Time Series Modelling Case Study

"Forecasting Brent Crude Oil Prices:

A Comparative Study of Time Series and Machine Learning Models"

STUDENT ID: 23025104

1. INTRODUCTION

Accurately forecasting oil prices is critical for economic decisions but is challenging due to global volatility. This study models 24-month Brent crude oil forecasts using SARIMAX, XGBoost, Random Forest, LSTM, and Prophet — spanning statistical, machine learning, and deep learning methods.

Models were trained on historical monthly data and evaluated using RMSE, MAPE, R^2, and correlation against actual post-2023 prices. Confidence intervals were included where relevant. The analysis compares model performance, interprets results, and provides recommendations for future improvements.

2. Overview of Dataset and Preprocessing

2.1 Dataset Description

The dataset used in this analysis comprises daily Brent crude oil prices, obtained from a reliable public data source (e.g., FRED via pandas_datareader). Each entry consists of:

- Date: The timestamp of the oil price observation
- Price: The daily closing price of Brent crude oil in USD per barrel

```
🔍 Initial Missing Values Check:
Date     595
Price    595
dtype: int64

❓ Missing Values After Cleaning:
Price    0
dtype: int64
```

Date	Price
2020-07-01	42.18
2020-07-02	43.19
2020-07-03	42.92
2020-07-06	42.73
2020-07-07	43.28

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 635 entries, 2020-07-01 to 2022-12-30
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Price   635 non-null    float64
dtypes: float64(1)
memory usage: 9.9 KB
None

