STOCK PRICE SENTIMENT
PADA SAHAM BBRI

TUGAS BESAR DATA MINING

Disusun oleh:

Muhammad Azka Nuril Islami (714220001)

Gaizka Wisnu Prawira (714220011)

Muhammad Fathir (714220021)

Salwa Mutfia Indah Putri (714220026)

Dosen Pengampu:

Nisa Hanum Harani, S.T., M.T., CDSP.,SFPC

NIK. 117.89.223

PROGRAM STUDI DIV TEKNIK INFORMATIKA

UNIVERSITAS LOGISTIK & BISNIS INTERNASIONAL

BANDUNG

2025

HALAMAN PERNYATAAN ORISINALITAS

Laporan tugas besar ini adalah hasil karya kami sendiri dan semua sumber, baik yang dikutip maupun dirujuk telah kami nyatakan dengan benar. Bilamana di kemudian hari ditemukan bahwa karya tulis ini menyalahi peraturan yang ada berkaitan dengan etika dan kaidah penulisan karya ilmiah yang berlaku, maka kami bersedia dituntut dan diproses sesuai dengan ketentuan yang berlaku.

Yang menyatakan,

Nama : Muhammad Azka Nuril Islami

NIM : 714220001

Tanda Tangan: .....

Tanggal: Kamis, 10 Juli 2025

Mengetahui,

Ketua :..... (.....tanda tangan. .....)

Dosen Pengampu Mata Kuliah : ..... (.....tanda tangan. .....)

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan Laporan Tugas Besar Data Mining ini yang berjudul "Stock Price Sentiment pada Saham BBRI".

Laporan ini disusun untuk memenuhi tugas akhir mata kuliah Data Mining pada Program Studi D4 Teknik Informatika, Universitas Logistik dan Bisnis Internasional.

Kami mengucapkan terima kasih kepada:

- Dosen pengampu mata kuliah Data Mining atas bimbingan dan ilmunya selama perkuliahan berlangsung.
- Orang tua dan keluarga yang selalu memberikan dukungan moril dan semangat.
- Rekan satu kelompok atas kerja sama dan komitmen dalam menyelesaikan tugas ini bersama.

Kami menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan demi perbaikan di masa mendatang.

Bandung, 10 Juli 2025

Penyusun,

Muhammad Azka Nuril

Gaizka Wisnu Prawira

Muhammad Fathir

Salwa Mutfia Indah Putri

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Logistik dan Bisnis Internasional, saya yang bertanda tangan di bawah ini:

Nama : Muhammad Azka Nuril Islami

NIM : 714220001

Selaku ketua kelompok, menyatakan menyetujui untuk memberikan kepada Universitas Logistik dan Bisnis Internasional, hak bebas royalti noneksklusif (non-exclusive royalty free right) atas karya ilmiah kami yang berjudul, "STOCK PRICE SENTIMENT PADA SAHAM BBRI" beserta perangkat yang ada (jika diperlukan). Dengan hak ini, ULBI berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir kami selama tetap mencantumkan nama kami sebagai penulis/pemilik hak cipta.

Demikian pernyataan ini saya buat dengan sebenar-benarnya.

Dibuat di : Bandung

Pada tanggal : 10 Juli 2025

Yang menyatakan,

Muhammad Azka Nuril Islami

Ketua Kelompok

ABSTRAK

ABSTRACT

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR RUMUS

DAFTAR NOTASI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pasar saham adalah komponen utama dari sistem keuangan suatu negara yang mencerminkan kesejahteraan ekonomi [1]. Fluktuasi harga saham biasanya dipengaruhi oleh berbagai faktor, seperti kondisi perusahaan, kebijakan ekonomi, dan faktor psikologis investor. Dalam praktiknya, keputusan pembelian dan penjualan saham tidak semata-mata didasarkan pada pertimbangan rasional, tetapi juga pada persepsi dan emosi para pelaku pasar yang dapat berubah sewaktu-waktu [2] . Keadaan ini berarti bahwa informasi dan opini publik dapat menjadi penyebab utama perubahan harga saham, terutama dalam jangka pendek [3].

Perkembangan teknologi informasi, khususnya media sosial seperti Twitter, telah mengubah cara individu dalam memberi dan menerima informasi. Twitter saat ini menjadi forum yang sangat aktif untuk memberikan komentar, berita, dan reaksi terhadap berbagai isu ekonomi dan perusahaan [4]. Kecepatan dan volume informasi yang disebarkan di Twitter berpotensi memberikan gambaran langsung tentang opini publik mengenai suatu perusahaan atau area bisnis[5]. Oleh karena itu, analisis sentimen tweet berpotensi menjadi sumber sekunder yang berharga untuk data dalam prediksi pasar [4][6].

Meskipun harga saham historis telah menjadi dasar model prediksi yang dibangun selama bertahun tahun, metode ini tidak lengkap. Model prediksi tradisional tidak dapat menangkap kekuatan psikologis dan pola pikir pasar yang berubah dengan cepat. Selama krisis atau pengumuman berita penting dalam situasi yang tidak stabil, data media sosial dapat memberikan peringatan lebih awal daripada indikator teknikal. Oleh karena itu, integrasi data sentimen dengan data historis sangat potensial untuk meningkatkan akurasi model prediksi saham.

Beberapa penelitian sebelumnya telah mengindikasikan bahwa sentimen media sosial berhubungan dengan pergerakan harga saham [7]. Hasil-hasil ini menunjukkan bahwa sentimen publik yang positif menyebabkan pergerakan harga saham naik, sementara sentimen negatif dapat menjadi pendahulu pergerakan harga turun. Namun, sebagian besar penelitian ini masih kurang dalam hal pengujian korelasi yang tepat tanpa mencoba meniru model prediksi yang lebih umum. Pendekatan yang lebih formal diperlukan untuk menggabungkan kedua jenis data di bawah kerangka model prediksi model prediksi yang terjamin.

Dengan potensi besar dari sentimen media sosial, penelitian ini bertujuan untuk membuat dan memvalidasi sebuah model untuk prediksi harga saham yang menggabungkan data masa lalu dan sentimen Twitter secara real-time [5][8]. Tujuannya adalah untuk mengidentifikasi apakah dengan menggabungkan kedua sumber tersebut dapat memberikan model yang lebih baik dan responsif terhadap perubahan pasar. Penelitian ini diharapkan dapat memberikan kontribusi yang berarti bagi

perkembangan teknologi keuangan modern dan menginformasikan pengambilan keputusan investasi yang lebih berwawasan.

1.2 Rumusan Masalah

Berdasarkan latar belakang sebelumnya, dapat dirumuskan beberapa pertanyaan utama sebagaimana berikut:

A. Apakah analisis sentimen real time dari Twitter memiliki korelasi signifikan dengan pergerakan harga saham?
B. Bagaimana kinerja model prediksi harga saham yang hanya menggunakan data historis dibandingkan dengan model yang menggabungkan data historis dan sentimen Twitter?
C. Sejauh mana penambahan fitur sentimen Twitter dapat meningkatkan akurasi dan ketahanan model terhadap volatilitas pasar?

1.3 Tujuan penelitian

1.3.1. Tujuan Umum

Menganalisis pengaruh integrasi data sentimen Twitter terhadap akurasi model prediksi harga saham dibandingkan dengan model prediksi yang hanya menggunakan data historis.

1.3.2. Tujuan Khusus

A. Melakukan analisis sentimen pada data Twitter menggunakan model RoBERTa.
B. Menggabungkan data hasil analisis sentiment dengan data historis berdasarkan tanggal data.
C. Membangun dan mengevaluasi model klasifikasi sentimen dengan algoritma XGBoost, SVR, Random Forest, MLP, dan Logistic Regression.

1.4 Manfaat Penelitian

1.1.1 Manfaat Teoretis

A. Memberikan kontribusi pada pengembangan ilmu pengetahuan di bidang keuangan dan data science, khususnya terkait integrasi analisis sentimen media sosial dalam prediksi harga saham.
B. Menjadi referensi akademik mengenai penggunaan data sentimen Twitter untuk mendukung model prediksi harga saham yang lebih akurat.

C. Memperkaya literatur terkait pemanfaatan big data dan text mining dalam analisis pasar modal.

1.1.2 Manfaat Praktis

A. Membantu investor dalam mengambil keputusan investasi yang lebih tepat dengan mempertimbangkan informasi sentimen publik dari media sosial.
B. Memberikan wawasan bagi perusahaan sekuritas dan analis pasar untuk mengembangkan sistem prediksi harga saham yang lebih responsif terhadap dinamika pasar.
C. Menjadi dasar bagi pengembangan aplikasi atau sistem prediksi harga saham yang memanfaatkan integrasi data historis dan data sentimen secara real-time.

1.5 Ruang Lingkup

Penelitian ini memiliki ruang lingkup yang difokuskan pada pemanfaatan data historis harga saham dan data sentimen dari media sosial Twitter untuk membangun model prediksi harga saham. Adapun ruang lingkup penelitian ini dijabarkan sebagai berikut:

Penelitian ini dibatasi pada:

A. Penelitian menggunakan data historis saham dari perusahaan tertentu yang terdaftar di bursa, dalam periode waktu tertentu (misalnya satu tahun terakhir)
B. Informasi diambil dari tweet publik yang relevan dengan saham perusahaan tersebut menggunakan kata kunci atau tagar tertentu.
C. Sentimen akan diklasifikasikan ke dalam kategori positif, negatif, atau netral menggunakan metode pemrosesan bahasa alami (Natural Language Processing/NLP) seperti IndoBERT atau RoBERTa.
D. Model prediksi harga saham akan dikembangkan dengan pendekatan machine learning (XGBoost, SVR, Random Forest, MLP, dan Logistic Regression) dan dibandingkan antara model berbasis data historis saja dan model yang juga mengintegrasikan sentimen Twitter.

BAB II

TINJAUAN PUSTAKA

2.1 Kajian Teori dan Konsep Penting

2.1.1 Data Mining dan Machine Learning

A. Data mining adalah proses ekstraksi informasi berharga, pola, dan pengetahuan yang tersembunyi dari kumpulan data yang besar dan kompleks. Tujuan utamanya adalah untuk mengidentifikasi hubungan dan tren yang dapat digunakan untuk mendukung pengambilan keputusan strategis. Dalam esensinya, data mining merupakan teknik analisis yang menggunakan metode statistik, matematika, dan kecerdasan buatan untuk menggali pengetahuan yang belum diketahui secara otomatis dari data [9].

B. Machine learning merupakan bagian dari artificial intelligence/kecerdasan buatan yang membutuhkan data-data valid untuk proses belajarnya. Machine learning dapat membuat keputusan yang tepat dan cepat, serta dapat memberikan solusi terhadap berbagai permasalahan. Machine learning memiliki kemampuan untuk belajar sendiri dan memutuskan sesuatu tanpa harus diprogram berulang kali oleh manu-sia, hal ini dapat terjadi karena adanya pengalaman berbagai data yang dimiliki [10].

2.1.2 Teknik yang Digunakan

A. Bi-LSTM dan RoBERTa

Bi-LSTM pengembangan dari model LSTM yang memproses data sekuensial dari dua arah, yaitu forward dan backward. Dengan arsitektur ini, Bi-LSTM mampu menangkap konteks kata sebelum dan sesudah secara lebih baik dibanding LSTM biasa. Hal ini membuat Bi-LSTM banyak digunakan dalam analisis sentimen karena dapat memahami dependensi kata dalam kalimat secara menyeluruh, sehingga meningkatkan akurasi klasifikasi sentimen teks.

RoBERTa (Robustly Optimized BERT Pretraining Approach) adalah model transformer yang merupakan pengembangan dari BERT dengan optimasi pada jumlah data pre-training, ukuran batch, dan strategi masking yang lebih dinamis. RoBERTa terbukti memiliki performa yang lebih tinggi dibanding BERT pada berbagai tugas NLP, termasuk analisis sentimen media sosial. Model ini dapat

memahami makna kata dalam konteks kalimat secara lebih mendalam dan kompleks, sehingga menghasilkan klasifikasi sentimen yang lebih akurat.

B. Algoritma Klasifikasi Sentimen
   a. Support Vector Regression (SVR): Meskipun SVR umumnya digunakan untuk regresi, dalam penelitian analisis sentimen SVR dapat digunakan untuk memprediksi skor sentimen yang kemudian dipetakan menjadi kategori sentimen positif, negatif, atau netral.
   b. Multilayer Perceptron (MLP): Merupakan model jaringan saraf tiruan (artificial neural network) yang terdiri dari beberapa lapisan tersembunyi (hidden layers) dan dapat menangkap pola non-linear kompleks dalam data teks.
   c. Logistic Regression: Model regresi untuk klasifikasi biner atau multi-kelas yang banyak digunakan sebagai baseline pada analisis sentimen karena interpretasinya yang sederhana dan proses training yang cepat.
   d. Extreme Gradient Boosting (XGBoost): Merupakan algoritma boosting berbasis decision tree yang memiliki performa tinggi dan efisien. XGBoost sering digunakan pada kompetisi data science karena akurasinya yang baik pada berbagai jenis dataset.
   e. Random Forest: Algoritma ensemble learning yang menggabungkan banyak decision tree untuk meningkatkan akurasi dan mengurangi overfitting, efektif pada data dengan banyak fitur seperti representasi TF-IDF.

C. Model Prediksi Harga Saham

   Untuk prediksi harga saham, model yang digunakan antara lain:
   a. Long Short-Term Memory (LSTM): Jaringan saraf tiruan yang dirancang khusus untuk menangani data deret waktu dengan memperhatikan dependensi jangka panjang.
   b. Random Forest dan SVR: Dapat digunakan untuk regresi harga saham atau klasifikasi arah pergerakan harga saham.

2.2 Studi Terkait (Penelitian Sejenis)

Beberapa penelitian sebelumnya telah dilakukan untuk mengintegrasikan data sentimen dengan data historis dalam prediksi harga saham. Penelitian oleh Maharani et al. melakukan post-training IndoBERT dengan korpus finansial Indonesia untuk meningkatkan akurasi analisis sentimen dan

topik di domain keuangan, menunjukkan potensi pengembangan model RoBERTa domain-spesifik untuk analisis sentimen finansial [11].

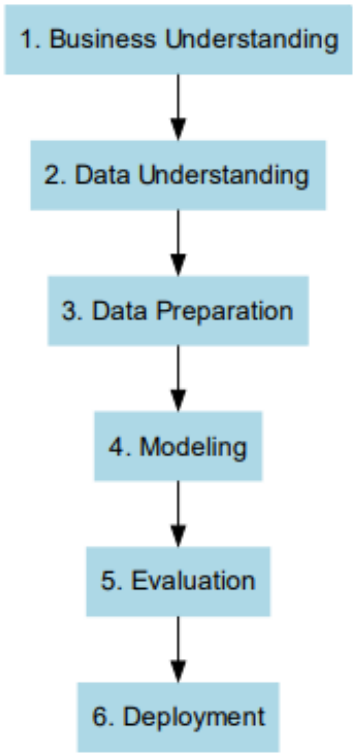
Penelitian lain menggabungkan data tweet dan berita dalam model prediksi harga saham menggunakan MLP dan LSTM, di mana hasilnya menunjukkan bahwa integrasi kedua sumber data tersebut dapat meningkatkan akurasi model dibandingkan hanya menggunakan data historis[12]. Selain itu, terdapat penelitian yang menggunakan analisis sentimen microblogging (Twitter) dan machine learning untuk prediksi harga saham, yang menunjukkan bahwa integrasi data sentimen dengan data historis mampu meningkatkan akurasi prediksi [13].

Penelitian lainnya juga menggabungkan data tweet dan berita dalam prediksi harga saham menggunakan metode machine learning seperti MLP dan LSTM, dan hasilnya menunjukkan bahwa kombinasi kedua sumber data tersebut dapat meningkatkan akurasi model prediksi dibandingkan hanya menggunakan data historis saham [14]. Studi lain yang menggunakan analisis sentimen microblogging (Twitter) dan machine learning untuk memprediksi pasar saham menunjukkan bahwa model yang mengintegrasikan data sentimen dari media sosial dengan data historis dapat memberikan hasil prediksi yang lebih akurat [15].

Selain itu, terdapat penelitian yang mengembangkan model prediksi harga saham dengan menggabungkan data historis dan sentimen Twitter menggunakan Bi-LSTM dan RoBERTa, dan hasil penelitian tersebut menunjukkan bahwa integrasi data sentimen dengan data historis secara signifikan meningkatkan akurasi prediksi dibandingkan model yang hanya menggunakan data historis [15]. Penelitian lainnya mengumpulkan lebih dari 12.000 komentar investor saham di China, dan menggunakan Bi-LSTM untuk prediksi harga saham dengan integrasi analisis sentimen. Hasilnya menunjukkan bahwa akurasi prediksi meningkat ketika sentimen dimasukkan ke dalam model [16]. Penelitian lain juga menggabungkan RoBERTa untuk analisis sentimen microblog (mirip Twitter) dan LSTM untuk prediksi harga saham, dan penelitian ini menunjukkan bahwa model integrasi mampu outperform model berbasis data historis saja [17].

2.3 Visualisasi (Diagram Alir Konsep)

Diagram alur berikut menggambarkan tahapan-tahapan utama yang dilakukan dalam penelitian ini, dimulai dari pemahaman permasalahan bisnis hingga tahap deployment. Penelitian ini mengadopsi model proses CRISP-DM (Cross Industry Standard Process for Data Mining) sebagai kerangka kerja utama karena model ini terbukti fleksibel dan banyak digunakan dalam proyek data mining. Setiap tahapan saling berkaitan secara iteratif, memungkinkan penyesuaian kembali terhadap proses sebelumnya bila ditemukan temuan baru dalam proses selanjutnya. Berikut adalah diagram alur dalam penelitian ini:



Gambar 2. 1 Diagram Alir Konsep

2.4 State of The Art

Perkembangan terkini dalam bidang prediksi harga saham menggunakan pendekatan data mining, khususnya yang menggabungkan data historis dan data sentimen media sosial. Dalam beberapa tahun terakhir, pemanfaatan media sosial seperti Twitter sebagai sumber data alternatif dalam prediksi pasar saham semakin berkembang.

Salah satu pendekatan mutakhir yang banyak digunakan adalah integrasi machine learning dengan data non-tradisional, seperti opini publik dari media sosial. Model seperti Long Short-Term Memory (LSTM) dan Random Forest telah digunakan secara luas karena kemampuannya dalam mengenali pola dari data waktu dan menangani ketidakpastian dalam data pasar yang fluktuatif.

Selain itu, teknologi Natural Language Processing (NLP) juga mengalami kemajuan pesat. Model analisis sentimen tidak lagi terbatas pada metode leksikal sederhana, tetapi mulai beralih ke model berbasis pembelajaran mendalam seperti BERT, XLNet, dan model pre-trained lainnya. Model ini dapat memahami konteks dan nuansa bahasa alami dengan lebih baik.

Secara umum, perkembangan terkini menunjukkan bahwa model prediksi yang menggabungkan data historis dan sentimen sosial dapat memberikan akurasi yang lebih tinggi dan respon yang lebih cepat terhadap perubahan pasar, terutama pada kondisi yang tidak stabil atau penuh ketidakpastian.

