Contoh Halaman Pengesahan

SISTEM PREDIKSI HARGA KOMODITAS PERTANIAN DI JAWA TIMUR BERBASIS WEBSITE

SKRIPSI

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

Oleh:

ALVINA MARCY S.P. **NIM. 2141720017**

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

HALAMAN PENGESAHAN

SISTEM PREDIKSI HARGA KOMODITAS PERTANIAN DI JAWA TIMUR BERBASIS WEBSITE

Disusun oleh:

ALVINA MARCY S.P. NIM. 2141720017

Laporan Akhir ini telah diuji pada tanggal 21 Juni 2021

Disetujui oleh:

1.	Pembimbing Utama	:	Imam Fahrur Rozi, S.T., M.T. NIP. 19840610 200812 1 004	.....
2.	Pembimbing Pendamping	:	Ir. Deddy Kusbianto P., M.MKom. NIP. 19621128 198811 1 001	.....
3.	Penguji Utama	:	Budi Harijanto, S.T., M.MKom. NIP. 19620105 199003 1 002	.....
4.	Penguji Pendamping	:	Hendra Pradibta, SE., M.Sc NIP. 19830521 200604 1 003	.....

Mengetahui,

Ketua Jurusan Teknologi Informasi	Ketua Program Studi Teknik Informatika
Rudy Ariyanto, S.T., M.Cs. NIP. 19711110 199903 1 002	Imam Fahrur Rozi, S.T., M.T. NIP. 19840610 200812 1 004

Contoh Halaman Pernyataan

PERNYATAAN

Dengan ini saya menyatakan bahwa pada Skripsi ini tidak terdapat karya, baik seluruh maupun sebagian, yang sudah pernah diajukan untuk memperoleh gelar akademik di Perguruan Tinggi manapun, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini serta disebutkan dalam daftar sitasi/pustaka.

Malang, 21 Juli 2025

Alvina Marcy S.P.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

ABSTRAK

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

ABSTRACT

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

KATA PENGANTAR

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

DAFTAR ISI

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

DAFTAR GAMBAR

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

DAFTAR TABEL

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

DAFTAR LAMPIRAN

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

BAB I. PENDAHULUAN

1.1 Latar Belakang

Badan Pusat Statistik Indonesia menyebutkan bahwa pada tahun 2024, jumlah penduduk Indonesia telah melebihi 280 juta jiwa (BPS Indonesia, 2024). Dengan populasi sebesar itu, kebutuhan pangan di Indonesia menjadi sangat tinggi. Menurut Badan Pusat Statistik Indonesia pertanian masih menjadi sektor utama dalam memenuhi kebutuhan tersebut, sehingga pertanian memainkan peran vital dalam pembangunan ekonomi dan ketahanan pangan nasional (BPS, 2024). Salah satu permasalahan utama yang dihadapi sektor pertanian adalah fluktuasi harga komoditas. Perubahan yang terjadi di sektor ini dapat membawa dampak signifikan terhadap kondisi sosial dan ekonomi negara, termasuk di Provinsi Jawa Timur. Salah satu permasalahan yang menonjol dalam sektor pertanian adalah fluktuasi harga komoditas yang sering kali tidak dapat diprediksi, seperti harga cabai rawit yang berubah dengan cepat akibat berbagai faktor seperti biaya produksi, jumlah produksi, kondisi cuaca, dan permintaan pasar (Husna et al., 2024).

Jawa Timur merupakan salah satu provinsi utama penghasil komoditas pertanian di Indonesia. Kontribusinya yang besar terhadap kebutuhan pangan nasional menjadikan provinsi ini sebagai objek yang menarik untuk diteliti, khususnya dalam mengatasi permasalahan fluktuasi harga. Ketidakstabilan harga tidak hanya memengaruhi pendapatan petani, tetapi juga mempersulit pelaku pasar dalam merencanakan strategi produksi dan distribusi. Oleh karena itu, penting untuk menyediakan alat bantu yang dapat memberikan informasi prediksi harga komoditas secara akurat di wilayah ini.

Permasalahan khusus yang dihadapi adalah kurangnya sistem prediksi harga komoditas yang dapat digunakan secara langsung oleh petani dan pelaku pasar. Penelitian terdahulu oleh Khalis Sofi et al. (2021) melakukan perbandingan antara LSTM, GRU dan Linear Regression untuk melakukan prediksi harga saham dengan data time-series. Penelitian ini menunjukkan bahwa model yang memiliki akurasi baik untuk melakukan prediksi harga yaitu model LSTM dengan akurasi RMSE sebesar 0.048 dan GRU dengan akurasi RMSE 0.034. Sedangkan Linear Regression

memiliki akurasi yang terbilang buruk untuk data time-series. Selain itu, penelitian oleh Fitria et al. (2017) yang membandingkan antara metode ARIMA dan Exponential Smoothing menunjukkan bahwa kedua model ini memiliki akurasi yang tinggi. Pada penelitian ini model ARIMA didapatkan bahwa pengujian MAPE menghasilkan 0.38% dan untuk model Exponential Smoothing mendapatkan nilai MAPE sebesar 0.378%.

Untuk mengatasi masalah ini, diperlukan pengujian akurasi dari berbagai metode prediksi. Dengan dataset yang memiliki historis dan time-series yang kompleks serta data yang cenderung memiliki trend dan musiman sehingga pendekatan prediksi dengan menggunakan LSTM, GRU, ARIMA dan Exponential Smoothing akan digunakan. Dimana LSTM dan GRU yang direkomendasikan untuk data time-series yang kompleks dan ARIMA dan Exponential Smoothing yang direkomendasikan untuk data yang memiliki tren dan musiman yang jelas. Dalam penelitian ini, prediksi harga akan dilakukan pada beberapa komoditas pertanian utama di Jawa Timur, yaitu **cabai rawit, cabai merah besar, bawang merah, bawang putih, dan tomat**. Pemilihan komoditas ini didasarkan pada peran pentingnya dalam pasar pertanian Jawa Timur serta tingkat volatilitas harga yang tinggi.

Tujuan dari pengujian ini adalah untuk mengetahui metode yang paling akurat dalam memprediksi harga komoditas pertanian menggunakan data time-series. Selain itu, dibutuhkan pengembangan sistem berbasis website yang memungkinkan pengguna, khususnya petani untuk dengan mudah memanfaatkan hasil prediksi tersebut dalam pengambilan keputusan. Penelitian ini akan menggunakan metode machine learning untuk membangun sistem prediksi harga komoditas pertanian yang terintegrasi dalam sebuah website. Sistem ini akan diuji secara fungsional dan akurasinya dibandingkan menggunakan metrik seperti RMSE, MAE, dan MAPE. Dengan hasil uji yang didapat sehingga metode dengan akurasi tertinggi akan digunakan dan diimplementasikan pada website.

Oleh sebab itu, pada penelitian ini diusulkan **"Sistem Prediksi Harga Komoditas Pertanian di Jawa Timur Berbasis Website"**. Diharapkan sistem ini dapat memberikan informasi prediksi harga komoditas yang akurat dan mudah

diakses. Sehingga, petani dapat merencanakan strategi penanaman dan pemasaran dengan lebih baik, serta membantu meningkatkan stabilitas ekonomi dan efisiensi pasar pertanian di Jawa Timur.

1.2 Rumusan Masalah

1. Metode prediksi apa yang paling akurat untuk memprediksi harga komoditas pertanian di Jawa Timur?
2. Bagaimana menyediakan informasi yang mudah di akses dan real-time bagi para petani?

1.3 Batasan Masalah

Agar pengembangan ini lebih terarah, batasan masalah ditentukan sebagaimana Berikut:

1. Prediksi harga dibatasi pada wilayah Jawa Timur.
2. Sistem hanya akan mencakup jenis komoditas **cabai rawit, cabai merah besar, bawang merah, bawang putih, dan tomat** pada website Siskaperbapo oleh Disperindag Jatim.
3. Sistem hanya mencakup prediksi sejauh 90 hari karena ketidakpastian faktor eksternal.
4. Sistem menggunakan data historis harga komoditas dari website Siskaperbapo Disperindag Jatim.
5. Metode prediksi pada algoritma tertentu, yaitu **LSTM**, **GRU**, **Exponential Smooting** dan **SARIMA**. Algoritma lain tidak diterapkan.

1.4 Tujuan

Tujuan dari dilakukannya skripsi dengan judul "**SISTEM PREDIKSI HARGA KOMODITAS PERTANIAN DI JAWA TIMUR**", adalah sebagai berikut:

- Mengetahui metode predikasi yang paling akurat untuk memprediksi harga komoditas pertanian di Jawa Timur.
- Memprediksi harga komoditas pertanian secara akurat untuk membantu petani dalam mengambil keputusan yang lebih baik.

- Menghasilkan sistem prediksi berbasis teknologi yang dapat diandalkan untuk memberikan rekomendasi harga komoditas secara tepat.
- Menyediakan informasi yang mudah diakses dan real-time untuk semua pemangku kepentingan di sektor pertanian.

1.5 Manfaat

Manfaat dari penelitian ini adalah untuk memberikan informasi prediksi harga yang akurat sehingga petani dapat menentukan waktu panen dan harga jual yang lebih menguntungkan. Selain itu prediksi harga dapat mengurangi resiko kerugian akibat fluktuasi harga yang tidak terduga.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

BAB II. LANDASAN TEORI

2.1 Studi Literatur

No	Penulis	Judul Penelitian	Hasil	Perbedaan
1	(Adi & Sudianto, 2022)	Prediksi Harga Komoditas Pangan Menggunakan Algoritma *Long Short-Term Memory* (LSTM)	Penelitian ini menunjukkan bahwa model LSTM memiliki akurasi tinggi dalam melakukan prediksi harga dengan hasil RMSE sebesar 79.1%	Penelitian ini hanya menggunakan metode LSTM tanpa metode pembanding dan model prediksi tidak diintegrasikan kedalam sistem.
2	(Khalis Sofi et al., 2021)	Perbandingan Algoritma Linear Regression, LSTM dan GRU dalam Memprediksi Harga Saham Dengan Model Time Series	Penelitian ini menunjukkan bahwa LSTM dan GRU memiliki akurasi yang baik namun untuk Linear Regression tidak cocok untuk memprediksi data dengan model time-series	Penelitian ini menggunakan harga saham sebagai objek yang di teliti serta model prediksi tidak diintegrasikan dalan sebuah sistem.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

