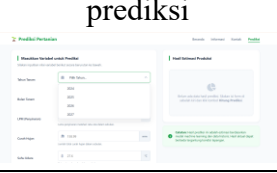
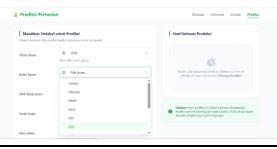
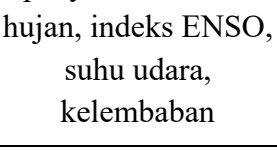
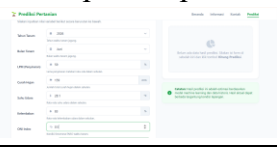


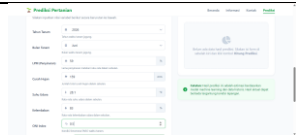
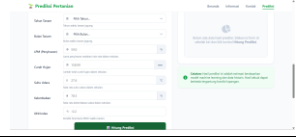
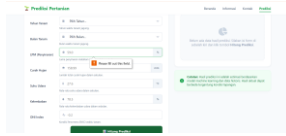
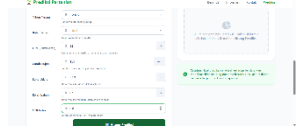
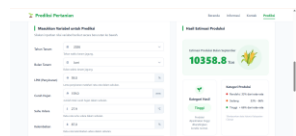
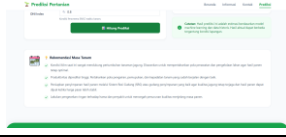
LAMPIRAN

Lampiran 1 *Black Box Testing*

a. Pengujian User

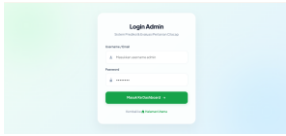
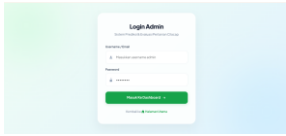
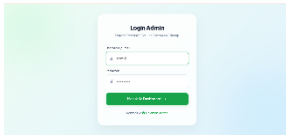
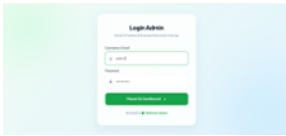
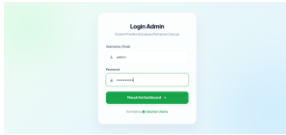
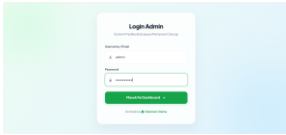
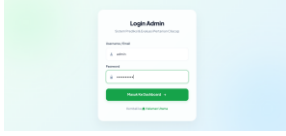
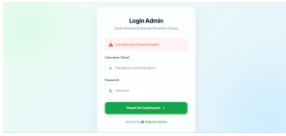
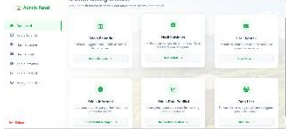
Tabel 11 Hasil Pengujian User

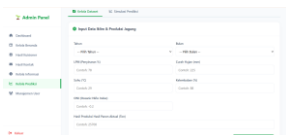
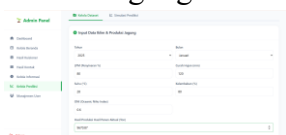
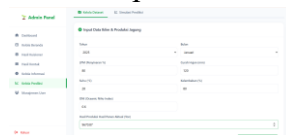
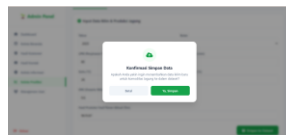
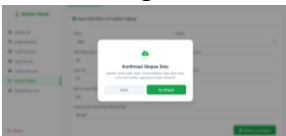
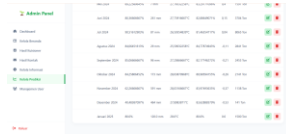
Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
Akses beranda	Mengakses halaman website 	Menampilkan halaman beranda 	Berhasil
Akses menu prediksi	Klik menu “Prediksi” 	Menampilkan pilihan komoditas prediksi 	Berhasil
Akses prediksi Jagung	Klik komoditas “Jagung” 	Menampilkan halaman prediksi Jagung 	Berhasil
Input tahun tanam	Mengisi kolom tahun prediksi 	Data tahun dapat diinput 	Berhasil
Input bulan tanam	Mengisi kolom bulan tanam 	Data bulan dapat diinput 	Berhasil
Input variabel iklim	Mengisi lama penyinaran, curah hujan, indeks ENSO, suhu udara, kelembaban 	Data variabel iklim dapat diinput 	Berhasil

