KERJA PRAKTIK - EF234603

Penerapan Algoritma Machine Learning XGBoost, CatBoost, Linear Regression, LightGBM, Random Forest dengan Metode Ensemble (Studi Kasus: Penjualan Emas)

Departemen Teknik Informatika - ITS
Jalan Teknik Kimia, Sukolilo, Kota Surabaya, Jawa Timur, 60111
Periode: 1 Januari 2025 - 30 Mei 2025

Oleh:

Mochammad Zharif Asyam Marzuqi 5025221163
Brendan Timothy Mannuel 5025221177

Pembimbing Jurusan
Imam Mustafa Kamal, S.ST, Ph.D
Pembimbing Lapangan
Setijo Agus S.T.

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

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK - EF234603

Penerapan Algoritma Machine Learning XGBoost, CatBoost, Linear Regression, LightGBM, Random Forest dengan Metode Ensemble (Studi Kasus: Penjualan Emas)

Departemen Teknik Informatika - ITS
Jalan Teknik Kimia, Sukolilo, Kota Surabaya, Jawa Timur, 60111
Periode: 1 Januari 2025 – 30 Mei 2025

Oleh:
Mochammad Zharif Asyam Marzuqi 5025221163
Brendan Timothy Mannuel 5025221177

Pembimbing Jurusan
Imam Mustafa Kamal, S.ST, Ph.D.

Pembimbing Lapangan
Setijo Agus S.T.

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

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

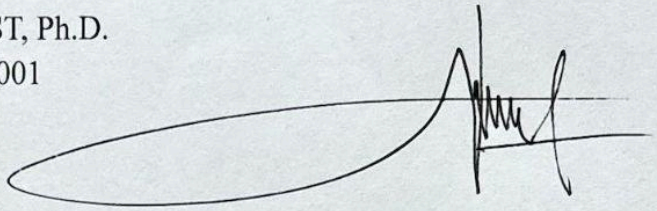
Penerapan Algoritma Machine Learning XGBoost, CatBoost, Linear Regression, LightGBM, Random Forest dengan Metode Ensemble (Studi Kasus: Penjualan Emas)

Oleh:

Mochammad Zharif Asyam Marzuqi	5025221163
Brendan Timothy Mannuel	5025221177

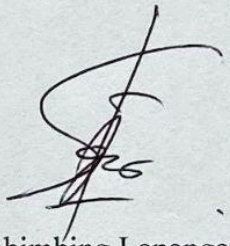
Disetujui oleh Pembimbing Kerja Praktik:

1. Imam Mustafa Kamal, S.ST, Ph.D.
   NIP 19920312 202406 1 001



(Dosen Pembimbing Departemen)

2. Setijo Agus S.T.



(Pembimbing Lapangan)

3. Herry Gunawan S.T.

(HRD)

[Halaman ini sengaja dikosongkan]

ABSTRAKSI

PT Untung Bersama Sejahtera (PT. UBS) merupakan perusahaan yang telah beroperasi selama lebih dari 40 tahun di industri perhiasan emas, menghadapi tantangan dalam merencanakan kebutuhan bahan baku emas akibat fluktuasi harga dan kondisi pasar global. Untuk mengatasi hal ini, diperlukan sistem peramalan (forecasting) penjualan emas yang akurat guna mendukung efisiensi perencanaan dan pengambilan keputusan strategis. Kerja praktik ini bertujuan untuk memenuhi kewajiban akademik sebesar 4 SKS dan membantu PT. UBS dalam mengembangkan model forecasting penjualan emas yang akurat dengan menerapkan beberapa algoritma machine learning, yaitu XGBoost, CatBoost, Linear Regression, LightGBM, dan Random Forest, yang dikombinasikan menggunakan metode Ensemble.

Metodologi yang digunakan meliputi perumusan masalah melalui eksplorasi data historis penjualan, studi literatur untuk pemilihan teknologi dan model, analisis dan perancangan sistem dengan arsitektur client-server, implementasi sistem, serta pengujian dan evaluasi. Sistem dikembangkan sebagai aplikasi berbasis web menggunakan Vue.js untuk frontend, Python untuk backend, dan DuckDB sebagai basis data, serta menerapkan queue worker untuk menangani proses forecasting yang intensif. Implementasi deployment dilakukan menggunakan Docker untuk portabilitas.

Hasil dari kerja praktik ini adalah sebuah aplikasi web yang mampu melakukan forecasting penjualan emas, memvisualisasikan data hasil prediksi beserta data aktual, serta memungkinkan pengguna untuk menambah data historis dan mengunduh hasil forecast. Diharapkan sistem ini dapat meningkatkan akurasi prediksi, mempercepat pengambilan keputusan, mengurangi risiko kelebihan atau kekurangan stok, dan pada akhirnya mengoptimalkan biaya operasional PT. UBS.

Kata Kunci : *Penjualan Emas, Forecasting, Machine Learning, Ensemble, XGBoost, CatBoost, Linear Regression, LightGBM, Random Forest, Vue.js, Python, DuckDB.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur kita panjatkan kepada Allah SWT atas berkah dan karunianya, sehingga penulis dapat menyelesaikan serangkaian kerja praktik di PT. Untung Bersama Sejahtera dengan judul "Penerapan Algoritma Machine Learning XGBoost, CatBoost, Linear Regression, LightGBM, Random Forest dengan Metode Ensemble (Studi Kasus: Penjualan Emas)", yang merupakan kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember Surabaya.

Penulis juga ingin mengucapkan terima kasih kepada seluruh pihak yang telah membantu penulis dalam menyelesaikan kewajiban kerja praktik secara langsung maupun tidak langsung, antara lain :

1. Orang tua penulis yang selalu memberikan dukungan tanpa henti.
2. Bapak Hadziq Fabroyir, S.Kom., Ph.D selaku koordinator Kerja Praktik Departemen Teknik Informatika.
3. Bapak Imam Mustafa Kamal, S.ST, Ph.D selaku dosen pembimbing departemen yang telah membantu dan membina penulis selama masa kerja praktik.
4. Bapak Setijo Agus S.T. selaku pembimbing lapangan dari PT. UBS yang sudah membimbing, mengarahkan dan mengevaluasi penulis selama melakukan kerja praktik
5. Seluruh teman-teman penulis yang senantiasa memberikan semangat kepada penulis.

Sebagai penyusun, penulis sadar akan adanya kekurangan dalam buku laporan kerja praktik ini, baik dari segi penyajian data maupun aspek teknis lainnya. Oleh karena itu, penulis menerima setiap saran dan kritik yang bersifat membangun dari pembaca dengan tulus, sebagai bahan perbaikan di masa depan.

Surabaya, 23 Mei 2025

Mochammad Zharif Asyam Marzuqi dan Brendan Timothy Mannuel

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

PT Untung Bersama Sejahtera (PT. UBS) telah beroperasi selama 42 tahun dalam industri perhiasan emas dan dikenal luas berkat kualitas serta desain produknya yang unggul. Seiring dengan pertumbuhan bisnis dan meningkatnya persaingan di pasar, perencanaan pasokan bahan baku menjadi faktor strategis yang semakin krusial. Salah satu tantangan utama dalam industri ini adalah mengelola kebutuhan emas sebagai bahan baku utama yang nilainya sangat fluktuatif dan sensitif terhadap kondisi pasar global.

Untuk menjaga kelancaran produksi serta mengoptimalkan biaya operasional, PT. UBS menerapkan proses *forecasting* kebutuhan emas. Pendekatan ini mempertimbangkan data historis penjualan, tren pasar, dan faktor musiman (*seasonality*) yang mempengaruhi permintaan produk perhiasan. Misalnya, permintaan emas cenderung meningkat pada periode tertentu seperti menjelang hari raya, di mana konsumen lebih aktif membeli perhiasan sebagai hadiah atau kebutuhan budaya.

Analisis tren juga menjadi bagian penting dalam forecasting, karena pola permintaan dapat berubah seiring perubahan preferensi konsumen dan kondisi ekonomi. Dengan memahami tren jangka panjang dan fluktuasi musiman secara menyeluruh, perusahaan dapat memperkirakan kebutuhan emas secara lebih akurat dan menyusun strategi pembelian serta produksi yang efisien.

Forecasting berbasis data ini memungkinkan PT. UBS untuk merespons dinamika pasar secara proaktif, menghindari kekurangan atau kelebihan stok, serta meminimalkan risiko kerugian akibat perubahan harga emas. Hal ini menjadi bagian penting dalam mendukung pengambilan keputusan strategis dan mempertahankan daya saing perusahaan di industri perhiasan yang terus berkembang.

1.2. Tujuan

Tujuan kerja praktik ini adalah untuk memenuhi kewajiban akademik sebesar 4 SKS dan membantu PT. UBS meningkatkan efisiensi perencanaan kebutuhan emas melalui penerapan model *forecasting* yang akurat.

1.3. Manfaat

Manfaat yang diperoleh dari kerja praktik di PT. UBS ini antara lain:

1. Meningkatkan akurasi prediksi penjualan untuk perencanaan stok dan mengurangi risiko kelebihan/kekurangan stok.
2. Memahami penerapan model prediksi berdasarkan data historis.
3. Mempercepat pengambilan keputusan dengan sistem *forecasting* otomatis.
4. Mengurangi lead time dengan model prediksi yang siap digunakan.
5. Menurunkan biaya stok tidak optimal dan mengurangi analisis manual berulang.

1.4. Rumusan Masalah

Rumusan masalah dari kegiatan magang ini adalah sebagai berikut:

1. Bagaimana mengembangkan model prediksi yang akurat untuk memperkirakan kebutuhan emas berdasarkan pola historis penjualan?
2. Bagaimana penerapan model *forecasting* dapat mempercepat proses pengambilan keputusan dibandingkan metode manual?
3. Bagaimana model prediksi dapat membantu mengurangi biaya akibat kelebihan atau kekurangan stok?

1.5. Lokasi dan Waktu Kerja Praktik

Kegiatan magang dilaksanakan secara hybrid (offline dan online) di kantor pusat PT. UBS, di Jl. Kenjeran 395-399, Kenjeran, Kec. Tambaksari, Kota Surabaya, Jawa Timur 60134. Adapun magang dilaksanakan pada periode Januari hingga Juni 2025. Dikarenakan pelaksanaan kerja praktik dilakukan bersamaan dengan pelaksanaan perkuliahan semester Genap 2024/2025 dan mayoritas secara *work from home,* waktu pelaksanaan tidak diatur secara ketat.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Untuk memahami kebutuhan sistem yang akan dikembangkan dalam forecasting kebutuhan emas, penulis melakukan eksplorasi data (EDA) berdasarkan file yang diperoleh dari PT. UBS serta mengadakan pertemuan dengan tim terkait. Proses ini memberikan wawasan mengenai pola dan tren dalam data, serta bagaimana aplikasi yang akan dibangun dapat mendukung pengambilan keputusan strategis perusahaan.

1.6.2. Studi Literatur

Setelah memahami bagaimana sistem tersebut berjalan, kami menganalisis dan menentukan teknologi serta *model* yang akan digunakan dalam pengembangan *website*. Dalam proses ini, kami memilih Vue.js, Python, dan DuckDB sebagai bagian dari implementasi, serta menerapkan metode *Ensemble* dengan model XGBoost, CatBoost, Linear Regression, LightGBM, dan Random Forest. Selain itu, kami menetapkan aturan penulisan konfigurasi agar mudah dipahami dan dikelola oleh pengembang lainnya.

1.6.3. Analisis dan Perancangan Sistem

Setelah tinjauan yang dipakai telah diberitahu, untuk merancang sistem yang baik perlu adanya sebuah desain arsitektur sistem. Pada website ini tim developer setuju Website ini dikembangkan dengan arsitektur *client-server*, menggunakan Vue.js untuk *frontend* dan Python sebagai *backend*, dengan DuckDB untuk manajemen data. Untuk forecasting kebutuhan emas, diterapkan metode *Ensemble* dengan model XGBoost, CatBoost, Linear Regression, LightGBM, dan Random Forest guna meningkatkan akurasi prediksi serta efisiensi sistem.

1.6.4. Implementasi Sistem

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini, penulis memulai dengan pencarian model *forecasting* yang paling akurat, lalu dilanjutkan dengan pengembangan sistem mencakup *database*, antarmuka pengguna, dan logika *backend* untuk mendukung analisis, efisiensi, dan prediksi.

1.6.5. Pengujian dan Evaluasi

Setelah website yang telah direncanakan telah jadi, perlu adanya evaluasi untuk menguji apakah website sesuai dengan harapan client. Evaluasi dilakukan dengan cara melakukan meeting dengan Tim IT PT.UBS dan mempresentasikan beberapa progress dari aplikasi penulis.

1.6.6. Kesimpulan dan Saran

Pengujian telah memenuhi syarat, dan hasil pekerjaan berupa *DBF, source code, ERD*, serta dokumentasi diserahkan kepada perusahaan. Kegiatan kerja praktik berjalan lancar.

1.7. Sistematika Laporan

Laporan kerja praktik terdiri dari tujuh bab dengan rincian sebagai berikut.

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 gambaran umum mengenai perusahaan PT. Untung Bersama Sejahtera

1.7.3 Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4 Analisis dan Perancangan Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5 Implementasi Sistem

Bab ini berisi penjelasan tahap-tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6 Pengujian dan Evaluasi

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

1.7.7 Kesimpulan dan Saran

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

BAB II
PROFIL PERUSAHAAN

2.1. Profil PT. Untung Bersama Sejahtera

PT. UBS adalah kepanjangan dari **PT. UNTUNG BERSAMA SEJAHTERA** yang bergerak dalam bidang industri perhiasan emas. Pada permulaan tahun 1981, PT. UBS hanyalah sebuah perusahaan home industri yang kemudian pada tahun 1985 berubah menjadi CV. Untung yang mulai merintis usahanya dalam bidang pengecoran perhiasan tradisional. Dengan perkembangan usaha yang cukup signifikan maka pada tahun 1991, semakin berkembang dan berubah menjadi PT. UBS dengan nomor Badan Usaha 188/T/Industri/1995 dan mulai mengadopsi beraneka macam teknologi, manajemen dan aktifitas pemasaran yang mampu menjadikan UBS sebagai perusahaan kelas dunia. UBS pun sudah membuktikan dirinya sebagai perusahaan kelas dunia yang mampu bertahan pada saat badai kritis ekonomi yang berkepanjangaan menerpa perekonomian Indonesia.

Visi

”Menjadi Pemimpin Pasar Perhiasan Emas Yang Kompetitif Dengan Merk Berkelas Dunia”

Misi

1. Menjamin kepuasan pelanggan dengan produk berkualitas, desain terlengkap dan inovatif, teknologi terkini dan sdm berkualitas
2. Menjamin keberhasilan dan pertumbuhan bisnis perusahaan melalui daya saing unggul, diversifikasi usaha, pertumbuhan pangsa pasar, basis pelanggan setia dan kinerja operasi yang efesien
3. Menyejahterakan SDM dan pemilik saham melalui program pengembangan, penilaian kinerja, fasilitas dan jenjang karir.
4. Memberikan kontribusi positif kepada masyarakat untuk perbaikan kualitas hidup dan memajukan bangsa
5. Melakukan kegiatan produksi yang ramah lingkungan

Nilai - Nilai Budaya

Nilai - Nilai Budaya UBS yang kita pakai sebagai pemandu kita dalam bekerja adalah sebagai berikut:

1. Integrity & Loyality
2. Unity
3. Creative & Inovatif
4. Working Smart
5. Tough

Dengan pedoman Visi, Misi dan nilai budaya, PT. UBS terus berusaha mewujudkan Visi & Misinya antara lain dengan penggabungan kecanggihan teknologi perhiasan (METHOD & MACHINE) , Sumber Daya Manusia (MAN) yang terampil dan selalu menjunjung tinggi persatuan antara setiap personil, serta terus menekankan sikap kreatif, disiplin dan bertanggung jawab di dalam setiap personilnya, maka diharapkan UBS dapat selalu berkembang dan dapat mewujudkan impianya.