3	(Fitria et al., 2017)	Perbandingan Metode ARIMA dan Double Exponential Smoothing pada Peramalan Harga Saham LQ45 Tiga Perusahaan dengan Nilai Earning Per Share (EPS) Tertinggi	Penelitian ini menyebutkan bahwa metode ARIMA dan Exponentian Smoothing memiliki akurasi yang baik untuk melakukan peramalan harga saham dengan metode Double Exponentian Smoothing yang sedikit lebih unggul dibanding ARIMA	Penelitian ini menggunakan harga saham sebagai objek penelitian dan model prediksi tidak diintegrasika dalam sebuah sistem.
4	(Junita & Primandari, 2023)	Perbandingan Metode Double Exponential Smoothing dan Metode Triple Exponential Smoothing untuk Harga Telur pada Produsen di Kabupaten Sukabumi	Penelitian ini menunjukkan bahwa *Double Exponential Smoothing* menghasilkan akurasi prediksi lebih tinggi dibandikan Triple Exponential Smoothing	Penelitian ini menggunakan dataset yang hanya memiliki trend tanpa musiman sedangkan penelitian penulis menggunakan dataset yang

				memiliki trend dan musiman. Model prediksi tidak diintergasikan dalam sebuah sistem.
5	(Sarah et al., 2024)	Peramalan Harga Cabai Rawit Merah di Jawa Timur Menggunakan Metode AutoRegressive Integrated Moving Average (ARIMA)	Penelitian ini menunjukkan bahwa metode ARIMA dapat digunakan dalam peramalan harga cabai merah dengan akurasi tertinggi didapatkan dengan ARIMA(4,2,4) yang menghasilkan MAPE sebesar 2.83625%	Pada penelitian ini prediksi hanya menggunakan satu metode dan pemilihan model tentatif terbaik menggunakan AIC. Sedangkan pada penelitian penulis
6	(Saputra et al., 2023)	Penerapan Deep Learning Menggunakan Gated Recurrent Unit Untuk Memprediksi	Prediksi menggunakna metode GRU untuk prediksi harga minyak mentah dunia	Pada penelitian ini prediksi hanya menggunakan satu metode

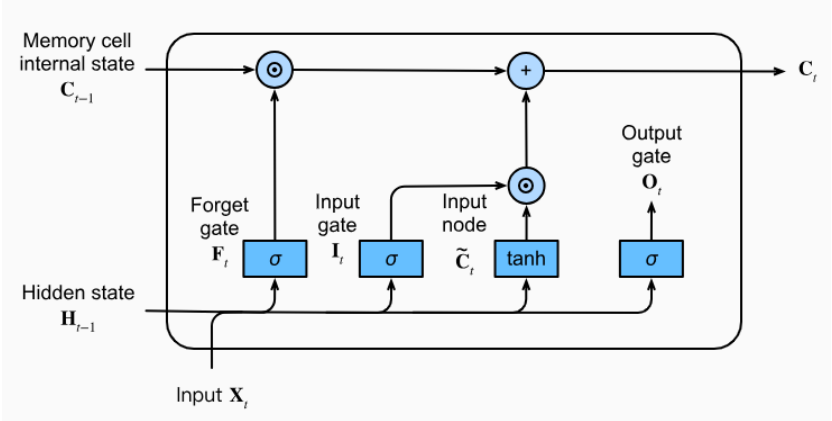
		Harga Minyak Mentah Dunia	menghasilkan akurasi tertinggi menggunakna *hyperparemeter* 30 *lockback* 50 unit GRU dan 256 *batch size*	dan objek prediksi harga minyak.

2.1.1

2.2 Dasar Teori

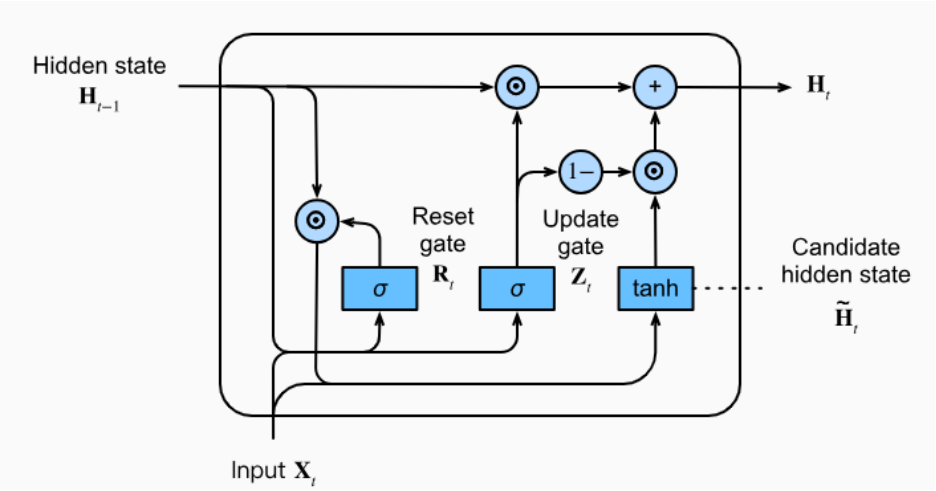
2.2.1 Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) merupakan turunan dari metode *Recurrent Neural Network* (RNN) yang rancang untuk mengatasi terjadinya *vanishing gradient* yang sering terjadi pada RNN. Dengan menambahkan *memory cell*, LSTM mampu menyimpan informasi dalam jangka waktu yang lama, sehingga dapat menangkap ketergantungan jangka panjang dalam data (Manaswi, 2018). Arsitektur LSTM terdiri dari tiga gerbang utama yaitu *input gate, forget gate, dan output gate*. (Trivusi, 2022)



2.2.2 Gated Recurrent Unit (GRU)

Gated Reccurent Unit (GRU) adalah varian dari *Recurrent Neural Network* (RNN) untuk menyederhanakan arsitektur dari LSTM dengan keunggulan penggunaan memori yang lebih efisien dan kemampuan yang efektif dalam memproses data sekuensial. GRU memiliki dua gerbang utama sebagai pengatur informasi yaitu update gate dan reset gate. Update gate berfungsi mempertahankan berapa banyak informasi masalalu yang tetap disimpan dan reset gate bertugas menghubungkan input baru dengan informasi masalalu dan menghapus informasi masalalu yang sudah tidak diperlukan (Prihatmikho et al., 2023). GRU bekerja dengan membuat setiap recurrent unit untuk dapat menangkap dependencies dalam skala waktu yang berbeda-beda secara adaptif, dengan maksud GRU hanya akan menggunakan informasi masalalu dengan skala waktu tertentu bukan menggunakan semua informasi masa lalu.



Gambar 2. 1 Arsitektur GRU (Zhang et al., 2024)

- *Reset Gate*

  Reset gate mengontrol bagaimana *hidden state* sebelumnya (H_{t-1}) digunakan dalam perhitungan *candidate hidden state (Zhang et al., 2024). Dengan rumus sebagai berikut:*

$$\mathbf{R}_t = \sigma(\mathbf{X}_t\mathbf{W}_{\text{xr}} + \mathbf{H}_{t-1}\mathbf{W}_{\text{hr}} + \mathbf{b}_{\text{r}})$$

- *Update Gate*

Update gate menentukan seberapa banyak informasi dari *hidden state* sebelumnya H_{t-1} yang akan dibawa ke langkah waktu berikutnya (Zhang et al., 2024). *Dengan rumus sebagai berikut:*

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t\mathbf{W}_{\text{xz}} + \mathbf{H}_{t-1}\mathbf{W}_{\text{hz}} + \mathbf{b}_z)$$

- *Candidate Hidden State*

  Candidate hidden state adalah nilai sementara yang merepresentasikan *hidden state* baru sebelum dikombinasikan dengan informasi dari *hidden state* sebelumnya (Zhang et al., 2024). Dengan rumus sebagai berikut:

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t\mathbf{W}_{\text{xh}} + (\mathbf{R}_t \odot \mathbf{H}_{t-1})\mathbf{W}_{\text{hh}} + \mathbf{b}_{\text{h}})$$

- *Final Hidden State*

  Hidden state pada langkah waktu ***t*** adalah kombinasi linier antara *hidden state* sebelumnya H_{t-1} dan *candidate hidden state* $\tilde{\mathbf{H}}_t$, dikontrol oleh update gate Z_t (Zhang et al., 2024). Dengan rumus sebagai berikut:

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t.$$

Keterangan :

X_t = Inputan saat ini

$W_{xr}, W_{hr}, W_{xz,}, W_{hz,}, W_{xh,} W_{hh,}$ = Matrix bobot

$b_r, b_{z,} b_h$ = Bias masing-masing gate

σ = Aktifasi Sigmoid

tanh = aktifasi hiperbolik tangen

2.2.3 Exponential Smoothing

Exponential Smoothing adalah metode peramalan yang melakukan perkiraan nilai di periode mendatang menggunakan nilai rata-rata dari data dengan membrikan bobot yang lebih tinggi pada data terbaru untuk mengurangi fluktuasi jangka pendek atau tren jangka panjang (Wijaya, 2023). Model *Exponential Smoothing* memiliki beberapa jenis metode yang dapat digunakan sesuai dengan tipe datanya. Untuk tipe data yang memiliki trend dan pola musiman maka metode Holts-Winters *Exponential Smoothing (Triple Exponential Smoothing)* paling cocok digunakan dengan pemulusan sebanyak tiga kali. Tiga perameter pemulusan

tersebut ada α (untuk level), β (untuk permulusan trend), dan γ (untuk komponen musiman) (Junita & Primandari, 2023). Berikut merupakan persamaan yang digunakan adalah :

Permulusasn Level : $S_t = \alpha(X_t - I_{t-s}) + (1-\alpha)(S_{t-1} + T_{t-1})$

Permulusan Trend : $T_t = \beta(S_t - S_{t-s}) + (1-\beta)T_{t-1}$

Permulusan Musiman : $I_t = \gamma(X_t - S_t) + (1-\gamma)I_{t-s}$

$F_{t+m} = S_t + mT_t + I_{t+m-I}$

Keterangan :

X_t = Data sebenarnya pada waktu t.
I_{t-s} = Komponen musiman periode sebelumnya.
S_{t-s} = Level periode sebelumnya.
F_{t+m} = Prediksi nilai periode t+m kedepan.
mT_t = Proyeksi tren kedepan untuk m periode.
I_{t+m-I} = Komponen musiman yang sesuai untuk periode yang di prediksi.

2.2.4 Autoregressive Integrated Moving Average (ARIMA)

ARIMA (*Autoregressive Integrated Moving Average)* merupakan metode peramalan yang didasarkan oleh deret waktu *(time series)* yang didapat dari kombinasi model AR *(Autoregressive)*, I *(Integrated)*, dan MA *(Moving Average) (Fitria et al., 2017).* Model ARIMA berbentuk (p,d,q) yang dinyatakan dalam persamaan (Maharani & Witanti, 2024) :

$$(1-B)^d Z_t = \varphi_1 Z_{t-1} + \cdots + \varphi_p Z_{t-p} + e_t + \theta_1 e_{t-1} + \cdots + \theta_p e_{t-p}$$

Keterangan :

Z_t = nilai pada waktu t

e_t = galat pada waktu t

θ = parameter pada MA

φ = koefisien pada AR

2.2.5 Mean Absolute Error (MAE)

MAE merupakan metrik yang mengukur nilai rata-rata dari selisih absolut antara nilai sebenarnya dan nilai yang diprediksi oleh model. MAE dikenal sebagai metode yang sedarhana untuk mengukur kesalahan prediksi dan data sekuensial. Nilai Prediksi MAE dihitung dengan rumus (Wijaya, 2023):

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} |Y_t - \hat{Y}_t|$$

Dengan Y_t sebagai nilai data aktual, $\hat{Y}_t$ sebagai nilai data prediksi dan n adalah jumlah pengamatan. MAE memberikan estimasi ukuran ketidak akuratan,tetapi bukan arahnya. (misalnya, apakah prediksi berlebih atau kurang). Semakin kecil nilai MAE yang didapatkan berarti bahwa model menghasilkan prediksi yang semakin baik.

2.2.6 Mean Absolute Percentage Error (MAPE)

MAPE merupakan metrik yang mengukur rata-rata kesalahan absolute antara nilai yang sebenarnya terjadi dengan nilai prediksi yang dihasilkan model. Nilai prediksi MAPE dihitung dengan rumus (Wijaya, 2023):

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n} \frac{|Y_t - \hat{Y}_t|}{Y_t}$$

Dengan Y_t sebagai nilai data aktual, $\hat{Y}_t$ sebagai nilai data prediksi dan n adalah jumlah pengamatan.