BAB III

METODOLOGI PENELITIAN

3.1 Tahapan penelitian

Dalam upaya memperoleh hasil analisis data yang terarah dan dapat dipertanggungjawabkan secara metodologis, digunakan pendekatan berbasis kerangka kerja yang telah teruji seperti CRISP-DM, sebuah model proses standar independen dari industri untuk data mining yang terdiri dari enam fase iterative [18].

Tabel 3. 1 Deskripsi proses CRISP-DM [2]

Fase	Deskripsi
Business Understanding	Memahami situasi bisnis, menentukan tujuan data mining, seperti klasifikasi (dalam laporan ini) serta kriteria keberhasilan, dan menyusun rencana proyek.
Data Understanding	Mengumpulkan data dari sumber yang relevan, mengeksplorasi, mendeskripsikan, dan memeriksa kualitas data menggunakan analisis statistik.
Data Preparation	Melakukan seleksi data dengan kriteria inklusi-eksklusi, membersihkan data yang berkualitas buruk, serta membangun atribut turunan sesuai model yang akan digunakan.
Modeling	Memilih teknik pemodelan, menyusun kasus uji, membangun model, menetapkan parameter, lalu mengevaluasi model sesuai kriteria yang telah ditentukan.
Evaluation	Memeriksa hasil model terhadap tujuan bisnis awal, menginterpretasi hasil, menentukan tindakan selanjutnya, dan melakukan review keseluruhan proses.
Deployment	Menerapkan hasil melalui laporan akhir atau komponen perangkat lunak, serta merencanakan pemantauan dan pemeliharaan implementasi model.

Namun, dalam laporan kali ini, proses metologi hanya akan dilaksanakan sampai pada tahap *Evaluation*, sehingga tahapan Deployment tidak menjadi fokus kajian.

3.2 Deskripsi Dataset

Tabel 3. 2 Dataset

Tipe	Sumber	Ukuran	Attribut
sentimen	x/twitter	2232	15 (conversation_id_str, created_at, favorite_count, full_text, id_str, image_url, in_reply_to_screen_name, lang, location, quote_count, reply_count, retweet_count, tweet_url, user_id_str, username)
historis	website (investing.com)	37 / day	7 (Tanggal, Terakhir, Pembukaan, Tertinggi, Terendah, Vol., Perubahan%)

3.3 Algoritma / Data Mining Tools

Tabel 3. 3 Algoritma

Algoritma	Alasan
Random Forest	Kuat pada data dengan interaksi antar fitur & tahan noise. Cocok untuk regresi harga saham berdasarkan banyak lag features.
XGBoost	Sering outperform model lain dalam kompetisi prediksi harga karena fokus pada residual dan regularisasi kuat.
Logistic Regression	Baseline linear sederhana untuk memeriksa apakah data cukup dijelaskan oleh relasi linear, sangat interpretatif.
SVR	Menangkap hubungan non-linear dengan kernel (misalnya RBF), memfokuskan prediksi pada pola utama dengan mengabaikan outlier kecil.
MLP	Neural network dasar untuk mempelajari representasi kompleks antar waktu tanpa harus memprogram interaksi secara manual.

Tabel 3. 4 Tools

Tools	Fungsi
pandas	Mengelola data frame, generate lag features, moving average, RSI.
numpy	Operasi numerik cepat (vectorized), misalnya menghitung return.
matplotlib / seaborn	Visualisasi distribusi harga, heatmap korelasi, plot prediksi.

scikit-learn	Pipeline training: Random Forest, Logistic Regression, SVR, MLP; preprocessing (StandardScaler/MinMaxScaler), metrics (accuracy, confusion matrix).
xgboost	Model XGBoostClassifier, powerful untuk data tabular.
statsmodels	Uji stasionaritas (ADF Test) jika ingin mengecek pola.
yellowbrick	Visualisasi residual, ROC, class prediction error.
mlflow / wandb	Tracking experiment untuk tuning hyperparameter & logging metrics.

3.4 Evalusi Kinerja

Tabel 3. 5 Evaluasi

Evaluasi	Penjelasan	Alasan
Accuracy	Proporsi prediksi arah benar (misalnya naik vs turun) dari seluruh prediksi.	Untuk baseline sederhana: seberapa sering model benar dalam memprediksi arah.
Precision	Dari semua yang diprediksi naik, berapa yang benar-benar naik.	Menghindari false signals, penting jika cost salah beli mahal.
Recall	Dari semua hari yang benar-benar naik, berapa yang terprediksi naik.	
F1-Score	Harmonik rata-rata precision dan recall.	Seimbang memerhatikan missed naik (false negative) dan false naik (false positive).
ROC AUC Score	Area under curve ROC yang membandingkan True Positive Rate vs False Positive Rate di semua threshold.	Untuk melihat kemampuan model membedakan naik vs turun terlepas threshold.
Confusion Matrix	Matriks jumlah prediksi naik/turun vs aktual naik/turun.	Memberi gambaran kesalahan model, bisa fokus memperbaiki misclassification.
Directional Accuracy	Persentase prediksi arah benar (mirip accuracy), tapi kadang dihitung dari return positif/negatif.	Sangat relevan dalam trading untuk memastikan prediksi arah lebih sering tepat.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Lingkungan Eksperimen

Seluruh proses eksperimen dilakukan menggunakan Google Colab dengan Python 3.10. Tools dan pustaka utama yang digunakan mencakup pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, dan transformers. Implementasi dilakukan dalam tiga tahap utama: preprocessing data, eksplorasi data (EDA), dan pembuatan serta evaluasi model prediksi.

4.2 Preprocessing Data

Proses *preprocessing* data merupakan tahap fundamental dalam analisis sentimen, terutama saat mengolah data mentah dari media sosial yang bersifat tidak terstruktur. Tahap ini bertujuan untuk menghasilkan data bersih, konsisten, dan siap digunakan dalam proses klasifikasi sentimen menggunakan model bahasa berbasis *transformer*. Berikut adalah tahapan yang dilakukan dalam preprocessing dataset sentimen:

4.2.1 Penggabungan Dataset

Empat dataset yang berasal dari sumber berbeda, yakni datasetX-Nuril2(2024-1).csv, datasetX-Nuril3(2024-2).csv, datasetX-Fathir(2025).csv, dan datasetX-wisnu(2025).csv digabungkan menggunakan fungsi pd.concat(). Tujuannya adalah untuk memperbesar ukuran data dan memperkaya variasi konteks kalimat yang digunakan dalam pelatihan dan evaluasi model.

```
df_x1 = pd.read_csv("../data/raw/datasetX-Nuril2(2024-1).csv")
df_x2 = pd.read_csv("../data/raw/datasetX-Nuril3(2024-2).csv")
df_x3 = pd.read_csv("../data/raw/datasetX-Fathir(2025).csv")
df_x4 = pd.read_csv("../data/raw/datasetX-wisnu(2025).csv")

df_combined_x = pd.concat([df_x1, df_x2, df_x3, df_x4],
ignore_index=True)
df_combined_x
```

4.2.2 Penghapusan Duplikasi

Langkah selanjutnya adalah menghapus entri duplikat berdasarkan kolom full_text untuk memastikan bahwa tidak ada redundansi data yang dapat mempengaruhi distribusi kelas sentimen.

```
df_combined_x = df_combined_x.drop_duplicates(subset='full_text')
```

4.2.3 Seleksi Kolom dan Tipe Data

Beberapa kolom yang dianggap tidak relevan terhadap proses analisis sentimen dihapus, seperti username, user_id_str, dan tweet_url. Selain itu, kolom created_at diubah menjadi format datetime untuk memudahkan proses sorting berdasarkan waktu.

```
df_combined_x['created_at'] =
pd.to_datetime(df_combined_x['created_at'])

df_combined_x = df_combined_x.drop(columns=['username', 'user_id_str',
'tweet_url', 'retweet_count', 'reply_count', 'quote_count', 'location',
'in_reply_to_screen_name', 'image_url', 'favorite_count',
'conversation_id_str'])
```

4.2.4 Seleksi Bahasa

Hanya data yang berbahasa Indonesia (kode 'in') yang dipertahankan. Hal ini penting untuk menjaga konsistensi bahasa dan agar sesuai dengan model klasifikasi sentimen yang digunakan, yaitu model pra-latih bahasa Indonesia.

```
df_combined_x = df_combined_x[df_combined_x['lang'] == 'in']
```

4.2.5 Klasifikasi Sentimen

Analisis sentimen dilakukan menggunakan transformers pipeline dari HuggingFace, dengan model "w11wo/indonesian-roberta-base-sentiment-classifier". Setiap entri teks diklasifikasikan ke dalam label sentimen (Positive, Neutral, atau Negative), dan disertai dengan skor kepercayaan dari model.

```
classifier = pipeline(
    "sentiment-analysis",
    model="w11wo/indonesian-roberta-base-sentiment-classifier",
)

df_combined_x['sentiment_result'] =
df_combined_x['full_text'].apply(lambda x: classifier(x)[0])
```

```
df_combined_x['sentiment'] =
df_combined_x['sentiment_result'].apply(lambda x: x['label'])
df_combined_x['score'] =
df_combined_x['sentiment_result'].apply(lambda x: x['score'])
```

4.2.6 Penyimpanan Dataset Bersih

Dataset yang telah diproses dan diklasifikasikan disimpan sebagai file CSV (dataset_sentiment.csv) untuk digunakan pada tahap analisis lanjutan atau pelatihan model pembelajaran mesin.

```
dataset_sentiment.to_csv('../data/processed/dataset_sentiment.csv',
index=False)
```

4.2.7 Hasil Preprocessing Data

Sebelum dilakukan analisis lebih lanjut, data mentah hasil pengumpulan dari media sosial perlu melalui tahapan preprocessing untuk memastikan kualitas dan konsistensi data. Proses ini mencakup pembersihan data, penghapusan duplikasi, konversi format waktu, dan identifikasi sentimen terhadap setiap entri teks. Tahapan ini bertujuan untuk menghasilkan dataset yang valid, terstruktur, dan siap digunakan dalam proses analisis sentimen dan pemodelan lebih lanjut. Pada subbab ini disajikan hasil dari proses preprocessing yang dilakukan terhadap dataset yang telah dikumpulkan.

1. Struktur Data Sentimen
   Dataset hasil scraping tweet terdiri dari 1038 entri dengan empat atribut utama: created_at, full_text, id_str, dan lang. Seluruh entri terisi lengkap tanpa adanya nilai kosong, sebagaimana ditunjukkan oleh kode berikut:

   ```
   df_combined_x.info()
   ```

   Hasil:

   ```
   <class 'pandas.core.frame.DataFrame'>
   Index: 1038 entries, 0 to 2231
   Data columns (total 4 columns):
    #   Column      Non-Null Count  Dtype
   ---  ------      --------------  -----
    0   created_at  1038 non-null   datetime64[ns, UTC]
    1   full_text   1038 non-null   object
    2   id_str      1038 non-null   int64
    3   lang        1038 non-null   object
   dtypes: datetime64[ns, UTC](1), int64(1), object(2)
   memory usage: 40.5+ KB
   ```

2. Distribusi Kelas Sentimen

Setelah klasifikasi sentimen dilakukan, distribusi label menunjukkan bahwa mayoritas tweet tergolong dalam kategori netral (736 entri), diikuti oleh negatif (159 entri) dan positif (141 entri). Hal ini menunjukkan potensi ketidakseimbangan kelas yang perlu diperhatikan dalam proses modeling.

```
dataset_sentiment['sentiment'].value_counts()
```

Hasil:
sentiment
neutral 736
negative 159
positive 141
Name: count, dtype: int64

3. Struktur Dataset Historis Saham

Dataset historis saham terdiri dari 39 entri yang mencerminkan data pergerakan harga saham dalam rentang waktu tertentu. Terdapat tujuh atribut utama, yaitu Tanggal, Terakhir, Pembukaan, Tertinggi, Terendah, Vol., dan Perubahan%. Semua entri pada dataset ini terisi penuh, yang menunjukkan tidak adanya nilai null.

```
dataset_historis.info()
```

Hasil:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 7 columns):

#	Column	Non-Null Count	Dtype
0	Tanggal	39 non-null	object
1	Terakhir	39 non-null	float64
2	Pembukaan	39 non-null	float64
3	Tertinggi	39 non-null	float64
4	Terendah	39 non-null	float64
5	Vol.	39 non-null	object
6	Perubahan%	39 non-null	object

dtypes: float64(4), object(3)
memory usage: 2.3+ KB

4. Struktur Dataset Model

Dataset model merupakan hasil integrasi antara data historis saham dan hasil agregasi dari analisis sentimen harian. Dataset ini juga terdiri dari 39 entri, dengan total 13 atribut yang mencakup informasi harga, volume, perubahan persentase, serta fitur-fitur sentimen seperti avg_signed_sentiment, count_positive, count_negative, count_neutral, dan total_tweets.

Hampir seluruh kolom terisi penuh, kecuali lima fitur sentimen yang memiliki satu entri kosong (NaN). Hal ini masih dapat ditangani dengan teknik interpolasi atau penghapusan baris, tergantung pendekatan analisis.

```
dataset_model.info()
```

Hasil:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Tanggal              39 non-null     datetime64[ns]
 1   Terakhir             39 non-null     float64
 2   Pembukaan            39 non-null     float64
 3   Tertinggi            39 non-null     float64
 4   Terendah             39 non-null     float64
 5   Vol.                 39 non-null     object
 6   Perubahan%           39 non-null     object
 7   date                 39 non-null     object
 8   avg_signed_sentiment 38 non-null     float64
 9   count_positive       38 non-null     float64
 10  count_negative       38 non-null     float64
 11  count_neutral        38 non-null     float64
 12  total_tweets         38 non-null     float64
dtypes: datetime64[ns](1), float64(9), object(3)
memory usage: 4.1+ KB
```

4.3 Pembentukan Dataset Model

Tahapan pembentukan dataset model dilakukan dengan cara menggabungkan data historis peristiwa dengan hasil analisis sentimen dari data media sosial. Langkah ini bertujuan untuk memperoleh satu data frame terpadu yang siap digunakan dalam proses pemodelan dan analisis lanjutan.

4.3.1 Membaca dan Membersihkan Data Historis

Langkah awal yaitu memuat data historis dari direktori ../data/processed/dataset_historis.csv dan memastikan kolom tanggal dalam format datetime.

```
dataset_historis = pd.read_csv("../data/processed/dataset_historis.csv")
```

```
dataset_historis['Tanggal'] =
pd.to_datetime(dataset_historis['Tanggal'], dayfirst=True,
errors='coerce')


dataset_sentiment['date'] = dataset_sentiment['created_at'].dt.date
dataset_historis['date'] =
pd.to_datetime(dataset_historis['Tanggal']).dt.date
```

Tahap ini memastikan bahwa data historis memiliki format tanggal yang sesuai dan dapat digunakan untuk penggabungan data selanjutnya.

4.3.2 Konversi dan Penandaan Sentimen

Data sentimen dari media sosial sebelumnya telah dianalisis dan diklasifikasikan ke dalam tiga kategori: positive, neutral, dan negative. Kategori ini kemudian dikonversikan ke dalam nilai numerik menggunakan dictionary sentiment_sign.

```
sentiment_sign = {'positive': 1, 'neutral': 0, 'negative': -1}
dataset_sentiment['sentiment_sign'] =
dataset_sentiment['sentiment'].map(sentiment_sign)
```

Selanjutnya, skor sentimen yang telah diberikan dari hasil analisis kemudian dikalikan dengan nilai tanda (sign) untuk mendapatkan skor bertanda (signed score).

```
dataset_sentiment['signed_score'] =
dataset_sentiment['sentiment_sign'] * dataset_sentiment['score']
```

4.3.3 Agregasi Sentimen Harian

Data sentimen kemudian dikelompokkan berdasarkan tanggal dengan melakukan agregasi terhadap beberapa metrik penting:

- avg_signed_sentiment: rerata skor sentimen bertanda per hari.
- count_positive, count_negative, count_neutral: jumlah masing-masing jenis sentimen dalam satu hari.
- total_tweets: total jumlah tweet yang dianalisis pada tanggal tersebut.

Kode berikut digunakan untuk menghasilkan agregasi ini:

```
dataset_grouped = dataset_sentiment.groupby('date').agg(
    avg_signed_sentiment=('signed_score', 'mean'),
    count_positive=('sentiment', lambda x: (x=='positive').sum()),
    count_negative=('sentiment', lambda x: (x=='negative').sum()),
    count_neutral=('sentiment', lambda x: (x=='neutral').sum()),
    total_tweets=('sentiment', 'count')
).reset_index()
```

Hasil dari agregasi ini menghasilkan data sentimen harian yang siap digabungkan dengan data historis.

4.3.4 Penggabungan Data Historis dan Sentimen

Setelah data historis dan data sentimen sama-sama memiliki kolom date, kedua data tersebut digabungkan menggunakan metode merge dengan jenis *left join*. Hal ini memastikan semua baris pada data historis tetap dipertahankan meskipun tidak semua tanggal memiliki data sentimen.

```
dataset_model = pd.merge(dataset_historis, dataset_grouped, on='date', how='left')
```

Langkah ini menghasilkan satu data frame baru bernama dataset_model yang menggabungkan informasi historis dan sentimen sosial secara lengkap.

4.3.5 Ekspor Dataset Model

Dataset model yang telah terbentuk kemudian disimpan ke dalam file .csv untuk keperluan analisis lanjutan.

```
dataset_model.to_csv('../data/processed/dataset_model.csv', index=False)
```

Dengan demikian, proses pembentukan dataset model telah selesai. Dataset ini berisi data historis peristiwa beserta dimensi sosial yang diwakili oleh analisis sentimen dari media sosial pada hari yang sama. Ini memungkinkan pendekatan pemodelan yang lebih komprehensif dengan mempertimbangkan faktor sosial dan temporal sekaligus.

4.4 Eksplorasi Dataset Saham dan Sentimen

Sebelum membangun model prediktif, penting untuk memahami karakteristik dasar dari data yang digunakan. Tahapan ini dikenal sebagai Exploratory Data Analysis (EDA). Melalui EDA, kita dapat mengidentifikasi pola, distribusi, outlier, nilai hilang, serta hubungan antar variabel yang dapat berpengaruh terhadap model akhir. Tahap ini juga berfungsi sebagai landasan dalam pengambilan keputusan pada proses pra-pemrosesan dan pemilihan fitur. Langkah pertama dalam EDA adalah memuat dataset dan memastikan bahwa format serta struktur data telah sesuai.

4.4.1 Memuat dan Menyiapkan Dataset

Langkah pertama dalam eksplorasi data adalah memuat dataset dari file yang telah diproses sebelumnya, yaitu dataset_model.csv. Dataset ini telah melewati proses pra-pemrosesan awal seperti pembersihan dan penggabungan sumber data. Setelah data dimuat, dilakukan pengecekan struktur data dan penghapusan nilai kosong (missing values) untuk memastikan kualitas data yang akan digunakan dalam analisis selanjutnya.

```
import pandas as pd

 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
 import matplotlib.pyplot as plt
 import seaborn as sns

 data = pd.read_csv("../data/processed/dataset_model.csv")

 data.info()

 data.dropna(inplace=True)
```

4.4.2 Pemeriksaan Autokorelasi Harga Saham

Untuk memahami pola waktu (time dependency) pada harga saham, digunakan analisis autokorelasi. Plot ACF (Autocorrelation Function) digunakan untuk melihat apakah nilai harga saham pada suatu waktu dipengaruhi oleh nilai pada waktu sebelumnya. Hal ini penting untuk mengidentifikasi kemungkinan adanya komponen musiman atau tren jangka panjang.

