**Dataset Description**

To conduct this study, we utilized daily historical crude oil prices from two reliable sources:

1. **Yahoo Finance (CL=F)**: This dataset provides the daily futures prices for **West Texas Intermediate (WTI)** crude oil, one of the key benchmarks in global oil pricing. The data spans from **June 2, 2015, to June 2, 2025**, and includes open, high, low, close, adjusted close, and volume values.
2. **FRED - Federal Reserve Economic Data (DCOILWTICO)**: This dataset offers the **WTI spot price** in dollars per barrel as published by the **U.S. Energy Information Administration (EIA)**. It serves as a complementary and authoritative source for validating price trends observed in the Yahoo Finance dataset.

https://fred.stlouisfed.org/series/DCOILWTICO

**Statistical Analysis of Crude Oil Time Series**

In this section, we perform an in-depth statistical analysis of the daily **West Texas Intermediate (WTI)** crude oil price series, obtained from the **Federal Reserve Economic Data (FRED)** database under the code DCOILWTICO. The dataset spans from **June 2, 2015, to June 2, 2025**, and consists of daily closing prices in US dollars per barrel. Prior to analysis, missing values were forward-filled and the series was sorted by date.

**1. Stationarity Tests**

A stationary time series is essential for many forecasting models, especially ARIMA-based methods. To assess the stationarity of the crude oil price series, we apply three widely accepted tests:

* **Augmented Dickey-Fuller (ADF)** test,
* **Kwiatkowski-Phillips-Schmidt-Shin (KPSS)** test,
* **Phillips-Perron (PP)** test.

| **Test** | **Test Statistic** | **p-value** | **Interpretation** |
| --- | --- | --- | --- |
| ADF | -2.1737 | 0.2159 | Non-stationary |
| KPSS | 4.1180 | 0.0100 | Non-stationary |
| Phillips-Perron | -2.2633 | 0.1841 | Non-stationary |

The ADF and PP tests fail to reject the null hypothesis of a unit root, while the KPSS test rejects the null hypothesis of stationarity. Together, these results confirm that the WTI crude oil price series is **non-stationary in levels**. Consequently, differencing or transformation is needed before applying traditional time series models.

**2. Noise and Randomness Analysis**

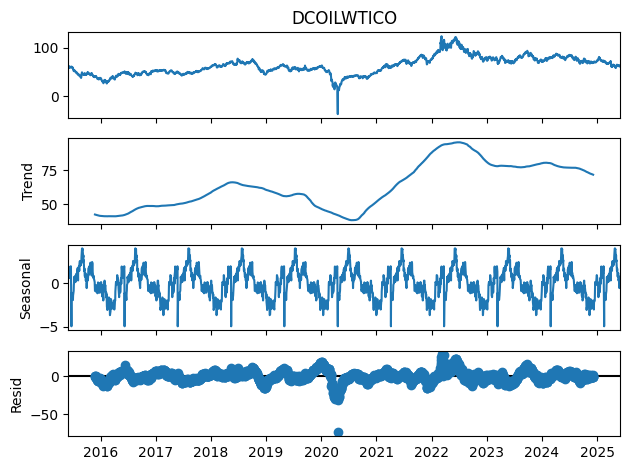
To further characterize the behavior of the series, we examine the randomness and autocorrelation of its first-differenced form (i.e., returns). The results are as follows:

* **Ljung-Box test** (lag 10):  
  Statistic = 113.26, p-value ≈ 1.17×10⁻¹⁹  
  → Indicates strong autocorrelation in the residuals.
* **Durbin-Watson test**:  
  DW = 2.3837  
  → Suggests slight **negative autocorrelation** in the returns.
* **Variance Ratio test**:  
  Statistic = -0.8667, p-value = 0.3861  
  → Fails to reject the null of a random walk, though the evidence is weak.

These findings suggest the presence of **dependence structures** in the returns and **deviation from a pure random walk**, making the series suitable for forecasting using autoregressive methods.

**3. Seasonal Decomposition**

We apply an additive seasonal decomposition to the original price series using a periodicity of 252 trading days (approximately one year). The decomposition reveals:

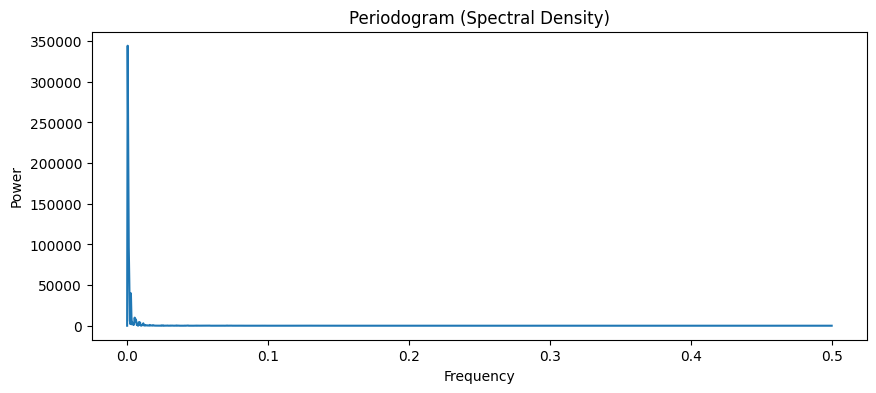
* A **strong trend component** that reflects long-term movement,
* A **minimal seasonal component**, confirming the absence of yearly cyclicality,
* A **residual (noise) component** with a mean close to zero but with noticeable outliers.
* 