2.2.7 Root Mean Squad Error (RMSE)

RMSE merupakan metrik pengukuran dengan menghitung perbedaan nilai dari prediksi sebuah model sebagai estimasi nilai observasi. RMSE merupakan akar dari Mean Square Error. Semakin kecil nilai RMSE yang dihasilkan maka semakin akurat pula model yang didapatkan. Perhitungan untuk memperoleh nilai RMSE suatu model menggunakan rumus (Adi & Sudianto, 2022):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Yang dimana :

y_t = nilai data aktual

$\hat{y}_i$ = nilai hasil peramalan

n = banyaknya data

BAB III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan di Laboratorium Teknologi Informasi, Gedung Pasca Sarjana Politeknik Negeri Malang. Penelitian dilaksanakan selama lima bulan mulai bulan Januari hingga Mei 2025.

3.2 Teknik Pengumpulan Data

Dalam penelitian ini, Teknik pengumpulan data yang digunakan adalah web scrapping dari situs resmi Siskaperbapo (Sistem Informasi Ketersediaan dan Perkembanagn Harga Bahan Pokok di Jawa Timur) yang dikelola oleh Pemerintah Provinsi Jawa Timur. Web scraping adalah proses otomatis untuk mengekstraksi data dari situs web menggunakan perangkat lunak khusus (Bhatt et al., 2023).

Langkah-langkah Web Scraping:

1. Identifikasi Sumber Data:

   Situs yang akan di-scrape adalah Siskaperbapo dengan tautan https://siskaperbapo.jatimprov.go.id/harga/tabel, yang menyediakan informasi harga bahan pokok di Jawa Timur. Data yang relevan mencakup tanggal, nama komoditas, wilayah, nama pasar, satuan, dan harga.
2. Pemilihan Alat dan Teknologi:

   Untuk melakukan web scraping, digunakan bahasa pemrograman Python dengan library seperti BeautifulSoup untuk parsing HTML, Requests untuk mengakses halaman web dan Psycopg2 yang digunakan untuk menyimpan data kedalam tabel basis data.
3. Pengambilan Data:

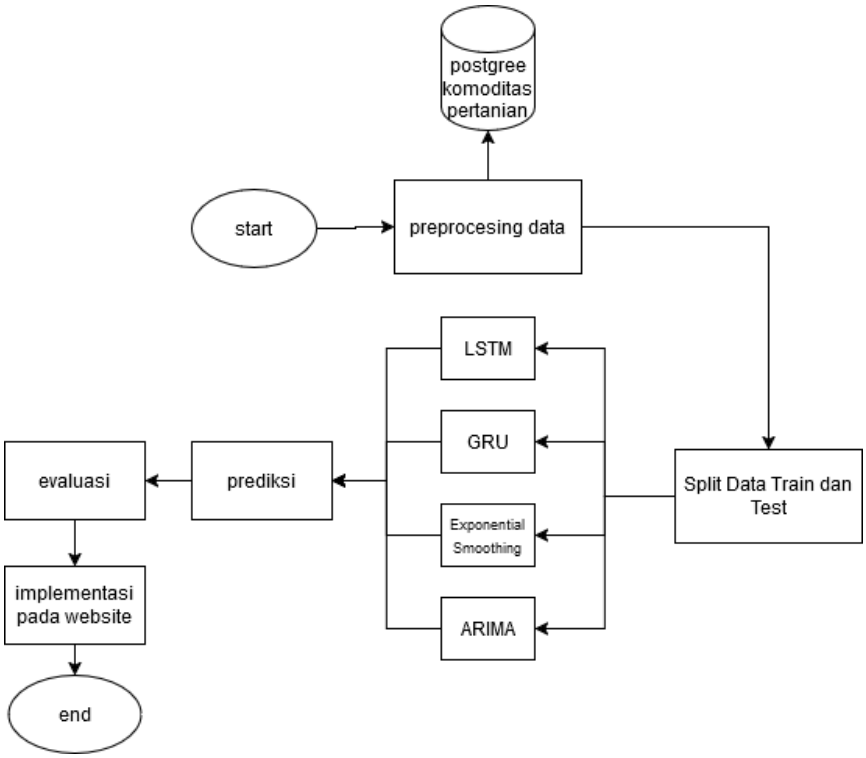
   Kode program akan ditulis untuk mengirim permintaan HTTP ke halaman Siskaperbapo dan mengambil konten HTML. Kemudian, mengekstrak elemen-elemen yang berisi informasi harga komoditas menggunakan BeautifulSoup.
4. Penyimpanan Data:

   Data yang berhasil di ekstrak kemudian akan disimpan dalam database PostgreeSQL untuk memudahkan akses dan analisis lebih lanjut.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

3.3 Teknik Pengolahan Data

Pada penelitian ini data yang telah di dapatkan dari tahap sebelumnya akan dilakukan pengolahan dengan tahapan yang disajikan dalam diagram blok pada Gambar 3.1.



Gambar 3. 1 Diagram Blok Pengolahan Data

3.3.1 Preprocessing Data

Proses pengelolaan data dimulai dengan preprocessing data berdasarkan data yang sebelumnya sudah dipersiapkan pada PostgreeSQL. Preprocessing data, yaitu langkah membersihkan dan mengelola data agar data siap untuk di analisis atau di modelkan. Langkah ini mencakup normalisasi data, penanganan data kosong atau outlier, serta transformasi data sesuai kebutuhan model prediktif. Proses ini bertujuan untuk memastikan kualitas data sehingga menghasilkan prediksi yang akurat. Preprocessing data yang digunakan akan berbeda-beda untuk tiap model mesin learning yang digunakan untuk menyesuaikan kebutuhan tiap model.

3.3.2 Split Data Train dan Data Tes

Setelah *preprocessing* dilakukan, selanjutnya adalah membagi data menjadi dua bagian, yaitu data pelatihan (training) dan pengujian (testing). Pembagian ini

dilakukan untuk melatih model menggunakan data pelatihan lalu menguji performa dari model menggunakan data pengujian. Dari hasil pelatihan model dan pengujian yang di dapat maka dapat diukur tingkat akurasi dan keandalan model yang digunakan.

3.3.3 Pemodelan Dengan Algoritma Prediksi

Setelah data siap, selanjutnya data akan masukkan kedalam setiap model prediksi yang digunakan yaitu LSTM, GRU, SARIMA dan Exponential Smoothing. Setiap model akan dijalankan satu persatu secara bergantian dan kemudian akan dicatat setiap hasil dari model untuk setiap komoditas yang akan di prediksi. Setiap model akan menganalisis data untuk memahami pola data kompleks yang ada.

3.3.4 Prediksi dan Evaluasi

Setelah model dilatih dan diuji, langkah prediksi dilakukan untuk menghasilkan nilai berdasarkan data yang diinputkan. Selanjutnya, evaluasi model dilakukan untuk mengukur kinerja setiap algoritma menggunakan pengukuran akurasi menggunakan RMSE, MAPE, dan MAE. Dengan hasil evaluasi dan pengujian akurasi akan didapatkan metode mana yang paling akurat dan sesuai untuk digunakan dalam memprediksi harga komoditas pertanian. Metode dengan akurasi terbaik inilah yang akan digunakan dalam memprediksi harga serta diimplementasikan pada website.

3.3.5 Implementasi Pada Website

Model prediksi yang telah melalui proses evaluasi diintegrasikan ke dalam sistem berbasis website. Pada tahap ini, pengguna dapat memasukkan data komoditas pertanian untuk memperoleh hasil prediksi secara real-time melalui antarmuka yang dirancang secara interaktif.

3.4 Pengujian Akurasi

Pengujian akurasi model dilakukan untuk mengukur akurasi antar algoritma yang di pakai. Proses ini mencakup pengukuran tingkat error prediksi menggunakan metrik seperti RMSE, MAPE dan MAE serta membandingkan hasil prediksi dengan data historis untuk memastikan keakuratan sistem.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

3.5 Uji Coba Sistem

Uji coba sistem dilakukan setelah seluruh tahap pembuatan sistem selesai. Tujuan dari uji coba sistem ini adalah untuk mengetahui seluruh sistem dapat berjalan sesuai dengan fungsi yang telah di tentukan. Pengujian Fungsional sistem bertujuan untuk memastikan bahwa seluruh fitur utama sistem, seperti proses input data, prediksi harga, dan tampilan hasil di website, berfungsi sesuai dengan spesifikasi yang dirancang. Uji ini berfokus pada pengujian alur pengguna, mulai dari memasukkan data komoditas hingga melihat hasil prediksi yang dihasilkan.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

BAB IV.ANALISIS DAN PERANCANGAN SISTEM

4.1 Deskripsi Sistem

Tujuan utama dari penelitian ini adalah untuk mengetahui model mana yang memiliki akurasi terbaik dan cocok untuk data harga komoditas yang digunakan. Dalam skripsi ini, sebuah sistem prediksi harga komoditas pertanian di Jawa Timur berbasis website dirancang dan dikembangkan untuk memberikan perkiraan harga yang dapat membantu para petani. Sistem ini membandingkan kinerja empat metode popular peramalan deret waktu: LSTM, GRU, ETS dan SARIMA.

Metode-metode ini dipilih karena kemampuannya dalam menangani data deret waktu yang kompleks, termasuk pola musiman dan tren yang seringkali ada pada data harga komoditas pertanian.setelah proses pengujian dan validasi, metode dengan akurasi terbaik akan diimplementasikan sebagai inti dari sistem prediksi. Akurasi model akan dievaluasi menggunakan metrik yang relevan seperti MAE, RMSE dan MAPE, untuk memastikan bahwa perkiraan yang dihasilkan dapat diandalkan.

Sistem berbasis website dirancang agar mudah diakses dan digunakan oleh para petani. Informasi prediksi harga yang akurat diharapkan dapat membantu petani dalam membuat keputusan yang lebih tepat terkait waktu penanaman, panen dan penjualan produk. Dengan demikian sistem ini tidak hanya bertujuan untuk menyediakan data, tetapi juga berkontribusi pada peningkatan kesejahteraan petani melalui manajemen resiko harga yang lebih baik dan optimalisasi strategis agribisnis.

4.2 Analisa Kebutuhan Sistem

Analisis kebutuhan marupakan tahap awal yang penting dalam pengembangan sistem untuk mendeskripsiskan proses, mempermudah implementasi, dan mengurangi kesalahan, karena pengaruh terhadap fungsionalitas sistem yang dibangun.

4.2.1 Analisis Pengguna

Dalam sistem ini hanya terdapat satu pengguna yaitu petani yang memiliki tiga layanan yaitu melihat tren harga, melihat prediksi harga dan melihat historis harga aktual.

4.2.2 Kebutuhan Fungsional

Kebutuhan fungsional mencakup fitur utama yang harus dimiliki sistem untuk memenuhi tujuan pengguna. Berikut adalah kebutuhan yang berhubungan langsung dengan apa yang harus dilakukan oleh perangkat lunak atau sistem.

No	Pengguna	Kebutuhan
1.	Petani	Sistem dapat menampilkan hasil prediksi harga beserta grafik visualisasinya
		Sistem dapat menampilkan histori data aktual dari setiap komoditas
		Sistem dapat menampilkan grafik visualisasi data historis untuk mempermudah analisa data.