```
plot_acf(data['Terakhir'].dropna(), lags=30)
plt.title("Autocorrelation of Closing Price")
plt.show()
```

Gambar 4. 1 Autocorrelation of Closing Price

Hasil plot menunjukkan bahwa nilai autokorelasi menurun secara bertahap dan tetap signifikan hingga sekitar lag ke-8, yang menunjukkan adanya autokorelasi positif kuat jangka pendek. Setelah

itu, nilai autokorelasi berubah menjadi negatif dan tetap signifikan secara moderat hingga sekitar lag ke-20 sebelum akhirnya mendekati nol.

4.4.3 Hubungan Harga Saham dan Rata-rata Sentimen

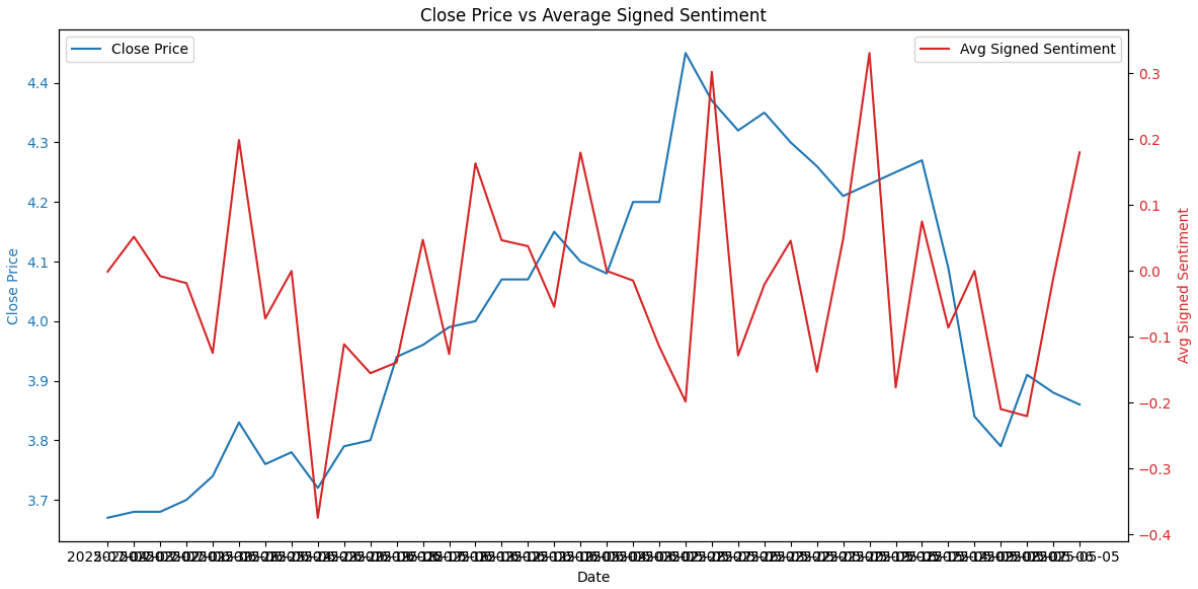
Visualisasi ini bertujuan untuk melihat keterkaitan antara pergerakan harga saham dengan sentimen publik yang diukur melalui nilai rata-rata sentimen bertanda (avg_signed_sentiment). Dengan menggunakan grafik dua sumbu, ditampilkan tren harga saham dan sentimen dalam rentang waktu yang sama untuk mengamati apakah perubahan sentimen dapat mempengaruhi harga saham.

```
fig, ax1 = plt.subplots(figsize=(12,6))

color = 'tab:blue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Close Price', color=color)
ax1.plot(data['date'],  data['Terakhir'],  color=color,  label='Close
Price')
ax1.tick_params(axis='y', labelcolor=color)
ax1.legend(loc='upper left')

ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Avg Signed Sentiment', color=color)
ax2.plot(data['date'],   data['avg_signed_sentiment'],   color=color,
label='Avg Signed Sentiment')
ax2.tick_params(axis='y', labelcolor=color)
ax2.legend(loc='upper right')

plt.title('Close Price vs Average Signed Sentiment')
plt.tight_layout()
plt.show()
```



Gambar 4. 2 Close Price vs Average Signed Sentiment

Berdasarkan visualisasi tersebut, terlihat bahwa pergerakan sentimen tidak secara langsung selaras dengan tren harga saham. Terdapat beberapa periode di mana lonjakan atau penurunan sentimen tidak diikuti oleh perubahan signifikan dalam harga saham, dan sebaliknya. Hal ini menandakan bahwa meskipun sentimen publik memiliki peran dalam membentuk ekspektasi pasar, hubungannya terhadap harga saham bersifat lemah atau tidak linier dalam jangka pendek.

4.4.4 Konversi Volume Transaksi

Kolom volume transaksi (Vol.) pada dataset masih dalam format string yang mengandung satuan seperti 'K' (ribu) dan 'M' (juta). Agar dapat digunakan dalam analisis numerik dan korelasi, kolom ini perlu dikonversi menjadi nilai numerik murni. Fungsi parse_volume digunakan untuk melakukan parsing nilai string menjadi angka desimal yang sesuai.

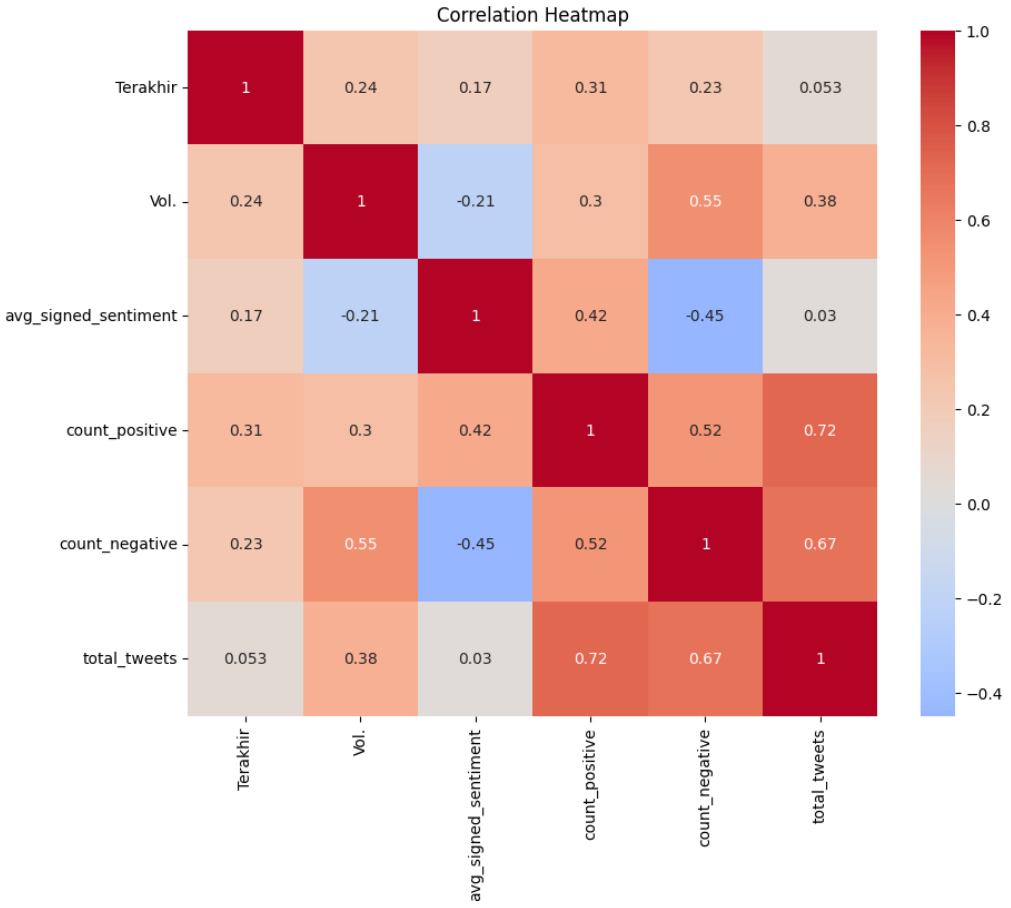
```
def parse_volume(vol_str):
    if isinstance(vol_str, str):
        vol_str = vol_str.replace(',', '.')  # ubah koma jadi titik
        if vol_str.endswith('M'):
            return float(vol_str[:-1]) * 1_000_000
        elif vol_str.endswith('K'):
            return float(vol_str[:-1]) * 1_000
        else:
            return float(vol_str)
    return vol_str  # jika sudah float atau NaN

data['Vol.'] = data['Vol.'].apply(parse_volume)
```

4.4.5 Korelasi Antar Variabel

Analisis korelasi membantu mengidentifikasi sejauh mana hubungan antar variabel numerik seperti harga saham, volume transaksi, sentimen, dan jumlah tweet. Heatmap korelasi memberikan visualisasi yang informatif mengenai kekuatan dan arah hubungan antar variabel, yang dapat digunakan sebagai dasar dalam pemilihan fitur untuk model prediktif.

```
plt.figure(figsize=(10,8))
corr = data[['Terakhir', 'Vol.', 'avg_signed_sentiment',
'count_positive', 'count_negative', 'total_tweets']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```



Gambar 4. 3 Corelation Heatmap

Berdasarkan heatmap di atas, ditemukan beberapa poin penting:

- Volume transaksi (Vol.) menunjukkan korelasi positif moderat terhadap harga penutupan saham (Terakhir) sebesar 0.24, yang mengindikasikan bahwa lonjakan aktivitas pasar cenderung berasosiasi dengan perubahan harga.
- Rata-rata sentimen (avg_signed_sentiment) hanya memiliki korelasi lemah terhadap harga saham (0.17). Ini menyiratkan bahwa meskipun sentimen memiliki nilai

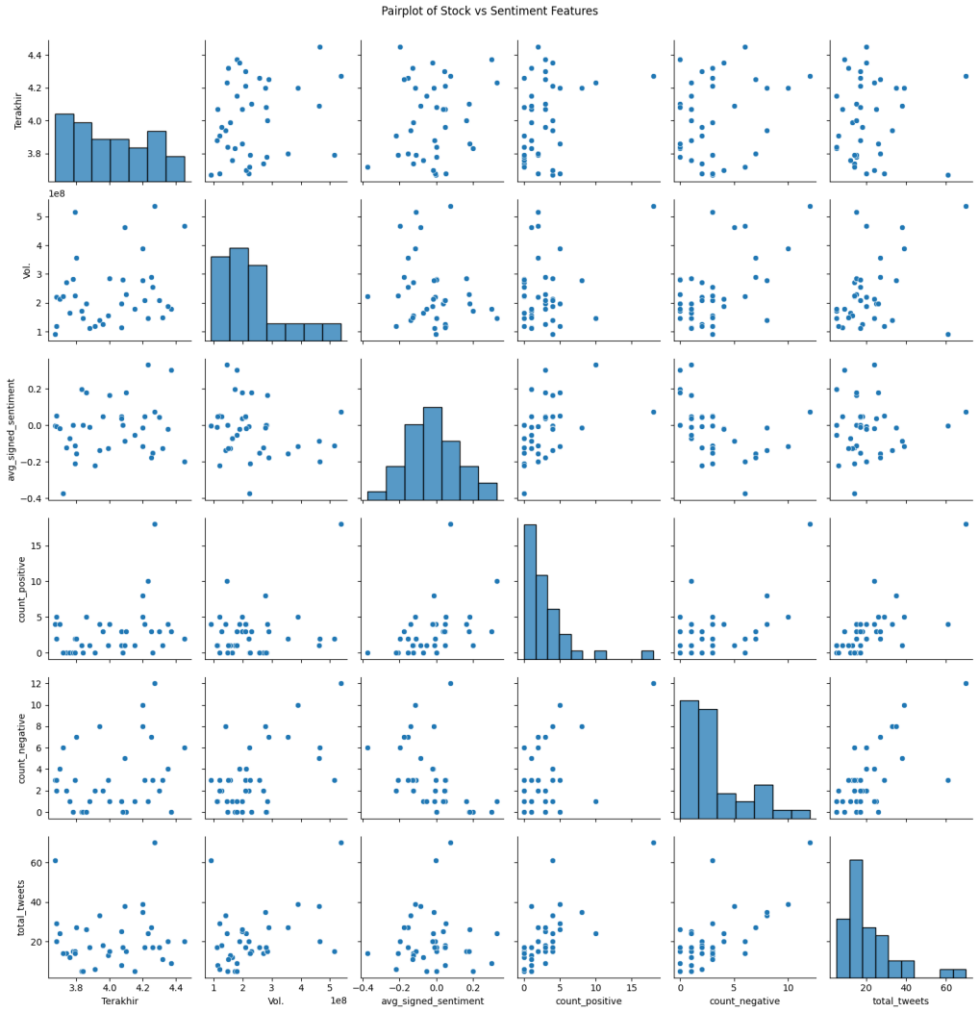
informasi, pengaruhnya terhadap harga bersifat tidak dominan dan kemungkinan berperan sebagai faktor pelengkap.

- Jumlah tweet positif dan negatif memiliki korelasi yang kuat terhadap total tweet, masing-masing sebesar 0.72 dan 0.67, yang logis mengingat total tweet merupakan akumulasi dari semua kategori sentimen.
- Jumlah tweet positif (count_positive) menunjukkan korelasi lebih kuat terhadap rata-rata sentimen (avg_signed_sentiment) (0.42) dibandingkan jumlah tweet negatif (-0.45), yang mengindikasikan bahwa sentimen rata-rata lebih sensitif terhadap sentimen positif.

4.4.6 Distribusi dan Hubungan Antar Fitur

Visualisasi distribusi dan hubungan antar variabel dapat dilakukan dengan menggunakan pairplot. Visualisasi ini memudahkan dalam melihat pola sebaran data dan kemungkinan hubungan linear/non-linear antar variabel seperti harga saham, volume, dan sentimen. Ini juga berguna untuk mendeteksi outlier secara visual.

```
sns.pairplot(data[['Terakhir', 'Vol.', 'avg_signed_sentiment',
'count_positive', 'count_negative', 'total_tweets']])
plt.suptitle("Pairplot of Stock vs Sentiment Features", y=1.02)
plt.show()
```



Gambar 4. 4 Pairplot of Stock vs Sentiment Features

Dari visualisasi ini, terlihat bahwa sebagian besar fitur seperti count_positive, count_negative, dan total_tweets menunjukkan pola distribusi miring ke kanan (right-skewed), mengindikasikan bahwa sebagian besar nilai berada pada rentang rendah dengan beberapa nilai ekstrem yang tinggi. Distribusi volume (Vol.) juga tampak sangat bervariasi, dengan nilai yang tersebar luas dan beberapa titik ekstrem.

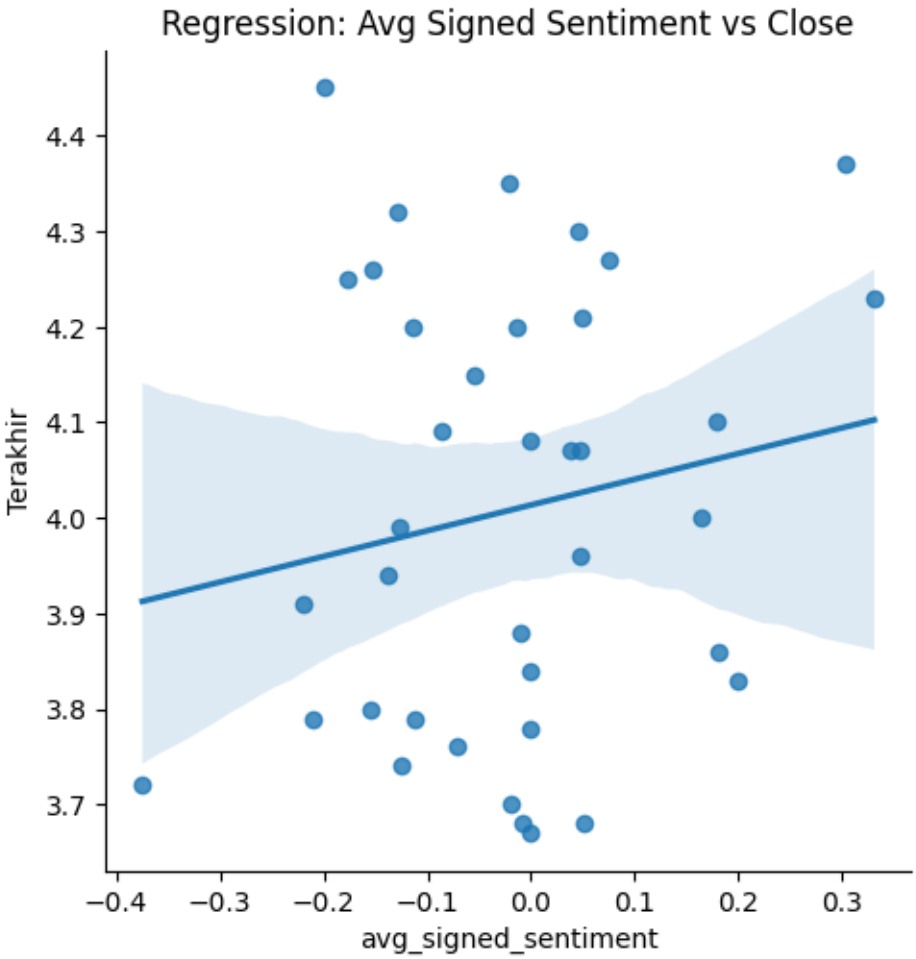
Hubungan antar variabel tampak lemah secara visual, terutama antara harga saham (Terakhir) dengan variabel sentimen. Meskipun terdapat beberapa pola hubungan positif moderat, seperti antara count_positive dan total_tweets, secara umum tidak ditemukan hubungan linear yang kuat antar sebagian besar fitur. Hal ini menguatkan hasil dari analisis korelasi sebelumnya bahwa faktor-faktor sentimen sosial media mungkin bersifat pelengkap daripada penentu utama dalam fluktuasi harga saham.

Secara keseluruhan, pairplot memberikan gambaran menyeluruh terhadap struktur data dan memberikan wawasan awal yang penting sebelum proses pemodelan lebih lanjut dilakukan.

4.4.7 Regresi Harga Saham terhadap Sentimen

Untuk mengevaluasi apakah sentimen publik berpengaruh secara linier terhadap harga saham, dilakukan analisis regresi linier sederhana antara avg_signed_sentiment dan Terakhir. Plot regresi memberikan gambaran tentang arah dan kekuatan hubungan antara kedua variabel tersebut, serta potensi signifikansinya secara statistik.

```
sns.lmplot(x='avg_signed_sentiment', y='Terakhir', data=data)
plt.title('Regression: Avg Signed Sentiment vs Close')
plt.show()
```



Gambar 4. 5 Regression: Avg Signed Sentiment vs Close

Untuk menguji potensi hubungan linier antara sentimen publik dan harga saham, dilakukan analisis regresi linier sederhana antara nilai rata-rata sentimen bertanda (avg_signed_sentiment) dan harga penutupan saham (Terakhir). Visualisasi ini bertujuan untuk mengevaluasi sejauh mana perubahan dalam sentimen publik dapat menjelaskan variasi harga saham.