Semua prinsip-prinsip kerja karyawan PT. UBS, tertuang di dalam nilai budaya perusahaan yang terus menerus kita pertahankan, junjung tinggi, patuhi dan kembangkan sehingga visi Perusahaan untuk **"Menjadi Pemimpin Pasar Perhiasan Emas Yang Kompetitif Dengan Merk Berkelas Dunia"** dapat terwujud.

Dengan jumlah karyawan lebih dari 2000 orang dan dalam usaha meningkatkan kinerja karyawannya, UBS pun tidak lupa untuk selalu memberikan fasilitas terbaik bagi karyawannya. Dalam upaya untuk meningkatkan ketrampilan dan skill para karyawannya, UBS juga terus menerus memperkaya diri dengan training-training yang up to date baik dalam bidang teknis maupun bidang-bidang yang lain (misal: Training komunikasi, Team work, motivasi, dll). Dengan adanya pembekalan tersebut, diharapkan setiap personil yang menjadi karyawan UBS dapat menjadi karyawan yang berkualitas dunia.

UBS juga merupakan perusahaan yang ramah lingkungan. Saat ini dengan adanya teknologi yang modern, persoalan limbah yang seringkali menjadi "momok" bagi perusahaan besar sudah dapat teratasi dengan baik. Kebersihan dan keselamatan kerja pun juga ditekankan di lingkungan perusahaan, produktifitas kerja setiap karyawan akan semakin meningkat. akhirnya, penulis berharap komitmen penulis untuk menjadi perusahaan perhiasan kelas dunia akan juga menjadi komitmen anda. Mari bergabunglah bersama UBS,

perusahaan kelas dunia, untuk juga menjadi karyawan kelas dunia. Dengan bertambahnya customer, maka pelayanan terhadap customer adalah fokus utama. Oleh karena itu PT. UBS mendirikan **INFO SERVICE CENTER (ISC)** di pertokoan yang mempunyai sentra penjualan perhiasan emas beberapa kota besar di Indonesia (Jakarta, Semarang, Surabaya, Solo, Denpasar, Makasar, Jember, Medan), dengan tujuan agar para pelanggan mendapatkan informasi secara langsung mengenai produk-produk UBS yang sedang beredar di pasaran.

2.2. Lokasi

Jl. Kenjeran 395-399, Kenjeran, Kec. Tambaksari, Kota Surabaya, Jawa Timur 60134

BAB III
TINJAUAN PUSTAKA

3.1. Pemrograman Web

Pemrograman web merupakan salah satu bidang utama dalam ilmu komputer yang berfokus pada pembangunan aplikasi dan situs yang dapat diakses melalui peramban web. Secara umum, pemrograman web terbagi menjadi dua bagian: sisi klien (client-side) dan sisi server (server-side). Pemrograman sisi klien mencakup elemen-elemen antarmuka yang berinteraksi langsung dengan pengguna, menggunakan teknologi seperti HTML, CSS, dan JavaScript. Sementara itu, sisi server menangani logika bisnis, pengelolaan data, dan komunikasi dengan basis data menggunakan bahasa pemrograman seperti PHP, Python, atau JavaScript berbasis Node.js.

Seiring berkembangnya teknologi, pengembangan web modern banyak didukung oleh kehadiran berbagai framework dan pustaka yang dirancang untuk mempercepat proses pengembangan sekaligus menjaga kualitas dan konsistensi kode. Salah satu contohnya adalah penggunaan framework JavaScript seperti Vue.js yang memungkinkan pengembangan antarmuka pengguna yang reaktif dan modular, sangat sesuai untuk membangun aplikasi satu halaman (Single Page Application).

Di sisi server, Python menjadi salah satu bahasa yang banyak digunakan karena kesederhanaannya serta dukungan ekosistem yang luas, termasuk dalam pengembangan web, pemrosesan data, dan integrasi basis data. Untuk penyimpanan dan analisis data yang ringan dan efisien, DuckDB muncul sebagai solusi database kolumnar yang dapat dijalankan secara lokal tanpa memerlukan infrastruktur server yang kompleks.

3.2. Vue.js

Vue.js adalah framework JavaScript progresif yang sangat cocok digunakan untuk membangun antarmuka pengguna yang dinamis dan interaktif dalam aplikasi web modern. Dalam konteks pengembangan aplikasi forecasting, Vue.js dapat dimanfaatkan untuk menyajikan hasil peramalan dalam bentuk visualisasi yang responsif, seperti grafik tren, tabel interaktif, atau komponen input untuk pengaturan parameter prediksi. Dengan arsitektur berbasis komponen, Vue.js memudahkan

pengembang untuk memisahkan logika presentasi dari logika bisnis, serta memungkinkan integrasi dengan pustaka visualisasi data seperti Chart.js atau D3.js.

Kemampuan reactive binding Vue.js memungkinkan data hasil perhitungan model peramalan yang dikirim dari backend ditampilkan secara real-time tanpa perlu pemuatan ulang halaman. Selain itu, Vue Router dan Vuex dapat digunakan untuk membangun navigasi halaman yang efisien dan manajemen status aplikasi yang stabil, terutama saat aplikasi memiliki fitur simulasi atau perbandingan hasil forecasting berdasarkan skenario berbeda. Dengan dokumentasi yang baik dan komunitas yang luas, Vue.js mendukung proses pengembangan antarmuka forecasting yang cepat, modular, dan ramah pengguna.

3.3. Python

Python adalah bahasa pemrograman yang sangat populer di bidang data science, termasuk dalam pengembangan model peramalan (forecasting). Dalam konteks ini, Python berperan penting di sisi backend untuk mengelola data historis, membangun model statistik atau machine learning, serta menghitung hasil prediksi berdasarkan parameter yang ditentukan oleh pengguna. Berbagai pustaka seperti NumPy, Pandas, Scikit-learn, Statsmodels, dan Prophet menyediakan alat yang kuat untuk melakukan analisis deret waktu (time series) dan implementasi metode peramalan klasik maupun modern.

Python juga banyak digunakan untuk membangun API (misalnya menggunakan Flask atau FastAPI) yang memungkinkan interaksi antara model peramalan dengan antarmuka pengguna yang dibangun menggunakan Vue.js. Ketika pengguna memasukkan data atau memilih rentang waktu tertentu melalui antarmuka, permintaan dapat diteruskan ke server Python untuk memproses input tersebut dan menghasilkan hasil prediksi. Kemampuan Python dalam mengelola alur data dan memprosesnya secara efisien menjadikannya komponen inti dalam aplikasi forecasting berbasis web yang kompleks.

3.4. DuckDB

DuckDB merupakan sistem manajemen basis data kolumnar yang ringan dan dioptimalkan untuk pemrosesan analitik secara lokal. Dalam pengembangan aplikasi forecasting, DuckDB sangat berguna untuk menyimpan dan mengambil data historis

dalam jumlah besar dengan efisiensi tinggi. Karena DuckDB bersifat in-process dan tidak memerlukan server database eksternal, ia sangat cocok digunakan dalam aplikasi berbasis Python yang melakukan analisis deret waktu secara langsung, termasuk dalam proses pembacaan dataset, agregasi data, dan penyimpanan hasil prediksi.

Dukungan DuckDB terhadap SQL standar memungkinkan pengembang dan analis untuk menulis query yang kompleks untuk menyiapkan data sebelum dilakukan pelatihan atau inferensi model peramalan. Integrasi langsung dengan Pandas juga mempercepat alur kerja, karena hasil kueri dapat langsung digunakan dalam pipeline Python. Keunggulan ini sangat relevan untuk aplikasi forecasting yang bersifat eksploratif atau dashboard-berbasis, di mana kecepatan dan fleksibilitas pemrosesan data sangat dibutuhkan.

3.5. Forecasting

Forecasting atau peramalan merupakan proses memprediksi nilai atau kejadian di masa depan berdasarkan data historis. Teknik ini sangat penting dalam berbagai bidang seperti bisnis, keuangan, pemasaran, dan manajemen rantai pasok. Dalam konteks teknologi informasi, forecasting biasanya dilakukan dengan menggunakan data deret waktu (time series data), di mana pola musiman, tren, dan fluktuasi historis dianalisis untuk menghasilkan prediksi masa depan. Pendekatan forecasting dapat bersifat statistik tradisional maupun berbasis pembelajaran mesin (machine learning), tergantung pada kompleksitas data dan kebutuhan aplikasi.

3.6. Linear Regression

Linear Regression adalah salah satu metode paling dasar dalam statistika dan machine learning untuk forecasting. Model ini memodelkan hubungan linier antara variabel independen dan variabel target. Dalam konteks peramalan, Linear Regression digunakan untuk mengestimasi nilai masa depan berdasarkan satu atau lebih prediktor. Meskipun sederhana, metode ini sangat berguna sebagai baseline dan bekerja cukup baik untuk dataset yang memiliki hubungan linier yang jelas. Namun, Linear Regression memiliki keterbatasan dalam menangani hubungan non-linier dan interaksi kompleks antar fitur.

3.7. Random Forest

Forecasting atau peramalan merupakan proses memprediksi nilai atau kejadian di masa depan berdasarkan data historis. Teknik ini sangat penting dalam berbagai bidang seperti bisnis, keuangan, pemasaran, dan manajemen rantai pasok. Dalam konteks teknologi informasi, forecasting biasanya dilakukan dengan menggunakan data deret waktu (time series data), di mana pola musiman, tren, dan fluktuasi historis dianalisis untuk menghasilkan prediksi masa depan. Pendekatan forecasting dapat bersifat statistik tradisional maupun berbasis pembelajaran mesin (machine learning), tergantung pada kompleksitas data dan kebutuhan aplikasi.

3.8. XGBoost (Extreme Gradient Boosting)

Random Forest adalah algoritma pembelajaran ensemble berbasis decision tree yang bekerja dengan membangun banyak pohon keputusan dan menggabungkan hasilnya melalui rata-rata (untuk regresi) atau voting (untuk klasifikasi). Dalam forecasting, Random Forest mampu menangkap hubungan non-linier dan interaksi antar fitur tanpa memerlukan asumsi distribusi data tertentu. Keunggulannya adalah ketahanan terhadap overfitting dan fleksibilitas dalam menangani data beragam. Namun, ia kurang efisien dibanding model boosting dalam hal performa prediksi ketika dataset sangat besar dan kompleks.

3.9. LightGBM (Light Gradient Boosting Machine)

LightGBM merupakan alternatif XGBoost yang dikembangkan oleh Microsoft, dengan keunggulan utama pada efisiensi dan kecepatan komputasi. Algoritma ini menggunakan teknik *leaf-wise tree growth* dan mendukung histogram-based training yang membuatnya sangat cepat dan ringan dalam penggunaan memori. LightGBM sangat cocok untuk proyek forecasting dengan data berskala besar dan waktu pelatihan yang terbatas. Dalam banyak kasus, LightGBM dapat memberikan akurasi yang setara atau bahkan lebih tinggi dibanding XGBoost, terutama ketika jumlah fitur sangat besar.

3.10. CatBoost

CatBoost adalah algoritma boosting berbasis pohon yang dikembangkan oleh Yandex dan dirancang untuk bekerja secara optimal dengan fitur kategorikal. Dalam konteks forecasting, CatBoost unggul ketika data memiliki banyak variabel non-numerik, karena mampu mengolah fitur kategorikal tanpa memerlukan pra-pemrosesan seperti

one-hot encoding. Selain itu, CatBoost dirancang agar lebih mudah digunakan dan lebih stabil terhadap *overfitting*, sehingga menjadi pilihan yang kuat untuk berbagai kasus prediksi dengan data kompleks dan bervariasi.

3.11. Metode Ensemble

Metode ensemble merupakan pendekatan dalam pembelajaran mesin yang menggabungkan beberapa model dasar (base learners) untuk meningkatkan akurasi dan stabilitas prediksi. Salah satu teknik ensemble yang umum digunakan adalah voting, di mana hasil prediksi dari berbagai model digabungkan untuk menghasilkan keputusan akhir. Dalam konteks regresi, voting dapat dilakukan melalui rata-rata sederhana (simple averaging) atau rata-rata berbobot (weighted averaging). Pendekatan rata-rata berbobot memberikan bobot lebih besar kepada model dengan performa prediksi yang lebih baik, sehingga kontribusinya dalam prediksi akhir menjadi lebih dominan. Misalnya, Chen dan Zheng (2025) mengusulkan metode RRMSE Voting Regressor yang menetapkan bobot berdasarkan Relative Root Mean Square Error (RRMSE) dari masing-masing model, menunjukkan peningkatan akurasi prediksi dibandingkan metode ensemble lainnya. Selain itu, Echtenbruck et al. (2022) mengembangkan pendekatan optimasi bobot menggunakan pemrograman kuadratik konveks untuk kombinasi model regresi heterogen, yang terbukti lebih efisien dan akurat. Pendekatan ini juga telah diterapkan dalam berbagai bidang, seperti peramalan permintaan makanan (Denis & Keerthana, 2024) dan prediksi limbah medis (Yildiz & Yilmaz, 2022), menunjukkan efektivitas metode voting regresi berbobot dalam meningkatkan kinerja prediksi di berbagai domain.

3.12. Arsitektur Client-Server

Arsitektur client-server merupakan model komunikasi dalam sistem terdistribusi yang memisahkan tanggung jawab antara dua entitas utama: klien (client) dan server. Dalam model ini, klien berperan sebagai pihak yang mengirimkan permintaan layanan (request), sedangkan server bertugas merespons permintaan tersebut dengan menyediakan data atau menjalankan proses tertentu. Arsitektur ini menjadi pondasi utama dalam pengembangan aplikasi web modern, termasuk aplikasi forecasting yang membutuhkan interaksi antara antarmuka pengguna dan sistem backend pengolahan data.

Pada umumnya, klien dijalankan di sisi pengguna melalui web browser atau aplikasi front-end, dan dibangun dengan teknologi seperti HTML, CSS, dan JavaScript, termasuk framework modern seperti Vue.js. Klien bertanggung jawab atas penyajian antarmuka serta interaksi pengguna. Sementara itu, server dijalankan di sisi backend, biasanya menggunakan bahasa pemrograman seperti Python, yang menangani logika bisnis, pengolahan data, dan manajemen basis data.

Komunikasi antara client dan server umumnya dilakukan melalui protokol HTTP/HTTPS dengan format data seperti JSON atau XML. Server dapat juga mengimplementasikan API berbasis REST atau GraphQL untuk mengatur alur pertukaran data. Dalam konteks aplikasi forecasting, server akan memproses data historis, menjalankan model peramalan (misalnya XGBoost atau Linear Regression), lalu mengembalikan hasil prediksi ke klien untuk ditampilkan.

Keunggulan utama dari arsitektur client-server adalah skalabilitas, pemisahan tanggung jawab yang jelas antara logika presentasi dan logika bisnis, serta kemudahan dalam pemeliharaan sistem. Model ini juga memungkinkan fleksibilitas dalam pengembangan dan pengujian masing-masing komponen secara independen. Dalam aplikasi forecasting berbasis web, arsitektur ini sangat penting untuk memastikan antarmuka pengguna tetap ringan dan responsif, sementara komputasi berat dilakukan secara efisien di sisi server.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem

4.1.1 Definisi Umum Sistem

Secara umum, sistem ini bekerja dengan melakukan pelatihan model machine learning berdasarkan data historis penjualan kemudian melakukan prediksi pada periode kedepannya. Agar memudahkan dalam penggunaan sistem ini, dibuat aplikasi berbasis web.

