

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
National Higher School of Statistics and Applied Economics



Dissertation Submitted for Obtaining the

Master's Degree in Statistics and Economic Foresight

The Use of Machine Learning Models in Forecasting the Crude Oil Prices Study Case: Saharan Blend

Presented by: ZENATI Kamelia

Supervised by: Dr. DJOUADI Issam

Evaluated by: Dr. ASRI Ayoub
Dr. CHINE Lazhar

Academic Year 2023-2024

Acknowledgment

I would like to thank my supervisor, **Dr. DJOUADI Issam** in guiding me throughout this research journey. Your expertise, patience, and encouragement have been invaluable in shaping my understanding of the topic and helping me to overcome the challenges I faced. I am grateful for your unwavering support and guidance.

I would also like to acknowledge the **National Higher School of Statistics and Applied Economics** for providing me with a conducive academic environment that has enabled me to pursue my research interests. The resources and facilities available at the school have been instrumental in the success of this project.

I would like to extend my gratitude to my teachers, particularly **Miss. AGOUNI Abir** and **Dr. ASRI Ayoub** who have taught me the skills and knowledge that have been essential in completing this thesis. Your guidance and mentorship have been instrumental in shaping my academic journey.

Finally, I present my endless gratitude for **Dr. CHINE Lazhar** and again **Dr. ASRI Ayoub** for accepting to evaluate my work. Your expertise and encouragement have been instrumental in shaping this thesis.

Dedication

*Thank you, **Allah**, for your endless blessings, guidance, and for always being there for me. I am forever grateful for your love and protection.*

To my beloved parents, Mom, Dad, thank you for your unconditional love, unwavering support, and for always believing in me. You have been my rock, my inspiration, and my guiding light. I am who I am today because of you.

To my dear sister, Dalya and brother Alilou, thank you for your love, laughter, and for being my constant companions. You have made my life richer and more meaningful.

To my whole family, my Grand-parents, Uncles and Aunts, particularly Sabrina and Lamia for being such amazing aunts to almost older sisters. To my cherished friends Lolla, Ibtihel, thank you

for your friendship, support, and for always being there for me. Your presence in my life is a true blessing.

To my school and esteemed teachers, thank you for providing me with an excellent education and for nurturing my growth and development. Your dedication and guidance have been invaluable.

I am truly blessed to have such amazing people in my life. Thank you all for your love, support, and for helping me become the person I am today. I am forever grateful.

Abstract

Algeria's economy is heavily dependent on revenues from the hydrocarbon sector, which accounted for 19% of GDP, 93% of exports, and 75% of budget revenues between 2016-2021. This reliance on oil and gas makes Algeria vulnerable to price fluctuations in global energy markets. The research systematically evaluated the performance of different machine learning models in forecasting crude oil prices, assessing the impact of dataset characteristics on model effectiveness and also the combination of the models into a hybrid model. The results demonstrate that hybrid machine learning models outperform standalone models, but model performance is influenced by dataset characteristics. Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models exhibit superior forecasting accuracy on datasets with larger dataframes and fewer explanatory variables, while Support Vector Regression (SVR), Random Forest, and XGBoost models perform better on datasets with fewer time frames but a larger number of input features. These insights can guide stakeholders in making informed decisions and developing effective investment strategies in the oil market, which is critical for Algeria given its heavy reliance on hydrocarbon revenues.

Keywords—

Hybrid model, LSTM, CNN, SVR, XGBoost, RandomForest, Crude oil, Petroleum, Oil industry, Prediction

Résumé

L'économie algérienne dépend fortement des revenus du secteur des hydrocarbures, qui ont représenté 19% du PIB, 93% des exportations et 75% des recettes budgétaires entre 2016 et 2021. Cette dépendance au pétrole et au gaz rend l'Algérie vulnérable aux fluctuations des prix sur les marchés mondiaux de l'énergie. La recherche a systématiquement évalué la performance de différents modèles d'apprentissage automatique dans la prévision des prix du pétrole brut, en évaluant l'impact des caractéristiques de l'ensemble des données sur l'efficacité du modèle et aussi la combinaison des modèles dans un modèle hybride. Les résultats démontrent que les modèles hybrides d'apprentissage automatique sont plus performants que les modèles autonomes, mais que la performance du modèle est influencée par les caractéristiques de l'ensemble des données. Les modèles de mémoire à long terme (LSTM) et de réseau neuronal convolutif (CNN) affichent une précision de prévision supérieure sur les ensembles de données comportant des périodes plus longues et moins de variables explicatives, tandis que les modèles de régression vectorielle de soutien (SVR), de forêt aléatoire et de XGBoost sont plus performants sur les ensembles de données comportant moins de périodes, mais un plus grand nombre de caractéristiques d'entrée. Ces informations peuvent aider les parties prenantes à prendre des décisions éclairées et à élaborer des stratégies d'investissement efficaces sur le marché pétrolier, ce qui est essentiel pour l'Algérie compte tenu de sa forte dépendance à l'égard des revenus des hydrocarbures.

