# BUKU PANDUAN

## Program Komputer Model *Random Forest* dengan *Bayesian Hyperparameter Tuning* (BO-RF) untuk Analisis Pergerakan dan Prediksi Harga Saham


**Pencipta:**

Martino Kristian Samuel Simatupang

Dr. Sutrisno S.Si., M.Sc.

Ratna Herdiana M.Sc., Ph.D.



**UNIVERSITAS DIPONEGORO**

**SEMARANG**

**2026**

**DAFTAR ISI**

## A. Informasi Program Komputer

Pasar saham memiliki karakteristik yang sangat dinamis, non-linear, dan bervolatilitas tinggi. Dalam praktik analisis dan prediksi harga saham, metode peramalan konvensional kerap kali tidak efektif dalam menangkap pola tersembunyi pada data runtun waktu (*time series*) saham yang fluktuatif. Dengan kemajuan teknologi, algoritma *machine learning* seperti *Random Forest* hadir dan menawarkan kemampuan prediksi yang kuat, dibersamai dengan proses optimasi tambahan yang mampu meningkatkan kualitas dari kinerja algoritma *Random Forest.* Dengan mengintegrasikan algoritma *Random Forest* dengan strategi *Bayesian Optimization Hyperparameter Tuning,* proses penentuan nilai optimum dari parameter algoritma *Random Forest* akan berjalan efisien melalui pendekatan probalistik yang cerdas, sekaligus meminimalisir *error* dalam proses prediksi harga saham harian. Model BO-RF ini dikembangkan sebagai alat bantu pengambilan keputusan calon investor dalam melakukan analisis teknikal pasar saham dengan melakukan prediksi harga penutupan harga saham (*Close Price*).

Buku panduan ini merupakan dokumen teknis dalam mengoperasikan Model *Random Forest* dengan *Bayesian Optimization Hyperparameter Tuning* (BO-RF) untuk menganalisis pergerakan dan prediksi harga saham harian dituangkan secara lengkap.

## B. Keunggulan Program Komputer

Model BO-RF mengintegrasikan dua komponen utama yaitu algoritma *Random Forest* dan metode *Bayesian Optimization* dalam pemilihan nilai *hyperparameter* optimum dari *Random Forest*. Penggunaan model BO-RF terbukti mampu diandalkan dalam melakukan analisis pergerakan dan prediksi harga saham didasari pada pengujian empiris (Studi kasus harga saham harian PT Bank Central Asia Tbk). Dalam pengujian empiris tersebut, Model BO-RF mampu mencapai nilai R-squared (Koefisien Determinasi) sebesar 96,33 dan MAPE (*Mean Absolute Percentage Error*) sebesar 1,23%. Meskipun model ini dikembangkan menggunakan studi kasus saham lokal, arsitektur dari model ini bersifat universal

melalui *pipeline* yang telah dibangun. Program dapat menerima input data historis harga saham perusahaan mana pun di seluruh dunia, apabila memuat lima variabel teknikal yaitu Harga Pembukaan (*Open*), Harga Tertinggi (*High*), Harga Terendah (*Low*), Harga Penutupan (*Close*), dan Volume Penjualan (*Volume*).

Arsitektur dari Model BO-RF dibangun berdasarkan *machine learning life cycle*, berupa fase-fase dalam membangun model *machine learning* (*Data Collection, Data Preparation. Feature Engineering, Model Development, Model Evaluation,* dan *Model Postprocessing*) yang direpresentasikan pada *pipeline* pembangun Model BO-RF secara *end-to-end*. Hal ini mempermudah proses implementasi model bagi para penggunanya yang dapat dijalankan secara otomatis, diperlengkapi proses penanganan nilai ekstrem (*outlier*) dalam data olahan, dan proses terotomasi dalam pembuatan variabel-variabel baru (*feature engineering*) seperti RSI, MA, MOM, dan variabel penanda waktu lainnya. Berdasarkan penjelasan tersebut, keunggulan dari Model BO-RF ialah akurasi tinggi, fleksibilitas global, ketahanan terhadap *outlier,* dan otomatisasi penuh.

## C. Kebaruan Program Komputer

Program dari Model BO-RF ini menawarkan nilai kebaruan yang membedakannya dari metode-metode lainnya dalam menganalisis harga saham. Kebaruan utama terletak pada penerapan algoritma *Bayesian Optimization* dengan metode *Upper Confidence Bound* (UCB) dalam proses *hyperparameter tuning*. Berbeda dengan algoritma *Grid Search* atau *Random Search* yang memakan waktu dan komputasi besar, algoritma *Bayesian Optimization* bekerja dalam iterasi tertentu dan mampu belajar dari iterasi sebelumnya guna menemukan nilai *hyperparameter* optimum secara presisi dan lebih cepat. Penggunaan algoritma *Random Forest* sebagai pembelajar data yang utama, menjadikan model BO-RF memiliki keunggulan dalam penyediaan *Feature Importance*. Program ini mampu memberitahu penggunanya variabel-variabel mana saja yang paling mempengaruhi harga saham yang dianalisis, tidak seperti model *Deep Learning* seperti LSTM yang terbatas dalam hal ini.

## D. Penjelasan Program Komputer

Bagian ini disusun sebagai panduan teknis dalam menjalankan program Model BO-RF yang dibangun dengan bahasa pemrograman Python. Penjelasan dirancang sistematis, argumentatif, dan memenuhi standar untuk kebutuhan HKI Hak Cipta.

1. Persiapan Lingkungan

   Tahapan pertama adalah mengimpor pustaka-pustaka (*library*) untuk mempersiapkan lingkungan bahasa pemrograman Python.

```
# Library
import pandas as pd
import numpy as np
from warnings import filterwarnings

# Grafik dan Visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# Indikator Teknikal Saham
import talib

# Machine Learning
from sklearn.ensemble import RandomForestRegressor
from bayes_opt import BayesianOptimization
from sklearn.model_selection import TimeSeriesSplit, cross_val_score
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, root_mean_squared_error, r2_score

filterwarnings('ignore')
plt.rcParams['font.size'] = 14
```

   Untuk kebutuhan operasi matematika dan manipulasi data digunakan fungsi Pandas dan Numpy, kebutuhan grafik dan visualisasi digunakan fungsi Matplotlib dan Seaborn, kebutuhan indikator teknikal saham digunakan fungsi Ta-Lib, kebutuhan *machine learning* dan metrik evaluasi digunakan fungsi Sklearn, dan kebutuhan strategi optimasi berupa *hyperparameter tuning* digunakan fungsi Bayes_Opt.

