model_dir = os.path.join(get_volume("training"),
```

```
sub_dir)
3       os.makedirs(model_dir, exist_ok=True)
4       model_path = os.path.join(model_dir,
f"best_model_{model_name}.pkl")
5       with open(model_path, "wb") as f:
6           pickle.dump(model, f)
7       return model_path
```

Pseudocode 5.29 Fungsi *save_model Layer* 6

```
1   def score_model(model, X_test, y_test, run_id):
2       y_pred = model.predict(X_test)
3       acc, f1 = accuracy_score(y_test, y_pred),
f1_score(y_test, y_pred, average="weighted")
4       with mlflow.start_run(run_id=run_id, nested=True):
5           mlflow.log_metric("accuracy", acc)
6           mlflow.log_metric("f1_score", f1)
7       return {"accuracy": acc, "f1_score": f1}
```

Pseudocode 5.30 Fungsi *score_model Layer* 6b

```
1   def log_model(model, input_example, metrics, params=None):
2       if params: mlflow.log_params(params)
3       for metric_name, metric_value in metrics.items():
4           mlflow.log_metric(metric_name, metric_value)
5       mlflow.sklearn.log_model(model, "model", input_example)
```

Pseudocode 5.31 Fungsi *log_model Layer* 6b

```
1   def generate_truth_table(x_test, y_test, identifier_column):
2       if identifier_column not in x_test.columns:
3           raise ValueError(f"Identifier column
'{identifier_column}' not found.")
4       return
x_test[[identifier_column]].assign(prospect_status=y_test)
```

Pseudocode 5.32 Fungsi *generate_truth_table Layer* 6b

5.1.9. Fungsi Pendukung Experiment Tracking

Seperti yang dapat dilihat pada *layer* 1 hingga 6, *experiment tracking* telah diimplementasikan dengan cara melakukan pelaporan terhadap data maupun modelling menggunakan Mlflow. Fungsi-fungsi yang digunakan sering digunakan untuk itu adalah sebagai berikut:

A. Fungsi *Start MLflow Run*

```
1   def start_mlflow_run(run_name):
2       if not mlflow.active_run():
```

```
3           mlflow.start_run(run_name=run_name)
4           print(f"Started new run: {run_name}")
5       else:
6           print(f"Using existing run:
{mlflow.active_run().info.run_id}")
7       return mlflow.active_run()
```

Pseudocode 5.33 Fungsi *Start Mlflow Run*

Fungsi *start_mlflow_run* digunakan untuk memulai atau melanjutkan sebuah MLflow run. Jika tidak ada run yang aktif, fungsi ini akan membuat run baru dengan nama yang diberikan. Jika ada run yang sedang aktif, fungsi ini akan menggunakan run tersebut.

B. *Fungsi Log Features*

```
1  def log_features(data_dict, phase, mode):
2       base_dir = get_volume(mode)
3       log_dir = os.path.join(base_dir, "log")
4       os.makedirs(log_dir, exist_ok=True)
5
6       file_path = os.path.join(log_dir,
f"{phase}_features.yaml")
7       artifact_path = "data"
8
9       try:
10          with open(file_path, 'r') as file:
11              existing_data = yaml.safe_load(file) or {}
12      except FileNotFoundError:
13          existing_data = {}
14
15      for table_name, data in data_dict.items():
16          if not isinstance(data, pd.DataFrame):
17              try:
18                  data = pd.DataFrame(data)
19              except ValueError:
20                  continue
21
22          feature_info = {
23              'table_info': {
24                  'data_types':
data.dtypes.apply(str).to_dict(),
25                  'table_name': table_name,
26                  'feature_numbers': len(data.columns),
27                  'row_number': len(data)
28              }
```

```
        }

        existing_data[table_name] = feature_info

    with open(file_path, 'w') as file:
        yaml.dump(existing_data, file,
default_flow_style=False)

    return artifact_path, file_path
```

Pseudocode 5.34 Fungsi *Log Features*

Fungsi ini mencatat informasi tentang fitur dari satu atau beberapa DataFrame ke dalam file YAML untuk keperluan dokumentasi dan pelacakan. Fungsi ini berguna untuk menyimpan metadata seperti tipe data, jumlah fitur, dan jumlah baris dari setiap DataFrame.

C. *Fungsi Log Artifact*