Mots clés— Modèle hybride, LSTM, CNN, SVR, XGBoost, Forêts aléatoires, Pétrole, Industrie pétrolière, Prédiction

Contents

List of Figures	iii
List of Tables	v
Introduction	1
1 Oil industry	4
Section 1: Global Oil Industry	4
1. Introduction to the oil industry	4
2. Market dynamics and pricing:	8
Section 02: Algerian oil industry	13
1. Introduction to the Algerian oil industry:	13
2. Market dynamics and the economic impact:	14
2 Machine Learning in the oil industry	16
Section 1: Theoretical basis of Machine Learning	16
1. Introduction	16
2. Types of Machine Learning algorithms:	18
3. Fundamentals of predictive modeling	29
Section 02: Literature reviews	31
1. Predicting using machine learning:	31
1. Forecasting using machine learning:	34
3 The use of Machine Learning to forecast oil prices	36
Section1: Data overview	36
1. Data presentation	36
2. Data Variables	37
Section 2: Forecasting oil prices	41
1. Data preparation	41
2. Forecasting Oil Prices	48
Section 3: Results	74
1. Models comparison	74
2. Data comparison	76
Conclusion	77
Bibliography	79

Appendix	1
A Daily data	1
A.1 The SVR model	1
A.2 The RandomForest model	3
A.3 The XGBoost model	5
A.4 The LSTM model	7
A.5 The CNN model	11
A.6 Hybrid RandomForest-XGBoost model	14
A.7 Hybrid CNN-LSTM model	15
B Monthly data	20
B.1 The SVR model	20
B.2 The Random Forest model	23
B.3 The XGBoost model	25
B.4 The LSTM model	27
B.5 The CNN model	29
B.6 The Hybrid RandomForest-XGBoost model	31
B.7 The Hybrid CNN-LSTM model	34

List of Figures

1.1	Oil Industry Sectors Source: ResearchGate by [Musa, 2023]	6
1.2	Global Oil Market Trends Source: [OPEC, 2024]	11
1.3	Oil and natural gas basins and pipeline infrastructure in Algeria Source: [IEA, 2016]	14
2.1	Relationship between data science, AI, and machine learning Source: [Otokiti, 2020]	16
2.2	Machine Learning types Source: [Dojo, 2020]	19
2.3	Supervised Machine learning Source: [GeeksforGeeks, 2023]	20
2.4	Unsupervised Machine Learning	24
3.1	Monthly dataset Source: By the student from Python	41
3.2	Daily dataset Source: By the student from Python	42
3.3	Oil price fluctuations through time	43
3.4	daily data Oil price fluctuations before and after denoising	48
3.5	The GridSearchCV node Source: By the student using Python	49
3.6	Scatter plot of actual prices vs predicted prices using SVR model Source: By the student using Python	50
3.7	RandomForest node Source: By the student using Python	51
3.8	Line plot of actual prices vs predicted prices using Random Forest Source: By the student using Python	52
3.9	Scatter plot of actual prices vs predicted prices using XGBoost	53
3.10	Loss curve: validation vs training Source: By the student using Python	55
3.11	Line plot of actual prices vs predicted prices using LSTM Source: By the student using Python	56
3.12	CNN Loss curve: validation vs training Source: By the student using Python	57
3.13	Scatter plot of actual prices vs predicted prices using CNN Source: By the student using Python	58
3.14	Scatter plot of actual prices vs predicted prices using hybrid RandomForest-XGBoost Source: By the student using Python	60
3.15	Scatter plot of actual prices vs predicted prices using Hybrid CNN-LSTM model Source: By the student using Python	62
3.16	GridSearchCV node for SVR Source: By the student using python	63
3.17	Comparison of actual prices vs predicted prices using SVR Source: By the student using Python	64
3.18	RandomForest node Source: By the student using Python	65
3.19	Scatter plot of actual prices vs predicted prices using RandomForest Source: By the student using Python	66

3.20	Scatter plot of actual prices vs predicted prices using XGBoost Source: By the student using Python	67
3.21	Scatter plot of actual prices vs predicted prices using LSTM Source: By the student using Python	69
3.22	Scatter plot of actual prices vs predicted prices using CNN	70
3.23	Scatter plot of actual prices vs predicted prices using Hybrid RandomForest-XGBoost	72
3.24	Scatter plot of actual prices vs predicted prices using Hybrid CNN-LSTM	74
3.25	Models' performance on daily data	75
3.26	Models' performance on monthly data	75