2. *Pipeline* Model Terotomatisasi

   Tahap kedua adalah memastikan *pipeline-pipeline* pembangun Model BO-RF sesuai dengan kebutuhan pengguna.

## DATA COLLECTION

```
def data_collection(path_data):
    print('--- Proses Data Collection ---')
    # Memasukkan data olahan harga saham harian
    try:
        df = pd.read_excel(path_data, parse_dates=['Date'], index_col='Date')
        print(f'Data berhasil dikoleksi dari {path_data}')
    except Exception as e:
        print(f'Error dalam koleksi data: {e}')
        return None

    # Inspeksi data awal
    print('Ukuran data:', df.shape)
    print('Lima data teratas:')
    display(df.head(5))
    print('Informasi umum data:')
    df.info()

    return df
```

## DATA PREPARATION

```
def data_augmentation(df):
    return df

def feature_engineering(df_awal):
    df_fitur = pd.DataFrame(index=df_awal.index)

    df_fitur['Intraday_Range'] = df_awal['High'] - df_awal['Low']
    df_fitur['Intraday_Change'] = df_awal['Close'] - df_awal['Open']
    df_fitur['MA_5'] = talib.MA(df_awal['Close'], timeperiod=5)
    df_fitur['MA_10'] = talib.MA(df_awal['Close'], timeperiod=10)
    df_fitur['RSI_7'] = talib.RSI(df_awal['Close'], timeperiod=7)
    df_fitur['RSI_14'] = talib.RSI(df_awal['Close'], timeperiod=14)
    df_fitur['MOM_10'] = talib.MOM(df_awal['Close'], timeperiod=10)
    df_fitur['ROC_10'] = talib.ROC(df_awal['Close'], timeperiod=10)
    df_fitur['ATR_14'] = talib.ATR(df_awal['High'], df_awal['Low'], df_awal['Close'], timeperiod=14)
    df_fitur['Volatility_10'] = df_awal['Close_Diff'].rolling(window=10).std()

    df_fitur['Open_t-1'] = df_awal['Open'].shift(1)
    df_fitur['High_t-1'] = df_awal['High'].shift(1)
    df_fitur['Low_t-1'] = df_awal['Low'].shift(1)
    df_fitur['Close_t-1'] = df_awal['Close'].shift(1)

    if 'Volume_Log' in df_awal.columns:
        df_fitur['Volume_Log_t-1'] = df_awal['Volume_Log'].shift(1)

    df_fitur['Close_Diff_t-1'] = df_awal['Close_Diff'].shift(1)

    df_fitur['Intraday_Range_t-1'] = df_fitur['Intraday_Range'].shift(1)
    df_fitur['Intraday_Change_t-1'] = df_fitur['Intraday_Change'].shift(1)
    df_fitur['MA_5_t-1'] = df_fitur['MA_5'].shift(1)
    df_fitur['MA_10_t-1'] = df_fitur['MA_10'].shift(1)
    df_fitur['RSI_7_t-1'] = df_fitur['RSI_7'].shift(1)
    df_fitur['RSI_14_t-1'] = df_fitur['RSI_14'].shift(1)
    df_fitur['MOM_10_t-1'] = df_fitur['MOM_10'].shift(1)
    df_fitur['ROC_10_t-1'] = df_fitur['ROC_10'].shift(1)
    df_fitur['ATR_14_t-1'] = df_fitur['ATR_14'].shift(1)
    df_fitur['Volatility_10_t-1'] = df_fitur['Volatility_10'].shift(1)

    df_fitur['Year'] = df_fitur.index.year
    df_fitur['Month'] = df_fitur.index.month
    df_fitur['Week'] = df_fitur.index.isocalendar().week.astype(float)
    df_fitur['DayOfWeek'] = df_fitur.index.dayofweek
    df_fitur['DayOfMonth'] = df_fitur.index.day
    df_fitur['DayOfYear'] = df_fitur.index.dayofyear
    df_fitur['Quarter'] = df_fitur.index.quarter
    df_fitur['IsMonthStart'] = df_fitur.index.is_month_start.astype(int)
    df_fitur['IsMonthEnd'] = df_fitur.index.is_month_end.astype(int)

    df_fitur.dropna(inplace=True)
    return df_fitur

def data_preparation(df):
    print('--- Proses Data Preparation ---')
    df_prep = df.copy()
```

## MODEL DEVELOPMENT

```
def model_development(X_train, y_train):
    print('--- Proses Model Development ---')

    pbounds = {'n_estimators': (200, 1000),
               'max_depth': (5, 50),
               'min_samples_leaf': (1, 15),
               'min_samples_split': (2, 20),
               'max_features': (0.1, 1.0)}

    def rf_cv_score(n_estimators, max_depth, min_samples_leaf, max_features, min_samples_split):
        try:
            n_estimators = int(n_estimators)
            max_depth = int(max_depth)
            min_samples_leaf = int(min_samples_leaf)
            min_samples_split = int(min_samples_split)

            model = RandomForestRegressor(n_estimators=n_estimators,
                                          max_depth=max_depth,
                                          min_samples_leaf=min_samples_leaf,
                                          max_features=max_features,
                                          min_samples_split=min_samples_split,
                                          random_state=42,
                                          n_jobs=1)

            tscv = TimeSeriesSplit(n_splits=5)
            scores = cross_val_score(model, X_train, y_train, cv=tscv, scoring='neg_root_mean_squared_error', n_jobs=1)
            return np.mean(scores)

        except Exception as e:
            print(f"Error: {e}")
            return -1e9

    optimizer = BayesianOptimization(f=rf_cv_score, pbounds=pbounds, random_state=42, verbose=2)
    optimizer.maximize(init_points=5, n_iter=50)

    best_params = optimizer.max['params']
    best_params_formatted = {'n_estimators': int(best_params['n_estimators']),
                             'max_depth': int(best_params['max_depth']),
                             'min_samples_leaf': int(best_params['min_samples_leaf']),
                             'min_samples_split': int(best_params['min_samples_split']),
                             'max_features': best_params['max_features']}

    print(f'Best Hyperparameters: {best_params_formatted}')

    best_rf_model = RandomForestRegressor(**best_params_formatted, random_state=42, n_jobs=1)
    best_rf_model.fit(X_train, y_train)

    return best_rf_model
```