Dari grafik regresi, terlihat adanya tren linier positif antara kedua variabel, yang menunjukkan bahwa peningkatan nilai sentimen cenderung diikuti oleh peningkatan harga saham. Namun demikian, sebaran data yang cukup menyebar di sekitar garis regresi mencerminkan adanya

variabilitas yang tinggi. Hal ini mengindikasikan bahwa meskipun terdapat kecenderungan arah hubungan yang positif, sentimen bukanlah satu-satunya faktor yang memengaruhi harga saham. Dengan demikian, hasil regresi ini memperkuat pemahaman bahwa faktor sentimen dapat menjadi salah satu indikator dalam model prediksi harga saham, tetapi perlu dikombinasikan dengan variabel-variabel fundamental atau teknikal lainnya untuk meningkatkan akurasi prediksi.

4.5 Tabel Hasil Eksperimen / Model

Tabel 4. 1 Tabel Hasil Eksperimen / Model

Model	Acc. Mean	Prec. Mean	Recall Mean	F1 Mean	ROC AUC	Dir. Acc	Acc. Std	Prec. Std	Recall Std	F1 Std	ROC AUC Std	Dir. Acc Std
RandomForest_3_fold	0.6111	0.4167	0.6667	0.5079	0.8958	0.5000	0.2079	0.3118	0.4714	0.3675	0.1062	0.1361
RandomForest_5_fold	0.6111	0.4167	0.6667	0.5079	0.8333	0.5000	0.2079	0.3118	0.4714	0.3675	0.1179	0.1361
RandomForest_10_fold	0.6111	0.4167	0.6667	0.5079	0.8958	0.5000	0.2079	0.3118	0.4714	0.3675	0.1062	0.1361
XGBoost_3_fold	0.5556	0.3833	0.6667	0.4762	0.8333	0.5000	0.2079	0.3064	0.4714	0.3563	0.2357	0.1361
XGBoost_5_fold	0.6667	0.4722	0.6667	0.5524	0.6759	0.6111	0.2357	0.3356	0.4714	0.3913	0.1249	0.0786
XGBoost_10_fold	0.5556	0.3833	0.6667	0.4762	0.8333	0.5000	0.2079	0.3064	0.4714	0.3563	0.2357	0.1361
LogReg_3_fold	0.5556	0.2500	0.3333	0.2857	0.8009	0.5556	0.2079	0.3536	0.4714	0.4041	0.1540	0.0786
LogReg_5_fold	0.6111	0.4167	0.5000	0.4524	0.7963	0.5556	0.2079	0.3118	0.4082	0.3515	0.2144	0.0786
LogReg_10_fold	0.5556	0.2500	0.3333	0.2857	0.8009	0.5556	0.2079	0.3536	0.4714	0.4041	0.1540	0.0786
SVC_3_fold	0.6667	0.5000	0.6667	0.5556	0.5833	0.6111	0.2722	0.4082	0.4714	0.4157	0.4249	0.2833
SVC_5_fold	0.6667	0.5000	0.6667	0.5556	0.5833	0.6111	0.2722	0.4082	0.4714	0.4157	0.4249	0.2833
SVC_10_fold	0.6667	0.5000	0.6667	0.5556	0.5833	0.6111	0.2722	0.4082	0.4714	0.4157	0.4249	0.2833
MLP_3_fold	0.6111	0.4167	0.5000	0.4524	0.7917	0.5556	0.2079	0.3118	0.4082	0.3515	0.1559	0.0786
MLP_5_fold	0.6111	0.4444	0.5000	0.4667	0.7917	0.6111	0.2833	0.4157	0.4082	0.4110	0.2946	0.2833
MLP_10_fold	0.6111	0.4444	0.5000	0.4667	0.7917	0.6111	0.2833	0.4157	0.4082	0.4110	0.2946	0.2833

4.6 Interpretasi Hasil Evaluasi Model

Tabel 4. 2 Tabel Interpretasi Hasil Evaluasi Model

Metrik	Model Terbaik	Nilai	Catatan
Accuracy	SVC dan XGBoost_5_fold	0.6667	Konsisten pada skema 3, 5, dan 10 fold
Precision	SVC (semua fold)	0.5000	Presisi tinggi, mengurangi false positive
Recall	Random Forest (semua fold), XG Boost (semua fold), SVC (semua fold)	0.6667	Model cukup baik mendeteksi kelas positif
F1-score	SVC dan XGBoost_5_fold	0.5556/0.5524	Menunjukkan keseimbangan antara presisi dan recall
ROC AUC	RandomForest (3&10fold)	0.8958	Kemampuan terbaik dalam membedakan kelas
Stabilitas	LogReg_5_fold, MLP_3_fold	std rendah	Variasi antar fold kecil, artinya performa relatif konsisten

4.7 Analisis Keunggulan dan Keterbatasan Model

1. Random Forest

Keunggulan:

- Memiliki ROC AUC tertinggi (0.8958), artinya sangat baik dalam membedakan antara kelas positif dan negatif.
- Cocok untuk data yang kompleks dan memiliki fitur non-linier.
- Tidak mudah overfitting karena menggunakan banyak pohon.

Keterbatasan:

- F1-score dan precision rendah, yang artinya banyak false positives atau false negatives.
- Performanya tidak meningkat signifikan meskipun jumlah fold ditambah (3, 5, dan 10 fold memiliki hasil serupa).

- Cenderung kurang presisi pada kelas minoritas.

2. XGBoost

Keunggulan:

- Mencapai kombinasi metrik yang seimbang terutama pada skema 5-fold (akurat dan presisi bagus).
- Cenderung memiliki generalisasi lebih baik karena optimisasi gradien dan regularisasi internal.
- F1-score tinggi (0.5524) menunjukkan keseimbangan presisi dan recall.

Keterbatasan:

- Versi 3-fold dan 10-fold menunjukkan penurunan performa, mengindikasikan sensitif terhadap jumlah data latih/validasi.
- Implementasi dan tuning lebih kompleks dibanding model lain.

3. SVC (Support Vector Classifier)

Keunggulan:

- F1-score tertinggi dan paling konsisten di semua fold (0.5556).
- Performa bagus meskipun dengan jumlah data terbatas.
- Cocok untuk data tidak seimbang, karena mencari hyperplane terbaik dengan margin maksimum.

Keterbatasan:

- ROC AUC rendah (0.5833), menunjukkan kelemahan dalam probabilistic scoring (tidak optimal jika thresholding diperlukan).
- Waktu pelatihan dan prediksi lebih lama pada dataset besar.

4. Logistic Regression

Keunggulan:

- Model yang sederhana dan mudah diinterpretasi.
- Stabil (standar deviasi rendah), artinya hasil tidak banyak berubah antar fold.
- Cocok sebagai baseline awal.

Keterbatasan:

- Performanya jauh lebih rendah dibanding model lain, terutama di F1 dan precision.
- Kurang mampu menangani data yang tidak linear atau kompleks.

5. MLP (Multi-Layer Perceptron)

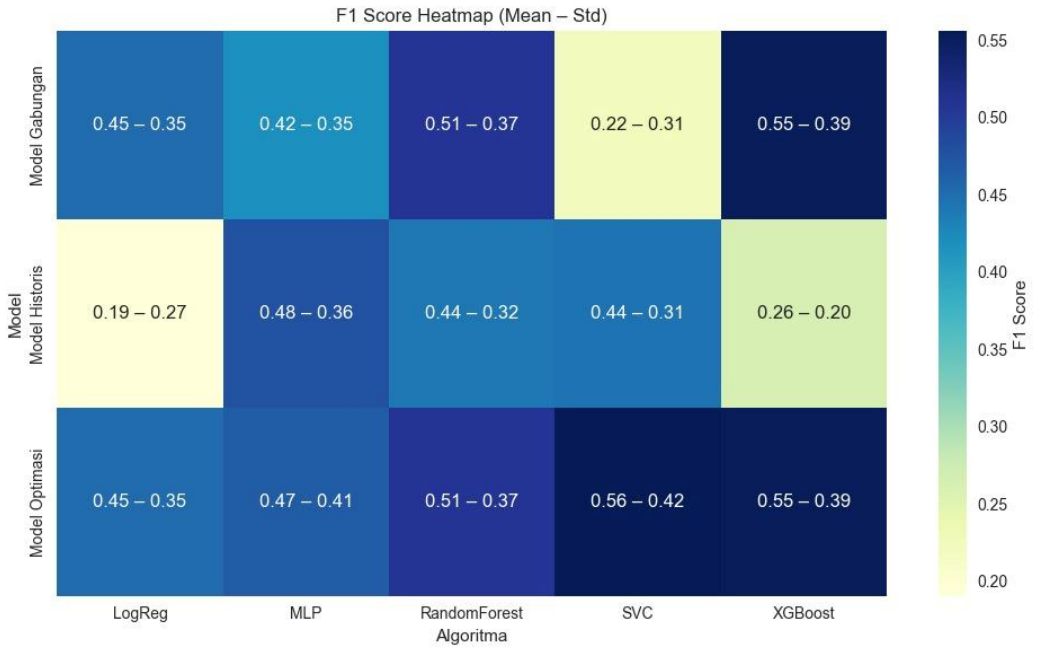
Keunggulan:

- Memiliki metrik yang cukup stabil seperti Logistic Regression.
- Dapat menangkap pola non-linear karena arsitektur jaringan saraf.

Keterbatasan:

- F1-score dan precision masih rendah (sekitar 0.46).
- Rentan terhadap overfitting tanpa regularisasi yang tepat.
- Butuh tuning lebih lanjut (jumlah layer, neuron, learning rate).

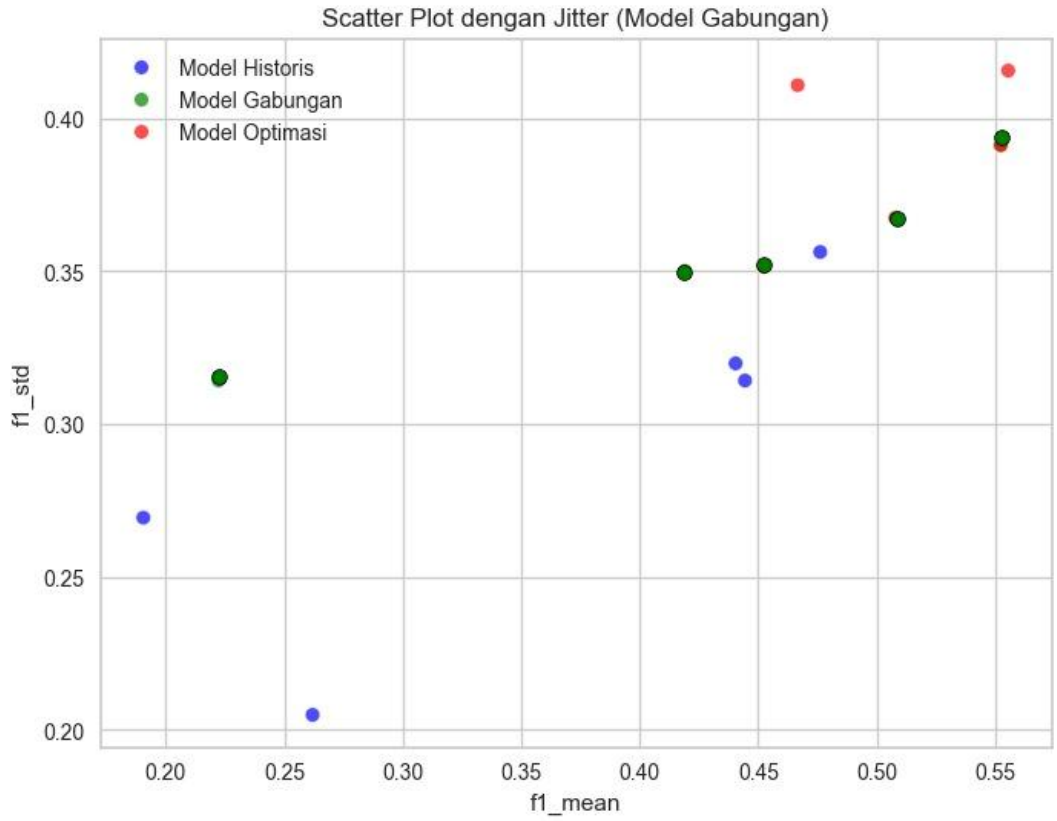
4.8 Visualisasi dan Hasil Model



Gambar 4. 6 F1 Score Heatmap

Heatmap di atas menunjukkan perbandingan skor F1 (dalam format mean – std) dari berbagai algoritma klasifikasi (LogReg, MLP, RandomForest, SVC, dan XGBoost) yang diterapkan pada tiga jenis model: Model Gabungan, Model Historis, dan Model Optimasi. Semakin tinggi nilai F1 mean dan semakin rendah standar deviasi (std), maka kinerja model tersebut dianggap lebih baik dan konsisten.

Secara umum, Model Optimasi menunjukkan performa terbaik secara konsisten, terutama pada algoritma SVC dan XGBoost yang masing-masing mencapai skor F1 0.56 – 0.42 dan 0.55 – 0.39. Ini menunjukkan bahwa setelah proses optimasi, model berhasil meningkatkan akurasi prediksi dengan tetap menjaga stabilitas. Model Gabungan juga menunjukkan performa yang kompetitif, terutama pada XGBoost (0.55 – 0.39) dan RandomForest (0.51 – 0.37), meskipun SVC tampil buruk (0.22 – 0.31). Sementara itu, Model Historis cenderung memiliki performa yang lebih rendah dan tidak konsisten, terutama pada LogReg yang hanya memiliki F1 score 0.19 – 0.27. Dengan demikian, heatmap ini mengindikasikan bahwa kombinasi optimasi dan pemilihan algoritma yang tepat (seperti SVC dan XGBoost) sangat berpengaruh terhadap peningkatan kinerja model secara keseluruhan.



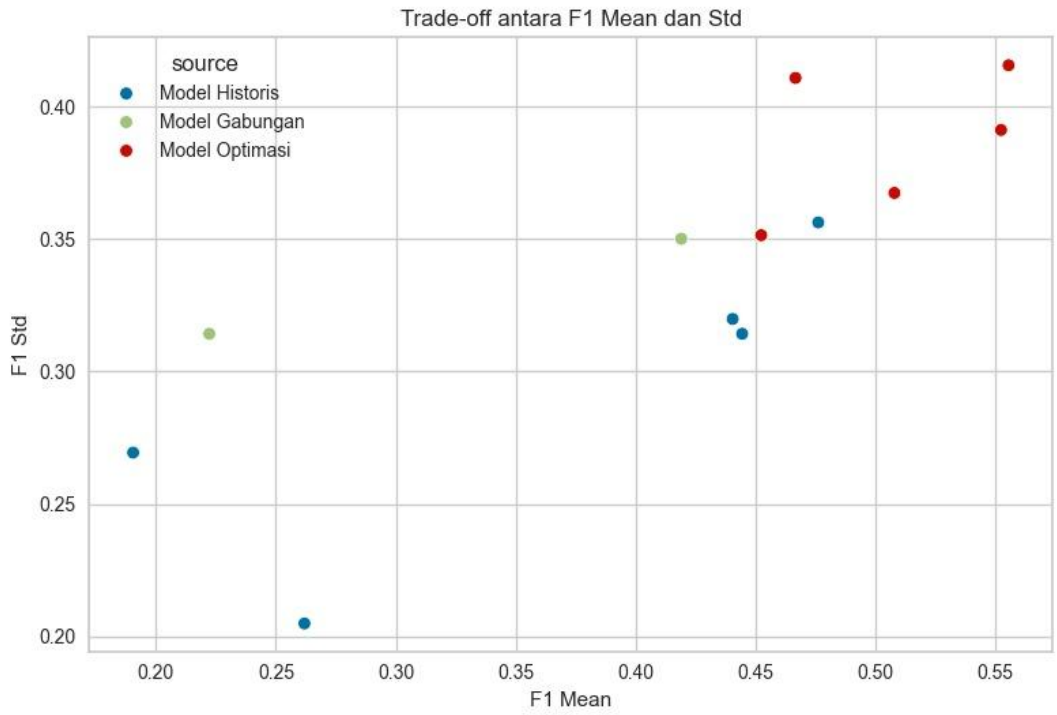
Gambar 4. 7 Scatter Plot dengan Jitter

Plot scatter di atas menggambarkan hubungan trade-off antara rata-rata skor F1 (F1 Mean) dan standar deviasi skor F1 (F1 Std) dari tiga jenis model: Model Historis, Model Gabungan, dan Model Optimasi. Sumbu horizontal merepresentasikan seberapa baik rata-rata kinerja model dalam hal klasifikasi (semakin ke kanan, semakin tinggi nilai F1 Mean), sedangkan sumbu vertikal menunjukkan tingkat konsistensi atau stabilitas model (semakin ke bawah, semakin kecil deviasinya, artinya lebih stabil).

Dari persebaran titik-titik, terlihat bahwa Model Optimasi (ditandai dengan warna merah) umumnya menempati area kanan atas, yang mengindikasikan bahwa model ini memiliki performa rata-rata yang tinggi namun dengan variasi antar percobaan yang juga tinggi—sebuah indikasi bahwa meskipun akurat, model ini belum sepenuhnya stabil. Model Gabungan (berwarna hijau) cenderung berada di tengah-tengah, menunjukkan kinerja yang cukup baik dengan tingkat stabilitas yang juga sedang. Sementara itu, Model Historis (berwarna biru) tersebar dari area kiri bawah hingga ke tengah, mencerminkan performa rata-rata yang lebih rendah namun dalam beberapa kasus menawarkan stabilitas yang lebih baik.

Secara keseluruhan, grafik ini mengilustrasikan bahwa ada peningkatan performa signifikan pada Model Optimasi dibandingkan model lainnya, namun dengan kompromi berupa peningkatan

ketidakstabilan. Sebaliknya, Model Historis mungkin menawarkan kestabilan yang lebih tinggi dalam beberapa kasus, tetapi dengan kemampuan prediksi yang relatif lebih rendah.



Gambar 4. 8 Trade-off antara F1 Mean dan Std

Plot scatter di atas menggambarkan hubungan trade-off antara rata-rata skor F1 (F1 Mean) dan standar deviasi skor F1 (F1 Std) dari tiga jenis model: Model Historis, Model Gabungan, dan Model Optimasi. Sumbu horizontal merepresentasikan seberapa baik rata-rata kinerja model dalam hal klasifikasi (semakin ke kanan, semakin tinggi nilai F1 Mean), sedangkan sumbu vertikal menunjukkan tingkat konsistensi atau stabilitas model (semakin ke bawah, semakin kecil deviasinya, artinya lebih stabil).

Dari persebaran titik-titik, terlihat bahwa Model Optimasi (ditandai dengan warna merah) umumnya menempati area kanan atas, yang mengindikasikan bahwa model ini memiliki performa rata-rata yang tinggi namun dengan variasi antar percobaan yang juga tinggi—sebuah indikasi bahwa meskipun akurat, model ini belum sepenuhnya stabil. Model Gabungan (berwarna hijau) cenderung berada di tengah-tengah, menunjukkan kinerja yang cukup baik dengan tingkat stabilitas yang juga sedang. Sementara itu, Model Historis (berwarna biru) tersebar dari area kiri bawah hingga ke tengah, mencerminkan performa rata-rata yang lebih rendah namun dalam beberapa kasus menawarkan stabilitas yang lebih baik.