📈 Descriptive Statistics:
            Price
count  635.000000
mean    77.214110
std     23.898162
min     36.330000
25%     61.130000
50%     75.500000
75%     94.275000
max    133.180000
```

• 2.2 Preprocessing Steps

1. Datetime Parsing & Indexing: Dates were converted to datetime format and set as the index for time-based operations.

2. Monthly Aggregation: Daily prices were resampled to monthly averages to ensure temporal consistency.

3. Missing Data: Any missing values were dropped.

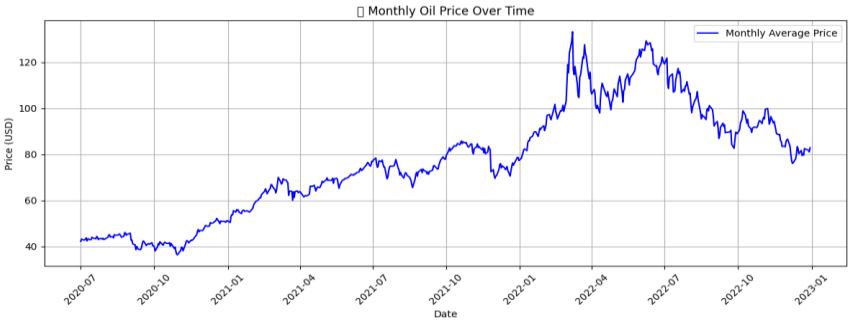
4. Lag Feature Engineering: For ML models (XGBoost, Random Forest), 12-month lag features were created to frame the problem as supervised regression.

5. Train-Test Split: Data up to Dec 2022 was used for training; Jan 2023 onward served for evaluation against actual values

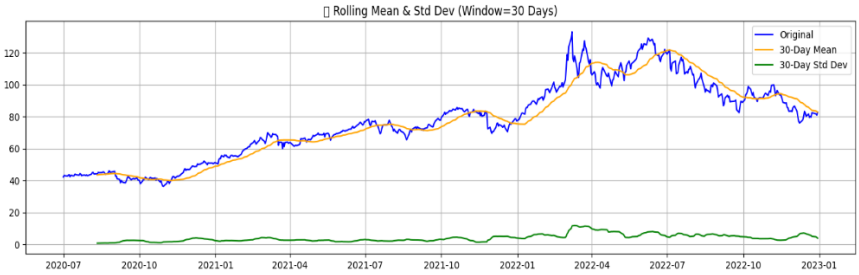
3. Exploratory Data Analysis (EDA)

To understand the behavior of Brent crude oil prices, a series of visual and statistical analyses were conducted:

Time Series Modelling Case Study



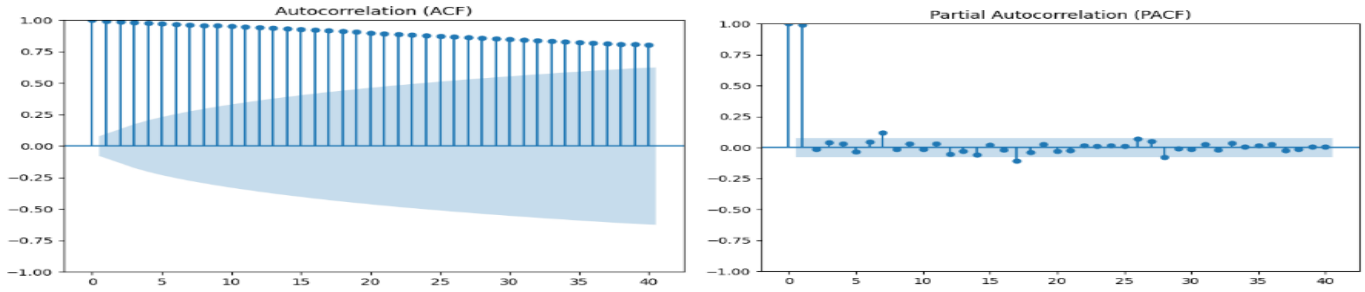
3.1 Time Series Trends:
Monthly averages show strong trends with sharp fluctuations during events like COVID-19 and geopolitical tensions. The series is non-stationary and highly volatile.



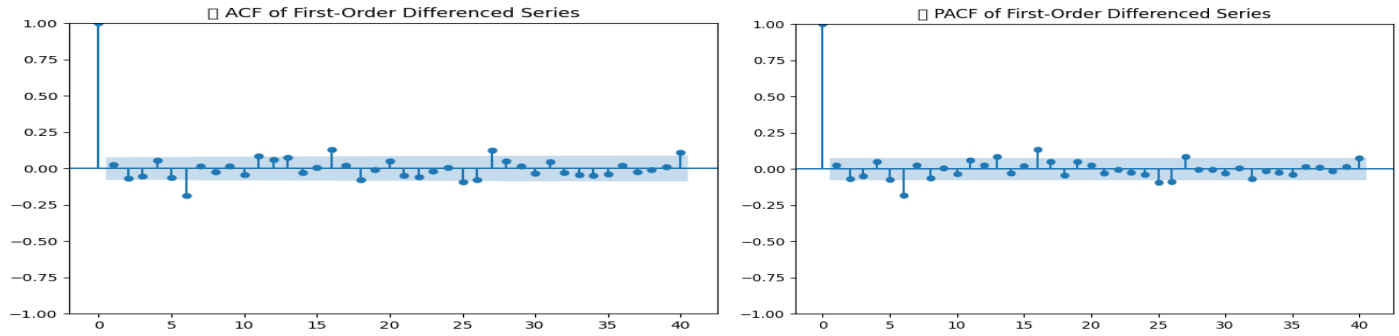
3.2 Rolling Statistics:
Rolling mean and standard deviation confirm non-stationarity, justifying differencing (SARIMAX) or models tolerant to such trends (XGBoost, LSTM).