## MODEL EVALUATION

```
def model_evaluation(model, X_test, y_test):
    print('--- Proses Model Evaluation ---')

    y_pred_arr = model.predict(X_test)
    y_pred = pd.Series(y_pred_arr, index=X_test.index, name='Pred_Close_Diff')

    rmse = root_mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print('RMSE:', rmse)
    print('MAE:', mae)
    print('R-squared:', r2)

    plt.figure(figsize=(12, 8))
    plt.plot(y_test.index, y_test, label='Close_Diff Aktual', color='blue')
    plt.plot(y_test.index, y_pred, label='Hasil Prediksi Close_Diff (BO-RF)', color='red', linestyle='--')
    plt.xlabel('Date')
    plt.ylabel('Close_Diff')
    plt.legend()
    plt.show()

    # Feature Importance dalam Model BO-RF
    importances = pd.Series(model.feature_importances_, index=variabel_input)
    importances_sorted = importances.sort_values(ascending=False)

    plt.figure(figsize=(10, 10))
    ax = sns.barplot(x=importances_sorted.values, y=importances_sorted.index, palette="viridis")
    for i, v in enumerate(importances_sorted.values):
        ax.text(v + 0.001, i, f"{v:.4f}", color='black', va='center')
    plt.title("Feature Importance")
    plt.show()

    return y_pred
```

MODEL POSTPROCESSING

```
def model_postprocessing(model, X_test, y_pred, df_keseluruhan, variabel_input):
    print('--- Proses Model Post-Processing ---')

    # Inversi ke skala harga penutupan 'Close'
    close_t_minus_1_test = df_keseluruhan.loc[X_test.index, 'Close_t-1']
    close_actual_test = df_keseluruhan.loc[X_test.index, 'Close']
    pred_close = close_t_minus_1_test + y_pred

    hasil_df = pd.DataFrame({'Close': close_actual_test,
                             'Close_t-1': close_t_minus_1_test,
                             'Pred_Close_Diff': y_pred,
                             'Pred_Close': pred_close},
                            index=X_test.index)

    # Metrik Evaluasi terhadap data terinversi
    rmse_new = root_mean_squared_error(hasil_df['Close'], hasil_df['Pred_Close'])
    mae_new = mean_absolute_error(hasil_df['Close'], hasil_df['Pred_Close'])
    mape_new = mean_absolute_percentage_error(hasil_df['Close'], hasil_df['Pred_Close'])
    r_squared_new = r2_score(hasil_df['Close'], hasil_df['Pred_Close'])

    print('RMSE (Terinversi):', rmse_new)
    print('MAE (Terinversi):', mae_new)
    print('MAPE (Terinversi):', mape_new)
    print('R-squared (Terinversi):', r_squared_new)

    # Visualisasi data aktual dan data terinversi
    plt.figure(figsize=(12, 8))
    plt.plot(hasil_df.index, hasil_df['Close'], label='Actual Close', color='blue')
    plt.plot(hasil_df.index, hasil_df['Pred_Close'], label='Predicted Close (BO-RF)', color='red', linestyle='--')
    plt.title("Actual vs Predicted Close Price")
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()

    display(hasil_df.head(10))
    return hasil_df
```

Model BO-RF dibangun berdasarkan *machine learning life cycle* antara lain *Data Collection, Data Preparation* (memuat *data cleaning, data transformation*), *feature engineering,* dan *data augmentation*), *Model Development, Model Evaluation,* dan *Model Postprocessing*. Pengguna mampu secara langsung mengaplikasikan *pipeline-pipeline* yang tersedia, tetapi tidak membatasi perubahan pada bagian tertentu di dalam pipeline yang dapat disesuaikan oleh kebutuhan pengguna.

Pada pipeline *Data Collection,* pengguna dalam mengubah ‘pd.read_excel’ menjadi ‘pd.read_csv’ apabila pengguna menginginkan input data olahan berformat .csv, serta inspeksi data awal yang semula ‘df.head(5)’ dapat diubah sebanyak data yang ingin dilihat pengguna dalam DataFrame (data disajikan dalam bentuk tubular). Pada pipeline *Data Preparation,* pengguna dapat mengubah bentuk strategi penanganan *missing value* pada variabel Volume pada fungsi definisi ‘data_preparation(df)’ bagian *Data Cleaning & Data Transformation*, menghapus atau menambahkan variabel pada fungsi definisi ‘feature_engineering(df_awal)’, mengkostumisasi pilihan variabel yang akan digunakan sebagai variabel input pada Model BO-RF serta komposisi data latih

dan data uji pada fungsi definisi ‘data_preparation(df)’ bagian *Feature Selection & Data Splitting*. Pada pipeline *Model Development*, pengguna dapat menyesuaikan *hyperparameter* target dari model *Random Forest* yang nantinya akan dipotimasi oleh algoritma *Bayesian Optimization Hyperparameter Tuning* beserta batasan nilai dari masing-masing *hyperparameter* target pada fungsi definisi ‘model_development(X_train, y_train)’ dan ‘rf_cv_score’ serta jumlah iterasi awal dan jumlah iterasi maksimum dari algoritma *Bayesian Optimization* dapat diubah pada ‘init_points’ dan ‘n_iter’. Pada pipeline *Model Evaluation* dan *Model Postprocessing*, pengguna dapat mengurangi atau menambahkan metrik evaluasi yang ingin digunakan serta mengkostumisasi ukuran dan detil hasil visualisasi data.