Descriptive statistics of the residual component are as follows:

| **Metric** | **Value** |
| --- | --- |
| Count | 2358 |
| Mean | -0.0688 |
| Std. Dev. | 6.8728 |
| Min | -74.10 |
| Max | 28.47 |

The high standard deviation and extreme minimum values suggest the presence of **outliers and shocks**, which are typical in commodity markets influenced by geopolitical and macroeconomic events.

1. **Spectral Analysis**



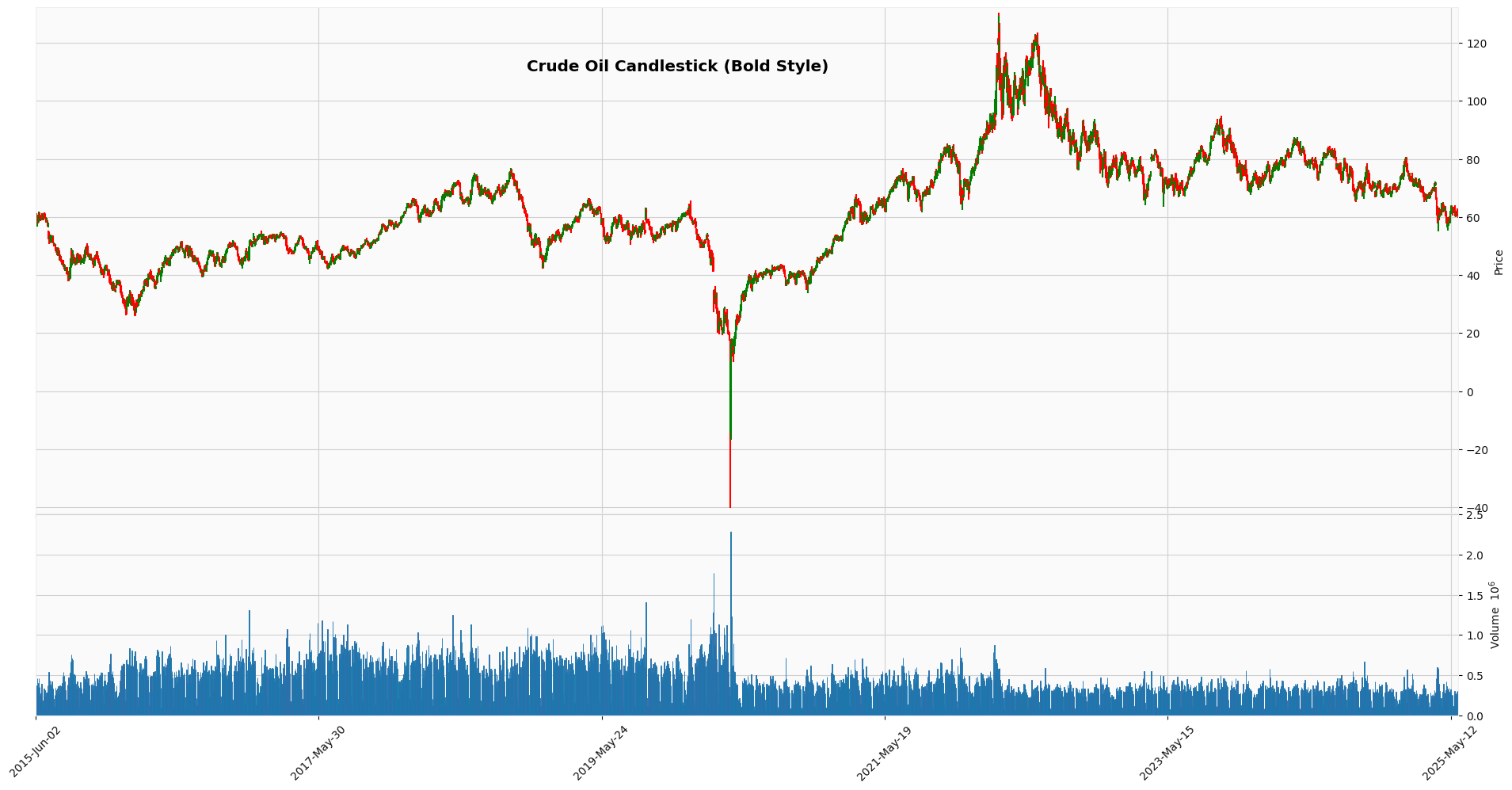
The periodogram of the crude oil price series, which quantifies the signal power across different frequencies, shows a dominant peak near zero. This indicates a **strong low-frequency component**, consistent with the presence of a **long-term trend**. Beyond this, the power quickly decays, suggesting **no dominant periodic behavior** or seasonal patterns.

