OIL DEMAND FORECASTING USING DEEP LEARNING TECHNIQUES

A PROJECT WORK I REPORT

Submitted by

BHUVANESHWARI C
21ADR009

SNEHAVARSHINI S
21ADR048

SOUNDHARYA M
21ADR050

in partial fulfillment of the requirements

for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI, ERODE-638 060

MAY 2024

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI, ERODE – 638 060

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that the Project Report titled **OIL DEMAND FORECASTING USING DEEP LEARNING TECHNIQUES** is the bonafide record of project work done by **BHUVANESHWARI C (21ADR009), SNEHAVARSHINI S (21ADR048), SOUNDHARYA M (21ADR050)** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Artificial Intelligence and Data Science of Anna University, Chennaiduring the year 2023-2024.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva voce examination held on.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI, ERODE – 638 060

MAY 2024

DECLARATION

We affirm that the Project Report titled **OIL DEMAND FORECASTING USING DEEP LEARNING TECHNIQUES** being submitted in partial fulfillment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed part of anyother project report or dissertation based on which a degree or award was conferred onan earlier occasion on this or any other candidate.

Date:

BHUVANESHWARI C
(21ADR009)

SNEHAVARSHINI S
(21ADR048)

SOUNDHARYA M
(21ADR050)

I certify that the declaration made by the above candidate is true to the best of my knowledge.

Date: Name & Signature of the Supervisor with seal

ABSTRACT

The goal of this research is to greatly advance forecasting methods in the oil business so that those involved may more confidently grasp new possibilities and handle market risks with skill. This study aims to provide decision-makers with the information they need to maximize resource allocation, reduce risks, and increase profitability in a dynamic industry environment by providing precise and timely projections. Accurate oil demand forecasting is becoming more and more necessary for the oil business to properly influence price and production plans. To meet this need, this work creates a comprehensive forecasting system that combines machine learning and statistical approaches. By applying techniques such as Moving Average, ARIMA, VARMA, EWMA, and LSTM (RNN), this study attempts to improve the accuracy and reliability of oil demand forecasts. Metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used in a painstaking evaluation procedure to thoroughly evaluate the forecasting models' effectiveness. This evaluation provides insightful information about each methodology's efficacy, enabling stakeholders to choose models with knowledge. In addition, a comparison study of several forecasting techniques, including statistical approaches and RNNs, is conducted to determine the best way for oil demand forecasting. The study also explores seasonal patterns in oil price and production, which enhances the forecasting model and promotes a better understanding of market dynamics.

ACKNOWLEDGEMENT

First and foremost, we acknowledge the abundant grace and presence of the almighty throughout different phases of the project and its successful completion.

We wish to express our sincere gratitude to our honorable Correspondent **Thiru.A.K.ILANGO B.Com., M.B.A., LLB.,** and other trust members for having providedus with all the necessary infrastructures to undertake this project.

We extend our hearty gratitude to our honorable Principal **Dr.V.BALUSAMY B.E.(Hons)., MTech., Ph.D.,** for his consistent encouragement throughout our college days.

We would like to express our profound interest and sincere gratitude to our respected Head of the department **Dr.C.S.KANIMOZHISELVI ME., Ph.D.,** for her valuable guidance.

A special debt is owed to the project coordinator **Ms. S. SANTHIYA B.E., M.E.**, Assistant Professor, Department of Artificial Intelligence for her encouragement and valuable advice that made us carry out project work successfully.

We extend our sincere gratitude to our beloved guide **Ms. O. ABHILA ANJU B.E., M.E.,** Department of Artificial Intelligence for her ideas and suggestions, which have been very helpful to complete the project.

We are grateful to all the faculty and staff members of the Department of Artificial Intelligence and persons who directly and indirectly supported this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ARIMA	:	Auto-Regressive Integrated Moving Average
VARMA	:	Vector AutoRegressive Moving Average
EWMA	:	Exponentially Weighted Moving Average
RNN	:	Recurrent Neural Network
LSTM	:	Long Short-Term Memory
RMSE	:	Root Mean Square Error
MAE	:	Mean Absolute Error
MAP	:	Mean Squared Error
CNN	:	Convolutional Neural Network

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Oil is a major energy source consumed by humans for various day-to-day activities. Thus, crude oil is one of the high-yielding sectors that utilize oil as its raw material. This oil is produced in different countries according to their demand levels, and the price may vary based on the abundance of this source. Factors like geological location, consumption level, production, imports, and exports may contribute to the change in the value of production level and its price. Many business investors are investing in the oil sector to earn a profit. About 31% of people use oil as a primary energy source. Early forecasting of oil production levels and price levels may help oil extracting and refinery companies produce according to demand levels and also help business sectors earn a profit when it is in demand.

The objective is to develop a model for early detection and forecasting of the oil production level and oil price level area in various locations. This is achieved by utilizing deep learning techniques with Statistical Models. By combining timeseries data analysis with deep learning models, the proposed method aims to accurately forecasts oil demand. This integration facilitates a holistic understanding of the factors influencing oil production levels and prices. By incorporating data from sources such as drilling reports, refinery capacities, geopolitical events, and economic forecasts, the model can provide more accurate and timely predictions. This enables stakeholders in the oil industry to make informed decisions regarding production levels, investment strategies, and market positioning.

To train and evaluate the forecasting model, a dataset comprising 20,000 records from the International Energy Agency is utilized. The performance of the selected statistical models, including ARIMA, VARMA, and EWMA, is compared to a traditional deep learning model, LSTM (RNN). This comparison aims to identify the most effective model architecture for accurately predicting oil production levels and prices. Additionally, the research proposes a method for evaluating the performance of the models by comparing their forecasting accuracy.

To enhance the accuracy of oil demand forecasting, models integrated with the Moving Average technique are utilized for improved performance. Additionally, seasonal trends are identified from the time series data, enabling better understanding and prediction of demand fluctuations. Evaluation of the models is conducted using standard metrics to comprehensively assess their performance. This approach combines statistical analysis with deep learning techniques, enhancing the precision of oil demand forecasts. Ultimately, these methods have the potential to facilitate more effective strategies for managing oil supply and demand dynamics.

1.2 OBJECTIVE

- To develop a forecasting model that accurately predicts the demand for oil needed for production forecasting and determines oil prices for real-world purposes.

- To analyze the efficiency of statistical models using evaluation metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

- To compare the performance of the RNN model with that of the statistical models to identify the better-performing model.

- To determine seasonal trends in oil production and pricing.

1.3 SCOPE

- Conduct feature importance analysis to identify the key factors driving oil demand and price fluctuations.
- The integration of statistical models and RNNs enhances forecasting accuracy, aiding in real-world decision-making for oil production and pricing strategies.

CHAPTER 2

SYSTEM ANALYSIS

2.1 LITERATURE REVIEW

In 2023, Ewees et al. [1] used the Aquila-optimized ANFIS model which offers an overview of the body of prior research on project management as well as studies conducted employing various models, algorithms, frameworks, and techniques. The ANFIS model, also known as AO-ANFIS, is employed in time-series forecasting of oil output. In tackling optimization issues, a recently created metaheuristic optimization algorithm performs noticeably well. In 2022, Abdulla et al. [2], To increase the prediction accuracy of ANFIS, AO has been used in this work to optimize its parameters. Not only have five ANFIS versions been updated, but five optimization algorithms—PSO, SCA, SSA, GWO, and ultimately SM—have also been used, and noteworthy outcomes have been obtained. RMSE, MAE, R^2, SD, time, AIC, and BIC have all been used to display the outcomes of the oilification of Alma Sila.

In 2022,Yanhui et al. [3] survey offers an overview of previously published publications as well as studies conducted in the field of project management utilizing various models, algorithms, frameworks, and techniques that have been presented. To maximize predictive modelling efficiency, a unique optimized VMD is implemented to split the Brent COP data into various modes. This project has been discussed at every level of the company and may include hundreds of workers in addition to one or more business divisions. Through the use of this high-accuracy method, the COP problem is thoroughly examined, the COP price's future fluctuation condition is muted, and the countermeasures against COP risks are suitably described to provide sustainable risk management for COP in both local and global.

In 2021, Nikoloas [4], introduced an innovative approach for forecasting oil production using a combination of empirical mode decomposition (EMD) and long short-term memory (LSTM) networks. We address a common challenge faced in time series forecasting, where the introduction of new data can disrupt the performance of pre-trained models due to changes in decomposition results. To tackle this issue, we develop a method based on the Dynamic Time Warping (DTW) algorithm to select appropriate decomposed components, known as Intrinsic Mode Functions (IMFs), as predictor variables. By optimizing the hyperparameters and architecture of the LSTM network using genetic algorithms (GA), we avoid the traditional "divide and conquer" strategy and enable direct prediction of oil production movements.

In 2022, Yoa Dong et al. [5], enhanced the model's ability to learn the variation trends and context information within the oil production series, leading to more accurate forecasting. We validate our method through case studies on SL and JD oilfields, comparing it against other approaches like EMD combined with Artificial Neural Networks (ANN) and Support Vector Machines (SVM). The results demonstrate that our EMD-LSTM model consistently outperforms these alternatives, achieving the lowest errors and highest determination coefficients in both cases. This suggests that our model can provide nearly perfect forecasts of oil production, offering significant value to reservoir engineers for real-time decision-making. Looking ahead, future research should focus on exploring uncertainty quantification and providing explanations derived from engineering insights for the predicted results. This will further enhance the applicability and robustness of our approach in practical reservoir management scenarios.