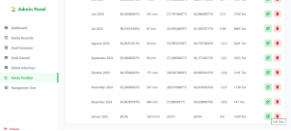
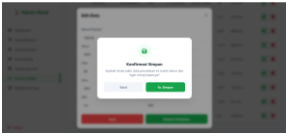
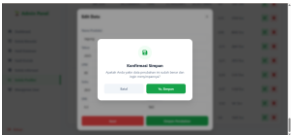
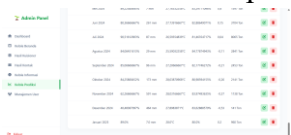
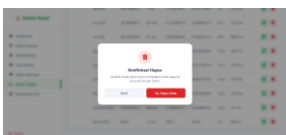
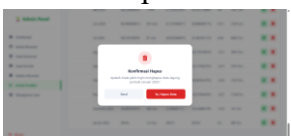
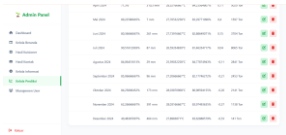
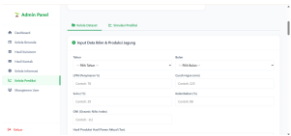
Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
			
Validasi input tidak lengkap	Klik tombol “Hitung Prediksi” tanpa melengkapi data 	Sistem menampilkan pesan error 	Berhasil
Proses prediksi	Klik tombol “Hitung Prediksi” dengan data lengkap 	Sistem memproses data dan menampilkan hasil prediksi 	Berhasil
Tampilan hasil prediksi	Setelah prediksi berhasil 	Sistem menampilkan estimasi bulan panen dan prediksi panen dalam ton 	Berhasil
Tampilan kategori prediksi	Setelah prediksi berhasil 	Sistem menampilkan keterangan sesuai kategori produksi (rendah, sedang, atau tinggi) 	Berhasil
Tampilan rekomendasi	Setelah prediksi berhasil 	Sistem menampilkan hasil prediksi dan rekomendasi 	Berhasil

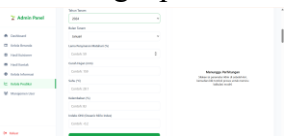
b. Pengujian Admin

Tabel 12 Hasil Pengujian Admin

Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
Login Admin	Masuk ke halaman login admin 	Menampilkan halaman login admin 	Berhasil
Input <i>username</i>	Mengisi kolom <i>username</i> 	<i>Username</i> dapat diinput 	Berhasil
Input password	Mengisi kolom password 	<i>Password</i> dapat diinput 	Berhasil
Validasi salah inputan	Klik tombol “Masuk ke Dashbord” dengan data yang salah 	Sistem menampilkan pesan eror 	Berhasil
Akses dashboard admin	Klik tombol “Masuk ke Dashbord” dengan data yang benar 	Menampilkan dashboard admin 	Berhasil

Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
Akses kelola prediksi	Klik menu “Kelola Prediksi” 	Menampilkan halaman kelola prediksi 	Berhasil
Akses kelola prediksi Jagung	Klik pilihan kelola prediksi Jagung 	Menampilkan halaman kelola prediksi Jagung 	Berhasil
Akses kelola dataset	Klik pilihan “Kelola Dataset” 	Menampilkan halaman kelola dataset 	Berhasil
Input data baru	Mengisi kolom tahun, bulan, prediksi, variabel iklim, dan produksi Jagung 	Data tahun, bulan, prediksi, variabel iklim, dan produksi Jagung dapat diinputkan 	Berhasil
Konfirmasi simpan data baru	Klik button “Simpan ke Dataset” 	Sistem menampilkan konfirmasi 	Berhasil
Simpan dataset baru	Klik button “Ya, Simpan” 	Data tersimpan di database dan riwayat dataset terdaftar 	Berhasil

Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
Akses Edit data	Pilih salah satu data lalu klik tombol edit 	Menampilkan data yang akan diedit 	Berhasil
Melakukan edit data	Mengganti data yang ada dengan data baru 	Data dapat diubah ke data baru 	Berhasil
Konfirmasi edit data	Klik button “Simpan Perubahan” 	Sistem menampilkan konfirmasi 	Berhasil
Simpan data	Klik button “Simpan” 	Data tersimpan di <i>database</i> dan riwayat dataset terdaftar 	Berhasil
Akses hapus data	Pilih salah satu data lalu klik tombol hapus 	Sistem menampilkan konfirmasi 	Berhasil
Konfirmasi edit data	Klik “Hapus Data” 	Data terhapus dari <i>database</i> dan riwayat dataset terdaftar 	Berhasil
Akses simulasi prediksi	Klik menu “Simulasi Prediksi” 	Menampilkan halaman simulasi prediksi	Berhasil

Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
			
Input tahun tanam	Mengisi kolom tahun prediksi 	Data tahun dapat diinput 	Berhasil
Input bulan tanam	Mengisi kolom bulan tanam 	Data bulan dapat diinput 	Berhasil
Input variabel iklim	Mengisi lama penyinaran, curah hujan, indeks ENSO, suhu udara, kelembaban 	Data variabel iklim dapat diinput 	Berhasil
Validasi input tidak lengkap	Klik tombol “Hitung Prediksi” tanpa melengkapi data 	Sistem menampilkan pesan error 	Berhasil
Proses prediksi	Klik tombol “Hitung Prediksi” dengan data lengkap 	Sistem memproses data dan menampilkan hasil prediksi 	Berhasil

Skenario Pengujian	Langkah Pengujian	Hasil yang diharapkan	Hasil Pengujian
(1)	(2)	(3)	(4)
Tampilan hasil prediksi	Setelah prediksi berhasil 	Sistem menampilkan estimasi bulan panen dan prediksi panen dalam ton 	Berhasil
Tampilan kategori prediksi	Setelah prediksi berhasil 	Sistem menampilkan keterangan sesuai kategori produksi (rendah, sedang, atau tinggi) 	Berhasil
Tampilan rekomendasi	Setelah prediksi berhasil 	Sistem menampilkan hasil prediksi dan rekomendasi 	Berhasil

Lampiran 2 Kode Model XGBoost Baseline

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

# =====
# LOAD DATA
# =====
df = pd.read_excel("data jagung.xlsx")

# =====
# 2. PROSES LAGGING (SEMUA VARIABEL IKLIM)
# =====
kolom_fitur = ['LPM (%)', 'CH (mm)', 'ONI', 'Suhu (°C)', 'Kelembaban (%)']

for col in kolom_fitur:
    df[f'{col}_Lag1'] = df[col].shift(1)
    df[f'{col}_Lag2'] = df[col].shift(2)
    df[f'{col}_Lag3'] = df[col].shift(3)

df_clean = df.dropna().reset_index(drop=True)

# =====
# FITUR (X) & TARGET (y)
# =====
features = [col for col in df_clean.columns if 'Lag' in col]
X = df_clean[features]
y = df_clean['P (ton)']

# =====
# 4. SPLIT DATA
# =====
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42 )

# =====
# MODEL XGBOOST
# =====
xgb = XGBRegressor(
    n_estimators=100,
```

```

        learning_rate=0.3,
        max_depth=6,
        subsample=1,
        colsample_bytree=1,
        random_state=42
    )
xgb.fit(X_train, y_train)

# =====
# PREDIKSI SELURUH DATA (57 BARIS)
# =====
y_all_pred = xgb.predict(X)

# =====
# 7. EVALUASI (SELURUH DATA)
# =====
r2_all = r2_score(y, y_all_pred)
mae_all = mean_absolute_error(y, y_all_pred)
rmse_all = np.sqrt(mean_squared_error(y, y_all_pred))

# =====
# VISUALISASI GRAFIK 57 BULAN
# =====
date_range = pd.date_range(start='2020-04-01', periods=len(y),
                             freq='MS')
date_labels = [d.strftime('%b-%y') for d in date_range]

plt.figure(figsize=(16, 8))

plt.plot(date_labels, y.values,
         label='Data Aktual (Asli)',
         color='#1f77b4',
         marker='o',
         linewidth=2)

plt.plot(date_labels, y_all_pred,
         label='Prediksi XGBoost',
         color='#d62728',
         linestyle='--',
         marker='s',
         alpha=0.8)

info_box = (f"Statistik Model (57 Bulan):\n"
            f"R2 Score: {round(r2_all, 3)}\n"
            f"MAE: {round(mae_all, 2)} Ton\n"
            f"RMSE: {round(rmse_all, 2)} Ton")

plt.gca().text(0.02, 0.95,
               info_box,
               transform=plt.gca().transAxes,

```