**5. Heteroskedasticity (ARCH Effects)**

We conducted an **ARCH test** on the returns to evaluate volatility clustering, a common feature in financial time series. The results are as follows:

* ARCH Test Statistic = 811.68
* p-value = 0.0000

The very low p-value leads us to reject the null hypothesis of homoskedasticity. This confirms the presence of **time-varying volatility**, suggesting that models such as **ARCH/GARCH**, or deep learning models capable of handling nonlinearities (e.g., LSTM), may be more appropriate for modeling crude oil prices.



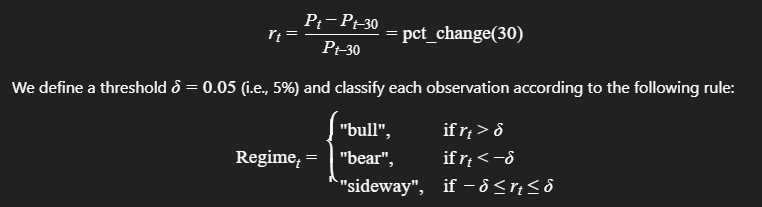
**Market Regime Classification**

To better understand the dynamics of crude oil price fluctuations and support regime-aware modeling, we implemented a **market regime classification** approach based on **30-day rolling returns**.

**5.1 Methodology**

Market regimes—such as bullish, bearish, or sideways trends—are essential for analyzing the structure and behavior of financial time series. In this study, we define these regimes using a **threshold-based approach** applied to the percentage return over a 30-day horizon.

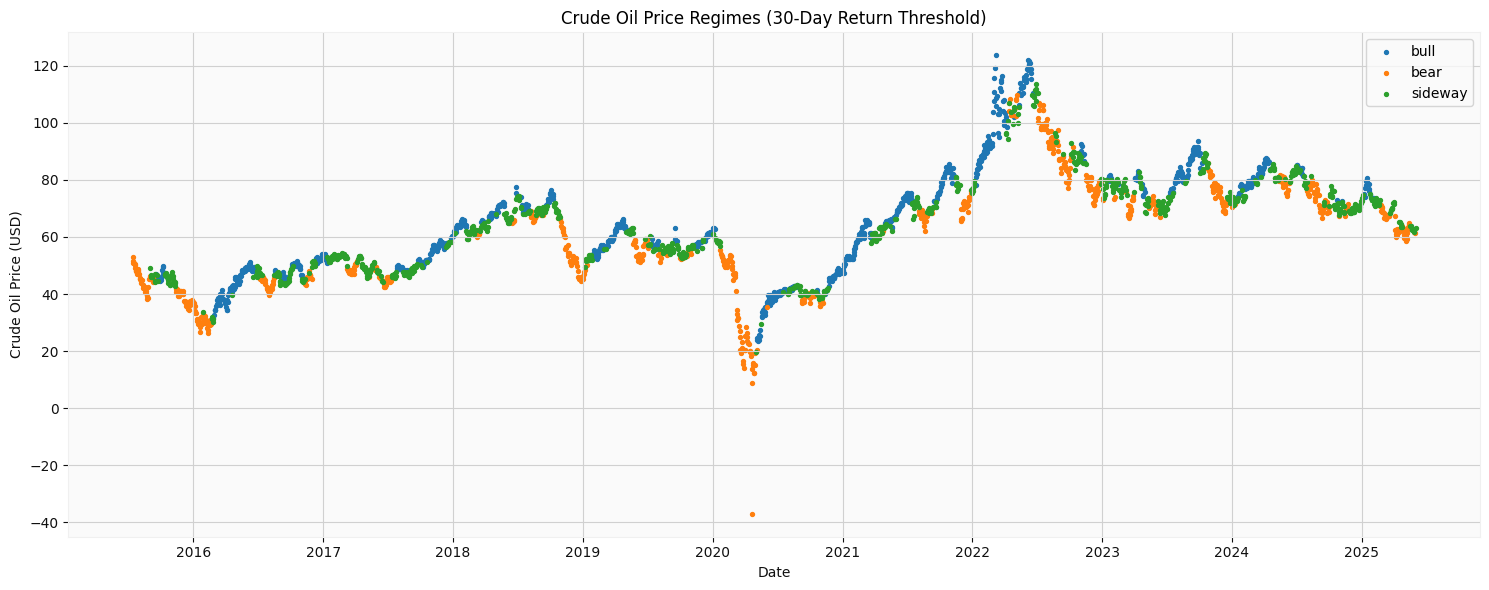
Let PtP\_tPt​ represent the crude oil price at time ttt. The **30-day return** is computed as:



This classification assigns each time point into one of the three distinct market conditions:

* **Bull market**: Strong upward trend over the past 30 days (return > 5%)
* **Bear market**: Significant downward trend (return < -5%)
* **Sideway market**: Low volatility or neutral price behavior (|return| ≤ 5%)

Notably, this regime analysis was used for **descriptive purposes only** and did **not serve as input** to any predictive models.