Secara keseluruhan, grafik ini mengilustrasikan bahwa ada peningkatan performa signifikan pada Model Optimasi dibandingkan model lainnya, namun dengan kompromi berupa peningkatan ketidakstabilan. Sebaliknya, Model Historis mungkin menawarkan kestabilan yang lebih tinggi dalam beberapa kasus, tetapi dengan kemampuan prediksi yang relatif lebih rendah.

BAB V

KESIMPULAN DAN SARAN

5.1 Ringkasan Temuan Utama

Penelitian ini menemukan bahwa integrasi data sentimen Twitter dengan data historis saham dapat meningkatkan akurasi model prediksi harga saham. Analisis menunjukkan:

- Sentimen positif pada tweet berkorelasi dengan kenaikan harga saham, sedangkan sentimen negatif cenderung diikuti oleh penurunan harga saham dalam jangka pendek.
- Model prediksi yang menggabungkan fitur sentimen Twitter dan data historis menghasilkan performa yang lebih baik dibanding model berbasis data historis saja, terutama pada periode volatilitas pasar.
- Dari algoritma yang diuji, XGBoost dan Random Forest menunjukkan kinerja yang lebih stabil dan akurat dalam menangkap fluktuasi harga saham ketika ditambahkan fitur sentimen.
- Model RoBERTa efektif dalam klasifikasi sentimen tweet berbahasa Indonesia sehingga dapat menghasilkan fitur sentimen yang relevan bagi prediksi harga saham.
- Penambahan fitur sentimen membantu meningkatkan sensitivitas model terhadap perubahan harga saham yang dipicu oleh faktor psikologis dan persepsi publik.

5.2 Jawaban Atas Rumusan Masalah

Hasil penelitian ini menunjukkan bahwa analisis sentimen real-time dari Twitter memiliki korelasi yang signifikan dengan pergerakan harga saham, di mana sentimen positif dalam tweet cenderung diikuti oleh kenaikan harga saham, sedangkan sentimen negatif sering mendahului penurunan harga saham. Selain itu, model prediksi yang hanya menggunakan data historis memiliki akurasi yang lebih rendah dibandingkan dengan model yang menggabungkan data historis dan sentimen Twitter. Penambahan fitur sentimen Twitter terbukti dapat meningkatkan akurasi prediksi secara signifikan serta membantu model menjadi lebih responsif dan tahan terhadap volatilitas pasar yang tinggi. Dengan demikian, integrasi data historis dan data sentimen Twitter memberikan hasil yang lebih optimal untuk memprediksi harga saham dibandingkan penggunaan data historis saja.

5.3 Saran Untuk Pengembangan Lanjut

1. Memperluas sumber data sentimen, misalnya dengan menambahkan data dari Instagram, YouTube, dan berita ekonomi online untuk meningkatkan cakupan analisis opini publik.
2. Menggunakan model deep learning seperti LSTM, Bi-LSTM, atau Transformer untuk menangkap dependensi waktu dan hubungan kompleks dalam data historis dan sentimen.
3. Membangun sistem prediksi real-time berbasis streaming data Twitter, dilengkapi pipeline big data untuk implementasi praktis di pasar modal.
4. Menambahkan variabel makroekonomi (kurs, inflasi, suku bunga, IHSG) dalam model prediksi untuk meningkatkan akurasi dan robust terhadap perubahan ekonomi nasional.

5. Menerapkan analisis multi-saham (portfolio prediction) untuk mengoptimasi alokasi aset berdasarkan prediksi harga dan sentimen pasar.
6. Mengembangkan dashboard interaktif untuk menampilkan prediksi harga saham dan analisis sentimen secara real-time bagi investor dan analis pasar.

DAFTAR PUSTAKA

[1] A. Bagheffar and C. Saous, "The Impact of Investor Sentiment on Stock Returns in the Indonesian Stock Market During the Period (2001-2022): An Econometric Study."

[2] "Investor sentiment and stock prices," *Academic Journal of Business & Management*, vol. 5, no. 22, 2023, doi: 10.25236/ajbm.2023.052215.

[3] G. Liu, Y. Yang, W. Mo, W. Gu, and R. Wang, "Private Placement, Investor Sentiment, and Stock Price Anomaly," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 27, no. 5, pp. 771–779, Sep. 2023, doi: 10.20965/jaciii.2023.p0771.

[4] Z. Janková, "CRITICAL REVIEW OF TEXT MINING AND SENTIMENT ANALYSIS FOR STOCK MARKET PREDICTION," *Journal of Business Economics and Management*, vol. 24, no. 1, pp. 177–198, Jan. 2023, doi: 10.3846/jbem.2023.18805.

[5] P. Patel, "Real-Time Sentiment Analysis of Twitter Streams for Stock Forecasting," *International Journal of Computer Trends and Technology*, vol. 72, no. 5, pp. 204–209, May 2024, doi: 10.14445/22312803/ijctt-v72i5p125.

[6] M. Mokhtari, A. Seraj, N. Saeedi, and A. Karshenas, "The Impact of Twitter Sentiments on Stock Market Trends."

[7] Z. Li, "The Impact of Social Media Sentiment on Stock Price Changes," *Advances in Economics, Management and Political Sciences*, vol. 170, no. 1, pp. 49–59, Jun. 2025, doi: 10.54254/2754-1169/2025.lh23972.

[8] E. Arif, S. Suherman, and A. P. Widodo, "Predicting Stock Prices of Digital Banks: A Machine Learning Approach Combining Historical Data and Social Media Sentiment from X," *Ingenierie des Systemes d'Information*, vol. 30, no. 3, pp. 687–701, Mar. 2025, doi: 10.18280/isi.300313.

[9] P. W. Rahayu *et al.*, *Buku Ajar Data Mining*. PT. Sonpedia Publishing Indonesia, 2024.

[10] G. H. F. N. S. D. B. I. T. A. V. Y. P. R. Y. I. I. A. L. S. C. A. C. R. A. I. R. S. M. R. S. R. Maulani, *Machine Learning*. CV. Mega Press Nusantara, 2025.

[11] N. P. I. Maharani, Y. Yustiawan, F. C. Rochim, and A. Purwarianti, "Domain-Specific Language Model Post-Training for Indonesian Financial NLP," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.09736

[12] H. Zolfagharinia, M. Najafi, S. Rizvi, and A. Haghighi, “Unleashing the Power of Tweets and News in Stock-Price Prediction Using Machine-Learning Techniques,” *Algorithms*, vol. 17, no. 6, Jun. 2024, doi: 10.3390/a17060234.

[13] P. Koukaras, C. Nousi, and C. Tjortjis, “Stock Market Prediction Using Microblogging Sentiment Analysis and Machine Learning,” *Telecom*, vol. 3, no. 2, pp. 358–378, Jun. 2022, doi: 10.3390/telecom3020019.

[14] H. Zolfagharinia, M. Najafi, S. Rizvi, and A. Haghighi, “Unleashing the Power of Tweets and News in Stock-Price Prediction Using Machine-Learning Techniques,” *Algorithms*, vol. 17, no. 6, Jun. 2024, doi: 10.3390/a17060234.

[15] P. Koukaras, C. Nousi, and C. Tjortjis, “Stock Market Prediction Using Microblogging Sentiment Analysis and Machine Learning,” *Telecom*, vol. 3, no. 2, pp. 358–378, Jun. 2022, doi: 10.3390/telecom3020019.

[16] Y. Luan, H. Zhang, C. Zhang, Y. Mu, and W. Wang, “Stock Price Prediction with Sentiment Analysis for Chinese Market,” 2024. [Online]. Available: http://www.data.csmar.com

[17] P. Koukaras, C. Nousi, and C. Tjortjis, “Stock Market Prediction Using Microblogging Sentiment Analysis and Machine Learning,” *Telecom*, vol. 3, no. 2, pp. 358–378, Jun. 2022, doi: 10.3390/telecom3020019.

[18] C. Schröer, F. Kruse, and J. M. Gómez, “A systematic literature review on applying CRISP-DM process model,” in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 526–534. doi: 10.1016/j.procs.2021.01.199.

LAMPIRAN

1. Lampiran A – Dataset dan Informasi Terkait

A. **Lampiran A1 – Deskripsi Dataset**

* Sumber Data: sentimen x/twitter dan historis website (investing.com)

* Jumlah Data: sentimen (2232) dan historis (37/day)

* Jumlah Atribut: sentimen (15) dan historis (7)

* Deskripsi Atribut Sentimen:

Kolom	Deskripsi
conversation_id_str	ID percakapan Twitter. Berguna untuk mengelompokkan tweet utama dengan reply atau quote terkait.
created_at	Tanggal dan waktu tweet dibuat. Penting untuk sinkronisasi dengan data harga saham pada waktu yang sama.
favorite_count	Jumlah likes pada tweet tersebut. Mengindikasikan seberapa populer atau menarik tweet tersebut bagi pengguna lain.
full_text	Isi lengkap tweet. Di sinilah analisis sentimen dilakukan (positif, negatif, netral).
id_str	ID unik tweet. Digunakan untuk keperluan pengambilan data lebih lanjut atau sebagai primary key.
image_url	Link gambar yang diunggah dalam tweet (jika ada). Bisa berguna jika gambar berisi informasi analisis teknikal atau laporan saham.
in_reply_to_screen_name	Jika tweet ini merupakan reply, akan menunjukkan username yang direply. Penting untuk menganalisis struktur percakapan.
lang	Bahasa tweet (misal: 'id' untuk Indonesia, 'en' untuk Inggris). Hanya tweet dalam bahasa tertentu yang dianalisis.
location	Lokasi pengguna yang menulis tweet. Berguna untuk segmentasi regional sentimen pasar.
quote_count	Jumlah quote tweet dari tweet tersebut. Menunjukkan tingkat interaksi lanjutan.

reply_count	Jumlah balasan pada tweet. Menggambarkan engagement dan diskusi yang muncul.
retweet_count	Jumlah retweet. Indikasi lain dari popularitas atau viralnya tweet tersebut.
tweet_url	Link langsung ke tweet tersebut. Memudahkan tracing sumber tweet untuk validasi.
user_id_str	ID unik pengguna. Untuk tracking pengguna tertentu dalam analisis historical.
username	Username Twitter pengguna. Biasanya digunakan untuk identifikasi publikasi hasil analisis.

* Deskripsi Atribut Historis:

Kolom	**Penjelasan**
Tanggal	Tanggal perdagangan saham. Kunci waktu untuk semua data.
Terakhir	Harga penutupan saham pada hari tersebut. Sering dipakai untuk analisis teknikal.
Pembukaan	Harga pembukaan saham saat pasar dibuka.
Tertinggi	Harga tertinggi saham pada hari itu.
Terendah	Harga terendah saham pada hari itu.
Vol	Volume transaksi (jumlah lot atau lembar saham yang ditransaksikan). Mengukur likuiditas dan minat pasar.
Perubahan%	Persentase perubahan harga penutupan dibandingkan hari sebelumnya. Indikasi tren naik/turun harian.

B. **Lampiran A2 – Contoh Dataset Mentah (Raw)**

* Contoh Dataset Sentimen:

	conversation_id_str	created_at	favorite_count	full_text
2	1941088397638152360	Fri Jul 04 14:30:19 +0000 2025	0	@PrepaTAP JP Morgan jual BBCA & BMRI tapi beli BBRI doang. Artinya? Percaya!
3	1941088397638152360	Fri Jul 04 14:30:01 +0000 2025	0	@PrepaTAP Koreksi BBRI cuma diskon. Siap-siap mantul!
4	1941088397638152360	Fri Jul 04 14:29:56 +0000 2025	0	@PrepaTAP Target 4700 makin kebuka nih! Gas pol BBRI!
5	1941088397638152360	Fri Jul 04 14:29:46 +0000 2025	0	@PrepaTAP Fix BBRI makin solid kepercayaan global balik lagi.
6	1941088397638152360	Fri Jul 04 14:28:48 +0000 2025	0	@PrepaTAP Hold BBRI = tidur nyenyak bangun bangun cuan.
7	1941088397638152360	Fri Jul 04 14:28:30 +0000 2025	0	@PrepaTAP JP Morgan balik nambah BBRI sinyal cuan makin tebal!
8	1941133667511967770	Fri Jul 04 13:55:05 +0000 2025	0	Prospek saham BBRI terang benderang di tengah awan global. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/l29cvt5vrw
9	1941133638575534416	Fri Jul 04 13:54:58 +0000 2025	0	Saat geopolitik tak pasti BBRI tetap stabil dan atraktif. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/yJATVD4inw
10	1941132988236124629	Fri Jul 04 13:52:23 +0000 2025	0	Saham BBRI tetap diminati karena prospeknya terukur dan stabil. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/gKtOGFNok2
11	1941132859827450254	Fri Jul 04 13:51:52 +0000 2025	0	BBRI bukan hanya saham tapi simbol kepercayaan jangka panjang. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/L1a6uWMgLE
12	1941132568902209690	Fri Jul 04 13:50:43 +0000 2025	0	Investor besar tahu: BBRI punya arah dan daya tahan jangka panjang. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/V7a3dGhnBY
13	1941132418771218704	Fri Jul 04 13:50:07 +0000 2025	0	bbri kok kaya tai
14	1941130700750803274	Fri Jul 04 13:43:18 +0000 2025	0	Saham BBRI jadi indikator kekuatan perbankan Indonesia. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/bt8lOU9MDa
15	1941130062390309354	Fri Jul 04 13:40:45 +0000 2025	0	Saham BBRI tetap bersinar meski pasar global bergejolak. #SahamBBRI #TransformasiBRI #qrisbri https://t.co/Or1lvSQ6xj

* Contoh Dataset Historis:

	Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah	Vol.	Perubahan%
2	16/06/2025	3.990	3.980	4.010	3.960	155,45M	-0,25%
3	13/06/2025	4.000	4.020	4.050	3.980	283,83M	-1,72%
4	12/06/2025	4.070	4.070	4.100	4.060	113,85M	0,00%
5	11/06/2025	4.070	4.130	4.140	4.060	196,56M	-1,93%
6	10/06/2025	4.150	4.100	4.160	4.100	179,83M	1,22%
7	05/06/2025	4.100	4.110	4.140	4.050	230,13M	0,49%
8	04/06/2025	4.080	4.230	4.230	4.080	279,76M	-2,86%
9	03/06/2025	4.200	4.210	4.230	4.120	277,90M	0,00%
10	02/06/2025	4.200	4.360	4.390	4.200	389,54M	-5,62%
11	28/05/2025	4.450	4.360	4.450	4.320	466,13M	1,83%
12	27/05/2025	4.370	4.320	4.370	4.280	180,03M	1,16%
13	26/05/2025	4.320	4.350	4.350	4.260	149,42M	-0,69%
14	23/05/2025	4.350	4.370	4.370	4.330	187,46M	1,16%
15	22/05/2025	4.300	4.280	4.310	4.240	208,29M	0,94%

2. Lampiran B – Proses Preprocessing

A. **Lampiran B1 – Data Cleaning**

Langkah-langkah pembersihan:

* Penanganan nilai kosong: Di Drop atau dihapus

* Duplikasi data: Dihapus

* Outlier: Tidak ada

B. **Lampiran B2 – Transformasi Data**

Jenis transformasi:

* Normalisasi/Standarisasi: Digunakan

* Encoding: Digunakan

* Binning/Discretization: Tidak digunakan

3. Lampiran C – Eksplorasi Data & Visualisasi (EDA)

A. **Lampiran C1 – Statistik Deskriptif**

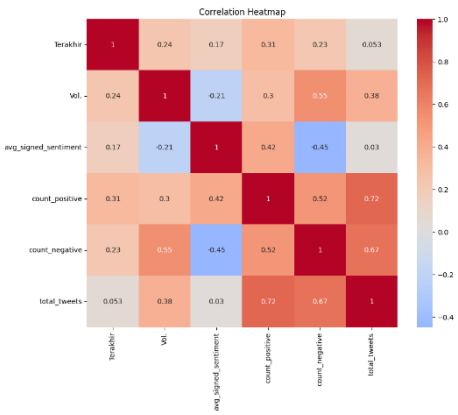
Tampilkan tabel statistik: mean, median, modus, min, max, std

	Tanggal	Terakhir	Pembukaan	Tertinggi	Terendah	avg_signed_sentiment	count_positive	count_negative	count_neutral	total_tweets
count	39	39.000000	39.000000	39.000000	39.000000	38.000000	38.000000	38.000000	38.000000	38.000000
mean	2025-06-03 19:41:32.307692288	4.004359	4.018205	4.055385	3.961795	-0.021298	2.842105	3.105263	15.026316	20.973684
min	2025-05-02 00:00:00	3.670000	3.650000	3.700000	3.640000	-0.374837	0.000000	0.000000	3.000000	5.000000
25%	2025-05-19 12:00:00	3.795000	3.835000	3.870000	3.765000	-0.125760	1.000000	1.000000	10.000000	14.000000
50%	2025-06-04 00:00:00	3.990000	4.000000	4.030000	3.960000	-0.012669	2.000000	2.500000	12.500000	17.000000
75%	2025-06-19 12:00:00	4.205000	4.220000	4.270000	4.145000	0.047222	4.000000	4.000000	17.750000	25.750000
max	2025-07-04 00:00:00	4.450000	4.370000	4.450000	4.330000	0.331064	18.000000	12.000000	54.000000	70.000000
std	NaN	0.228874	0.219795	0.223923	0.214450	0.146523	3.405293	2.993592	9.936248	13.835885

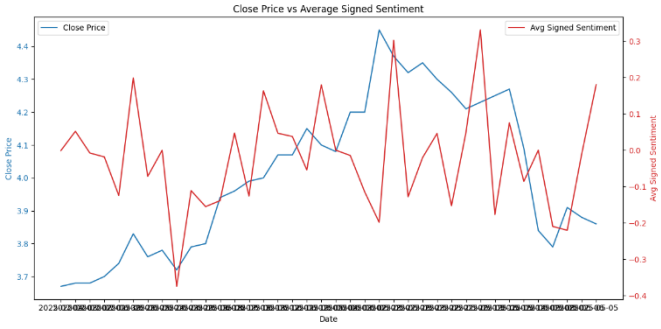
B. **Lampiran C2 – Grafik dan Visualisasi**

(Tambahkan visualisasi EDA: histogram, *boxplot*, *scatterplot*, *heatmap*)