4.1.2 Analisis Kebutuhan

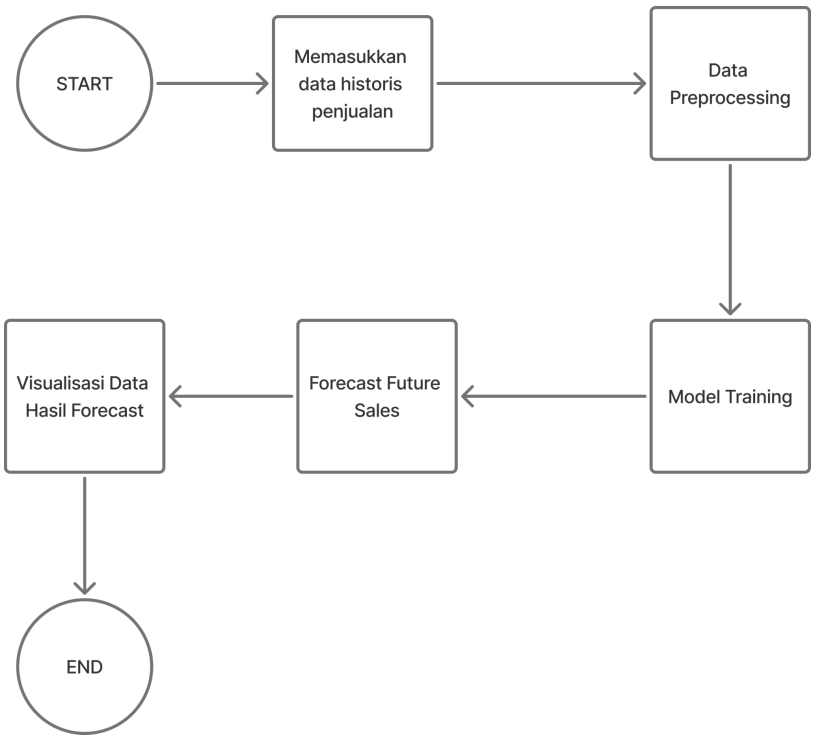
Dalam sistem ini, terdapat beberapa analisis kebutuhan yang didapatkan setelah melakukan wawancara dan meeting dengan tim HRD PT. UBS, dimana tim dr PT.UBS menjelaskan apa saja yang harus dilakukan pengguna sistem. Kebutuhan pada aplikasi dijelaskan pada tabel sebagai berikut.

Tabel 4.1 Deskripsi Kebutuhan Aplikasi

Kode Kebutuhan	Deskripsi Kebutuhan
C01	Melakukan forecasting penjualan
C02	Melihat visualisasi hasil forecasting dan data asli
C03	Menambah data historis penjualan
C04	Mengunduh data hasil forecasting

4.1.3 Flowchart Proses Bisnis

Proses bisnis adalah serangkaian aktivitas yang mendefinisikan bagaimana aplikasi berfungsi. Pada kerja praktik ini, proses bisnis untuk aplikasi ini mengikuti *pipeline* umum dari sistem prediksi menggunakan algoritma machine learning. Flowchart proses bisnis dari sistem ini dapat dilihat pada gambar berikut.

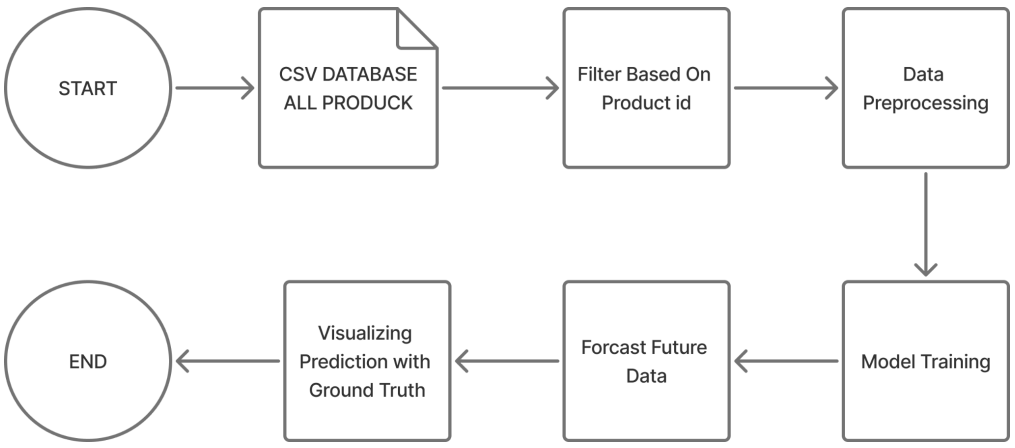


Gambar 4.1 Flowchart Proses Bisnis

4.2. Perancangan Sistem

4.2.1 Flowchart Umum Sistem Forecasting

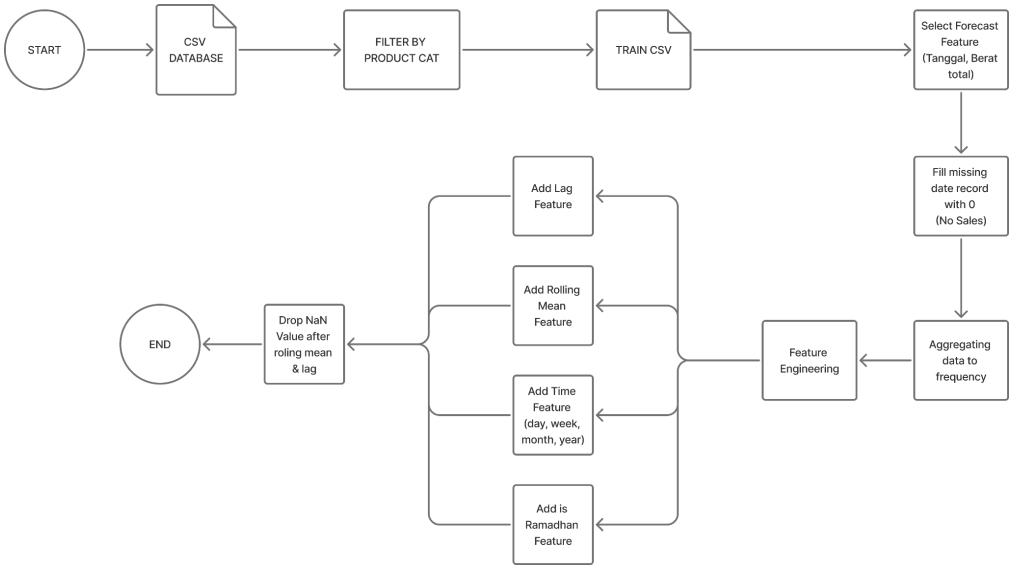
Dari flowchart yang diberikan oleh PT. UBS, penulis mencoba untuk membuat flowchart sistem yang lebih terdefinisi pada aplikasi. Berikut adalah gambar untuk flowchart sistem.



Gambar 4.2 Flowchart Umum Sistem Forecasting

4.2.2 Flowchart Data Preprocessing

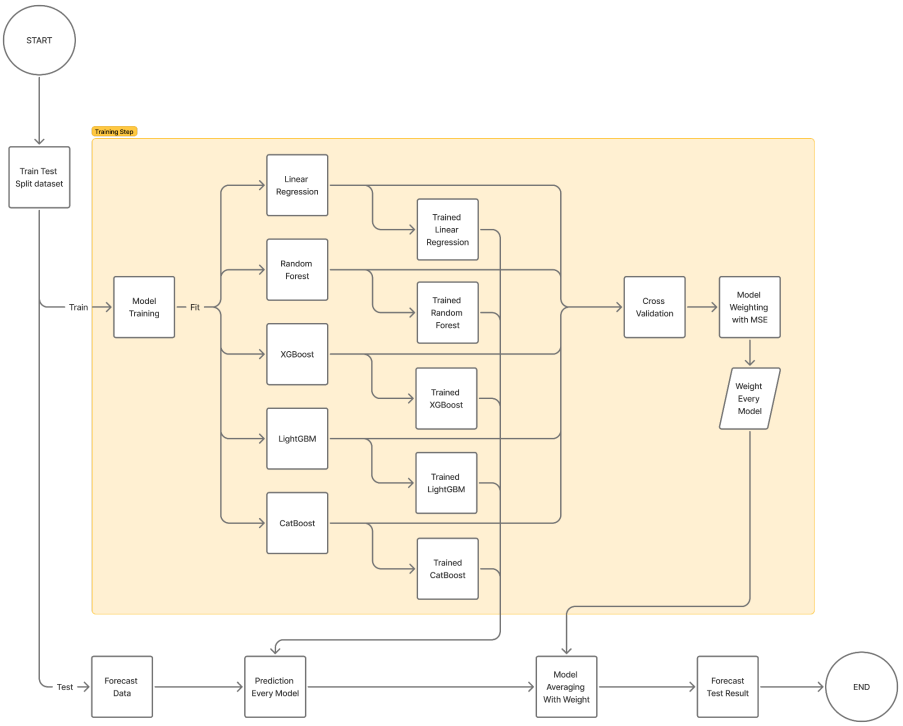
Dari Generalisasi Flowchart umum sistem *forecasting*, untuk bagian data preprocessing dapat dijabarkan lebih detail pada diagram berikut.



Gambar 4.3 Flowchart Data Preprocessing

4.2.3 Flowchart Model Training

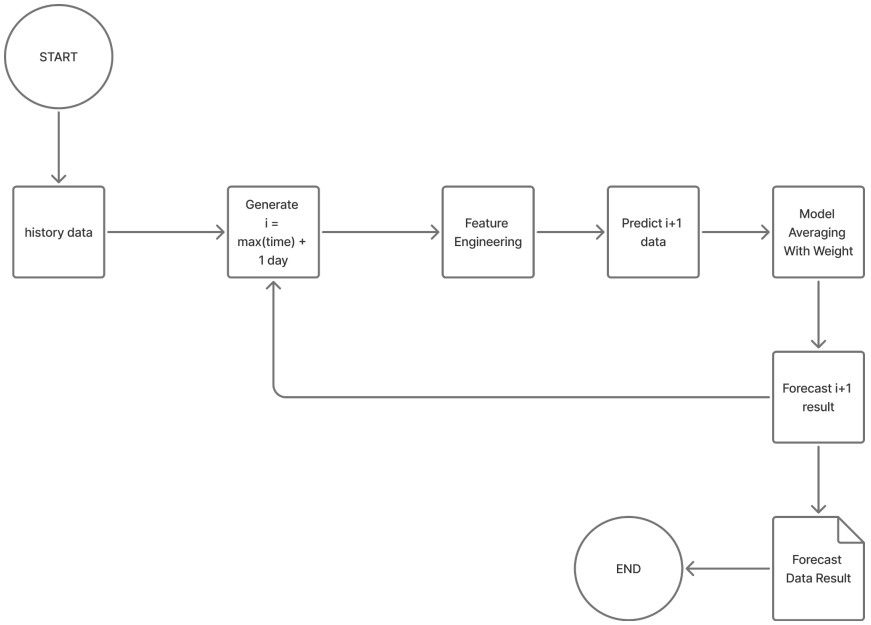
Dari Generalisasi Flowchart umum sistem *forecasting*, untuk bagian *model training* dapat dijabarkan lebih detail pada diagram berikut.



Gambar 4.4 Flowchart Model Training

4.2.4 Flowchart Forecast Future Data

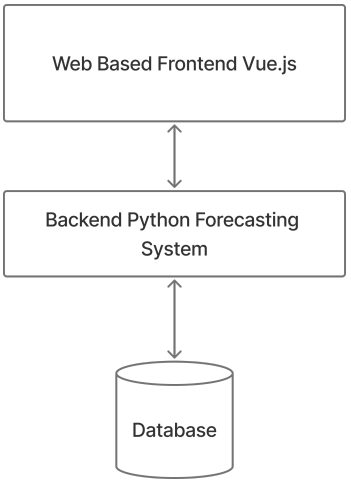
Dari Generalisasi Flowchart umum sistem forecasting, untuk bagian Forecast Future Data dapat dijabarkan lebih detail pada diagram berikut.



Gambar 4.5 Flowchart Forecast Future Data

4.2.5 Diagram Arsitektur Sistem

Karena dibutuhkan aplikasi berbasis web sebagai antarmuka pengguna dengan model, dibuatlah aplikasi berbasis web dengan arsitektur client server untuk memudahkan dalam proses development aplikasi. Sehingga user tidak perlu hanya perlu berinteraksi dengan frontend dari website. Gambaran dari arsitektur yang dipakai dalam sistem ini sebagai berikut.



Gambar 4.6 Diagram Arsitektur Sistem

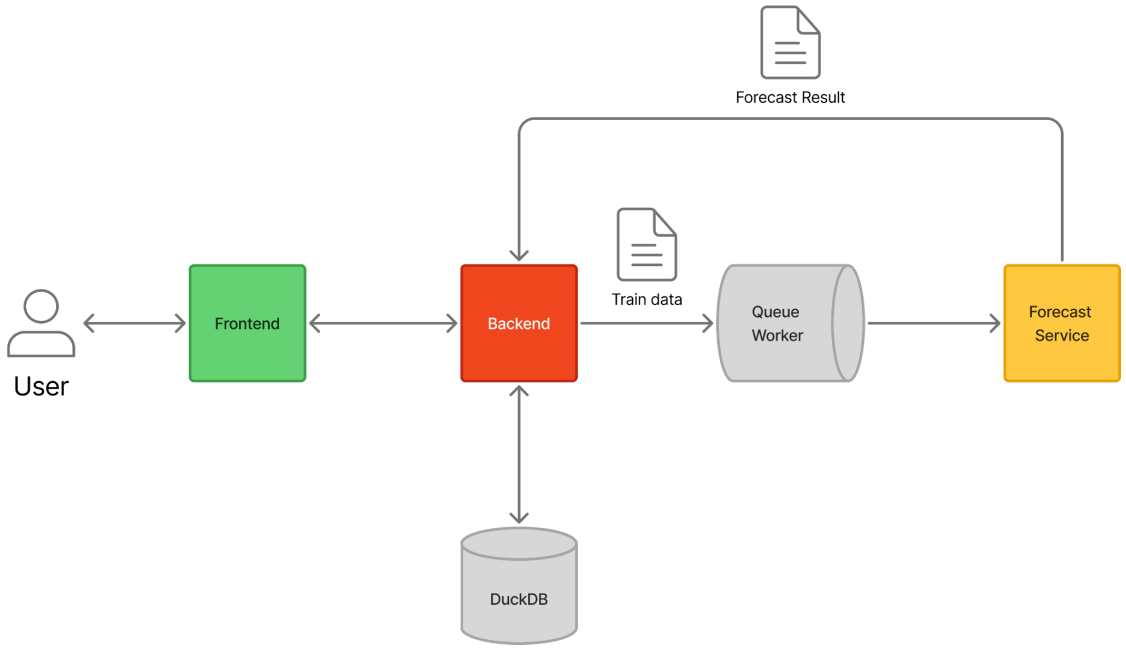
4.2.6 Desain Antarmuka Aplikasi

Setelah menganalisis kebutuhan sistem, maka langkah selanjutnya adalah mendesain tampilan untuk fungsi utama dari aplikasi, yakni memasukkan data historis penjualan. Website dibuat sesimple mungkin dengan memperhatikan kebutuhan dari user. Sehingga diperlukan minimal 4 halaman dengan rincian :

1. Halaman untuk memasukkan data historis penjualan
2. Halaman untuk melakukan forecasting data sesuai produk yang dipilih
3. Halaman untuk melakukan visualisasi pada data hasil prediksi dan data sebenarnya
4. Halaman untuk melakukan unduh data hasil prediksi