List of Tables

3.1	Daily dataset	36
3.2	Monthly dataset	37
3.3	Correlation matrix: Daily data variables	37
3.4	Correlation matrix: monthly data variables	38
3.5	Number of NAs in monthly data	44
3.6	Monthly dataset before and after handling NAs	44
3.7	Number of NAs in daily data	45
3.8	Daily dataset before and after handling NAs	45
3.9	Daily data outliers	46
3.10	Outlier-free Daily data	46
3.11	Comparison table between actual prices and predicted values using SVR Source: By the student using Python	50
3.12	Comparison table between actual prices and predicted values using RandomForest Source: By the student using Python	51
3.13	Comparison between actual prices and predicted values using XGBoost Source: By the student using Python	53
3.14	Comparison table of actual prices and predicted values using LSTM Source: By the student using Python	56
3.15	Comparison table of actual prices and predicted values using CNN Source: By the student using Python	58
3.16	Comparison table of actual prices and predicted values using Hybrid RandomForest- XGBoost Source: By the student using Python	59
3.17	Hybrid CNN-LSTM model architecture Source: By the student using Python	61
3.18	Comparison table of actual prices and predicted values using Hybrid CNN-LSTM Source: By the student using Python	62
3.19	Comparison between actual prices and predicted values using SVR Source: By the student using Python	64
3.20	Comparison table between actual prices and predicted values using Random Forest Source: By the student using Python	65
3.21	Comparison table between actual prices and predicted values using XGBoost Source: By the student using Python	67
3.22	Comparison table between actual prices and predicted values using LSTM Source: By the student using Python	68
3.23	Comparison table between actual prices and predicted values using CNN Source: By the student using Python	70
3.24	Comparison table between actual prices and predicted values using Hybrid RandomForest- XGBoost Source: By the student using Python	71

3.25 Comparison table between actual prices and predicted values using Hybrid CNN-LSTM Source: By the student using Python	73
--	----

Introduction

Crude oil is a strategic and essential resource for Algeria, serving as a cornerstone in its economy, politics, and societal framework. It accounts 19% of the GDP and represents a significant portion of the country's exports (93%) and budget revenues (75%). [Bank, 2023]. It plays a pivotal role in the world economy between 2016 and 2021. As a major exporter of crude oil, Algeria's economy is heavily dependent on the revenue generated from oil exports, making it particularly vulnerable to the volatility of oil price. [Desorgues and AFP, 2021]

The fluctuations in crude oil prices can have significant impacts on the global and local markets, leading to economic instability, inflation, and financial losses for oil-dependent countries like Algeria [Guechari, 2017]. These price swings are often driven by a complex interplay of factors, including geopolitical tensions, supply and demand dynamics, and exchange rates, making them inherently difficult to predict using traditional statistical and econometric techniques. [Yan et al., 2021b] [Fernandois and Medel, 2020]

In recent years, machine learning (ML) has emerged as a promising solution to the challenge of crude oil price forecasting. ML algorithms can capture the nonlinear and dynamic nature of oil price movements by leveraging historical data and various influencing factors. [Abdullah and Zeng, 2010] Also a hybrid deep learning model combining CNN and LSTM has been proposed for crude oil price prediction. This model can make both one-step and multi-step predictions, demonstrating its potential for practical applications. [Aldabagh et al., 2023]

By adopting ML-based approaches, decision makers can potentially mitigate the risks associated with oil price volatility, enabling them to make more informed decisions and better manage their economic resources.

Problem statement:

Accurately forecasting crude oil prices is a critical challenge due to the complex and volatile nature of the oil market. While traditional econometric models have limitations in capturing the nonlinear dynamics of oil price movements, the application of advanced machine learning techniques holds promise for improving forecasting accuracy. However, the relative performance of different machine learning models, as well as the impact of dataset characteristics on model effectiveness, is not well understood and to do so, we need to answer this question:

How do different machine learning models and dataset characteristics influence the accuracy of crude oil price forecasts, and which approaches yield the most reliable predictions?

In order to be able to answer the problematic, we need to divide it into secondary questions as follow:

⇒ Do hybrid machine learning models outperform individual models in forecasting crude oil prices?

- ⇒ Do machine learning models perform better datasets with larger dataframe or the opposite?
- ⇒ How does the number of input features significantly impact the forecasting accuracy of machine learning models?

The working hypotheses that form the basis for this work and that should be confirmed or denied at the end of this study are summarized as follow:

- ⇒ Hybrid machine learning models will outperform individual models in forecasting crude oil prices.
- ⇒ Some machine learning models will exhibit superior forecasting performance when trained on datasets of larger sample size.
- ⇒ The number of input features will have a significant impact on the forecasting accuracy of machine learning models with models trained on datasets with more relevant features outperforming models trained on datasets with fewer features.