```
def log_artifact(
    artifact_path: str,
    file_to_log: str
) -> None:
    try:
        # Check if there's an active run
        active_run = mlflow.active_run()

        if active_run:
            # Log artifact directly if there's an active
run
            mlflow.log_artifact(file_to_log, artifact_path)
            print(f"File '{file_to_log}' successfully
logged to artifact path '{artifact_path}' in active run ID:
{active_run.info.run_id}")
         else:
            # Start a new run if none is active
            with mlflow.start_run() as run:
                mlflow.log_artifact(file_to_log,
artifact_path)
                print(f"File '{file_to_log}' successfully
logged to artifact path '{artifact_path}' in new run ID:
{run.info.run_id}")

     except Exception as e:
         print(f"Failed to log artifact '{file_to_log}' to
path '{artifact_path}': {e}")
```

Pseudocode 5.35 Fungsi *Log Artifact*

Fungsi ini digunakan untuk mencatat file tertentu sebagai artifact dalam run MLflow yang aktif. Fungsi ini memeriksa apakah ada run aktif, dan jika tidak ada, maka akan membuat run baru untuk mencatat artifact.

D. *Fungsi Load Model from Mlflow*

```
1  def load_model_from_mlflow_alias(model_name, alias=None):
2      client = MlflowClient()
3
4      model_version = (
5          client.get_model_version_by_alias(name=model_name,
alias=alias)
6          if alias else
client.get_latest_versions(name=model_name)[0]
7      )
8
9      artifact_uri = model_version.source
10      model_path =
mlflow.artifacts.download_artifacts(artifact_uri)
11      flavor_file = os.path.join(model_path, "MLmodel")
12
13      if not os.path.exists(flavor_file):
14          raise FileNotFoundError(f"No MLmodel file found at
{artifact_uri}.")
15
16      with open(flavor_file, "r") as f:
17          flavors = yaml.safe_load(f).get("flavors", {})
18
19      loader_module = flavors.get("python_function",
{}).get("loader_module")
20      if loader_module:
21          if loader_module == "mlflow.pyfunc.model":
22              return mlflow.pyfunc.load_model(artifact_uri)
23          elif loader_module == "mlflow.sklearn":
24              return mlflow.sklearn.load_model(artifact_uri)
25          elif loader_module == "mlflow.tensorflow":
26              return
mlflow.tensorflow.load_model(artifact_uri)
27          elif loader_module == "mlflow.pytorch":
28              return mlflow.pytorch.load_model(artifact_uri)
29          elif loader_module == "mlflow.xgboost":
30              return mlflow.xgboost.load_model(artifact_uri)
31          else:
32              raise ValueError(f"Unsupported loader_module:
```