BAB V
IMPLEMENTASI SISTEM

5.1. Implementasi Dari Perancangan Sistem Forecast

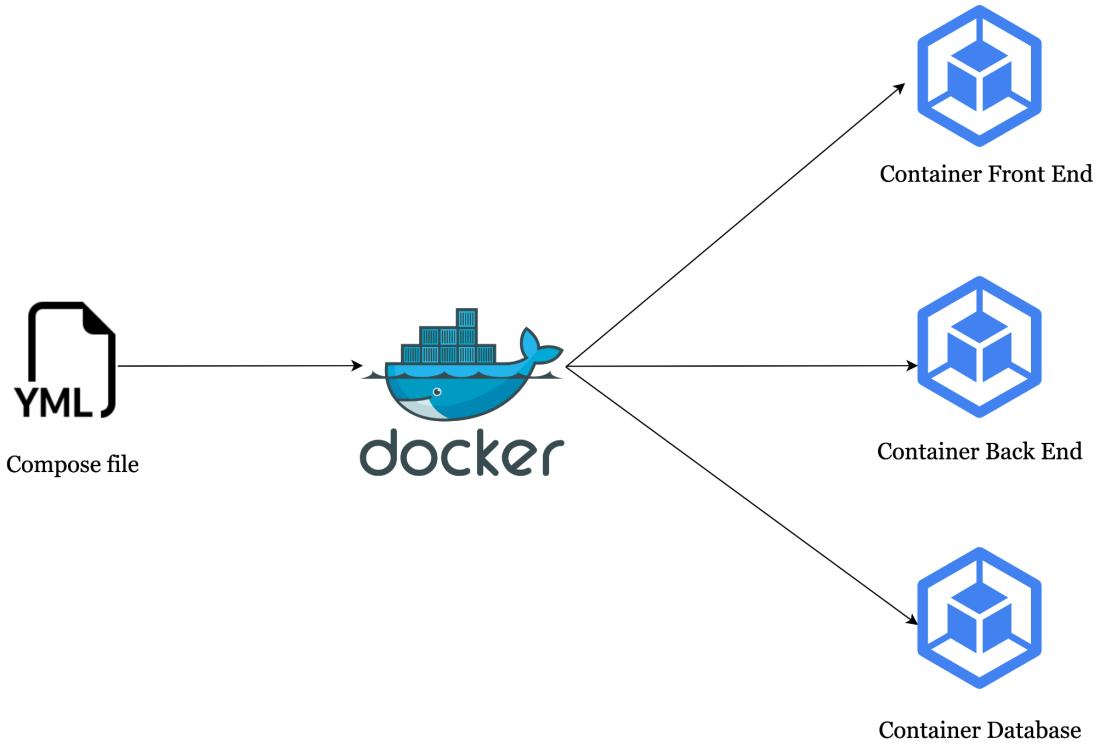


Gambar 5.1 Diagram Perancangan Sistem Forecast

Aplikasi ini dibangun dengan memisahkan service forecast dan mengimplementasikan queue worker untuk menangani task yang perlu diproses lama. Pendekatan ini memungkinkan client tidak perlu mempertahankan koneksi dengan server pada waktu forecasting sales. Interaksi dimulai dari pengguna yang mengakses sistem melalui Frontend Vue.js, yang berfungsi sebagai antarmuka utama. Segala bentuk input, baik itu permintaan data, pengunggahan dokumen, ataupun pemicu untuk proses prediksi, akan diterima oleh Frontend dan diteruskan ke Backend. Komponen Backend ini bertindak sebagai pusat kendali dan logika bisnis aplikasi. Untuk permintaan yang bersifat ringan dan membutuhkan akses data cepat, seperti menampilkan informasi historis, Backend akan langsung berinteraksi dengan database DuckDB yang tersimpan. Namun, untuk tugas-tugas yang lebih kompleks dan memakan waktu, misalnya pemrosesan dokumen yang baru saja diunggah atau persiapan data untuk analisis prediktif yang rumit, Backend akan mendelegasikannya agar tidak mengganggu responsivitas sistem secara keseluruhan di mata pengguna.

Tugas-tugas berat tersebut akan dimasukkan ke dalam sebuah antrian (queue) yang kemudian dikelola oleh Queue Worker. Mekanisme antrian ini krusial untuk memastikan Backend tetap dapat merespons permintaan lain dengan cepat dan pengguna merasakan aplikasi yang tetap gesit. Queue Worker akan mengambil tugas dari antrian secara berurutan dan memprosesnya secara terpisah (asynchronous), artinya proses ini berjalan di latar belakang tanpa membuat pengguna menunggu langsung. Proses yang dilakukan oleh Worker ini seringkali berupa persiapan data, validasi, atau transformasi yang diperlukan sebelum data siap diolah lebih lanjut oleh Forecast Service. Layanan inilah yang secara khusus didesain untuk menjalankan algoritma atau model prediksi berdasarkan data yang telah matang disiapkan. Setelah proses prediksi selesai, hasil dari Forecast Service berupa angka atau informasi peramalan akan disimpan kembali ke dalam DuckDB untuk keperluan analisis lebih lanjut atau pencatatan, dan juga dapat dikirimkan kembali ke Backend untuk kemudian ditampilkan secara informatif kepada pengguna melalui Frontend.

5.2. Implementasi Deployment



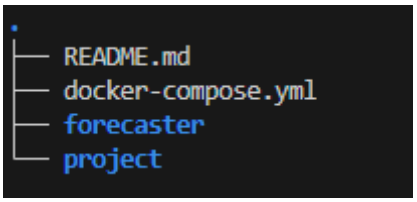
Gambar 5.2 Diagram Docker

Dalam kerangka strategis implementasi menuju fase deployment, sebuah pendekatan komprehensif berupa kontainerisasi diterapkan untuk mengenkapsulasi

komponen-komponen fundamental aplikasi, yakni frontend dan backend. Eksekusi teknis dari inisiatif kontainerisasi tersebut dimanifestasikan melalui pemanfaatan platform Docker sebagai fondasi teknologi, yang diperkuat dengan utilitas Docker Compose guna mengatur orkestrasi serta konfigurasi beragam instans kontainer secara terpadu dan deklaratif, sebagaimana terdokumentasi dalam artefak konfigurasi docker-compose.yml. Konsekuensi logis dari adopsi metodologi ini adalah tercapainya tingkat portabilitas aplikasi yang superior, sehingga memungkinkan eksekusi yang konsisten dan reliabel lintas beragam lanskap sistem operasi, dengan prasyarat esensial tunggal berupa keberadaan runtime Docker yang fungsional pada infrastruktur host.

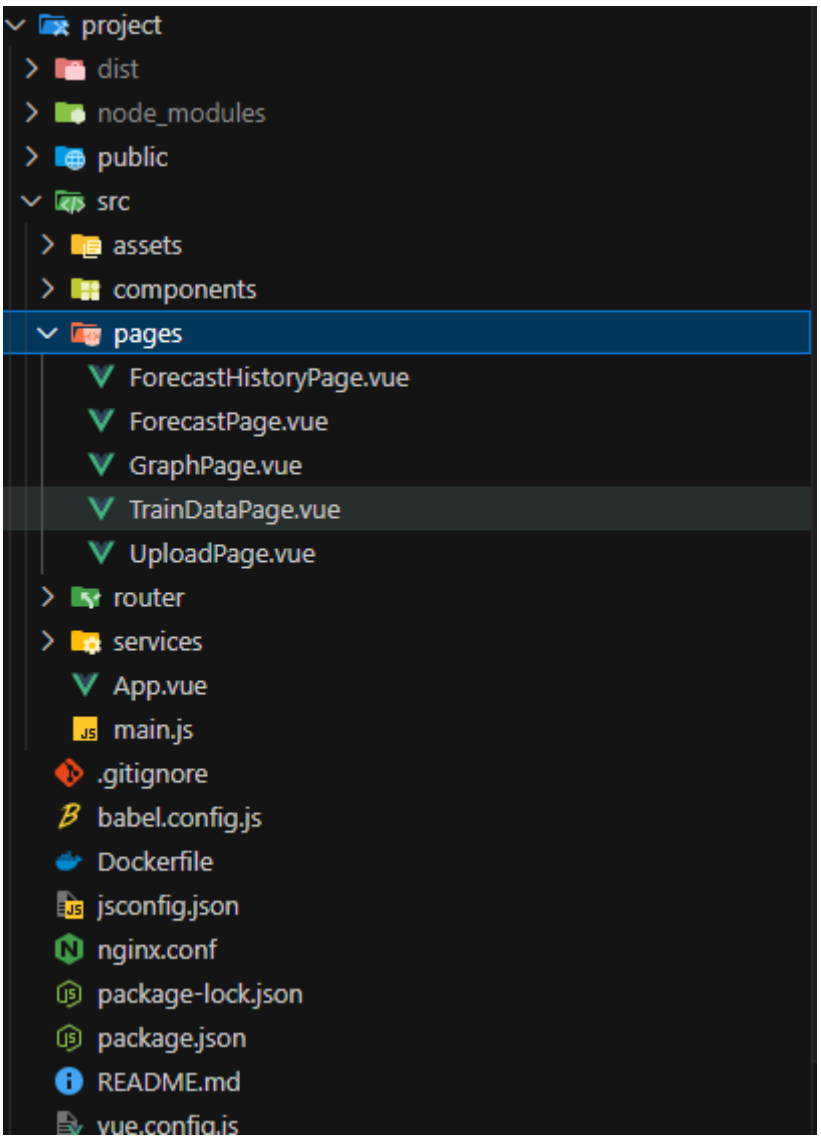
5.3. Struktur Kerangka Kode Sumber

Kode sumber menggunakan Javascript sebagai sisi client/frontend dan python sebagai server/backend. Berikut adalah gambaran mengenai struktur kode sumber dari aplikasi.



Gambar 5.3 Struktur Kode

Pada folder project merupakan client side yang merupakan kode berisi file javascript untuk mengimplementasikan frontend dengan menggunakan framework Vue.js. Sedangkan pada folder forecaster merupakan kode berisi file python untuk mengimplementasikan backend dari sistem ini.



Gambar 5.4 Struktur Kode Folder Project

Berikut adalah penjelasan masing-masing folder yang ada pada folder *application*:

1. public/

   Berisi file statis seperti index.html dan gambar logo.

   a. index.html adalah entry point aplikasi web.

2. src/

   Folder utama untuk source code aplikasi Vue.js:

   a. main.js

      Entry point JavaScript yang menginisialisasi Vue, mengimpor global style, dan mount ke elemen #app.

   b. App.vue

      Komponen root utama aplikasi.

3. assets/

   Berisi aset seperti CSS global (global.css) dan gambar.

4. components/

   Komponen Vue yang dapat digunakan ulang, misal Forecast.vue dan Header.vue.

5. pages/

   Halaman-halaman utama aplikasi, masing-masing biasanya mewakili satu route, seperti:

   a. GraphPage.vue: Halaman utama grafik forecast.
   b. UploadPage.vue: Halaman upload file.
   c. ForecastHistoryPage.vue: Riwayat forecast.
   d. ForecastPage.vue: Proses forecasting.
   e. TrainDataPage.vue: Data pelatihan.

6. router/

   Konfigurasi routing aplikasi.

7. services/

   Berisi utilitas untuk komunikasi API, misal api.js.

8. package.json

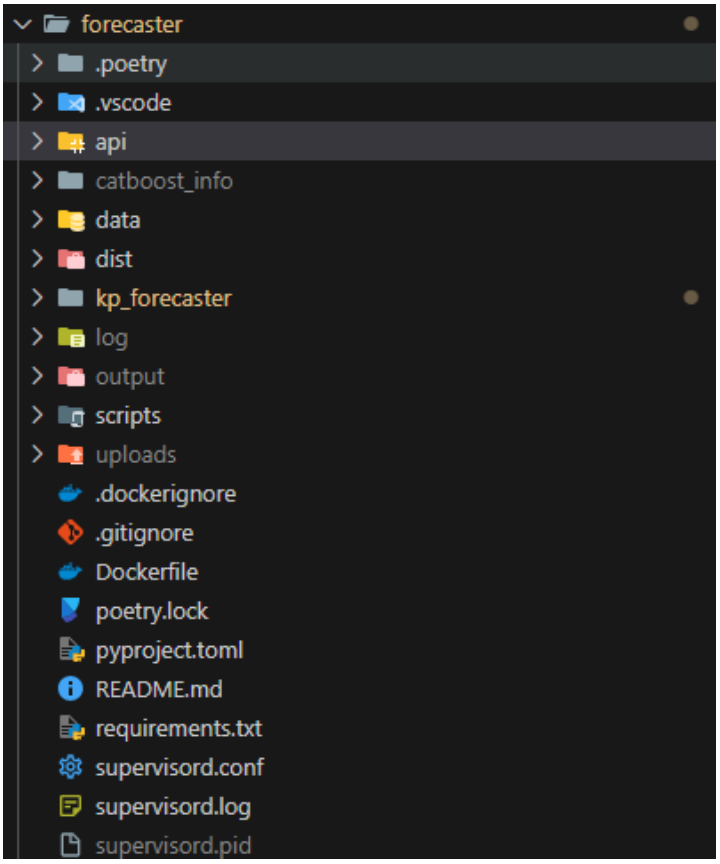
   Konfigurasi dependensi dan script npm.

9. vue.config.js

   Konfigurasi build untuk Vue CLI.

10. babel.config.js

    Konfigurasi Babel untuk transpile JavaScript.



Gambar 5.5 Struktur Kode Folder Forecaster

1. api/

   Menyimpan kode backend API (FastAPI) dan logika terkait.

   - main.py: Entry point aplikasi FastAPI.
   - endpoints.py: Mendefinisikan endpoint API (upload, forecast, dsb).
   - db.py: Koneksi database, query, dan utilitas penyimpanan data.
   - middleware.py: Middleware untuk API (misal CORS).
   - tasks.py: Definisi task Celery untuk proses asynchronous.

2. kp_forecaster/

   Menyimpan kode utama untuk pipeline forecasting.

3. data/

   Menyimpan data mentah dan hasil preprocessing.

   - processed.csv: Data hasil preprocessing.
   - train.duckdb: Database DuckDB untuk penyimpanan data.
   - train.duckdb.wal: Write-ahead log DuckDB.

4. log/

   Menyimpan file log aplikasi.

5. output/

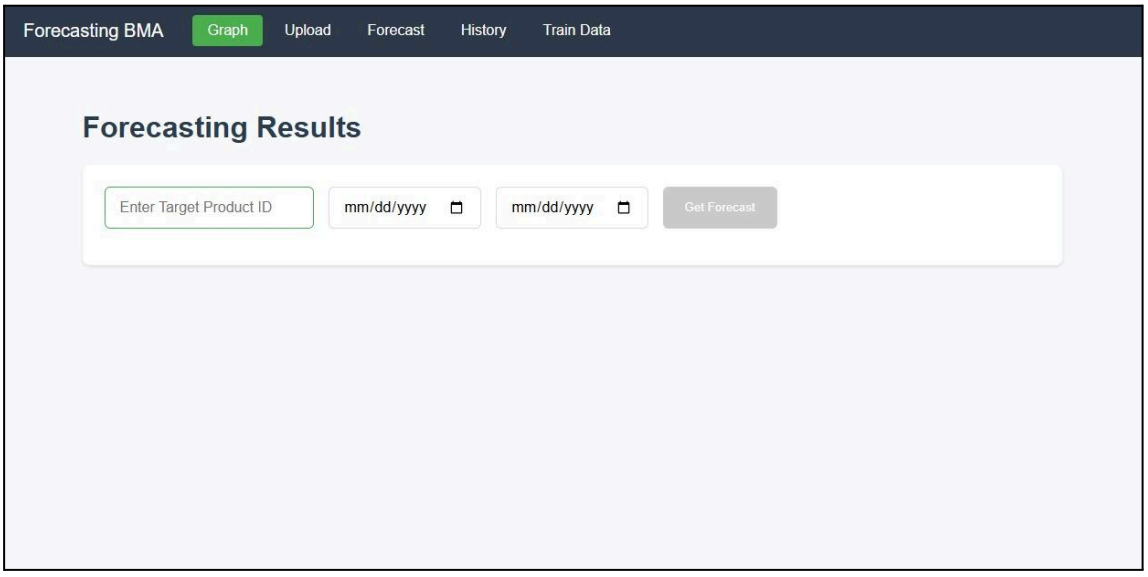
Menyimpan hasil output forecasting (CSV, gambar plot, dsb).