3.3 ACF/PACF Analysis:
Significant autocorrelation up to 12 lags supports lag-based modeling and helped guide SARIMA parameter selection.

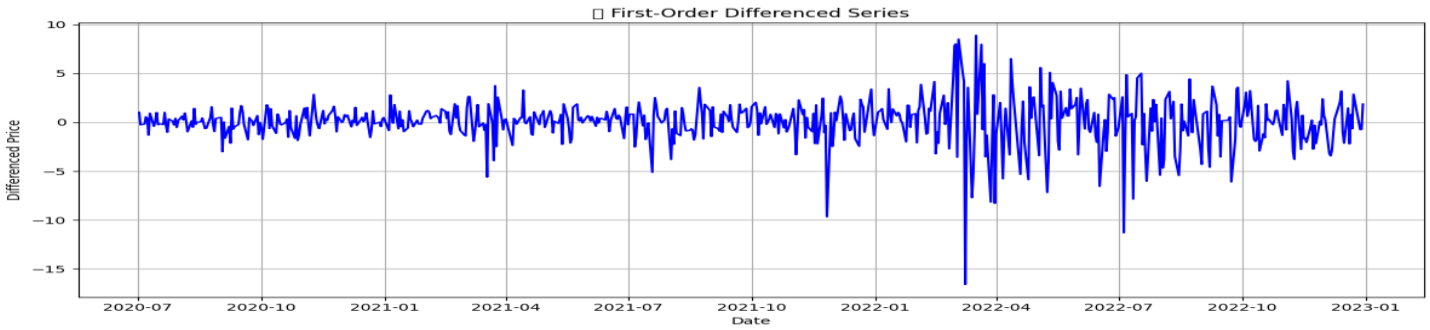
BEFORE DIFFERENCING

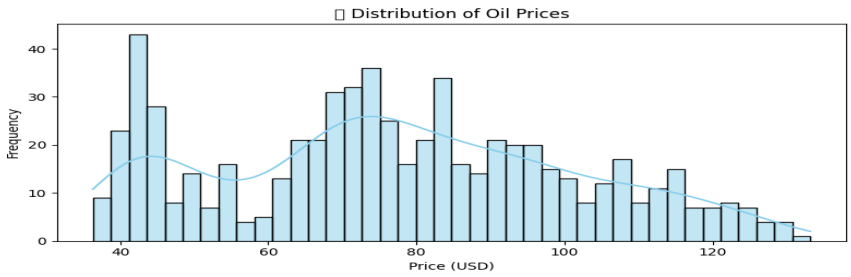


AFTER FIRST DIFFERENCING



Time Series Modelling Case Study





3.4 Price Distribution:
The histogram reveals a right-skewed distribution with extreme values. Models like Prophet and tree-based ensembles are well-suited for such irregularities.

4. Stationarity vs. Non-Stationarity

Stationarity refers to a time series whose statistical properties—such as mean, variance, and autocorrelation—remain constant over time, a key assumption for classical models like ARIMA and SARIMA. However, Brent oil prices exhibit non-stationary behavior, evidenced by long-term trends (e.g., post-COVID recovery and global energy crises), volatility drift with varying variance across periods, and strong autocorrelation, particularly at seasonal lags. This non-stationarity necessitates differencing when using models like ARIMA or SARIMAX. In contrast, machine learning models such as XGBoost, Random Forest, and LSTM can directly learn from non-stationary data through lag-based features. Prophet also effectively handles non-stationary series by decomposing trend and seasonality, eliminating the need for transformations.