```
{loader_module}")
33
34      for flavor in ["sklearn", "tensorflow", "pytorch",
"xgboost"]:
35          if flavor in flavors:
36              return
globals()[f"mlflow.{flavor}"].load_model(artifact_uri)
37
38      raise ValueError(f"Unsupported model flavor:
{list(flavors.keys())}")
```

Pseudocode 5.36 Fungsi *Load Model from MLflow*

Fungsi ini bertugas memuat model dari MLflow Model Registry berdasarkan nama model dan alias (misalnya, champion model). Fungsi ini mendukung berbagai flavor model, seperti scikit-learn, TensorFlow, PyTorch, XGBoost, dan lainnya.

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap *pipeline machine learning* yang telah dirancang.

6.1. Tujuan Pengujian

Pengujian dilakukan untuk mengevaluasi fungsionalitas *pipeline machine learning*, khususnya pada kemampuan penerapan *experiment tracking*.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut:

a. Kemampuan *pipeline machine learning* untuk berjalan secara *end-to-end*.

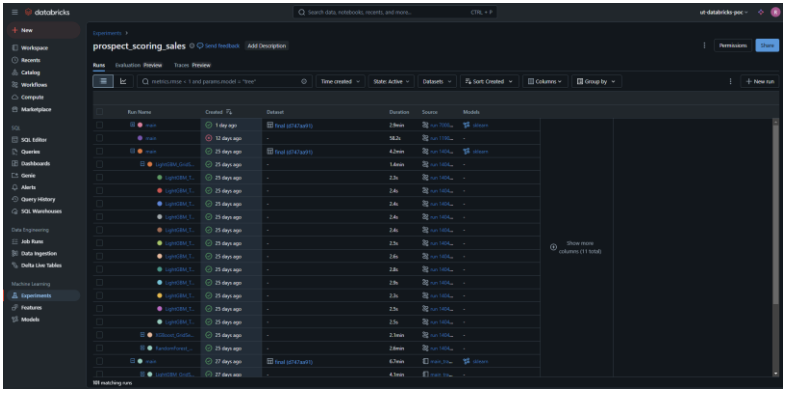
b. Kemampuan *pipeline machine learning* untuk melakukan *experiment tracking* ke Mlflow.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan menguji fungsionalitas *pipeline machine learning* secara *end-to-end* pada platform Databricks. *Pipeline* diharapkan dapat menerapkan kinerja dari setiap *layer* dengan optimal dan melaporkan segala hasilnya ke Mlflow untuk sisi *experiment tracking*-nya. Selain itu, model yang dikembangkan pada penerapan *pipeline machine learning* terhadap proyek *loss-deal prediction* juga akan dievaluasi hasilnya. *Metric* yang akan diuji adalah akurasi dan F1 Score dari model terbaik pada eksperimennya.

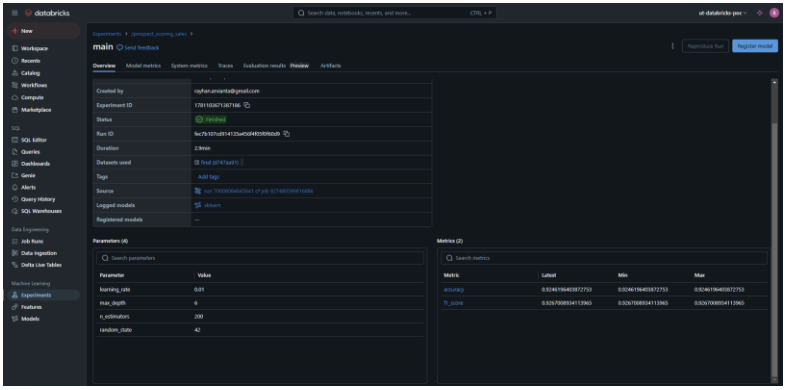
6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai kinerja *pipeline machine learning* terhadap skenario uji coba sesuai dengan kriteria pengujian yang ditetapkan.



Gambar 6.1 *Experiment Tracking* di MLflow

Experiment Tracking telah berhasil diimplementasikan menggunakan platform Mlflow yang terintegrasi dengan platform Databricks.

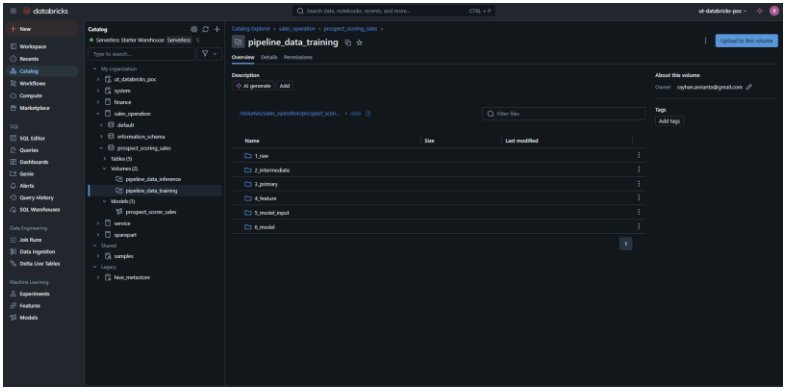


Gambar 6.2 Penyimpanan Model Terbaik pada Mlflow

Model terbaik yang telah dilatih akan otomatis tersimpan pada *run* utama (*main*) di MLflow. Dapat dilihat pada gambar 6.2 bahwa *metrics* yang direkam dari pelatihan model adalah akurasi dengan nilai 0,924 dan F1 Score 0,926. Dapat dilihat juga bahwa parameter yang digunakan dalam pelatihan model tersebut juga direkam.

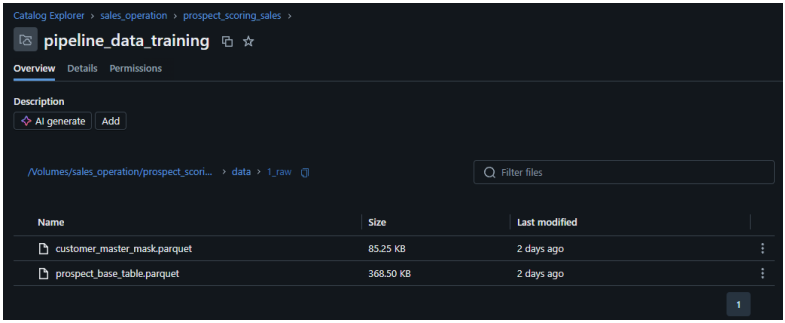
Parameter yang direkam dalam hal ini adalah *learning rate* dengan nilai 0.01, *max depth* dengan nilai 6, *n estimators* dengan nilai 200, dan *random state* dengan nilai 42.