**4.4 Feature Engineering with Technical Indicators**

To extract meaningful patterns from crude oil price dynamics, we enriched the dataset using a set of widely-used **technical indicators** from the ta (technical analysis) library along with **trading volume data** sourced from Yahoo Finance. These features are commonly employed in financial time series analysis to capture momentum, trend, and volatility.

**4.4.1 Added Technical Indicators**

The following indicators were calculated using the daily crude oil price (DCOILWTICO):

* **Relative Strength Index (RSI):**  
  A momentum oscillator that measures the speed and change of price movements over a 14-day window. It ranges between 0 and 100, where values above 70 indicate overbought conditions and below 30 indicate oversold.
* **Moving Average Convergence Divergence (MACD):**  
  A trend-following momentum indicator derived from the difference between short-term (12-day) and long-term (26-day) exponential moving averages. We also initially included the MACD signal line (9-day EMA of MACD), which is used for crossover signals.
* **Simple Moving Averages (MA10, MA50):**  
  Calculated over 10 and 50 days respectively, these help smooth out short- and mid-term trends in price.
* **Bollinger Bands (Upper and Lower):**  
  These represent dynamic upper and lower bounds calculated as ±2 standard deviations from a 20-day moving average. They capture volatility and identify potential breakout points.

**Multicollinearity Reduction**

To prevent issues arising from **multicollinearity**—which can distort model interpretations and inflate variance—we computed the **Pearson correlation matrix** across all engineered features. The matrix revealed several pairs with **correlation coefficients exceeding 0.95**, especially among moving averages and Bollinger Bands.

For example:

* MA10 and MA50: corr = 0.97
* BB\_High and MA10: corr = 0.98
* BB\_Low and MA10: corr = 0.98
* MACD and MACD\_Signal: corr = 0.96

**Final Feature Set for Analysis**

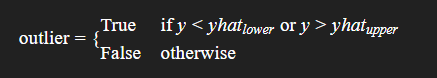
The final dataset used for modeling and analysis contains the following features:

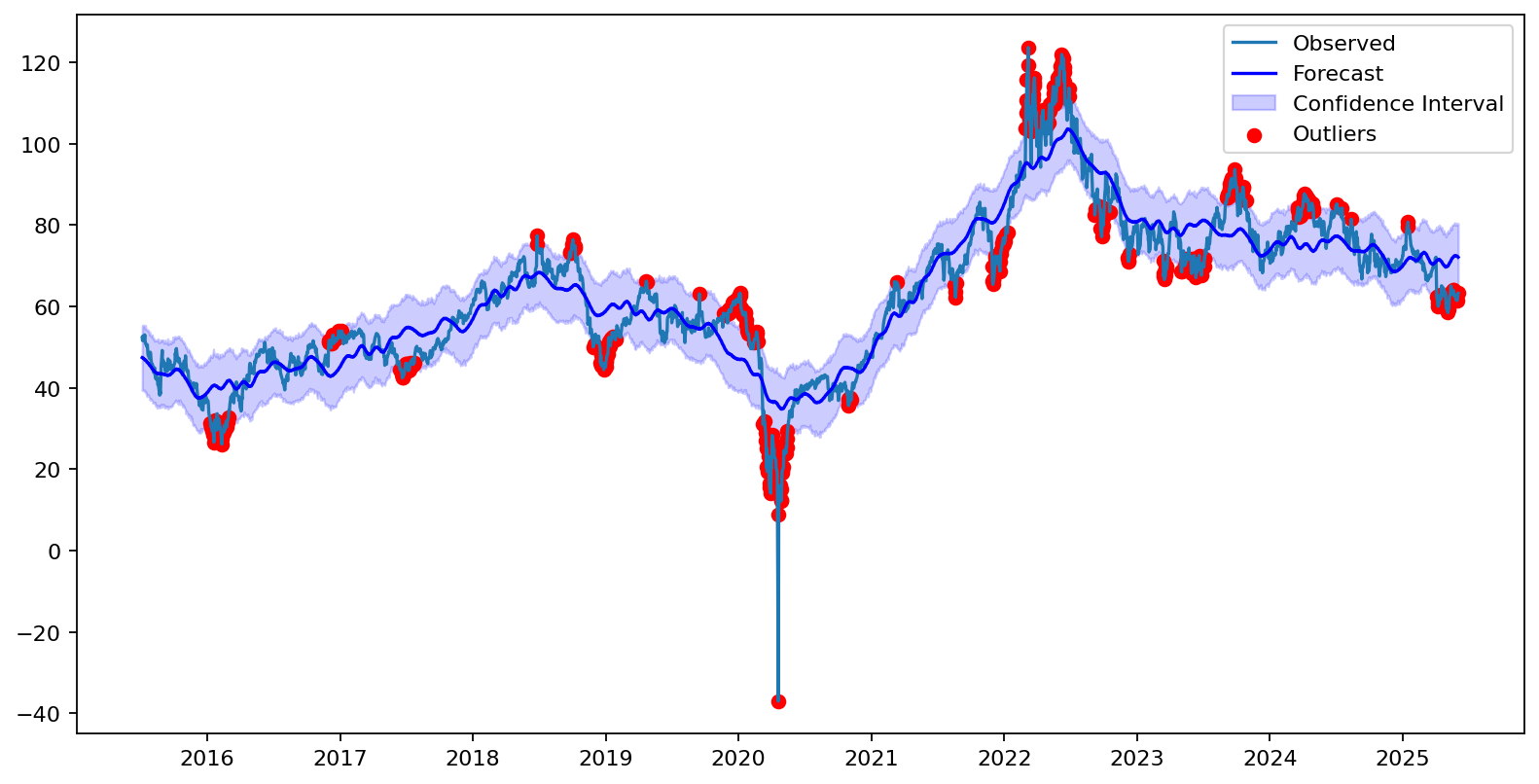
* **DCOILWTICO**: Crude oil price
* **RSI**: Relative Strength Index
* **MACD**
* **MA10**: 10-day Moving Average
* **Volume**: Daily trading volume