6. scripts/
   Menyimpan script utilitas untuk testing atau maintenance (misal: test_db.py).
7. uploads/
   Menyimpan file yang di-upload oleh user.
8. .dockerignore, Dockerfile
   Konfigurasi Docker untuk containerisasi aplikasi.
9. poetry.lock, pyproject.toml, requirements.txt
   Konfigurasi dependensi dan environment Python.
10. supervisord.conf, supervisord.log, supervisord.pid
    Konfigurasi dan file runtime untuk process manager Supervisor.

5.4. Implementasi Antarmuka Pengguna

Berikut adalah implementasi antarmuka pengguna yang sudah dibuat oleh penulis.

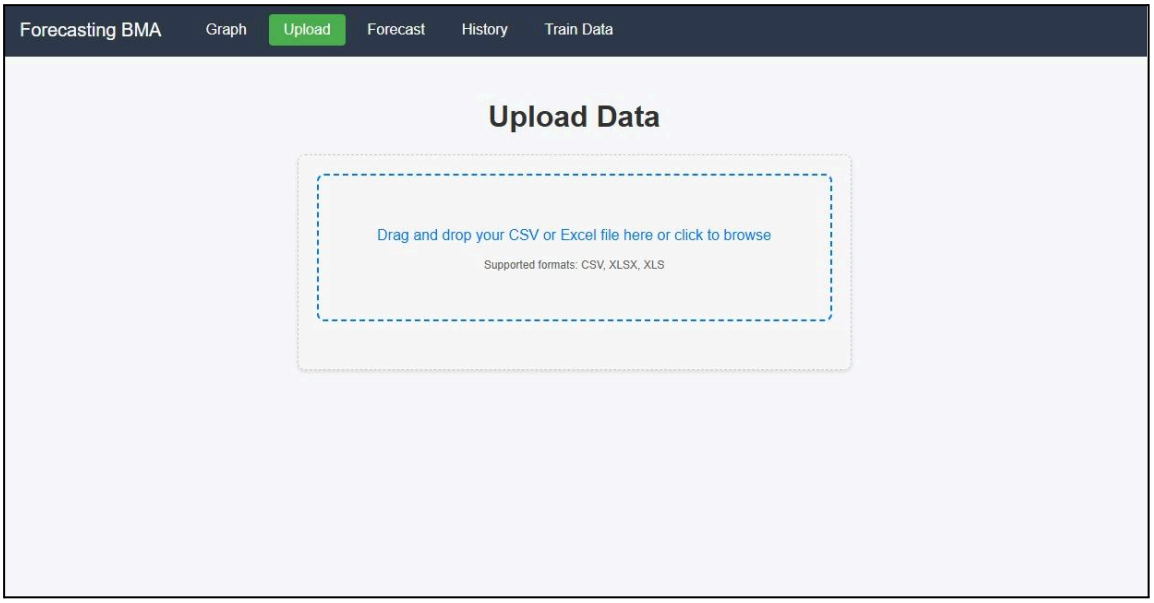
1. Halaman Graph
   Halaman ini menampilkan Grafik dari suatu forecast yang telah dilakukan sebelumnya dengan memasukan *Target Product Id* dan tanggal awal dan akhir dari forecast



Gambar 5.6 Halaman Graph

2. Halaman Mengunggah Data

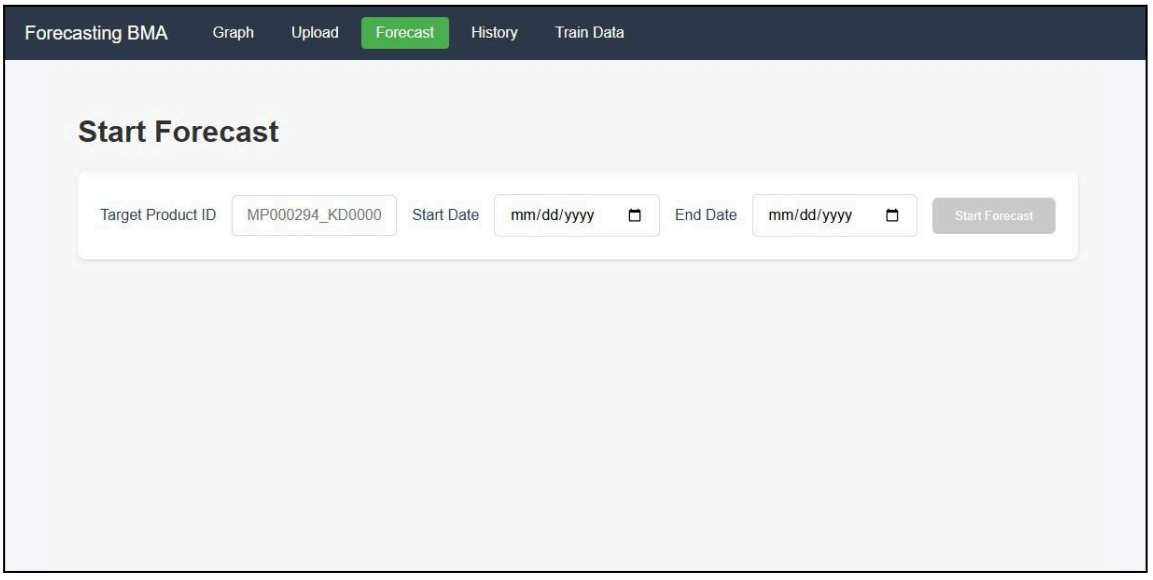
Halaman ini digunakan untuk mengunggah data dalam bentuk .csv atau .xlsx yang akan dimasukan kedalam database.



Gambar 5.7 Halaman Upload Data

3. Halaman Forecast

Halaman ini akan digunakan untuk memulai forecasting dengan memasukan *Target Product ID, Start Date,* dan *End Date*.



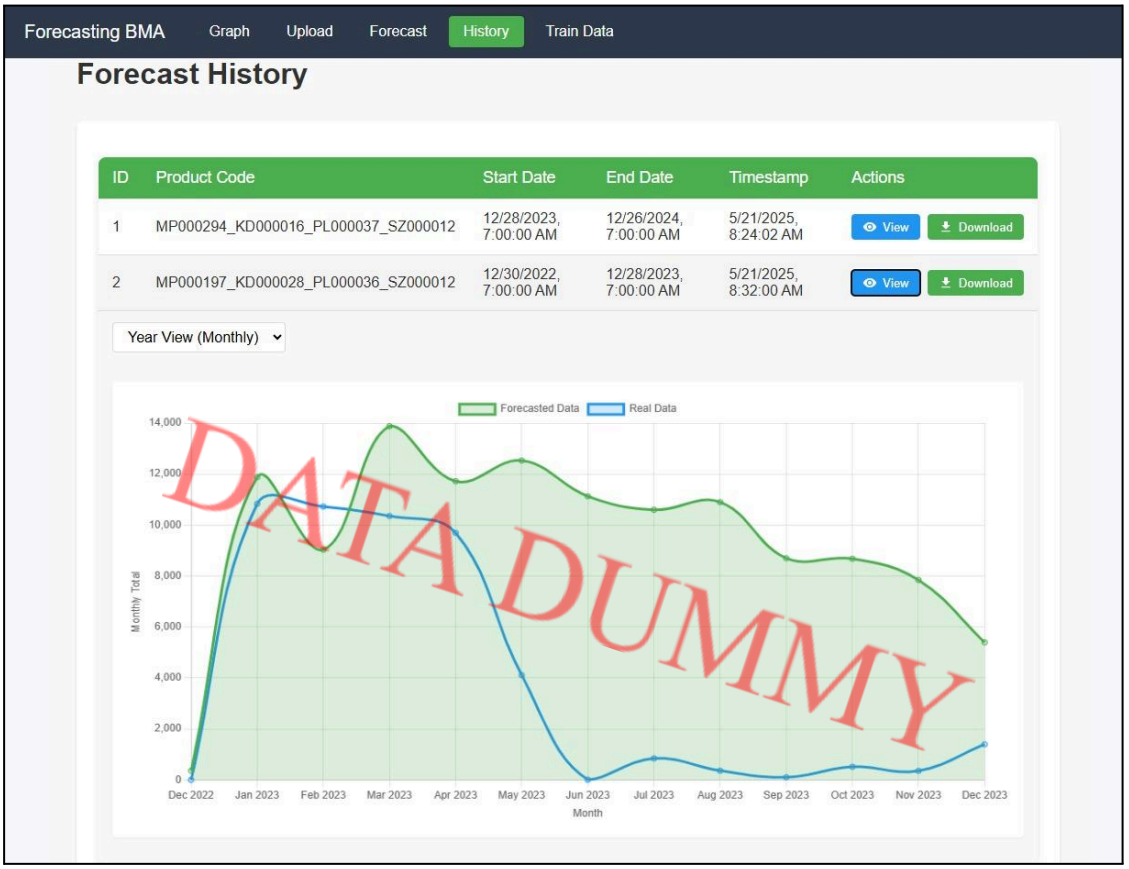
Gambar 5.8 Halaman Forecast

4. Halaman History

Halaman ini berfungsi untuk menampilkan riwayat hasil forecasting yang telah dilakukan oleh pengguna. Setiap entri riwayat mencakup informasi seperti *Product Code, Start Date, End Date*, serta *Timestamp* yang menunjukkan waktu proses dilakukan. Terdapat dua tombol utama di kolom *Actions*:

- *View*: untuk melihat grafik perbandingan antara data hasil prediksi (*Forecasted Data*) dan data aktual (*Real Data*).
- *Download*: untuk mengunduh hasil forecasting dalam format file.

Di bagian bawah halaman, terdapat grafik visualisasi yang menampilkan tren hasil prediksi dibandingkan dengan data nyata berdasarkan periode waktu (bulanan), yang dapat diatur melalui dropdown "*Year View (Monthly)*". Grafik ini membantu pengguna untuk menganalisis akurasi prediksi serta memantau pola permintaan dari waktu ke waktu.



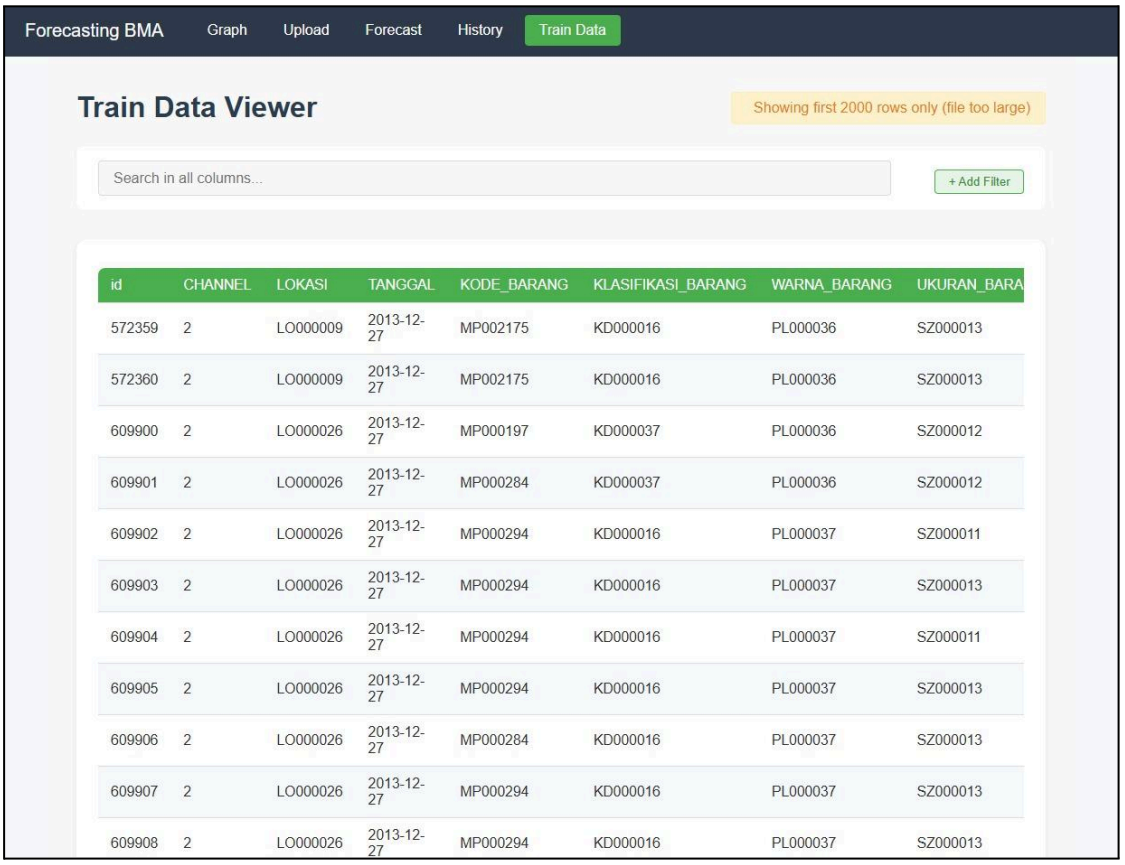
Gambar 5.9 Halaman History

5. Halaman Train Data.

Halaman ini berfungsi untuk menampilkan data pelatihan (*train data*) yang telah diunggah oleh pengguna dan digunakan dalam proses *forecasting*. Data ini berisi informasi historis seperti tanggal, kode barang, klasifikasi barang, warna, ukuran, lokasi, dan channel penjualan. Fitur-fitur yang tersedia di halaman ini meliputi:

- Kolom pencarian: memungkinkan pengguna untuk mencari data tertentu berdasarkan kata kunci di semua kolom.
- *Filter data (+ Add Filter)*: untuk menyaring data berdasarkan kriteria spesifik.

Halaman ini penting untuk memverifikasi apakah data yang diunggah telah masuk dengan benar dan lengkap sebelum digunakan dalam proses prediksi di sistem *forecasting*.



Gambar 5.10 Halaman Train Data

BAB VI
PENGUJIAN DAN EVALUASI

6.1 Tujuan Pengujian

Pengujian dilakukan kepada aplikasi UBS Forecasting dengan persona sebagai user. Aplikasi UBS Forecasting dilakukan untuk menguji kesesuaian fungsionalitas aplikasi dengan semua kebutuhan yang dibutuhkan oleh pengguna dari PT. UBS.

6.2 Kriteria Pengujian

Penilaian pengujian dari aplikasi UBS Forecasting harus memenuhi beberapa aspek sebagai berikut:

1. Kemampuan aplikasi untuk menampilkan semua data tabel penjualan sudah ada.
2. Kemampuan aplikasi untuk mencari data tabel penjualan berdasarkan keyword yang dimasukkan.
3. Kemampuan aplikasi untuk menambah data penjualan baru dengan lengkap ke dalam database.
4. Kemampuan aplikasi untuk melakukan forecast untuk jangka waktu kedepan dengan akurat.
5. Kemampuan aplikasi untuk melihat sejarah forecast yang telah dilakukan.
6. Kemampuan aplikasi untuk menunjukan data dalam bentuk grafik yang jelas.