The objective of this work is to systematically evaluate the performance of different machine learning models in forecasting crude oil prices, while also assessing the impact of dataset characteristics on model effectiveness. By comparing the predictive accuracy of individual and hybrid machine learning models, as well as analyzing the influence of dataset size and feature selection, this research aims to provide insights into the most reliable and robust approaches for crude oil price forecasting.

The motivation for this work stems from the critical importance of crude oil as a strategic resource for Algeria's economy, politics, and society. Given the significant contribution of oil exports to Algeria's GDP, trade balance, and government revenues, accurately forecasting oil prices is essential for effective economic planning and risk management.

This research aims to fill this gap by conducting a comprehensive evaluation of machine learning models for crude oil price forecasting, with a focus on identifying the most reliable and robust approaches.

Methodology: In this thesis, we have divided this work into three main chapters, each focusing on a distinct aspect of our research:

The first chapter provides an in-depth exploration of the global oil industry, delving into its history, key dynamics, trends, and challenges. Within this broader context, we have examined the Algerian oil industry and analyzed its unique characteristics, role, and significance within the global landscape.

The second chapter is dedicated to establishing a solid foundation in machine learning concepts and techniques. We have covered the essential principles, algorithms, and applications of machine learning, providing a comprehensive overview of this field. Additionally, we have conducted a thorough literature review, examining the existing research and insights relevant to the application of machine learning in the oil industry.

The 3rd and final chapter presents the practical case study, where in the perspective of forecasting of oil prices, we employed a variety of machine learning models to analyze both daily and monthly datasets. For both datasets, we trained and tested the same set of models including SVR, Random

Forest, XGBoost, LSTM, and CNN, after the data pre-processing, . Additionally, we experimented with hybrid models by combining the strengths of different models, such as Random Forest-XGBoost, and CNN-LSTM. This will allow us to compare the performance of individual models with that of hybrid models, thereby evaluating whether using a single model or combining multiple models would yield better results.

Also, we compared the performance of the models in each dataset. This will allow us to compare the performance of each model across different time scales, thereby identifying the conditions and environmental factors that contribute to better forecasting outcomes for each model.

Chapter 1

Oil industry

Section 1: Global oil industry

1. Introduction to the oil industry

1.1. Definitions

1.1.1. Petroleum:

Petroleum, derived from the Latin term meaning 'rock oil,' is a vital natural resource consisting primarily of hydrogen and carbon molecules known as hydrocarbons. This substance, commonly referred to as crude oil, is extracted from the earth's crust and serves as a key source of energy globally. In addition to its natural form, petroleum can also encompass synthetic hydrocarbons produced to mimic its properties, although these are not yet produced on a large scale. Refined petroleum products like motor gasoline and fuel oil are derived from processing crude oil and play a significant role in various industries. As one of the primary fossil fuels, petroleum has its origins in ancient marine organisms, algae, and bacteria, which over millions of years transformed into carbon-rich substances due to geological processes involving heat and pressure. This transformation has made petroleum a crucial resource for fuel and a wide range of commodities essential for modern society. [[MacFadyen and Watkins, 2014](#)], [[Turgeon and Morse, 2023](#)]

1.1.2. Oil industry:

The oil industry is an economic sector that is responsible for the exploration, extraction, refining, transportation, and sale of crude oil and its refined products, such as gasoline, diesel fuel, and lubricants. [[MacFadyen and Watkins, 2014](#)]

1.2. Historical Overview:

The global oil industry stands as a testament to human ingenuity. Back in the mid 19th century, in Pennsylvania, Edwin Drake's pioneering drilling and John D. Rockefeller's innovative business strategies has not only unlocked abundant crude oil reserves but also established business models that would shape the industry for decades. Visionaries like Henry Ford and Karl Benz further propelled the industry forward by catalyzing mass automobile adoption, driving demand for

gasoline and lubricant, which ushered in a new era of expansion and innovation, fostering the emergence of major oil companies and extensive refining and distribution networks.

As the 20th century unfolded, oil evolved from a commodity into a pivotal strategic asset, integral to the military and economic ambitions of nations worldwide. World War I and World War II underscored the critical importance of controlling oil resources in determining the outcomes of conflicts, shaping military strategies and alliances. Also, the Cold War era heightened these dynamics, with geopolitical tensions between the United States and the Soviet Union spurring proxy wars and covert operations in oil-rich regions like the Middle East

Throughout this period, oil remained at the forefront of global power dynamics, influencing diplomatic relations, intelligence activities and military interventions. Its profound influence extended across various sectors, fueling swift industrial growth, urban development and technological progress. By that time, major oil giants like Exxon, Shell and BP have emerged as key global actors, shaping the evolution of the industry from exploration and production to refining, distribution and marketing.