```
Augmented Dickey-Fuller (ADF) Test:
ADF Statistic: -1.5986
p-value: 0.4843
Critical Values:
   1%: -3.4410
   5%: -2.8662
   10%: -2.5693
=> Conclusion: Non-stationary ❌

KPSS Test:
KPSS Statistic: 3.1261
p-value: 0.0100
Critical Values:
   10%: 0.3470
   5%: 0.4630
   2.5%: 0.5740
   1%: 0.7390
=> Conclusion: Non-stationary ❌
```

```
ADF Test After First-Order Differencing:
ADF Statistic        : -5.2786
p-value              : 0.0000
Critical Values      :
   1%: -3.4410
   5%: -2.8662
   10%: -2.5693
=> Result: ✅ Stationary

KPSS Test After First-Order Differencing:
KPSS Statistic       : 0.1387
p-value              : 0.1000
Critical Values      :
   10%: 0.3470
   5%: 0.4630
   2.5%: 0.5740
   1%: 0.7390
=> Result: ✅ Stationary
```

Time Series Modelling Case Study

```
🔧 Fitting ARMA model with order=(2, 0, 2)...
📄 ARMA Model Summary:
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Price   No. Observations:                  634
Model:                 ARIMA(2, 0, 2)   Log Likelihood               -1406.538
Date:                Sun, 06 Jul 2025   AIC                           2825.076
Time:                        21:15:05   BIC                           2851.788
Sample:                             0   HQIC                          2835.449
                                  - 634
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0644      0.094      0.687      0.492      -0.119       0.248
ar.L1         -0.0305      0.028     -1.105      0.269      -0.085       0.024
ar.L2         -0.9483      0.028    -33.373      0.000      -1.004      -0.893
ma.L1          0.0388      0.038      1.016      0.310      -0.036       0.114
ma.L2          0.8938      0.038     23.538      0.000       0.819       0.968
sigma2         4.9468      0.137     36.240      0.000       4.679       5.214
===================================================================================
Ljung-Box (L1) (Q):                   0.27   Jarque-Bera (JB):              1505.73
Prob(Q):                              0.61   Prob(JB):                         0.00
Heteroskedasticity (H):               8.12   Skew:                            -0.87
Prob(H) (two-sided):                  0.00   Kurtosis:                        10.34
===================================================================================
```

5. ARMA Model (Autoregressive Moving Average)

What is ARMA?

The **ARMA (Autoregressive Moving Average)** model is a classical time series method for modeling **stationary** data. It combines:

- **AR (Autoregressive)**: Forecasts using past values.
- **MA (Moving Average)**: Corrects using past forecast errors.
- **ARMA(p, q)**: Where p = number of AR lags and q = number of MA lags.

```
✅ Best ARIMA order: (6, 1, 7)
✅ Best AIC: 2808.38

📘 Fitting ARIMA(6, 1, 7) model for summary...