3. Eksekusi Model BO-RF

```
# Lokasi penyimpanan file data harga saham harian
path_data = r"Masukkan Path Data Anda"

# Data Collection
df_awal = data_collection(path_data)

if df_awal is not None:
    # Data Preparation
    X_train, y_train, X_test, y_test, df_keseluruhan, variabel_input = data_preparation(df_awal)

    # Model Development
    best_model = model_development(X_train, y_train)

    # Model Evaluation
    y_pred_diff = model_evaluation(best_model, X_test, y_test)

    # Model Postprocessing
    df_results = model_postprocessing(best_model, X_test, y_pred_diff, df_keseluruhan, variabel_input)
```

Langkah ketiga ialah menjalankan bagian Main Flow untuk menjalankan setiap pipeline Model BO-RF secara berturut dan otomatis. Dimulai dengan mengunduh data harga saham harian dari saham suatu perusahaan dalam bentuk file excel (.xlsx) yang memuat variabel OHLCV: Harga Pembukaan (*Open*), Harga Tertinggi (*High*), Harga Terendah (*Low*), Harga Penutupan (*Close*), dan Volume Penjualan (*Volume*); dengan indeks waktu Harian (*Date*) melalui website seperti https://finance.yahoo.com/ atau https://bulkstockdatadownloader.app/. Kemudian, letak penyimpanan (*path data*) file data historis harga saham tersebut diletakkan pada fungsi ‘path_data’. Setelah itu, pengguna dapat menunggu Model BO-RF bekerja kemudian mendapatkan output model.

## E. Listing Program dan Contoh Output Model BO-RF untuk Analisis Pergerakan dan Prediksi Harga Saham Harian

Bagian ini menyajikan listing program sekaligus contoh output dari Model BO-RF dengan data input berupa data harga saham harian PT Bank Central Asia Tbk dalam jangka waktu 01-10-2019 sampai dengan 01-10-2025.

### MODEL RANDOM FOREST DENGAN BAYESIAN OPTIMIZATION HYPERPARAMETER TUNING (BO-RF) UNTUK ANALISIS PERGERAKAN DAN PREDIKSI HARGA SAHAM

#### IMPORT LIBRARY

```
# Install Library yang dibutuhkan
#%pip install TA-Lib
#%pip install bayesian-optimization
```

```
# Library
import pandas as pd
import numpy as np
from warnings import filterwarnings

# Grafik dan Visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# Indikator Teknikal Saham
import talib

# Machine Learning
from sklearn.ensemble import RandomForestRegressor
from bayes_opt import BayesianOptimization
from sklearn.model_selection import TimeSeriesSplit, cross_val_score
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, root_mean_squared_error, r2_score

filterwarnings('ignore')
plt.rcParams['font.size'] = 14
```

#### DATA COLLECTION

```
def data_collection(path_data):
    print('--- Proses Data Collection ---')
    # Memasukkan data olahan harga saham harian
    try:
        df = pd.read_excel(path_data, parse_dates=['Date'], index_col='Date')
        print(f'Data berhasil dikoleksi dari {path_data}')
    except Exception as e:
        print(f'Error dalam koleksi data: {e}')
        return None

    # Inspeksi data awal
    print('Ukuran data:', df.shape)
    print('Lima data teratas:')
    display(df.head(5))
    print('Informasi umum data:')
    df.info()

    return df
```

#### DATA PREPARATION

```
def data_augmentation(df):
    return df

def feature_engineering(df_awal):
    df_fitur = pd.DataFrame(index=df_awal.index)

    df_fitur['Intraday_Range'] = df_awal['High'] - df_awal['Low']
    df_fitur['Intraday_Change'] = df_awal['Close'] - df_awal['Open']
    df_fitur['MA_5'] = talib.MA(df_awal['Close'], timeperiod=5)
    df_fitur['MA_10'] = talib.MA(df_awal['Close'], timeperiod=10)
    df_fitur['RSI_7'] = talib.RSI(df_awal['Close'], timeperiod=7)
    df_fitur['RSI_14'] = talib.RSI(df_awal['Close'], timeperiod=14)
    df_fitur['MOM_10'] = talib.MOM(df_awal['Close'], timeperiod=10)
    df_fitur['ROC_10'] = talib.ROC(df_awal['Close'], timeperiod=10)
    df_fitur['ATR_14'] = talib.ATR(df_awal['High'], df_awal['Low'], df_awal['Close'], timeperiod=14)
    df_fitur['Volatility_10'] = df_awal['Close_Diff'].rolling(window=10).std()

    df_fitur['Open_t-1'] = df_awal['Open'].shift(1)
    df_fitur['High_t-1'] = df_awal['High'].shift(1)
    df_fitur['Low_t-1'] = df_awal['Low'].shift(1)
```

```
    df_fitur['Close_t-1'] = df_awal['Close'].shift(1)

    if 'Volume_Log' in df_awal.columns:
        df_fitur['Volume_Log_t-1'] = df_awal['Volume_Log'].shift(1)

    df_fitur['Close_Diff_t-1'] = df_awal['Close_Diff'].shift(1)

    df_fitur['Intraday_Range_t-1'] = df_fitur['Intraday_Range'].shift(1)
    df_fitur['Intraday_Change_t-1'] = df_fitur['Intraday_Change'].shift(1)
    df_fitur['MA_5_t-1'] = df_fitur['MA_5'].shift(1)
    df_fitur['MA_10_t-1'] = df_fitur['MA_10'].shift(1)
    df_fitur['RSI_7_t-1'] = df_fitur['RSI_7'].shift(1)
    df_fitur['RSI_14_t-1'] = df_fitur['RSI_14'].shift(1)
    df_fitur['MOM_10_t-1'] = df_fitur['MOM_10'].shift(1)
    df_fitur['ROC_10_t-1'] = df_fitur['ROC_10'].shift(1)
    df_fitur['ATR_14_t-1'] = df_fitur['ATR_14'].shift(1)
    df_fitur['Volatility_10_t-1'] = df_fitur['Volatility_10'].shift(1)

    df_fitur['Year'] = df_fitur.index.year
    df_fitur['Month'] = df_fitur.index.month
    df_fitur['Week'] = df_fitur.index.isocalendar().week.astype(float)
    df_fitur['DayOfWeek'] = df_fitur.index.dayofweek
    df_fitur['DayOfMonth'] = df_fitur.index.day
    df_fitur['DayOfYear'] = df_fitur.index.dayofyear
    df_fitur['Quarter'] = df_fitur.index.quarter
    df_fitur['IsMonthStart'] = df_fitur.index.is_month_start.astype(int)
    df_fitur['IsMonthEnd'] = df_fitur.index.is_month_end.astype(int)

    df_fitur.dropna(inplace=True)
    return df_fitur

def data_preparation(df):
    print('--- Proses Data Preparation ---')
    df_prep = df.copy()

    # Data Cleaning & Data Transformation
    if 'Volume' in df_prep.columns:
        df_prep['Volume'] = df_prep['Volume'].ffill().fillna(0)
        df_prep['Volume_Log'] = np.log1p(df_prep['Volume'])
        df_prep = df_prep.drop('Volume', axis=1)

    df_prep['Close_Diff'] = df_prep['Close'].diff()

    # Feature Engineering
    df_fitur = feature_engineering(df_prep)

    # Data Augmentation
    df_keseluruhan = pd.concat([df_prep, df_fitur], axis=1)
    df_keseluruhan.dropna(inplace=True)
    df_keseluruhan = data_augmentation(df_keseluruhan)

    # Feature Selection & Data Splitting
    variabel_input = [col for col in df_keseluruhan.columns if col not in['Open', 'High', 'Low', 'Volume_Log', 'Close', 'Close_Diff',
                                                                          'Intraday_Range', 'Intraday_Change', 'MA_5', 'MA_10', 'RSI_7',
                                                                          'RSI_14', 'MOM_10', 'ROC_10', 'ATR_14', 'Volatility_10',
                                                                          'Open_t-1', 'High_t-1', 'Low_t-1', 'RSI_14_t-1', 'MA_10_t-1',
                                                                          'ROC_10_t-1']]

    X = df_keseluruhan[variabel_input]
    y = df_keseluruhan['Close_Diff']

    train_size = int(len(df_keseluruhan) * 0.8)
    X_train = X.iloc[:train_size]
    X_test = X.iloc[train_size:]
    y_train = y.iloc[:train_size]
    y_test = y.iloc[train_size:]

    print(f'Dimensi X_train: {X_train.shape}')
    print(f'Dimensi X_test: {X_test.shape}')
    print(f'Dimensi y_train: {y_train.shape}')
    print(f'Dimensi y_test: {y_test.shape}')

    return X_train, y_train, X_test, y_test, df_keseluruhan, variabel_input
```