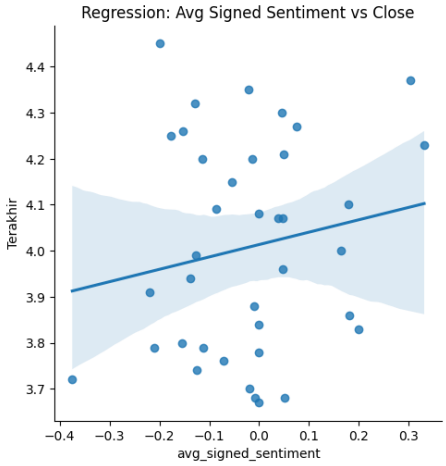
heatmap



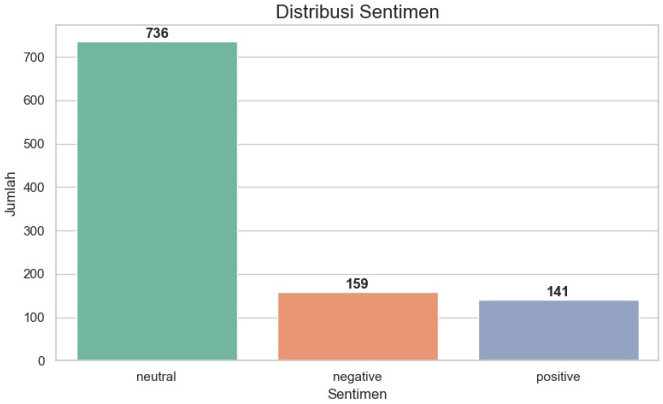
Line Plot



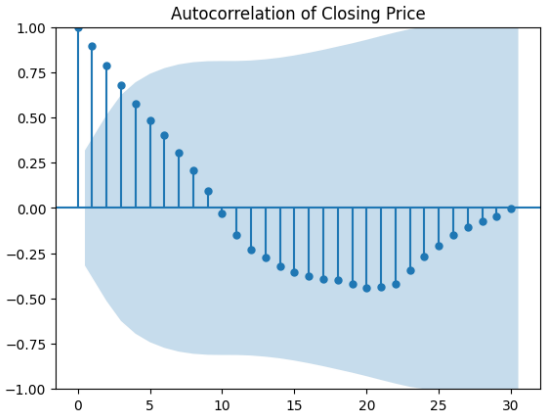
Autocorrelation of Closing Price



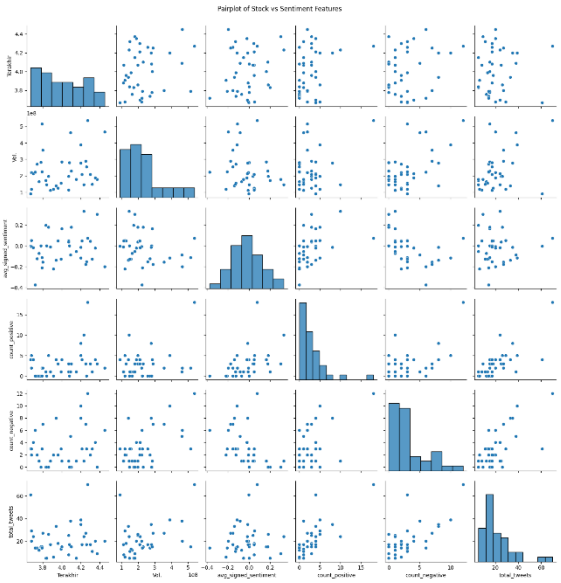
Pairplot of Stock vs Sentiment Features



Regression Plot



Bar Plot



4. Lampiran D – Pemodelan dan Evaluasi

A. **Lampiran D1 – Rincian Model**

* Model yang digunakan:

1. Support Vector Regression (SVR)
2. Multilayer Perceptron (MLP)
3. Logistic Regression
4. Extreme Gradient Boosting (XGBoost)
5. Random Forest

* Parameter model:

1. Historis: Terakhir, Pembukaan, Tertinggi, Terendah, Vol., Perubahan%
2. Gabungan (historis+sentimen): Terakhir, Pembukaan, Tertinggi, Terendah, Vol., Perubahan%, avg_signed_sentiment, count_positive, count_negative, count_neutral, total_tweets, range, day_return, sentiment_ratio, tweet_intensity, lag_1, lag_2

B. **Lampiran D2 – Hasil Evaluasi Model**

* Confusion Matrix

* Classification Report (Accuracy, Precision, Recall, F1)

* ROC Curve / AUC / RMSE (jika regresi)

5. Lampiran E – Kode Program

A. **Lampiran E1 – *Script* Python/R/Notebook**

```
import numpy as np
import pandas as pd
import re

import seaborn as sns
import matplotlib.pyplot as plt
import mlflow
from mlflow.models.signature import infer_signature

from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve, confusion_
from yellowbrick.classifier import ROCAUC

import warnings
warnings.filterwarnings("ignore")
```

Lampiran E1 1 Import Library

Potongan kode di atas merupakan bagian dari persiapan awal dalam proyek machine learning berbasis Python. Baris pertama mengimpor beberapa library penting untuk manipulasi data dan analisis numerik seperti numpy, pandas, serta modul regular expression re. Kemudian, untuk keperluan visualisasi, digunakan seaborn dan matplotlib.pyplot, sedangkan mlflow diimpor untuk mendukung pelacakan eksperimen dan pengelolaan model secara otomatis, termasuk fungsi infer_signature untuk menangkap input-output model.

Selanjutnya, modul scikit-learn digunakan secara luas untuk keperluan pemodelan, mulai dari pemisahan data dengan TimeSeriesSplit dan pencarian hyperparameter menggunakan GridSearchCV, hingga pembentukan pipeline dan preprocessing standar menggunakan Pipeline dan StandardScaler. Untuk klasifikasi, digunakan berbagai algoritma seperti RandomForestClassifier, LogisticRegression, MLPClassifier (neural network), dan SVC (Support Vector Machine). Juga terdapat XGBClassifier dari pustaka xgboost, yang dikenal untuk performa tinggi dalam berbagai kompetisi machine learning.

Untuk evaluasi model, berbagai metrik diekstrak dari sklearn.metrics seperti accuracy_score, precision_score, recall_score, f1_score, dan ROC-related metrics, serta digunakan juga ROCAUC dari yellowbrick untuk visualisasi performa model. Di bagian akhir, modul warnings diimpor dan diarahkan untuk mengabaikan peringatan sistem guna menjaga agar output tidak terganggu oleh notifikasi yang tidak penting selama proses eksekusi notebook atau script.

```
df_default = pd.read_csv('../data/processed/dataset_model.csv')
df_default
```

Lampiran E1 2 Membaca Dataset

Baris kode di atas digunakan untuk membaca file dataset yang telah diproses sebelumnya dan disimpan dalam format CSV. File tersebut berada dalam folder ../data/processed/ dengan nama dataset_model.csv. Fungsi pd.read_csv() dari pustaka pandas digunakan untuk memuat data ini ke dalam sebuah DataFrame bernama df_default, yang kemudian dapat digunakan untuk eksplorasi, preprocessing, maupun pelatihan model machine learning. Pemanggilan df_default di akhir baris akan menampilkan isi data tersebut dalam notebook.

```
df_default.info()
```

Lampiran E1 3 Informasi Struktur Dataset

Baris kode df_default.info() digunakan untuk menampilkan ringkasan struktur DataFrame df_default, termasuk jumlah total baris, jumlah kolom, nama masing-masing kolom, jumlah nilai non-null pada setiap kolom, tipe data setiap kolom, serta penggunaan memori secara

keseluruhan. Fungsi ini sangat berguna untuk memahami kondisi awal dataset, mendeteksi adanya nilai kosong (missing values), serta memastikan bahwa tipe data yang digunakan sesuai dengan kebutuhan pemodelan machine learning.

```
df_default.dropna(inplace=True)
```

Lampiran E1 4 Menghapus Nilai Kosong dari Dataset

Baris kode df_default.dropna(inplace=True) digunakan untuk menghapus seluruh baris dalam DataFrame df_default yang mengandung setidaknya satu nilai kosong (NaN). Argumen inplace=True memastikan bahwa perubahan dilakukan langsung pada DataFrame tanpa perlu membuat salinan baru. Langkah ini penting untuk membersihkan data sebelum digunakan dalam proses pelatihan model, karena keberadaan nilai kosong dapat mengganggu kinerja algoritma machine learning.

```
df_default.drop(columns=['date'])
```

Lampiran E1 5 Menghapus Kolom Tanggal dari Dataset

Baris kode df_default.drop(columns=['date']) digunakan untuk menghapus kolom date dari DataFrame df_default. Perintah ini menghasilkan salinan baru dari DataFrame tanpa kolom tersebut, namun karena tidak menggunakan inplace=True, DataFrame aslinya (df_default) tidak akan berubah kecuali hasilnya disimpan kembali. Penghapusan kolom ini umumnya dilakukan jika informasi tanggal dianggap tidak relevan atau tidak diperlukan dalam proses pemodelan.

```
df_default['Tanggal'] = pd.to_datetime(df_default['Tanggal'])
df_default.sort_values(by='Tanggal', inplace=True)
```

Lampiran E1 6 Konversi dan Pengurutan Kolom Tanggal

Dua baris kode di atas digunakan untuk memastikan bahwa kolom `Tanggal` pada DataFrame `df_default` diperlakukan sebagai tipe data datetime dan kemudian mengurutkan seluruh DataFrame berdasarkan urutan tanggal tersebut. Baris pertama mengonversi nilai dalam kolom `Tanggal` menjadi format datetime menggunakan `pd.to_datetime()`, yang penting untuk analisis deret waktu atau pemrosesan berdasarkan kronologi. Baris kedua mengurutkan baris-baris data berdasarkan nilai tanggal secara menaik (dari yang paling lama ke yang terbaru), dan perubahan ini dilakukan langsung pada DataFrame karena menggunakan `inplace=True`.

```python
def parse_volume(vol_str):
    if isinstance(vol_str, str):
        vol_str = vol_str.replace(',', '.')
        if vol_str.endswith('M'):
            return float(vol_str[:-1]) * 1_000_000
        elif vol_str.endswith('K'):
            return float(vol_str[:-1]) * 1_000
        else:
            return float(vol_str)
    return vol_str

df_default['Vol.'] = df_default['Vol.'].apply(parse_volume)
```

Lampiran E1 7 Parsing dan Konversi Kolom Volume

Potongan kode di atas bertujuan untuk membersihkan dan mengubah format data pada kolom Vol. menjadi nilai numerik dalam satuan yang konsisten. Fungsi parse_volume didefinisikan untuk menangani nilai volume yang awalnya berbentuk string dengan satuan ribuan (K) atau jutaan (M), serta tanda koma sebagai pemisah desimal. Jika nilai berakhiran M, maka angka tersebut dikalikan satu juta, sedangkan jika berakhiran K, dikalikan seribu. Jika tidak memiliki akhiran, nilai langsung dikonversi menjadi float. Fungsi ini kemudian diterapkan ke seluruh elemen kolom Vol. menggunakan apply, sehingga setiap entri dalam kolom tersebut diubah menjadi angka desimal yang seragam dan siap digunakan dalam analisis kuantitatif.

```python
df_default['Perubahan%'] = df_default['Perubahan%'].str.replace('%', '', regex=False)
df_default['Perubahan%'] = df_default['Perubahan%'].str.replace(',', '.', regex=False).astype(float)
```

Lampiran E1 8 Pembersihan dan Konversi Kolom Persentase Perubahan

Dua baris kode di atas digunakan untuk membersihkan dan mengubah format data pada kolom Perubahan% dari format string menjadi tipe numerik float. Baris pertama menghapus karakter persen (%) dari setiap nilai dalam kolom tersebut menggunakan str.replace. Baris kedua mengganti tanda koma menjadi titik sebagai pemisah desimal agar sesuai dengan format numerik standar dalam Python, lalu seluruh kolom dikonversi ke tipe data float. Transformasi ini penting agar nilai persentase dapat digunakan dalam perhitungan matematis dan analisis statistik lebih lanjut.

```python
df_default
```

Lampiran E1 9 Menampilkan Isi Dataset

Baris kode df_default digunakan untuk menampilkan isi DataFrame setelah dilakukan proses pembersihan dan transformasi data sebelumnya. Dengan menuliskan nama variabel DataFrame tanpa fungsi tambahan, notebook akan secara otomatis menampilkan tabel berisi seluruh baris

dan kolom yang tersimpan dalam df_default, termasuk kolom-kolom yang sudah dikonversi seperti Tanggal, Vol., dan Perubahan%. Langkah ini biasanya dilakukan untuk memverifikasi bahwa data sudah dalam format yang sesuai sebelum digunakan dalam analisis atau pemodelan.

```
# Fitur tambahan
df_default['range'] = df_default['Tertinggi'] - df_default['Terendah']
df_default['day_return'] = df_default['Terakhir'].pct_change()
df_default['sentiment_ratio'] = df_default['count_positive'] / (df_default['count_negative'] + 1)
df_default['tweet_intensity'] = df_default['total_tweets'] / (df_default['Vol.'] + 1)

# Lag features (harga hari sebelumnya)
df_default['lag_1'] = df_default['Terakhir'].shift(1)
df_default['lag_2'] = df_default['Terakhir'].shift(2)

# Target: apakah harga besok lebih tinggi dari hari ini?
df_default['target'] = (df_default['Terakhir'].shift(-1) > df_default['Terakhir']).astype(int)

# Tambahkan kolom deskripsi target
df_default['keterangan_target'] = df_default['target'].map({1: 'Naik', 0: 'Turun/Stagnan'})
```

Lampiran E1 10 Penambahan Fitur dan Target pada Dataset

Potongan kode di atas menambahkan sejumlah fitur baru ke dalam DataFrame df_default guna memperkaya informasi yang dapat digunakan dalam pemodelan prediktif. Kolom range dihitung sebagai selisih antara harga tertinggi dan terendah dalam satu hari, yang mencerminkan volatilitas harian. Kolom day_return menunjukkan persentase perubahan harga penutupan dari hari sebelumnya menggunakan fungsi pct_change(). Kolom sentiment_ratio menggambarkan rasio antara jumlah tweet positif terhadap tweet negatif (ditambah satu untuk menghindari pembagian dengan nol), sedangkan tweet_intensity mengukur intensitas percakapan di media sosial relatif terhadap volume perdagangan.

Fitur lag lag_1 dan lag_2 menyimpan harga penutupan satu dan dua hari sebelumnya, masing-masing menggunakan fungsi shift(), yang berguna untuk mengenali pola historis. Selanjutnya, target prediksi didefinisikan dalam kolom target, di mana nilai 1 menunjukkan bahwa harga keesokan harinya lebih tinggi dari hari ini, dan 0 sebaliknya. Terakhir, kolom keterangan_target ditambahkan untuk memberikan deskripsi kategorikal terhadap nilai target, yaitu 'Naik' untuk 1 dan 'Turun/Stagnan' untuk 0, sehingga memudahkan dalam interpretasi hasil klasifikasi.

```
# Drop baris NaN
df_default = df_default.dropna()
```

Lampiran E1 11 Menghapus Baris dengan Nilai Kosong

Baris kode df_default = df_default.dropna() digunakan untuk menghapus seluruh baris dalam DataFrame df_default yang masih mengandung nilai kosong (NaN) setelah proses pembuatan fitur baru dan target. Tidak seperti sebelumnya yang menggunakan inplace=True, di sini hasil penghapusan disimpan kembali ke variabel df_default, memastikan bahwa DataFrame yang

digunakan selanjutnya sudah bersih sepenuhnya dan siap digunakan untuk proses analisis atau pelatihan model machine learning tanpa gangguan akibat data yang hilang.

```
df_all = df_default.copy()
```

Lampiran E1 12 Menyalin Dataset ke Variabel Baru

Baris kode df_all = df_default.copy() digunakan untuk membuat salinan independen dari DataFrame df_default dan menyimpannya dalam variabel baru bernama df_all. Salinan ini bersifat mendalam (deep copy), artinya perubahan yang dilakukan pada df_all tidak akan memengaruhi df_default, begitu pula sebaliknya. Langkah ini biasanya dilakukan untuk menjaga versi asli dari data yang telah dibersihkan, sambil memungkinkan eksplorasi atau pemrosesan lebih lanjut pada salinannya tanpa risiko kehilangan data awal.

```
y = df_all['target']
```

Lampiran E1 13 Mendefinisikan Variabel Target

Baris kode y = df_all['target'] digunakan untuk mengekstrak kolom target dari DataFrame df_all dan menyimpannya ke dalam variabel y. Variabel ini berfungsi sebagai label atau target yang akan diprediksi oleh model machine learning, yaitu apakah harga saham pada hari berikutnya naik atau tidak. Nilai 1 menandakan bahwa harga naik, sedangkan 0 menunjukkan harga turun atau stagnan. Variabel y ini akan digunakan dalam proses pelatihan dan evaluasi model klasifikasi.

```python
# 🔷 Model dictionary
models = {
    "RandomForest": RandomForestClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(eval_metric='logloss'),
    "LogReg": LogisticRegression(max_iter=1000),
    "SVC": SVC(probability=True),  # Untuk ROC AUC perlu probabilitas
    "MLP": MLPClassifier(max_iter=1000, random_state=42)
}

# 🔷 Directional accuracy
def directional_accuracy(y_true, y_pred):
    y_true = pd.Series(y_true).reset_index(drop=True)
    y_pred = pd.Series(y_pred).reset_index(drop=True)
    return np.mean(np.sign(y_true.diff().fillna(0)) == np.sign(y_pred.diff().fillna(0)))

# 🔷 Evaluasi model
def evaluate_model(model_or_pipeline, X, y, model_name="Model"):
    tscv = TimeSeriesSplit(n_splits=5)
    metrics = {'accuracy': [], 'precision': [], 'recall': [], 'f1': [], 'roc_auc': [], 'directional_acc': []}
    all_conf_matrices = []

    fold = 1
    for train_idx, test_idx in tscv.split(X):
        X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
        y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

        # Buat pipeline
        if isinstance(model_or_pipeline, Pipeline):
            pipeline = model_or_pipeline
        else:
            pipeline = Pipeline([
                ('scaler', StandardScaler()),
                ('model', model_or_pipeline)
            ])

        pipeline.fit(X_train, y_train)
        y_pred = pipeline.predict(X_test)
```

Lampiran E1 14 Pendefinisian Model, Fungsi Akurasi Arah, dan Evaluasi Model

Potongan kode di atas memuat tiga bagian utama yang berperan penting dalam proses evaluasi performa model klasifikasi berbasis data deret waktu. Pertama, sebuah dictionary bernama models didefinisikan untuk menyimpan lima jenis model machine learning yang akan digunakan: RandomForestClassifier, XGBClassifier, LogisticRegression, SVC (Support Vector Classifier), dan MLPClassifier (Multilayer Perceptron). Masing-masing model disiapkan dengan parameter dasar, seperti jumlah estimator, metrik evaluasi, batas iterasi, atau opsi probabilitas yang diperlukan untuk pengukuran ROC AUC.

Selanjutnya, fungsi directional_accuracy dibuat untuk mengukur seberapa baik model dalam memprediksi arah perubahan target dari hari ke hari, bukan hanya klasifikasi akhir. Fungsi ini membandingkan arah perubahan nilai aktual dan prediksi menggunakan np.sign() terhadap diff() dari masing-masing nilai, lalu menghitung proporsi kesamaan arah.