```

        fontsize=12,
        verticalalignment='top',
        bbox=dict(boxstyle='round', facecolor='white',
alpha=0.9))

plt.title('Grafik Prediksi Produksi Jagung (Baseline)',
        fontsize=16, pad=20)

plt.xlabel('Periode Waktu (Bulan-Tahun)', fontsize=12)
plt.ylabel('Produksi Jagung (Ton)', fontsize=12)
plt.xticks(rotation=45, fontsize=9)
plt.legend(loc='upper right', fontsize=11)
plt.grid(True, linestyle=':', alpha=0.6)

plt.tight_layout()

# --- BAGIAN MODIFIKASI PENYIMPANAN ---
# 1. Memastikan folder 'grafik' ada di direktori kerja saat ini
if not os.path.exists("grafik"): os.makedirs("grafik")

plt.show()
plt.close()

# =====
# OUTPUT HASIL DI TERMINAL
# =====
print("\n" + "="*40)
print(f"HASIL AKHIR R²: {r2_all:.3f}")
print(f"HASIL AKHIR MAE: {mae_all:.2f} Ton")
print(f"HASIL AKHIR RMSE: {rmse_all:.2f} Ton")
print("="*40)

# =====
# FEATURE IMPORTANCE
# =====
importances = pd.Series(xgb.feature_importances_, index=features)
importances = importances.sort_values(ascending=False)

print("\nTOP 5 VARIABEL PALING BERPENGARUH:")
print(importances.head(5) * 100)

```

Lampiran 3 Kode Model Optimasi XGBoost

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

# =====
# LOAD DATA & PREPROCESSING
# =====
df = pd.read_excel("data jagung.xlsx")
kolom_fitur = ['LPM (%)', 'CH (mm)', 'ONI', 'Suhu (°C)', 'Kelembaban (%)']

for col in kolom_fitur:
    df[f'{col}_Lag1'] = df[col].shift(1)
    df[f'{col}_Lag2'] = df[col].shift(2)
    df[f'{col}_Lag3'] = df[col].shift(3)

df_clean = df.dropna().reset_index(drop=True)
features = [col for col in df_clean.columns if 'Lag' in col]
X = df_clean[features]
y = df_clean['P (ton)']

# Split data (shuffle=False untuk menjaga urutan waktu)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False, )

# =====
# RANDOM SEARCH OPTIMIZATION
# =====
param_dist = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [3, 4, 5, 6],
    'learning_rate': [0.01, 0.05, 0.1, 0.3],
    'subsample': [0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.7, 0.8, 0.9, 1.0]
}
xgb_base = XGBRegressor(random_state=42, n_jobs=1)
search = RandomizedSearchCV(
    estimator=xgb_base,
    param_distributions=param_dist,
    n_iter=20,
    cv=3,
    scoring='r2',
```

```

        n_jobs=-1, # Menggunakan semua core
        random_state=42
    )
search.fit(X_train, y_train)

# Ambil model terbaik
model = search.best_estimator_
print("Best Params:", search.best_params_)
y_all_pred = model.predict(X)

# =====
# EVALUASI (SELURUH DATA)
# =====
r2_all = r2_score(y, y_all_pred)
mae_all = mean_absolute_error(y, y_all_pred)
rmse_all = np.sqrt(mean_squared_error(y, y_all_pred))

# =====
# VISUALISASI
# =====
date_range = pd.date_range(start='2020-04-01', periods=len(y),
                             freq='MS')
date_labels = [d.strftime('%b-%y') for d in date_range]

plt.figure(figsize=(16, 8))
plt.plot(date_labels, y.values, label='Data Aktual (Asli)',
         color='#1f77b4', marker='o', linewidth=2)
plt.plot(date_labels, y_all_pred, label='Prediksi XGBoost (Optimasi)',
         color='#d62728', linestyle='--', marker='s', alpha=0.8)

y_combined = np.concatenate([y.values, y_all_pred])
y_min = y_combined.min()
y_max = y_combined.max()

# Memberikan sedikit 'ruang' (padding) sebesar 10% agar grafik tidak
# menempel di tepi atas/bawah
padding = (y_max - y_min) * 0.1
plt.ylim(y_min - padding, y_max + padding)

info_box = (f"Statistik Model (Optimasi):\n"
            f"R2 Score: {round(r2_all, 3)}\n"
            f"MAE: {round(mae_all, 2)} Ton\n"
            f"RMSE: {round(rmse_all, 2)} Ton")

plt.gca().text(0.02, 0.95, info_box, transform=plt.gca().transAxes,
               fontsize=12,
               verticalalignment='top', bbox=dict(boxstyle='round',
               facecolor='white', alpha=0.9))