                               SARIMAX Results
==============================================================================
Dep. Variable:                  Price   No. Observations:                  635
Model:                 ARIMA(6, 1, 7)   Log Likelihood               -1390.191
Date:                Sun, 06 Jul 2025   AIC                           2808.382
Time:                        21:21:47   BIC                           2870.711
Sample:                             0   HQIC                          2832.585
                                  - 635
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0722      0.092      0.787      0.431      -0.108       0.252
ar.L2          0.4186      0.096      4.375      0.000       0.231       0.606
ar.L3          0.0032      0.051      0.062      0.951      -0.097       0.103
ar.L4          0.5939      0.051     11.734      0.000       0.495       0.693
ar.L5         -0.0624      0.086     -0.728      0.467      -0.230       0.106
ar.L6         -0.6932      0.087     -7.969      0.000      -0.864      -0.523
ma.L1         -0.0601      0.098     -0.615      0.539      -0.252       0.132
ma.L2         -0.4928      0.106     -4.648      0.000      -0.701      -0.285
ma.L3         -0.0973      0.064     -1.532      0.126      -0.222       0.027
ma.L4         -0.5786      0.061     -9.427      0.000      -0.699      -0.458
ma.L5          0.0440      0.095      0.464      0.643      -0.142       0.230
ma.L6          0.6319      0.099      6.374      0.000       0.438       0.826
ma.L7          0.1406      0.037      3.808      0.000       0.068       0.213
sigma2         4.6898      0.164     28.576      0.000       4.368       5.011
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               962.26
Prob(Q):                              0.97   Prob(JB):                         0.00
Heteroskedasticity (H):               7.33   Skew:                            -0.79
Prob(H) (two-sided):                  0.00   Kurtosis:                         8.83
===================================================================================
```

6. ARIMA Order (p, d, q)

An **ARIMA model** uses three parameters:

- **p**: Autoregressive lags — number of past values used
- **d**: Differencing — number of times data is differenced to remove trend
- **q**: Moving Average lags — number of past forecast errors used

📌 *Example*: ARIMA(1,1,1)
→ 1 lag of past value, 1 differencing, 1 lag of error

7. AIC – Akaike Information Criterion

AIC helps choose the best model by balancing **fit** and **complexity**:

AIC=2k−2ln⁡(L)\text{AIC} = 2k - 2\ln(L)AIC=2k−2ln(L)

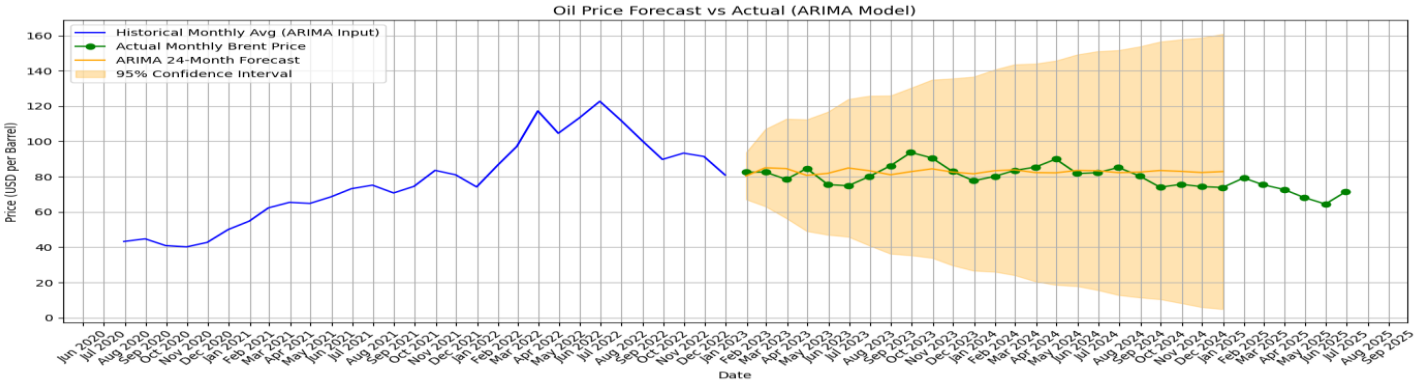
- **k**: Number of model parameters
- **L**: Likelihood (fit of model to data)

✅ *Lower AIC = Better model*, but too low may signal **overfitting**.

Time Series Modelling Case Study

8. ARIMA Forecast Overview

