

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

IMPORT LIBRARY

```
In [9]: # Install Library yang dibutuhkan
        #pip install TA-Lib
        #pip install bayesian-optimization
```

```
In [10]: # 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

```
In [11]: 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

```
In [12]: 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

```

In [13]: 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 EVALUATION

MODEL POSTPROCESSING

```

        '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

```

In [ ]: # 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)

```