```

```

plt.title('Grafik Prediksi Produksi (Optimasi Random Search)',
          fontsize=16, pad=20)
plt.xlabel('Periode Waktu', fontsize=12)
plt.ylabel('Produksi (Ton)', fontsize=12)
plt.xticks(rotation=45, fontsize=9)
plt.legend(loc='upper right', fontsize=11)
plt.grid(True, linestyle=':', alpha=0.6)
plt.tight_layout()

if not os.path.exists("grafik"): os.makedirs("grafik")
plt.savefig("grafik/grafik_jagung_optimasi_RS.png", dpi=300,
           bbox_inches='tight')
plt.show()

# =====
# OUTPUT & FEATURE IMPORTANCE
# =====
print(f"\nHASIL AKHIR R2: {r2_all:.3f}")
print(f"HASIL AKHIR MAE: {mae_all:.2f} Ton")
print(f"HASIL AKHIR RMSE: {rmse_all:.2f} Ton")
importances = pd.Series(model.feature_importances_,
                        index=features).sort_values(ascending=False)
print("\nTOP 5 VARIABEL:\n", importances.head(5) * 100)

```

Lampiran 4 Kode train-jagung.py

```
import os
import sys
import joblib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from supabase import create_client, Client
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

url = "https://ooytznctvpdjsvbkpqtd.supabase.co"
key = "sb_publishable_jtHY1NMtcfh1Bzjr3nOpBA_2fVODiy8"
supabase = create_client(url, key)

def update_metadata_to_supabase(supabase_client, komoditas, r2,
model_name):
    # R2 diubah ke string persentase
    r2_str = f"{r2 * 100:.2f}%"

    # Update dokumen metadata/akurasi_model
    data = {
        "komoditas": komoditas.lower(),
        "r2": r2_str,
        "model_name": model_name
    }
    supabase_client.table("metadata").upsert(data,
on_conflict="komoditas").execute()
    print(f"✓ Metadata {komoditas} di Supabase telah diperbarui")

def jalankan_training_jagung(supabase_client: Client):
    """
    Fungsi core untuk melakukan retraining model jagung menggunakan
    objek supabase_client
    yang dikirimkan langsung dari server FastAPI utama (main.py).
    """
    # Menentukan lokasi absolut skrip
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))
    CHART_DIR = os.path.join(BASE_DIR, "..", "static", "charts")
    MODEL_DIR = os.path.join(BASE_DIR, "..", "models")

    # Buat folder jika belum ada
    for folder in [CHART_DIR, MODEL_DIR]:
        if not os.path.exists(folder):
```

```

        os.makedirs(folder)

    print("--- Mengambil Data Jagung Terbaru dari Supabase ---")
    # Mengambil data dari tabel 'data_jagung' diurutkan berdasarkan
    tahun
    response =
supabase_client.table('data_jagung').select('*').order('tahun').execute()
    data_list = response.data

    if not data_list:
        print("Error: Tidak ada data jagung di Supabase untuk
    ditraining. Periksa kembali dokumen Anda.")
        return False

    df = pd.DataFrame(data_list)

    # Sinkronisasi 1: Paksa nama kolom hasil tarikan Supabase
    menjadi huruf kapital semua
    df.columns = [str(col).strip().upper() for col in df.columns]

    # --- PEMBERSIHAN DATA ---
    def clean_number(value):
        if pd.isna(value): return 0.0
        try:
            # Mengatasi jika data bertipe string dengan format koma
Indonesia (e.g. "159,0" menjadi 159.0)
            if isinstance(value, str):
                value = value.replace('.', '').replace(',', '.')
            return float(value)
        except:
            return 0.0

    # Disesuaikan dengan nama kolom kapital setelah standarisasi
    (PRODUKSI menggantikan P)
    cols_to_clean = ['LPM', 'CH', 'ONI', 'SUHU', 'KELEMBABAN',
    'PRODUKSI']
    for col in cols_to_clean:
        if col in df.columns:
            df[col] = df[col].apply(clean_number)

    # Sinkronisasi 2: Menangani variasi penulisan bulan oleh admin
    (Contoh: "Januari", "januari")
    month_map = {
        'januari': 1, 'februari': 2, 'maret': 3, 'april': 4,
        'mei': 5, 'juni': 6, 'juli': 7, 'agustus': 8,