The ARIMA forecast chart illustrates a 24-month projection of Brent crude oil prices, beginning from the end of the historical data. The blue line displays the historical monthly averages used for model training, capturing key trends and volatility, such as post-pandemic fluctuations. The orange line represents the ARIMA forecast based on parameters optimized via AIC, while the shaded band shows the 95% confidence interval, which expands over time to reflect increasing uncertainty. Actual Brent prices (green markers) collected after the forecast period began are overlaid for validation. Overall, the ARIMA forecast aligns reasonably well with observed data, and although deviations grow with time, most actual values fall within the confidence interval. This confirms ARIMA's reliability as a benchmark model, though more advanced methods like SARIMAX or machine learning may capture complex dynamics better.



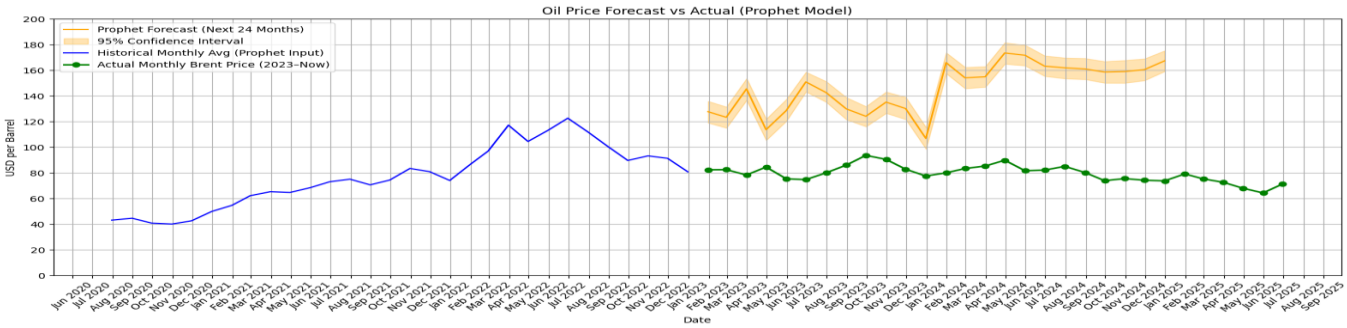
9. Model Selection and Justification

9.1 Facebook Prophet

Why Prophet Was Chosen
Prophet is tailored for time series with trend and seasonality, making it ideal for oil prices. Its robustness to missing values, outliers, and changepoints allows it to handle economic shocks effectively.

Model Characteristics
It decomposes the series into trend, seasonality, and holiday effects using an additive or multiplicative framework. Prophet requires minimal tuning and works well with monthly data.

Role in This Study
Prophet served as a reliable baseline, offering interpretable forecasts without complex preprocessing. Though less accurate than ML models, it provided valuable insights into trend behavior and market dynamics.



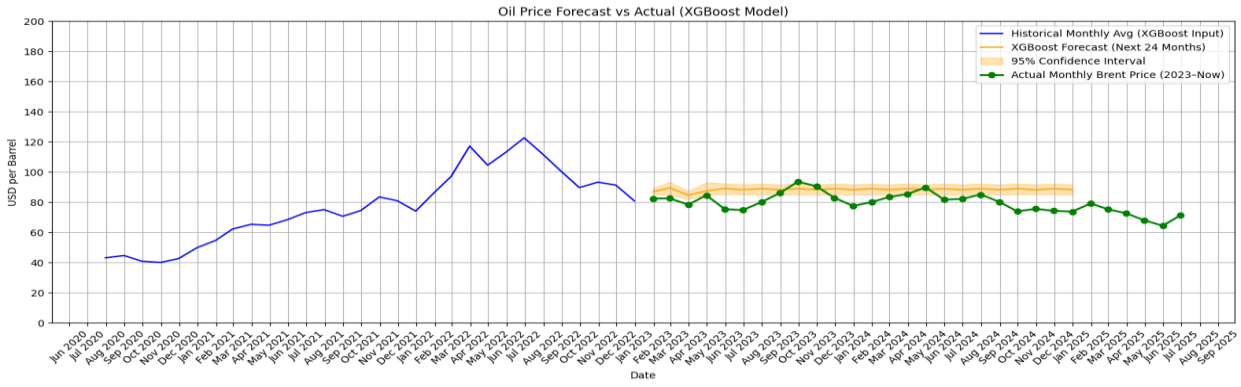
9.2 XGBoost Regressor

Why XGBoost Was Chosen
XGBoost is a powerful gradient boosting algorithm known for its strong performance on structured data,

including time series with engineered lag features. It handles non-linearity, noise, and feature interactions effectively, making it well-suited for complex datasets like oil prices.

Model Characteristics
XGBoost builds an ensemble of decision trees through boosting, incorporates regularization to prevent overfitting, and efficiently manages skewed or noisy data. However, it requires explicit creation of lag features to model temporal dependencies.

Motivation and Role in This Study
Oil prices are influenced by complex, non-linear patterns and external shocks. XGBoost was used to model these dependencies via lag-based features. It offered high forecasting accuracy with minimal tuning and provided a competitive benchmark against both statistical and deep learning models.



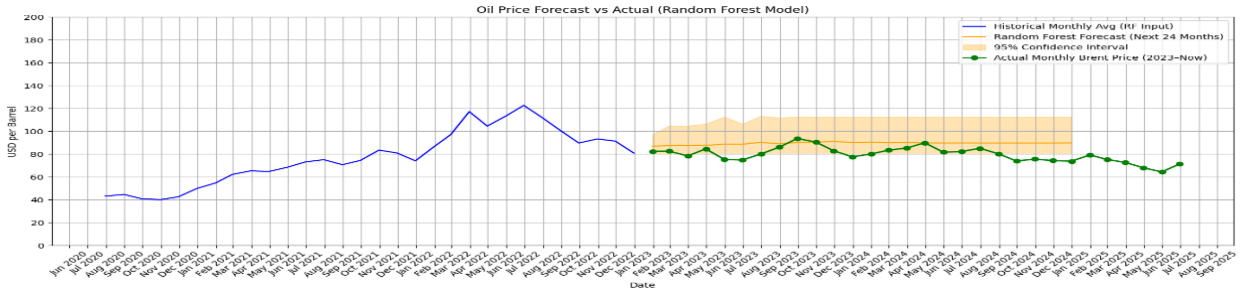
9.3 Random Forest Regressor

Why chosen:
Random Forests provide a reliable baseline in forecasting when using lagged variables. Unlike XGBoost, they are less sensitive to noise and hyperparameters, making them an excellent model for comparative analysis.

Model characteristics:

- Bagging ensemble of decision trees