4.2.3 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional meliputi aspek pendukung sistem seperti kebutuhan perangkat lunak (software) dan perangkat keras (hardware), yang tidak terkait langsung dengan fungsi utama namun tetap penting untuk operasional sistem. Berikut kebutuhan non-fungsional dari sistem prediksi harga komoditas pertanian:

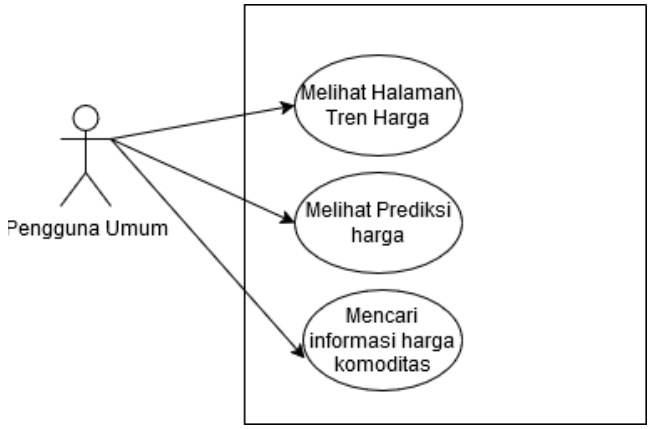
No	Atribut Kualitas	Kebutuhan
1	Kinerja	Waktu response aplikasi cepat, terutama dalam proses menampilkan prediksi harga
2	Kegunaan	Antarmuka pengguna intuitif dan mudah digunakan bagi para petani yang kurang familiar dengan teknologi.
3	Aksebilitas	Sistem dapat diakses dari berbagai perangkat dan *browser* web yang berbeda.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

4.3 Perancangan Sistem

Untuk penggambaran mengenai alur dan tampilan yang akan digunakan untuk pengimplementasia pada website prediksi harga saham ini, dibuat sebuah gambar alur kerja dan perancangan antarmukan website.

4.3.1 Use Case Diagram

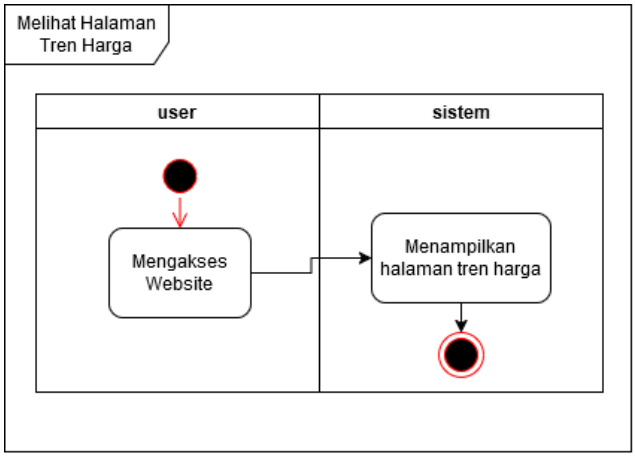


Gambar 3. 2 Use Case Diagram

Berdasarkan Gambar 3.4 di atas, pengguna akan dapat melihat prediksi harga komoditas sebagai fitur utama dari website ini. Selain itu pengguna juga dapat melihat halaman tren sebagai pertimbangan para petani dalam menentukan harga serta mencari informasi histori harga sesungguhnya untuk setiap komoditas.

4.3.2 Activity Diagram

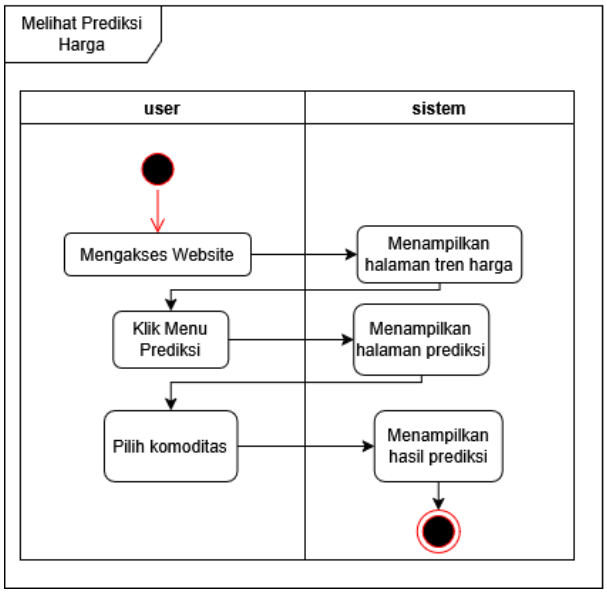
1) Melihat Halaman Tren Harga



Gambar 3. 3 Melihat Halaman Tren Harga

Gambar 3.5 menunjukkan aktifitas *user* ingin mengakses website sistem prediksi dengan memasukkan URl website, lalu sistem akan menampilkan halaman trenharga yang juga merupakan home page dari website.

2) Melihat Prediksi Harga

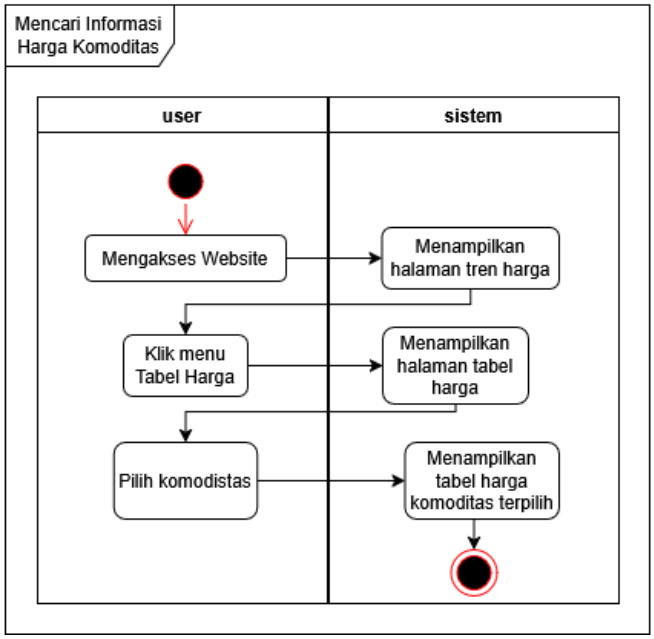


Gambar 3. 4 Melihat Prediksi Harga

Gambar 3.6 menunjukkan aktifitas *user* ingin melihat prediksi harga dengan memilih menu prediksi. *User* diharuskan memilih komoditas yang ingin dilakukan prediksi untuk dapat melihat hasil prediksi harga komoditas yang dipilih.

3) Mencari Informasi Harga Komoditas

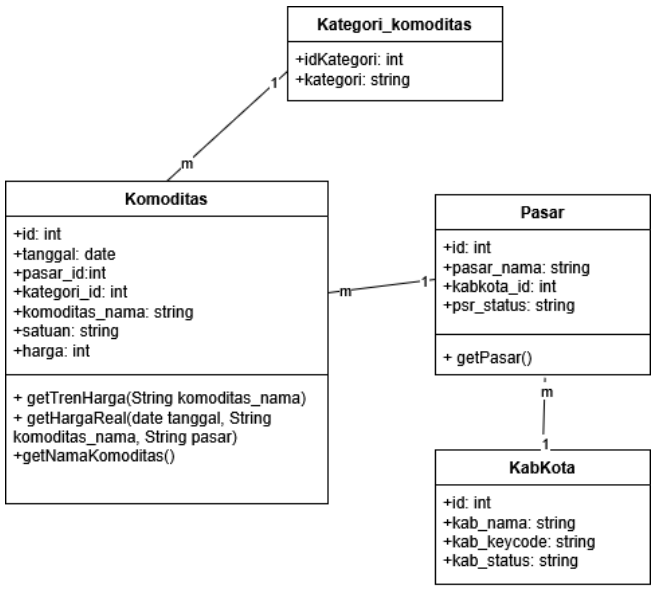
Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.



Gambar 3. 5 Aktifitas mencari informasi harga komoditas

Gambar 3.7 menunjukkan aktifitas *user* ingin melihat histori harga sesungguhnya untuk komoditas. *User* perlu memilih komoditas mana yang ingin ditampilkan data harga historisnya.

4.3.3 Class Diagram



Gambar 3. 6 Class Diagram

Class Diagram pada sistem ini dirancang berdasarkan kebutuhan dari struktur database yang telah ditentukan serta fungsionalitas yang diharapkan dalam use case

diagram. Diagram ini terdiri dari beberapa kelas utama yang berhubungan satu sama lain, yaitu:

1. **Komoditas**
   - Merepresentasikan data komoditas yang mencakup informasi seperti nama komoditas, harga sebelum dan harga saat ini, serta perubahan harga.
   - Memiliki hubungan dengan kelas **Pasar** dan **KategoriKomoditas**.
2. **Pasar**
   - Merepresentasikan informasi mengenai pasar tempat suatu komoditas diperjualbelikan.
   - Berelasi dengan kelas KabKota, yang menunjukkan bahwa suatu pasar berada dalam wilayah tertentu.
3. **KabKota**
   - Merepresentasikan informasi daerah atau kabupaten/kota yang menaungi suatu pasar.
   - Memiliki atribut seperti nama daerah dan status aktif/nonaktif.
4. **KategoriKomoditas**
   - Menyediakan informasi kategori dari suatu komoditas, misalnya bahan pokok, sayuran, atau buah-buahan.

4.3.4 Flowchat

Berdasarakan gambaran flowchart di atas alur kerja website untuk menampilkan hasil prediksi harga komoditas sebagai berikut:

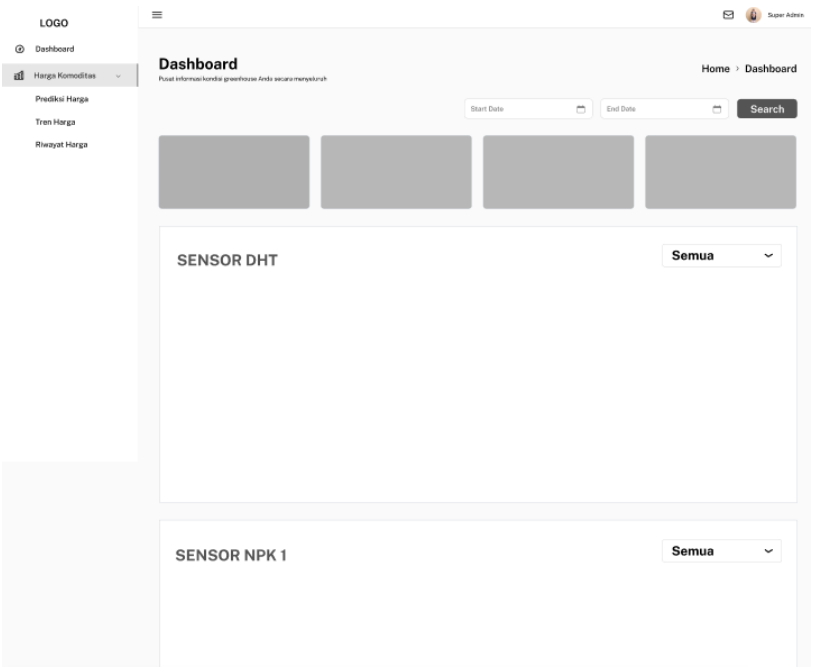
4.3.5 Perancangan Antarmuka

Perancangann antarmuka atau desain sistem merupakan tampilan dasar antarmuka (UI) dari sebuah aplikasi atau *website* yang digunakan untuk mempermudah dalam membangun sistem pada penelitian ini.