## MODEL DEVELOPMENT

```
def model_development(X_train, y_train):
    print('--- Proses Model Development ---')

    pbounds = {'n_estimators': (200, 1000),
               'max_depth': (5, 50),
               'min_samples_leaf': (1, 15),
               'min_samples_split': (2, 20),
```

```
            'max_features': (0.1, 1.0)}

    def rf_cv_score(n_estimators, max_depth, min_samples_leaf, max_features, min_samples_split):
        try:
            n_estimators = int(n_estimators)
            max_depth = int(max_depth)
            min_samples_leaf = int(min_samples_leaf)
            min_samples_split = int(min_samples_split)

            model = RandomForestRegressor(n_estimators=n_estimators,
                                          max_depth=max_depth,
                                          min_samples_leaf=min_samples_leaf,
                                          max_features=max_features,
                                          min_samples_split=min_samples_split,
                                          random_state=42,
                                          n_jobs=1)

            tscv = TimeSeriesSplit(n_splits=5)
            scores = cross_val_score(model, X_train, y_train, cv=tscv, scoring='neg_root_mean_squared_error', n_jobs=1)
            return np.mean(scores)

        except Exception as e:
            print(f"Error: {e}")
            return -1e9

    optimizer = BayesianOptimization(f=rf_cv_score, pbounds=pbounds, random_state=42, verbose=2)
    optimizer.maximize(init_points=5, n_iter=50)

    best_params = optimizer.max['params']
    best_params_formatted = {'n_estimators': int(best_params['n_estimators']),
                             'max_depth': int(best_params['max_depth']),
                             'min_samples_leaf': int(best_params['min_samples_leaf']),
                             'min_samples_split': int(best_params['min_samples_split']),
                             'max_features': best_params['max_features']}

    print(f'Best Hyperparameters: {best_params_formatted}')

    best_rf_model = RandomForestRegressor(**best_params_formatted, random_state=42, n_jobs=1)
    best_rf_model.fit(X_train, y_train)

    return best_rf_model
```

## MODEL EVALUATION

```
def model_evaluation(model, X_test, y_test):
    print('--- Proses Model Evaluation ---')

    y_pred_arr = model.predict(X_test)
    y_pred = pd.Series(y_pred_arr, index=X_test.index, name='Pred_Close_Diff')

    rmse = root_mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print('RMSE:', rmse)
    print('MAE:', mae)
    print('R-squared:', r2)

    plt.figure(figsize=(12, 8))
    plt.plot(y_test.index, y_test, label='Close_Diff Aktual', color='blue')
    plt.plot(y_test.index, y_pred, label='Hasil Prediksi Close_Diff (BO-RF)', color='red', linestyle='--')
    plt.xlabel('Date')
    plt.ylabel('Close_Diff')
    plt.legend()
    plt.show()

    # Feature Importance dalam Model BO-RF
    importances = pd.Series(model.feature_importances_, index=variabel_input)
    importances_sorted = importances.sort_values(ascending=False)

    plt.figure(figsize=(10, 10))
    ax = sns.barplot(x=importances_sorted.values, y=importances_sorted.index, palette="viridis")
    for i, v in enumerate(importances_sorted.values):
        ax.text(v + 0.001, i, f"{v:.4f}", color='black', va='center')
    plt.title("Feature Importance")
    plt.show()

    return y_pred
```

## MODEL POSTPROCESSING

```
def model_postprocessing(model, X_test, y_pred, df_keseluruhan, variabel_input):
    print('--- Proses Model Post-Processing ---')

    # Inversi ke skala harga penutupan 'Close'
    close_t_minus_1_test = df_keseluruhan.loc[X_test.index, 'Close_t-1']
    close_actual_test = df_keseluruhan.loc[X_test.index, 'Close']
    pred_close = close_t_minus_1_test + y_pred

    hasil_df = pd.DataFrame({'Close': close_actual_test,
                             'Close_t-1': close_t_minus_1_test,
                             'Pred_Close_Diff': y_pred,
                             'Pred_Close': pred_close},
                            index=X_test.index)

    # Metrik Evaluasi terhadap data terinversi
    rmse_new = root_mean_squared_error(hasil_df['Close'], hasil_df['Pred_Close'])
    mae_new = mean_absolute_error(hasil_df['Close'], hasil_df['Pred_Close'])
    mape_new = mean_absolute_percentage_error(hasil_df['Close'], hasil_df['Pred_Close'])
    r_squared_new = r2_score(hasil_df['Close'], hasil_df['Pred_Close'])

    print('RMSE (Terinversi):', rmse_new)
    print('MAE (Terinversi):', mae_new)
    print('MAPE (Terinversi):', mape_new)
    print('R-squared (Terinversi):', r_squared_new)

    # Visualisasi data aktual dan data terinversi
    plt.figure(figsize=(12, 8))
    plt.plot(hasil_df.index, hasil_df['Close'], label='Actual Close', color='blue')
    plt.plot(hasil_df.index, hasil_df['Pred_Close'], label='Predicted Close (BO-RF)', color='red', linestyle='--')
    plt.title("Actual vs Predicted Close Price")
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.legend()
    plt.show()

    display(hasil_df.head(10))
    return hasil_df
```

