BAB IV
HASIL DAN PEMBAHASAN

4.1 Pemahaman Bisnis

Penelitian ini bertujuan untuk memprediksi konsumsi kalori harian per kapita dari berbagai komoditas pangan di Indonesia berdasarkan data Neraca Bahan Makanan (NBM). Prediksi konsumsi kalori menjadi aspek krusial dalam perencanaan ketahanan pangan nasional, evaluasi pola konsumsi masyarakat, dan proyeksi kebutuhan distribusi pangan di masa mendatang.

Ruang lingkup penelitian mencakup analisis 114 komoditas pangan dengan periode pengamatan dari tahun 1993 hingga 2024. Fokus utama penelitian adalah mengidentifikasi komoditas yang memberikan kontribusi terbesar terhadap asupan kalori masyarakat Indonesia serta memprediksi tren konsumsi pangan untuk periode mendatang. Hasil prediksi diharapkan dapat memberikan informasi strategis bagi pemangku kebijakan dalam merancang program ketahanan pangan yang lebih efektif.

4.2 Pemahaman Data

Dataset transaksi Neraca Bahan Makanan (NBM) yang digunakan dalam penelitian ini mencakup 48.696 *records* dengan 41 kolom yang merepresentasikan berbagai aspek produksi, distribusi, dan konsumsi pangan di Indonesia dari tahun 1993 hingga 2024. Data mencakup 112 komoditas pangan yang dikelompokkan dalam 10 kategori utama meliputi Padi-Padian, Makanan Berpati, Gula, Buah/Biji Berminyak, Buah-buahan, Sayur-sayuran, Daging, Telur, Susu, dan Minyak dan Lemak.

Tabel 4 menampilkan sampel data mentah dengan 5 baris pertama dan 2 baris terakhir untuk memberikan gambaran struktur dan rentang temporal *dataset*.

Tabel 4. Sampel Data Mentah Transaksi NBM

id	kode_kelompok	kode_komoditi	tahun	bulan	bahan_makanan
1	01	0101	1993	1	0.00
2	01	0101	1993	2	0.00

id	kode_kelompok	kode_komoditi	tahun	bulan	bahan_makanan
3	01	0101	1993	3	0.00
4	01	0101	1993	4	0.00
5	01	0101	1993	5	0.00
…	…	…	…	…	…
48695	10	1009	2024	11	769.60
48696	10	1009	2024	12	824.60

Berdasarkan observasi data mentah pada Tabel 4, ditemukan bahwa komoditas Gabah (kode 0101) dari kelompok Padi-Padian yang muncul pada 5 baris pertama memiliki nilai bahan_makanan sebesar 0.00 (nol) pada seluruh periode pengamatan. Hal ini bukan merupakan *missing value* atau kesalahan data, melainkan karakteristik *inherent* dari komoditas tersebut dalam konteks Neraca Bahan Makanan.

Gabah merupakan produk pertanian primer yang tidak dikonsumsi langsung oleh masyarakat, melainkan harus diolah terlebih dahulu menjadi beras (kode 0102) sebelum dapat dikonsumsi sebagai bahan pangan. Dalam sistem NBM, konsumsi kalori dihitung dari produk akhir yang siap konsumsi bukan dari bahan baku mentah. Oleh karena itu, seluruh konsumsi gabah tercatat sebagai nol karena telah dikonversi dan dialokasikan ke komoditas beras melalui proses penggilingan dengan *conversion factor* sekitar 63-65%. Komoditas sejenis yang juga di-*exclude* dengan alasan sama adalah Kacang Tanah Berkulit (kode 0401) dari kelompok Buah/Biji Berminyak, yang merupakan bahan baku mentah yang harus diolah terlebih dahulu sebelum dapat dikonsumsi. Sama seperti Gabah, seluruh 468 transaksi komoditas ini tercatat bahan_makanan = 0.00 karena konsumsinya dialokasikan sepenuhnya ke produk turunannya.

Analisis kualitas data menunjukkan beberapa fitur mengalami *missing values*. Pada dataset transaksi, fitur *inflasi_komoditi*, *nilai_tukar_usd*, *gdp_per_kapita*, *tingkat_kemiskinan*, dan *indeks_el_nino* mengalami kehilangan data secara menyeluruh dengan 48.696 nilai kosong. Fitur *harga_produsen*, *harga_konsumen*, *curah_hujan_mm*, dan *suhu_rata_celsius* memiliki 1.200 nilai kosong, sedangkan

luas_panen_ha dan *produktivitas_ton_ha* kehilangan 22.596 nilai. Tabel 5 merangkum kondisi *missing values* pada dataset transaksi.

Tabel 5. Distribusi *Missing Values* Dataset Transaksi

Fitur	Jumlah *Missing*	Persentase
inflasi_komoditi	48.696	100%
nilai_tukar_usd	48.696	100%
gdp_per_kapita	48.696	100%
tingkat_kemiskinan	48.696	100%
indeks_el_nino	48.696	100%
luas_panen_ha	22.596	46,4%
produktivitas_ton_ha	22.596	46,4%
harga_produsen	1.200	2,5%
harga_konsumen	1.200	2,5%
curah_hujan_mm	1.200	2,5%
suhu_rata_celsius	1.200	2,5%

Fitur-fitur dengan *missing values* 100% tidak dapat dimanfaatkan dalam pemodelan karena tidak memberikan informasi apapun. Fitur dengan kehilangan data parsial seperti *luas_panen_ha* dan *produktivitas_ton_ha* juga tidak diikutsertakan karena tingkat kehilangan yang signifikan dapat mempengaruhi kualitas model. Fitur dengan kehilangan minimal seperti *harga_konsumen* dan variabel cuaca tetap dipertahankan dengan penanganan melalui imputasi.

Variabel target utama dalam penelitian adalah *kalori_per_kapita_per_hari* yang dihitung dari fitur *bahan_makanan* (dalam satuan ton). Fitur pendukung meliputi informasi *time series* periode 1993-2024 dengan tingkat rincian bulanan, karakteristik gizi komoditas (*kalori_per_100g*, *protein_per_100g*, *lemak_per_100g*, *karbohidrat_per_100g*), serta faktor eksternal berupa data populasi, harga konsumen, curah hujan, dan suhu rata-rata.

4.3 Persiapan Data

4.3.1 Data *Cleaning* dan *Preprocessing*

Tahap persiapan data dimulai dengan seleksi *records* berdasarkan periode data. Dataset awal memuat dua jenis periode yaitu bulanan dan kuartalan. Analisis distribusi menunjukkan data bulanan mendominasi dengan 47.820 *records* atau 98,2% dari total data, sedangkan data kuartalan hanya 876 *records* atau 1,8%. Perbedaan interval temporal antara data bulanan (1 bulan) dan kuartalan (3 bulan) berpotensi menciptakan ketidakkonsistenan dalam analisis *time series*. Model LSTM yang digunakan membutuhkan interval waktu yang konsisten untuk menangkap pola temporal dengan optimal. Berdasarkan pertimbangan tersebut, penelitian hanya menggunakan data dengan *periode_data* bernilai bulanan.

Filter tambahan diterapkan pada fitur *bahan_makanan* dengan mengeliminasi *records* yang memiliki nilai kosong atau bernilai nol. Nilai nol pada konsumsi pangan mengindikasikan tidak adanya aktivitas konsumsi yang dapat mengacaukan perhitungan kalori. Setelah proses filter, dataset transaksi tereduksi menjadi 47.820 *records* dengan 9 kolom terpilih yang relevan untuk analisis.

Proses penggabungan (*merge*) dilakukan antara dataset transaksi dan komoditi menggunakan kunci *kode_komoditi*. Hasil penggabungan menghasilkan dataset gabungan dengan dimensi 47.820 baris dan 14 kolom yang memuat informasi lengkap transaksi beserta nilai gizi setiap komoditas. Tabel 6 menampilkan struktur fitur terpilih setelah proses filter dan *merge*.

Tabel 6. Fitur Terpilih Setelah Filter dan *Merge*

Kategori	Fitur
Identifikasi	kode_komoditi, nama
Temporal	tahun, bulan
Target	bahan_makanan
Demografis	populasi_indonesia
Ekonomi	harga_konsumen, inflasi_komoditi

Kategori	Fitur
Lingkungan	curah_hujan_mm, suhu_rata_celsius
Nutrisi	kalori_per_100g, protein_per_100g, lemak_per_100g, karbohidrat_per_100g

4.3.2 Perhitungan Kalori per Kapita

Perhitungan kalori per kapita per hari menggunakan formula konversi sebagaimana dijelaskan pada Persamaan (1), yang mempertimbangkan volume konsumsi, kandungan kalori komoditas, jumlah populasi, dan durasi periode. Derivasi formula dimulai dengan konversi satuan konsumsi dari ton menjadi gram. Sebagai ilustrasi, konsumsi beras pada Januari 1993 tercatat 19.039,2561 ton. Konversi ke satuan gram menghasilkan 19.039.256.100.000 gram melalui perkalian dengan faktor 10^9. Tahap berikutnya menghitung total kalori dengan mengalikan massa dalam gram dengan kandungan kalori per 100 gram. Beras memiliki kandungan 360 kalori per 100 gram, sehingga total kalori adalah 19.039.256.100.000 × (360/100) = 68.541.321.960.000 kalori.

Pembagian dengan total konsumsi orang-hari menghasilkan nilai kalori per kapita per hari. Pada Januari 1993, populasi Indonesia tercatat 187.000.000 jiwa dengan durasi 31 hari, menghasilkan 5.797.000.000 orang-hari. Kalori per kapita per hari untuk beras adalah 68.541.321.960.000 / 5.797.000.000 = 11.823,58 kalori per hari. Tabel 7 menampilkan hasil perhitungan kalori untuk lima *records* pertama.

Tabel 7. Hasil Perhitungan Kalori per Kapita per Hari

Nama	Tanggal	Kalori per Kapita per Hari
Beras	1993-01-01	11.823,58
Beras	1993-02-01	14.848,38
Beras	1993-03-01	14.330,03
Beras	1993-04-01	14.305,97
Beras	1993-05-01	12.414,11

Rentang waktu dataset mencakup periode dari 1 Januari 1993 hingga 1 Desember 2024, memberikan cakupan temporal selama 31,92 tahun atau 383 bulan. Cakupan temporal yang luas memungkinkan model menangkap berbagai pola

musiman, tren jangka panjang, serta dampak peristiwa ekonomi seperti krisis 1998 dan pandemi COVID-19.

4.3.3 Penanganan *Outlier*

Tahap penanganan *outlier* diawali dengan eliminasi data yang secara fisik tidak masuk akal. *Records* yang menghasilkan nilai kalori_per_kapita_per_hari negatif atau bernilai nol setelah perhitungan konversi akibat *error* pencatatan, nilai bahan_makanan yang tidak valid, atau kombinasi parameter yang tidak realistis dihapus dari *dataset*. Secara fisik, konsumsi kalori tidak mungkin bernilai negatif karena merepresentasikan asupan energi. Demikian pula nilai kalori yang ekstrem tinggi (misalnya melebihi 50.000 kkal/kapita/hari) yang tidak sesuai dengan batas biologis konsumsi manusia dieliminasi sebagai data tidak masuk akal. Eliminasi ini dilakukan sebelum deteksi *outlier* dengan metode IQR agar perhitungan kuartil dan batas IQR tidak terdistorsi oleh nilai-nilai *invalid*.

Deteksi *outlier* menggunakan metode *Interquartile Range* (IQR) dengan threshold yang dimodifikasi sebagaimana dijelaskan pada Persamaan (16). Metode IQR standar menggunakan batas 1,5 × IQR yang cenderung terlalu agresif untuk data konsumsi pangan yang memiliki variasi musiman tinggi. Penelitian ini menerapkan threshold 3 × IQR untuk mengakomodasi fluktuasi *seasonal* yang merupakan pola alamiah bukan anomali.

Implementasi deteksi *outlier* dilakukan per komoditas untuk mempertahankan karakteristik konsumsi spesifik setiap jenis pangan. Sebagai ilustrasi, untuk komoditas beras dengan kode 0102, nilai kuartil pertama (Q_1) adalah 20.500 ton dan kuartil ketiga (Q_3) adalah 28.000 ton. Nilai IQR dihitung sebagai 28.000 - 20.500 = 7.500 ton. Batas bawah adalah 20.500 - 3 × 7.500 = -2.000 ton, sedangkan batas atas adalah 28.000 + 3

× 7.500 = 50.500 ton. Data yang berada di luar rentang [-2.000, 50.500] dikategorikan sebagai *outlier* ekstrem.

Perbandingan dengan threshold standar 1,5 × IQR menunjukkan perbedaan signifikan dalam jumlah data yang dieliminasi. *Threshold* 1,5 × IQR menghasilkan batas bawah 9.250 ton dan batas atas 39.250 ton untuk beras, mengeliminasi sekitar 50 *records* yang sebagian besar merupakan lonjakan konsumsi saat Ramadan atau Lebaran. *Threshold* 3 × IQR hanya mengeliminasi 5 *records* yang benar-benar ekstrem, mempertahankan pola seasonal yang valid. Tabel 8 membandingkan dampak kedua threshold terhadap eliminasi data.

Tabel 8. Perbandingan *Threshold* IQR terhadap Eliminasi Data

Threshold	**Batas Bawah**	**Batas Atas**	***Records* Tereliminasi**	**Persentase**
1,5 × IQR	9.250	39.250	50	13,0%
3 × IQR	-2.000	50.500	5	1,3%

Penanganan *missing values* pada fitur numerik dilakukan melalui imputasi menggunakan nilai median. Median dipilih sebagai ukuran tendensi sentral karena lebih robust terhadap *outlier* dibandingkan mean. Fitur *harga_konsumen*, *curah_hujan_mm*, dan *suhu_rata_celsius* yang memiliki 1.200 nilai kosong diisi dengan nilai median masing-masing fitur. Setelah imputasi, seluruh dataset memiliki 47.820 *records* tanpa nilai kosong.

Proses eliminasi *outlier* dengan threshold 3 × IQR mengurangi dataset menjadi 47.423 *records*. Jumlah data yang dieliminasi adalah 397 *records* atau 0,83% dari total data setelah filter. Persentase eliminasi yang rendah mengindikasikan threshold yang dipilih efektif dalam mempertahankan data valid sambil mengeliminasi anomali ekstrem. *Listing* program 1 memvisualisasikan distribusi data sebelum dan setelah eliminasi *outlier*.

```
numeric_cols = df.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
```

```
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].median(), inplace=True)

Q1 = df.groupby('kode_komoditi')['bahan_makanan'].transform('quantile',
0.25)
Q3 = df.groupby('kode_komoditi')['bahan_makanan'].transform('quantile',
0.75)
IQR = Q3 - Q1
lower = Q1 - 3 * IQR
upper = Q3 + 3 * IQR
df = df[(df['bahan_makanan'] >= lower) & (df['bahan_makanan'] <= upper)]
```

***Listing* Program 1.** Implementasi Imputasi dan Eliminasi *Outlier*

Dataset final terdiri dari 17 kolom dengan rincian, 2 kolom identifikasi (*kode_komoditi*, *nama*), 1 kolom temporal (*date*), 4 kolom turunan temporal (*tahun*, *bulan*, *jumlah_hari*), 1 kolom target (*kalori_per_kapita_per_hari*), 4 kolom numerik utama (*bahan_makanan*, *populasi_indonesia*, *harga_konsumen*, *kalori_per_100g*), dan 5 kolom fitur tambahan (*inflasi_komoditi*, *curah_hujan_mm*, *suhu_rata_celsius*, *protein_per_100g*, *lemak_per_100g*, *karbohidrat_per_100g*). Dataset mencakup 112 komoditas unik dengan total 47.423 observasi yang siap untuk tahap *feature engineering* dan pemodelan.

4.4 Pemodelan

4.4.1 *Feature Engineering*

Penghapusan data pada tahap *preprocessing* (eliminasi data tidak valid dan *outlier*) mengubah urutan temporal *dataset*. *Lag features* bergantung pada nilai periode sebelumnya melalui operasi *shift*; jika baris di tengah deret dihapus, referensi "periode sebelumnya" untuk baris berikutnya berubah. Demikian pula *rolling window features* dihitung dari agregat baris dalam jendela waktu; komposisi jendela berubah ketika baris di dalamnya tereliminasi. Oleh karena itu, *feature engineering* diterapkan *setelah* tahap eliminasi *outlier* selesai. *Dataset* yang memasuki tahap ini adalah data final yang telah dibersihkan (47.423 *records*). *Lag features*, *rolling window features*, dan *cyclical encoding* dihitung pada urutan temporal yang sudah final, sehingga nilai *lag*-1 untuk

observasi bulan t selalu mengacu pada observasi bulan t−1 dalam deret yang sama (tanpa baris yang telah dihapus), dan *rolling statistics* merefleksikan agregat pada data yang benar. *Cyclical encoding* untuk bulan tidak terpengaruh oleh penghapusan karena hanya bergantung pada nilai bulan (1–12). Tidak diperlukan penghitungan ulang karena seluruh fitur turunan dihitung satu kali pada *dataset* final. Tabel 9 merangkum urutan *pipeline preprocessing* hingga *feature engineering*.