Pada akhir setiap *layer* dari *pipeline* yang berjalan secara *end to end*, *pipeline* secara otomatis merekam data yang diproses pada setiap layernya, mulai dari data mentah hingga data yang sudah siap untuk digunakan dalam pelatihan model. Data ini disimpan di sebuah direktori bernama *data*. Dalam pengujian *pipeline*, volume pada Unity Catalog Databricks digunakan sebagai tempat penyimpanan utama yang dapat diakses layaknya seperti direktori pada pengembangan lokal.

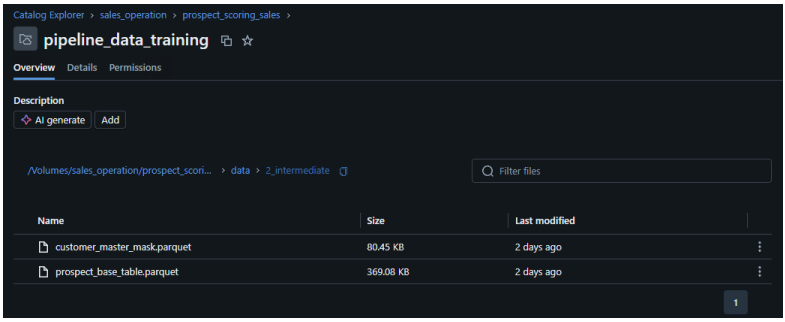


Gambar 6.3 Volume Penyimpanan Data setiap *Layer*

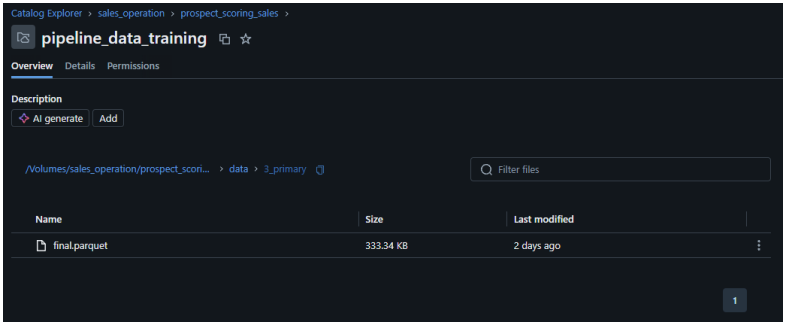
Dapat dilihat bahwa terdapat enam sub direktori di dalam direktori *data*, setiap sub direktori menyimpan data dari layer pada *pipeline*, mulai dari sub direktori *1_raw* yang menyimpan data mentah, hingga direktori *6_model* yang menyimpan data yang berkaitan dengan model, hingga *file* dari modelnya sendiri. Penyimpanan data yang dilakukan disini bertujuan untuk memudahkan dilakukannya *debugging* dan juga berfungsi sebagai penyimpanan sementara apabila terjadi hal-hal yang tidak terduga. Untuk cuplikan datanya tidak tersedia untuk menjaga kerahasiaan data perusahaan.



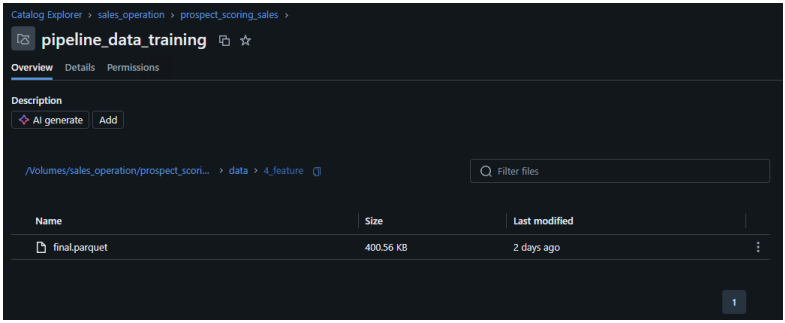
Gambar 6.4 Sub Direktori Data *Layer* 1



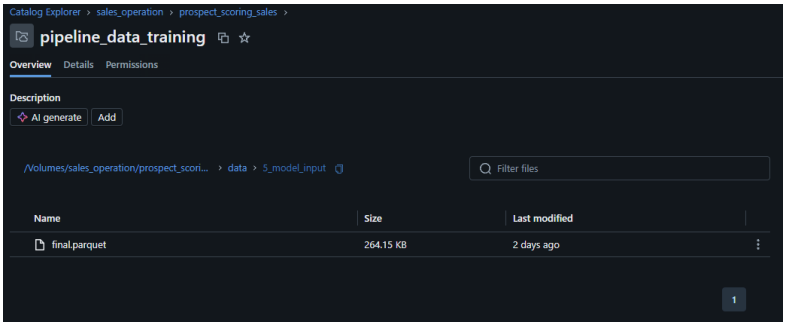
Gambar 6.5 Sub Direktori Data *Layer* 2



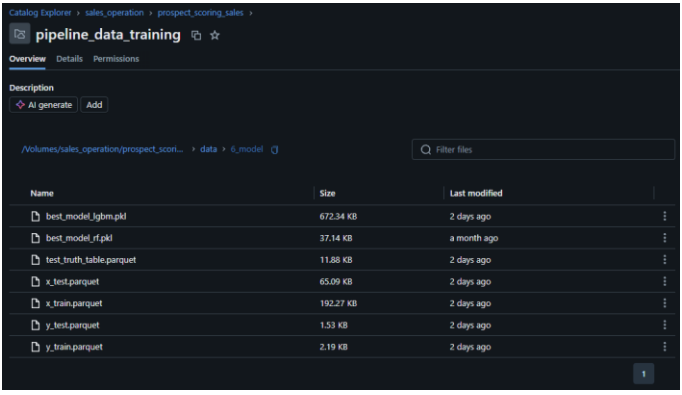
Gambar 6.6 Sub Direktori Data *Layer* 3



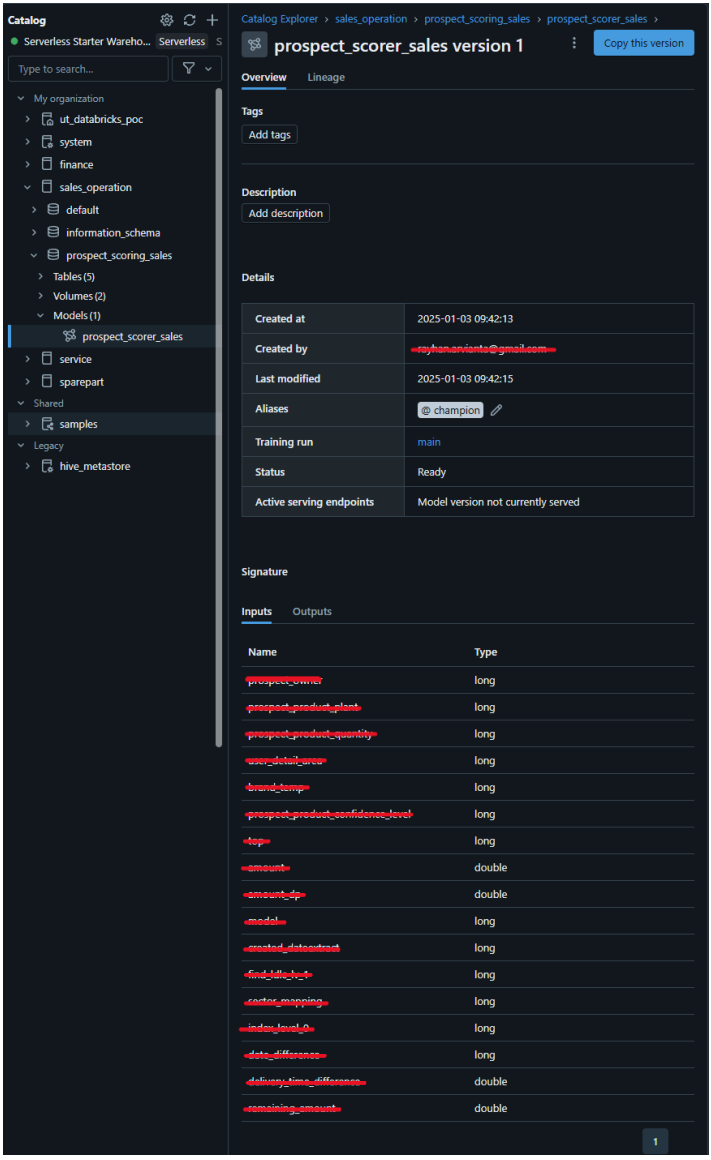
Gambar 6.7 Sub Direktori Data *Layer* 4



Gambar 6.8 Sub Direktori Data *Layer* 5



Gambar 6.9 Sub Direktori Data *Layer* 6



Gambar 6.10 Model Terbaik Tersimpan di Unity Catalog

Tabel 6. 1 Hasil Evaluasi Pengujian

No	Kriteria Pengujian	Hasil Pengujian
1	*Pipeline Machine Learning* dapat berjalan secara *end-to-end*	Terpenuhi
2	*Pipeline Machine Learning* dapat menerapakan *experiment tracking* dengan MLflow	Terpenuhi

Dari segala pengujian yang telah dilakukan, dapat disimpulkan bahwa *pipeline machine learning* telah berhasil berjalan secara *end to end*, serta *Experiment Tracking* telah terimplementasikan dengan baik menggunakan platform MLflow. *Log output* dari *pipeline machine learning*nya dapat dilihat pada lampiran 1.

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan perancangan *pipeline machine learning* dengan prinsip MLOps di PT United Tractors Tbk adalah sebagai berikut:

a. *Pipeline machine learning* yang terstandarisasi telah berhasil berjalan secara *end-to-end*

b. *Pipeline machine learning* berbasis MLOps berhasil diimplementasikan dengan menerapkan prinsip MLOps yaitu *experiment tracking*.

7.2. Saran

Saran yang dapat diberikan untuk penerapan *pipeline machine learning* berbasis MLOps di PT United Tractors adalah sebagai berikut:

a. Perlu mengembangkan standarisasi untuk bentuk model yang unik (*custom*) agar segala *usecase* dapat ditangani dengan baik.

b. *Pipeline* dapat dikembangkan kembali untuk implementasi monitoring model, *schedulling*, serta CI/CD yang matang untuk meningkatkan efisiensi kerja.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] United Tractors. (tanpa tahun). *Company Overview*. Diakses pada 21 Januari 2025, dari https://www.unitedtractors.com/en/company-overview/

[2] United Tractors (tanpa tahun). Struktur Organisasi. Diakses pada 21 Januari 2025, dari https://en.wikipedia.org/wiki/United_Tractors

[3] United Tractors (tanpa tahun). Visi dan Misi. Diakses pada 21 Januari 2025, dari https://www.unitedtractors.com/visi-misi/

[4] Rachman, Vicky. (2023). United Tractors, Kembangkan Talenta Digital untuk Akselerasi Transformasi Digital. Diakses pada 21 Januari 2025, dari https://swa.co.id/read/423531/united-tractors-kembangkan-talenta-digital-untuk-akselerasi-transformasi-digital

[5] Microsoft. (tanpa tahun). Apa itu Pembelajaran Mesin. Diakses pada 22 Januari 2025, dari https://azure.microsoft.com/id-id/resources/cloud-computing-dictionary/what-is-machine-learning-platform

[6] Dicoding. (2020). Apa itu *Machine Learning*? Beserta Pengertian dan Cara Kerjanya. Diakses pada 22 Januari 2025, dari https://www.dicoding.com/blog/machine-learning-adalah/

[7] Python Institute. (tanpa tahun). Python – *The Language of Today and Tomorrow*. Diakses pada 22 Januari 2025, dari https://pythoninstitute.org/about-python

[8] Kreuzberger dkk. (2023). *Machine Learning Operations (MLOps): Overview, Definition, and Architecture*. *IEEE Access, 11(3)*, 31866–31877.

[9] Zaharia, M. & Simeone, C. (2018). *Introducing Mlflow: an Open Source Machine Learning Platform*. Diakses pada 22 Januari 2025, dari https://www.databricks.com/blog/2018/06/05/introducing-mlflow-an-open-source-machine-learning-platform.html

[10] Mlflow. (tanpa tahun). *Mlflow Overview*. Diakses pada 22 Januari 2025, dari https://mlflow.org/docs/latest/introduction/index.html
[11] Databricks. (tanpa tahun). *Data Lakehouse*. Diakses pada 22 Januari 2025, dari https://www.databricks.com/glossary/data-lakehouse
[12] Databricks. (2024). *What is Unity Catalog*. Diakses pada 22 Januari 2025, dari https://docs.databricks.com/en/data-governance/unity-catalog/index.html
[13] Databricks. (2024). *Schedule and Orchestrate Workflows*. Diakses pada 22 Januari 2025, dari https://docs.databricks.com/en/jobs/index.html
[14] Microsoft. (2024). *The Scope of the Lakehouse Platform*. Diakses pada 22 Januari 2025, dari https://learn.microsoft.com/en-us/azure/databricks/lakehouse-architecture/scope

LAMPIRAN

Lampiran 1. *Output Log Pipeline Machine Learning*