**Outlier Detection Using Prophet**

To identify outliers in the time series data, we utilized the **Prophet** forecasting model developed by Facebook, which is designed for handling time series with strong seasonal effects and trends.

1. **Model Fitting**:  
   The Prophet model was first fitted to the historical data (df\_prophet). This involves learning the underlying trend and seasonality components present in the data.
2. **Prediction and Uncertainty Intervals**:  
   After fitting, we generated forecasts for the observed time points (using the same dates ds in the dataset). Prophet not only provides point predictions (yhat) but also estimates uncertainty intervals (yhat\_lower and yhat\_upper). These intervals represent the model’s confidence bounds, typically corresponding to a 95% confidence interval by default.
3. **Outlier Identification**:  
   An observation was labeled as an outlier if its actual value y fell **outside the predicted confidence interval**—specifically, if it was **less than the lower bound** (yhat\_lower) or **greater than the upper bound** (yhat\_upper). Formally:





Related Articles:

<https://jfin-swufe.springeropen.com/articles/10.1186/s40854-024-00637-z>

<https://www.mdpi.com/1911-8074/17/9/415>

<https://link.springer.com/article/10.1007/s10479-023-05810-8>

https://www.mdpi.com/2076-3417/15/3/1055

**Hybrid Deep Learning Model for Crude Oil Price Forecasting with Optuna-Based Hyperparameter Tuning**

Abstract

Forecasting crude oil prices is critical due to its wide-ranging economic implications. In this study, we introduce a hybrid deep learning model that integrates Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM) layers, and Transformer encoder blocks to capture both local and long-range dependencies in time series data. We further employ Optuna, a powerful hyperparameter optimization framework, to fine-tune the model’s performance. Evaluation on real-world data demonstrates that the proposed model achieves high accuracy across multiple statistical metrics including RMSE, MAE, R², sMAPE, and MASE, outperforming baseline methods in short-term prediction.

1. Introduction

Crude oil prices are influenced by various market and geopolitical factors, making accurate forecasting a significant challenge. Traditional statistical models often fall short in capturing complex nonlinear dependencies. Recently, deep learning models such as LSTM and Transformers have proven successful for time series forecasting. However, model performance is highly sensitive to hyperparameters and architecture design.

This research proposes a hybrid model combining CNN, BiLSTM, and Transformer layers and leverages Optuna for automatic hyperparameter tuning. The model is evaluated on historical crude oil prices and technical indicators, with strong results across all performance metrics.

2. Data Description and Preprocessing

The dataset includes daily entries of the following indicators:

* DCOILWTICO – Crude oil price (USD),
* Volume – Trading volume,
* MA10 – 10-day moving average,
* RSI – Relative Strength Index,
* MACD – Moving Average Convergence Divergence.

Preprocessing:

* Missing values are handled using forward-fill imputation.
* Data is normalized using MinMaxScaler.
* The dataset is split into train (1950 samples), validation (350 samples), and test (remaining samples).
* A sliding window technique is used to convert data into sequences with a fixed length (seq\_length), which are then used as input for the neural network.