In 2019, Wei liu. [6] introduced a model utilizing both CNN and LSTM architectures to forecast WTI crude oil prices, addressing the challenge posed by its high volatility. By leveraging deep learning techniques renowned for modeling nonlinear dynamics, the hybrid CNN-LSTM model exhibited superior performance compared to individual models, achieving notably lower RMSE and MAPE values. Utilizing publicly available crude oil price data across short-, medium-, and long-term datasets, the study enables investors to tailor their investment strategies accordingly. The long-term model aids in devising enduring investment plans, while the short-term model facilitates swift decision-making. Extending the analysis to predict multiple steps, up to seven days into the future, the study compares the vector output CNN-LSTM model with the encoder-decoder

LSTM model. While both models demonstrated comparable accuracy, the former exhibited slight superiority over the latter. The research outlines four stages in the prediction model for oil prices, hinting at future investigations into the impact of varying the number of stages on overall model performance.

In 2019, Chen zang [7], developed a new method, the ARIMA-SVR-POT model, to better predict the risk of investing in crude oil futures. They compared it with three other methods and found that traditional models often underestimated the risk, especially in extreme situations. The new ARIMA-SVR-POT model, however, performed well across various confidence levels, indicating its reliability in estimating potential losses. In essence, this hybrid model seems to be a more accurate tool for assessing the risks associated with investing in crude oil futures, offering investors greater confidence in their decision-making process.

2.2 SUMMARY

This compilation of research studies showcases various innovative methodologies employed in forecasting oil production and prices. Techniques such as ANFIS, VMD, EMD-LSTM, and hybrid CNN-LSTM models are explored to enhance prediction accuracy. Evaluation metrics, including RMSE, MAE, R^2, and AIC/BIC, are utilized to validate model performance across different datasets. Notably, the integration of deep learning with traditional forecasting methods proves advantageous, demonstrating superior performance in capturing nonlinear dynamics and volatile market behaviors. Additionally, the development of hybrid models, such as ARIMA-SVR-POT, addresses the challenge of accurately assessing investment risks associated with crude oil futures. These findings offer valuable insights for stakeholders in the oil industry, enabling informed decision-making and risk management strategies amidst fluctuating market conditions.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

CPU type	:	Intel corei5 processors
Ram size	:	8 GB
Hard disk capacity	:	500 GB

3.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 10
Language	:	Python (version 3+)
Tool	:	Google Colab

3.3 SOFTWARE DESCRIPTION

3.3.1 Python

Python is simple to learn and a powerful programming language. Its extensive use of information structures and straightforward but effective approach to handling object-oriented programming. Python is a perfect language for prearranging and swift application advancement in many fields in the best stages due to its refined phrase structure and imperative composing, which are close to its made sense nature. The core ideas and capabilities of Python 3 are covered in this essay. A large number of the functions that come with Python when it is installed make up its standard library. There are numerous additional libraries that the Python language can use to experiment with more things available online. These libraries provide it strength and enable it to do a wide range of tasks. In the provided Python code snippets, several essential libraries and methods are employed to perform a range of tasks. The Pandas library is utilized for data manipulation and analysis, particularly for reading and processing data from CSV files, enabling data preprocessing. Scikit- Learn (sklearn) comes into play as a machine

learning library, assisting in model selection, data splitting for training and testing, and the calculation of accuracy scores.

3.3.2 Google Colab Notebook

Users may easily execute and exchange code with Google Colaboratory, a cloud-based tool for working in Jupyter notebooks. It provides a deep learning, machine learning, and data analysis environment powered by GPUs and TPUs. Users may keep their work on Google Drive and collaborate in real time. The sharing of Colab notebooks is similar to that of Sheets or Google Docs. Either follow these instructions for sharing files from Google Drive, or click the Share icon in the top right corner of any Colab notebook. For Colab laptops, Google Drive search functions are provided. Clicking the Colab logo in the upper left corner of the notebook view will make all of the notebooks in Drive visible. Choosing File > Open Notebook will allow you to can furthermore search for notebooks that you have just opened. Code is executed on your account-only virtual computer. The Colab service enforces a limited lifetime for virtual computers, after which they are removed after a period of inactivity. You may access any Colab notebook you've made using the File menu in Colab, by following these instructions, or by downloading it from Google Drive.

CHAPTER 4

PROPOSED SYSTEM

4.1 SYSTEM ARCHITECTURE

The proposed methodology for time series forecasting epitomizes a holistic and systematic approach, meticulously designed to navigate the complexities inherent in temporal data analysis. Its foundation lies in data preprocessing, where a comprehensive suite of techniques is employed to cleanse and refine the dataset, rectifying anomalies and extracting salient features to uncover underlying patterns, trends, and seasonality. This preparatory phase serves as a crucial precursor to the subsequent steps, ensuring the dataset's integrity and priming it for effective modeling. Following data preprocessing, the methodology meticulously partitions the dataset into training and testing sets, a strategic maneuver aimed at preserving temporal order and simulating real-world forecasting scenarios.

This division facilitates robust model evaluation and validation, enabling stakeholders to gauge the model's performance accurately and assess its generalization capabilities.

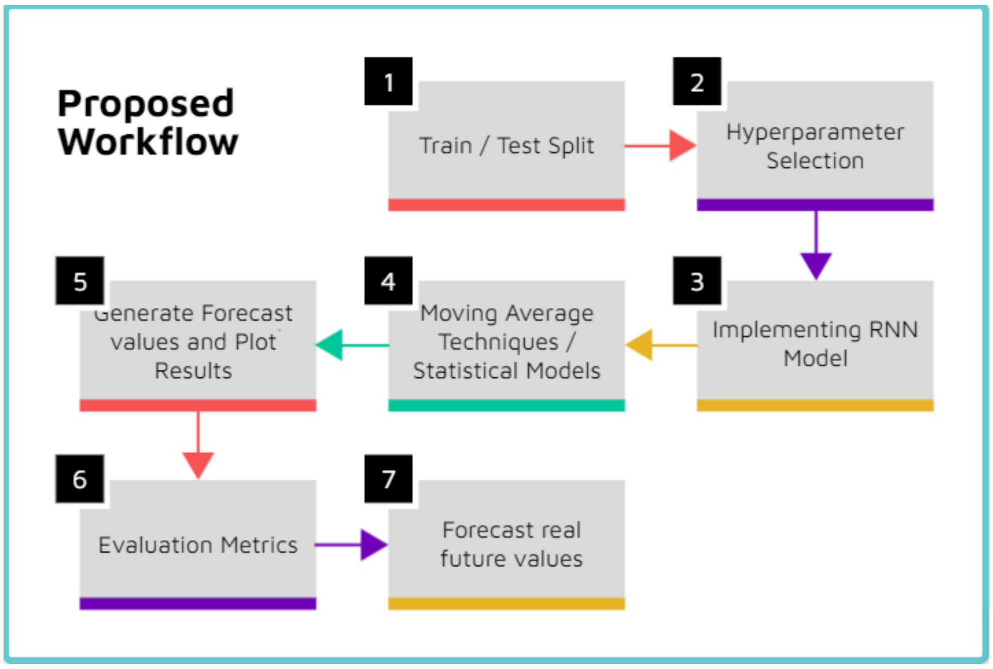

Proposed Workflow
1
Train / Test Split
2
Hyperparameter Selection
3
Implementing RNN Model
4
Moving Average Techniques / Statistical Models
5
Generate Forecast values and Plot Results
6
Evaluation Metrics
7
Forecast real future values

Fig. 4.1 Proposed model workflow

Hyperparameter selection emerges as a pivotal phase, characterized by an exhaustive exploration of the parameter space through sophisticated techniques like grid or random search. Coupled with cross-validation methodologies such as k-fold cross-validation, this process optimizes model parameters, mitigates overfitting, and enhances the model's predictive accuracy and reliability. Central to the methodology is the implementation of a Recurrent Neural Network (RNN) model, revered for its ability to capture temporal dependencies within sequential data.

Here, meticulous attention is devoted to architectural design and parameter optimization, harnessing the RNN's memory capabilities to discern intricate temporal patterns and deliver precise forecasts. Parallel to the RNN model, traditional statistical methods such as moving average techniques and exponential smoothing are enlisted, offering invaluable insights and serving as benchmarks to evaluate the RNN's performance while providing alternative forecasting strategies. The visualization of forecasted values alongside actual data emerges as a vital tool for interpretation, affording stakeholders a clear and intuitive understanding of the model's efficacy and guiding strategic decision-making processes.

Furthermore, an array of evaluation metrics, spanning (MAE), (MSE), (RMSE), and (MAPE), is meticulously computed to quantify forecast accuracy, empowering stakeholders to discern the model's reliability and make well-informed decisions. As the forecasting journey advances into real-time forecasting, the methodology exhibits its adaptability and resilience, with continuous monitoring and model updating ensuring accuracy and reliability in predicting future events. In summation, this methodology represents the pinnacle of data-driven forecasting techniques, offering a robust and versatile framework that is not only adaptable to dynamic environments but also continuously refined for optimal performance and relevance.