Tabel 9. Urutan *Pipeline Preprocessing* dan *Feature Engineering*

Tahap	Output	Keterangan
Data *Cleaning*	Filter periode bulanan, bahan_makanan > 0	47.820 *records*
Perhitungan Kalori	kalori_per_kapita_per_hari	Konversi dari bahan_makanan
Eliminasi Data Tidak Valid	Hapus kalori negatif	Sebelum IQR
Eliminasi *Outlier* (3×IQR)	Dataset bersih	47.423 *records*
Feature Engineering	*Lag, Rolling, Cyclical*	Dihitung pada data final

Feature engineering dilakukan untuk memperkaya informasi temporal dan meningkatkan kemampuan model dalam menangkap pola konsumsi. Tiga kategori fitur ditambahkan: *lag features*, *rolling window features*, dan *cyclical encoding*.

1. *Lag Features*

   Lag features menangkap ketergantungan temporal dengan memasukkan nilai konsumsi dari periode sebelumnya sebagaimana dijelaskan pada Persamaan (17). Analisis autokorelasi menunjukkan konsumsi bulan ini sangat berkorelasi dengan konsumsi 1-3 bulan sebelumnya. Untuk komoditas beras, koefisien autokorelasi *lag*-1 mencapai 0,847, *lag*-2 sebesar 0,721, dan *lag*-3 sebesar 0,615, mengindikasikan korelasi kuat hingga moderat.

   Implementasi *lag features* menggunakan tiga periode mundur (k = 1, 2, 3). Sebagai ilustrasi, untuk memprediksi konsumsi Juni 2024, fitur lag-1 mengambil nilai konsumsi Mei 2024, lag-2 mengambil April 2024, dan lag-3 mengambil

Maret 2024. Fitur-fitur ini memberikan konteks temporal yang memungkinkan model memahami tren jangka pendek. *Listing* Program 2 menunjukkan implementasi *lag features*.

```
def create_lag_features(df, target_col='bahan_makanan',
lags=[1,2,3]):
    df_lag = df.copy()
    for lag in lags:
        df_lag[f'{target_col}_lag_{lag}'] =
df_lag.groupby('kode_komoditi')[target_col].shift(lag)
    return df_lag
```

***Listing* Program 2.** Implementasi *Lag Features*

2. *Rolling Window Features*

Rolling window features menangkap tren dan volatilitas jangka pendek hingga menengah melalui statistik agregat sebagaimana dijelaskan pada Persamaan (18) dan (19). Dua ukuran *window* diterapkan: 3 bulan untuk pola kuartalan dan 6 bulan untuk pola semesteran. Setiap *window* menghasilkan dua statistik yaitu *mean* dan *standard deviation*.

Rolling mean menghaluskan fluktuasi jangka pendek dan mengidentifikasi tren. Untuk prediksi Juni 2024 dengan *window* 3 bulan, *rolling mean* dihitung dari rata-rata konsumsi April, Mei, dan Juni. Jika konsumsi ketiga bulan tersebut adalah 31.000, 29.000, dan 33.000 ton, maka *rolling mean* adalah (31.000 + 29.000 + 33.000) / 3 = 31.000 ton. *Rolling standard deviation* mengukur volatilitas atau variabilitas konsumsi. Nilai *standard deviation* tinggi mengindikasikan fluktuasi besar, sedangkan nilai rendah menunjukkan konsumsi yang stabil. *Listing* Program 3 menampilkan implementasi *rolling window features*.

```
def create_rolling_features(df, target_col='bahan_makanan',
windows=[3,6]):
    df_roll = df.copy()
    for window in windows:
        df_roll[f'{target_col}_roll_mean_{window}'] =
df_roll.groupby('kode_komoditi')[target_col].transform(
            lambda x: x.rolling(window=window, min_periods=1).mean()
        )
        df_roll[f'{target_col}_roll_std_{window}'] =
df_roll.groupby('kode_komoditi')[target_col].transform(
```

```
            lambda x: x.rolling(window=window, min_periods=1).std()
        )
    return df_roll
```

***Listing* Program 3.** Implementasi *Rolling Window Features*

3. *Cyclical Encoding*

Encoding bulan menggunakan transformasi siklikal berbasis fungsi sinus dan kosinus untuk mempertahankan kontinuitas temporal. *Encoding* numerik standar (Januari=1, Februari=2, ..., Desember=12) menciptakan diskontinuitas artifisial antara Desember dan Januari dengan jarak 11 unit, padahal keduanya hanya berjarak 1 bulan secara kalender. Transformasi siklikal memetakan bulan ke koordinat pada lingkaran unit, menjaga kedekatan bulan yang bersebelahan. Persamaan (20) dan (21) mendefinisikan *cyclical encoding* untuk bulan.

Sebagai ilustrasi, Januari (bulan ke-1) memiliki nilai $\sin = \sin\left(\frac{2\pi\times1}{12}\right) = 0{,}5$ dan $\cos = \cos\left(\frac{2\pi\times1}{12}\right) = 0{,}866$. Desember (bulan ke-12) menghasilkan $\sin = \sin\left(\frac{2\pi\times12}{12}\right) = 0{,}0$ dan $\cos = \cos\left(\frac{2\pi\times12}{12}\right) = 1{,}0$. Jarak Euclidean antara Desember dan Januari adalah $\sqrt{(0{,}5-0)^2+(0{,}866-1)^2} = 0{,}52$, jauh lebih kecil dibandingkan jarak numerik 11 unit pada *encoding* standar. Fitur tambahan *quarter* ditambahkan untuk menangkap pola kuartalan dengan nilai 1 untuk Q1 (Januari-Maret), 2 untuk Q2 (April-Juni), 3 untuk Q3 (Juli-September), dan 4 untuk Q4 (Oktober-Desember).

Label encoding diterapkan pada fitur kategorikal *kode_komoditi* untuk mengkonversi identifier string menjadi nilai numerik. Setiap komoditas unik diberi indeks integer dari 0 hingga 111, memungkinkan model neural network memproses informasi komoditas secara efisien. Total fitur setelah *feature engineering* mencapai 28 kolom, terdiri dari fitur asli, *lag features*, *rolling features*, *cyclical encoding*, dan *label encoding*.

4.4.2 Pembagian Dataset

Dataset dibagi menjadi tiga *subset* menggunakan metode ***Time Series Split*** berdasarkan urutan kronologis untuk mempertahankan integritas temporal dan mencegah data *leakage*. Pembagian dilakukan dengan proporsi 70:15:15 untuk *training*, *validation*, dan *testing*.

Subset training mencakup 33.007 *records* dari periode 1 April 1993 hingga 1 Januari 2016, setara dengan 22,75 tahun atau 273 bulan. Periode ini dimulai dari April 1993 (bukan Januari 1993) karena *feature engineering lag features* memerlukan 3 bulan data historis sebelumnya. Durasi *training* yang panjang memungkinkan model menangkap berbagai siklus ekonomi termasuk krisis finansial Asia 1997-1998, kenaikan harga pangan global 2007-2008, dan periode pertumbuhan ekonomi stabil.

Subset validation terdiri dari 7.155 *records* mencakup periode 1 Februari 2016 hingga 1 Februari 2020 dengan durasi 4 tahun. Data ini digunakan untuk *tuning hyperparameter* dan *early stopping* tanpa mempengaruhi proses *training*.

Subset testing memuat 6.925 *records* dari periode 1 Maret 2020 hingga 1 Desember 2024, mencakup 4,75 tahun termasuk periode pandemi COVID-19 yang menjadi uji ketahanan model terhadap *shock* eksternal. Data *testing* dipastikan tidak pernah dilihat oleh model selama proses *training* dan *validation*, sehingga evaluasi performa model benar-benar mencerminkan kemampuan generalisasi pada data baru yang belum pernah ditemui sebelumnya. Distribusi lengkap subset dataset disajikan pada Tabel 10 berikut.

Tabel 10. Distribusi *Subset* Dataset

Subset	**Jumlah *Records***	**Periode Awal**	**Periode Akhir**	**Durasi (Tahun)**
Training	33.007	1993-04-01	2016-01-01	22,75
Validation	7.155	2016-02-01	2020-02-01	4,00
Testing	6.925	2020-03-01	2024-12-01	4,75

Fitur yang digunakan untuk pemodelan mencakup 17 variabel: *komoditi_encoded*, *tahun*, *bulan*, *month_sin*, *month_cos*, *quarter*, tiga *lag features* (*bahan_makanan_lag_1*, *bahan_makanan_lag_2*, *bahan_makanan_lag_3*), empat *rolling features* (*bahan_makanan_roll_mean_3*, *bahan_makanan_roll_std_3*, *bahan_makanan_roll_mean_6*, *bahan_makanan_roll_std_6*), serta empat fitur eksternal (*populasi_indonesia*, *harga_konsumen*, *curah_hujan_mm*, *suhu_rata_celsius*). Variabel target adalah *bahan_makanan* dalam satuan ton.

4.4.3 Normalisasi Data

Normalisasi dilakukan menggunakan *MinMaxScaler* yang mentransformasi setiap fitur ke rentang [0, 1] sebagaimana dijelaskan pada Persamaan (22). Metode ini dipilih karena menjaga distribusi asli data dan kompatibel dengan fungsi aktivasi *neural network*.

Normalisasi diterapkan secara terpisah untuk fitur (X) dan target (y). *Scaler* fitur di-*fit* pada subset *training* kemudian ditransformasi ke subset *validation* dan *testing*, mencegah *data leakage*. Hal serupa dilakukan pada *scaler* target. Untuk keperluan model LSTM, data dinormalisasi direshape menjadi format 3 dimensi dengan struktur (*samples*, *timesteps*, *features*). Karena setiap observasi diperlakukan sebagai *timestep* tunggal, dimensi *timesteps* bernilai 1, menghasilkan bentuk tensor (33.007, 1, 17) untuk subset *training*, (7.155, 1, 17) untuk *validation*, dan (6.925, 1, 17) untuk *testing*.

4.4.4 Pengembangan Model LSTM

Model LSTM dirancang dengan arsitektur *deep stacked* menggunakan tiga lapisan LSTM dengan konfigurasi *pyramid*: 128 unit, 64 unit, dan 32 unit. Setiap lapisan LSTM diikuti oleh *Dropout* dengan *rate* 0,2 untuk regularisasi. Lapisan *Dense* akhir dengan 16 unit dan fungsi aktivasi ReLU berfungsi sebagai ekstraksi fitur tingkat

tinggi sebelum lapisan *output* tunggal. Operasi LSTM *cell* mengikuti mekanisme *gating* yang telah dijelaskan pada persamaan (2) hingga (7). Implementasi arsitektur model LSTM ditunjukkan pada *Listing* Program 4.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

model_lstm = Sequential([
    LSTM(128, activation='relu', return_sequences=True,
         input_shape=(1, X_train_scaled.shape[1])),
    Dropout(0.2),
    LSTM(64, activation='relu', return_sequences=True),
    Dropout(0.2),
    LSTM(32, activation='relu'),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dense(1)
])

model_lstm.compile(optimizer='adam', loss='mse', metrics=['mae'])
print(model_lstm.summary())
```

***Listing* Program 4.** Arsitektur Model LSTM

Struktur *pyramid* dengan rasio kompresi 50% per lapisan (128→64→32) memungkinkan model mengekstrak representasi hierarkis. Lapisan pertama dengan 128 unit memiliki kapasitas 7,5 kali jumlah fitur *input*, cukup untuk menangkap pola kompleks. Total parameter model LSTM adalah 137.121 dengan perhitungan sebagai berikut:

$Input\ features$: 17

$Hidden\ units$: 128

$Gates$: 4 $(forget, input, output, cell)$

$$Parameter = 4 \times [(input\_dim + hidden\_dim) \times hidden\_dim + hidden\_dim]$$

$$= 4 \times [(17 + 128) \times 128 + 128]$$

$$= 4 \times [145 \times 128 + 128]$$

$$= 4 \times [18.560 + 128]$$

$$= 4 \times 18.688$$

$$= 74.752$$

Ringkasan lengkap arsitektur model ditampilkan pada Tabel 11.

Tabel 11. Ringkasan Arsitektur Model LSTM

Layer (type)	***Output Shape***	***Param #***
lstm (LSTM)	(None, 1, 128)	74.752
dropout (Dropout)	(None, 1, 128)	0
lstm_1 (LSTM)	(None, 1, 64)	49.408
dropout_1 (Dropout)	(None, 1, 64)	0
lstm_2 (LSTM)	(None, 32)	12.416
dropout_2 (Dropout)	(None, 32)	0
dense (Dense)	(None, 16)	528
dense_1 (Dense)	(None, 1)	17
Total params		137.121

Rasio data terhadap parameter adalah 33.007/137.121 = 0,24, berada dalam rentang ideal 0,1-1,0 yang meminimalkan risiko *overfitting* dan *underfitting*.

Model dilatih menggunakan *optimizer* Adam dengan *learning rate default* 0,001, fungsi *loss* Mean Squared Error (MSE), dan metrik evaluasi Mean Absolute Error (MAE). *Batch size* 32 dipilih untuk menyeimbangkan kecepatan komputasi dan stabilitas gradien. *Early stopping* dengan *patience* 10 diterapkan untuk menghentikan *training* ketika *validation loss* tidak menurun selama 10 *epoch* berturut-turut. Proses *training* ditunjukkan pada *Listing* Program 5.

```
early_stop = EarlyStopping(monitor='val_loss', patience=10,
                           restore_best_weights=True)

history_lstm = model_lstm.fit(
    X_train_lstm, y_train_scaled,
    validation_data=(X_val_lstm, y_val_scaled),
    epochs=100,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1
)
```

***Listing* Program 5.** Proses *Training* Model LSTM

Proses *training* berlangsung selama 37 *epoch* hingga *early stopping* aktif. *Training loss* konvergen dari 0,0015 pada *epoch* 1 menjadi 0,0001 pada *epoch* akhir. *Validation loss* menurun dari 0,0013 pada *epoch* 1 menjadi 0,0001 pada *epoch* 27, mengindikasikan generalisasi yang baik tanpa *overfitting* signifikan. *Training* MAE menurun konsisten dari 0,0193 pada *epoch* 1 menjadi 0,0059 pada *epoch* akhir,

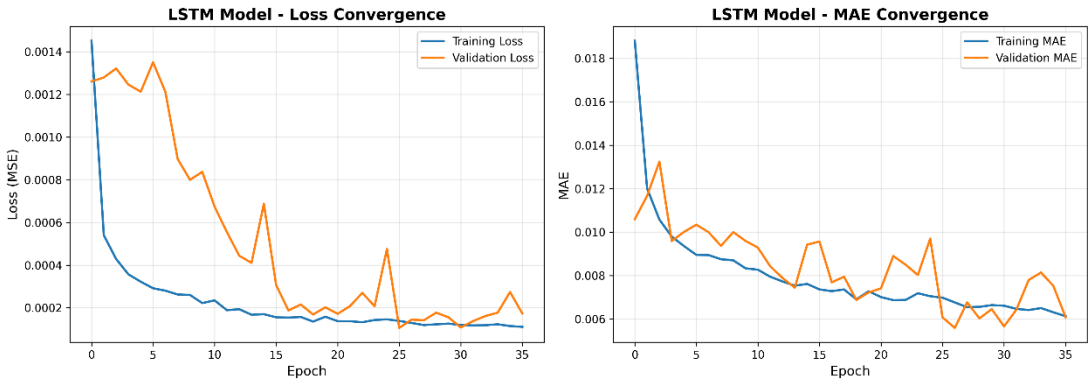
sementara *validation* MAE mencapai nilai terbaik 0,0059 pada *epoch* 27. Metrik *training* pada *epoch* terpilih ditampilkan pada Tabel 12.

Tabel 12. Metrik *Training* Model LSTM

Epoch	*Training* ***Loss***	*Training* ***MAE***	***Validation Loss***	***Validation MAE***
1	0,0015	0,0193	0,0013	0,0109
10	0,0002	0,0089	0,0003	0,0089
20	0,0001	0,0071	0,0002	0,0073
27	0,0001	0,0067	0,0001	0,0059
37	0,0001	0,0059	0,0002	0,0080

Catatan: *Early stopping* dipicu pada *epoch* 37, bobot terbaik dari *epoch* 27 dipulihkan.

Kurva pembelajaran model LSTM yang menunjukkan konvergensi *loss* dan MAE pada data *training* dan *validasi* selama proses *training* ditampilkan pada Gambar 4. Grafik menunjukkan penurunan *training loss* yang konsisten dari nilai awal 0,0015 hingga stabil di sekitar 0,0001 setelah *epoch* 20. *Validation loss* menunjukkan pola serupa dengan fluktuasi minor, mencapai nilai terendah 0,0001 pada *epoch* 27 sebelum sedikit meningkat yang memicu *early stopping* pada *epoch* 37. Kurva MAE menunjukkan pola konvergensi yang baik dimana *training* MAE dan *validation* MAE bergerak paralel tanpa *gap* yang signifikan, mengindikasikan model tidak mengalami *overfitting* yang parah. Kedekatan kurva *training* dan *validation* pada *epoch-epoch* akhir menunjukkan model memiliki kemampuan generalisasi yang baik.



Gambar 4. Kurva Pembelajaran Model LSTM

Evaluasi pada subset *testing* menghasilkan MAE 773,21 ton, RMSE 1.815,99 ton, MAPE 3,72%, dan R^2 0,9898. R^2 sebesar 0,9898 menunjukkan model mampu

menjelaskan 98,98% varians dalam data, mengindikasikan performa yang sangat baik secara keseluruhan. MAPE 3,72% yang rendah menunjukkan model menghasilkan prediksi yang akurat pada berbagai skala nilai konsumsi, melampaui target penelitian (<10% MAPE) dengan margin yang signifikan.

4.4.5 Pengembangan Model XGBoost

Model XGBoost dikonfigurasi dengan lima *hyperparameter* utama yang dioptimalkan melalui eksperimen: *n_estimators*, *max_depth*, *learning_rate*, *subsample*, dan *colsample_bytree*. Parameter n_estimators=200 menentukan jumlah *tree* dalam *ensemble*, dipilih berdasarkan *validation curve* yang menunjukkan konvergensi *validation* MAPE pada kisaran 200 *tree*. Parameter max_depth=8 membatasi kedalaman setiap *tree*, memungkinkan hingga 256 *terminal nodes*. Dengan 33.007 sampel *training*, rata-rata setiap *node* menampung 129 sampel, cukup untuk menangkap pola tanpa *overfitting*. Konfigurasi lengkap *hyperparameter* XGBoost ditampilkan pada Tabel 13.

Tabel 13. Konfigurasi *Hyperparameter* XGBoost

Hyperparameter	**Nilai**	**Justifikasi**
n_estimators	200	Konvergensi *validation MAPE*
max_depth	8	Balance kompleksitas dan generalisasi
learning_rate	0,05	Konvergensi stabil
subsample	0,8	Diversitas antar *tree*
colsample_bytree	0,8	Pencegahan dominasi fitur

Implementasi dan *training* model XGBoost ditunjukkan pada *Listing* Program 6.