3. Model Architecture

The model architecture comprises three major components:

3.1 Convolutional Layers (CNN)

Two 1D convolution layers are applied to extract local features and reduce noise:

* Conv1D (128 filters, kernel size 3) → ReLU
* MaxPooling1D

3.2 Bidirectional LSTM Layers

These layers capture long-range dependencies from both past and future directions:

* BiLSTM (65 units) → return\_sequences=True
* BiLSTM (32 units)

3.3 Transformer Encoder Layer

Incorporates multi-head self-attention and feed-forward networks to model dynamic dependencies:

* 6 attention heads
* Feed-forward dimension = 196
* Dropout = 0.129

The final layers include:

* GlobalAveragePooling1D
* Dense (64 units, ReLU)
* Dropout
* Output layer: Dense(1)

4. Hyperparameter Optimization with Optuna

We used Optuna to tune the following hyperparameters:

* Sequence length (seq\_length): 8
* LSTM units: 65
* Transformer heads: 6
* Feed-forward dimension: 196
* Dropout: 0.129
* Learning rate: 0.000565

Best Trial Metrics (on Validation Data):

* RMSE: 0.00013
* MSE: 0.00013
* MAE: 0.0089
* R² Score: 0.9003
* sMAPE: 1.25%
* MASE: 1.29

These results guided the configuration of the final model used in the evaluation phase.

5. Model Training and Evaluation

The final model was trained using:

* Optimizer: Adam (learning rate = 0.000565)
* Loss function: Mean Squared Error (MSE)
* Callbacks: EarlyStopping (patience = 50), ReduceLROnPlateau

5.1 Training Results

* RMSE: 0.0194
* MSE: 0.00038
* MAE: 0.0130
* sMAPE: 21.50%
* MASE: 18.83
* R² Score: 0.9734

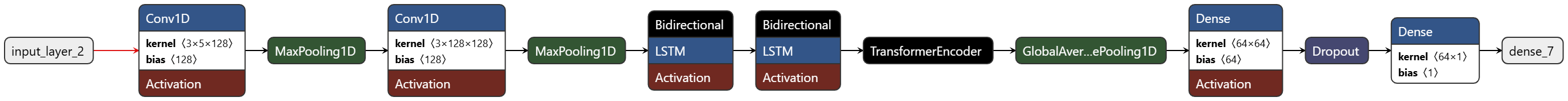
5.2 Validation Results

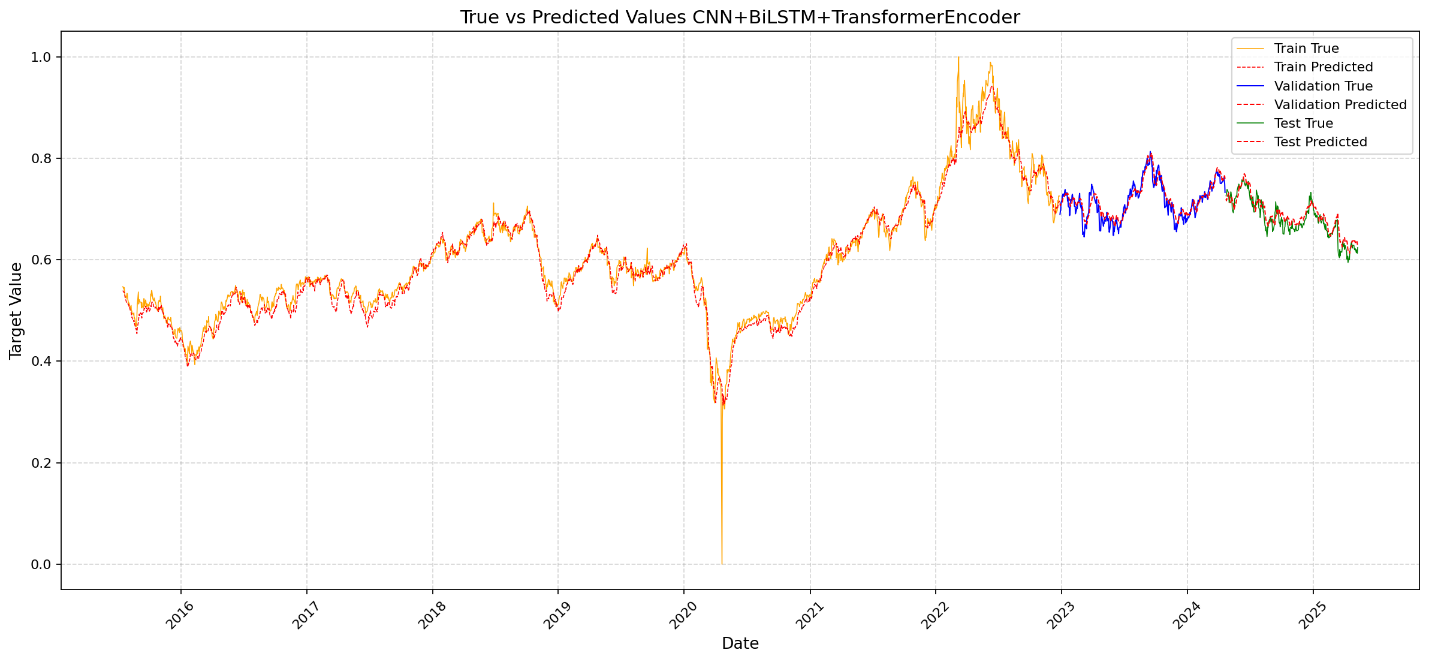
* RMSE: 0.0143
* MSE: 0.00020
* MAE: 0.0110
* sMAPE: 5.41%
* MASE: 5.33
* R² Score: 0.8439