The Hydrocarbon Age ushered in significant geopolitical shifts, elevating oil-rich nations like the United States, Saudi Arabia, and Russia as major global players with substantial influence over energy markets and geopolitics. Nowadays, the enduring legacy of oil persists, underscoring its pivotal role in shaping the contemporary world, driving economic development, and influencing geopolitical dynamics. [Yergin, 2011]

1.3. Key players in the oil industry:

In the complex world of the oil industry, companies are strategically divided according to the different stages of the crude oil process they go through, from initial extraction to final delivery. This breakdown is generally divided into three key sectors: upstream, midstream and downstream. Each sector plays a vital role in the oil supply chain with distinct and complementary functions. The key sectors are also grouped into three main categories:

1.3.1. Key sectors:

The key sectors of the oil industry are crucial components that make up the complex processes from extraction to distribution. The industry is typically divided into three main segments: upstream, midstream, and downstream. [McClay, Rebecca, 2022]

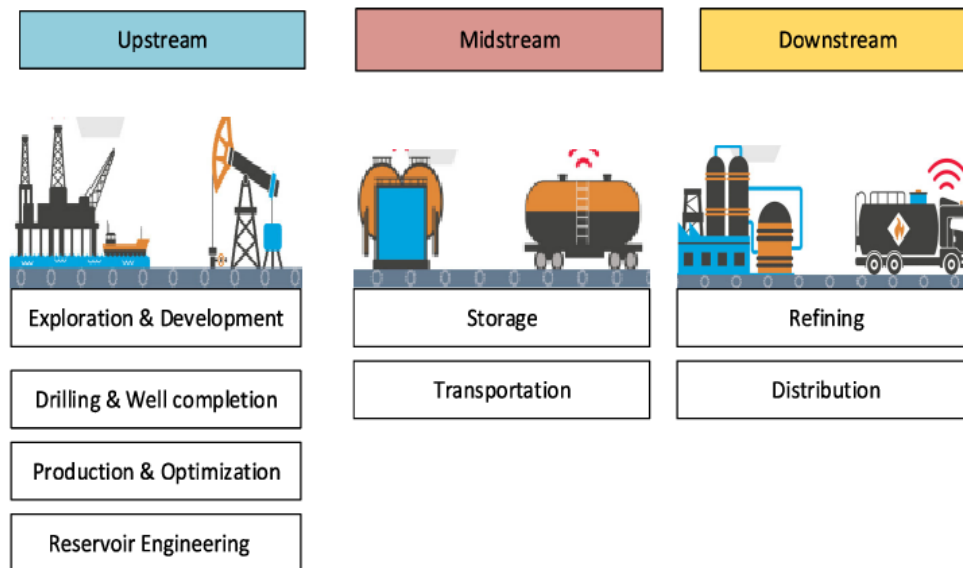


Figure 1.1: Oil Industry Sectors
 Source: ResearchGate by [Musa, 2023]

A. Upstream:

The upstream segment of the oil industry encompasses two main operations: exploration and production.

- **Exploration:**

Exploration involves conducting geological surveys to identify potential oil reserves, as well as acquiring land rights for drilling. Exploration for oil involves the process of identifying potential drilling sites, using a range of methods and technologies. Traditionally, early explorers relied on surface indicators like natural oil seeps, but advancements have led to more sophisticated approaches [MacFadyen and Watkins, 2014].

Geological surveys employ techniques such as subsoil testing for onshore sites and seismic imaging for offshore locations. Energy companies compete for access to mineral rights through agreements with governments, either conceding ownership of discovered resources or sharing production proceeds. Exploration is a costly and risky endeavor primarily funded by corporate investments. [Raymond and Leffler, 2017]

- **Production:**

Production activities include both onshore and offshore drilling to extract the oil from the ground. Once the oil reserves identified, companies initiate drilling plans. Often, they engage specialized drilling firms, covering expenses such as labor crew and rig day rates. Factors like drilling depths, rock hardness, weather conditions, and site distance influence the duration of drilling operations. [MacFadyen and Watkins, 2014]

Employing smart technologies for data tracking enhances drilling efficiency and well performance by offering real-time insights and trends. Despite sharing fundamental

components, drilling methods vary according to the specific type of oil or gas and the geological characteristics of the site. (Paris: Centre for Economic Management, Institut Français du Pétrole, c2004)

B. Midstream:

The midstream sector covers transportation, storage, and trading of crude oil, natural gas, and refined products.