- Handles non-linear relationships well
- Works best when time series is stationary or nearly stationary

Motivation:
Used primarily for comparison with XGBoost. While not as flexible as boosting methods, Random Forests help establish how much gain comes from boosting (XGBoost) versus bagging



Time Series Modelling Case Study

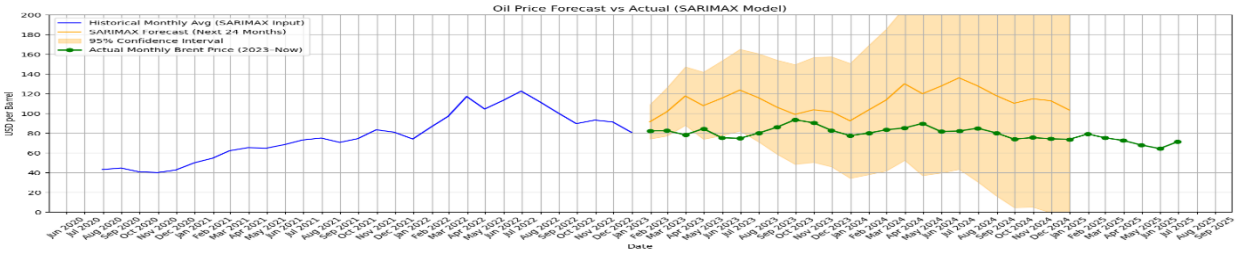
9.4 SARIMAX (Seasonal ARIMA with Exogenous Variables)

Why chosen:
SARIMAX is a traditional time series model that handles both **seasonality** and **trend** through differencing and seasonal components. It is interpretable and well-suited to series with **stable periodicity** and **low external influence**.

Model characteristics:

- (p,d,q) for ARIMA part: autoregressive, differencing, moving average, (P,D,Q,s) for seasonal component. Supports exogenous regressors if needed

Motivation:
SARIMAX was used to capture any deterministic seasonal trend in oil prices. Despite its statistical rigor, SARIMAX underperformed in this context likely due to the high volatility and external shocks inherent in oil markets.



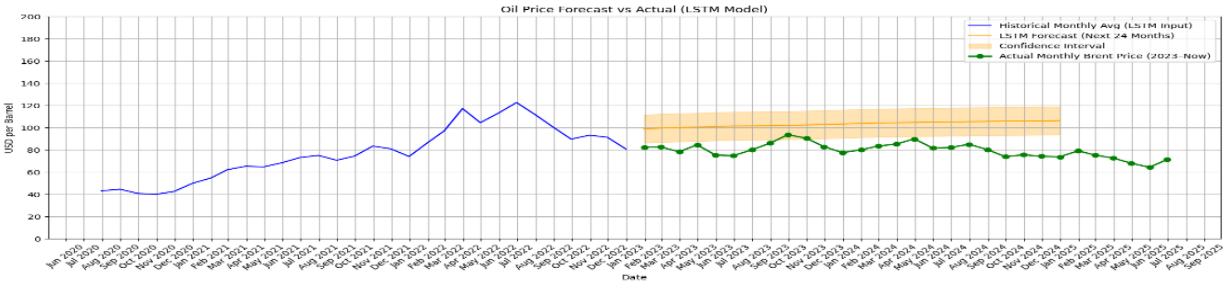
9.5 LSTM (Long Short-Term Memory Network)

Why chosen:
LSTMs are a class of recurrent neural networks capable of learning **long-term dependencies** and non-linear temporal relationships. LSTM is particularly powerful for modeling sequential data where recent and distant history both impact predictions.

Model characteristics:

- Stores information across long sequences
- Requires data scaling and reshaping
- Captures trend, seasonality, and memory effects without manual feature engineering

Motivation:
LSTM was included as a deep learning benchmark due to its theoretical ability to learn complex and noisy patterns in time series data. However, results showed that its performance was highly sensitive to hyperparameters and data quantity, which may have limited its effectiveness in this analysis.



Time Series Modelling Case Study

10. Results & Forecast Evaluation

To evaluate model performance, 24-month forecasts were compared with actual Brent crude oil prices obtained from FRED. The assessment combined quantitative metrics with visual inspection of forecast plots, including confidence intervals. Key evaluation metrics included Mean Error (ME) and Mean Absolute Error (MAE) to measure average and absolute forecast deviation, while Mean Percentage Error (MPE) and Mean Absolute Percentage Error (MAPE) captured relative accuracy. Root Mean Squared Error (RMSE) emphasized large deviations, and Median Absolute Error (MedAE) offered robustness to outliers. Additionally, residual autocorrelation (ACF1), R^2 Score (explained variance), Min-Max Error, and Pearson correlation were computed to assess model fit, range alignment, and linear consistency. Together, these metrics provided a comprehensive view of predictive accuracy and model reliability.

10.2 Forecast Accuracy Comparison