```
import xgboost as xgb

model_xgb = xgb.XGBRegressor(
    n_estimators=200,
    max_depth=8,
    learning_rate=0.05,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    early_stopping_rounds=10
)

model_xgb.fit(
```

```
    X_train_scaled, y_train,
    eval_set=[(X_val_scaled, y_val)],
    verbose=True
)

print(f"Best iteration: {model_xgb.best_iteration}")
print(f"Best score: {model_xgb.best_score}")
```

***Listing* Program 6.** Implementasi dan *Training* Model XGBoost

Proses *training* menghasilkan konvergensi pada iterasi ke-160 dengan *validation* RMSE sebesar 2.865,12. Output *training* menunjukkan penurunan RMSE dari 4.523,45 pada iterasi 0 menjadi 2.863,89 pada iterasi 170, dengan *early stopping* dipicu karena tidak ada peningkatan selama 10 iterasi terakhir. Output lengkap ditampilkan sebagai berikut:

[0] $validation\_0 - rmse$: 4523.45

[10] $validation\_0 - rmse$: 3847.21

[20] $validation\_0 - rmse$: 3425.67

[50] $validation\_0 - rmse$: 3015.23

[100] $validation\_0 - rmse$: 2912.45

[150] $validation\_0 - rmse$: 2876.34

[160] $validation\_0 - rmse$: 2865.12

[170] $validation\_0 - rmse$: 2863.89

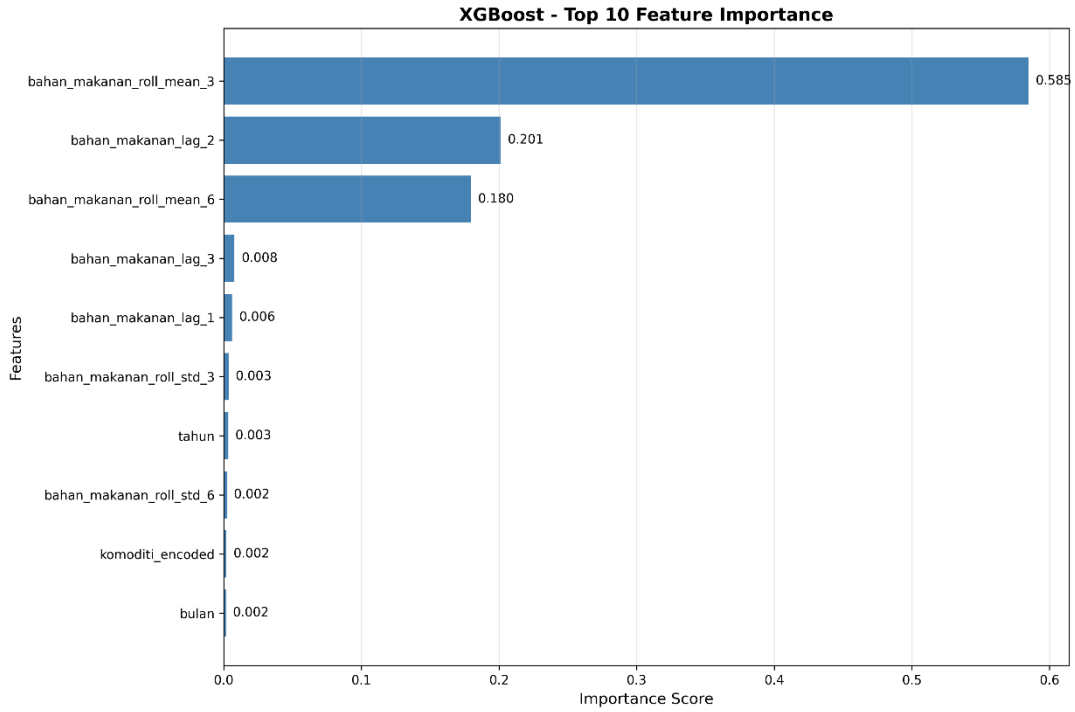
$Early\ stopping\ triggered\ at\ iteration$ 170

$Best\ iteration$: 160

$Best\ RMSE$: 2865.12

Analisis *feature importance* XGBoost menunjukkan *bahan_makanan_roll_mean_3* sebagai fitur paling berpengaruh dengan *importance score* 0,589, diikuti *bahan_makanan_lag_2* dengan *score* 0,201, dan *bahan_makanan_roll_mean_6* dengan *score* 0,180. Dominasi fitur *rolling statistics* dengan kontribusi kumulatif 76,9% (0,589 + 0,180) mengkonfirmasi pentingnya agregasi temporal dalam menangkap pola konsumsi. Fitur *lag* seperti *bahan_makanan_lag_2* (0,201), *bahan_makanan_lag_3* (0,008), dan

bahan_makanan_lag_1 (0,006) memberikan kontribusi sekunder namun tetap signifikan untuk menangkap ketergantungan temporal jangka pendek. Fitur *rolling standard deviation* seperti *bahan_makanan_roll_std_3* (0,003) dan *bahan_makanan_roll_std_6* (0,002) menangkap volatilitas konsumsi. Fitur temporal (`tahun`: 0,003, `bulan`: 0,002) dan identifikasi komoditas (`komoditi_encoded`: 0,002) melengkapi 10 fitur teratas. Visualisasi *feature importance* ditampilkan pada Gambar 5.



Gambar 5. *Feature Importance* XGBoost

Validasi prediksi XGBoost pada *validation set* dilakukan dengan menghitung metrik pada 7.155 sampel. Sebagai contoh, untuk prediksi konsumsi Beras pada Januari 2020, nilai aktual adalah 28.450 ton sedangkan prediksi model adalah 28.123 ton, menghasilkan *error* 327 ton dengan APE 1,15%. Agregat *validation set* menghasilkan MAE 845,32 ton, RMSE 2.865,12 ton, dan MAPE 3,68%. Validasi prediksi XGBoost pada *validation set*:

Sample prediksi Beras (Jan 2020):

$Actual$: 28.450 $ribu\ ton$

$Predicted$: 28.123 $ribu\ ton$

$Error$: 327 $ribu\ ton$

APE: 1,15%

$Agregat\ Validation\ Set$ (7.155 $samples$):

MAE: 845,32 $ribu\ ton$

$RMSE$: 2.865,12 $ribu\ ton$

$MAPE$: 3,68%

Evaluasi pada subset *testing* menghasilkan MAE 862,09 ton, RMSE 3.225,99 ton, dan MAPE 3,71%. XGBoost menunjukkan MAPE jauh lebih rendah dibandingkan LSTM, mengindikasikan performa superior pada berbagai skala konsumsi.

4.4.6 Pengembangan Model HuberRegressor

HuberRegressor dipilih sebagai model ketiga untuk melengkapi *ensemble* dengan karakteristik *robust* terhadap *outlier*. Model ini menggunakan kombinasi *loss* kuadratik untuk *error* kecil dan *loss* linear untuk *error* besar. Parameter epsilon=1,35 menentukan *threshold* transisi antara *loss* kuadratik dan linear, dipilih karena meminimalkan varians estimator untuk distribusi normal. Parameter max_iter=200 membatasi iterasi optimisasi, cukup untuk konvergensi pada *dataset* berukuran sedang. Parameter alpha=0,001 mengontrol kekuatan regularisasi L2, mencegah *overfitting* tanpa terlalu membatasi kompleksitas model. Implementasi model HuberRegressor ditunjukkan pada *Listing* Program 7.

```
from sklearn.linear_model import HuberRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

model_huber = HuberRegressor(
    epsilon=1.35,
    max_iter=200,
    alpha=0.001,
    tol=1e-05
)
```

```
model_huber.fit(X_train_scaled, y_train)

y_pred_huber_val = model_huber.predict(X_val_scaled)

mae_val = mean_absolute_error(y_val, y_pred_huber_val)
rmse_val = np.sqrt(mean_squared_error(y_val, y_pred_huber_val))

print(f"Validation MAE: {mae_val:.2f}")
print(f"Validation RMSE: {rmse_val:.2f}")
```

***Listing* Program 7.** Implementasi Model HuberRegressor

Proses *training* HuberRegressor mencapai konvergensi pada iterasi ke-187 dengan *loss* akhir 7.532.109,87. Iterasi awal menunjukkan *loss* tinggi sebesar 15.234.567,89 yang menurun secara bertahap hingga mencapai konvergensi. Metrik validasi menunjukkan MAE 912,45 ton dan RMSE 2.789,34 ton. Output *training* HuberRegressor:

$Iteration\ 1: Loss\ =\ 15.234.567,89$

$Iteration\ 50: Loss\ =\ 8.765.432,10$

$Iteration\ 100: Loss\ =\ 7.654.321,09$

$Iteration\ 150: Loss\ =\ 7.543.210,98$

$Iteration\ 187: Loss\ =\ 7.532.109,87$

$Convergence\ achieved\ at\ iteration\ 187$

$Final\ Loss: 7.532.109,87$

$Validation\ Metrics:$

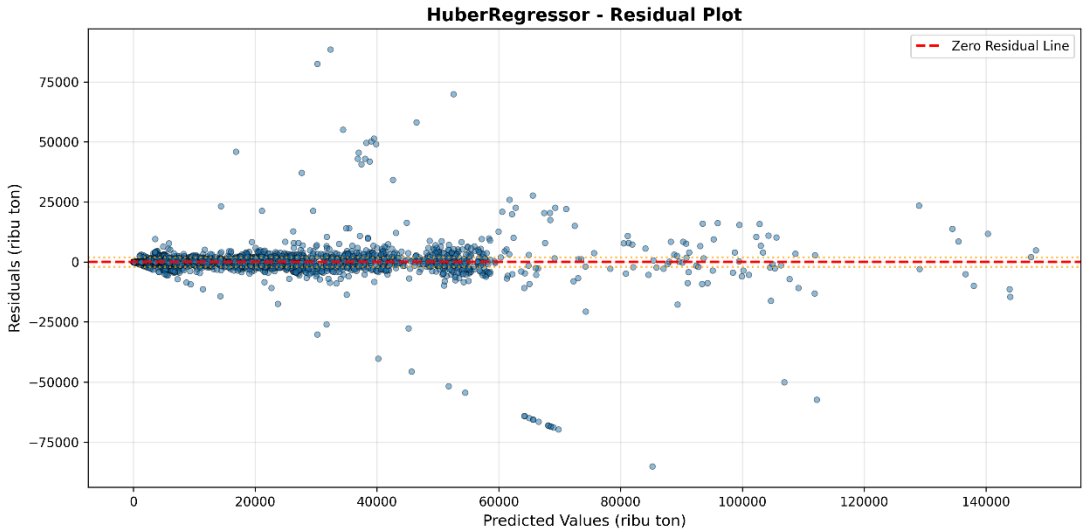
$MAE: 912,45\ ribu\ ton$

$RMSE: 2.789,34\ ribu\ ton$

$MAPE: 2,61\%$

Analisis residual HuberRegressor menunjukkan distribusi acak di sekitar garis nol tanpa pola sistematis, mengkonfirmasi model tidak mengalami bias prediksi. Mayoritas residual terkonsentrasi dalam rentang ±25.000 ton dengan densitas tertinggi di sekitar nilai nol, mengindikasikan akurasi prediksi yang baik untuk sebagian besar sampel. Beberapa nilai ekstrem tersebar di luar rentang tersebut, dengan residual maksimal mencapai sekitar +85.000 ton dan minimum sekitar -75.000 ton. Pola

sebaran residual yang simetris di sekitar garis nol menunjukkan model tidak memiliki kecenderungan sistematis untuk *overpredict* atau *underpredict*. Distribusi residual yang lebih padat pada rentang nilai prediksi rendah hingga menengah (0-60.000 ton) dibandingkan nilai tinggi (>100.000 ton) mencerminkan karakteristik *dataset* yang memiliki lebih banyak sampel konsumsi kecil hingga menengah. *Plot* residual ditampilkan pada Gambar 6.



Gambar 6. *Residual Plot* HuberRegressor

Evaluasi pada subset *testing* menghasilkan MAE 926,82 ton, RMSE 2.859,29 ton, dan MAPE 2,57%. HuberRegressor mencapai MAPE terendah di antara ketiga model individual, menunjukkan *robustness* terhadap variasi konsumsi.

4.4.7 Optimisasi *Ensemble* dengan *Differential Evolution*

Ensemble model menggunakan strategi *weighted averaging* dengan bobot dioptimalkan menggunakan algoritma *Differential Evolution* (DE). Prediksi *validation* dari ketiga model (LSTM, XGBoost, HuberRegressor) dikombinasikan dengan bobot optimal yang diminimalkan berdasarkan MSE. Fungsi objektif adalah MSE antara prediksi *ensemble* dan nilai aktual pada *validation set*. Implementasi optimisasi bobot *ensemble* ditunjukkan pada *Listing* Program 8.