Terakhir, fungsi evaluate_model digunakan untuk mengevaluasi performa setiap model melalui validasi silang berbasis waktu (TimeSeriesSplit) sebanyak lima lipatan. Dalam setiap iterasi, data dilatih dan diuji menggunakan pipeline yang berisi tahap normalisasi (StandardScaler) dan model klasifikasi. Model kemudian dilatih menggunakan data latih dan menghasilkan prediksi pada data uji (y_pred). Fungsi ini nantinya juga mengumpulkan berbagai metrik evaluasi seperti

akurasi, presisi, recall, F1-score, ROC AUC, serta directional accuracy untuk masing-masing fold.

```
        # Prediksi probabilitas
        if hasattr(pipeline.named_steps['model'], "predict_proba"):
            y_proba = pipeline.predict_proba(X_test)[:, 1]
        elif hasattr(pipeline.named_steps['model'], "decision_function"):
            y_proba = pipeline.decision_function(X_test)
        else:
            y_proba = None

        if len(np.unique(y_test)) > 1:
            acc = accuracy_score(y_test, y_pred)
            prec = precision_score(y_test, y_pred, zero_division=0)
            rec = recall_score(y_test, y_pred, zero_division=0)
            f1 = f1_score(y_test, y_pred, zero_division=0)
            cm = confusion_matrix(y_test, y_pred)
            da = directional_accuracy(y_test, y_pred)

            # ROC AUC hanya jika valid
            if y_proba is not None and len(np.unique(y_test)) > 1:
                try:
                    roc_auc = roc_auc_score(y_test, y_proba)
                except:
                    roc_auc = 0
            else:
                roc_auc = 0

            metrics['accuracy'].append(acc)
            metrics['precision'].append(prec)
            metrics['recall'].append(rec)
            metrics['f1'].append(f1)
            metrics['roc_auc'].append(roc_auc)
            metrics['directional_acc'].append(da)
            all_conf_matrices.append(cm)

            print(f"\n📊 Fold {fold} Confusion Matrix:")
            print(classification_report(y_test, y_pred, zero_division=0))
```

Lampiran E1 15 Prediksi Probabilitas dan Evaluasi Metrik Klasifikasi

Bagian kode di atas melanjutkan proses evaluasi model dengan melakukan prediksi probabilitas serta menghitung berbagai metrik performa untuk setiap fold dalam validasi silang. Langkah pertama memeriksa apakah model dalam pipeline mendukung metode predict_proba, yang digunakan untuk menghasilkan probabilitas kelas, atau decision_function, yang menghasilkan skor keputusan. Jika salah satu tersedia, nilai probabilitas untuk kelas positif ([:, 1]) atau skor keputusan disimpan dalam variabel y_proba. Jika tidak tersedia, maka y_proba diatur ke None.

Evaluasi hanya dilanjutkan jika label target pada data uji memiliki lebih dari satu kelas, untuk menghindari error pada perhitungan metrik klasifikasi. Dalam kondisi tersebut, nilai akurasi, presisi, recall, dan F1-score dihitung menggunakan fungsi dari sklearn.metrics. Selain itu, confusion matrix juga dihitung dan disimpan, sedangkan directional_accuracy dihitung untuk mengukur ketepatan arah prediksi.

Jika prediksi probabilitas tersedia dan label target valid, nilai ROC AUC dihitung menggunakan roc_auc_score. Namun, jika tidak tersedia atau terjadi kesalahan, skor ROC AUC akan diset menjadi nol. Semua metrik yang dihitung kemudian ditambahkan ke dalam dictionary metrics, dan confusion matrix untuk setiap fold ditampilkan melalui print, disertai laporan klasifikasi lengkap menggunakan classification_report. Seluruh proses ini memungkinkan penilaian performa model secara menyeluruh dan konsisten untuk tiap fold validasi.

```
        else:
            print(f"\n⚠️ Fold {fold}: Only one class present in y_test, skipping metrics.")

        fold += 1

    # Rata-rata dan deviasi
    mean_metrics = {f"{k}_mean": np.mean(v) if v else 0 for k, v in metrics.items()}
    std_metrics = {f"{k}_std": np.std(v) if v else 0 for k, v in metrics.items()}

    # Logging ke MLflow
    with mlflow.start_run(run_name=model_name):
        for k, v in mean_metrics.items():
            mlflow.log_metric(k, v)
        for k, v in std_metrics.items():
            mlflow.log_metric(k, v)

        input_example = X_test.head(1)
        signature = infer_signature(X_test, y_pred)
        mlflow.sklearn.log_model(pipeline, f"{model_name}_model", input_example=input_example, signature=signature)

    return {**mean_metrics, **std_metrics}
```

Lampiran E1 16 Penanganan Kelas Tunggal, Perhitungan Rata-Rata Metrik, dan Logging ke MLflow

Bagian akhir dari fungsi evaluate_model ini menyelesaikan proses evaluasi dengan menangani kasus khusus dan mencatat hasil ke MLflow. Jika data uji (y_test) dalam suatu fold hanya mengandung satu kelas, maka bagian else akan mencetak peringatan bahwa metrik tidak dihitung untuk fold tersebut, karena metrik klasifikasi seperti precision atau recall tidak relevan tanpa variasi kelas.

Setelah seluruh fold selesai dievaluasi, rata-rata (mean) dan standar deviasi (std) dari setiap metrik dikalkulasi menggunakan dictionary comprehension. Ini bertujuan untuk merangkum performa model secara keseluruhan dan melihat seberapa konsisten model pada setiap fold.

Selanjutnya, proses logging ke MLflow dimulai dalam konteks with mlflow.start_run, dengan setiap metrik rata-rata dan standar deviasi dicatat ke sistem pelacakan eksperimen menggunakan mlflow.log_metric. Model yang sudah dilatih juga disimpan menggunakan mlflow.sklearn.log_model, lengkap dengan input_example sebagai sampel input dan signature untuk mendokumentasikan struktur input dan output model.

Akhirnya, fungsi ini mengembalikan dictionary gabungan dari semua metrik rata-rata dan deviasi, yang dapat digunakan untuk analisis lebih lanjut atau visualisasi performa model secara sistematis.

```
# Tahap 1 - Evaluasi data historis
features_hist = ['Terakhir', 'Pembukaan', 'Tertinggi', 'Terendah', 'Vol.', 'Perubahan%', 'range', 'day_return', 'lag_1',
X_hist = df_all[features_hist]

print("🔷 Tahap 1: Data Historis")
results_hist = {}

for name, model in models.items():
    print(f"\n🔍 Evaluating {name} with historical data...")
    results_hist[name] = evaluate_model(model, X_hist, y)
```

Lampiran E1 17 Evaluasi Model Menggunakan Fitur Data Historis

Bagian kode di atas menjalankan proses evaluasi awal (Tahap 1) terhadap performa model dengan hanya menggunakan fitur-fitur historis dari data pasar. Variabel features_hist berisi daftar kolom yang merepresentasikan data harga dan volume saham seperti Terakhir, Pembukaan, Tertinggi, Terendah, serta fitur turunan seperti Vol., Perubahan%, range, day_return, dan dua fitur lag (lag_1 dan lag_2). Semua fitur tersebut diambil dari DataFrame df_all dan disimpan dalam variabel X_hist sebagai data input.

Selanjutnya, program mencetak informasi bahwa proses evaluasi pada tahap data historis sedang berlangsung. Dictionary kosong bernama results_hist disiapkan untuk menyimpan hasil evaluasi setiap model. Kemudian, melalui loop for, setiap model dalam dictionary models dievaluasi menggunakan fungsi evaluate_model dengan data input X_hist dan target y. Selama proses ini, nama model dan status evaluasi akan dicetak ke layar, dan hasil evaluasi tiap model disimpan dalam results_hist untuk analisis selanjutnya.

```
# Format output sebagai DataFrame
df_historis = pd.DataFrame(results_hist).T
print("\n📈 Rangkuman Evaluasi Model (Tahap Historis):")
df_historis
```

Lampiran E1 18 Menyusun dan Menampilkan Hasil Evaluasi Model Historis

Baris kode di atas digunakan untuk menyusun hasil evaluasi model pada tahap data historis ke dalam bentuk tabel yang lebih mudah dianalisis. Dictionary results_hist, yang sebelumnya berisi hasil evaluasi tiap model, dikonversi menjadi sebuah DataFrame bernama df_historis dengan menggunakan pd.DataFrame(results_hist).T, di mana .T digunakan untuk melakukan transpose agar model menjadi indeks baris dan metrik evaluasi menjadi kolom. Setelah itu, tabel hasil evaluasi ini ditampilkan ke layar dengan mencetak judul ringkasan dan memanggil langsung nama variabel df_historis, yang memungkinkan notebook menampilkan isi tabel tersebut, termasuk nilai rata-rata dan deviasi dari berbagai metrik seperti akurasi, precision, recall, F1-score, ROC AUC, dan directional accuracy.

```
# Tahap 2 - Evaluasi data gabungan
features_combined = [
    'Terakhir', 'Pembukaan', 'Tertinggi', 'Terendah', 'Vol.', 'Perubahan%',
    'avg_signed_sentiment', 'count_positive', 'count_negative', 'count_neutral', 'total_tweets',
    'range', 'day_return', 'sentiment_ratio', 'tweet_intensity',
    'lag_1', 'lag_2'
]
X_combined = df_all[features_combined]

print("🔷 Tahap 2: Data Gabungan")
results_combined = {}

for name, model in models.items():
    print(f"\n🔍 Evaluating {name} with combined data...")
    results_combined[name] = evaluate_model(model, X_combined, y)
```

Lampiran E1 19 Evaluasi Model Menggunakan Data Gabungan Historis dan Sentimen

Baris kode di atas digunakan untuk melakukan evaluasi performa model prediktif menggunakan kombinasi fitur data historis saham dan data sentimen publik. Pertama, didefinisikan daftar variabel features_combined yang mencakup indikator pasar saham seperti harga terakhir, pembukaan, tertinggi, terendah, volume, dan persentase perubahan, serta indikator sentimen seperti rata-rata sentimen bertanda, jumlah tweet positif, negatif, netral, total tweet, rasio sentimen, dan intensitas tweet, ditambah juga fitur teknikal seperti rentang harga harian, return harian, serta nilai tertinggal (lag) satu dan dua hari. Data input X_combined kemudian diambil dari DataFrame df_all berdasarkan fitur-fitur tersebut. Setelah itu, hasil evaluasi setiap model dalam dictionary models disimpan ke dalam results_combined melalui perulangan, dengan setiap model dievaluasi menggunakan fungsi evaluate_model terhadap data gabungan ini. Proses ini mencetak pesan ke layar agar pengguna mengetahui model mana yang sedang dievaluasi.

```
df_combined = pd.DataFrame(results_combined).T
print("\n📈 Rangkuman Evaluasi Model (Tahap Gabungan):")
df_combined
```

Lampiran E1 20 Rangkuman Evaluasi Model (Tahap Data Gabungan)

Baris kode di atas digunakan untuk menyusun hasil evaluasi dari setiap model yang telah diuji pada data gabungan menjadi sebuah tabel berformat DataFrame. Objek results_combined, yang sebelumnya berisi metrik evaluasi dari berbagai model, dikonversi menjadi DataFrame menggunakan pd.DataFrame(results_combined).T, dengan .T untuk mentranspos agar nama model menjadi indeks baris dan metrik evaluasinya menjadi kolom. Setelah itu, hasil evaluasi tersebut dicetak ke layar dengan judul yang menandakan bahwa ini adalah ringkasan performa model pada tahap evaluasi dengan data gabungan.

```python
def optimize_model(model, param_grid, X, y, n_splits=5):
    tscv = TimeSeriesSplit(n_splits=n_splits)
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('model', model)
    ])
    grid = GridSearchCV(pipeline, param_grid=param_grid, cv=tscv, scoring='f1', n_jobs=-1)
    grid.fit(X, y)
    return grid.best_estimator_, grid.best_params_
```

Lampiran E1 21 Fungsi Optimasi Model dengan Grid Search dan Time Series Split

Potongan kode di atas mendefinisikan sebuah fungsi bernama optimize_model yang bertujuan untuk melakukan pencarian parameter terbaik dari suatu model machine learning dengan pendekatan grid search dan validasi silang berbasis urutan waktu (time series cross-validation). Di dalam fungsi ini, TimeSeriesSplit digunakan untuk membagi data secara berurutan sebanyak jumlah lipatan yang ditentukan (n_splits), menjaga urutan temporal agar tidak terjadi data leakage. Kemudian, sebuah pipeline dibuat untuk melakukan normalisasi fitur menggunakan StandardScaler sebelum model dilatih. Proses pencarian parameter dilakukan menggunakan GridSearchCV dengan metrik evaluasi berupa nilai f1-score, dan pelatihan dilakukan pada data X dan y. Fungsi ini mengembalikan model terbaik hasil pencarian (best_estimator_) serta parameter optimal yang ditemukan (best_params_).

```python
print("\n🔷 Tahap 3: Optimasi Model")

optimized_models = {}
optimized_folds = [3, 5]
```

Lampiran E1 22 Optimasi Model

Baris kode di atas mencetak teks sebagai penanda bahwa proses telah memasuki tahap ketiga, yaitu optimasi model. Selanjutnya, dibuat sebuah dictionary kosong bernama optimized_models yang akan menyimpan hasil model terbaik dari proses optimasi. Lalu, daftar optimized_folds didefinisikan sebagai list yang berisi dua nilai, yaitu 3 dan 5, yang merepresentasikan jumlah lipatan (folds) untuk validasi silang berbasis waktu yang akan digunakan dalam proses Grid Search pada tahap optimasi.

```python
# Random Forest

rf_grid = {
    'model__n_estimators': [100, 200],
    'model__max_depth': [None, 10, 20],
}

optimized_models['RandomForest'] = {}  # Buat dict nested untuk tiap fold

for n_fold in optimized_folds:
    print(f"\n🔷 Optimasi Random Forest dengan {n_fold}-fold TimeSeriesSplit")
    best_rf, best_rf_params = optimize_model(
        RandomForestClassifier(random_state=42),
        rf_grid,
        X_combined, y,
        n_splits=n_fold  # pastikan fungsi optimize_model menerima ini
    )
    print(f"✅ Best Params ({n_fold} fold):", best_rf_params)

    # Simpan berdasarkan jumlah fold
    optimized_models['RandomForest'][f'{n_fold}_fold'] = {
        'model': best_rf,
        'params': best_rf_params
    }
```

Lampiran E1 23 Optimasi Model Random Forest

Baris kode di atas mendefinisikan proses optimasi model Random Forest menggunakan pendekatan validasi silang TimeSeriesSplit dengan parameter grid tertentu. Pertama, rf_grid berisi kombinasi parameter yang akan diuji, yaitu jumlah pohon (n_estimators) dan kedalaman maksimum (max_depth) model Random Forest. Kemudian, dictionary kosong optimized_models['RandomForest'] disiapkan untuk menyimpan hasil optimasi berdasarkan jumlah lipatan (fold).

Selanjutnya, untuk setiap nilai n_fold dalam optimized_folds, dilakukan pencetakan informasi bahwa proses optimasi sedang berlangsung. Fungsi optimize_model() dipanggil dengan model RandomForestClassifier, parameter grid rf_grid, data fitur X_combined, dan target y, serta jumlah lipatan validasi silang sesuai nilai n_fold. Fungsi ini mengembalikan model terbaik (best_rf) dan parameter terbaiknya (best_rf_params), yang kemudian dicetak dan disimpan dalam struktur nested dictionary sesuai jumlah fold.

```python
# XGBoost

xgb_grid = {
    'model__n_estimators': [100, 200],
    'model__max_depth': [3, 6],
    'model__learning_rate': [0.05, 0.1]
}

optimized_models['XGBoost'] = {}  # Buat dict nested untuk tiap fold

for n_fold in optimized_folds:
    print(f"\n🔷 Optimasi XGBoost dengan {n_fold}-fold TimeSeriesSplit")
    best_xgb, best_xgb_params = optimize_model(
        XGBClassifier(eval_metric='logloss'),
        xgb_grid,
        X_combined, y,
        n_splits=n_fold  # pastikan fungsi optimize_model menerima ini
    )
    print(f"✅ Best Params ({n_fold} fold):", best_xgb_params)

    # Simpan berdasarkan jumlah fold
    optimized_models['XGBoost'][f'{n_fold}_fold'] = {
        'model': best_xgb,
        'params': best_xgb_params
    }
```

Lampiran E1 24 Optimasi Model XGBoost

Baris kode di atas menjalankan proses optimasi terhadap model XGBoost menggunakan teknik validasi silang TimeSeriesSplit dengan dua pilihan jumlah fold. Parameter grid xgb_grid didefinisikan terlebih dahulu, yang mencakup jumlah estimator (n_estimators), kedalaman maksimum pohon (max_depth), dan laju pembelajaran (learning_rate). Kemudian, dictionary optimized_models['XGBoost'] disiapkan untuk menyimpan hasil optimasi berdasarkan jumlah fold.

Untuk setiap nilai n_fold dalam optimized_folds, ditampilkan informasi bahwa optimasi sedang dilakukan. Fungsi optimize_model() kemudian dipanggil dengan model XGBClassifier dan parameter evaluasi logloss, menggunakan data X_combined dan target y, serta jumlah fold yang ditentukan. Fungsi tersebut mengembalikan model terbaik (best_xgb) dan kombinasi parameter terbaiknya (best_xgb_params). Hasilnya dicetak ke layar dan disimpan dalam dictionary optimized_models berdasarkan nilai fold yang digunakan.

```
# LogReg

logreg_grid = {
    'model__C': [0.01, 0.1, 1, 10],
    'model__penalty': ['l2'],
    'model__solver': ['lbfgs']
}

optimized_models['LogReg'] = {}  # Buat dict nested untuk tiap fold

for n_fold in optimized_folds:
    print(f"\n🔷 Optimasi LogReg dengan {n_fold}-fold TimeSeriesSplit")
    best_logreg, best_logreg_params = optimize_model(
        LogisticRegression(max_iter=1000),
        logreg_grid,
        X_combined, y,
        n_splits=n_fold  # pastikan fungsi optimize_model menerima ini
    )
    print(f"✅ Best Params ({n_fold} fold):", best_logreg_params)

    # Simpan berdasarkan jumlah fold
    optimized_models['LogReg'][f'{n_fold}_fold'] = {
        'model': best_logreg,
        'params': best_logreg_params
    }
```

Lampiran E1 25 Optimasi Model Logistic Regression

Baris kode di atas digunakan untuk melakukan proses optimasi model Logistic Regression menggunakan metode validasi silang TimeSeriesSplit dengan jumlah lipatan (fold) sebanyak 3 dan 5. Grid parameter logreg_grid berisi nilai-nilai hyperparameter yang akan diuji, yaitu C sebagai parameter regularisasi, penalty dengan pilihan 'l2', dan solver menggunakan 'lbfgs'. Dictionary optimized_models['LogReg'] dibuat untuk menyimpan model dan parameter terbaik dari hasil optimasi berdasarkan jumlah fold yang digunakan.