a. Tampilan Dashboard

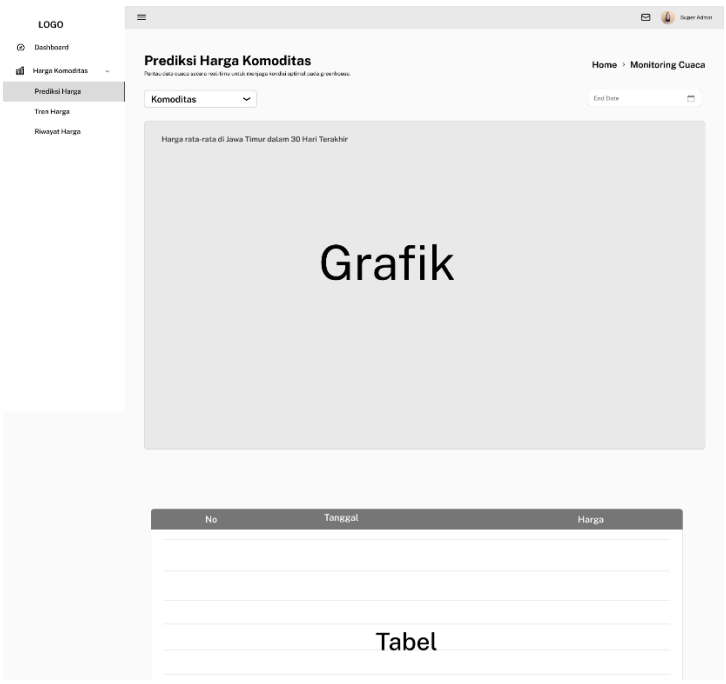
Halaman ini merupakan halaman utama dalam sistem ini, halaman digunakan untuk menampilkan dashboard.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.



b. Tampilan Prediksi

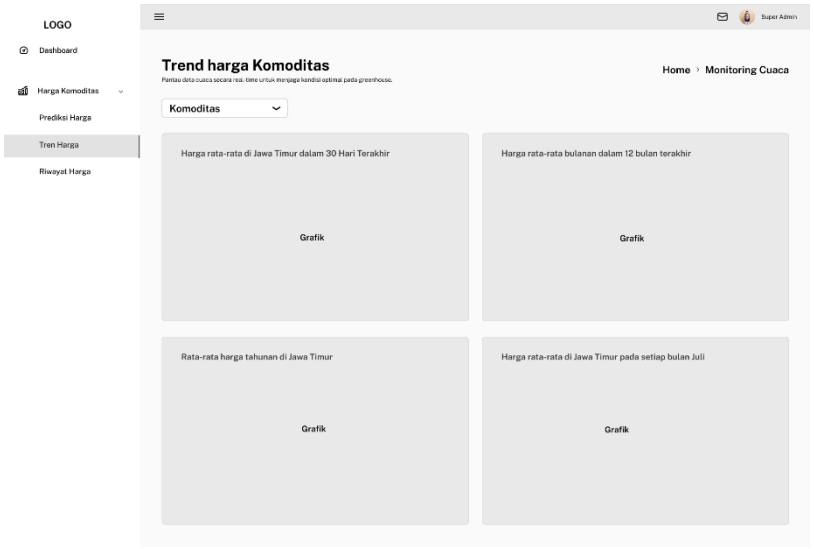
Halaman prediksi merupakan fitur utama untuk menampilkan hasil perhitungan prediksi yang sudah dilakukan pada bagian back-end dari sistem. Pengguna dapat memilih komoditas mana yang ingin ditampilkan dan memilih tanggal kapan data di prediksi.



Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

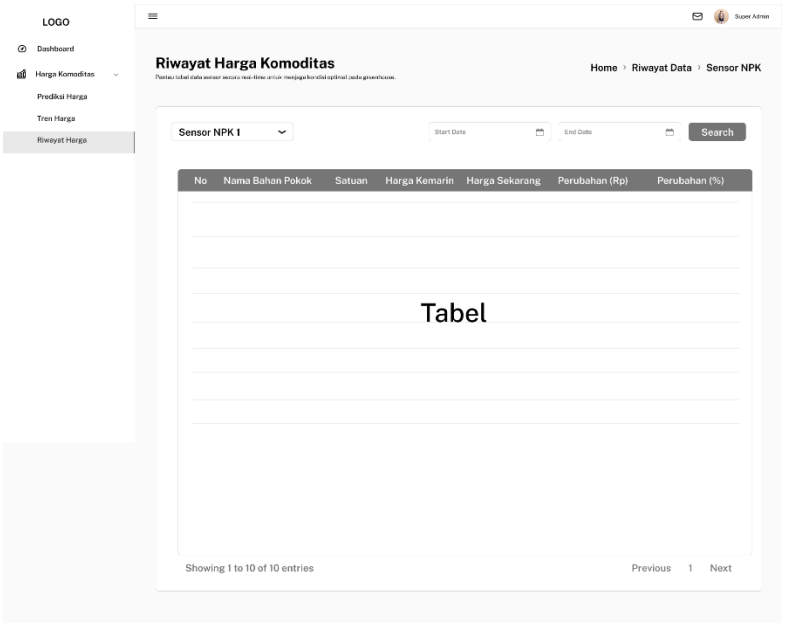
c. Tampilan Tren Harga

Halaman tren harga digunakan untuk menampilkan grafik-grafik analisis yang dapat digunakan oleh para petani menganalisa data sebagai pembantu pengambilan keputusan. Pada halaman ini pengguna dapat memilih komoditas yang ingin ditampilkan data analisisnya.



d. Tampilan Data Aktual

Halaman data aktual digunakan untuk menampilkan data sesungguhnya dan riwayat data sesungguhnya sehingga petani dapat mengetahui data sebenarnya. Pada halaman ini pengguna dapat memilih data komoditas yang ingin ditampilkan beserta filter sesuai data pasar dengan memiliki kabupaten/kota terlebih dahulu.



Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan melalui metode **web scraping** dari situs resmi milik Pemerintah Provinsi Jawa Timur, yaitu Siskaperbapo (Sistem Informasi Ketersediaan dan Perkembangan Harga Bahan Pokok) yang beralamat di https://siskaperbapo.jatimprov.go.id. Situs ini menyediakan data harga komoditas pertanian harian berdasarkan wilayah, pasar, dan jenis komoditas.

Scraping data dilakukan secara otomatis menggunakan bahasa pemrograman Python, dengan memanfaatkan berbagai pustaka (*library*) seperti:

- Request untuk mengirim permintaan HTTP,
- Beautifulsoup untuk mem-parsing HTML,
- Psycopg2 untuk menyimpan data ke dalam database PostgreSQL,
- serta datetime untuk mengelola rentang tanggal scraping.

5.1.1.1 Langkah-langkah Implementasi:

1. **Penentuan Rentang Tanggal**

Sistem mengambil tanggal terakhir yang tersedia di database komoditas_rata-rata sebagai acuan awal (start_date) dan melakukan scraping hingga satu hari sebelum tanggal saat ini (end_date). Rentang tanggal ini dihitung otomatis, sehingga scraping bisa dilakukan secara berkala.

2. **Scraping Data Harga Rata-Rata Komoditas**

Untuk setiap tanggal dalam rentang tersebut, sistem melakukan permintaan data harga komoditas rata-rata melalui metode HTTP POST ke URL https://siskaperbapo.jatimprov.go.id/harga/tabel.nodesign/ . Data yang berhasil diambil meliputi:

   - Tanggal
   - Kategori komoditas

- Nama komoditas
- Satuan
- Harga saat ini

Data ini disimpan ke dalam tabel komoditas_rata-rata di PostgreSQL setelah disesuaikan dengan ID kategori yang relevan dari tabel kategori_komoditas.

3. **Scraping Data Harga Komoditas Berdasarkan Pasar dan Kabupaten/Kota**

Sistem juga mengakses endpoint https://siskaperbapo.jatimprov.go.id/harga/pasar.json/{kabkota} untuk mengambil daftar pasar berdasarkan kabupaten/kota di Jawa Timur. Terdapat lebih dari 35 kabupaten/kota yang datanya diakses. Untuk setiap pasar, dilakukan scraping harga komoditas harian dengan parameter tanggal, kabkota, dan pasar, yang mencakup:

- Nama pasar dan kabupaten
- Nama komoditas
- Harga hari sebelumnya (last price)
- Harga saat ini
- Perubahan harga
- Satuan

Data ini dimasukkan ke dalam tabel komoditas setelah dicocokkan dengan ID kategori dari tabel kategori_komoditas.

4. **Penyimpanan ke Basis Data**

Semua data yang berhasil diambil kemudian disimpan ke dalam database **PostgreSQL** bernama smartfarming pada tabel komoditas_rata-rata dan komoditas. Sebelum disimpan, dilakukan proses pencocokan kategori menggunakan query pencarian berbasis LIKE agar setiap komoditas terasosiasi dengan kategori yang sesuai.

5. **Otomatisasi Proses Scraping**

Seluruh proses dilakukan secara otomatis untuk setiap tanggal, sehingga scraping dapat dijalankan harian tanpa duplikasi data. Proses ini juga memastikan bahwa data historis akan terus bertambah seiring waktu.

5.2 Implementasi Model

Pada bagian ini, akan dijelaskan bagaimana tahapan-tahapan intik mengimplementasikan metode-metode yang digunakan dalam penelitian ini. Pada proses implementasinya model akan menggunakan bahasa pemrograman python dan *libraries* pendukung untuk data komoditas pertanian.

5.2.1 Pre-processing Data

Tahapan preprocessing data dilakukan melalui beberapa langkah berikut:

1. Penanganan Data Hilang (*Missing Value*)

Langkah pertama dalam *preprocessing* adalah melakukan pengecekan terhadap nilai-nilai yang hilang pada *dataset*. Hal ini penting untuk menghindari *error* pada proses pelatihan model dan menjaga keakuratan prediksi. Pengecekan dilakukan menggunakan fungsi `.isnull()` atau `.isna()` dari pustaka `pandas`.

```
print(df.isnull().sum())
```

Jika ditemukan data kosong maka akan dilakukan pengisin data menggunakan fungsi `.interpolate()`

```
def handle_missing_values(df):
    missing_before = df['harga'].isna().sum()
    print(f"📉 Missing value sebelum interpolasi:
{missing_before}")

    # Interpolasi linier berbasis index datetime
    df['harga'] = df['harga'].interpolate(method='time')
    df['harga'] = df['harga'].bfill().ffill()

    missing_after = df['harga'].isna().sum()
    print(f"✅ Missing value sesudah interpolasi:
{missing_after}")
    return df
```

2. Uji Stasioneritas dengan Augmented Dickey-Fuller (ADF Test)

Sebelum melanjutkan ke pemodelan time series, penting untuk mengetahui apakah data bersifat stasioner atau tidak. Data dikatakan stasioner jika nilai statistiknya (rata-rata, varians, dan kovariansi) tidak berubah terhadap waktu. Salah satu cara untuk menguji stasioneritas adalah dengan Augmented Dickey-Fuller (ADF Test). Data dianggap stasioner jika nilai p-value < 0.05. Analisa stasioner hanya digunakan untuk model SARIMA sebagai penentu nilai parameter d.

```
result = adfuller(df['harga'])
print("ADF Statistic:", result[0])
print("p-value:", result[1])
```

3. Analisis Autokorelasi dan Parsial Autokorelasi (ACF dan PACF)

Setelah memastikan data bersifat stasioner, dilakukan analisis ACF (*Autocorrelation Function*) dan PACF (*Partial Autocorrelation Function*) untuk memahami pola korelasi dalam data. Langkah ini penting terutama untuk model SARIMA dalam menentukan parameter `p` dan `q`.