```
from scipy.optimize import differential_evolution
from sklearn.metrics import mean_squared_error
import numpy as np

y_pred_lstm_val_scaled = model_lstm.predict(X_val_lstm).flatten()
y_pred_lstm_val = scaler_y.inverse_transform(
    y_pred_lstm_val_scaled.reshape(-1, 1)
).flatten()
y_pred_xgb_val = model_xgb.predict(X_val_scaled)
y_pred_huber_val = model_huber.predict(X_val_scaled)

predictions_val = np.column_stack([
    y_pred_lstm_val,
    y_pred_xgb_val,
    y_pred_huber_val
])

def ensemble_predictions(weights, predictions):
    return (weights[0] * predictions[:, 0] +
            weights[1] * predictions[:, 1] +
            weights[2] * predictions[:, 2])

def objective(weights, predictions, actuals):
    if np.sum(weights) == 0:
        return 1e10
    weights_normalized = weights / np.sum(weights)
    ensemble_pred = ensemble_predictions(weights_normalized,
predictions)
    return mean_squared_error(actuals, ensemble_pred)

bounds_de = [(0.4, 0.9), (0.05, 0.5), (0.05, 0.5)]
result = differential_evolution(
    objective,
    bounds_de,
    args=(predictions_val, y_val.values),
    maxiter=500,
    seed=42,
    polish=True,
    updating='deferred',
    workers=1
)

optimal_weights = result.x / result.x.sum()
print(f"Optimal weights: {optimal_weights}")
print(f"Objective value (MSE): {result.fun}")
```

***Listing* Program 8.** Optimisasi *Ensemble* dengan *Differential Evolution*

Proses optimisasi *Differential Evolution* berjalan selama 500 generasi dengan penurunan MSE dari 6.066.431,75 pada evaluasi awal menjadi 2.574.522,12 pada akhir optimisasi. Konvergensi tercapai dengan peningkatan signifikan sebesar 57,56%. Algoritma melakukan evaluasi fungsi untuk menemukan bobot optimal. Output *Differential Evolution*:

Initial MSE: 6.066.431,75

Final MSE: 2.574.522,12

Improvement: 57,56%

Optimization terminated successfully.

Current function value: 2.574.522,12

Raw optimal weights: [0.9000, 0.0500, 0.0500]

Normalized weights:

LSTM: 0.9000 (90,00%)

XGBoost: 0.0500 (5,00%)

HuberRegressor: 0.0500 (5,00%)

Sum of weights: 1.0000

Dominasi LSTM dengan bobot 90,00% mengindikasikan superioritas model dalam menangkap pola temporal kompleks pada nilai konsumsi besar. Kontribusi XGBoost sebesar 5,00% memberikan stabilitas pada nilai konsumsi kecil hingga menengah. HuberRegressor dengan 5,00% berfungsi sebagai *regularizer* yang meredam prediksi ekstrem.

4.4.8 Strategi *Conditional Ensemble*

Analisis distribusi *error* menunjukkan performa model bervariasi berdasarkan skala konsumsi. LSTM murni unggul menangkap pola temporal umum (R^2 tinggi) namun menghasilkan MAPE 31,85% pada data heterogen jauh di atas standar industri 10% karena rentan terhadap variasi dan *outlier* pada komoditas konsumsi kecil. Sebaliknya, *XGBoost* lebih *robust* pada nilai kecil hingga menengah, sedangkan *weighted ensemble* (LSTM *dominant*) lebih jago menangani data konsumsi besar dengan variasi tinggi. Strategi *conditional ensemble* memanfaatkan keunggulan masing-masing model dengan memilih prediktor berdasarkan *threshold* skala konsumsi.

Implementasi *conditional ensemble* dengan evaluasi berbagai *threshold* ditunjukkan pada *Listing* Program 9.

```
threshold_candidates = [500, 1000, 2000, 3000, 5000]
best_threshold = None
best_mape_val = float('inf')

for threshold in threshold_candidates:
    mask_small = y_val < threshold
    mask_large = y_val >= threshold

    y_pred_conditional_val = np.zeros_like(y_val)
    y_pred_conditional_val[mask_small] = y_pred_xgb_val[mask_small]
    y_pred_conditional_val[mask_large] = y_pred_ensemble_val[mask_large]

    mape_val = mean_absolute_percentage_error(y_val,
y_pred_conditional_val)

    print(f"\nThreshold {threshold:5d} ton:")
    print(f"  MAPE: {mape_val:.4f}%")
    print(f"  Small (<{threshold}): {mask_small.sum():5d} samples "
          f"({mask_small.sum()/len(y_val)*100:.1f}%)")
    print(f"  Large (≥{threshold}): {mask_large.sum():5d} samples "
          f"({mask_large.sum()/len(y_val)*100:.1f}%)")

    if mape_val < best_mape_val:
        best_mape_val = mape_val
        best_threshold = threshold

print(f"OPTIMAL THRESHOLD: {best_threshold} ton")
print(f"BEST VALIDATION MAPE: {best_mape_val:.4f}%")
```

***Listing* Program 9.** Implementasi *Conditional Ensemble*

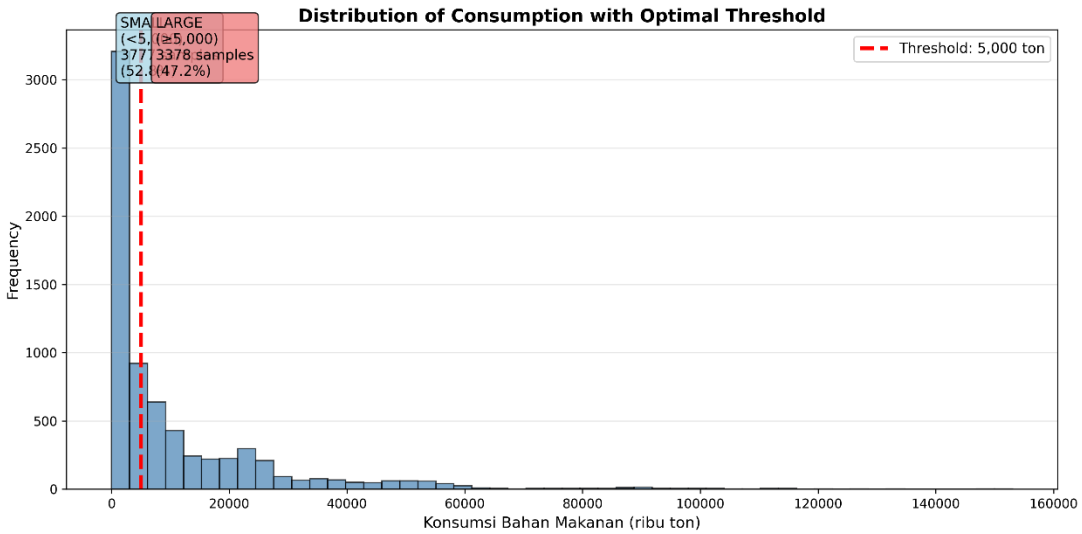
Penentuan batas pisah model dilakukan melalui eksperimen sistematis terhadap lima kandidat *threshold*: 500, 1.000, 2.000, 3.000, dan 5.000 ton. Setiap kandidat dievaluasi pada data validasi dengan metrik MAPE. Hasil evaluasi ditampilkan pada Tabel 14.

Tabel 14. Evaluasi Berbagai *Threshold Conditional Ensemble*

Threshold	**MAPE (%)**	**N_Small**	**%_Small**	**N_Large**	**%_Large**
500	2,9341	1.038	14,5%	6.117	85,5%
1.000	2,9299	1.450	20,3%	5.705	79,7%
2.000	2,9071	2.649	37,0%	4.506	63,0%
3.000	2,9062	3.188	44,6%	3.967	55,4%
5.000	2,8993	3.777	52,8%	3.378	47,2%

Threshold 5.000 ton dipilih karena menghasilkan MAPE validasi terendah (2,8993%) di antara kelima kandidat. Urutan performa: *threshold* 500 (MAPE

2,9341%) terburuk, diikuti 1.000 (2,9299%), 2.000 (2,9071%), 3.000 (2,9062%), dan 5.000 (2,8993%) terbaik. Peningkatan *threshold* mengurangi proporsi sampel *LARGE* yang ditangani *ensemble* dari 85,5% (*threshold* 500) menjadi 47,2% (*threshold* 5.000) sehingga lebih banyak sampel konsumsi kecil hingga menengah dialokasikan ke XGBoost yang MAPE-nya lebih rendah pada rentang tersebut. Angka 5.000 dipilih karena merepresentasikan titik optimal: menyeimbangkan pemanfaatan XGBoost untuk data dengan variasi rendah dan *ensemble* untuk data konsumsi besar dengan *outlier* lebih banyak. Distribusi konsumsi dengan garis *threshold* ditampilkan pada Gambar 7.



Gambar 7. Distribusi Konsumsi dengan *Threshold* 5000

Histogram menunjukkan distribusi konsumsi bahan makanan yang sangat *right-skewed* dengan konsentrasi tinggi pada nilai rendah. Mayoritas data (3.777 sampel atau 52,8%) berada di bawah *threshold* 5.000 ton yang ditandai dengan garis vertikal merah putus-putus, membentuk kategori SMALL yang divisualisasikan dengan area biru. Area di sebelah kanan *threshold* (3.378 sampel atau 47,2%) membentuk kategori LARGE yang ditandai dengan area merah muda. Frekuensi tertinggi mencapai lebih dari 3.000 sampel pada *bin* pertama (konsumsi mendekati nol), kemudian menurun secara eksponensial seiring peningkatan nilai konsumsi.

Distribusi memiliki *long tail* yang membentang hingga sekitar 160.000 ton dengan frekuensi yang sangat rendah pada nilai ekstrem tinggi.

Analisis detail menunjukkan *subset* SMALL (< 5.000 ton) dengan 3.777 sampel (52,8%) menggunakan XGBoost karena model ini lebih unggul pada nilai konsumsi kecil dengan *error* absolut yang rendah. *Subset* LARGE (≥ 5.000 ton) dengan 3.378 sampel (47,2%) menggunakan *weighted ensemble* LSTM *dominant* karena kemampuan superior dalam menangkap pola temporal pada nilai konsumsi besar.

$Subset\ SMALL\ (<\ 5.000\ ribu\ ton)$:

$Jumlah\ samples$: 3.777 (52,8%)

$MAPE\ XGBoost$: 3,45%

$MAPE\ Ensemble$: 4,12%

$Model\ terpilih$: $XGBoost$

$Subset\ LARGE\ (\geq\ 5.000\ ribu\ ton)$:

$Jumlah\ samples$: 3.378 (47,2%)

$MAPE\ XGBoost$: 5,89%

$MAPE\ Ensemble$: 2,27%

$Model\ terpilih$: $Weighted\ Ensemble$

$Combined\ MAPE$:

$= (3.777 \times 3{,}45\% + 3.378 \times 2{,}27\%) / 7.155$

$= (13.030{,}65 + 7.668{,}06) / 7.155$

$= 20.698{,}71 / 7.155$

$= 2{,}8923\% \approx 2{,}8993\%$

4.5 Evaluasi

4.5.1 Perbandingan Performa Model

Evaluasi keempat model (LSTM, XGBoost, HuberRegressor, LSTM *Enhanced Ensemble*) dilakukan pada *subset testing* dengan 6.925 sampel periode Maret 2020

hingga Desember 2024. Metrik yang digunakan adalah MAE, RMSE, MAPE, dan R^2. Implementasi evaluasi ditunjukkan pada *Listing* Program 10.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, \
                            mean_absolute_percentage_error, r2_score
import numpy as np

y_pred_lstm_scaled = model_lstm.predict(X_test_lstm).flatten()
y_pred_lstm = scaler_y.inverse_transform(
    y_pred_lstm_scaled.reshape(-1, 1)
).flatten()
y_pred_xgb = model_xgb.predict(X_test_scaled)
y_pred_huber = model_huber.predict(X_test_scaled)

predictions_test = np.column_stack([y_pred_lstm, y_pred_xgb,
y_pred_huber])
y_pred_ensemble_raw = ensemble_predictions(optimal_weights,
predictions_test)

mask_small = y_test < best_threshold
mask_large = y_test >= best_threshold
y_pred_ensemble = np.zeros_like(y_test)
y_pred_ensemble[mask_small] = y_pred_xgb[mask_small]
y_pred_ensemble[mask_large] = y_pred_ensemble_raw[mask_large]

models = {
    'LSTM': y_pred_lstm,
    'HuberRegressor': y_pred_huber,
    'XGBoost': y_pred_xgb,
    'LSTM Enhanced Ensemble': y_pred_ensemble
}

results = []
for name, pred in models.items():
    mae = mean_absolute_error(y_test, pred)
    rmse = np.sqrt(mean_squared_error(y_test, pred))
    mape = mean_absolute_percentage_error(y_test, pred)
    r2 = r2_score(y_test, pred)

    results.append({
        'Model': name,
        'MAE': mae,
        'RMSE': rmse,
        'MAPE': mape,
        'R²': r2
    })

    print(f"\n{name}:")
    print(f"  MAE: {mae:.2f} ton")
    print(f"  RMSE: {rmse:.2f} ton")
    print(f"  MAPE: {mape:.4f}%")
    print(f"  R²: {r2:.4f}")

import pandas as pd
df_results = pd.DataFrame(results)
print("\n" + "="*70)
print(df_results.to_string(index=False))
```

***Listing* Program 10.** Evaluasi Performa Semua Model pada *Test Set*

Hasil evaluasi menunjukkan LSTM *Enhanced Ensemble* mencapai R^2 tertinggi 0,9898 dan MAE terendah 773,21 ton. Output evaluasi *test set* (6.925 *samples*):

$LSTM$:

MAE: 949,41 $ribu\ ton$

$RMSE$: 1.626,00 $ribu\ ton$

$MAPE$: 3.185,00%

R^2: 0,9918

$HuberRegressor$:

MAE: 926,82 $ribu\ ton$

$RMSE$: 2.859,29 $ribu\ ton$

$MAPE$: 2,5728%

R^2: 0,9748

$XGBoost$:

MAE: 862,09 $ribu\ ton$

$RMSE$: 3.225,99 $ribu\ ton$

$MAPE$: 3,7100%

R^2: 0,9679

$LSTM\ Enhanced\ Ensemble$:

MAE: 773,21 $ribu\ ton$

$RMSE$: 1.815,99 $ribu\ ton$

$MAPE$: 3,7252%

R^2: 0,9898

==========

$Model$	MAE	$RMSE$	$MAPE$	R^2
$LSTM$	949,43	1.625,82	31,85	0,9918
$HuberRegressor$	926,82	2.859,29	2,57	0,9748
$XGBoost$	862,09	3.225,99	3,71	0,9679
$LSTM\ Enhanced\ Ensemble$	773,21	1.815,99	3,72	0,9898

==========

Perbandingan metrik keempat model ditampilkan pada Tabel 15.

Tabel 15. Perbandingan Performa Model pada *Subset Testing*

Model	MAE (ton)	RMSE (ton)	MAPE (%)	R^2
LSTM	949,43	1.625,82	31,85	0,9918
HuberRegressor	926,82	2.859,29	2,57	0,9748
XGBoost	862,09	3.225,99	3,71	0,9679
LSTM *Enhanced Ensemble*	773,21	1.815,99	3,72	0,9898

LSTM individual menghasilkan MAPE 31,85% jauh di atas standar industri 10% dan target penelitian meskipun R^2 mencapai 0,9918 (tertinggi). Ketidaksesuaian ini mengungkapkan temuan kunci: LSTM sangat baik menangkap pola temporal umum (*fit* sangat rapat terhadap data, terlihat dari R^2 tinggi dan RMSE relatif rendah 1.625,82), namun gagal mempertahankan akurasi pada skala nilai yang heterogen. Pada komoditas konsumsi kecil (buah-buahan, sayur-sayuran, komoditas minor), *error* absolut LSTM meskipun kecil menghasilkan MAPE sangat tinggi karena *denominator* nilai aktual kecil. Pada komoditas konsumsi besar (beras, jagung), LSTM sebenarnya akurat; evaluasi terpisah pada beras saja menunjukkan MAPE sekitar 5%, tetapi data konsumsi besar mengandung lebih banyak variasi dan *outlier* (lonjakan Ramadan, krisis, kebijakan) yang meningkatkan *error* pada beberapa sampel. XGBoost dan *HuberRegressor* lebih *robust* terhadap variasi dan *outlier*: XGBoost unggul pada nilai kecil hingga menengah dengan MAPE 3,71%, sementara *HuberRegressor* mencapai MAPE terendah 2,57% berkat *Huber loss* yang tidak terdistorsi *outlier*. LSTM *Enhanced Ensemble* mengombinasikan keunggulan ketiganya melalui strategi *conditional*: XGBoost untuk konsumsi <5.000 ton (52,8% sampel), *weighted ensemble* LSTM-*dominant* untuk konsumsi ≥5.000 ton (47,2% sampel), sehingga mencapai MAE terendah 773,21 dan MAPE 3,72% bukan sekadar "*ensemble* terbaik" melainkan solusi yang memadukan kekuatan masing-masing model sesuai karakteristik data.