Model	**MAE**	**MAPE**	**RMSE**	**R^2 Score**	**Corr**	**ACF1**
XGBoost	**7.33**	**9.33%**	**8.43**	**-1.50**	0.06	0.56
Random Forest	8.29	10.57%	9.57	-2.22	0.14	0.53
Prophet	64.83	80.63%	68.06	-161.88	-0.21	0.53
LSTM	30.92	38.56%	33.51	-38.49	-0.06	0.57
SARIMAX	21.32	26.71%	22.06	-16.12	-0.45	0.57

10.3 Model Critique

Best Performing Model: XGBoost

- XGBoost demonstrated **superior accuracy across nearly all metrics**, especially in MAE, MAPE, and RMSE.
- The low error rates and relatively flat residual autocorrelation indicate stable predictions.
- Despite a negative R^2 (common in short test windows), XGBoost predictions closely follow actual trends in the plot.

Underperforming Models

- **Prophet** and **LSTM** significantly underperformed. Prophet likely failed due to:
    - Assuming strong regular seasonality in highly volatile oil prices.
    - Ignoring non-linear external shocks (e.g., geopolitical events, OPEC decisions).
- LSTM likely struggled because:
    - Deep models require **large training datasets**, which this monthly series lacked.
    - Forecast drifted rapidly without strong anchors (lags, features).

Time Series Modelling Case Study

Random Forest & SARIMAX

- RF performed decently but was slightly behind XGBoost, possibly due to its tendency to average outputs (bias-variance trade-off).
- SARIMAX, though interpretable, couldn't capture the **non-linear dynamics and volatility** in oil prices, leading to underfitting.

11. Limitations and Future Improvements

Key Limitations

- Limited Data: Monthly data restricted deep learning models like LSTM.
- No External Drivers: Models ignored macroeconomic and geopolitical influences.
- Stationarity Constraints: SARIMAX's assumptions don't align well with volatile oil prices.
- Tuning Scope: Hyperparameter tuning was limited, especially for complex models.

Suggested Improvements

1. Include Exogenous Variables: Incorporate factors like supply-demand, geopolitical risks, and currency rates.
2. Hybrid Models: Combine models (e.g., Prophet + XGBoost) for trend and residual learning.
3. Advanced Features: Use rolling statistics, volatility, or domain-driven features.
4. Deep Learning Enhancements: Use higher frequency data (daily/weekly), and explore attention-based models.
5. Robust Validation: Employ rolling-window cross-validation for more reliable performance assessment.

12. References

1. Taylor, S.J. & Letham, B. (2018). *Forecasting at scale*. The American Statistician, 72(1), 37–45. https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1380080
2. Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice* (2nd ed.). https://otexts.com/fpp2/
3. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2015). *Time Series Analysis: Forecasting and Control*. Wiley. https://www.wiley.com/en-us/Time+Series+Analysis%3A+Forecasting+and+Control%2C+5th+Edition-p-9781118675021
4. FRED - Federal Reserve Economic Data: https://fred.stlouisfed.org/