4.2 MODULE DESCRIPTION

4.2.1 Dataset Collection

4.2.1.1 International Energy Agency

The dataset for Price Forecasting is taken from the IEA. The International Energy Agency (IEA) is an independent intergovernmental organization established to promote energy security and cooperation among member countries. Its extensive, reliable, and transparent datasets make it an ideal choice for our project report. Utilizing IEA data ensures credibility, facilitates informed decision-making, and enhances the validity of our research findings, given its widely recognized benchmarks in the energy sector. It consists of about 21000 records which store the details of Oil Value.

Table 4.1 Feature description of the IEA dataset

COUNTRY	Country where the price holds.
PRODUCT	Type of Product (Oil)
FLOW	Total Price / Monthly Price
UNIT	Type of Currency
TIME	Time taken for Production
VALUE	The value of Product (Oil) in Market

4.2.1.2 Maritime Transport Costs (MTC)

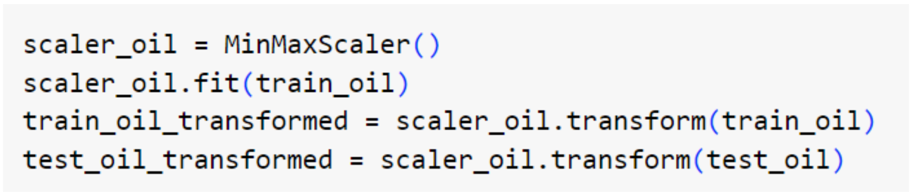
Maritime Transport Costs (MTC) refers to the expenses associated with shipping goods via sea routes. It encompasses various factors such as fuel costs, port fees, insurance, and vessel maintenance. Choosing MTC data for our project report ensures a comprehensive understanding of global trade dynamics. The transparency and reliability of MTC data enable informed decision-making, providing valuable insights into shipping trends, cost fluctuations, and their impact on international commerce. The dataset consists of about 25000 records.

Table 4.2 Feature description of the MTC dataset

DATE	Date of
OILRIG_COUNT	Type of Product (Oil)
OIL_PRICE	Total Price / Monthly Price
OILPRODUCTION_COUNT	Type of Currency
OILCONSUMPTION_COUNT	Time taken for Production
OILSTORAGE	The value of Product (Oil) in Market

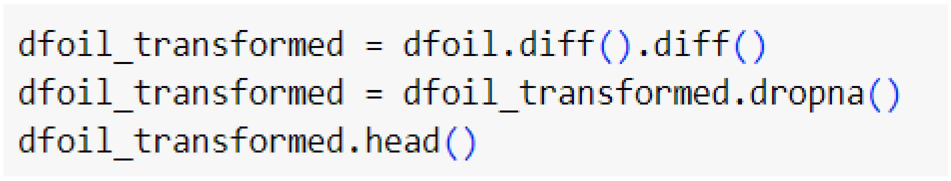
4.2.2 Data pre-processing

The initial phase of our oil demand forecasting project focuses on data preprocessing, encompassing essential steps such as MinMax scaling for normalization, second-order differentiation to capture trend changes.



```
scaler_oil = MinMaxScaler()
scaler_oil.fit(train_oil)
train_oil_transformed = scaler_oil.transform(train_oil)
test_oil_transformed = scaler_oil.transform(test_oil)
```

Fig. 4.2 Min Max Scaling

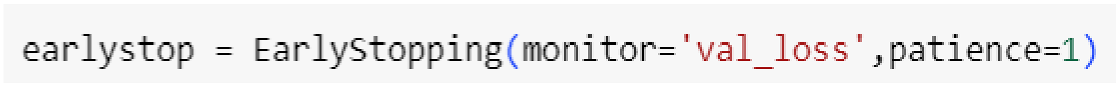


```
dfoil_transformed = dfoil.diff().diff()
dfoil_transformed = dfoil_transformed.dropna()
dfoil_transformed.head()
```

Fig 4.3 Second Order Differentiation

4.2.3 Regularization technique

In the neural network training phase of the oil demand forecasting project, regularization techniques like early stopping are integrated. This inclusion prevents overfitting, thereby enhancing the model's ability to generalize accurately for precise demand predictions.

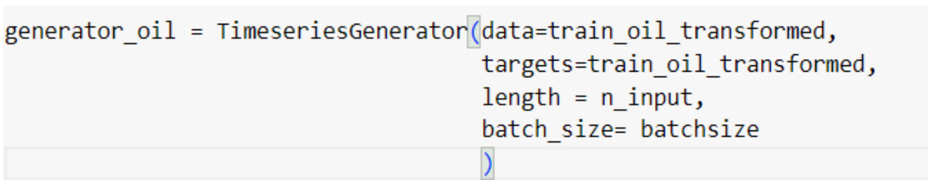


```
earlystop = EarlyStopping(monitor='val_loss',patience=1)
```

Fig 4.4 Early Stopping

4.2.4 Time Series Generator

The TimeSeriesGenerator function is essential for our oil demand forecasting project, enabling the creation of sequential input-output pairs crucial for training predictive models. It systematically generates batches of temporal data, efficiently incorporating historical oil demand data. By specifying parameters like the number of time steps and batch size, we tailor the generator to our forecasting needs. Its seamless integration into our workflow constructs training data capturing temporal dependencies, enhancing the accuracy of predictive models.



```
generator_oil = TimeseriesGenerator(data=train_oil_transformed,
                                    targets=train_oil_transformed,
                                    length = n_input,
                                    batch_size= batchsize
                                    )
```

Fig 4.5 Time Series Generator

4.2.5 Implementation of Statistical Models

Statistical models are effective instruments for data analysis and forecasting. These models allow us to extract meaningful information from complicated datasets by quantifying relationships, finding patterns, and using probability theory. Statistical models are essential to our project because they help us comprehend the underlying structure of the data and create prediction models that help us make decisions. Our goal in this study is to examine the foundations of statistical modelling in machine learning and show how it may be used to solve real-world issues.

Moving Average

Moving averages are used in time series analysis to average successive data points across a certain interval to smooth out volatility and show underlying patterns. It facilitates future value forecasting by lowering noise and enhancing the visibility of trends. Moving average enhances forecasting skills when combined with other models, such as ARIMA or exponential smoothing, by balancing out their shortcomings and enhancing their predictive strengths. This increases time series analysis's overall accuracy and resilience.

4.2.5.1 ARIMA

A popular statistical technique for time series forecasting, the ARIMA (AutoRegressive Integrated Moving Average) model is renowned for its ability to capture a variety of patterns and trends in sequential data. These three primary parts makes ARIMA model.

- The autoregressive (AR) component shows how the current observation and its lag values relate to one another. Regressing the present value on its historical values is what it entails.
- The integrated (I) part entails differentiating the raw data to stabilise the time

series and guarantee a consistent mean and variance over time.

- The link between the current observation and a residual error term from a moving average model applied to lag data is represented by the moving average (MA) portion.

The ARIMA model is expressed as:

$$y'(t) = c + \phi 1 * y'(t-1) + \cdots + \phi p * y'(t-p) + \theta 1 * \varepsilon(t-1) + \cdots + \theta q * \varepsilon(t-q) + \varepsilon t$$

Fig 4.6 ARIMA Model Formula

where y′t is the differential parameter, ϕ1 is the coefficient of AR component, p is the order of the AR component, θ1 is the coefficient of the first MA component, q is the order of the MA component and εt is the error component.

The significance of the ARIMA model is in its capacity to account for trends, seasonality, and other patterns while capturing both linear and nonlinear relationships in time series data. ARIMA models are an essential tool for analysts and researchers working with sequential data because they provide strong forecasting skills across a wide range of domains by taking into account residual errors, differencing to stabilise variance, and integrating prior observations.

4.2.5.2 VARMA

A strong extension of the ARIMA model, the VARMA (Vector Autoregressive Moving-Average) model is very helpful for evaluating and projecting multivariate time series data. It allows for the modelling of dependencies and interactions between several time series by combining moving average and autoregressive components across numerous variables.

The VARMA model is represented as a set of equations in which error terms are regressed using a moving average procedure, together with each variable's own lagged values and the lagged values of other variables. It has the following mathematical representation:

$$X_t = \sum_{i=1}^{p} \Phi_i X_{t-i} + \epsilon_t - \sum_{j=1}^{q} \Theta_j \, \epsilon_{t-j}$$

Fig 4.7 VARMA Model Formula

Where Xt is the vector of time series of variables, Φi is the matrices of autoregression parameters, Θj is the matrices of moving average parameter.

The value of VARMA comes from its capacity to represent the dynamic interactions and feedback loops among several time series variables, which makes it appropriate for modelling intricate systems with interactions and feedback loops. VARMA models provide useful insights into the dynamics of multivariate data and allow precise forecasting of future values by taking into account the combined behaviour of variables over time and adding lagged effects and error factors.

4.2.5.3 EWMA

A moving average model variation called the exponential weighted moving average (EWMA) is frequently used in time series analysis to smooth data and identify patterns or behavioural shifts over time. It gives more weight to new data points and progressively lessens the relevance of earlier observations by exponentially reducing the weights assigned to previous observations.

The EWMA model may be stated mathematically as:

$$\mathrm{EWMA}(t) = a * x(t) + (1-a) * \mathrm{EWMA}(t-1)$$

Fig 4.8 EWMA Model Formula

Where a is the EWMA parameter of values 0 or 1, x(t) = value of signal x at time t.