6.3 Skenario Pengujian

Skenario pengujian sistem pengguna dilakukan melalui pengujian internal oleh penulis dan presentasi kepada tim IT PT. UBS. Berikut adalah langkah pengujian yang dijalankan pada aplikasi UBS Forecasting.

1. Pengguna mengunggah data yang diperlukan untuk proses forecasting melalui halaman upload.
2. Setelah proses unggah selesai, pengguna membuka halaman "Train Data" untuk memastikan bahwa data yang diunggah telah masuk dan ditampilkan dengan benar.
3. Pengguna kemudian memasukkan tanggal yang diinginkan untuk melakukan proses forecasting, sesuai dengan periode prediksi yang dibutuhkan.

4. Sistem akan memproses data dan menghasilkan hasil prediksi berdasarkan tanggal yang telah dimasukkan.
5. Setelah proses selesai, pengguna membuka halaman "History" untuk melihat hasil forecasting yang telah dilakukan, termasuk data prediksi dan waktu proses.
6. Pengguna dapat memverifikasi bahwa hasil prediksi sesuai dengan input yang telah diberikan dan digunakan untuk analisis lebih lanjut atau perencanaan ke depan.

6.4 Evaluasi Pengujian

Setelah dilakukan skenario pengujian, berikut adalah hasil dari pengujian yang dilakukan.

Tabel 6.1 Tabel Kriteria dan Hasil Pengujian

Kriteria Pengujian	Hasil Pengujian
Kemampuan aplikasi untuk menampilkan semua data tabel penjualan sudah ada.	Terpenuhi
Kemampuan aplikasi untuk mencari data tabel penjualan berdasarkan keyword yang dimasukkan.	Terpenuhi
Kemampuan aplikasi untuk menambah data penjualan baru dengan lengkap ke dalam database.	Terpenuhi
Kemampuan aplikasi untuk melakukan forecast untuk jangka waktu kedepan dengan akurat.	Terpenuhi
Kemampuan aplikasi untuk melihat sejarah forecast yang telah dilakukan.	Terpenuhi
Kemampuan aplikasi untuk menunjukan data dalam bentuk grafik yang jelas.	Terpenuhi

BAB VII
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Setelah melakukan kerja praktik di PT. UBS, penulis dapat mengambil beberapa kesimpulan :

1. Telah berhasil dirancang dan diimplementasikan sebuah sistem aplikasi berbasis web untuk melakukan forecasting penjualan emas. Sistem ini menerapkan beberapa algoritma machine learning yaitu XGBoost, CatBoost, Linear Regression, LightGBM, dan Random Forest, yang dikombinasikan menggunakan metode Ensemble untuk meningkatkan akurasi prediksi.
2. Aplikasi yang dikembangkan menyediakan fungsionalitas bagi pengguna untuk memasukkan data historis penjualan, melakukan proses forecasting untuk periode mendatang, serta memvisualisasikan hasil prediksi dibandingkan dengan data aktual. Hal ini sejalan dengan tujuan untuk membantu PT. UBS meningkatkan efisiensi perencanaan kebutuhan emas.
3. Penggunaan arsitektur *client-server* dengan Vue.js sebagai *frontend*, Python sebagai *backend*, dan DuckDB sebagai basis data, serta pemanfaatan Docker untuk *deployment*, memungkinkan pengembangan sistem yang modular, portabel, dan mudah dikelola. Penerapan *queue worker* juga membantu dalam menangani tugas forecasting yang membutuhkan waktu proses lama secara efisien.
4. Sistem forecasting yang dibangun berpotensi memberikan manfaat bagi PT. UBS dalam hal meningkatkan akurasi prediksi penjualan, mempercepat pengambilan keputusan, dan membantu mengurangi biaya yang timbul akibat kelebihan atau kekurangan stok emas.

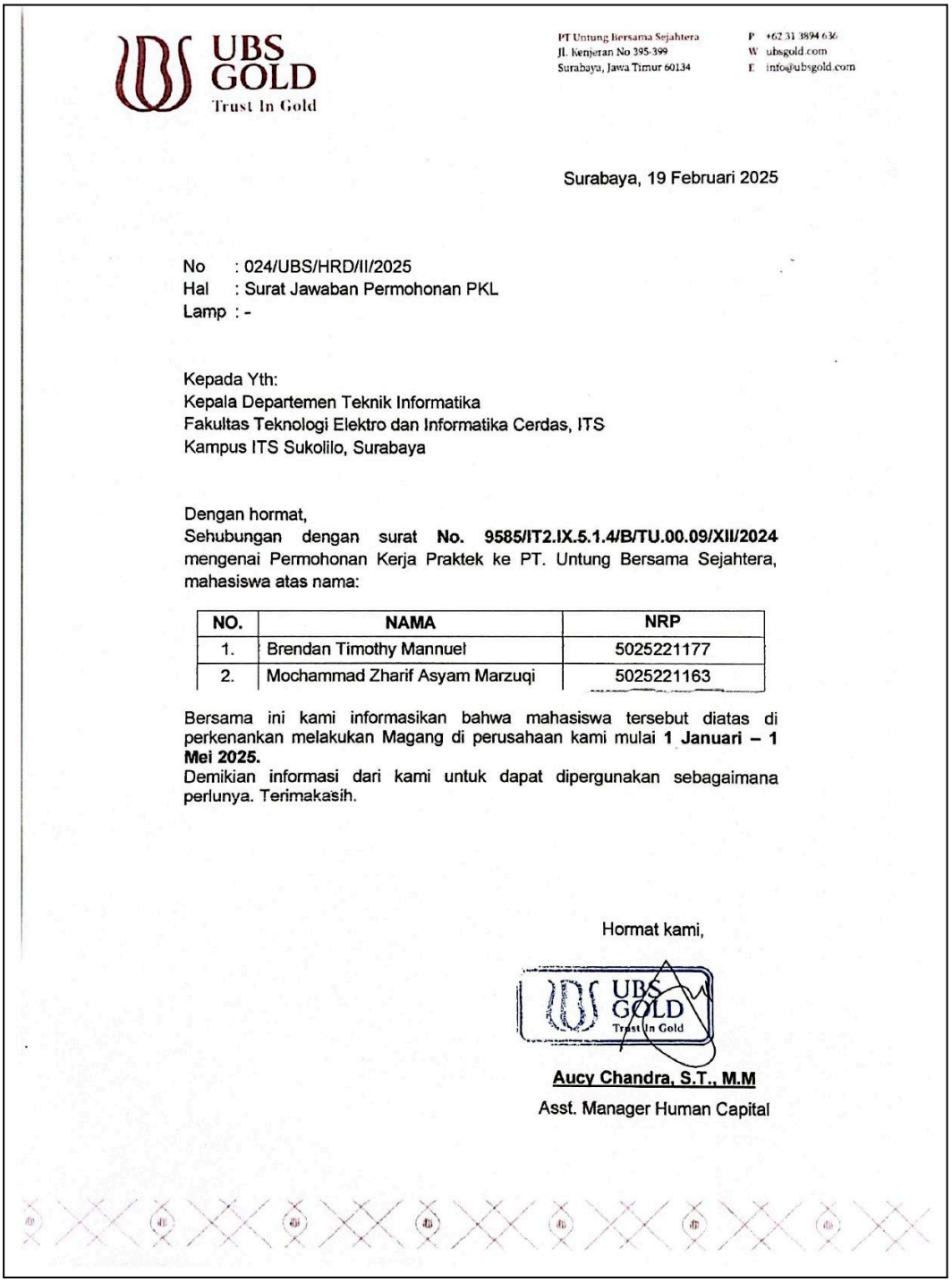
7.2 Saran

Setelah mengikuti kerja praktik di PT. UBS, beberapa saran yang dapat diberikan oleh penulis adalah sebagai berikut :

1. Fokus pada peningkatan berkelanjutan akurasi model dengan mengintegrasikan data eksternal yang relevan, melakukan *feature engineering*, menerapkan *retraining* model secara rutin, dan menjajaki penggunaan algoritma *machine learning* atau *deep learning* yang lebih mutakhir untuk hasil prediksi yang lebih presisi.
2. Kembangkan fungsionalitas aplikasi dengan menambahkan alat analisis yang lebih mendalam seperti analisis sensitivitas dan dasbor interaktif, serta upayakan integrasi sistem forecasting ini dengan sistem manajemen inventaris atau ERP perusahaan untuk meningkatkan efisiensi operasional dan aliran data.
3. Pastikan keberlanjutan dan optimalisasi penggunaan sistem melalui *monitoring* performa model secara berkala, penyediaan dokumentasi dan pelatihan pengguna yang efektif, serta pemeliharaan sistem dengan melakukan pembaruan teknologi dan pustaka secara reguler.

LAMPIRAN

1. Bukti Penerimaan Kerja Praktik



UBS GOLD
Trust In Gold

PT Untung Bersama Sejahtera
Jl. Kenjeran No 395-399
Surabaya, Jawa Timur 60134

P +62 31 3894 636
W ubsgold.com
E info@ubsgold.com

Surabaya, 19 Februari 2025

No : 024/UBS/HRD/II/2025
Hal : Surat Jawaban Permohonan PKL
Lamp : -

Kepada Yth:
Kepala Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Kampus ITS Sukolilo, Surabaya

Dengan hormat,
Sehubungan dengan surat **No. 9585/IT2.IX.5.1.4/B/TU.00.09/XII/2024** mengenai Permohonan Kerja Praktek ke PT. Untung Bersama Sejahtera, mahasiswa atas nama:

NO.	NAMA	NRP
1.	Brendan Timothy Mannuel	5025221177
2.	Mochammad Zharif Asyam Marzuqi	5025221163

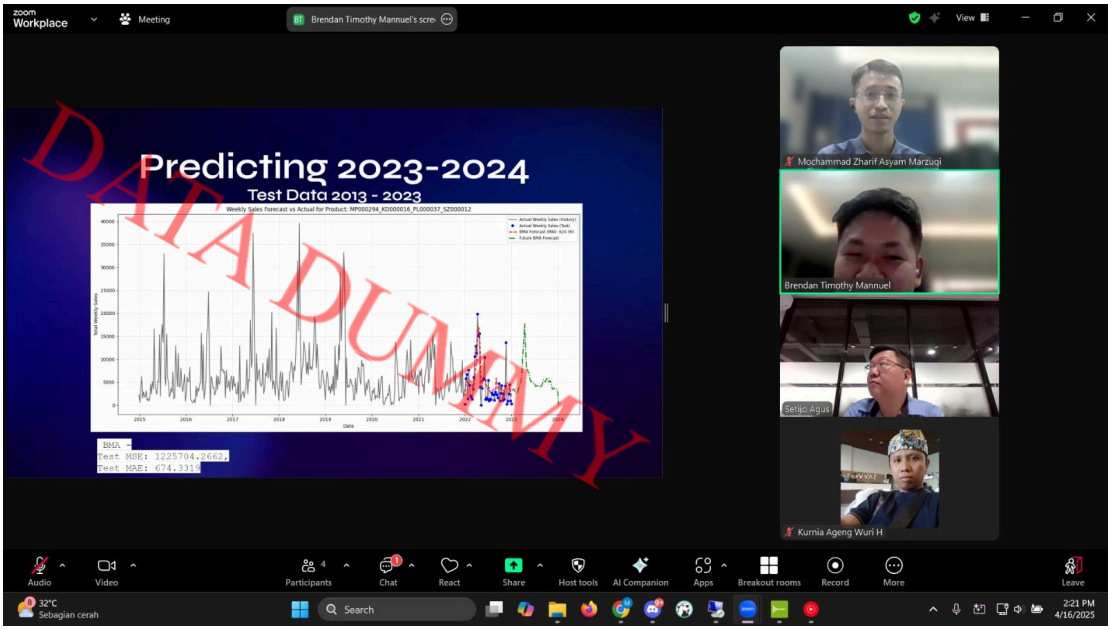
Bersama ini kami informasikan bahwa mahasiswa tersebut diatas di perkenankan melakukan Magang di perusahaan kami mulai **1 Januari – 1 Mei 2025.**
Demikian informasi dari kami untuk dapat dipergunakan sebagaimana perlunya. Terimakasih.