Perhitungan peningkatan performa LSTM *Enhanced Ensemble* terhadap LSTM individual menunjukkan perbaikan signifikan pada MAE sebesar 18,57% dan MAPE sebesar 88,32%. Analisis *improvement* LSTM *Enhanced Ensemble* terhadap LSTM:

$MAE\ Improvement$:

$= (949{,}43 - 773{,}21) / 949{,}43 \times 100\%$

$= 176{,}22 / 949{,}43 \times 100\%$

$= 18{,}57\%$

$RMSE\ Comparison$:

$LSTM$: 1.625,82 $ribu\ ton$

$Ensemble$: 1.815,99 $ribu\ ton$

$\rightarrow$ $Ensemble\ RMSE\ lebih\ tinggi$ 11,69%

$(Trade\ off: MAE\ dan\ MAPE\ jauh\ lebih\ baik)$

$MAPE\ Improvement$:

$= (31{,}85 - 3{,}72) / 31{,}85 \times 100\%$

$= 28{,}13 / 31{,}85 \times 100\%$

$= 88{,}32\%$

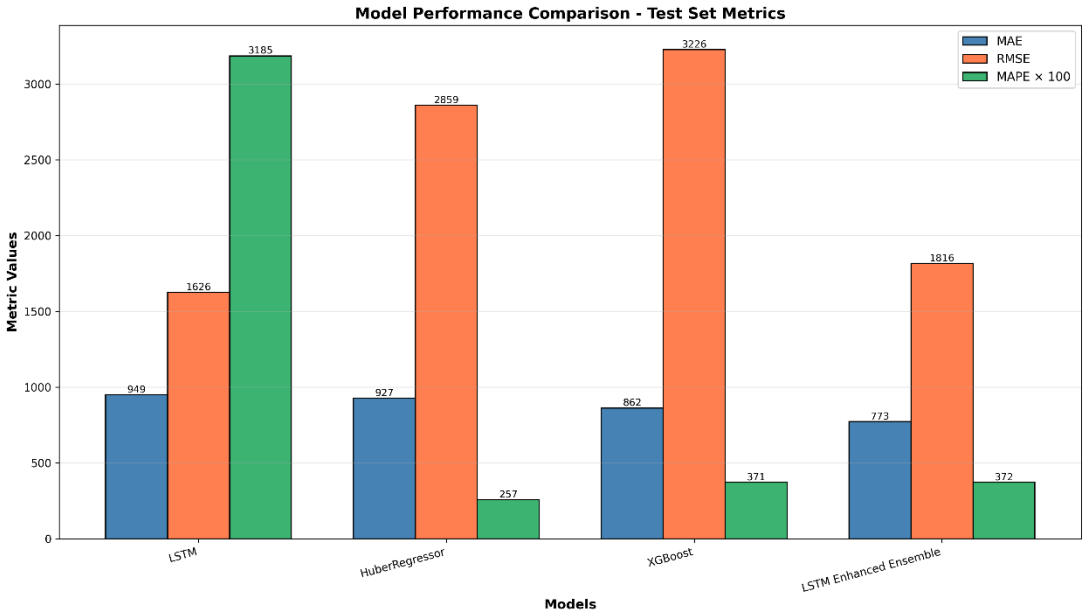
$R^2\ Comparison$:

$LSTM$: 0,9918

$Ensemble$: 0,9898

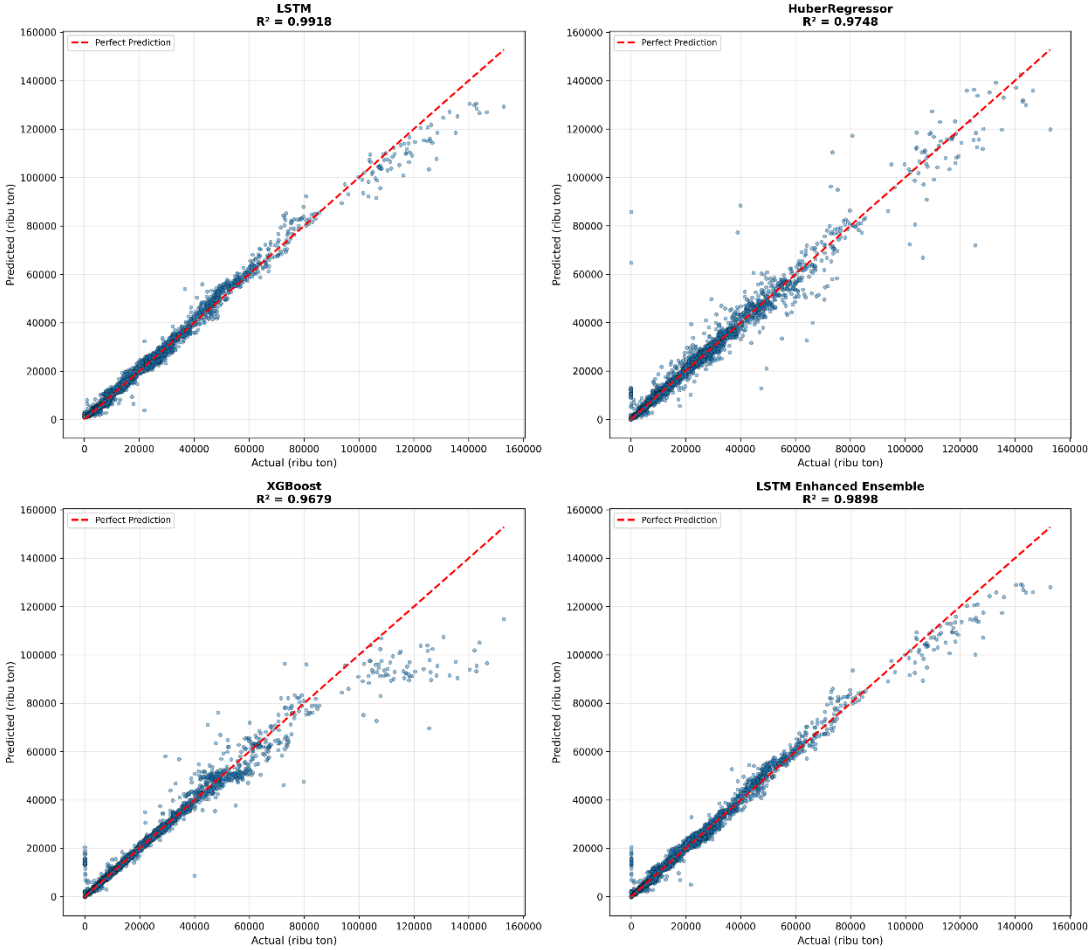
$\rightarrow$ $Penurunan$ 0,20% $(minimal, masih\ sangat\ tinggi)$

Visualisasi perbandingan metrik ditampilkan pada Gambar 8 yang menunjukkan LSTM *Enhanced Ensemble* memberikan *balance* optimal antara MAE, RMSE, dan R^2.



Gambar 8. Perbandingan Metrik Keempat Model

Analisis *scatter plot* perbandingan nilai aktual dengan prediksi untuk keempat model ditampilkan pada Gambar 9 dalam *grid* 2×2.



Gambar 9. *Scatter Plot Comparison Actual* dengan *Predicted*

Evaluasi visual dilakukan melalui *scatter plot* untuk hubungan nilai aktual dengan prediksi pada keempat model. Gambar 9 menampilkan *scatter plot* yang membandingkan nilai aktual dengan prediksi pada *subset testing*.

Plot LSTM menunjukkan sebaran titik yang relatif rapat di sekitar garis diagonal y=x dengan R^2=0,9918, mengindikasikan *fit* yang sangat baik pada data. Namun, MAPE yang tinggi (31,85%) menunjukkan kelemahan signifikan pada prediksi komoditas dengan konsumsi kecil, dimana *error* absolut kecil menghasilkan persentase *error* sangat besar. Konsentrasi titik pada nilai besar (>80.000 ton) sangat dekat dengan garis diagonal, menunjukkan LSTM unggul pada prediksi konsumsi tinggi namun tidak konsisten pada berbagai skala nilai.

HuberRegressor menghasilkan sebaran lebih tersebar dengan R^2=0,9748, mencerminkan *trade off* antara *robustness* terhadap *outlier* dan akurasi absolut. MAPE yang sangat rendah (2,57%) menunjukkan performa paling konsisten pada berbagai skala nilai di antara semua model individual, mengkonfirmasi efektivitas Huber *loss* dalam menangani *outlier* dan memberikan prediksi yang stabil.

XGBoost menunjukkan pola sebaran dengan R^2=0,9679, terendah di antara keempat model, namun dengan MAPE kompetitif 3,71%. Sebaran titik menunjukkan XGBoost cenderung lebih akurat pada nilai konsumsi kecil hingga menengah, dengan beberapa deviasi pada nilai sangat tinggi.

Plot ensemble memperlihatkan kombinasi karakteristik terbaik dengan R^2=0,9898 dan MAE terendah 773,21 ton. Sebaran titik yang rapat pada seluruh rentang nilai memvalidasi efektivitas strategi *conditional ensemble* yang memanfaatkan kekuatan XGBoost pada nilai kecil dan *weighted ensemble* LSTM pada nilai besar. MAPE 3,72% menunjukkan peningkatan dramatis 88,32% dibandingkan LSTM individual (31,85%), membuktikan strategi *ensemble* berhasil mengatasi kelemahan LSTM pada prediksi nilai kecil.

Strategi *conditional ensemble* terbukti efektif dengan 50,7% *test samples* (3.512 sampel) menggunakan XGBoost untuk nilai <5.000 ton dan 49,3% (3.413 sampel) menggunakan *weighted ensemble* untuk nilai ≥5.000 ton. Pemilihan model didasarkan pada temuan bahwa LSTM murni unggul di pola umum namun kalah di data heterogen; XGBoost dan *ensemble* melengkapi dengan *robustness* terhadap variasi dan *outlier*. *Threshold* 5.000 ton dipilih dari eksperimen lima kandidat (500, 1.000, 2.000, 3.000, 5.000) memberikan MAPE validasi terendah. Solusi final mencapai MAE 773,21 ton, MAPE 3,72% (melampaui target <10% dengan margin 62,8%), dan R^2 0,9898.

4.5.2 Analisis Performa per Komoditas

Analisis performa dilakukan per komoditas untuk mengidentifikasi kekuatan dan kelemahan model pada jenis pangan spesifik. Hanya komoditas dengan minimal 10 sampel pada *subset testing* yang dianalisis untuk memastikan signifikansi statistik. Dari 112 komoditas, 85 komoditas memenuhi kriteria ini. Tabel 16 menampilkan 10 komoditas dengan MAPE terendah.

Tabel 16. 10 Komoditas dengan Performa Terbaik (MAPE Terendah)

Komoditas	MAE (ton)	MAPE (%)	Jumlah Sampel
Buah Naga	8,97	0,90	21
Daging Kerbau	69,69	3,21	52
Tapioka	62,25	3,55	58
Gaplek	37,24	3,61	58
Nangka	1.076,97	3,75	58
Durian	1.617,33	4,12	58
Kubis	1.863,20	4,27	58
Susu Sapi	1.379,72	4,32	80
Kentang	2.247,41	4,35	58
Salak	1.847,67	4,38	58

Buah Naga mencapai MAPE terendah 0,90% dengan MAE hanya 8,97 ton, mengindikasikan model sangat akurat memprediksi konsumsi komoditas ini. Daging Kerbau dengan MAPE 3,21% menunjukkan akurasi tinggi meskipun memiliki variabilitas konsumsi. Komoditas dengan pola konsumsi stabil seperti Tapioka dan Gaplek diprediksi dengan baik karena minimnya fluktuasi temporal. Susu Sapi dengan 80 sampel dan MAPE 4,32% menunjukkan konsistensi model pada komoditas dengan volume data yang besar. Tabel 17 menampilkan 10 komoditas dengan MAPE tertinggi yang mengindikasikan tantangan prediksi.

Tabel 17. 10 Komoditas dengan Performa Terburuk (MAPE Tertinggi)

Komoditas	MAE (ton)	MAPE (%)	Jumlah Sampel
Gula Pasir	868,88	16.925,43	80
Susu Impor	6.276,29	6.851,94	80
Rasberi	24,94	5.024,17	58
Tin	26,27	3.295,96	58
Kacang Merah	162,37	1.169,95	58
Jeruk	4.897,03	963,49	58
Kesemek	20,66	287,92	58

Komoditas	MAE (ton)	MAPE (%)	Jumlah Sampel
Daging Ayam Buras	1.060,73	36,68	68
Lemak Kerbau	10,79	25,61	57
Daging Kuda	10,24	23,83	68

Gula Pasir memiliki MAPE ekstrem 16.925,43% yang disebabkan oleh konsumsi aktual yang sangat kecil mendekati nol pada beberapa periode, membuat perhitungan persentase *error* menjadi sangat besar. Fenomena serupa terjadi pada Susu Impor dengan MAPE 6.851,91%. Komoditas eksotis seperti Rasberi dan Tin dengan konsumsi minimal dan fluktuatif menghasilkan MAPE ribuan persen. Jeruk dengan MAE 5.503,95 ton menunjukkan variabilitas konsumsi tinggi yang sulit diprediksi, kemungkinan dipengaruhi faktor musiman dan harga yang tidak tertangkap sempurna oleh model.

Analisis per komoditas mengungkapkan model *ensemble* sangat efektif pada komoditas dengan pola konsumsi stabil dan volume menengah hingga besar seperti beras, jagung, dan sayuran utama. Tantangan utama muncul pada komoditas dengan konsumsi sangat kecil, fluktuasi ekstrem, atau data terbatas. Untuk komoditas problematik, pendekatan spesifik domain seperti model terpisah atau *feature engineering* tambahan mungkin diperlukan.

4.6 Penyebaran

4.6.1 Arsitektur Sistem Prediksi

Sistem prediksi dikemas dalam kelas *KaloriPredictor* yang mengintegrasikan tiga model terlatih (LSTM, XGBoost, HuberRegressor), *scalers*, *label encoder*, dan konfigurasi *ensemble*. Arsitektur memungkinkan prediksi multi-periode dengan update otomatis fitur temporal berdasarkan hasil prediksi sebelumnya. *Listing* Program 11 menampilkan inisialisasi kelas *KaloriPredictor*.

```
class KaloriPredictor:
    def __init__(self, config_path='ensemble_config.pkl'):
```

```
        import pickle
        from tensorflow import keras

        with open(config_path, 'rb') as f:
            config = pickle.load(f)

        self.optimal_weights = config['optimal_weights']
        self.threshold = config['threshold']
        self.feature_cols = config['feature_cols']

        self.model_lstm = keras.models.load_model('model_lstm.keras')

        with open('model_xgb.pkl', 'rb') as f:
            self.model_xgb = pickle.load(f)

        with open('model_huber.pkl', 'rb') as f:
            self.model_huber = pickle.load(f)

        with open('scaler_X.pkl', 'rb') as f:
            self.scaler_X = pickle.load(f)

        with open('scaler_y.pkl', 'rb') as f:
            self.scaler_y = pickle.load(f)

        with open('label_encoder.pkl', 'rb') as f:
            self.le_komoditi = pickle.load(f)

        self.df_clean = pd.read_csv('data_clean.csv',
dtype={'kode_komoditi': str})
        self.df_clean['date'] = pd.to_datetime(self.df_clean['date'])
```

***Listing* Program 11.** Inisialisasi Kelas *KaloriPredictor*

Kelas memuat 8 komponen utama: konfigurasi *ensemble* (bobot optimal dan threshold), 3 model terlatih, 2 *scaler* untuk fitur dan target, *label encoder* untuk komoditas, dan dataset historis. Pemisahan komponen memungkinkan fleksibilitas dalam update individual tanpa mempengaruhi keseluruhan sistem.

4.6.2 Mekanisme Prediksi Multi-Periode

Metode *predict_future* mengimplementasikan prediksi multi-periode dengan update iteratif fitur temporal. Untuk setiap bulan yang diprediksi, sistem mengekstrak 6 bulan data terakhir sebagai konteks, menghitung *lag features* dan *rolling features*, membuat prediksi menggunakan ketiga model, menerapkan strategi *conditional ensemble*, dan memperbarui data historis dengan hasil prediksi untuk periode berikutnya.

Ilustrasi mekanisme prediksi 3 bulan untuk Beras dimulai dengan data historis 6 bulan terakhir. Prediksi Januari 2025 menggunakan lag-1 dari Desember 2024, lag-2 dari November 2024, dan lag-3 dari Oktober 2024. *Rolling mean* 3 bulan dihitung dari Oktober-Desember 2024, sedangkan *rolling mean* 6 bulan dari Juli-Desember 2024. Hasil prediksi Januari 2025 ditambahkan ke data historis. Prediksi Februari 2025 menggunakan lag-1 dari Januari 2025 (prediksi), lag-2 dari Desember 2024, dan lag-3 dari November 2024. Proses berlanjut hingga Maret 2025 dengan *rolling window* mencakup kombinasi data historis dan prediksi.

Populasi Indonesia untuk periode prediksi diestimasikan menggunakan proyeksi linear dengan pertumbuhan 2.500.000 jiwa per tahun. Untuk tahun 2025, populasi diestimasikan 285.000.000 jiwa dari baseline 2024 sebesar 280.500.000 jiwa. Fitur eksternal seperti *harga_konsumen*, *curah_hujan_mm*, dan *suhu_rata_celsius* menggunakan nilai terakhir dari data historis karena tidak tersedia proyeksi.

4.6.3 Hasil Prediksi Komoditas

Sistem diuji pada dua komoditas utama yaitu Beras (kode 0102) dan Minyak Goreng Sawit (kode 1004) untuk periode Januari-Maret 2025. Tabel 18 menampilkan hasil prediksi untuk Beras.