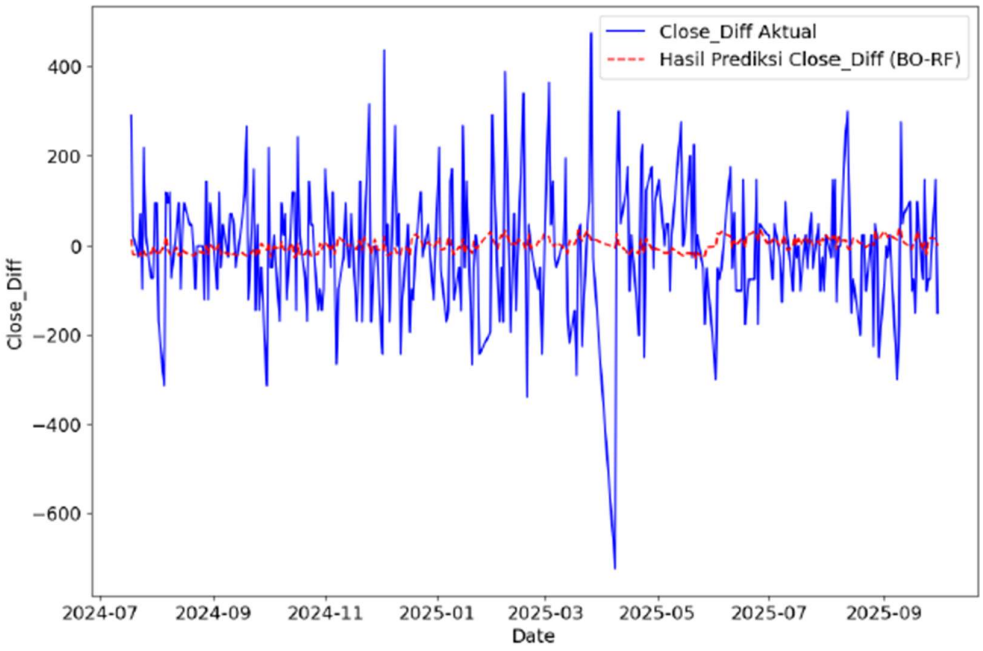
## MAIN FLOW

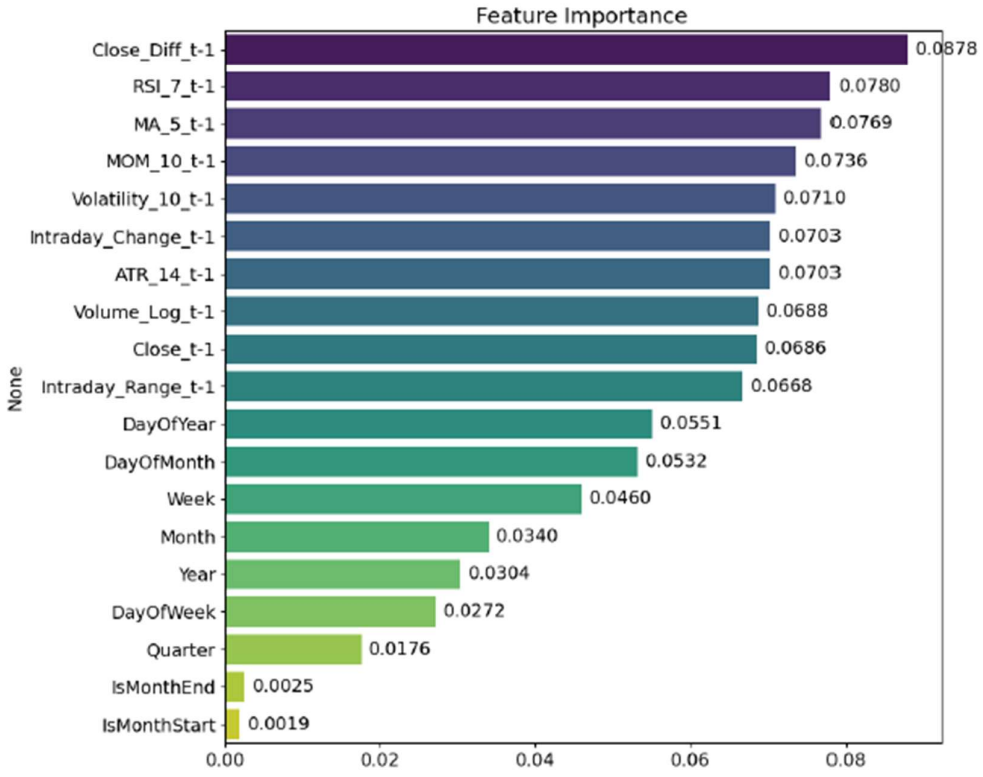
```
# Lokasi penyimpanan file data harga saham harian
path_data = r"D:\file\[KULIAHAN]\SKRIPSI\Resources\Data Olahan\BBCA_01102019_01102025.xlsx"

# Data Collection
df_awal = data_collection(path_data)

if df_awal is not None:
    # Data Preparation
    X_train, y_train, X_test, y_test, df_keseluruhan, variabel_input = data_preparation(df_awal)

    # Model Development
    best_model = model_development(X_train, y_train)

    # Model Evaluation
    y_pred_diff = model_evaluation(best_model, X_test, y_test)

    # Model Postprocessing
    df_results = model_postprocessing(best_model, X_test, y_pred_diff, df_keseluruhan, variabel_input)
```

```
--- Proses Data Collection ---
Data berhasil dikoleksi dari D:\file\[KULIAHAN]\SKRIPSI\Resources\Data Olahan\BBCA_01102019_01102025.xlsx
Ukuran data: (1447, 5)
Lima data teratas:
```