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plot_acf(df['harga'], lags=30)
plot_pacf(df['harga'], lags=30)
```

4. Deteksi dan Penanganan Outlier

Deteksi outlier dilakukan menggunakan visualisasi (boxplot). Jika ditemukan outlier yang signifikan, maka dilakukan transformasi Box-cox untuk menormalkan distribusi data dan mereduksi pengaruh data ekstrem terhadap model.

```
from scipy.stats import boxcox
df['harga'], lambda_ = boxcox(df['harga'] + 1)  # +1 untuk
menghindari log(0)
```

5. Normalisasi Data

Data hasil transformasi selanjutnya dinormalisasi ke dalam rentang [0, 1] menggunakan Min-Max Scaler agar modeldapat memproses input dengan lebih optimal. Tahap Min-Max Scaler ini hanya digunakan pada model LSTM dan GRU.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(0, 1)
df['harga_scaled'] = scaler.fit_transform(df.values)
```

6. Split Data

Data dibagi menjadi dua bagian: data pelatihan (training) dan data pengujian (testing) dengan rasio 80:20. Data pelatihan digunakan untuk membangun model, sedangkan data pengujian digunakan untuk mengevaluasi kinerja model terhadap data baru.

```
def create_dataset(data, look_back=1, steps_ahead=1):
    X, y = [], []
    for i in range(len(data)-look_back-steps_ahead+1):
        X.append(data[i:(i+look_back), 0])
```

```
        y.append(data[(i+look_back):(i+look_back+steps_ahead),
0])
    return np.array(X), np.array(y)

# Parameter
look_back = 90  # Jumlah hari sebelumnya untuk memprediksi
steps_ahead = 90  # Jumlah hari yang akan diprediksi

# Membuat dataset
X, y = create_dataset(scaled_data, look_back, steps_ahead)

# Split data train dan test
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape data untuk LSTM [samples, timesteps, features]
X_train = np.reshape(X_train, (X_train.shape[0],
X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],
1))
```

Pembentukan dataset Time-Series dengan Teknik Sliding Window yang memanfaatkan urutan data historis sepanjang window_size sebagai input untuk memprediksi target di waktu selanjutnya dengan window_size di angka 90 hari sebelum hari prediksi. Selanjutnya data akan dibagi untuk pelatihan dan pengujian dengan rasio 80% pelatihan dan 30% pengujian dengan urutan yang tidak acak untuk menjaga urutan yang dimiliki data karena ini sangat penting untuk prediksi harga time series.

5.2.2 Perancangan Model

Tahap selanjutnya adalah perancangan model dengan deep learning dan model statsitik yang digunakan sebagai objek penelitian kali ini yaitu LSTM, GRU ETS dan SARIMA

5.2.2.1 Model LSTM

1. Perancangan model

```
from keras.callbacks import EarlyStopping
# Model untuk 60 hari prediksi
model = Sequential()

# Lapisan LSTM
model.add(LSTM(units=60, activation='relu',
return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.3))
```

```
model.add(LSTM(units=80, activation='relu',
return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=120, activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(units=steps_ahead))

model.compile(optimizer='adam', loss='mse')
early_stop = EarlyStopping(monitor='val_loss', patience=5)


# Training model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test), verbose=1,
callbacks=[early_stop])
```

Model prediksi harga komoditas pertanian Long Short-Term Memory (LSTM) berlapis dibangun dengan framework Keras (TensorFlow backend). Model ini dirancang untuk memprediksi harga selama 90 hari ke depan (*steps ahead*), menggunakan data historis sebelumnya sebagai input. Arsitektur model terdiri atas empat lapisan LSTM bertingkat dengan jumlah unit yang meningkat secara progresif, yaitu 50, 60, 80, dan 120 unit. Setiap lapisan LSTM diikuti oleh lapisan Dropout dengan nilai dropout bertingkat (0.2, 0.3, 0.4, dan 0.5) untuk mencegah overfitting dan meningkatkan generalisasi model. Lapisan terakhir adalah Dense Layer dengan jumlah neuron sebanyak `steps_ahead`, yang merepresentasikan output prediksi harga selama 90 hari ke depan. Model ini dikompilasi menggunakan optimizer Adam dan fungsi loss Mean Squared Error (MSE) yang umum digunakan untuk regresi.

Selama pelatihan, digunakan teknik Early Stopping dengan parameter `patience=5` yang memantau nilai *validation loss*. Jika model tidak menunjukkan perbaikan dalam 5 epoch berturut-turut pada data validasi, maka pelatihan akan dihentikan secara otomatis untuk menghindari overfitting dan pemborosan sumber daya. Model dilatih selama maksimal 100 epoch dengan batch size 32. Data pelatihan terdiri dari `X_train` sebagai input dan `y_train` sebagai target, sedangkan `X_test` dan `y_test` digunakan sebagai data validasi. Struktur input data mengikuti format 3D `(samples, timesteps, features)` yang sesuai dengan kebutuhan LSTM.

2. Proses prediksi

```
from scipy.special import inv_boxcox
# Prediksi pada data test
test_predict = model.predict(X_test)

# Invers transform untuk mendapatkan nilai asli
test_predict = scaler.inverse_transform(test_predict)
test_predict = inv_boxcox(test_predict, lambda_)-1
y_test = scaler.inverse_transform(y_test)
y_test = inv_boxcox(y_test, lambda_)-1

# Prediksi 30 hari ke depan
last_sequence = scaled_data[-look_back:]  # Ambil 30 hari
terakhir
last_sequence = np.reshape(last_sequence, (1, look_back, 1))
future_predictions = model.predict(last_sequence)
future_predictions =
scaler.inverse_transform(future_predictions)[0]
future_predictions = inv_boxcox(future_predictions, lambda_)-1

# Buat tanggal untuk prediksi
last_date = df.index[-1]
future_dates = pd.date_range(start=last_date +
pd.Timedelta(days=1), periods=steps_ahead)
```

Pada tahap ini, dilakukan proses prediksi menggunakan data input `X_test`, yang menghasilkan sebanyak 90 data prediksi secara langsung. Hasil prediksi tersebut kemudian dilakukan transformasi balik (inverse transform), yaitu dengan menerapkan `inverse_scaler` dan `inverse_boxcox` secara berurutan, sesuai dengan tahapan pre-processing yang telah dilakukan sebelumnya. Tujuannya adalah untuk mengembalikan skala data ke bentuk aslinya agar nilai prediksi dapat dibandingkan secara langsung dengan data aktual. Setelah diperoleh hasil prediksi dalam skala harga sebenarnya, kemudian dibuat rentang tanggal prediksi sebagai sumbu waktu untuk keperluan visualisasi agar hasil grafik lebih informatif dan mudah dipahami.

3. Visualisasi hasil training

```
# Plot training loss
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
```

```
plt.legend()
plt.show()
```

Pada tahap ini menampilkan visualisasi hasil training dengan membandingkan nilai loss pada data training dan validasi selama proses training. Hasil visualisasi digunakan untuk mempermudah dalam menganalisa hasil dari model dan keakuratan model yang sedang dilatih dan membantu mengidentifikasi apakah model mengalami overfitting atau underfitting.

4. Visualisasi hasil prediksi

```
# Plot prediksi vs aktual
plt.figure(figsize=(10, 6))
plt.plot(df_test['harga'], label='Actual Data')
plt.plot(df_forecast['harga_prediksi'],    'r-',    label='90-day
Forecast')
plt.title(f'30-Day Price Forecast for Cabe Rawit Merah')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

Bagian ini menampilkan visualisasi hasil prediksi dengan membandingkan dengan data tes aktual guna mempermudah mengamati sejauh mana perbedaan hasil prediksi dengan data sesungguhnya.

5.2.2.2 Model GRU

1. Perancangan Model

```
from keras.callbacks import EarlyStopping, LearningRateScheduler
from keras import optimizers
import numpy as np
from keras.layers import Bidirectional

# Hyperparameters
learning_rate = 0.0001
batch_size = 256
epochs = 100
steps_ahead = 90  # Pastikan ini sesuai dengan output layer

# --- Arsitektur GRU yang Dioptimasi ---

regressorGRU = Sequential([
    # Layer 1
    GRU(units=256,
        return_sequences=True,
        input_shape=(X_train.shape[1], 1),
        activation='tanh'),
```

```
    Dropout(0.3),

    # Layer 2
    GRU(units=128,
        return_sequences=True,
        activation='tanh'),
    Dropout(0.3),

    # Layer 3
    GRU(units=64,  # Ditambah dari 32 untuk meningkatkan
kapasitas
        return_sequences=False,
        activation='tanh'),
    Dropout(0.2),

    # Output layer
    Dense(steps_ahead)
])
# Compile dengan learning rate custom
optimizer = optimizers.Adam(learning_rate=learning_rate)
regressorGRU.compile(optimizer=optimizer, loss='mse')
```

Pada tahap ini, dilakukan pembangunan model jaringan saraf tiruan dengan arsitektur Gated Recurrent Unit (GRU) yang telah dioptimasi untuk melakukan prediksi harga komoditas. Model dibangun menggunakan tiga lapisan GRU bertingkat untuk menangkap pola sekuensial pada data historis harga. Lapisan pertama terdiri atas 256 unit GRU dengan aktivasi *tanh* dan return_sequences=True, yang memungkinkan keluaran dari setiap langkah waktu diteruskan ke lapisan berikutnya. Setelah itu, diterapkan lapisan Dropout dengan rasio 0.3 untuk mencegah overfitting.

Lapisan kedua terdiri atas 128 unit GRU, juga dengan aktivasi *tanh* dan return_sequences=True, yang masih mempertahankan keluaran sekuensial agar dapat diproses lebih lanjut. Dropout sebesar 0.3 juga diterapkan setelah lapisan ini. Selanjutnya, lapisan ketiga menggunakan 64 unit GRU dengan return_sequences=False, yang berarti hanya keluaran terakhir dari sekuens yang diteruskan ke lapisan berikutnya. Dropout sebesar 0.2 diterapkan untuk menjaga regularisasi.

Lapisan output berupa lapisan *Dense* dengan jumlah neuron sebanyak steps_ahead (dalam hal ini 90 neuron), yang bertujuan untuk menghasilkan prediksi sebanyak 90 langkah ke depan secara langsung dalam satu kali proses inferensi. Model ini dikompilasi menggunakan fungsi loss *Mean Squared Error (MSE)*, dan dioptimasi dengan algoritma Adam menggunakan nilai *learning rate* sebesar

0.0001. Nilai *batch size* ditetapkan sebesar 256, dan proses pelatihan dilakukan selama 100 *epoch*. Parameter-parameter tersebut dipilih melalui eksplorasi bertahap untuk mendapatkan performa terbaik dalam mempelajari pola harga dari data historis.

2. Pelatihan Model

```
# --- Callbacks ---
def lr_scheduler(epoch, lr):
    if epoch % 10 == 0 and epoch != 0:
        return lr * 0.9  # Reduksi learning rate 10% setiap 10
epoch
    return lr

early_stop = EarlyStopping(monitor='val_loss', patience=15,
restore_best_weights=True)
scheduler = LearningRateScheduler(lr_scheduler)
callbacks = [early_stop, scheduler]