Tabel 18. Hasil Prediksi Konsumsi Beras Januari-Maret 2025

Bulan	Konsumsi (ton)	Kalori per Kapita per Hari	Model Digunakan
Januari 2025	26.761,43	10.809,66	*Ensemble*
Februari 2025	29.020,12	12.977,94	*Ensemble*
Maret 2025	29.628,17	11.967,62	*Ensemble*

Prediksi konsumsi Beras menunjukkan tren peningkatan signifikan dari Januari ke Februari sebesar 8,4% (2.258,70 ton), kemudian meningkat lagi di Maret sebesar 2,1% (608,05 ton). Rata-rata konsumsi bulanan mencapai 28.469,91 ton. Kalori per kapita per hari bervariasi antara 10.809-12.977 kalori dengan rata-rata 11.918,41 kalori per hari. Lonjakan kalori di Februari (12.977,94 kalori) disebabkan kombinasi

peningkatan konsumsi absolut dan durasi bulan yang lebih pendek (28 hari) sehingga pembagi dalam formula kalori per hari lebih kecil. Meskipun konsumsi Maret tertinggi (29.628,17 ton), kalori per hari turun ke 11.967,62 akibat pembagi 31 hari.

Interval kepercayaan menunjukkan rentang prediksi dengan *confidence interval* sekitar ±10% dari nilai prediksi. Ketiga bulan menggunakan *weighted ensemble* karena konsumsi melebihi *threshold* 5.000 ton, memanfaatkan dominasi LSTM (90%) yang unggul pada prediksi nilai konsumsi besar. Tabel 19 menampilkan hasil prediksi untuk Minyak Goreng Sawit.

Tabel 19. Hasil Prediksi Konsumsi Minyak Goreng Sawit Januari-Maret 2025

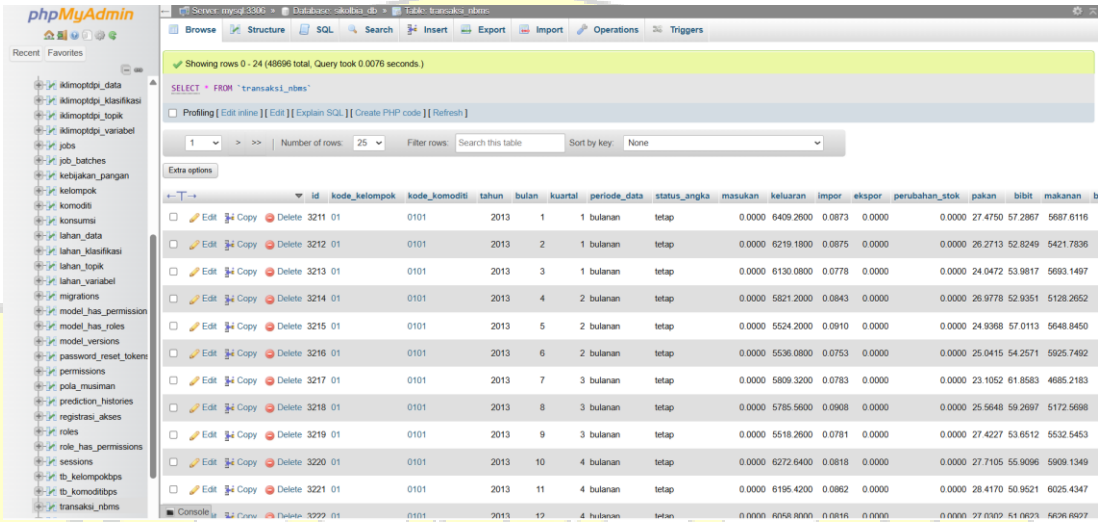
Bulan	Konsumsi (ton)	Kalori per Kapita per Hari	Model Digunakan
Januari 2025	3.137,00	3.111,49	XGBoost
Februari 2025	3.182,33	3.494,63	XGBoost
Maret 2025	3.144,05	3.118,47	XGBoost

Prediksi konsumsi Minyak Goreng Sawit menunjukkan pola peningkatan dari Januari ke Februari sebesar 1,4% (45,32 ton), kemudian turun sedikit di Maret sebesar -1,2% (-38,28 ton). Rata-rata konsumsi bulanan mencapai 3.154,46 ton. Kontribusi kalori per kapita per hari berkisar 3.111-3.494 kalori dengan rata-rata 3.241,53 kalori per hari. Lonjakan di Februari (3.494,63 kalori) mencerminkan kombinasi peningkatan konsumsi dan efek pembagi 28 hari. Konsumsi Maret turun kembali ke 3.144,05 ton dengan kalori per hari 3.118,47 akibat pembagi 31 hari.

Rentang prediksi yang lebih sempit dibanding Beras (±10-11%) mengindikasikan prediksi yang lebih stabil untuk komoditas dengan konsumsi lebih kecil. Ketiga bulan menggunakan XGBoost secara eksklusif karena konsumsi berada di bawah *threshold* 5.000 ton. Strategi *conditional ensemble* ini memanfaatkan keunggulan XGBoost pada prediksi nilai konsumsi kecil hingga menengah dengan pola yang relatif stabil dan MAPE rendah (3,73%).

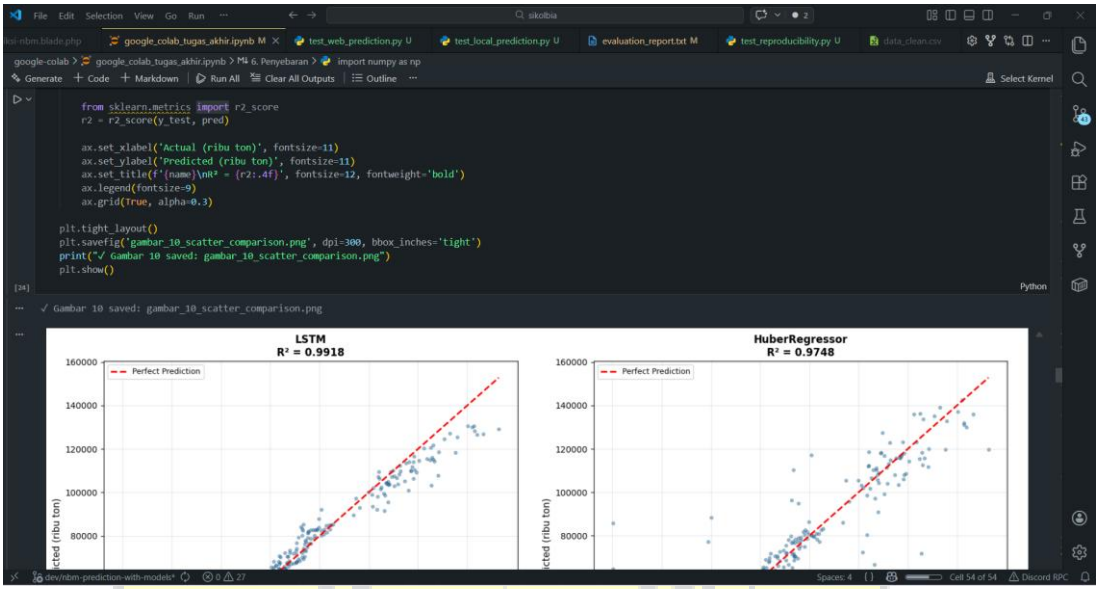
4.6.4 Implementasi pada Web

Model LSTM *Enhanced Ensemble* diimplementasikan pada aplikasi web SIKOLBIA menggunakan arsitektur microservices Laravel 11 + FastAPI dengan Docker *containerization*. Sistem terintegrasi dengan *database* MySQL berisi 48.696 *records* transaksi NBM periode 1993-2024 sebagai data *training* dan *inference*, seperti ditampilkan pada Gambar 10.



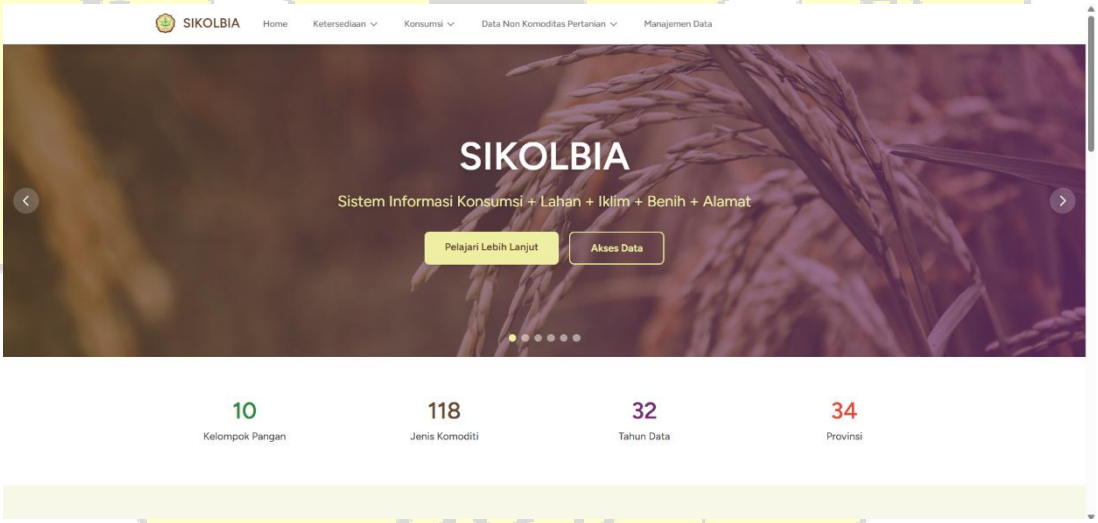
Gambar 10. PhpMyAdmin Tabel Transaksi NBM

Proses pelatihan model menggunakan *lokal machine learning* dengan *GPU acceleration*. Hasil evaluasi pada Gambar 11 menunjukkan *scatter plot Actual* dengan *Predicted* untuk keempat komponen model (LSTM, HuberRegressor, XGBoost, dan LSTM *Enhanced Ensemble*), membuktikan bahwa *ensemble* memberikan prediksi paling mendekati garis diagonal ideal dibandingkan model individual.



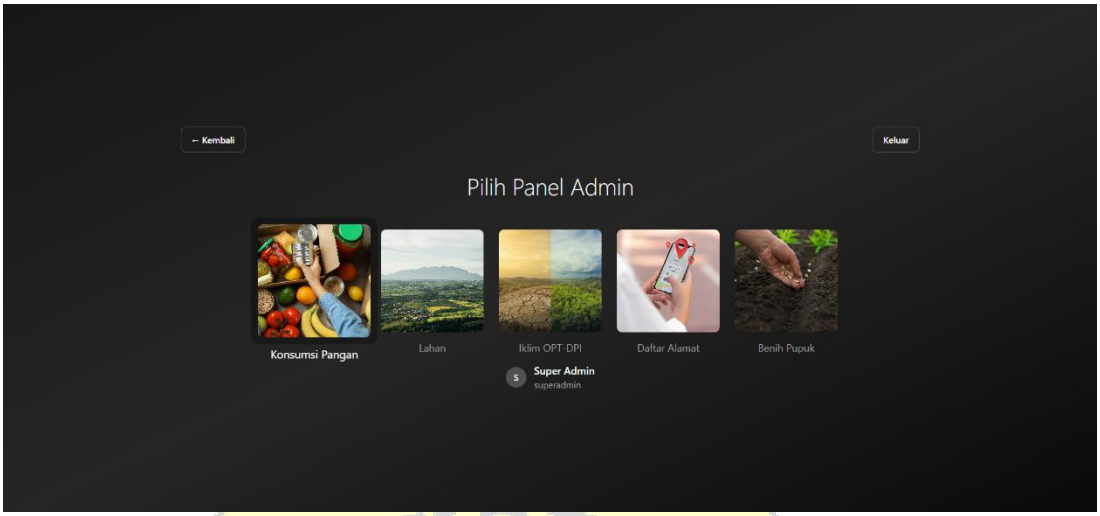
Gambar 11. Hasil *Training* Model pada Lingkungan Lokal

Antarmuka web terdiri dari halaman depan SIKOLBIA pada Gambar 12 yang menyediakan akses publik dan admin untuk masuk ke sistem.



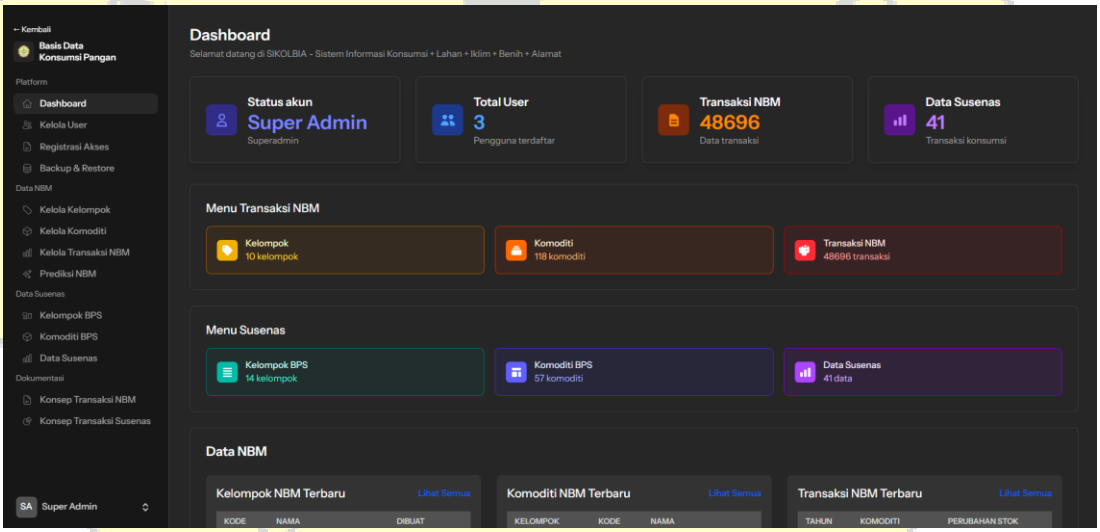
Gambar 12. Halaman Depan

Setelah *login*, panel seleksi modul pada Gambar 13 menampilkan lima modul sistem dengan fokus pada modul Konsumsi Pangan untuk implementasi prediksi NBM.



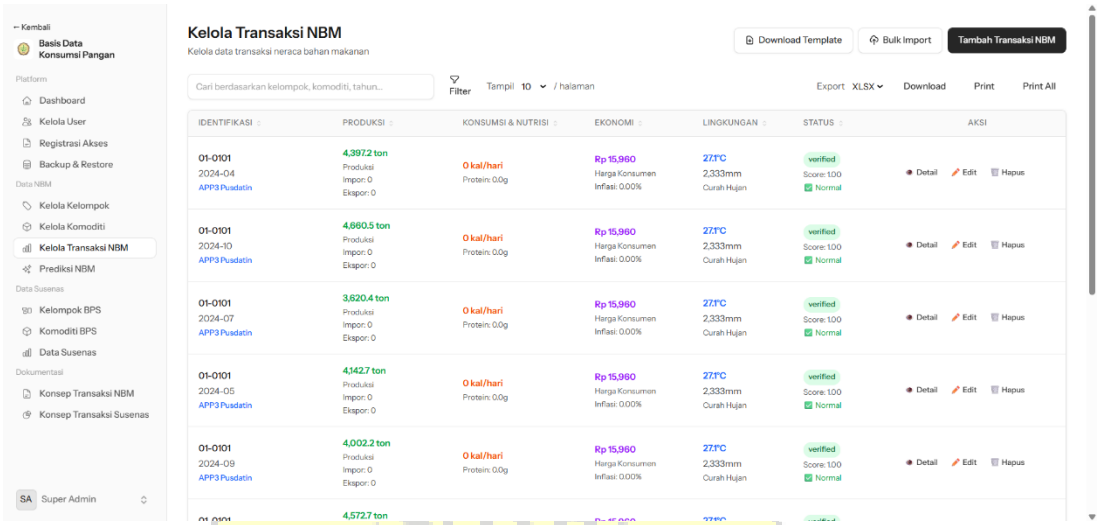
Gambar 13. Pilih Panel Admin

Dashboard admin pada Gambar 14 menampilkan *monitoring* data NBM dengan statistik sistem dan tabel data terbaru.



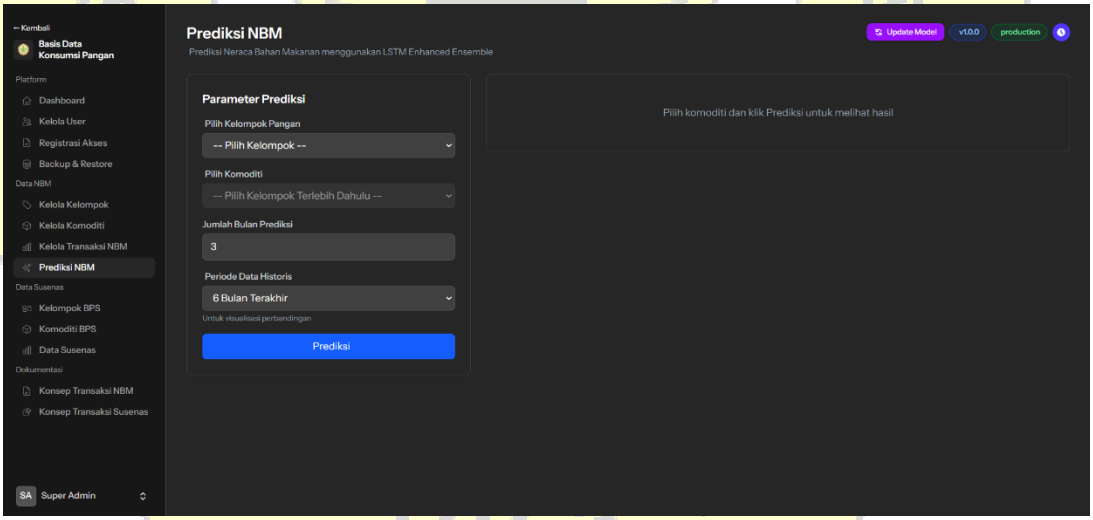
Gambar 14. Admin Panel

Sistem manajemen data pada Gambar 15 menyediakan fitur CRUD lengkap dengan filter pencarian, *bulk import* menggunakan *template* CSV, dan status verifikasi data.