```
MLflow configured in databricks mode.
Started new MLflow run with name: main
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/1_raw_features.yaml
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/2_intermediate_features.yaml
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/3_primary_features.yaml
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/4_feature_features.yaml
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/5_model_input_features.yaml
Deleted:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/6_model_features.yaml
All .yaml files in
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/ have been cleared.
Using SparkSession with app name: DatabricksIngestion
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/1_raw
Ingesting data for Databricks table: customer_master_mask
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/1_raw/customer_master_mask.parquet
Data for `customer_master_mask` saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/1_raw/customer_master_mask.parquet
Ingesting data for Databricks table: prospect_base_table
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/1_raw/prospect_base_table.parquet
Data for `prospect_base_table` saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/1_raw/prospect_base_table.parquet
Feature information for all tables saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/1_raw_features.yaml
File
```

```
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/1_raw_features.yaml' successfully logged to
artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee
Feature information for all tables saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/2_intermediate_features.yaml
File
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/2_intermediate_features.yaml' successfully logged
to artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee


/Workspace/sales_operation/.bundle/prospect_scoring_sales/dev/f
iles/src/pipeline/l03_primary/node_cleaning.py:25: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  train['estimate_deal_date'] =
pd.to_datetime(train['estimate_deal_date'], errors='coerce')
/Workspace/sales_operation/.bundle/prospect_scoring_sales/dev/f
iles/src/pipeline/l03_primary/node_cleaning.py:26: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  train['req_delivery_date'] =
pd.to_datetime(train['req_delivery_date'], errors='coerce')
/Workspace/sales_operation/.bundle/prospect_scoring_sales/dev/f
iles/src/pipeline/l03_primary/node_cleaning.py:27: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  train['valid_from'] = pd.to_datetime(train['valid_from'],
errors='coerce')
/Workspace/sales_operation/.bundle/prospect_scoring_sales/dev/f
iles/src/pipeline/l03_primary/node_cleaning.py:28: UserWarning:
Parsing dates in DD/MM/YYYY format when dayfirst=False (the
default) was specified. This may lead to inconsistently parsed
dates! Specify a format to ensure consistent parsing.
  train['valid_to'] = pd.to_datetime(train['valid_to'],
errors='coerce')