# --- Training ---
history = regressorGRU.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=epochs,
    batch_size=batch_size,
    callbacks=callbacks,
    verbose=1
)
```

Proses pelatihan model dilakukan dengan menggunakan data latih `X_train` dan `y_train`, serta divalidasi dengan data `X_test` dan `y_test`. Untuk meningkatkan efisiensi pelatihan, diterapkan dua callback, yaitu Early Stopping dan Learning Rate Scheduler. Callback *Early Stopping* digunakan untuk menghentikan pelatihan jika tidak terjadi penurunan nilai *validation loss* selama 15 epoch berturut-turut, serta mengembalikan bobot model terbaik. Sementara itu, *Learning Rate Scheduler* digunakan untuk menurunkan nilai *learning rate* sebesar 10% setiap 10 epoch, agar proses pembelajaran tetap stabil dan menghindari overshooting. Pelatihan dilakukan selama maksimal 100 epoch dengan *batch size* sebesar 256, dan nilai loss dicatat dalam objek `history` untuk keperluan analisis dan visualisasi performa model.

3. Proses prediksi

```
last_sequence = scaled_data[-90:]  # Ambil 30 hari terakhir
last_sequence = np.reshape(last_sequence, (1, 90, 1))
future_predictions = regressorGRU.predict(last_sequence)
```

```
future_predictions =
scaler.inverse_transform(future_predictions.reshape(-1,
1)).flatten()

df_forecast = pd.DataFrame({
    'tanggal': df_test.index,
    'Actual': df_test['harga'].values,
    'Prediksi_Harga': future_predictions
})
df_forecast.set_index('tanggal', inplace=True)
# Hasil akhir
print("Prediksi 30 Hari ke Depan:")
print(df_forecast)
```

Pada tahap ini, dilakukan proses prediksi untuk 90 hari ke depan menggunakan model GRU yang telah dilatih. Input berupa data sekuens sebanyak 90 hari terakhir dari data historis, yang telah diskalakan sebelumnya. Data input kemudian diubah ke dalam bentuk tiga dimensi sesuai kebutuhan model, yaitu (1, 90, 1). Hasil prediksi kemudian di-reshape dan dikembalikan ke skala aslinya melalui proses inverse transform menggunakan MinMaxScaler.

Hasil prediksi disusun dalam bentuk tabel yang memuat tanggal, nilai aktual, dan hasil prediksi. Tabel ini menjadi dasar untuk mengevaluasi sejauh mana model mampu memproyeksikan harga komoditas pada periode mendatang.

4. Visualisasi hasil prediksi

```
# --- 7. Visualisasi Hasil ---
plt.figure(figsize=(10, 6))
#   plt.plot(df.index,   df['harga'],   label='Data   Historis',
color='blue')
plt.plot(df_test, label='Aktual', color='blue')
plt.plot(df_test.index,future_predictions,   label='Prediksi   30
Hari', color='red', linestyle='--')
plt.title(f'Prediksi Harga Cabai rawit Merah 30 Hari ke Depan (GRU
Model)')
plt.xlabel('Tanggal')
plt.ylabel('Harga')
plt.legend()
plt.show()
```

Bagian ini menampilkan visualisasi hasil prediksi dengan membandingkan dengan data tes aktual guna mempermudah mengamati sejauh mana perbedaan hasil prediksi dengan data sesungguhnya.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

5.2.2.3 Model Exponential Smoothing

1. Pencarian hiper parameter terbaik

```
# Fungsi optimasi untuk data harian
def daily_optimizer(train, test, seasonal_period=7,
max_alpha=0.99, max_beta=0.99, max_gamma=0.99):
    # Generate parameter space
    alphas = np.round(np.arange(0.1, max_alpha, 0.1), 2)
    betas = np.round(np.arange(0.1, max_beta, 0.1), 2)
    gammas = np.round(np.arange(0.1, max_gamma, 0.1), 2)

    abg_combinations = list(itertools.product(alphas, betas,
gammas))

    best_params = {
        'alpha': None,
        'beta': None,
        'gamma': None,
        'mae': float('inf')
    }

    for alpha, beta, gamma in abg_combinations:
        try:
            # Model untuk data harian dengan seasonal period 7
(mingguan)
            model = ExponentialSmoothing(
                train,
                trend='add',
                seasonal='add',
                seasonal_periods=seasonal_period,  # 7 untuk
pola mingguan
                initialization_method='estimated'
            ).fit(
                smoothing_level=alpha,
                smoothing_trend=beta,
                smoothing_seasonal=gamma
            )

            # Forecast
            forecast = model.forecast(len(test))

            # Evaluasi
            current_mae = mean_absolute_error(test, forecast)

            # Update best params
            if current_mae < best_params['mae']:
                best_params.update({
                    'alpha': alpha,
                    'beta': beta,
                    'gamma': gamma,
                    'mae': current_mae
                })

            print(f"Alpha: {alpha:.2f} | Beta: {beta:.2f} |
Gamma: {gamma:.2f} | MAE: {current_mae:.2f}")

        except Exception as e:
```

```
            print(f"Error with alpha={alpha}, beta={beta},
gamma={gamma}: {str(e)}")
            continue

    print("\nBest Parameters Found:")
    print(f"Alpha: {best_params['alpha']:.2f}")
    print(f"Beta: {best_params['beta']:.2f}")
    print(f"Gamma: {best_params['gamma']:.2f}")
    print(f"Best MAE: {best_params['mae']:.4f}")

    return best_params

# Contoh penggunaan
train_data = data_train  # Data training harian
test_data = data_test   # Data testing harian

# Optimasi untuk pola mingguan (7 hari)
best_params = daily_optimizer(
    train=train_data,
    test=test_data,
    seasonal_period=30,  # Ubah menjadi 30 untuk pola bulanan
    max_alpha=0.99,     # Batas maksimal alpha
    max_beta=0.5,       # Batas lebih konservatif untuk beta
    max_gamma=0.5       # Batas lebih konservatif untuk gamma
)
```

Fungsi `daily_optimizer()` digunakan untuk mencari kombinasi hyperparameter terbaik (`alpha`, `beta`, dan `gamma`) pada model Exponential Smoothing yang diterapkan pada data time series harian. Fungsi ini melakukan pencarian grid (grid search) terhadap berbagai kombinasi parameter smoothing dalam ruang nilai yang telah ditentukan. Model dievaluasi menggunakan Mean Absolute Error (MAE) pada data pengujian (test set) untuk setiap kombinasi parameter, dan parameter dengan nilai MAE terkecil akan dipilih sebagai konfigurasi terbaik. Dalam implementasi ini, seasonal period ditetapkan sebanyak 30 hari untuk menangkap pola musiman bulanan, dan batas maksimum nilai `beta` serta `gamma` dibatasi lebih konservatif untuk menjaga kestabilan model.

2. Pelatihan model

```
final_model = ExponentialSmoothing(data['harga'], trend="add",
seasonal="add", seasonal_periods=30).\
    fit(smoothing_level=best_params['alpha'],
smoothing_trend=best_params['beta'],
smoothing_seasonal=best_params['gamma'])
```

Pelatihan model dilakukan untuk menghasilkan prediksi harga dan mengevaluasi tingkat akurasi model terhadap data aktual. Pada tahap ini, digunakan metode Exponential Smoothing dengan konfigurasi tren dan musiman bertipe aditif (menyesuaikan komoditas). Nilai hyperparameter `alpha`, `beta`, dan `gamma` yang diperoleh dari proses optimasi sebelumnya digunakan masing-masing sebagai `smoothing_level`, `smoothing_trend`, dan `smoothing_seasonal`. Dengan penggunaan hyperparameter yang telah dioptimalkan, diharapkan kestabilan dan performa model dalam mempelajari pola data dapat ditingkatkan. Selain itu, setiap komoditas dapat memiliki kombinasi parameter yang berbeda sesuai dengan karakteristik pola harga masing-masing.

3. Tahap Prediksi

```
forecast_predictions =
final_model.forecast(steps=len(data_test))
```

Pada tahap ini, dilakukan proses prediksi terhadap data menggunakan model yang telah dilatih sebelumnya. Prediksi dilakukan sebanyak jumlah data pada *test set* menggunakan fungsi `forecast()`, sehingga diperoleh nilai-nilai harga yang diperkirakan untuk periode waktu mendatang.

4. Visualisasi hasil prediksi

```
mae = mean_absolute_error(data_test['harga'],
forecast_predictions)
# plt.figure(figsize=(10, 6))
data_train['harga'].plot(label="Data Test",color='green')
data_test['harga'].plot(label="Data Real",color='blue')
forecast_predictions.plot(label="PREDICTION",color='red',
linestyle='--')

# Tambahkan judul dengan nilai MAE
plt.title(f"Triple Exponential Smoothing - MAE: {round(mae,
2)}")
plt.xlabel("Tanggal")
plt.ylabel("Harga")
plt.grid()
plt.show()
```

Visualisasi dilakukan untuk membandingkan antara data aktual dan hasil prediksi model. Garis prediksi ditampilkan dengan warna merah putus-putus, sedangkan data aktual ditampilkan dengan warna biru. Nilai Mean Absolute Error (MAE) juga dihitung dan ditampilkan pada judul grafik sebagai indikator akurasi

model. Visualisasi ini membantu dalam mengevaluasi sejauh mana model dapat mengikuti pola data aktual secara visual.

5.2.2.4 Model SARIMA

1. Pemodelan SARIMA

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
import warnings

warnings.filterwarnings("ignore")

model = SARIMAX(data,
                order=(2, 0, 2),
                seasonal_order=(1, 0, 1, 30),
                enforce_stationarity=False,
                enforce_invertibility=False)

results = model.fit()

# Melihat ringkasan model
print(results.summary())

# Plot hasil prediksi
results.plot_diagnostics(figsize=(15, 8))
plt.show()
```

Pada tahap ini dilakukan pemodelan menggunakan metode Seasonal ARIMA (SARIMA). Parameter order dan seasonal_order mencerminkan pola musiman bulanan. Pemodelan dilakukan menggunakan library `statsmodels` dengan opsi `enforce_stationarity` dan `enforce_invertibility` diset ke `False` untuk memberikan fleksibilitas pada proses estimasi. Setelah model dilatih, ditampilkan ringkasan parameter hasil estimasi serta grafik diagnostik residual untuk mengevaluasi asumsi dan kecocokan model terhadap data.

2. Percobaan Prediksi

```
forecast = results.get_forecast(steps=len(data_test))

forecast_ci = forecast.conf_int()

ax = data.plot(label='Observed', figsize=(16, 6))
forecast.predicted_mean.plot(ax=ax, label='Forecast',
color='green')
data_test.plot(ax=ax, label='Test', color='red')
ax.fill_between(forecast_ci.index,
                forecast_ci.iloc[:, 0],
                forecast_ci.iloc[:, 1], color='green', alpha=.2)
ax.set_xlabel('Tanggal')
```

```
ax.set_ylabel('Harga')
plt.legend()
plt.show()
```

Setelah model SARIMA dilatih, dilakukan percobaan prediksi sepanjang periode data uji. Prediksi dihasilkan menggunakan metode `get_forecast()` dan disertai dengan interval kepercayaan untuk memberikan gambaran ketidakpastian prediksi. Visualisasi menunjukkan data aktual, hasil prediksi, serta area interval kepercayaan 95%. Grafik ini membantu dalam mengevaluasi kemampuan model dalam mengikuti pola data aktual dan menunjukkan sejauh mana prediksi berada dalam rentang yang dapat diterima.