```

```

        'september': 9, 'oktober': 10, 'november': 11, 'desember':
12    }

    if 'BULAN' in df.columns and 'TAHUN' in df.columns:
        df['BULAN_CLEAN'] =
df['BULAN'].astype(str).str.lower().str.strip()
        df['BULAN_NUM'] = df['BULAN_CLEAN'].map(month_map)
        df['BULAN_NUM'] = df['BULAN_NUM'].fillna(1).astype(int) #
Fallback aman jika kosong

        # [DIPINDAHKAN KE SINI]: Data disorting SEBELUM fitur
lagging dibuat agar runtun
        df = df.sort_values(['TAHUN',
'BULAN_NUM']).reset_index(drop=True)

        # =====
        # 2. PREPROCESSING & LAGGING
        # =====
        kolom_fitur = ['LPM', 'CH', 'ONI', 'SUHU', 'KELEMBABAN']

        # Sekarang fungsi .shift() di bawah ini aman karena datanya
dipastikan sudahurut kronologis
        for col in kolom_fitur:
            df[f'{col}_LAG1'] = df[col].shift(1)
            df[f'{col}_LAG2'] = df[col].shift(2)
            df[f'{col}_LAG3'] = df[col].shift(3)

        # Urutan fitur akhir: LPM_LAG1, LPM_LAG2, LPM_LAG3, CH_LAG1,
CH_LAG2...
        features = []
        for col in kolom_fitur:
            features.extend([f'{col}_LAG1', f'{col}_LAG2',
f'{col}_LAG3'])

        df_clean_g = df.dropna(subset=[f'{col}_LAG3' for col in
kolom_fitur]).reset_index(drop=True)

        t_low = df['PRODUKSI'].quantile(0.33) # Menggunakan tercil
sesuai penyesuaian (0.33)
        t_high = df['PRODUKSI'].quantile(0.66)

        best_params = {
            'n_estimators': 200,
            'max_depth': 3,
            'learning_rate': 0.05,
            'subsample': 1.0,

```

```

        'colsample_bytree': 0.9,
    }

# =====
# 3. PROSES GRAFIK EVALUASI
# =====
print("--- Mengoptimasi Model Grafik Jagung ---")
X_g = df_clean_g[features]
y_g = df_clean_g['PRODUKSI']

# Gunakan train_test_split dengan shuffle=False untuk
mempertahankan urutan waktu historis
X_train_g, X_test_g, y_train_g, y_test_g = train_test_split(X_g,
y_g, test_size=0.2, shuffle=False)

# Fit tunggal & deterministik dengan parameter tetap (lihat
catatan best_params di atas).
# n_jobs=1 + tree_method="exact" memaksa XGBoost single-thread
tanpa randomness
# urutan floating-point antar core, sehingga hasilnya
reproducible.
best_model_grafik = XGBRegressor(
    **best_params,
    random_state=42,
    n_jobs=1,
    tree_method="exact",
)
best_model_grafik.fit(X_train_g, y_train_g)

# Prediksi SELURUH DATA untuk visualisasi grafik historis
y_all_pred = best_model_grafik.predict(X_g)

r2_all = r2_score(y_g, y_all_pred)
mae_all = mean_absolute_error(y_g, y_all_pred)
rmse_all = np.sqrt(mean_squared_error(y_g, y_all_pred))

y_test_pred = best_model_grafik.predict(X_test_g)
r2_test = r2_score(y_test_g, y_test_pred)
print(f"Skor R2 pada data yang tidak dilihat model (Test Set):
{r2_test}")
# =====
# 4. PROSES MODEL APK (SHIFTED T+3)
# =====
print("--- Mengoptimasi Model APK Jagung (Shifted T+3) ---")
df_shifted = df.copy()
df_shifted['TARGET_PRODUKSI'] = df_shifted['PRODUKSI'].shift(-3)