The EWMA model's simplicity and adaptability to shifting patterns and trends in time series data are what make it so important. Analysts can manage the trade-off between stability in identifying long-term patterns and reactivity to current changes by modifying the smoothing value. The EWMA model is also computationally efficient and appropriate for real-time applications that call for quick updates and modifications.

4.2.6 Implementation of Neural Network Models

4.2.6.1 LSTM

LSTM (Long Short-Term Memory) is a well-liked recurrent neural network (RNN) architecture in deep learning. Identifying long-term dependencies is one of its strong points, which makes sequence prediction work ideal for it. Unlike ordinary neural networks, LSTM can handle whole data sequences rather than just individual data points since it contains feedback connections.

For this reason, it excels at recognising and predicting patterns in sequential data, including time series, text, and speech. These three LSTM unit components are referred to as "gates". They control the data that enters and leaves the memory cell, sometimes referred to as the LSM cell. First is the forget gate; second is the input gate; and third is the output gate.

1.Forget Gate

Forget Gate:

- $f_t = \sigma\,(x_t * U_f + H_{t-1} * W_f)$

(4.1)

2.Input Gate

Input Gate:

- $i_t = \sigma\,(x_t * U_i + H_{t-1} * W_i)$

(4.2)

3.Output Gate

Output Gate:

- $o_t = \sigma\,(x_t * U_o + H_{t-1} * W_o)$

(4.3)

Where Xt: Input at the current timestamp t, Ui: weight matrix of input, Ht-1: A hidden state at the previous timestamp, Wi: Weight matrix of input associated with hidden state.

4.3 UNVEILING SEASONAL PATTERNS

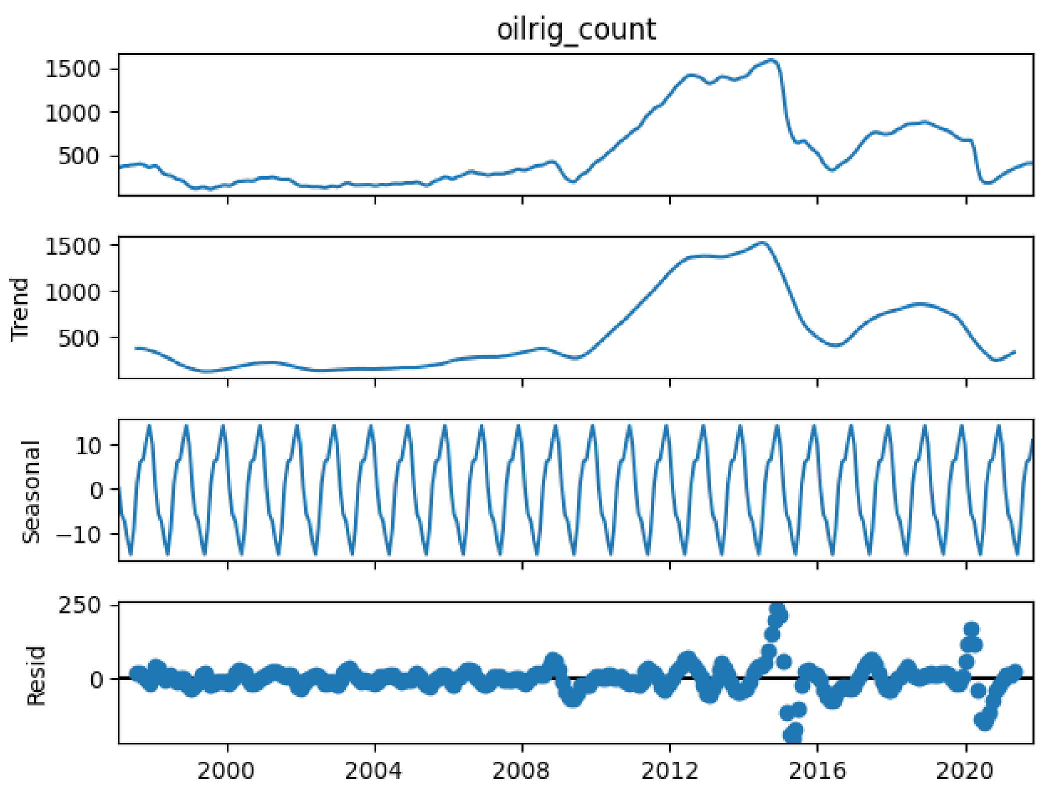


Fig 4.9 Seasonal Decomposition of Data

A statistical method called seasonal decomposition is used to separate a time series into its trend, seasonal, and residual components. Analysts can better comprehend the underlying patterns and structures in the data by using this breakdown.

- Original Time Series Plot: Shows the observed values across time without decomposition by displaying the raw time series data.
- Trend Component Plot: Shows the time series' long-term directionality or behavior, emphasizing general trends while downplaying shorter-term oscillations.
- A seasonal component plot: Shows daily, weekly, monthly, or annual pattern, shows recurring cycles or patterns in the time series at predetermined intervals.
- The residual component plot: which represents random fluctuations or noise, shows the unpredictability in the time series that remains after trend and seasonal components have been eliminated.