5.2.3 Evaluasi Model

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, mean_absolute_percentage_error
# Hitung RMSE
rmse = np.sqrt(mean_squared_error(y_test, test_predict))
print(f'RMSE: {rmse}')

# Hitung MAE
mae = mean_absolute_error(y_test, test_predict)
print(f'MAE: {mae}')

# Hitung MAPE
mape = mean_absolute_percentage_error(y_test, test_predict)* 100
print(f'MAPE: {mape}%')
```

Evaluasi model dilakukan dengan memprediksi nilai pada data uji menggunakan model yang sudah dilatih sebelumnya, pada model ini menghasilkan dua output yaitu prediksi harga saham (y_test) dan bobot attention (attention_weights_output). Nilai aktual harga saham pada data uji dikembalikan ke skala aslinya dengan menggunakan inverse transform dari scaler, yang dilakukan dengan menggabungkan fitur terakhir dari data uji dengan nilai target asli, lalu mengambil kolom harga close. Prediksi model juga 78 dikembalikan ke skala asli dengan cara yang serupa. Setelah itu, performa model diukur menggunakan 3 matriks evaluasi, yaitu Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), dan Mean Absolute Percentage Error (MAPE) yang memudahkan interpretasi dalam konteks bisnis atau investasi. Hasil evaluasi ini memberikan

gambaran kuantitatif tentang akurasi model dalam memprediksi harga saham berdasarkan data historis.

5.2.4 Penyimpana Model dan Objek Scaler

```
from tensorflow.keras.models import save_model
from joblib import dump
save_model(model, 'model_bwgPth.keras')  # Model LSTM
dump(scaler, 'scaler_bwgPth.joblib')
```

Berdasarkan hasil analisa dari evaluasi setiap model yang telah dilakukan sebelumnya akan mendapatkan model dengan akurasi terbaik pada setiap komoditasnya. Apabila LSTM/GRU menjadi model terbaik pada komoditas maka hasil pelatihan model dan scaler yang digunakan dalam preprocessing akan disimpan, agar model dapat digunakan kembali tanpa perlu melatih ulang dari awal dan menjaga skala input data stabil sehingga dapat menjaga keakuratan hasil prediksi nantinya.

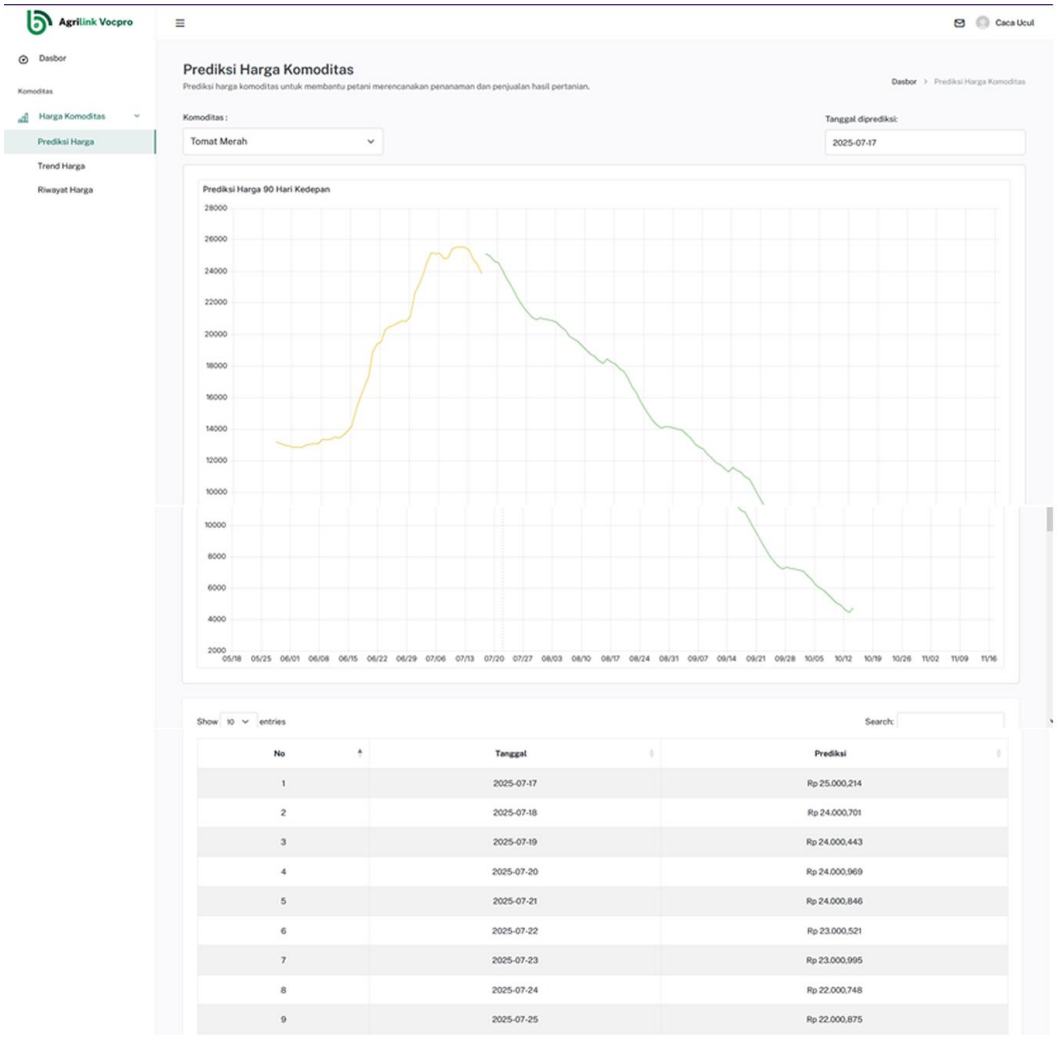
5.3 Implementasi Backend

Pada implementasi backend ini akan dijelaskan terkait penerapan model hasil pelatihan dan pengujian yang sudah dibuat pada sebuah website prediksi harga komoditas pertanian.

5.4 Implementasi Frontend

5.4.1 Halaman Prediksi

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

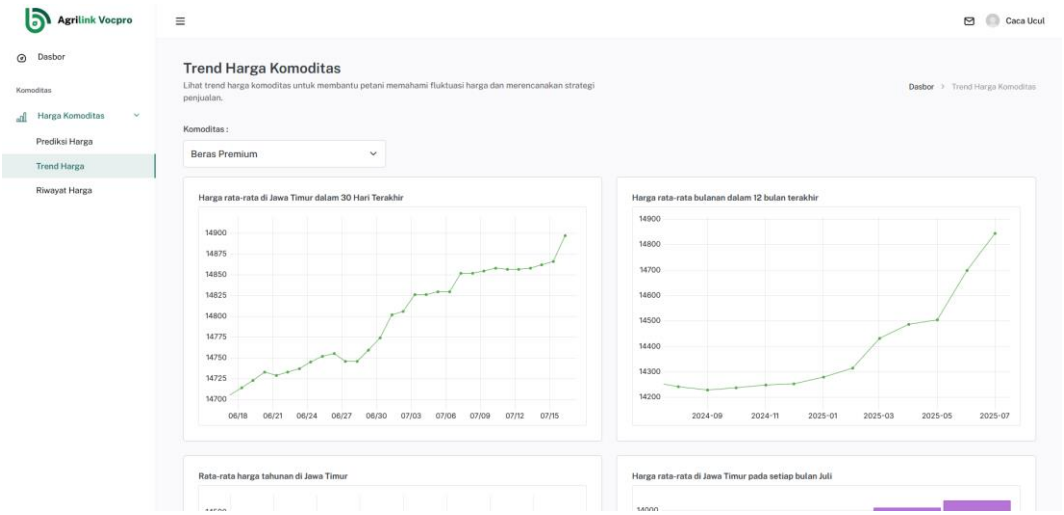


Gambar 5. 1 Halaman Prediksi Harga

Pada halaman ini pengguna dapat melihat prediksi harga dari komoditas pertanian yang dipilih. Pengguna dapat memilih waktu prediksi dilakukan dan hasil prediksi pada saat itu. Selain itu pengguna dapat memilih komoditas mana yang ingin ditampilkan hasil prediksinya. Pada bagian dibawah grafik terdapat tabel yang memperjelas nilai hasil prediksi.

5.4.2 Halaman Trend Harga

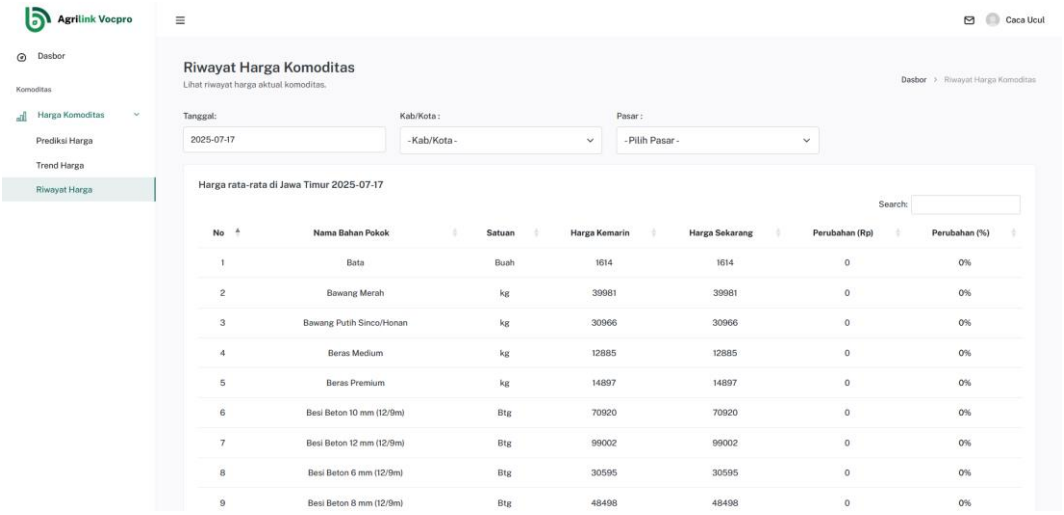
Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.



Gambar 5. 2 Halaman Trend Harga

Pada halaman trend harga pengguna dapat melihat berbagai tampilan grafik yang dapat mempermudah analisis harga komoditas pertanian yang ada sehingga dapat membantu mengguna dalam mengambil keputusan. Pada halaman ini pengguna dapat memilihkomoditas mana yang ingin ditampilkan datanya.

5.4.3 Halaman Riwayat Harga Komoditas



Gambar 5. 3 Halaman Riwayat Harga Komoditas

Pada halaman riwayat ini menampilkan harga aktual dari semua komoditas. Pengguna dapat memilih tanggal untuk menampilkan harga jual sebenarnya pada saat itu. Tidak hanya itu, pengguna juga dapat melihat lebih spesifik harga dari setiap pasar yang tersedia dengan memilihi wilayah dan nama pasar yang ingin ditampilkan.

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

5.5 Pengujian

5.5.1 Pengujian Perbandingan Akurasi Model

Contoh Kata Pengantar
Tidak harus sama persis. Boleh diubahsuaikan dengan kebutuhan.

BAB VI. HASIL DAN PEMBAHASAN