5.3 Test Results

* RMSE: 0.0151
* MSE: 0.00023
* MAE: 0.0120
* sMAPE: 6.12%
* MASE: 6.28
* R² Score: 0.8483

These results confirm that the hybrid architecture generalizes well across unseen data.





Convolutional Neural Network (CNN) Model

In this section, we present the development and performance of a Convolutional Neural Network (CNN) model applied to time series forecasting. CNNs, originally designed for image processing, have shown significant effectiveness in learning local temporal patterns in sequential data. Their ability to capture features through convolutional filters makes them well-suited for forecasting tasks, particularly when the input sequence contains short-term dependencies and recurring patterns.

3.4.1 Model Design and Input Structure

To utilize CNNs for time series prediction, the raw data was first transformed into a supervised learning format using a sliding window approach. Specifically, five consecutive time steps were used as inputs to predict the subsequent time step. This fixed-size input sequence enabled the model to learn temporal features efficiently. Each input sequence thus acted like a "window" capturing local trends and variations in the data.

The CNN model architecture comprised two one-dimensional convolutional layers, each followed by a max pooling layer. These layers served to extract and downsample the most significant patterns from the input sequence. Following the convolutional layers, the output was flattened and passed through a dense (fully connected) layer to interpret the learned features and produce a final prediction. A dropout layer was included before the final output to prevent overfitting by randomly omitting a fraction of neurons during training.

3.4.2 Hyperparameter Optimization

To enhance the model's performance, a comprehensive hyperparameter tuning process was conducted using Optuna, an efficient and scalable hyperparameter optimization framework. The optimization process explored various configurations, including the number of filters, kernel sizes, the size of the dense layer, the dropout rate, and the learning rate. The objective was to minimize the mean squared error on the validation set. The optimization algorithm iteratively evaluated multiple combinations and identified the best-performing set of parameters for the model.

Best trial:

{'filters1': 128, 'kernel\_size1': 3, 'filters2': 64, 'kernel\_size2': 2, 'dense\_units': 77, 'dropout\_rate': 0.23429945185590811, 'learning\_rate': 0.00887466936279092, 'batch\_size': 64}