4.4 VISUALIZATION

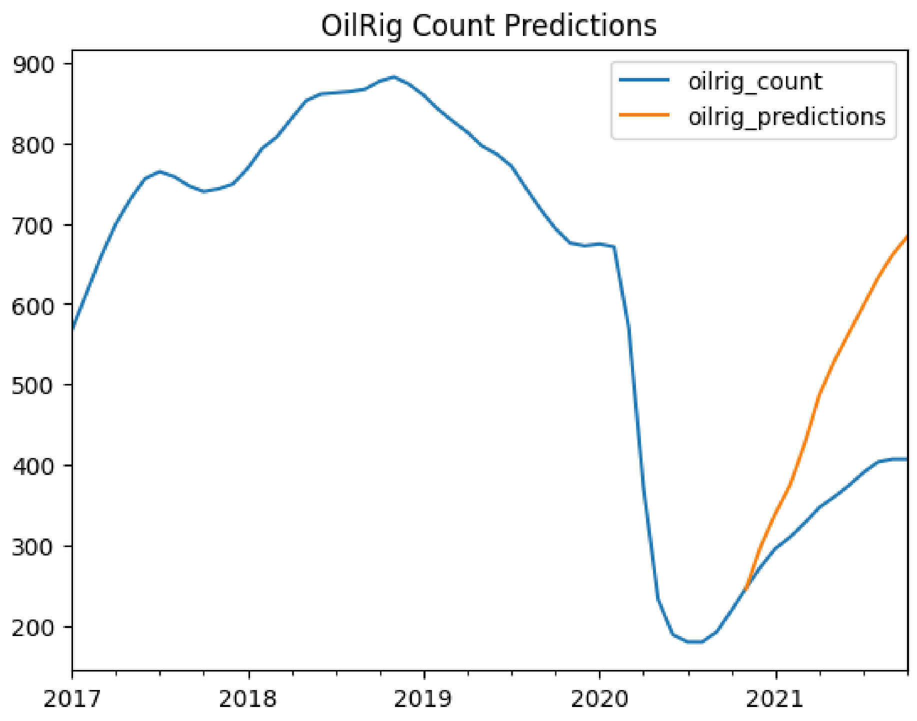


Fig 4.10 Prediction of Oil Region Count using LSTM

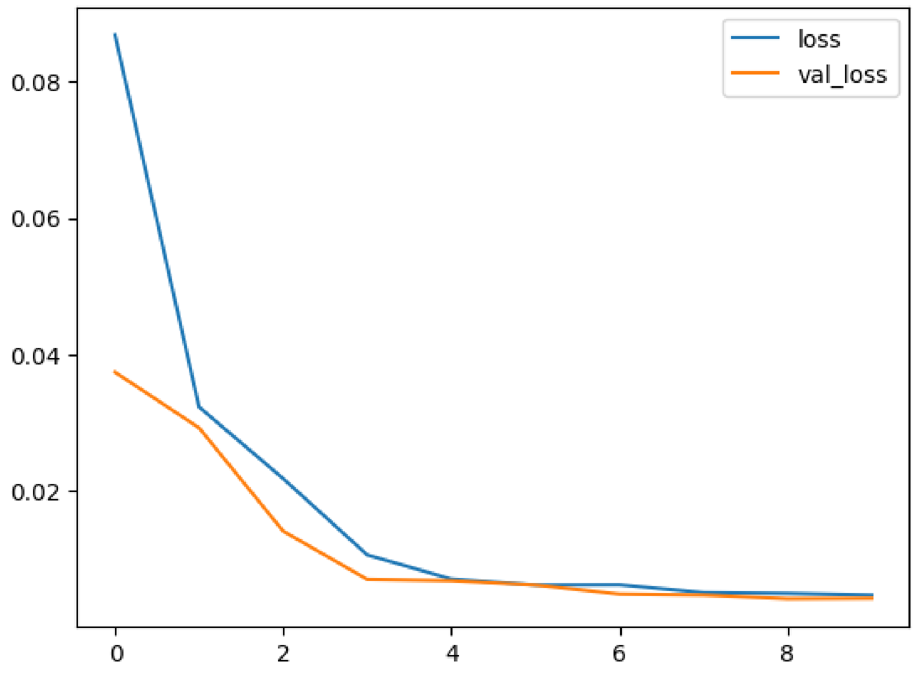


Fig 4.11 Loss graph of LSTM Model

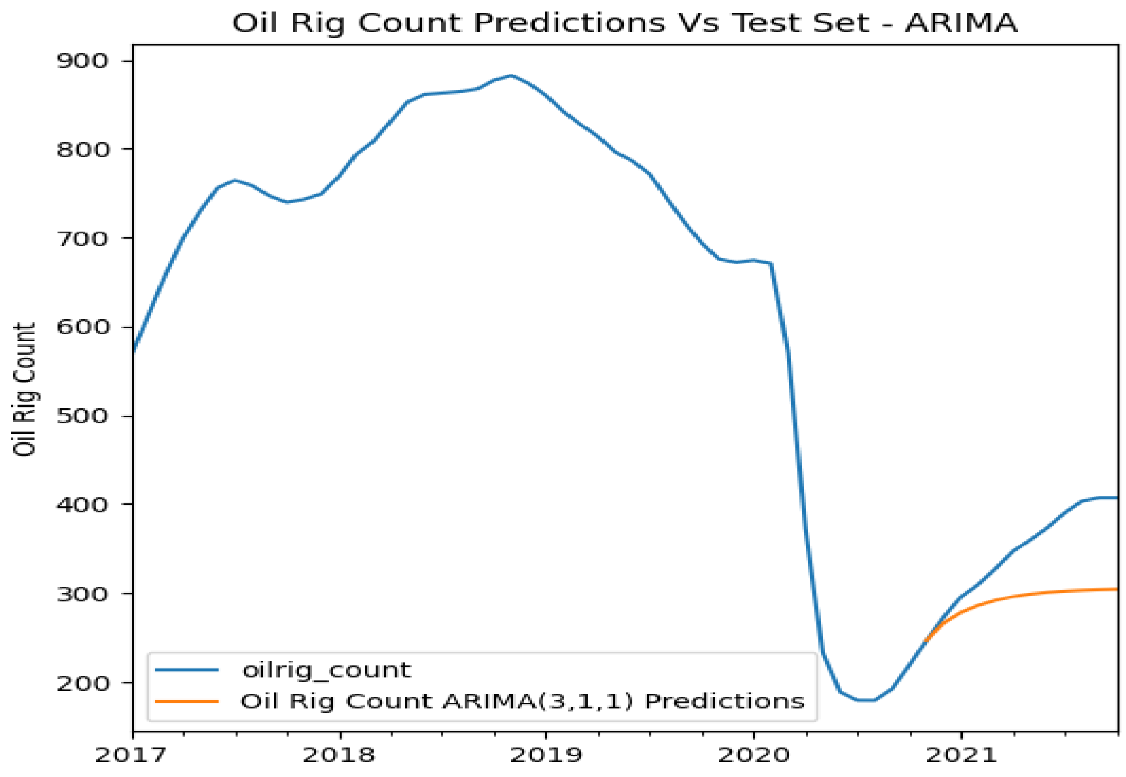


Fig 4.12 Prediction of Oil Region Count using ARIMA

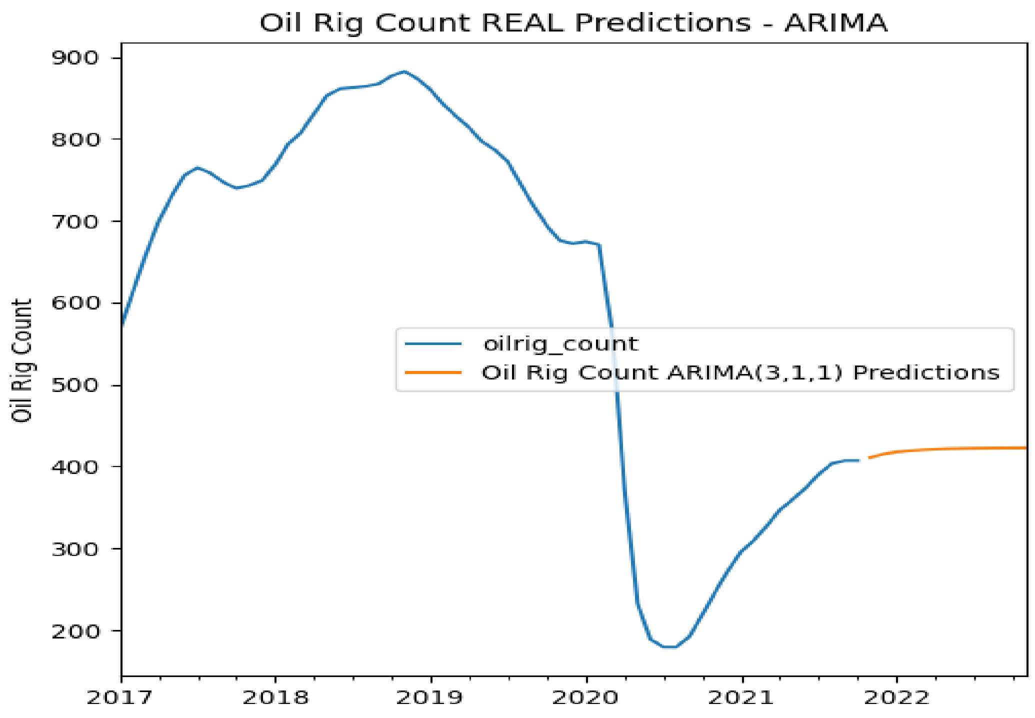


Fig 4.13 Forecasting of Oil Region Count using ARIMA

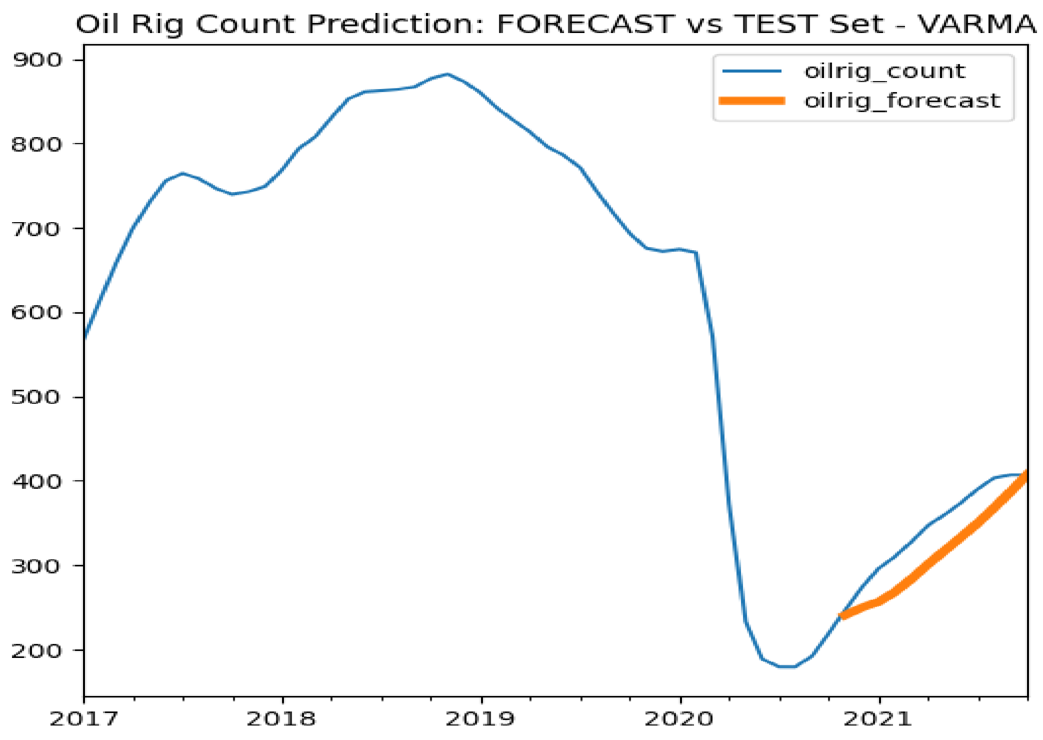


Fig 4.14 Prediction of Oil Region Count using VARMA

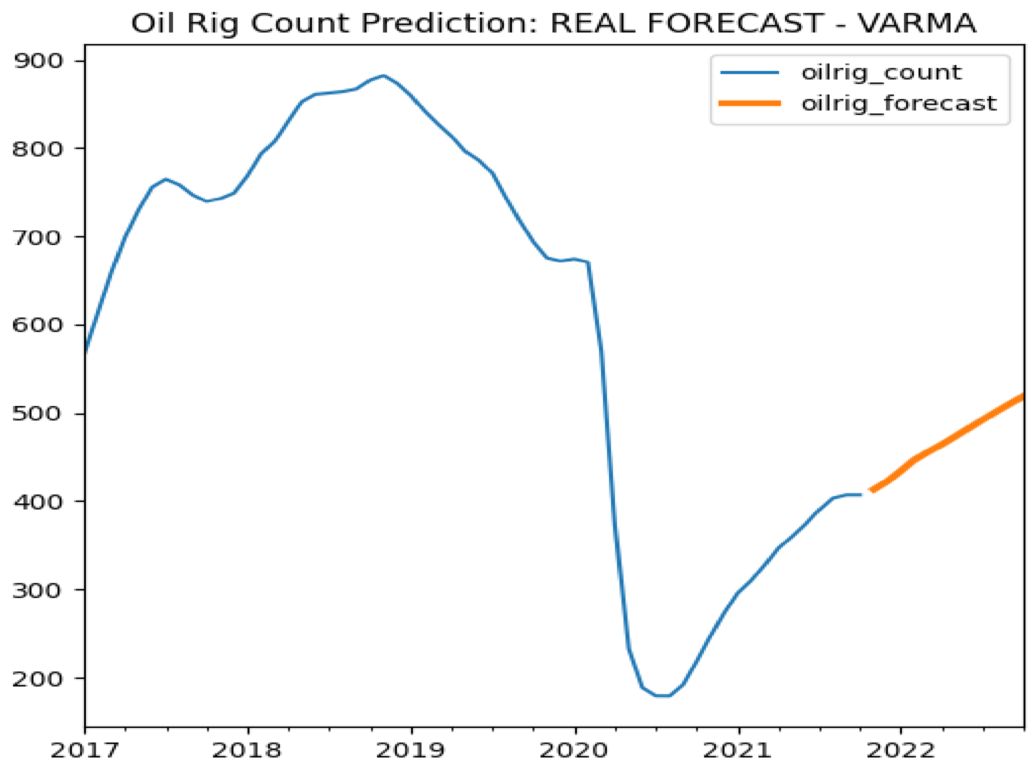


Fig 4.15 Forecasting of Oil Region Count using VARMA

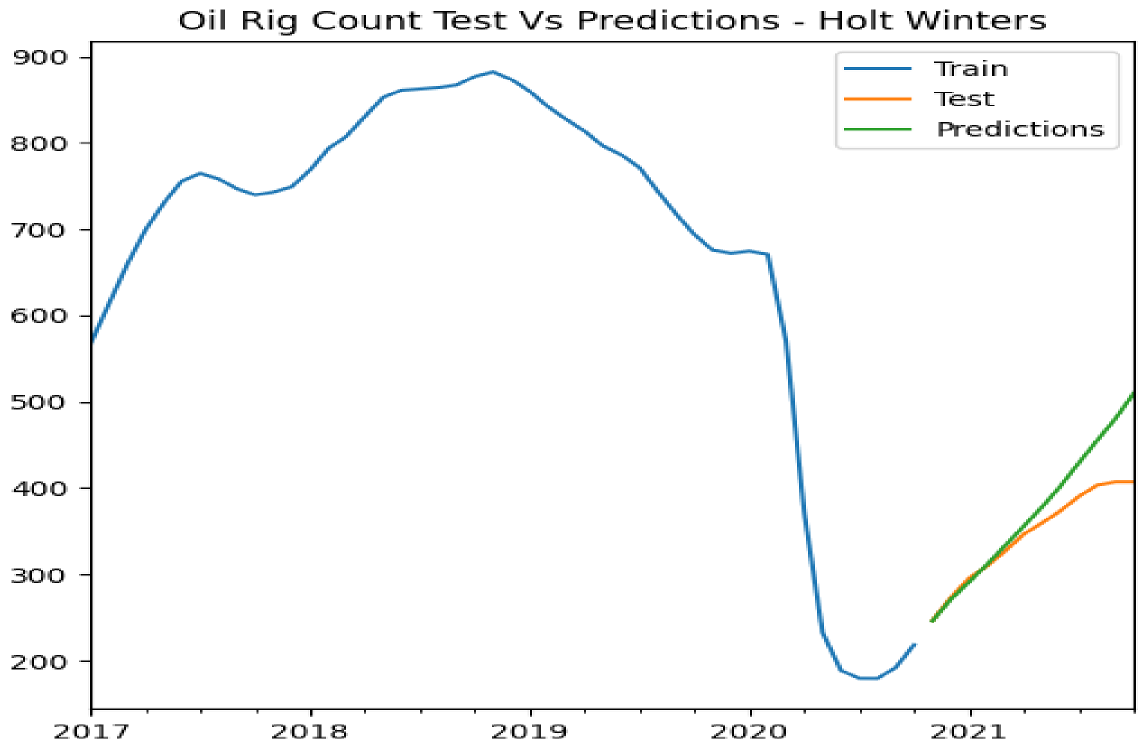


Fig 4.16 Prediction and Forecasting of Oil Region Count using EWMA

CHAPTER 5

PERFORMANCE ANALYSIS

The effectiveness of a regression model can be assessed using metrics like Mean Absolute Error(MAE), Root Mean Square Error(RMSE) and using Coefficient of Determination (r2 – value).

The average of the absolute difference between the dataset's actual and anticipated values is represented by the mean absolute error. It calculates the dataset's average residual.

$$MAE = \frac{|(yi - yp)|}{n} \tag{5.1}$$

The average of the squared differences between the data set's original and anticipated values is called the mean square error. It calculates the residuals' variance. Root The square root of Mean Squared error is called Mean Squared Error. It calculates the residuals' standard deviation.

$$RMSE = \sqrt{\frac{\sum(y_i - y_\theta)2}{n}} \tag{5.2}$$

The coefficient of determination, or R2, is a statistical measure used in regression models that indicates the extent to which the independent variable or variables may be used to predict the variation in the dependent variable. Stated differently, R2 quantifies the degree to which the regression model accounts for the observed data.

$$\mathrm{R}^2 = 1 - \frac{RSS}{TSS} \tag{5.3}$$

Table 5.1 Comparison of different models

Methodologies	**MAE**	**RMSE**	**R2**
LSTM	4196828996.6404395	64782.937542538464	-0.98487498810923