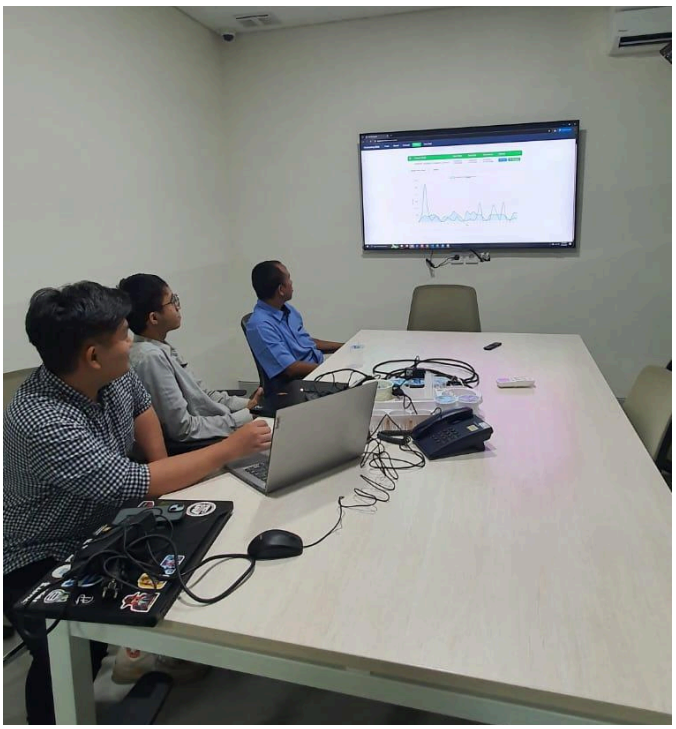
Hormat kami,

UBS GOLD
Trust In Gold

Aucy Chandra, S.T., M.M
Asst. Manager Human Capital

2. Dokumentasi Kerja Praktik

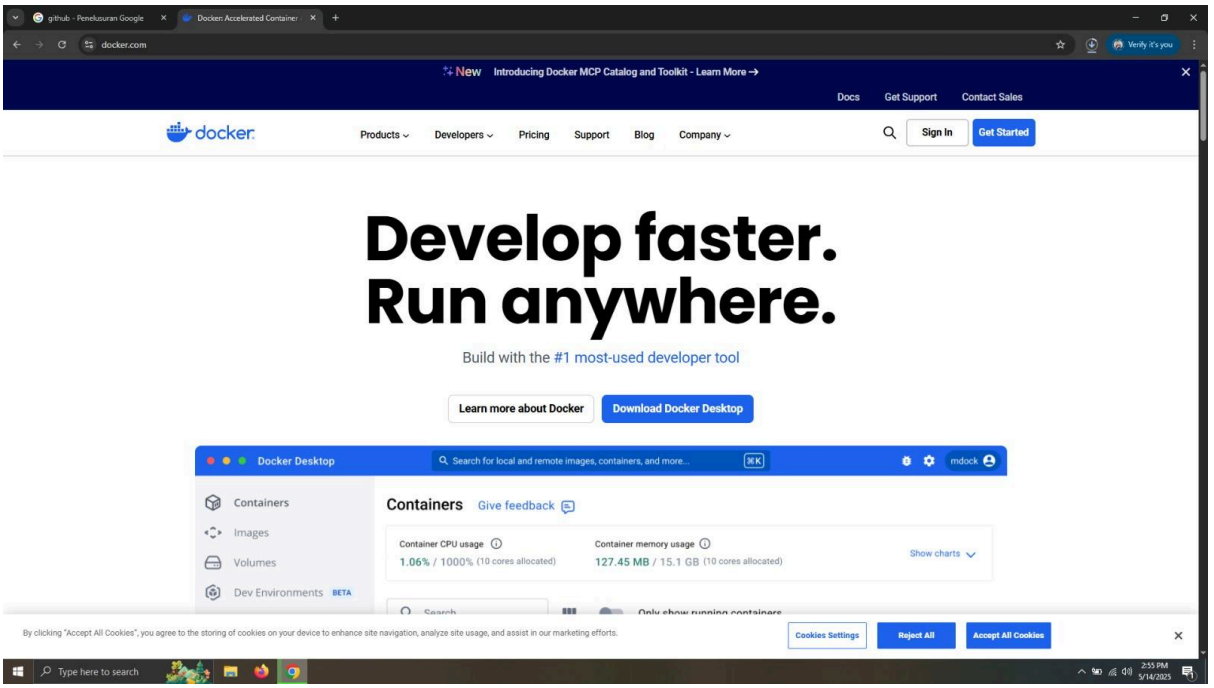




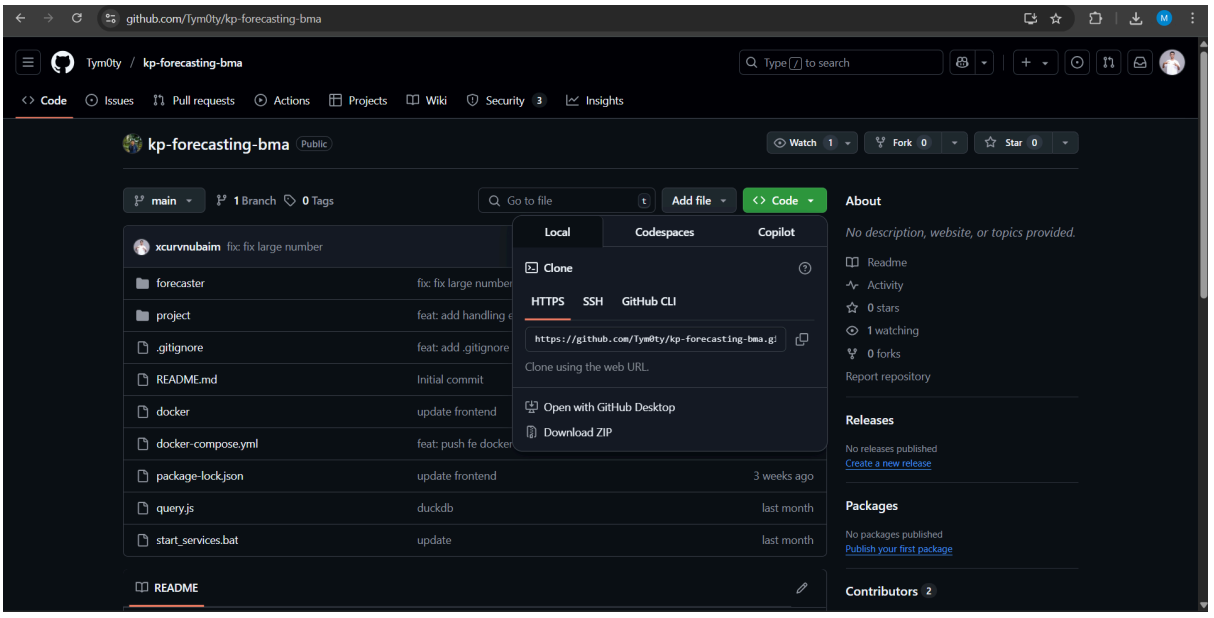
3. Cara Instalasi Aplikasi

Langkah-langkah dalam melakukan instalasi adalah sebagai berikut :

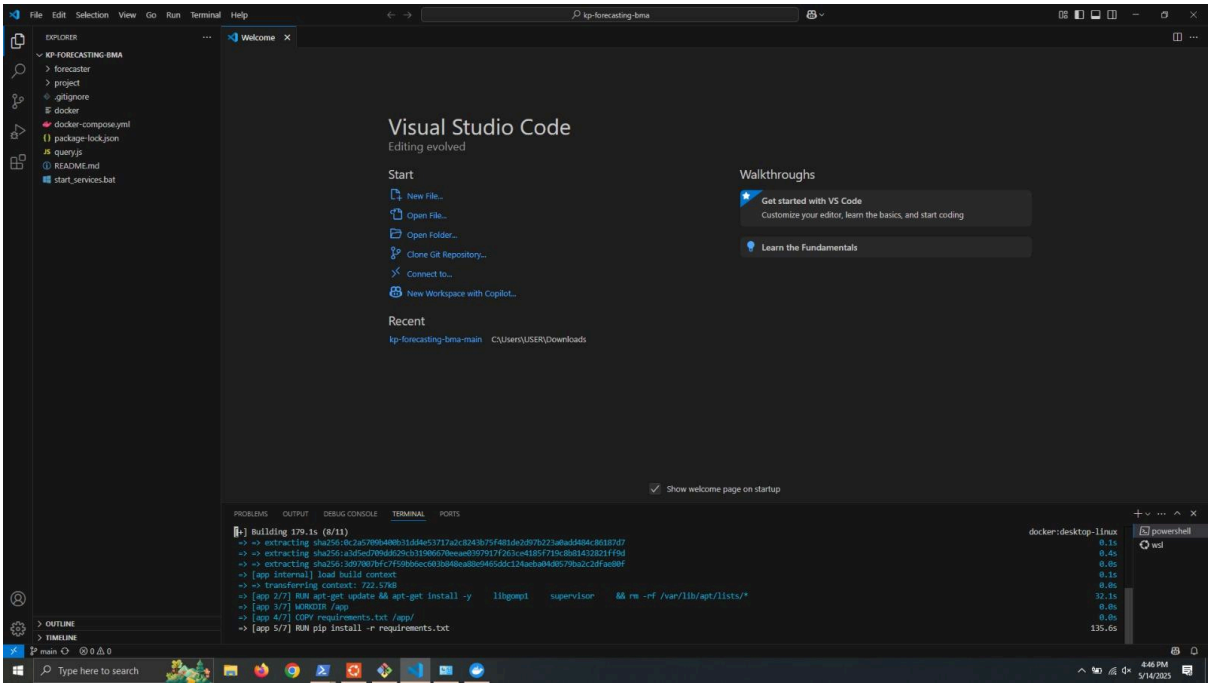
1. Menginstall docker



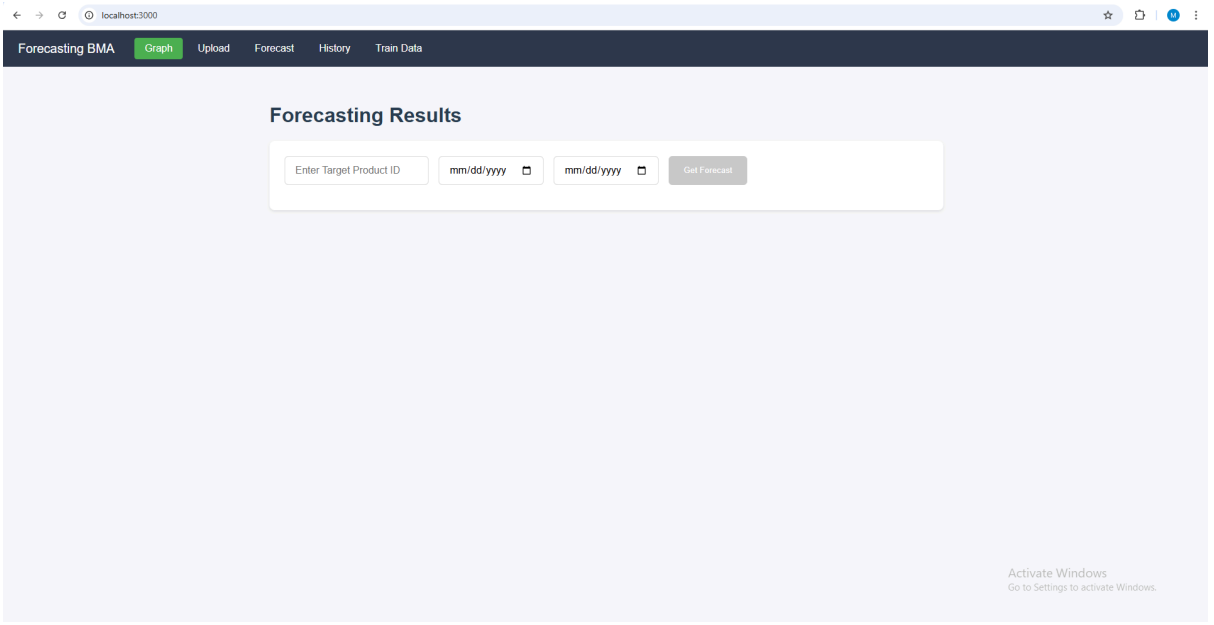
2. Melakukan clone pada github repository
https://github.com/Tym0ty/kp-forecasting-bma



3. Melakukan build dan start container aplikasi dengan command docker compose up



4. Menununggu proses build aplikasi berhasil, kemudian akses aplikasi di browser lewat url http://localhost:3000



4. Dokumentasi Sample Code

Berikut ini merupakan sampe code dari salah satu page aplikasi

- Page Forecast'

```
<template>
  <div class="forecast-page">
    <h1>Start Forecast</h1>

    <!-- Input Section -->
    <div class="input-section">
      <label for="target">Target Product ID</label>
      <input
        id="target"
        v-model="targetProduct"
        class="product-input"
        placeholder="MP000294_KD000016_PL000037_SZ000012"
      />
      <label for="start-date">Start Date</label>
      <input
        id="start-date"
        type="date"
        v-model="startDate"
        class="product-input"
      />
      <label for="end-date">End Date</label>
      <input
        id="end-date"
        type="date"
        v-model="endDate"
        class="product-input"
      />
      <button
        class="fetch-button"
        :disabled="!targetProduct || !startDate || !endDate ||
loading"
        @click="startForecast"
      >
        {{ loading ? 'Starting...' : 'Start Forecast' }}
      </button>
```

```
    </div>

    <!-- Status / Loading -->
    <div v-if="loadingStatus" class="status">
      <p>{{ statusMessage }}</p>
      <div class="spinner"></div>
    </div>

    <!-- Error -->
    <div v-if="error" class="error">{{ error }}</div>

    <!-- Success / Download -->
    <div v-if="finished && result?.output_file" class="result">
      <p>Forecast completed!</p>
      <a
        :href="downloadUrl"
        class="download-button"
        target="_blank"
      >
        Download Forecast CSV
      </a>
    </div>
  </div>
</template>

<script>
import axios from 'axios'

export default {
  name: 'ForecastPage',
  data() {
    return {
      targetProduct: '',
      startDate: '',      // Add this
      endDate: '',        // Add this
      loading: false,          // for initial POST
      loadingStatus: false,    // for polling spinner
      statusMessage: '',
      taskId: null,
      error: null,
      result: null,
      finished: false,
```

```
      pollInterval: null
    }
  },
  computed: {
    downloadUrl() {
      // Extract filename and build download link
      const path = this.result.output_file || ''
      const filename = path.split('/').pop()
      return `http://localhost:8000/download/${filename}`
    }
  },
  methods: {
    async startForecast() {
      this.error = null
      this.result = null
      this.finished = false
      this.loading = true

      try {
        const res = await axios.post(
          `http://localhost:8000/process-csv/`,
          null,
          {
            params: {
              target_product_id: this.targetProduct,
              start_date: this.startDate,
              end_date: this.endDate
            }
          }
        )
        this.taskId = res.data.task_id
        this.loading = false
        this.loadingStatus = true
        this.statusMessage = 'Forecast queued...'
        this.pollInterval = setInterval(this.checkStatus, 2000)
      } catch (err) {
        this.error = 'Failed to start forecast. Please try
again.'
        this.loading = false
      }
    },
    async checkStatus() {
```

```
      if (!this.taskId) return

      try {
        const res = await axios.get(
          `http://localhost:8000/task-status/${this.taskId}`
        )
        const { state, status, result } = res.data

        if (state === 'PENDING' || state === 'PROGRESS') {
          this.statusMessage = status || 'Processing...'
        } else if (state === 'SUCCESS') {
          clearInterval(this.pollInterval)
          this.result = result || {}
          this.statusMessage = 'Done'
          this.loadingStatus = false
          this.finished = true

          // show a success popup
          window.alert('Forecast completed successfully!')

          // redirect to history page
          this.$router.push('/forecast-history')
        } else {
          // FAILURE or other
          clearInterval(this.pollInterval)
          this.statusMessage = status || 'Error during
processing.'
          this.loadingStatus = false
          this.error = this.statusMessage
        }
      } catch (err) {
        clearInterval(this.pollInterval)
        this.loadingStatus = false
        this.error = 'Error fetching status.'
      }
    }
  },
  beforeUnmount() {
    if (this.pollInterval) {
      clearInterval(this.pollInterval)
    }
  }
```

```
}
</script>
```