======== Train Metrics ========

Train RMSE: 0.017613178511907364

Train MSE: 0.00031022405729231526

Train MAE: 0.010767255688230847

Train sMAPE: 20.75%

Train MASE: 18.273

Train R² Score: 0.9780

======== Validation Metrics ========

Validation RMSE: 0.013294962682448837

Validation MSE: 0.00017675603272770718

Validation MAE: 0.010543324135521809

Validation sMAPE: 5.52%

Validation MASE: 5.393

Validation R² Score: 0.8636

======== Test Metrics ========

Test RMSE: 0.012150133148054113

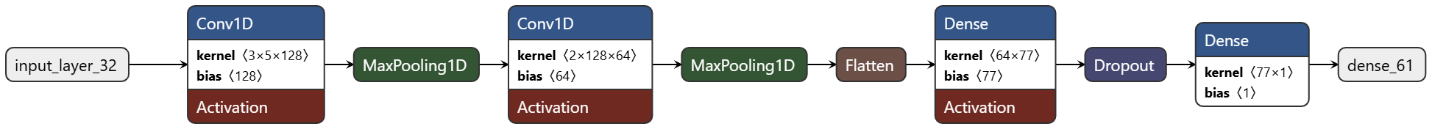
Test MSE: 0.00014762573551544336

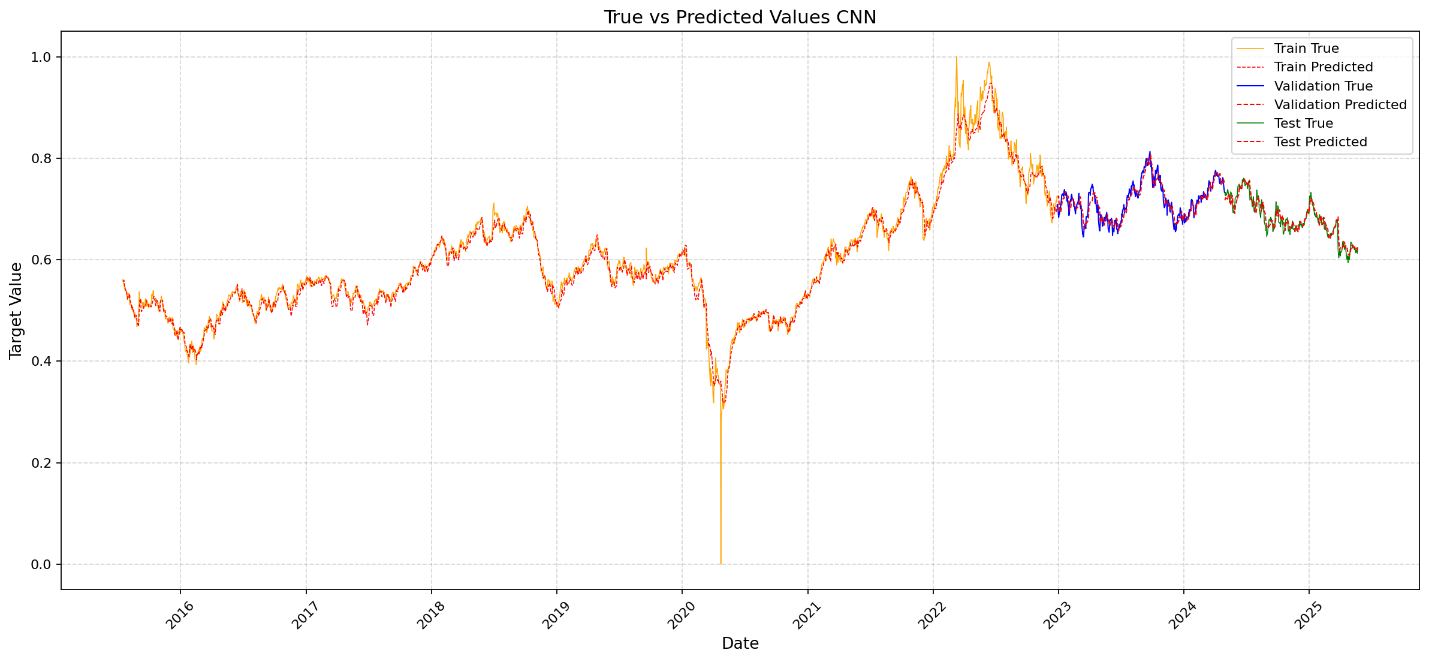
Test MAE: 0.009180678390349237

Test sMAPE: 6.27%

Test MASE: 6.418

Test R² Score: 0.9025





**Bidirectional LSTM**

Methodology

Sequence Preparation

Time series data was converted into supervised sequences suitable for LSTM input using a sliding window approach with a sequence length of 5 days. The target variable was the normalized crude oil price for the next day.

Model Architecture

A Bidirectional LSTM architecture was employed to capture temporal dependencies from past and future directions, enhancing context understanding. The network includes:

* Two stacked Bidirectional LSTM layers with dropout for regularization.
* Dense layers for nonlinear transformation leading to a single output predicting the next day's price.

Hyperparameter Optimization

Optuna, a state-of-the-art hyperparameter optimization framework, was utilized to tune:

* Number of LSTM units in each layer.
* Dropout rate.
* Learning rate.
* Optimizer type (Adam or RMSprop).
* Batch size.

This automated search efficiently explored the hyperparameter space over 30 trials, minimizing the validation mean squared error.

Model Training

The best hyperparameters identified by Optuna were used to train the final model. Early stopping and learning rate reduction callbacks prevented overfitting and optimized convergence.

Evaluation Metrics

Model performance was evaluated using several metrics:

* Root Mean Squared Error (RMSE): Measures standard deviation of residuals.
* Mean Absolute Error (MAE): Average absolute difference between predicted and actual values.
* Symmetric Mean Absolute Percentage Error (sMAPE): Normalized measure of prediction accuracy, suitable for data with large value ranges.
* Mean Absolute Scaled Error (MASE): Scales error relative to a naive baseline.
* Coefficient of Determination (R²): Proportion of variance explained by the model.

These metrics were computed on training, validation, and test sets to assess model generalization.

======== Train Metrics ========

Train RMSE: 0.018889880533778828

Train MSE: 0.00035682758658043623

Train MAE: 0.012594045742580787

Train sMAPE: 20.51%

Train MASE: 18.119

Train R² Score: 0.9746

======== Validation Metrics ========

Validation RMSE: 0.013734568130086462

Validation MSE: 0.00018863836171998673

Validation MAE: 0.01085284502551401

Validation sMAPE: 5.61%

Validation MASE: 5.477

Validation R² Score: 0.8544

======== Test Metrics ========

Test RMSE: 0.012347151712374696

Test MSE: 0.00015245215540839738

Test MAE: 0.009601637282520177

Test sMAPE: 6.31%

Test MASE: 6.447

Test R² Score: 0.8993

Best trial:

Loss: 0.0001122517860494554

Params:

n\_units1: 142

n\_units2: 71

dropout: 0.37887591262535264

learning\_rate: 0.0017616121186570944

optimizer: Adam

batch\_size: 64