ARIMA	55.252	66.322	-0.598
VARMA	31.681	35.022	0.554
EWMA	29.241	43.342	0.317

CHAPTER 6

RESULTS AND DISCUSSION

In summary, the analysis of oil demand forecasting evaluated various models using MAE, RMSE, and R2. LSTM and ARIMA models demonstrated poor performance, with high MAE, RMSE, and negative R2 values. In contrast, VARMA and EWMA models showed relatively better performance, with lower MAE, RMSE, and positive R2 values. VARMA had a MAE of 31.681, RMSE of 35.022, and R2 of 0.554, while EWMA exhibited comparable results with a slightly lower MAE of 29.241, RMSE of 43.342, and R2 of 0.317. Further optimization may enhance forecasting accuracy.

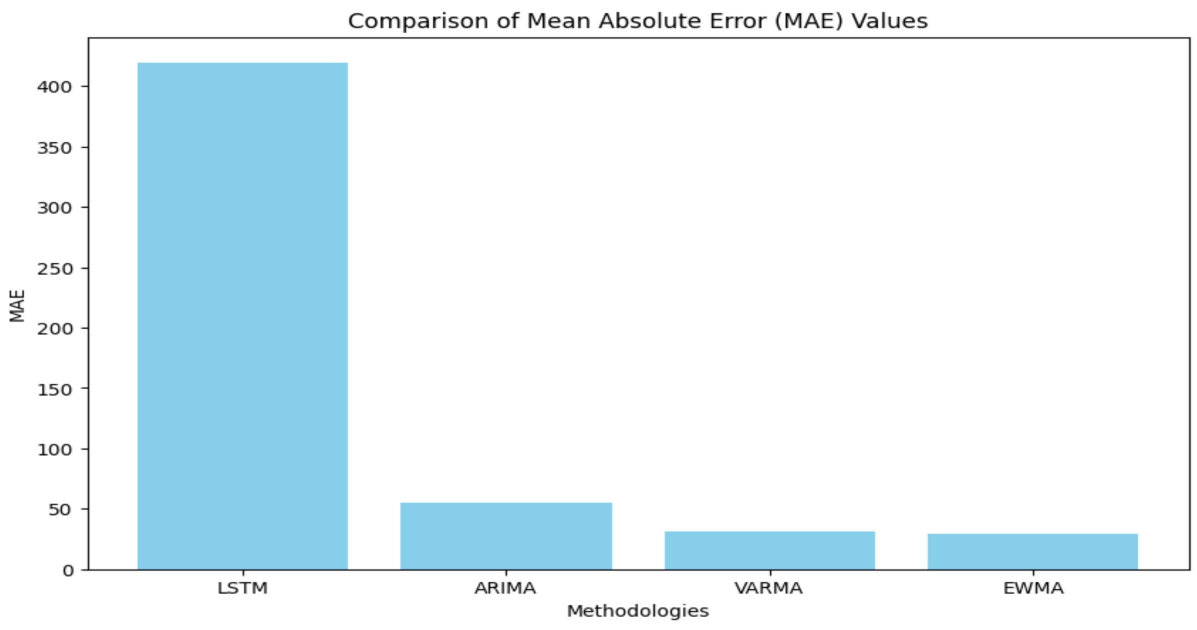


Fig.6.1 Comparison of different models

These results imply that both VARMA and EWMA models outperformed LSTM and ARIMA models in predicting oil demand, with EWMA showing a slightly better performance in terms of MAE and R2. Further analysis and fine-tuning may be necessary to optimize forecasting accuracy.

CHAPTER 7

CONCLUSION

In conclusion, the project underscores the critical importance of precise oil demand forecasting for a myriad of stakeholders, ranging from industry players to policymakers. While traditional models like LSTM and ARIMA reveal certain limitations in their predictive capabilities, the VARMA and EWMA models emerge as promising alternatives, showcasing potential in bolstering forecasting accuracy. Moreover, the project illuminates the path towards future enhancements, envisioning the integration of cutting-edge technologies such as real-time data collection through IoT sensors. By leveraging these advancements, the project's efficacy can be further augmented, empowering stakeholders with timely and insightful predictions to navigate the dynamic landscape of oil demand. In essence, the project not only contributes to the realm of oil demand forecasting but also lays the foundation for continuous innovation and refinement, ensuring its relevance and impact in an ever-evolving domain.