- **Transportation:**

Before the refinement, the oil is mainly transported via tankers along water routes and pipelines across regions. [MacFadyen and Watkins, 2014] After extraction and separation from natural gas, pipelines carry the oil to a refinery or another transport carrier. Refined petroleum products reach the market through tankers, trucks, trains, or additional pipelines.

- **Storage:**

Storage of oil helps smooth out supply and demand discrepancies. Companies store more when the prices are lower than they would like and withdraw when prices are high. [MacFadyen and Watkins, 2014]

C. Downstream:

The downstream sector covers refining and marketing.

- **Refining:**

Refining is intricate, but its objective is clear: to convert crude oil, essentially unusable in its raw form, into diverse petroleum products vital for heating homes, powering vehicles, and manufacturing petrochemical plastics. [MacFadyen and Watkins, 2014]

- **Marketing and Distribution:**

Marketing is the wholesale and retail distribution of refined petroleum products cater to business, industry, government, and public consumers. Typically, crude oil and its derivatives are directed to markets offering the greatest value to suppliers, often prioritizing proximity for reduced transportation expenses and increased revenue. Yet, various factors such as refining setups, demand variations, and quality requirements can influence trade flows, deviating from this norm. [MacFadyen and Watkins, 2014]

1.3.2. Categories of oil companies:

A. National Oil Companies (NOCs):

National Oil Companies, also known as “*state-owned companies*”, are oil companies characterized by substantial ownership by their respective government entities. These companies are tasked with operating in a manner that aligns with the national interests of their country [Al-Fattah, 2014]. They are also referred to as “economic giants”, they control at least \$3 trillion in assets and produce most of the world’s oil and gas, and are expanding globally, capturing market share, and influencing the oil industry through their international activities (David Manley, David Mihalyi, Patrick R.P. Heller, “Hidden giants”, December 2019, <https://www.imf.org/>). They account for 58% of the global reserves and 50% of global production. [Aboelkheir, 2022]

B. International Oil Companies (IOCs):

International Oil Companies, also called “*Integrated Oil Companies*” [Al-Fattah, 2014], are the Oil & Gas companies which are privately owned and operate globally.

C. Independent Oil Companies:

In recent years, national oil companies have become not only global partners in joint ventures with major oil companies, but also increasingly rivals of international oil companies. [Olcott, 2022]

2. Market dynamics and pricing:

The dynamics of the oil market are influenced by various factors, including supply and demand dynamics, geopolitical tensions, economic growth, technological advancements, environmental regulations, and the actions of major oil-producing countries and organizations.

2.1. Factors influencing the oil prices:

The influencing factors that affect the fluctuation of crude oil prices are diverse and complex. Researchers have identified various factors that impact crude oil prices, including fundamental factors i.e. supply and demand [Yan et al., 2021b], economic indicators, geopolitical tensions, exchange rates and, factors relating to the structure of the crude oil market; OPEC policies. [Fernandois and Medel, 2020]

2.1.1. Supply and Demand:

According to the latest projections of the International Energy Agency (IEA), the global oil production in 2023 is expected to reach 101.7 million barrels per day (mb/d) [IEA, 2023], and 102.9 mb/d in 2024 [IEA, 2024]. Meanwhile, the worldwide oil consumption is forecasted to rise to 102.21 mb/d in 2023 [IEA, 2023], whereas it will increase to 104.46 mb/d in 2024 [IEA, 2024]. According to Gallo et al.’s research, it is shown that the main factor that leads to the change in crude oil price is supply, when the demand of crude oil is the result of the prices change [Gallo et al., 2010]. Before the 20th century, supply and demand were not considered as influencing factor on oil price [Breitenfellner et al., 2009].

2.1.2. Speculation:

In recent years, there has been significant debate surrounding the impact of speculation on oil prices. Some argue that the surge in commodity-linked investments by financial traders has led to oil prices surpassing levels justified by supply and demand dynamics. However, a notable limitation of this argument is the assumption that investors can accurately differentiate between price fluctuations caused by speculative trading and those influenced by changes in inventory demand. Various studies have explored the correlation between oil price fluctuations and open interest in the futures market, but the effectiveness of this approach has been questioned due to challenges in accurately categorizing traders [Awan, 2019]. Cologni et al. suggest that economic fundamentals, rather than speculation, were the primary drivers of the surge in oil prices during the 2003-2008 period, although speculation may have contributed to short-term price volatility [Cologni et al., 2015].