- Forecaster Pipeline

```
import pandas as pd
from .preprocessing import load_and_prepare_data, filter_product
from .feature_engineering import add_lag_features,
add_rolling_features, add_time_features, add_ramadhan_feature,
create_features_for_step
from .models import get_models, get_base_models
from .evaluation import evaluate, evaluate_predictions
from .config import *
from .bma import calculate_bma_weights
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
import numpy as np

def run_pipeline(filepath):
    df = load_and_prepare_data(filepath)
    df = filter_product(df, TARGET_ID)
    df = add_lag_features(df, N_LAGS, N_WEEKS)
    df = add_rolling_features(df, ROLL_WINDOWS)
    df = add_time_features(df)
    df = add_ramadhan_feature(df)

    df = df.dropna()
    X = df.drop(columns=["TOTAL_JUMLAH"])
    y = df["TOTAL_JUMLAH"]

    X_train, X_test = X[:-TEST_SIZE], X[-TEST_SIZE:]
    y_train, y_test = y[:-TEST_SIZE], y[-TEST_SIZE:]

    results = {}
    for name, model in get_models().items():
        model.fit(X_train, y_train)
        preds = model.predict(X_test)
        metrics = evaluate(y_test, preds)
```

```
        results[name] = metrics
        print(f"[{name.upper()}] MSE: {metrics['MSE']:.2f} |
MAPE: {metrics['MAPE']:.2%}")

    return results

def run_bma_pipeline(file_path, target_product_id,
future_step=FUTURE_STEPS):
    """
    Runs the full pipeline including BMA weight calculation,
evaluation,
    and future forecasting. (LSTM functionality removed)

    Args:
        filepath (str): Path to the data file.
        target_product_id (str/int): The ID of the product to
forecast.

    Returns:
        dict: A dictionary containing BMA weights, individual
model forecasts (test set),
              the BMA forecast (test set), actual values (test
set), evaluation metrics,
              CV scores, and the future forecast series.
              Returns None if the pipeline fails.
    """
    # --- Load Configuration ---
    # Store config in a dictionary for easier passing
    config = {
        'N_LAGS': N_LAGS, 'N_WEEKS': N_WEEKS, 'ROLL_WINDOWS':
ROLL_WINDOWS,
        'TEST_SIZE': TEST_SIZE, 'N_SPLITS_BMA': N_SPLITS_BMA,
        'RANDOM_STATE': 42, 'FUTURE_FORECAST_STEP': future_step
    }


    print(f"--- Starting BMA Pipeline for Product
{target_product_id} ---")

    # 1. Load and Prepare Data
    try:
        df_raw = load_and_prepare_data(file_path)
```

```
        df_filtered = filter_product(df_raw, target_product_id)
        if df_filtered.empty:
            print(f"ERROR: No data found for product
{target_product_id}")
            return None
    except Exception as e:
        print(f"ERROR during data loading/filtering: {e}")
        return None

    # 2. Feature Engineering
    try:
        # Keep the df before dropping NAs, might be useful for
feature calculation history
        df_featured = add_lag_features(df_filtered,
config['N_LAGS'], config['N_WEEKS'])
        df_featured = add_rolling_features(df_featured,
config['ROLL_WINDOWS'])
        df_featured = add_time_features(df_featured)
        df_featured = add_ramadhan_feature(df_featured) #
Optional

        df_processed = df_featured.dropna() # This df is used
for training/testing

        if df_processed.empty:
            print("ERROR: DataFrame is empty after feature
engineering and dropping NA.")
            return None

        # Define features (X) and target (y) based on processed
data
        # Ensure target is excluded, and any other non-feature
columns like PRODUCT_ID
        potential_non_features = ["TOTAL_JUMLAH", "PRODUCT_ID"]
        feature_names = [col for col in df_processed.columns if
col not in potential_non_features]

        if not feature_names:
             print("ERROR: No feature columns found after
processing.")
             return None
```

```
        X = df_processed[feature_names]
        y = df_processed["TOTAL_JUMLAH"]

    except Exception as e:
        print(f"ERROR during feature engineering: {e}")
        return None

    # Check if enough data for split
    if len(X) <= config['TEST_SIZE']:
        print(f"ERROR: Not enough data ({len(X)} samples) for
test set size {config['TEST_SIZE']}")
        return None

    # 3. Split Data
    X_train, X_test = X[:-config['TEST_SIZE']],
X[-config['TEST_SIZE']:]
    y_train, y_test = y[:-config['TEST_SIZE']],
y[-config['TEST_SIZE']:]
    print(f"Train set size: {len(X_train)}, Test set size:
{len(X_test)}")
    test_indices = X_test.index # Store index for test set
results

    # --- BMA Specific Steps ---

    # 4. Define Base Models
    base_models = get_base_models()

    # 5. Calculate BMA Weights using Cross-Validation on
Training Data
    print(f"\n--- Performing {config['N_SPLITS_BMA']}-Fold CV on
Training Data for BMA Weights ---")
    kf = KFold(n_splits=config['N_SPLITS_BMA'], shuffle=True,
random_state=config['RANDOM_STATE'])
    cv_mse_scores = {}

    for model_name, model in base_models.items():
        print(f"Cross-validating {model_name}...")
        fold_errors = []
        for fold, (train_index, val_index) in
enumerate(kf.split(X_train)):
            X_tr, X_val = X_train.iloc[train_index],
```

```
X_train.iloc[val_index]
            y_tr, y_val = y_train.iloc[train_index],
y_train.iloc[val_index]
            try:
                current_model = model # Assumes fit overwrites
previous state
                current_model.fit(X_tr, y_tr)
                y_pred_val = current_model.predict(X_val)
                error = mean_squared_error(y_val, y_pred_val)
                fold_errors.append(error)
            except Exception as e:
                print(f"  ERROR during CV Fold {fold+1} for
{model_name}: {e}")
        if fold_errors:
            avg_cv_error = np.mean(fold_errors)
            cv_mse_scores[model_name] = max(avg_cv_error, 1e-9)
            print(f"  {model_name} Avg CV MSE:
{avg_cv_error:.4f}")
        else:
            print(f"  {model_name} failed all CV folds.")
            cv_mse_scores[model_name] = float('inf')

    bma_weights = calculate_bma_weights(cv_mse_scores)
    if bma_weights is None: return None
    print("\nCalculated BMA Weights:")
    for name, weight in bma_weights.items():
        if weight > 0: print(f"  {name}: {weight:.4f}")

    # --- Final Model Fitting and Prediction on Test Set ---

    # 6. Fit Final Models on Full Training Data
    print("\n--- Fitting Final Models on Full Training Data
---")
    fitted_models = {}
    temp_bma_weights = bma_weights.copy() # Work with a copy for
renormalization

    for model_name, model in base_models.items():
        if temp_bma_weights.get(model_name, 0) > 0:
            print(f"Fitting {model_name}...")
            try:
                model.fit(X_train, y_train)
```

```
                fitted_models[model_name] = model
            except Exception as e:
                print(f"  ERROR fitting final {model_name}:
{e}")
                temp_bma_weights[model_name] = 0 # Set weight to
0 if final fit fails

    # Renormalize weights if any models failed the final fit
    active_weights = {name: w for name, w in
temp_bma_weights.items() if name in fitted_models and w > 0}
    if not active_weights:
        print("ERROR: All models for BMA failed to fit.")
        return None
    weight_sum = sum(active_weights.values())
    if weight_sum < 1e-9 :
         print("ERROR: Sum of weights for successfully fitted
models is zero.")
         return None
    elif abs(weight_sum - 1.0) > 1e-6 :
        print("Renormalizing weights after final fit failures.")
        final_bma_weights = {name: (w / weight_sum if name in
active_weights else 0)
                             for name, w in
temp_bma_weights.items()}
        print("Renormalized BMA Weights:")
        for name, weight in final_bma_weights.items():
             if weight > 0: print(f"  {name}: {weight:.4f}")
    else:
        final_bma_weights = active_weights # Use the weights of
successfully fitted models

    # 7. Generate Forecasts on Test Data
    print("\n--- Generating Forecasts on Test Data ---")
    individual_forecasts_test = {}
    bma_forecast_test = np.zeros(len(X_test))
    temp_bma_weights_pred = final_bma_weights.copy() # Copy for
prediction renormalization

    for name, model in fitted_models.items():
        weight = temp_bma_weights_pred.get(name, 0)
        if weight > 0:
            print(f"Predicting test set with {name} (Weight:
```

```
{weight:.4f})")
            try:
                forecast = model.predict(X_test)
                forecast[forecast < 0] = 0 # Ensure non-negative
                individual_forecasts_test[name] = forecast
                bma_forecast_test += weight * forecast
            except Exception as e:
                print(f"  ERROR predicting test set with {name}:
{e}")
                individual_forecasts_test[name] =
np.zeros(len(X_test))
                temp_bma_weights_pred[name] = 0 # Set weight to
0 if prediction fails

    # Renormalize weights AGAIN if predictions failed for some
models
    active_weights_pred = {name: w for name, w in
temp_bma_weights_pred.items() if w > 0 and name in
fitted_models}
    if not active_weights_pred:
         print("ERROR: All models failed during test set
prediction.")
         return None
    weight_sum_pred = sum(active_weights_pred.values())
    if weight_sum_pred < 1e-9 :
        print("ERROR: Sum of weights for successfully predicting
models is zero.")
        return None
    elif abs(weight_sum_pred - 1.0) > 1e-6:
        print("Renormalizing weights after prediction failures
on test set.")
        final_bma_forecast_test = np.zeros(len(X_test))
        final_bma_weights_for_forecast = {}
        for name, forecast in individual_forecasts_test.items():
             if temp_bma_weights_pred.get(name, 0) > 0 :
                 new_weight = temp_bma_weights_pred[name] /
weight_sum_pred
                 final_bma_weights_for_forecast[name] =
new_weight
                 final_bma_forecast_test += new_weight *
forecast
        bma_forecast_test = final_bma_forecast_test
```

```
        final_weights = final_bma_weights_for_forecast # These
are the weights used for test and future
        print("Final Renormalized BMA Weights used for
forecast:")
        for name, weight in final_weights.items(): print(f"
{name}: {weight:.4f}")
    else:
        # If no renormalization needed, the final weights are
those that successfully predicted
        final_weights = active_weights_pred

    print(f"\nFinal BMA Forecast on Test Set (first 5):
{np.round(bma_forecast_test[:5], 2)}")
    print(f"Actual Values on Test Set (first 5):
{y_test.values[:5]}")

    # 8. Evaluate BMA Forecast on Test Set
    print("\n--- Evaluating Final BMA Forecast on Test Set ---")
    bma_metrics_test = evaluate_predictions(y_test,
bma_forecast_test)
    print(f"[BMA Test Set] MSE: {bma_metrics_test['MSE']:.2f} |
MAPE: {bma_metrics_test['MAPE']:.2%}")

    individual_metrics_test = {}
    for name, forecast in individual_forecasts_test.items():
        if final_weights.get(name,0) > 0 :
             metrics = evaluate_predictions(y_test, forecast)
             individual_metrics_test[name] = metrics
             print(f"  [{name.upper()} Test Set] MSE:
{metrics['MSE']:.2f} | MAPE: {metrics['MAPE']:.2%}")


    # --- 9. Forecast Future Values ---
    print(f"\n--- Forecasting {config['FUTURE_FORECAST_STEP']}
Steps into the Future ---")

    # Determine the frequency of the data (e.g., 'W-MON', 'D')
for date range generation
    # Infer frequency from the training data index if possible
    inferred_freq = pd.infer_freq(df_processed.index)
    if inferred_freq is None:
        # Default or raise error if frequency cannot be
```

```
determined - IMPORTANT
        print("WARNING: Could not infer data frequency. Assuming
weekly ('W-MON'). Adjust if necessary.")
        inferred_freq = 'W-MON' # Or 'D' for daily, 'M' for
monthly etc.

    last_date = df_processed.index[-1]
    future_dates = pd.date_range(start=last_date +
pd.Timedelta(days=1 if inferred_freq == 'D' else 7), # Adjust
step based on freq

periods=config['FUTURE_FORECAST_STEP'], freq=inferred_freq)

    # Initialize the series for recursive forecasting with
historical target values
    # Using df_processed ensures alignment with data used for
training features
    extended_series = df_processed["TOTAL_JUMLAH"].copy()
    future_predictions = []

    # Use only models and weights that were successful through
fitting and test prediction
    active_models_for_future = {k: v for k, v in
fitted_models.items() if k in final_weights}

    print(f"Generating future forecast using models:
{list(active_models_for_future.keys())}")

    for future_date in future_dates:
        # Create features for the current future step
        feature_df = create_features_for_step(future_date,
extended_series, feature_names, config)

        if feature_df is None:
            print(f"ERROR: Failed to create features for
{future_date}. Stopping future forecast.")
            # Handle this - maybe return partial results or
None
            future_pred_series = pd.Series(future_predictions,
index=future_dates[:len(future_predictions)], name='Forecast') #
Partial results
            break # Exit the loop
```

```
        # --- Perform BMA prediction for the single future step
---
        bma_pred_step = 0
        for model_name, model in
active_models_for_future.items():
            weight = final_weights.get(model_name, 0) # Should
always be > 0 here
            try:
                # Predict expects a DataFrame
                pred = model.predict(feature_df)[0] # Get single
prediction value
                bma_pred_step += weight * max(pred, 0) # Apply
weight and ensure non-negative
            except Exception as e:
                 print(f"  ERROR predicting future step with
{model_name} for {future_date}: {e}")
                 # How to handle model failure here? Could skip
model, renormalize weights for the step, or stop.
                 # For simplicity, we might ignore the failing
model's contribution for this step,
                 # but this will slightly skew the BMA result. A
more robust solution might be needed.
                 pass # Ignoring contribution for now

        # Store prediction and update the series for the next
step's feature calculation
        future_predictions.append(bma_pred_step)
        # Use .loc for safe assignment with timestamp index
        extended_series.loc[future_date] = bma_pred_step

    else: # If the loop completed without break
        # Create final future predictions series
        future_pred_series = pd.Series(future_predictions,
index=future_dates, name='Forecast')
        print("\nFuture Forecast (first 5 steps):")
        print(future_pred_series.head())

    # --- 10. Return Results ---
    results = {
        "final_bma_weights": final_weights, # Weights used for
test set and future forecast
```

```
        "individual_forecasts_test": {name: pd.Series(forecast,
index=test_indices)
                                      for name, forecast in
individual_forecasts_test.items() if name in final_weights},
        "bma_forecast_test": pd.Series(bma_forecast_test,
index=test_indices),
        "actual_values_test": pd.DataFrame({'TANGGAL':
df_filtered.index[-config['TEST_SIZE']:], 'TOTAL_JUMLAH':
y_test}),
        "predictions_test": pd.DataFrame({'TANGGAL':
df_filtered.index[-config['TEST_SIZE']:], 'TOTAL_JUMLAH':
bma_forecast_test}),
        "bma_metrics_test": bma_metrics_test,
        "individual_metrics_test": individual_metrics_test,
        "cv_mse_scores": cv_mse_scores,
        "future_forecast": pd.DataFrame({'TANGGAL':
future_dates, 'TOTAL_JUMLAH': future_pred_series}),
    }

    print(f"\n--- BMA Pipeline for Product {target_product_id}
Finished ---")
    return results
```

- Main Python App

```
from kp_forecaster.pipeline import run_pipeline,
run_bma_pipeline
from kp_forecaster.plot import plot_forecast, plot_test_forecast
import time

if __name__ == "__main__":
    filepath = "data/processed.csv"
    # Optional: Create dummy file if needed for placeholder
    # try: load_and_prepare_data(filepath).to_csv(filepath)
    # except Exception as e: print(f"Could not create dummy
file: {e}")
    target_product_id = 'MP000294_KD000016_PL000037_SZ000012'

    pipeline_results = run_bma_pipeline(filepath,
```

```
target_product_id)

    # Save the results to a CSV file
    if pipeline_results:
        results_df = pipeline_results['future_forecast']
        timestamp = time.time()

results_df.to_csv(f'output/{timestamp}_future_forecast.csv',
index=False)
        print("BMA results saved to future_forecast.csv")

    if pipeline_results:
        plot_forecast(results_df, target_product_id)
        plot_test_forecast(pipeline_results['predictions_test'],
pipeline_results['actual_values_test'], target_product_id)
    else:
        print("BMA pipeline failed, no plot generated.")

    if pipeline_results:
        print("\n--- Pipeline Results Summary ---")
        print("Final BMA Weights:",
pipeline_results['final_bma_weights'])
        print("\nBMA Test Set Metrics:",
pipeline_results['bma_metrics_test'])
        if pipeline_results['future_forecast'] is not None:
            print("\nFuture Forecast:")
            print(pipeline_results['future_forecast'])
        else:
            print("\nFuture Forecast generation failed or was
incomplete.")
```

DAFTAR PUSTAKA

Chen, S., & Zheng, W. (2025). RRMSE-enhanced weighted voting regressor for improved ensemble regression. PLOS ONE, 20(3), e0319515. https://doi.org/10.1371/journal.pone.0319515

Echtenbruck, P., Echtenbruck, M., Batenburg, J., Bäck, T., Naujoks, B., & Emmerich, M. (2022). Optimally Weighted Ensembles of Regression Models: Exact Weight Optimization and Applications. arXiv preprint arXiv:2206.11263. https://arxiv.org/abs/2206.11263

Denis, R., & Keerthana, D. (2024). Performance Analysis of Voting Regression-Based Ensemble Learning Methods for Food Demand Forecasting. International Journal of Data Informatics and Intelligent Computing, 3(2), 34–41. https://doi.org/10.59461/ijdiic.v3i2.114

Yildiz, A. R., & Yilmaz, M. (2022). Ensemble Voting Regression Based on Machine Learning for Predicting Medical Waste: A Case from Turkey. Mathematics, 10(14), 2466. https://doi.org/10.3390/math10142466

BIODATA PENULIS

BIODATA PENULIS I

Nama	: Brendan Timothy Mannuel
Tempat, Tanggal Lahir	: Surabaya, 25 Mei 2004
Jenis Kelamin	: Laki-Laki
Telepon	: +6281515528712
Email	: brendantimothym@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2022
Semester	: 6 (Enam)

BIODATA PENULIS II

Nama	: Mochammad Zharif Asyam Marzuqi
Tempat, Tanggal Lahir	: Trenggalek, 12 Juni 2003
Jenis Kelamin	: Laki-laki
Telepon	: +6285233550996
Email	: zhariffmarzuqi@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2022
Semester	: 6 (Enam)