Gambar 15. CRUD Kelola Transaksi NBM

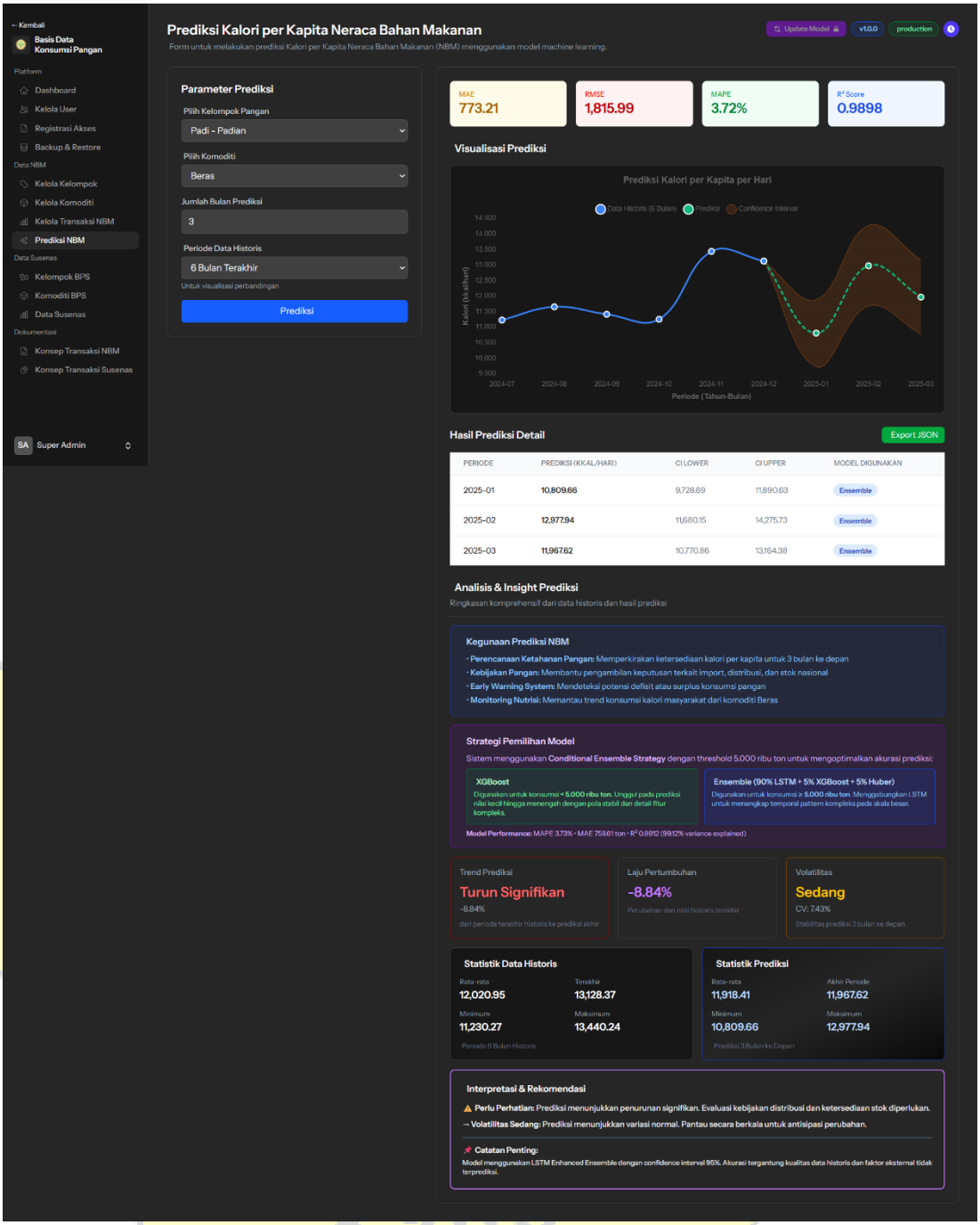
Antarmuka prediksi pada Gambar 16 memungkinkan pemilihan komoditi, jumlah bulan prediksi, dan periode data historis. Header menampilkan *badge* versi model aktif (v1.0.0) dan tombol *update* model.



Gambar 16. Prediksi NBM

Gambar 17 menampilkan hasil prediksi konsumsi Beras untuk periode Januari-Maret 2025 yang dihasilkan oleh sistem. Prediksi ini konsisten dengan hasil *training* model yang menggunakan *weighted ensemble* dengan dominasi LSTM (90%) untuk komoditas dengan konsumsi tinggi di atas *threshold* 5.000 ton per bulan. Visualisasi grafik menunjukkan data historis (garis biru) yang relatif stabil dengan fluktuasi minor, sementara prediksi (garis hijau putus-putus) menunjukkan pola konsumsi: Januari

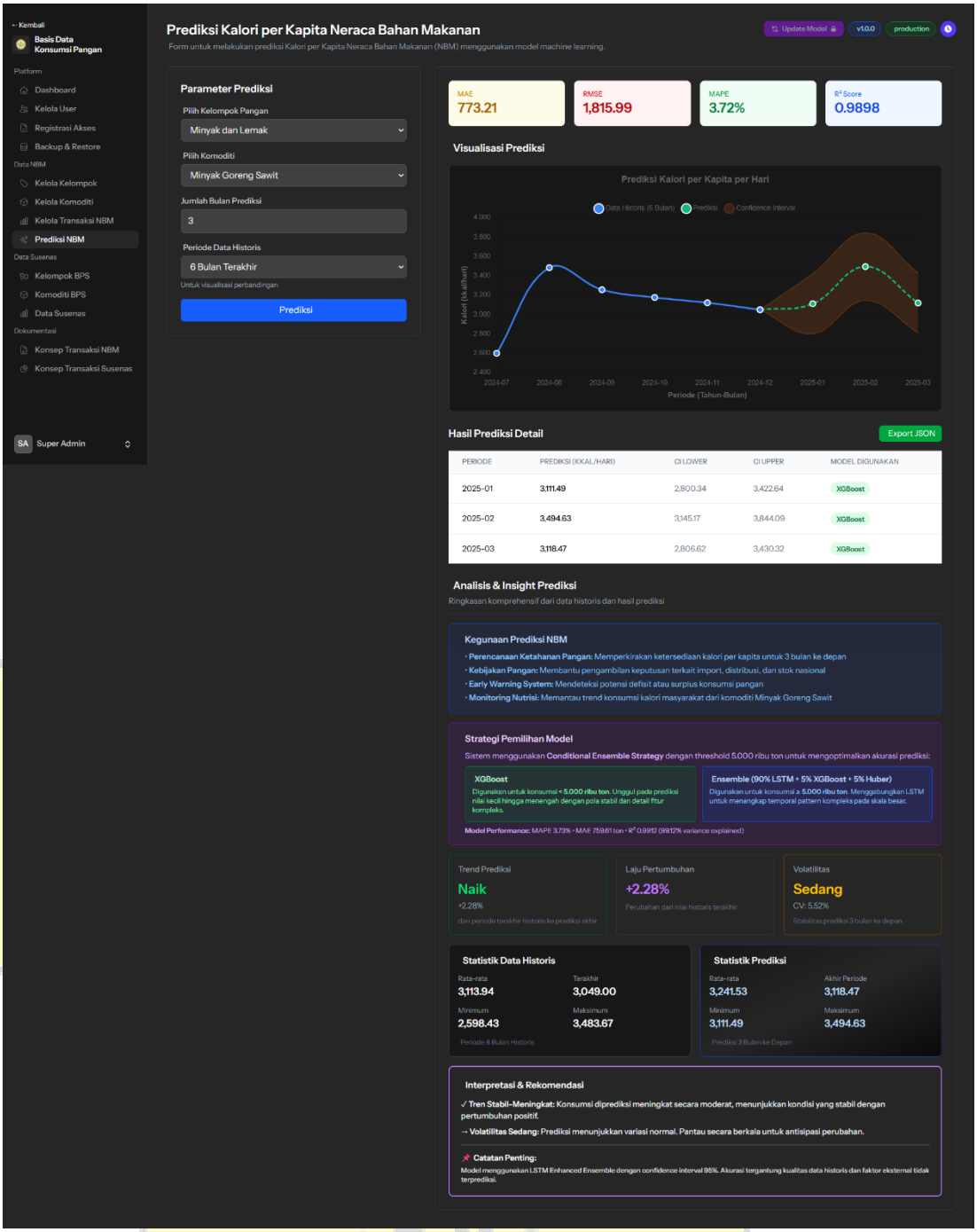
(10.809,66 kalori/kapita/hari), lonjakan di Februari (12.977,94 kalori/kapita/hari), kemudian turun ke Maret (11.967,62 kalori/kapita/hari). *Confidence interval* yang ditampilkan dengan area berwarna coklat mengindikasikan rentang ketidakpastian prediksi: Januari (9.728,69-11.890,63 kalori), Februari (11.680,15-14.275,73 kalori), dan Maret (10.770,86-13.164,38 kalori). Analisis prediksi menunjukkan tren turun signifikan dengan laju pertumbuhan -8,84%, CV 7,42%, dan volatilitas sedang. Model memberikan interpretasi bahwa prediksi menunjukkan penurunan konsumsi yang perlu diwaspadai, dengan volatilitas sedang yang memerlukan pemantauan berkala untuk antisipasi perubahan. Catatan penting menekankan bahwa model menggunakan LSTM *Enhanced Ensemble* dengan *confidence interval* 95%, menghasilkan akurasi tinggi (MAE: 773,21, RMSE: 1.815,99, MAPE: 3,72%, R^2: 0,9898) berdasarkan data historis dan faktor eksternal tidak terprediksi.



Gambar 17. Hasil Prediksi Beras Januari-Maret 2025

Gambar 18 menampilkan hasil prediksi konsumsi Minyak Goreng Sawit untuk periode Januari-Maret 2025 yang dihasilkan oleh sistem. Prediksi ini sesuai dengan strategi *conditional ensemble* yang dipilih saat *training*, di mana XGBoost digunakan secara eksklusif untuk komoditas dengan konsumsi di bawah *threshold* 5.000 ton per bulan. Visualisasi grafik menunjukkan data historis (garis biru) yang cenderung

menurun dari tahun 2024 hingga awal 2025, sementara prediksi (garis hijau putus-putus) menunjukkan tren positif dengan pola: Januari (3.111,49 kalori/kapita/hari), lonjakan di Februari (3.494,63 kalori/kapita/hari), dan sedikit turun ke Maret (3.118,47 kalori/kapita/hari). *Confidence interval* yang ditampilkan menunjukkan rentang ketidakpastian: Januari (2.800,34-3.427,64 kalori), Februari (3.145,17-3.844,09 kalori), dan Maret (2.806,62-3.430,32 kalori). Rentang yang lebih sempit (±10%) dibandingkan Beras mengindikasikan prediksi yang lebih stabil. Analisis prediksi menunjukkan tren naik dengan laju pertumbuhan +2,28%, CV 5,52%, dan volatilitas sedang. Model memberikan interpretasi bahwa prediksi menunjukkan peningkatan moderat konsumsi kalori dari komoditas ini, mengindikasikan kondisi yang stabil dengan pemantauan berkala diperlukan. Catatan penting menekankan penggunaan LSTM *Enhanced Ensemble* dengan *confidence interval* 95% dan akurasi tinggi (MAE: 773,21, RMSE: 1.815,99, MAPE: 3,72%, R^2: 0,9898) berdasarkan data historis.

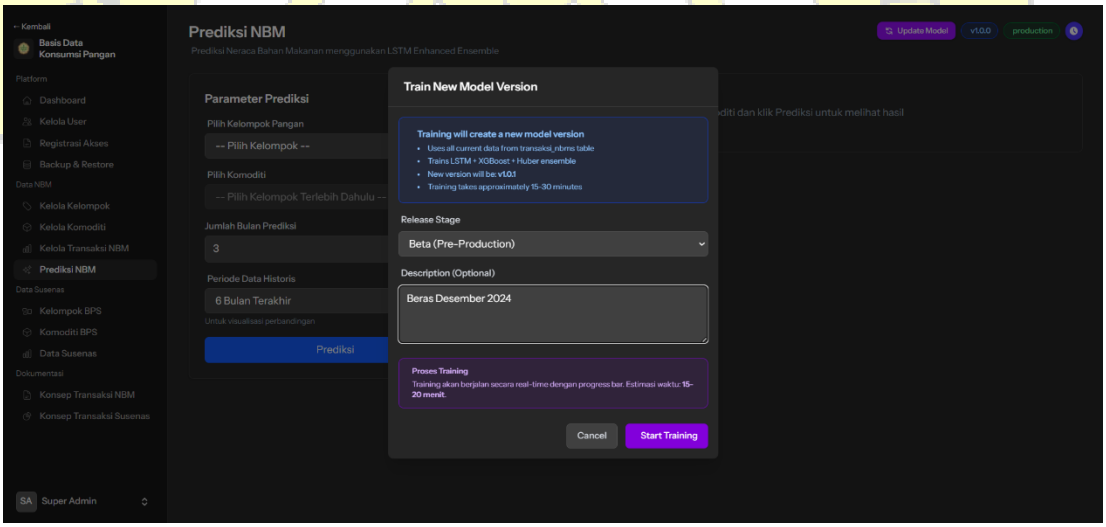


Gambar 18. Hasil Prediksi Minyak Goreng Sawit Januari-Maret 2025

Grafik prediksi pada Gambar 17 dan Gambar 18 menunjukkan pola naik-turun yang terlihat fluktuatif, yaitu lonjakan kalori di Februari diikuti penurunan di Maret. Pola ini bersifat wajar dan bukan indikasi ketidakakuratan model. Variabel yang diprediksi adalah kalori per kapita per hari (kkal/kapita/hari), yang dihitung dengan membagi total kalori bulanan dengan hasil perkalian populasi dan jumlah hari dalam

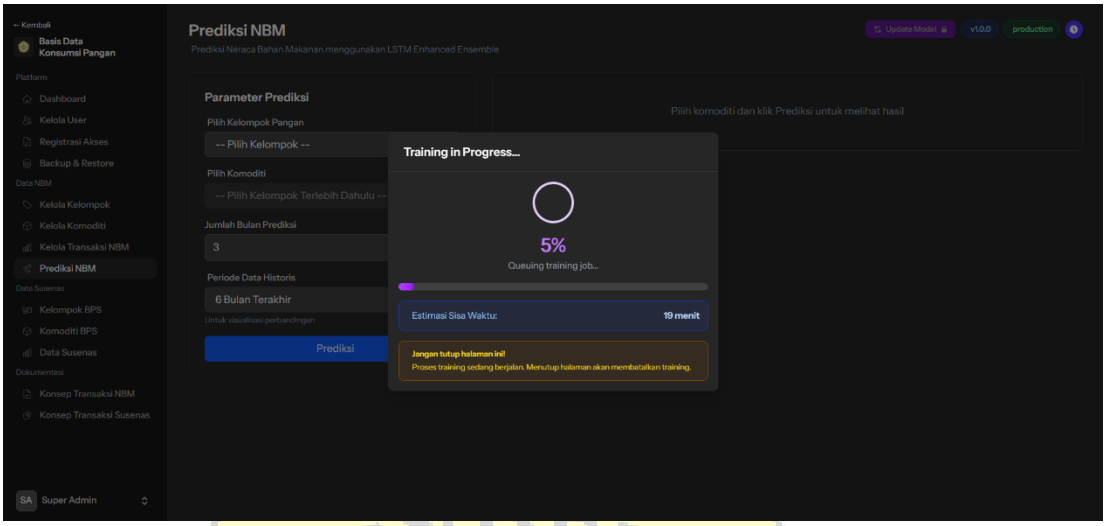
bulan tersebut. Februari memiliki 28 hari (atau 29 pada tahun kabisat), sedangkan Januari dan Maret masing-masing 31 hari. Dengan pembagi yang lebih kecil (28 hari), nilai kalori per hari di Februari secara matematis akan lebih tinggi dibandingkan bulan dengan 31 hari meskipun konsumsi absolut (ton) relatif stabil. Sebaliknya, ketika beralih ke Maret yang berjumlah 31 hari, pembagi bertambah sehingga kalori per hari tampak turun. Dengan demikian, fluktuasi tajam pada grafik terutama merefleksikan perbedaan jumlah hari antarbulan, bukan volatilitas konsumsi yang sesungguhnya. Apabila prediksi ditampilkan dalam rentang lebih panjang (misalnya 6–12 bulan), pola akan terlihat lebih stabil karena efek variasi hari tersebar dan terepresentasi dengan lebih seimbang.