2.1.3. Exchange rates:

According to R. A. Lizardo and A. V. Mollick research on the relationship between the oil price fluctuation and U.S. dollar exchange rates, it is found that there is a significant negative correlation between the two [Lizardo and Mollick, 2010]. The relationship between crude oil prices and U.S. dollar exchange rates is found to be a non-linear relationship according to L. Kilian et al. [Kilian and Vigfusson, 2011]. On the other hand, Chen et al. found that this non-linear relationship lacks significance. However, they highlighted that during supply and demand-induced price shocks, the dollar exchange rate can have varying effects on crude oil prices, indicating that other factors beyond the exchange rate also impact oil prices [Chen et al., 2016]. Bouoiyour et al.'s research demonstrated a positive correlation between crude oil prices and the Russian exchange rate [Bouoiyour et al., 2014], while Blokhina et al. further supported the strong link between oil prices and the Rouble exchange rate. [Blokhina et al., 2016]

2.1.4. Geopolitical tensions/events:

Historically, there is no clear-cut relationship between oil prices and geopolitical events like emerging tensions between countries or terrorist attacks. After major events, oil prices initially surged but then dropped significantly shortly after. (https://www.ecb.europa.eu/press/economic-bulletin/focus/2024/html/ecb.ebbox202308_02_ed883ebf56.en.html) , until when Monge et al. argue that crude oil price behavior depends on any event that has the potential to disrupt the flow of oil [Monge et al., 2016a]. In fact, geopolitical tensions play a crucial role in shaping crude oil price volatility, impacting not only the current oil price but also forecasts and their consensus. Geopolitical tensions, including unexpected events and news related to major oil-producing countries, not only influence the current price of oil but also have a profound impact on future forecasts and the consensus surrounding them. The relationship between geopolitical tensions and oil prices is complex, as these tensions can trigger fluctuations in oil prices through two main channels, the Economic Activity Channel where higher geopolitical tensions can act as a negative global demand shock, leading to a contraction in global economic activity, which dampens global oil demand and prices. On the other hand, the Risk Channel which involves financial markets pricing in higher risks to future oil supply, putting upward pressure on oil prices. [Fernandois and Medel, 2020]

Recent research indicates that improved US-China political relationships and higher geopolitical risks can both drive up the price of oil. Positive shocks in political relationships are associated with

optimistic expectations regarding economic activity, while positive shocks in geopolitical risks reflect fears of supply disruption [Mignon and Saadaoui, 2023]. However, the correlation between geopolitical risks and oil prices is time- and frequency-varying, suggesting a nuanced and dynamic relationship between these factors. [Li et al., 2020]

2.2. Global oil market trends:

The perception of oil as a cheap and abundant resource has shifted due to various trends in the oil market. Since the 1970s, prices have been volatile and consistently rising, influenced by factors such as the establishment of OPEC and the emergence of national oil companies. These changes have led to boom-and-bust cycles, making long-term planning difficult for smaller firms. While OPEC dominated oil pricing in the 1970s-1990s, the rise of non-OPEC producers like the U.S. and the growth of oil demand in Asia have made the market more complex and less dominated by OPEC [Yoshino and Alekhina, 2019]. In the early/mid 2000s, the transformation of the oil market mechanism went through three main phases:

- Relatively Calm Market Period (January 07, 2000, to March 12, 2004) in which the oil market exhibited stability and relatively low volatility in prices.
- Bubble Accumulation Period (March 19, 2004, to June 06, 2008) which saw a buildup of speculative bubbles in the oil market, leading to significant price increases and heightened market activity.
- Global Economic Crisis Period (June 13, 2008, to September 11, 2009) which impacted the oil market, causing fluctuations and changes in oil prices due to economic instability and reduced demand. [Huang et al., 2011]

In fact, crude oil prices rose from around \$30 per barrel in 2000 to over \$130 per barrel by 2008 and fell sharply in the second half of 2008 to a low of below \$40 as the global financial crisis hit. In the late 2000s, from 2004 to 2009, the oil prices stabilized and generally remained in a broad trading range between \$70 and \$120 per barrel, this period saw a recovery in oil prices as the global economy started to rebound from the recession [Gränitz, 2020]. By 2018, the U.S. became the largest oil producer in the world, The United States went on to become the largest oil producer in the world by 2018, driven by a dramatic increase in shale oil production, which had a significant impact on global oil market dynamics. [Nakhle, 2022]

According to the OPEC, the global oil demand growth is expected to slow, with a projected increase of 1.1 million barrels per day (mb/d) in 2024 and 1.2 mb/d in 2025. This slowdown is attributed to macroeconomic headwinds, tighter efficiency standards, and the expansion of the electric vehicle fleet. In contrast, global oil supply growth is anticipated to rise, driven by non-OPEC+ output, which is expected to increase by 1.4 mb/d in 2024 and 1.8 mb/d in 2025. However, OPEC+ crude oil production is expected to decline by 840,000 barrels per day (kb/d) in 2024 due to voluntary cuts, before gradually recovering to a growth rate of 330 kb/d in 2025. Additionally, global oil inventories are expected to increase, with a significant build-up in April 2024, followed by further builds in subsequent months. These trends collectively suggest a more balanced market in 2024, with oil prices initially rising due to supply constraints but eventually declining as inventory levels increase. [OPEC, 2024]