Dalam setiap iterasi untuk nilai n_fold yang telah ditentukan, sistem akan mencetak informasi bahwa proses optimasi sedang berlangsung. Fungsi optimize_model() dipanggil dengan model LogisticRegression, grid parameter, data prediktor X_combined, dan target y, serta jumlah lipatan yang sesuai. Fungsi ini akan mengembalikan model terbaik (best_logreg) beserta kombinasi parameter terbaiknya (best_logreg_params), yang kemudian dicetak dan disimpan ke dalam struktur dictionary optimized_models berdasarkan jumlah fold.

```
# SVC

svc_grid = {
    'model__C': [0.1, 1, 10],
    'model__kernel': ['linear', 'rbf'],
    'model__gamma': ['scale', 'auto']
}

optimized_models['SVC'] = {}  # Buat dict nested untuk tiap fold

for n_fold in optimized_folds:
    print(f"\n🔷 Optimasi SVC dengan {n_fold}-fold TimeSeriesSplit")
    best_svc, best_svc_params = optimize_model(
        SVC(),
        svc_grid,
        X_combined, y,
        n_splits=n_fold  # pastikan fungsi optimize_model menerima ini
    )
    print(f"✅ Best Params ({n_fold} fold):", best_svc_params)

    # Simpan berdasarkan jumlah fold
    optimized_models['SVC'][f'{n_fold}_fold'] = {
        'model': best_svc,
        'params': best_svc_params
    }
```

Lampiran E1 26 Optimasi Model Support Vector Classifier (SVC)

Kode di atas digunakan untuk mengoptimalkan model Support Vector Classifier (SVC) menggunakan validasi silang TimeSeriesSplit dengan jumlah fold 3 dan 5. Grid parameter svc_grid berisi beberapa kombinasi nilai hyperparameter yang akan dievaluasi, termasuk parameter C untuk regularisasi, kernel untuk jenis kernel yang digunakan (linear atau radial basis function/rbf), dan gamma untuk metode pemrosesan parameter gamma (scale atau auto).

Sebuah dictionary bernama optimized_models['SVC'] dibuat untuk menyimpan hasil optimasi untuk masing-masing jumlah fold. Dalam setiap iterasi terhadap jumlah fold yang ditentukan dalam optimized_folds, sistem mencetak bahwa proses optimasi sedang berlangsung. Fungsi optimize_model() akan mencari model terbaik berdasarkan kombinasi parameter dari svc_grid, lalu mengembalikan model terbaik (best_svc) dan parameter terbaiknya (best_svc_params). Hasil terbaik dari setiap konfigurasi disimpan ke dalam dictionary optimized_models berdasarkan jumlah fold yang digunakan.

```python
# MLP

mlp_grid = {
    'model__hidden_layer_sizes': [(100,), (50, 50)],
    'model__activation': ['relu', 'tanh'],
    'model__alpha': [0.0001, 0.001]
}

optimized_models['MLP'] = {}  # Buat dict nested untuk tiap fold

for n_fold in optimized_folds:
    print(f"\n🔷 Optimasi mlp dengan {n_fold}-fold TimeSeriesSplit")
    best_mlp, best_mlp_params = optimize_model(
        MLPClassifier(max_iter=1000, random_state=42),
        mlp_grid,
        X_combined, y,
        n_splits=n_fold  # pastikan fungsi optimize_model menerima ini
    )
    print(f"✅ Best Params ({n_fold} fold):", best_mlp_params)

    # Simpan berdasarkan jumlah fold
    optimized_models['MLP'][f'{n_fold}_fold'] = {
        'model': best_mlp,
        'params': best_mlp_params
    }
```

Lampiran E1 27 Optimasi Model Multi-Layer Perceptron (MLP)

Kode di atas digunakan untuk melakukan optimasi model Multi-Layer Perceptron (MLPClassifier) dengan pendekatan validasi silang menggunakan TimeSeriesSplit sebanyak 3 dan 5 fold. Parameter yang diuji dalam grid mlp_grid meliputi hidden_layer_sizes untuk menentukan arsitektur jaringan tersembunyi, activation untuk memilih fungsi aktivasi seperti ReLU atau tanh, serta alpha yang merupakan parameter regularisasi L2.

Dictionary optimized_models['MLP'] disiapkan untuk menyimpan hasil model terbaik dari masing-masing jumlah fold. Dalam setiap iterasi jumlah fold, fungsi optimize_model() dijalankan dengan parameter grid dan konfigurasi tertentu, lalu hasil terbaiknya—baik model maupun parameter terbaik—dicetak dan disimpan ke dalam dictionary berdasarkan nama fold-nya. Model MLP dikonfigurasi agar melakukan pelatihan maksimal sebanyak 1000 iterasi dengan nilai random_state ditetapkan untuk memastikan reprodusibilitas.

```python
print("\n☑ Evaluasi Ulang Setelah Optimasi")
results_optimized = {}
for name, folds_dict in optimized_models.items():
    for fold_name, content in folds_dict.items():
        model = content['model']
        key = f"{name}_{fold_name}"
        results_optimized[key] = evaluate_model(model, X_combined, y, model_name=key)
```

Lampiran E1 28 Evaluasi Model Setelah Optimasi

Kode di atas digunakan untuk melakukan evaluasi ulang terhadap seluruh model yang telah dioptimasi. Langkah ini dimulai dengan mencetak notifikasi bahwa proses evaluasi dimulai. Sebuah dictionary kosong bernama results_optimized disiapkan untuk menyimpan hasil evaluasi dari masing-masing model. Selanjutnya, dilakukan iterasi terhadap setiap model dalam optimized_models, yang sudah dikelompokkan berdasarkan nama model dan jumlah fold validasi silang. Untuk setiap kombinasi model dan jumlah fold, objek model diambil dari dictionary, kemudian dievaluasi menggunakan fungsi evaluate_model() dengan input fitur X_combined, target y, dan label nama model yang sudah digabung dengan informasi fold. Hasil evaluasi ini disimpan ke dalam dictionary results_optimized dengan kunci nama model dan jumlah fold-nya.

```
df_optimized = pd.DataFrame(results_optimized).T
print("\n📈 Rangkuman Evaluasi Model (Tahap Optimasi):")
df_optimized
```

Lampiran E1 29 Rangkuman Evaluasi Model Tahap Optimasi

Baris kode di atas membuat sebuah DataFrame baru bernama df_optimized dari dictionary hasil evaluasi results_optimized, kemudian melakukan transpose agar nama model dan fold menjadi indeks baris dan metrik evaluasi menjadi kolom. Setelah itu, mencetak teks untuk menunjukkan bahwa proses menampilkan rangkuman hasil evaluasi model setelah optimasi telah dimulai. Terakhir, kode menampilkan isi dari df_optimized, yang memuat metrik performa seperti akurasi, presisi, recall, dan F1-score dari setiap model hasil optimasi berdasarkan jumlah fold yang digunakan.

```
df_optimized_pro = df_optimized.copy()

df_optimized_pro = df_optimized_pro.reset_index()
df_optimized_pro.rename(columns={'index': 'model_name'}, inplace=True)

# Ekstrak nama algoritma: RandomForest, XGBoost, LogReg, SVC, MLP
df_optimized_pro['model'] = df_optimized_pro['model_name'].str.extract(r'^([A-Za-z]+)')

# Pastikan kolom f1_mean bertipe float (hindari NaN/string)
df_optimized_pro['f1_mean'] = pd.to_numeric(df_optimized_pro['f1_mean'], errors='coerce')

# Ambil model terbaik (berdasarkan f1 tertinggi) untuk setiap algoritma
best_models = df_optimized_pro.loc[df_optimized_pro.groupby('model')['f1_mean'].idxmax()]
best_models
```

Lampiran E1 30 Seleksi Model Terbaik Berdasarkan F1-Score Tertinggi

Baris kode di atas melakukan proses seleksi model terbaik dari hasil optimasi berdasarkan nilai F1-score rata-rata tertinggi. Pertama, DataFrame df_optimized disalin ke df_optimized_pro untuk menjaga data asli. Kemudian, indeks direset dan kolom indeks diubah namanya menjadi model_name. Nama algoritma seperti RandomForest, XGBoost, LogReg, SVC, dan MLP

diekstrak dari kolom model_name menggunakan ekspresi reguler, lalu disimpan dalam kolom model. Untuk memastikan perhitungan yang valid, kolom f1_mean dikonversi menjadi tipe float, menghindari kemungkinan nilai kosong atau bertipe string. Terakhir, dari setiap jenis algoritma, dipilih satu model dengan nilai f1_mean tertinggi, lalu hasilnya ditampilkan sebagai model-model terbaik dari setiap algoritma yang telah dioptimasi.

```
df_historis_pro = df_historis.copy()

df_historis_pro = df_historis_pro.reset_index()
df_historis_pro.rename(columns={'index': 'model'}, inplace=True)

# Ekstrak nama algoritma: RandomForest, XGBoost, LogReg, SVC, MLP
df_historis_pro['model'] = df_historis_pro['model'].str.extract(r'^([A-Za-z]+)')
df_historis_pro
```

Lampiran E1 31 Transformasi Data Historis untuk Visualisasi Model

Baris kode di atas menyiapkan ulang DataFrame df_historis agar siap digunakan untuk analisis atau visualisasi lebih lanjut. Pertama, df_historis disalin ke variabel baru df_historis_pro untuk menjaga data asli tetap utuh. Kemudian, indeks direset agar menjadi kolom biasa, dan kolom tersebut diubah namanya menjadi model. Selanjutnya, bagian nama algoritma dari kolom model diekstrak menggunakan ekspresi reguler sehingga hanya menyisakan nama algoritma seperti RandomForest, XGBoost, LogReg, SVC, atau MLP. Data yang sudah diproses ini ditampilkan sebagai output.

```
df_combined_pro = df_combined.copy()

df_combined_pro = df_combined_pro.reset_index()
df_combined_pro.rename(columns={'index': 'model'}, inplace=True)

# Ekstrak nama algoritma: RandomForest, XGBoost, LogReg, SVC, MLP
df_combined_pro['model'] = df_combined_pro['model'].str.extract(r'^([A-Za-z]+)')
df_combined_pro
```

Lampiran E1 32 Transformasi Data Gabungan untuk Analisis Perbandingan

Baris kode di atas digunakan untuk menyiapkan ulang DataFrame df_combined agar lebih mudah dianalisis secara agregat. Salinan data disimpan ke dalam df_combined_pro untuk mencegah perubahan pada data asli. Indeks DataFrame diubah menjadi kolom biasa dan dinamai ulang menjadi model. Setelah itu, nama algoritma utama diekstrak dari kolom model menggunakan ekspresi reguler sehingga hanya menyisakan nama seperti RandomForest, XGBoost, LogReg, SVC, atau MLP. Data hasil transformasi ini ditampilkan untuk dianalisis lebih lanjut.

```
# Tambahkan kolom untuk penanda dataframe (eksperimen)
df_historis_pro['source'] = 'Model Historis'
df_combined_pro['source'] = 'Model Gabungan'
best_models['source'] = 'Model Optimasi'

# Gabungkan semua dataframe
df_all = pd.concat([df_historis_pro, df_combined_pro, best_models], ignore_index=True)

order = ['Model Historis', 'Model Gabungan', 'Model Optimasi']
df_all['source'] = pd.Categorical(df_all['source'], categories=order, ordered=True)
```

Lampiran E1 33 Penggabungan Dataframe Berdasarkan Sumber Eksperimen

Kode di atas menambahkan kolom source ke dalam setiap DataFrame (df_historis_pro, df_combined_pro, dan best_models) untuk menandai asal data, masing-masing diberi label 'Model Historis', 'Model Gabungan', dan 'Model Optimasi'. Ketiga DataFrame tersebut kemudian digabungkan menjadi satu kesatuan DataFrame baru bernama df_all menggunakan fungsi pd.concat(). Untuk memastikan urutan kategori pada kolom source sesuai dengan urutan logis eksperimen, kolom ini dikonversi menjadi tipe kategori terurut dengan urutan yang telah ditentukan.

```
df_all
```

Lampiran E1 34 Menampilkan Dataframe Gabungan Semua Model

Baris kode df_all digunakan untuk menampilkan isi dari DataFrame gabungan yang sebelumnya telah dibentuk melalui proses penggabungan antara model historis, model gabungan, dan model optimasi. DataFrame ini memuat seluruh informasi performa model dari ketiga sumber tersebut secara bersamaan, lengkap dengan kolom identifikasi sumber (source) untuk keperluan analisis perbandingan antar model.

```
# Korelasi Pearson antara F1 mean dan std
correlation = df_all['f1_mean'].corr(df_all['f1_std'], method='pearson')

print(f"Korelasi antara F1 mean dan F1 std: {correlation:.2f}")
```

Lampiran E1 35 Korelasi Pearson antara Rata-rata dan Standar Deviasi F1-Score

Baris kode di atas menghitung nilai korelasi Pearson antara kolom f1_mean dan f1_std dalam DataFrame df_all. Korelasi ini menunjukkan derajat hubungan linier antara rata-rata F1-score dan penyebarannya (standar deviasi). Nilai korelasi disimpan dalam variabel correlation dan kemudian ditampilkan dalam format dua desimal untuk menginterpretasikan apakah model dengan performa rata-rata tinggi juga memiliki stabilitas (penyimpangan kecil) atau sebaliknya.

```
sns.scatterplot(data=df_all, x='f1_mean', y='f1_std', hue='source')
plt.title("Trade-off antara F1 Mean dan Std")
plt.xlabel("F1 Mean")
plt.ylabel("F1 Std")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Lampiran E1 36 Visualisasi Trade-off antara Rata-rata dan Standar Deviasi F1-Score

Baris kode di atas menghasilkan plot sebar (scatter plot) untuk menggambarkan hubungan antara nilai rata-rata F1-score (f1_mean) dan standar deviasinya (f1_std) dari seluruh model yang dianalisis dalam df_all. Data dikelompokkan berdasarkan kolom source, yang menunjukkan asal model (historis, gabungan, atau hasil optimasi), dan dibedakan warnanya dengan parameter hue. Judul grafik serta label sumbu x dan y ditambahkan untuk memperjelas makna visualisasi. Fitur plt.grid(True) mengaktifkan grid untuk memudahkan pembacaan nilai, sementara plt.tight_layout() memastikan elemen-elemen visual tertata rapi tanpa tumpang tindih, lalu grafik ditampilkan dengan plt.show().

```
plt.figure(figsize=(8,6))
sns.scatterplot(
    data=df_all,
    x="f1_mean",
    y="f1_std",
    hue="source",
    palette={"Model Historis": "blue", "Model Gabungan": "green", "Model Optimasi": "red"},
    alpha=0.7
)

# Tambahkan jitter kecil secara manual jika perlu
df_jittered = df_all.copy()
df_jittered['f1_mean'] += np.random.normal(0, 0.001, size=len(df_all))  # jitter di mean
df_jittered['f1_std'] += np.random.normal(0, 0.001, size=len(df_all))   # jitter di std

sns.scatterplot(
    data=df_jittered[df_jittered['source'] == 'Model Gabungan'],
    x="f1_mean",
    y="f1_std",
    color="green",
    marker='o',
    edgecolor='black'
)
plt.title("Scatter Plot dengan Jitter (Model Gabungan)")
plt.show()
```

Lampiran E1 37 Scatter Plot dengan Jitter pada Model Gabungan

Kode di atas menghasilkan scatter plot yang menampilkan hubungan antara nilai rata-rata F1-score (f1_mean) dan standar deviasi F1-score (f1_std) dari seluruh model, dengan pewarnaan berdasarkan asal model (source). Warna biru digunakan untuk Model Historis, hijau untuk Model Gabungan, dan merah untuk Model Optimasi, sementara transparansi titik diatur dengan alpha=0.7. Untuk mengatasi kemungkinan tumpang tindih antar titik data, ditambahkan jitter

secara manual ke nilai f1_mean dan f1_std khusus pada Model Gabungan, dengan menyisipkan gangguan acak berdistribusi normal kecil. Titik-titik Model Gabungan yang telah diberi jitter ditampilkan kembali dengan tepi berwarna hitam, agar perbedaan posisi lebih terlihat. Visualisasi ini bertujuan memberikan gambaran distribusi performa dan variasi antar model secara lebih jelas.

```
# Gabungkan mean dan std sebagai string anotasi
df_all['f1_label'] = df_all.apply(lambda x: f"{x['f1_mean']:.2f} - {x['f1_std']:.2f}", axis=1)

# Pivot untuk heatmap dengan anotasi gabungan
pivot_val = df_all.pivot(index='source', columns='model', values='f1_mean')
pivot_label = df_all.pivot(index='source', columns='model', values='f1_label')

# Visualisasi heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(pivot_val, annot=pivot_label, cmap='YlGnBu', fmt="", cbar_kws={'label': 'F1 Score'})
plt.title("F1 Score Heatmap (Mean - Std)")
plt.xlabel("Algoritma")
plt.ylabel("Model")
plt.tight_layout()
plt.show()
```

Lampiran E1 38 Visualisasi Heatmap F1 Score (Mean – Std)

Kode di atas menghasilkan heatmap yang menampilkan nilai F1-score rata-rata (mean) dan standar deviasi (std) dari berbagai model dan algoritma dalam format anotasi gabungan. Pertama-tama, dibuat kolom baru bernama f1_label pada dataframe df_all, yang menggabungkan nilai f1_mean dan f1_std dalam satu string dengan format “mean – std”. Data kemudian dipivot menjadi dua bentuk: satu untuk nilai (pivot_val) yang akan divisualisasikan sebagai warna dalam heatmap, dan satu lagi (pivot_label) untuk teks anotasi pada masing-masing sel. Heatmap dibuat dengan palet warna 'YlGnBu' untuk menunjukkan tingkat performa, dilengkapi dengan bar warna yang merepresentasikan nilai F1-score. Label sumbu horizontal menunjukkan jenis algoritma, sedangkan sumbu vertikal menunjukkan tipe model (misalnya Model Historis, Gabungan, atau Optimasi). Hasil visualisasi ini membantu membandingkan performa antar model secara menyeluruh dan terstruktur.

Link Kode Pemrograman : - Preprocessing Dataset : Preprocessing Dataset Code

- EDA Dataset : EDA Dataset Code

- Main Notebook : Main Notebook Code

B. **Lampiran E2 – Struktur *Folder* Proyek**

```
stock-price-sentiment/
│
├── data/                              # Folder untuk menyimpan dataset
│   ├── raw/                           # Data mentah (belum diproses)
│   └── processed/                     # Data setelah preprocessing
│
├── notebook/                          # Jupyter Notebook interaktif
│   ├── eda_template.ipynb             # Template untuk eksplorasi data
│   └── preprocessing_template.ipynb   # Template untuk preprocessing
│
├── report/                            # Template laporan akhir
│   ├── laporan-akhir.docx
│   ├── laporan-akhir.pdf
│   └── presentasi-akhir.pdf
│
├── src/                               # Source code modular
│   ├── data_loader.py                 # Fungsi load dan simpan data
│   ├── preprocessing.py               # Fungsi preprocessing data dan pengadaan fitur-fitur pendukung baru
│   ├── model.py                       # Fungsi template model
│   ├── historis.py                    # Fungsi model berdasarkan sekedar data dan fitur historis
│   ├── merged.py                      # Fungsi model berdasarkan data dan fitur gabungan antara historis dan sentimen
│   ├── optimalization.py              # Fungsi mengoptimalkan model gabungan berdasarkan pendekatan parameter terbaik
│   ├── visualization.py               # Fungsi visualisasi dari hasil perbandingan semua model
│   ├── main.py                        # Main pipeline untuk dijalankan via terminal
│   └── main_notebook.ipynb            # Versi notebook dari main.py
│
├── run.sh                             # Script bash untuk menjalankan pipeline
├── requirements.txt                   # Daftar dependensi Python
└── README.md                          # Dokumentasi ini
```