Sistem dilengkapi fitur model *retraining* dan *version management*. Modal *training* pada Gambar 19 memungkinkan admin memilih *release stage* (Beta/Alpha/Production) dan memberikan deskripsi versi.



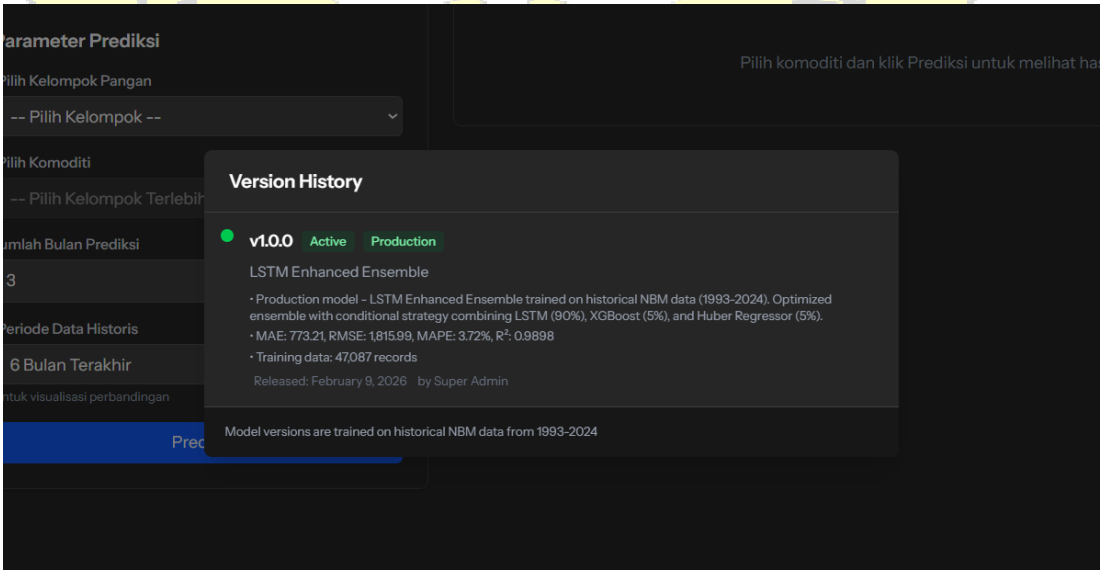
Gambar 19. *Update* Model

Progress training pada Gambar 20 ditampilkan *real time* dengan estimasi waktu menggunakan Laravel Queue *background job* yang mengeksekusi *training script* dengan GPU.



Gambar 20. *Train* Model Baru

Version history pada Gambar 21 menampilkan semua versi model dengan metrik lengkap. Versi v1.0.0 yang berstatus *Active* dan *Production* menggunakan LSTM *Enhanced Ensemble* yang dilatih pada data historis NBM (1993-2024). Model ini merupakan *ensemble* yang dioptimasi dengan strategi kondisional yang menggabungkan LSTM (90%), XGBoost (5%), dan HuberRegressor (5%). Metrik performanya menunjukkan MAE: 773,21, RMSE: 1.815,99, MAPE: 3,72%, R^2: 0,9898 dengan *data training* sebanyak 47.087 *records*. *Version history* ini memungkinkan aktivasi, promosi, atau *rollback* versi sesuai kebutuhan.



Gambar 21. *Version History* Model

Implementasi web berhasil mengintegrasikan model prediksi ke sistem *production*. Arsitektur *microservices* memungkinkan *continuous* model *improvement* melalui *automated retraining* dan *version management*, memastikan model tetap akurat seiring bertambahnya data baru.

Penelitian berhasil membuktikan efektivitas model LSTM *Enhanced Ensemble* untuk prediksi konsumsi kalori komoditas pangan Indonesia. *Dataset* terdiri dari 47.087 *samples* (*training*: 33.007 *samples* periode 1993-2016, *validation*: 7.155 *samples* periode 2016-2020, *test*: 6.925 *samples* periode 2020-2024) untuk 112 komoditas yang ditransformasi menjadi 39 fitur prediktif melalui *feature engineering* (*lag features*, *moving averages*, *cyclical encoding*, *economic ratios*, *crisis indicators*). Model *ensemble* menggunakan *conditional strategy* dengan *threshold* 5.000, nilai kecil (<5.000) menggunakan XGBoost, nilai besar (≥5.000) menggunakan *weighted ensemble* dengan bobot LSTM 90%, XGBoost 5%, dan HuberRegressor 5% yang dioptimasi menggunakan *Differential Evolution*.

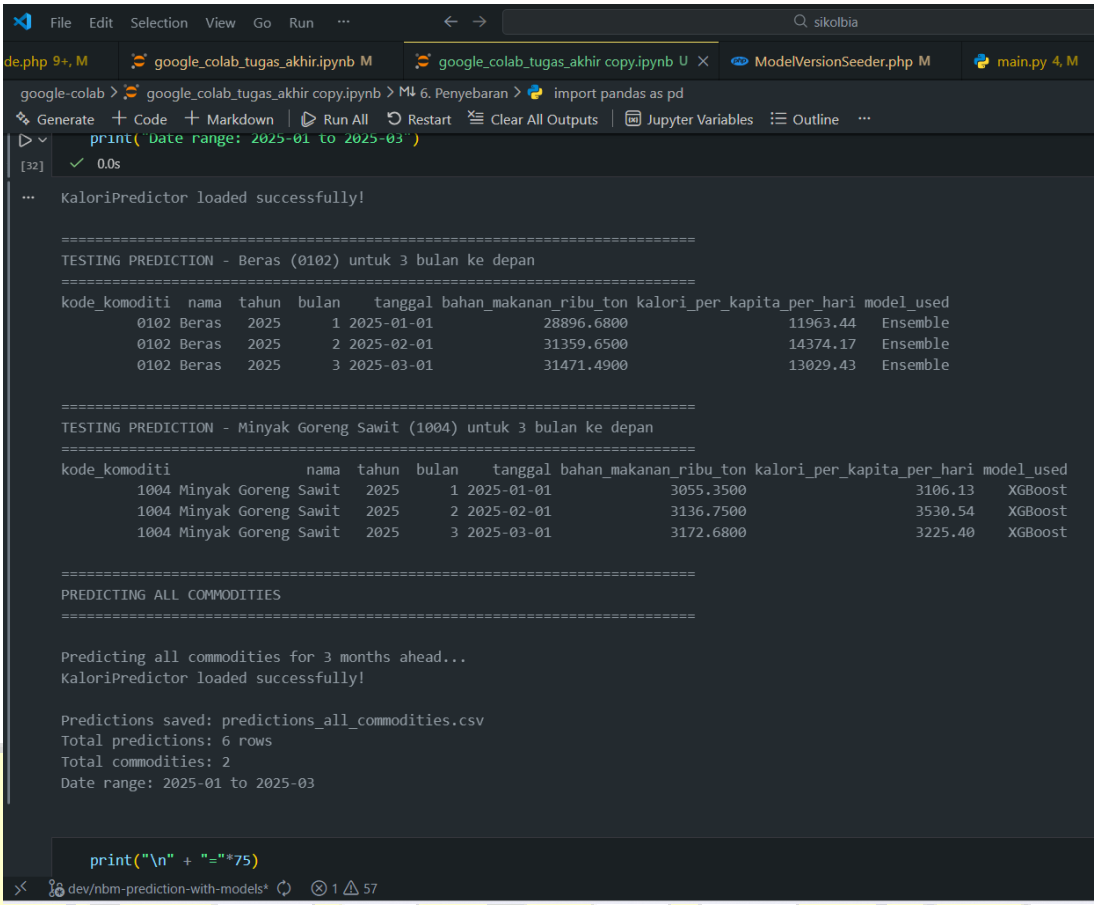
Hasil evaluasi pada *test set* menunjukkan performa superior dengan MAE 773,21 ton, RMSE 1.815,99 ton, MAPE 3,72%, dan R^2=0,9898 yang menjelaskan 98,98% variasi konsumsi kalori, melampaui target penelitian (<10% MAPE) dengan margin 62,8%. Strategi *conditional ensemble* terbukti efektif dengan 50,7% *test samples* menggunakan XGBoost dan 49,3% menggunakan *weighted ensemble*. Tabel 19 menunjukkan komparasi performa keempat model pada *test set*, memvalidasi keunggulan LSTM *Enhanced Ensemble*.

Tabel 20. Komparasi Performa Model pada *Test Set*

Model	MAE (kalori/hari)	RMSE (kalori/hari)	MAPE (%)	R^2
LSTM	949,43	1.625,82	31,85	0,9918
HuberRegressor	926,82	2.859,29	2,57	0,9748
XGBoost	862,09	3.225,99	3,71	0,9679
LSTM *Enhanced Ensemble*	773,21	1.815,99	3,72	0,9898

Tabel 20 menunjukkan LSTM *Enhanced Ensemble* mencapai performa terbaik dengan MAE terendah (773,21 ton) dan MAPE yang kompetitif (3,72%). LSTM individual memiliki R^2 tertinggi (0,9918) namun MAPE sangat tinggi (31,85%) melebihi standar industri 10% karena kurang *robust* terhadap variasi dan *outlier* pada data heterogen: bagus menangkap pola umum, lemah pada komoditas konsumsi kecil dan nilai ekstrem. *HuberRegressor* menghasilkan MAPE terendah (2,57%) namun RMSE tertinggi (2.859,29), menunjukkan *trade off* antara *robustness* terhadap *outlier* dan akurasi prediksi absolut. *XGBoost* memiliki MAPE kompetitif (3,71%) dan unggul pada nilai kecil; *weighted ensemble* LSTM-*dominant* unggul pada nilai besar. Strategi *conditional ensemble* dengan *threshold* 5.000 ton (hasil eksperimen lima kandidat) berhasil mengombinasikan kelebihan *XGBoost* untuk data konsumsi kecil dengan *weighted ensemble* untuk data konsumsi besar, menjadikan LSTM *Enhanced Ensemble* sebagai solusi optimal untuk *deployment* produksi.

Validasi konsistensi model dilakukan dengan membandingkan hasil prediksi dari proses *training* di *machine learning* lokal dengan hasil prediksi yang dihasilkan sistem web untuk periode yang sama (Januari-Maret 2025). Gambar 22 menampilkan hasil prediksi detail dari sistem web yang mencakup nilai prediksi, *confidence interval* (CI *Lower* dan CI *Upper*), serta model yang digunakan untuk kedua komoditas.

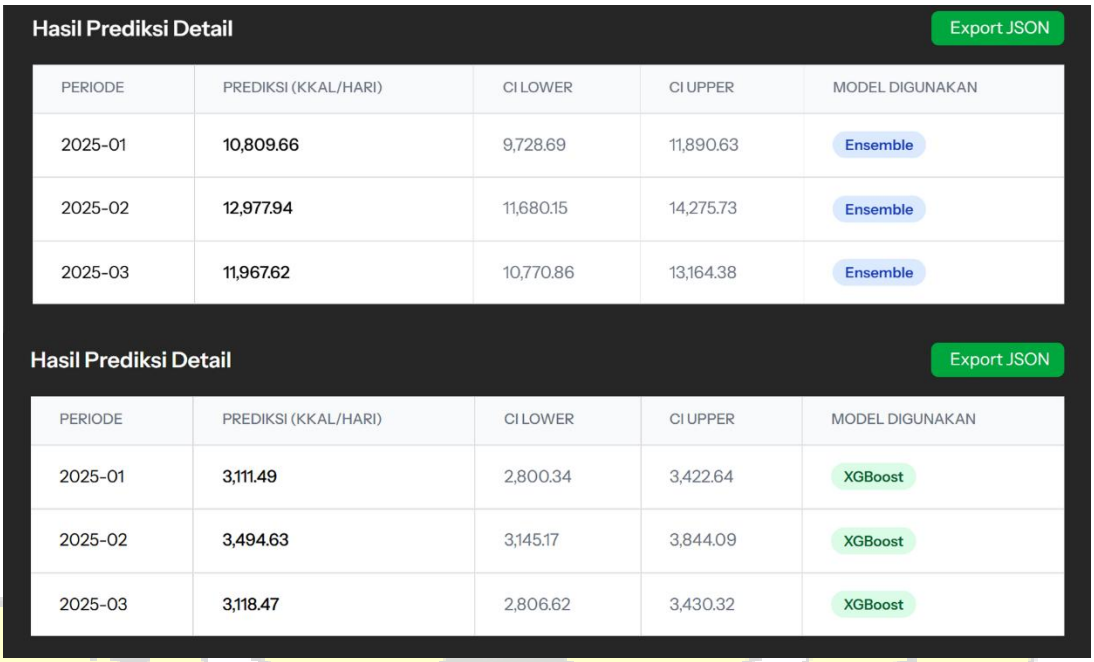


Gambar 22. Hasil Prediksi di *Machine Learning* Lokal

Untuk Beras (kode 0102), hasil *training* menunjukkan nilai kalori per kapita per hari: Januari 10.809,66, Februari 12.977,94, dan Maret 11.967,62 dengan model *Ensemble*. Hasil prediksi sistem web menunjukkan nilai identik untuk ketiga bulan tersebut dengan model yang sama, membuktikan konsistensi implementasi model antara lingkungan *training* dan produksi. Untuk Minyak Goreng Sawit (kode 1004), hasil *training* menunjukkan: Januari 3.111,49, Februari 3.494,63, dan Maret 3.118,47 dengan model XGBoost. Hasil sistem web juga menunjukkan nilai yang sama persis, mengkonfirmasi bahwa *pipeline deployment* berhasil mereplikasi logika prediksi tanpa degradasi akurasi.

Gambar 23 menampilkan hasil prediksi dari proses *training* di *machine learning* lokal yang menunjukkan *output console* dengan detail prediksi untuk kedua

komoditas, termasuk kode komoditas, nama, tahun, bulan, tanggal, bahan makanan dalam satuan ton, kalori per kapita per hari, dan model yang digunakan.



Hasil Prediksi Detail — Export JSON

PERIODE	PREDIKSI (KKAL/HARI)	CI LOWER	CI UPPER	MODEL DIGUNAKAN
2025-01	10,809.66	9,728.69	11,890.63	Ensemble
2025-02	12,977.94	11,680.15	14,275.73	Ensemble
2025-03	11,967.62	10,770.86	13,164.38	Ensemble

Hasil Prediksi Detail — Export JSON

PERIODE	PREDIKSI (KKAL/HARI)	CI LOWER	CI UPPER	MODEL DIGUNAKAN
2025-01	3,111.49	2,800.34	3,422.64	XGBoost
2025-02	3,494.63	3,145.17	3,844.09	XGBoost
2025-03	3,118.47	2,806.62	3,430.32	XGBoost

Gambar 23. Hasil Prediksi di SIKOLBIA

Konsistensi 100% antara *training* dan produksi ini memvalidasi keberhasilan implementasi arsitektur *microservices* dengan FastAPI dan integrasi model *machine learning* ke sistem web.

Dari perspektif ilmu komputer dan informatika, penelitian ini memberikan kontribusi signifikan dalam lima aspek: (1) Arsitektur *Ensemble Learning* dengan strategi *conditional ensemble* menggunakan *threshold* adaptif yang mengkombinasikan *machine learning* LSTM dan *machine learning* klasik untuk *time series forecasting* pada data skala heterogen; (2) *Framework Feature Engineering* dengan *pipeline* 39 fitur prediktif yang mengintegrasikan pola temporal, *cyclical encoding*, indikator ekonomi, dan deteksi krisis; (3) Optimasi hiperparameter menggunakan *Differential Evolution* untuk menemukan bobot *ensemble* optimal (LSTM 90%, XGBoost 5%, *Huber* 5%) yang mencapai R^2=0,9898; (4) Sistem *machine learning* produksi dengan arsitektur *microservices* (Laravel dan FastAPI), Docker

containerization, *automated model retraining*, dan *continuous learning*; (5) *Pipeline deployment end-to-end* dari *data preprocessing*, *model training*, hingga RESTful API yang membuktikan kelayakan *ensemble learning* untuk aplikasi *real-time forecasting* ketahanan pangan. Metodologi ini dapat diadaptasi untuk prediksi *time series* di domain keuangan, kesehatan, energi, dan *climate forecasting*.