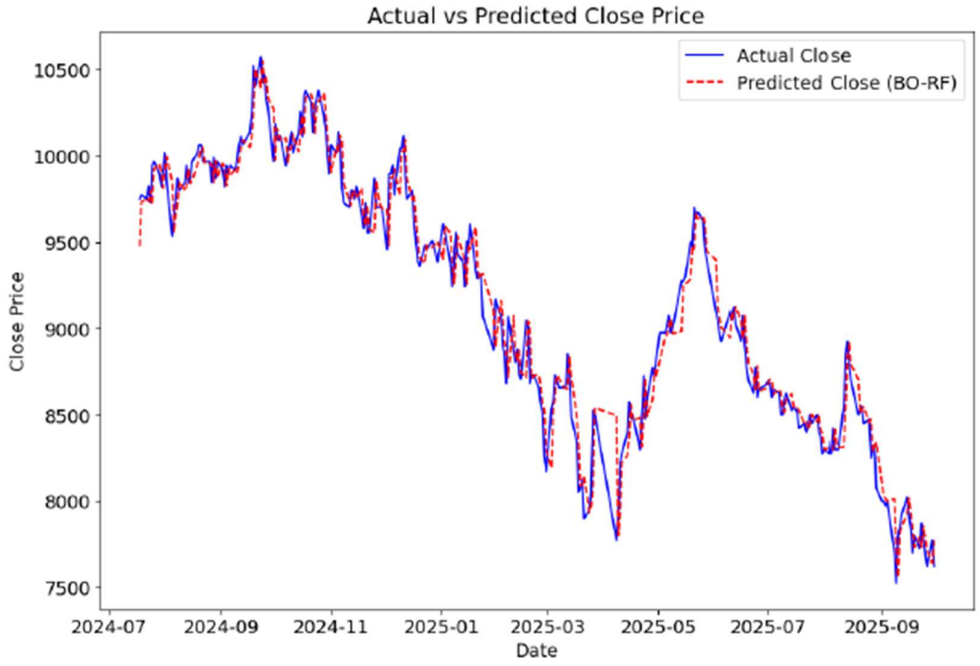
| Date | Open | High | Low | Close | Volume |
|---|---|---|---|---|---|
| 2019-10-01 | 5212.767129 | 5260.471321 | 5212.767129 | 5238.787598 | 40057000.0 |
| 2019-10-02 | 5212.767466 | 5264.808406 | 5212.767466 | 5247.461426 | 52376500.0 |
| 2019-10-03 | 5212.767129 | 5243.124342 | 5160.726193 | 5238.787598 | 50269000.0 |
| 2019-10-04 | 5186.746829 | 5260.471491 | 5186.746829 | 5243.124512 | 56890500.0 |
| 2019-10-07 | 5264.808105 | 5273.481595 | 5247.461127 | 5264.808105 | 56705500.0 |

```
Informasi umum data:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1447 entries, 2019-10-01 to 2025-09-30
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Open    1447 non-null   float64
 1   High    1447 non-null   float64
 2   Low     1447 non-null   float64
 3   Close   1447 non-null   float64
 4   Volume  1445 non-null   float64
dtypes: float64(5)
memory usage: 67.8 KB
--- Proses Data Preparation ---
Dimensi X_train: (1145, 19)
Dimensi X_test: (287, 19)
Dimensi y_train: (1145,)
Dimensi y_test: (287,)
--- Proses Model Development ---
|   iter    |  target   | n_esti... | max_depth | min_sa... | min_sa... | max_fe... |
-------------------------------------------------------------------------------------
| 1         | -100.6624 | 499.6320… | 47.78214… | 11.24791… | 12.77585… | 0.2404167 |
| 2         | -101.3908 | 324.7956… | 7.613762… | 13.12646… | 12.82007… | 0.7372653 |
| 3         | -100.9022 | 216.4675… | 48.64594… | 12.65419… | 5.822103… | 0.2636424 |
| 4         | -101.2157 | 346.7236… | 18.69090… | 8.346590… | 9.775010… | 0.3621062 |
| 5         | -102.0053 | 689.4823… | 11.27722… | 5.098025… | 8.594513… | 0.5104629 |
| 6         | -100.9085 | 452.8336… | 5.0       | 1.0       | 2.0       | 0.1       |
| 7         | -101.2998 | 999.8073… | 47.76123… | 12.24237… | 3.414141… | 0.7625173 |
| 8         | -100.5066 | 500.3972… | 47.26812… | 11.23830… | 13.41583… | 0.1749395 |
| 9         | -101.2358 | 503.6688… | 44.78179… | 9.472437… | 14.11575… | 0.4603522 |
| 10        | -100.2009 | 500.2149… | 47.68393… | 12.99775… | 15.33808… | 0.1       |
| 11        | -100.2311 | 502.5901… | 49.79454… | 14.25236… | 15.11252… | 0.1       |
| 12        | -100.2009 | 500.9994… | 50.0      | 12.49075… | 19.21919… | 0.1       |
| 13        | -100.2497 | 501.2467… | 45.87677… | 15.0      | 20.0      | 0.1       |
| 14        | -100.2432 | 495.5578… | 48.64438… | 15.0      | 20.0      | 0.1       |
| 15        | -100.2483 | 494.6306… | 41.56701… | 15.0      | 20.0      | 0.1       |
| 16        | -100.4400 | 494.4759… | 45.16298… | 8.910114… | 20.0      | 0.1       |
| 17        | -100.2475 | 487.3111… | 44.54739… | 15.0      | 20.0      | 0.1       |
| 18        | -100.2294 | 487.3661… | 36.56455… | 14.24271… | 20.0      | 0.1       |
| 19        | -100.2467 | 488.2818… | 39.83766… | 15.0      | 13.24605… | 0.1       |
| 20        | -101.9661 | 482.7370… | 40.29460… | 9.352467… | 16.72666… | 1.0       |
| 21        | -100.2452 | 492.6447… | 34.67791… | 15.0      | 15.98154… | 0.1       |
| 22        | -100.2467 | 488.9596… | 28.83963… | 15.0      | 20.0      | 0.1       |
| 23        | -100.2475 | 487.8085… | 29.68776… | 15.0      | 11.51674… | 0.1       |
| 24        | -100.2463 | 491.7434… | 35.44122… | 15.0      | 6.636961… | 0.1       |
| 25        | -100.2432 | 495.6641… | 27.26395… | 15.0      | 9.592651… | 0.1       |
| 26        | -100.2450 | 490.2201… | 27.16044… | 15.0      | 2.713301… | 0.1       |
| 27        | -100.3767 | 492.1268… | 28.49236… | 7.832870… | 6.892819… | 0.1       |
| 28        | -100.2293 | 489.5765… | 20.65754… | 14.97271… | 9.868920… | 0.1       |
| 29        | -101.8276 | 493.3170… | 22.83007… | 10.29829… | 16.13922… | 1.0       |
| 30        | -100.2491 | 483.7549… | 22.90984… | 15.0      | 6.006184… | 0.1       |
| 31        | -100.2450 | 490.7766… | 18.62018… | 15.0      | 2.767188… | 0.1       |
| 32        | -100.2505 | 498.4905… | 24.70340… | 15.0      | 2.294142… | 0.1       |
| 33        | -100.2508 | 499.3377… | 32.55044… | 15.0      | 4.029236… | 0.1       |
| 34        | -100.2494 | 484.4134… | 13.95217… | 15.0      | 7.360878… | 0.1       |
| 35        | -100.3750 | 485.2276… | 18.38858… | 7.788273… | 2.761525… | 0.1       |
| 36        | -100.2504 | 479.1419… | 19.18290… | 15.0      | 13.21987… | 0.1       |
| 37        | -101.5842 | 475.5628… | 16.56165… | 15.0      | 5.081159… | 1.0       |
| 38        | -100.2506 | 480.6859… | 25.76868… | 15.0      | 17.81060… | 0.1       |
| 39        | -100.2421 | 491.7934… | 10.34279… | 15.0      | 6.904030… | 0.1       |
| 40        | -100.2420 | 484.8162… | 11.69914… | 15.0      | 15.33408… | 0.1       |
| 41        | -100.2494 | 478.9312… | 16.49690… | 15.0      | 20.0      | 0.1       |
| 42        | -100.3196 | 486.7419… | 7.710761… | 8.425385… | 9.387003… | 0.1       |
| 43        | -100.3291 | 493.0871… | 11.45942… | 7.354206… | 2.0       | 0.1       |
| 44        | -100.2497 | 500.2320… | 13.98327… | 15.0      | 2.0       | 0.1       |
| 45        | -101.1741 | 498.3541… | 5.0       | 13.88048… | 2.0       | 1.0       |
| 46        | -100.3860 | 497.7604… | 19.31809… | 8.151689… | 2.0       | 0.1       |
| 47        | -100.2482 | 506.6445… | 19.90774… | 15.0      | 3.531875… | 0.1       |
| 48        | -100.2493 | 505.8637… | 27.67702… | 15.0      | 7.726219… | 0.1       |
| 49        | -100.3920 | 505.6190… | 27.35471… | 8.524272… | 2.0       | 0.1       |
| 50        | -100.4441 | 479.8113… | 12.05466… | 7.290578… | 17.37246… | 0.1       |
| 51        | -101.5403 | 514.1212… | 26.00568… | 15.0      | 3.059775… | 1.0       |
| 52        | -100.2497 | 501.0826… | 19.06808… | 15.0      | 7.701193… | 0.1       |
| 53        | -100.2531 | 472.2837… | 22.66451… | 15.0      | 20.0      | 0.1       |
| 54        | -100.4868 | 476.1961… | 21.85970… | 7.683068… | 20.0      | 0.1       |
| 55        | -100.2491 | 483.1761… | 32.68888… | 15.0      | 3.255449… | 0.1       |
=====================================================================================
Best Hyperparameters: {'n_estimators': 500, 'max_depth': 47, 'min_samples_leaf': 12, 'min_samples_split': 15, 'max_features': 0.1}
--- Proses Model Evaluation ---
RMSE: 145.3246447220812
MAE: 111.37879057104509
R-squared: 0.012606542350644623
```