```

```

df_clean_m = df_shifted.dropna(subset=['TARGET_PRODUKSI'] +
[f'{col}_LAG3' for col in kolom_fitur]).reset_index(drop=True)
X_m = df_clean_m[features]
y_m = df_clean_m['TARGET_PRODUKSI']

# Latih ulang parameter terbaik dengan data shift (transfer
learning) untuk kebutuhan APK
best_model_apk = best_model_grafik
best_model_apk.fit(X_m, y_m)

# Simpan bundle model.pkl resmi jagung
joblib.dump({
    'model': best_model_apk,
    'features': features,
    'best_hyperparameters': best_params,
    'thresholds': {'q1': float(t_low), 'q2': float(t_high)}
}, os.path.join(MODEL_DIR, "model_jagung_shifted.pkl"))

# =====
# 5. VISUALISASI GRAFIK DI DASHBOARD WEB
# =====
df_clean_g['TANGGAL'] =
pd.to_datetime(df_clean_g['TAHUN'].astype(str) + '-' +
df_clean_g['BULAN_NUM'].astype(str) + '-1')
date_labels = df_clean_g['TANGGAL'].dt.strftime('%b-%y').values

plt.figure(figsize=(16, 8))
x_indexes = np.arange(len(y_g))

# Plot Data Aktual dan Prediksi (Sesuai standarisasi visual
teratur)
plt.plot(x_indexes, y_g.values, label='Data Aktual',
color='#1f77b4', marker='o', linewidth=1.5)
plt.plot(x_indexes, y_all_pred, label='Prediksi XGBoost
(Optimasi)', color='#d62728', linestyle='--', marker='s', alpha=0.8,
linewidth=1.5)

plt.xticks(x_indexes, date_labels, rotation=45, fontsize=8,
ha='right')

# Kotak Informasi Statistik (Info Box)
info_box = (f"Statistik Model Jagung:\n"
f"R²: {round(r2_all, 3)}\n"
f"MAE: {round(mae_all, 2)} Ton\n"
f"RMSE: {round(rmse_all, 2)} Ton")

```

```

plt.gca().text(0.02, 0.95, info_box,
transform=plt.gca().transAxes, fontsize=12,
               verticalalignment='top',
bbox=dict(boxstyle='round', facecolor='white', alpha=0.9))

# Judul dan Labeling
plt.title('Grafik Analisis Histori: Produksi Jagung',
          fontsize=14, pad=15)
plt.xlabel('Periode', fontsize=10)
plt.ylabel('Produksi (Ton)', fontsize=10)
plt.legend(loc='upper right')

plt.grid(True, linestyle='-', alpha=0.5)
plt.tight_layout()

# Simpan hasil grafik png untuk dashboard web
plt.savefig(os.path.join(CHART_DIR,
"grafik_prediksi_jagung.png"), dpi=150)
plt.close()

# Update hasil performa R2 ke tabel metadata Supabase
update_metadata_to_supabase(supabase_client, "Jagung", r2_all,
"Optimasi XGBoost")

print(f"✓ Training Jagung Selesai & Berhasil Disimpan. R²:
{r2_all:.3f}")
print("=====")
print("  THRESHOLD PRODUKSI (HASIL TRAINING)")
print(f"  Batas Rendah (Q1/33%): {t_low:.2f} Ton")
print(f"  Batas Tinggi (Q2/66%): {t_high:.2f} Ton")
print("=====")
return True

# Digunakan untuk pengujian mandiri jika file ini dijalankan
langsung (bukan via main.py)
if __name__ == "__main__":
    SUPABASE_URL = "https://ooytznctvpdjsvbkpqtd.supabase.co"
    SUPABASE_KEY = "sb_publishable_jtHY1NMtcfh1Bzjr3n0pBA_2fVODiy8"

    test_supabase = create_client(SUPABASE_URL, SUPABASE_KEY)
    jalankan_training_jagung(test_supabase)

```