CHAPTER 8

APPENDICES

8.1APPENDIX – 1 CODING

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import time
import tensorflow
from numpy.random import seed
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.model_selection import TimeSeriesSplit
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM
from tensorflow.keras.callbacks import EarlyStopping
from pylab import rcParams
dfoil = pd.read_csv('dfoil.csv',index_col=0,parse_dates=True)
freq_time = 'M'
nobs = 12
n_input = 12
batchsize = 1
#RNN
LSTM_neurons = 20
activation_fct = 'relu'
n_features_oil = len(dfoil.columns)
n_epochs = 30
```

```
dfoil = dfoil.resample(freq_time).mean()
dfoil.index.freq = freq_time
results = seasonal_decompose(dfoil['oilrig_count'])
results.plot();
results.seasonal.loc['2016-04':'2017-04'].plot()
train_oil = dfoil.iloc[:len(dfoil)-nobs]
test_oil = dfoil.iloc[len(dfoil)-nobs:]
scaler_oil = MinMaxScaler()
scaler_oil.fit(train_oil)
train_oil_transformed = scaler_oil.transform(train_oil)
test_oil_transformed = scaler_oil.transform(test_oil)
earlystop = EarlyStopping(monitor='val_loss',patience=1)
generator_oil = TimeseriesGenerator(data=train_oil_transformed,
                                targets=train_oil_transformed,
                                length = n_input,
                                batch_size= batchsize
                                )
X,Y = generator_oil[0]
X.shape
Y
model_oil = Sequential()
model_oil.add(LSTM(LSTM_neurons, activation=activation_fct, input_shape=(n_input,
n_features_oil)))
model_oil.add(Dense(10, activation=activation_fct))
model_oil.add(Dense(n_features_oil))
model_oil.compile(optimizer='adam', loss='mse')
model_oil.fit_generator(generator_oil, epochs=n_epochs, validation_data=generator_oil,
callbacks=[earlystop])
model_oil.summary()
plt.plot(range(len(model_oil.history.history['loss'])),model_oil.history.history['loss'])
losses = pd.DataFrame(model_oil.history.history)
losses.plot()
testoil_predictions = [] #store the predictions
evalbatch_oil = train_oil_transformed[-n_input:]
```

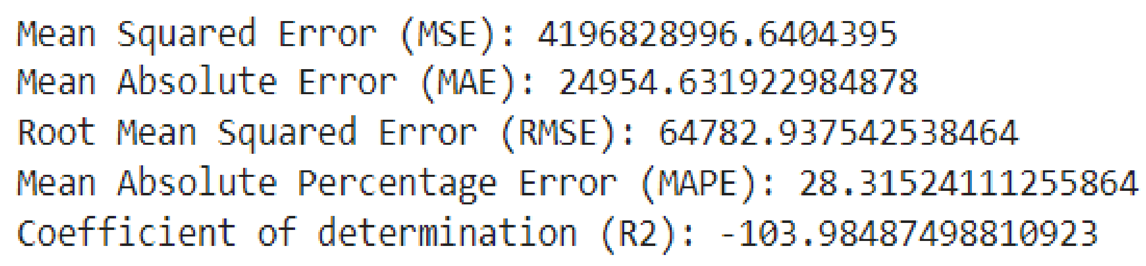
```
print(f'Before Reshape:\n {evalbatch_oil}, \nShape: {evalbatch_oil.shape}')
evalbatch_oil = evalbatch_oil.reshape((1,n_input,n_features_oil)) #reshape to what RNN wants
print(f'\nAfter Reshape:\n {evalbatch_oil}, \nShape: {evalbatch_oil.shape}')
for i in range(nobs):
    one_pred = model_oil.predict(evalbatch_oil)[0] #Grab the numbers only, not the array
    testoil_predictions.append(one_pred)
    evalbatch_oil = np.append(evalbatch_oil[:,1:,:],[[one_pred]],axis=1)
testoil_predictions = scaler_oil.inverse_transform(testoil_predictions)
testoil_predictions =
pd.DataFrame(data=testoil_predictions,columns=test_oil.columns,index=test_oil.index)
testoil_predictions.rename(columns={'oilrig_count':'oilrig_predictions'},inplace=True)
dfoil.loc['2017':]['oilrig_count'].plot(legend=True,title='OilRig Count Predictions')
testoil_predictions['oilrig_predictions'].plot(legend=True)
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
import matplotlib.pyplot as plt
# Calculate metrics
mse = mean_squared_error(test_oil, testoil_predictions)
mae = mean_absolute_error(test_oil, testoil_predictions)
rmse = np.sqrt(mse)
mape = np.mean(np.abs((test_oil - testoil_predictions) / test_oil)) * 100
r2 = r2_score(test_oil, testoil_predictions)
# Print metrics
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Percentage Error (MAPE):", mape)
print("Coefficient of determination (R2):", r2)
# Plot actual vs predicted
plt.figure(figsize=(10, 6))
plt.plot(test_oil.index, test_oil['oilrig_count'], label='Actual')
plt.plot(testoil_predictions.index, testoil_predictions['oilrig_predictions'], label='Predicted')
plt.title('OilRig Count: Actual vs. Predicted')
plt.xlabel('Date')
plt.ylabel('OilRig Count')
plt.legend()
plt.show()
**Arima Model**
!pip install pmdarima
from statsmodels.tsa.arima.model import ARIMA,ARIMAResults
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tools.eval_measures import rmse, mse, meanabs
from sklearn.model_selection import TimeSeriesSplit
import time
from pmdarima import auto_arima
from pylab import rcParams
rcParams['figure.figsize'] = 6,6
nobs = 12
stepwise_fit = auto_arima(dfoil['oilrig_count'], start_p=0, start_q=0,
                    max_p=6, max_q=3, m=nobs,
                    seasonal=True,
                    d=None, trace=True,
                    error_action='ignore',
                    suppress_warnings=True,
                    stepwise=True)
train = dfoil.iloc[:len(dfoil)-nobs]
test = dfoil.iloc[len(dfoil)-nobs:]
oil_model = ARIMA(train['oilrig_count'],order=(3,1,1))
oil_results = oil_model.fit()
start=len(train)
end=len(train)+len(test)-1
oil_predictions = oil_results.predict(start=start, end=end, dynamic=False, typ='levels').rename('Oil
Rig Count ARIMA(3,1,1) Predictions')
title = 'Oil Rig Count Predictions Vs Test Set - ARIMA'
ylabel='Oil Rig Count'
xlabel=''
```