Feature information for all tables saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/3_primary_features.yaml
File
```

```
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/3_primary_features.yaml' successfully logged to
artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee
Mapping untuk kolom 'prospect_owner':
    prospect_owner  Encoded_Value
0              bc1              0
1             bc10              1
2            bc100              2
3            bc101              3
4            bc103              4
..             ...            ...
119           bc90            119
120           bc91            120
121           bc92            121
122           bc95            122
123           bc99            123

[124 rows x 2 columns]

==================================================

Mapping untuk kolom 'prospect_product_plant':
   prospect_product_plant  Encoded_Value
0                     agr              0
1                     amb              1
2                     bjm              2
3                     blg              3
4                     blp              4
5                     cen              5
6                     con              6
7                     for              7
8                     jbi              8
9                     jkt              9
10                    jyp             10
11                    mdn             11
12                    mdo             12
13                    mng             13
14                    mns             14
15                    pdg             15
16                    pkb             16
17                    plb             17
18                    plu             18
19                    ptk             19
20                    sby             20
21                    smd             21
22                    smg             22
23                    spt             23
```

```
24                    trk             24
25                    upg             25

==================================================

Mapping untuk kolom 'user_detail_area':
  user_detail_area  Encoded_Value
0              agf              0
1           area 1              1
2           area 2              2
3           area 3              3
4              cng              4
5              mng              5
6              mns              6
7             none              7
8              tru              8

==================================================

Mapping untuk kolom 'brand_temp':
  brand_temp  Encoded_Value
0        kcm              0

==================================================

Mapping untuk kolom 'prospect_product_confidence_level':
  prospect_product_confidence_level  Encoded_Value
0                       belum yakin              0
1                 potensi lost deal              1
2                             yakin              2

==================================================

Mapping untuk kolom 'top':
     top  Encoded_Value
0   pj00              0
1   pj01              1
2   pj10              2
3   pj15              3
4   pj20              4
5   pj25              5
6   pj30              6
7   pj35              7
8   pj40              8
9   pj50              9
10  pj55             10
11  pj60             11
12  pj65             12
```

```
13  pj70             13
14  pj80             14
15  pj90             15

==================================================

Mapping untuk kolom 'model':
        model  Encoded_Value
0        875.              0
1     d155a-6              1
2     d21pl-8              2
3    d31ex-22              3
4    d31px-22              4
..        ...            ...
60   wa500-6r             60
61   wa600-3a             61
62   wa800-3a             62
63   wa900-8r             63
64  wb93r-5e0             64

[65 rows x 2 columns]

==================================================

Mapping untuk kolom 'sector_mapping':
  sector_mapping  Encoded_Value
0            agr              0
1            cmt              1
2            col              2
3            con              3
4            for              4
5            mng              5
6            uof              6

==================================================

Feature information for all tables saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/4_feature_features.yaml
File
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/4_feature_features.yaml' successfully logged to
artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee
Datetime columns identified and dropped: ['valid_from',
'valid_to', 'estimate_deal_date', 'req_delivery_date']
Datetime columns identified and dropped: []
Feature information for all tables saved to
```

```
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/5_model_input_features.yaml
File
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/5_model_input_features.yaml' successfully logged
to artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee
Running in training mode...