Close_Diff Aktual
Hasil Prediksi Close_Diff (BO-RF)
Close_Diff
Date
400
200
0
−200
−400
−600
2024-07
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09


Feature Importance
None
Close_Diff_t-1 0.0878
RSI_7_t-1 0.0780
MA_5_t-1 0.0769
MOM_10_t-1 0.0736
Volatility_10_t-1 0.0710
Intraday_Change_t-1 0.0703
ATR_14_t-1 0.0703
Volume_Log_t-1 0.0688
Close_t-1 0.0686
Intraday_Range_t-1 0.0668
DayOfYear 0.0551
DayOfMonth 0.0532
Week 0.0460
Month 0.0340
Year 0.0304
DayOfWeek 0.0272
Quarter 0.0176
IsMonthEnd 0.0025
IsMonthStart 0.0019
0.00
0.02
0.04
0.06
0.08

```
--- Proses Model Post-Processing ---
RMSE (Terinversi): 145.32464472208116
MAE (Terinversi): 111.37879057104506
MAPE (Terinversi): 0.01230504613385758
R-squared (Terinversi): 0.9633985860557127
```




| Date | Close | Close_t-1 | Pred_Close_Diff | Pred_Close |
|---|---|---|---|---|
| 2024-07-18 | 9749.879883 | 9460.279297 | 15.181230 | 9475.460527 |
| 2024-07-19 | 9774.013672 | 9749.879883 | -18.090978 | 9731.788905 |
| 2024-07-22 | 9749.879883 | 9774.013672 | -20.884219 | 9753.129453 |
| 2024-07-23 | 9822.280273 | 9749.879883 | -15.104390 | 9734.775493 |
| 2024-07-24 | 9725.746094 | 9822.280273 | -22.214779 | 9800.065494 |
| 2024-07-25 | 9942.946289 | 9725.746094 | -10.442897 | 9715.303197 |
| 2024-07-26 | 9967.080078 | 9942.946289 | -20.506522 | 9922.439767 |
| 2024-07-29 | 9894.679688 | 9967.080078 | -15.591789 | 9951.488289 |
| 2024-07-30 | 9822.280273 | 9894.679688 | 0.124386 | 9894.804073 |
| 2024-07-31 | 9918.812500 | 9822.280273 | -9.732546 | 9812.547728 |