```
ax = dfoil['oilrig_count'].loc['2017':].plot(legend=True,title=title)
oil_predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
RMSE1 = rmse(dfoil['oilrig_count'].loc[test.index.min():],oil_predictions)
MSE1 = mse(dfoil['oilrig_count'].loc[test.index.min():],oil_predictions)
MAE1 = meanabs(dfoil['oilrig_count'].loc[test.index.min():],oil_predictions)
r2 = r2_score(dfoil['oilrig_count'].loc[test.index.min():], oil_predictions)
print(f"""Oil Rig ARIMA(3,1,1) RMSE: {RMSE1:.3f}
Oil Rig ARIMA(3,1,1) MSE: {MSE1:.3f}
Oil Rig ARIMA(3,1,1) MAE: {MAE1:.3f}
Oil Rig ARIMA(3,1,1) R2: {r2:.3f}\n
Oil Rig Mean Value: {dfoil['oilrig_count'][-nobs:].mean():.3f}
Oil Rig std Value: {dfoil['oilrig_count'][-nobs:].std():.3f}
Oil Rig Percent Change: {RMSE1/dfoil['oilrig_count'][-nobs:].mean()*100:.3f}%
""")
startTime = time.time()
oil_model = ARIMA(dfoil['oilrig_count'],order=(3,1,1))
oil_results = oil_model.fit()
start=len(dfoil)
end=len(dfoil)+nobs
oil_predictions = oil_results.predict(start=start, end=end, dynamic=False, typ='levels').rename('Oil
Rig Count ARIMA(3,1,1) Predictions')
executionTime = (time.time() - startTime)
print('Execution time in seconds: ' + str(executionTime))
title = 'Oil Rig Count REAL Predictions - ARIMA'
ylabel='Oil Rig Count'
xlabel=''
ax = dfoil['oilrig_count'].loc['2017':].plot(legend=True,title=title)
oil_predictions.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel);
```

Varma Model Implementationt

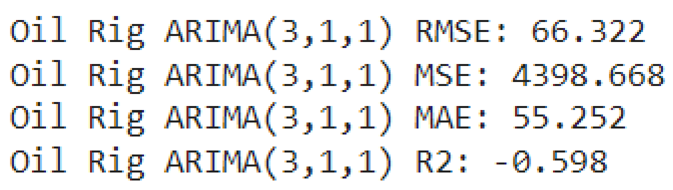
```
from statsmodels.tsa.statespace.varmax import VARMAX, VARMAXResults
from statsmodels.tsa.stattools import adfuller, grangercausalitytests
from statsmodels.tools.eval_measures import rmse, mse, meanabs
from sklearn.model_selection import TimeSeriesSplit
import time
from pylab import rcParams
dfoil_transformed = dfoil.diff().diff()
dfoil_transformed = dfoil_transformed.dropna()
dfoil_transformed.headzz()
nobs=12
train, test = dfoil_transformed[0:-nobs], dfoil_transformed[-nobs:]
oil_model = VARMAX(train, order=(2,2), trend='ctt')
oil_results = oil_model.fit(maxiter=1000, disp=True)
oil_results.summary()
dfoil_forecast = oil_results.forecast(nobs)
dfoil_forecast
dfoil_forecast['oilrig_count1d'] = (dfoil['oilrig_count'].iloc[-nobs-1]-dfoil['oilrig_count'].iloc[-nobs-
2])+dfoil_forecast['oilrig_count'].cumsum()
dfoil_forecast['oilrig_forecast'] = dfoil['oilrig_count'].iloc[-nobs-1] +
dfoil_forecast['oilrig_count1d'].cumsum()
dfoil['oilrig_count'].loc['2017':].plot(legend=True)
dfoil_forecast['oilrig_forecast'].plot(legend=True,linewidth=4,title='Oil Rig Count Prediction:
FORECAST vs TEST Set - VARMA')
RMSE1 = rmse(dfoil['oilrig_count'].loc[test.index.min():],dfoil_forecast['oilrig_forecast'])
MSE1 = mse(dfoil['oilrig_count'].loc[test.index.min():],dfoil_forecast['oilrig_forecast'])
MAE1 = meanabs(dfoil['oilrig_count'].loc[test.index.min():],dfoil_forecast['oilrig_forecast'])
r2 = r2_score(dfoil['oilrig_count'].loc[test.index.min():],dfoil_forecast['oilrig_forecast'])
print(f"Oil Rig VARMA(2,2) RMSE: {RMSE1:.3f}")
print(f"Oil Rig VARMA(2,2) MSE: {MSE1:.3f}")
print(f"Oil Rig VARMA(2,2) MAE: {MAE1:.3f}\n")
print(f"Oil Rig VARMA(2,2) R2: {r2:.3f}\n")
print(f"Oil Rig Mean Value: {dfoil['oilrig_count'][-nobs:].mean():.3f}")
print(f"Oil Rig std Value: {dfoil['oilrig_count'][-nobs:].std():.3f}")
print(f"Oil Rig Percent Change: {RMSE1/dfoil['oilrig_count'][-nobs:].mean()*100:.3f}%")
oil_model = VARMAX(dfoil_transformed, order=(2,2), trend='ctt')
oil_results = oil_model.fit(maxiter=1000, disp=False)
dfoil_forecast = oil_results.forecast(nobs)
dfoil_forecast['oilrig_count1d'] = (dfoil['oilrig_count'].iloc[-1]-dfoil['oilrig_count'].iloc[-
2])+dfoil_forecast['oilrig_count'].cumsum()
dfoil_forecast['oilrig_forecast'] = dfoil['oilrig_count'].iloc[-1] +
dfoil_forecast['oilrig_count1d'].cumsum()
dfoil['oilrig_count'].loc['2017':].plot(legend=True)
dfoil_forecast['oilrig_forecast'].plot(legend=True,linewidth=3,title='Oil Rig Count Prediction:
REAL FORECAST - VARMA')
**EWMA**
from statsmodels.tsa.seasonal import seasonal_decompose
import datetime
from pylab import rcParams
from statsmodels.tsa.holtwinters import SimpleExpSmoothing #for single exponential smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing #for double exponential smoothing
from statsmodels. tools.eval_measures import rmse, mse, meanabs
from sklearn.model_selection import TimeSeriesSplit
import time
from pylab import rcParams
oilmodel_exp =
ExponentialSmoothing(train_oil['oilrig_count'],trend='add',seasonal='mul',seasonal_periods=nobs)
oilmodel_exp = oilmodel_exp.fit()
dfoil_forecast = oilmodel_exp.forecast(nobs)
train_oil['oilrig_count'].loc['2017':].plot(legend=True,label='Train',title='Oil Rig Count Test Vs
Predictions - Holt Winters')
test_oil['oilrig_count'].plot(legend=True,label='Test')
dfoil_forecast.plot(legend=True,label='Predictions');
RMSE1 = rmse(dfoil['oilrig_count'].loc[test_oil.index.min():],dfoil_forecast)
MSE1 = mse(dfoil['oilrig_count'].loc[test_oil.index.min():],dfoil_forecast)
MAE1 = meanabs(dfoil['oilrig_count'].loc[test_oil.index.min():],dfoil_forecast)
r2 = r2_score(dfoil['oilrig_count'].loc[test_oil.index.min():],dfoil_forecast)
print(f"""Oil Rig VAR(4) RMSE: {RMSE1:.3f}
Oil Rig VAR(4) MSE: {MSE1:.3f}
Oil Rig VAR(4) MAE: {MAE1:.3f}
Oil Rig VAR(4) R2: {r2:.3f}    \n
Oil Rig Mean Value: {dfoil['oilrig_count'][-nobs:].mean():.3f}
Oil Rig std Value: {dfoil['oilrig_count'][-nobs:].std():.3f}
Oil Rig Percent Change: {RMSE1/dfoil['oilrig_count'][-nobs:].mean()*100:.3f}%""")
```

8.2 APPENDIX – 2 SCREENSHOTS



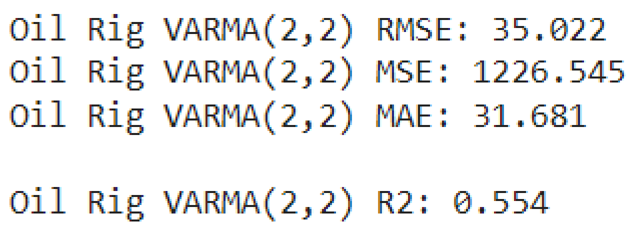
```
Mean Squared Error (MSE): 4196828996.6404395
Mean Absolute Error (MAE): 24954.631922984878
Root Mean Squared Error (RMSE): 64782.937542538464
Mean Absolute Percentage Error (MAPE): 28.31524111255864
Coefficient of determination (R2): -103.98487498810923
```

Fig A2.1: Output of LSTM



```
Oil Rig ARIMA(3,1,1) RMSE: 66.322
Oil Rig ARIMA(3,1,1) MSE: 4398.668
Oil Rig ARIMA(3,1,1) MAE: 55.252
Oil Rig ARIMA(3,1,1) R2: -0.598
```

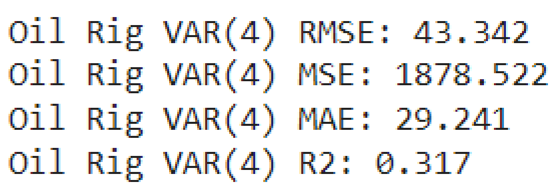
Fig A2.2:Output of ARIMA



```
Oil Rig VARMA(2,2) RMSE: 35.022
Oil Rig VARMA(2,2) MSE: 1226.545
Oil Rig VARMA(2,2) MAE: 31.681

Oil Rig VARMA(2,2) R2: 0.554
```

Fig A2.3:Output of VARMA



```
Oil Rig VAR(4) RMSE: 43.342
Oil Rig VAR(4) MSE: 1878.522
Oil Rig VAR(4) MAE: 29.241
Oil Rig VAR(4) R2: 0.317
```

Fig A2.4: Output of EWMA

REFERENCES

1. AlRassas, A.M.; Al-qaness, M.A.A.; Ewees, A.A.; Ren, S.; Abd Elaziz, M.; Damaševiˇcius, R.; Krilaviˇcius, T. Optimized ANFIS Model Using Aquila Optimizer for Oil Production Forecasting. Processes 2021, 9, 1194.
2. Deep learning framework for predictive modeling of crude oil price for sustainable management in oil markets:Abdullah Ali Salamai Received 18 March 2022, Revised 18 August 2022, Accepted 20 August 2022, Available online 28 August 2022, Version of Record 2 September 2022.
3. Forecasting Crude Oil Prices: a Deep Learning based Model:Yanhui Chen , Kaijian He, Geoffrey K.F. Tso,School of Economics and Management, Shanghai Maritime University, Shanghai, 201306, China
4. A deep learning method for the prediction of ship fuel consumption in real operational conditions:Mingyang Zhang a, Nikolaos Tsoulakos b, Pentti Kujala a, Spyros Hirdaris Department of Mechanical Engineering, Marine Technology Group, Aalto University, Espoo, Finland Laskaridis Shipping Co. Ltd., Athens, Greece.
5. A novel crude oil price forecasting model using decomposition and deep learning networks:Yao Dong , He Jiang , Yunting Guo , Jianzhou Wang School of Statistics and Data Science, JiangXi University of Finance and Economics, Nanchang JiangXi 330013, China
6. Forecasting oil production using ensemble empirical model decomposition based Long Short-Term Memory neural network:Wei Liu, Wei David Liu, Jianwei Gu China University of Petroleum (East China), China Received 10 August 2019, Revised 28 December 2019, Accepted 29 January 2020, Available online 1 February 2020, Version of Record 27 February 2020.
7. Forecasting value-at-risk of crude oil futures using a hybrid ARIMA-SVR-POT model: Chen Zhang and Xinmiao Zhou |Article notes Copyright and License information PMC Disclaimer-Business School, Ningbo University, 315211, Ningbo, China