/databricks/python/lib/python3.11/site-
packages/mlflow/data/dataset_source_registry.py:150:
UserWarning: The specified dataset source can be interpreted in
multiple ways: LocalArtifactDatasetSource,
LocalArtifactDatasetSource. MLflow will assume that this is a
LocalArtifactDatasetSource source.
  return _dataset_source_registry.resolve(
/databricks/python/lib/python3.11/site-
packages/mlflow/types/utils.py:393: UserWarning: Hint: Inferred
schema contains integer column(s). Integer columns in Python
cannot represent missing values. If your input data contains
missing values at inference time, it will be encoded as floats
and will cause a schema enforcement error. The best way to
avoid this problem is to infer the model schema based on a
realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as
doubles (float64) whenever these columns may have missing
values. See `Handling Integers With Missing Values
<https://www.mlflow.org/docs/latest/models.html#handling-
integers-with-missing-values>`_ for more details.
  warnings.warn(


Converted y_train to DataFrame.
Converted y_test to DataFrame.
Feature information for all tables saved to
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/log/6_model_features.yaml
File
'/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_
training/log/6_model_features.yaml' successfully logged to
artifact path 'data' in active run ID:
6f03a38f4c4444208718a611137c6cee
Fitting 3 folds for each of 4 candidates, totalling 12 fits
Fitting 3 folds for each of 8 candidates, totalling 24 fits
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[LightGBM] [Info] Number of positive: 3157, number of negative:
696
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.037713 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.000441 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
696
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.000349 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
```

```
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.000329 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.000844 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
696
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.010629 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3156, number of negative:
```

```
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.023529 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1791
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819102 ->
initscore=1.510275
[LightGBM] [Info] Start training from score 1.510275
[LightGBM] [Info] Number of positive: 3156, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.017389 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1791
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819102 ->
initscore=1.510275
[LightGBM] [Info] Start training from score 1.510275
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
```

```
696
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.113623 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[

*** WARNING: max output size exceeded, skipping output. ***

an set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1791
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819102 ->
initscore=1.510275
[LightGBM] [Info] Start training from score 1.510275
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
696
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.019821 seconds.
You can set `force_col_wise=true` to remove the overhead.
```

```
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
[LightGBM] [Info] Number of positive: 3156, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.088922 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1791
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819102 ->
initscore=1.510275
[LightGBM] [Info] Start training from score 1.510275
[LightGBM] [Info] Number of positive: 3156, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.010062 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1791
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819102 ->
initscore=1.510275
[LightGBM] [Info] Start training from score 1.510275
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
696
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.001447 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1790
[LightGBM] [Info] Number of data points in the train set: 3853,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819362 ->
initscore=1.512028
[LightGBM] [Info] Start training from score 1.512028
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the
overhead of testing was 0.036801 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
```

```
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.037169 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Info] Number of positive: 3157, number of negative:
697
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.015905 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1795
[LightGBM] [Info] Number of data points in the train set: 3854,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819149 ->
initscore=1.510592
[LightGBM] [Info] Start training from score 1.510592
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
```

```
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Info] Number of positive: 4735, number of negative:
1045
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the
overhead of testing was 0.000503 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1804
[LightGBM] [Info] Number of data points in the train set: 5780,
number of used features: 16
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.819204 ->
initscore=1.510965
[LightGBM] [Info] Start training from score 1.510965
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best
gain: -inf


/databricks/python/lib/python3.11/site-
packages/mlflow/types/utils.py:393: UserWarning: Hint: Inferred
schema contains integer column(s). Integer columns in Python
cannot represent missing values. If your input data contains
missing values at inference time, it will be encoded as floats
and will cause a schema enforcement error. The best way to
avoid this problem is to infer the model schema based on a
realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as
doubles (float64) whenever these columns may have missing
values. See `Handling Integers With Missing Values
<https://www.mlflow.org/docs/latest/models.html#handling-
integers-with-missing-values>`_ for more details.
  warnings.warn(
/databricks/python/lib/python3.11/site-
packages/mlflow/types/utils.py:393: UserWarning: Hint: Inferred
schema contains integer column(s). Integer columns in Python
cannot represent missing values. If your input data contains
missing values at inference time, it will be encoded as floats
```

```
and will cause a schema enforcement error. The best way to
avoid this problem is to infer the model schema based on a
realistic data sample (training dataset) that includes missing
values. Alternatively, you can declare integer columns as
doubles (float64) whenever these columns may have missing
values. See `Handling Integers With Missing Values
<https://www.mlflow.org/docs/latest/models.html#handling-
integers-with-missing-values>`_ for more details.
  warnings.warn(
/databricks/python/lib/python3.11/site-
packages/_distutils_hack/__init__.py:33: UserWarning:
Setuptools is replacing distutils.
  warnings.warn("Setuptools is replacing distutils.")


Model saved to:
/Volumes/sales_operation/prospect_scoring_sales/pipeline_data_t
raining/data/6_model/best_model_lgbm.pkl
```

BIODATA PENULIS

Nama	: Rayhan Arvianta Bayuputra
Tempat, Tanggal Lahir	: Jakarta, 20 November 2003
Jenis Kelamin	: Laki-laki
Telepon	: +6281282351996
Email	: rayhan.arvianta@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 8 (Delapan)