World oil demand and supply balance	2021	2022	2023	1Q24	2Q24	3Q24	4Q24	2024	1Q25	2Q25	3Q25	4Q25	2025
World demand													
Americas	24.0	24.8	25.0	24.6	25.4	25.6	25.4	25.2	24.6	25.4	25.7	25.5	25.3
of which US	19.8	20.2	20.4	20.0	20.7	20.7	20.8	20.5	20.0	20.7	20.7	20.9	20.6
Europe	13.2	13.5	13.4	13.2	13.6	13.7	13.3	13.4	13.2	13.6	13.7	13.4	13.5
Asia Pacific	7.3	7.4	7.3	7.8	7.0	7.1	7.5	7.3	7.8	7.0	7.1	7.5	7.3
Total OECD	44.5	45.7	45.8	45.5	45.9	46.4	46.3	46.0	45.6	46.0	46.5	46.4	46.1
China	15.4	15.0	16.3	16.5	16.8	17.2	17.3	17.0	16.9	17.2	17.7	17.7	17.4
India	4.8	5.1	5.3	5.7	5.6	5.4	5.6	5.6	5.9	5.9	5.6	5.8	5.8
Other Asia	8.7	9.1	9.3	9.7	9.7	9.5	9.5	9.6	10.0	10.1	9.8	9.8	9.9
Latin America	6.2	6.4	6.7	6.8	6.9	7.0	6.9	6.9	7.0	7.1	7.2	7.1	7.1
Middle East	7.8	8.3	8.6	8.8	8.6	9.2	9.0	8.9	9.1	8.9	9.7	9.4	9.3
Africa	4.2	4.4	4.5	4.6	4.4	4.4	4.8	4.6	4.8	4.5	4.5	4.9	4.7
Russia	3.6	3.8	3.8	3.9	3.8	4.0	4.1	3.9	4.0	3.9	4.0	4.1	4.0
Other Eurasia	1.2	1.2	1.2	1.3	1.2	1.1	1.3	1.2	1.3	1.3	1.1	1.3	1.3
Other Europe	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Total Non-OECD	52.7	54.0	56.5	58.0	57.8	58.5	59.3	58.4	59.8	59.5	60.4	61.0	60.2
(a) Total world demand	97.2	99.7	102.2	103.6	103.7	104.9	105.6	104.5	105.4	105.5	107.0	107.4	106.3
Y-o-y change	5.9	2.5	2.6	2.4	2.0	2.5	2.1	2.2	1.8	1.7	2.1	1.8	1.8
Non-DoC liquids production													
Americas	23.5	24.9	26.6	26.9	27.0	27.4	27.9	27.3	27.9	27.7	27.9	28.3	27.9
of which US	18.1	19.3	20.9	20.9	21.2	21.4	21.8	21.3	21.7	21.8	21.8	22.0	21.8
Europe	3.8	3.6	3.7	3.7	3.8	3.7	3.9	3.8	3.9	3.8	3.8	3.9	3.9
Asia Pacific	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Total OECD	27.8	29.0	30.7	31.0	31.2	31.5	32.2	31.5	32.2	31.9	32.1	32.6	32.2
China	4.3	4.4	4.5	4.6	4.6	4.5	4.5	4.5	4.6	4.6	4.5	4.5	4.5
India	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Other Asia	1.7	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.5	1.6
Latin America	6.0	6.3	7.0	7.3	7.3	7.4	7.4	7.3	7.5	7.5	7.6	7.8	7.6
Middle East	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Africa	2.3	2.3	2.2	2.3	2.2	2.2	2.3	2.2	2.3	2.3	2.3	2.3	2.3
Other Eurasia	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Other Europe	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Total Non-OECD	17.5	17.9	18.6	19.0	19.0	18.9	19.0	19.0	19.2	19.2	19.2	19.4	19.3
Total Non-DoC production	45.4	46.9	49.3	50.0	50.2	50.4	51.2	50.4	51.4	51.1	51.4	52.0	51.5
Processing gains	2.3	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.6	2.6	2.6	2.6	2.6
Total Non-DoC liquids production	47.7	49.3	51.7	52.5	52.7	52.9	53.7	53.0	54.0	53.7	54.0	54.6	54.1
DoC NGLs	7.6	7.9	8.2	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.2	8.3	8.3
(b) Total Non-DoC liquids production and DoC NGLs	55.3	57.2	59.9	60.8	61.0	61.2	62.0	61.3	62.3	62.0	62.2	62.9	62.4
Y-o-y change	0.6	1.9	2.7	1.9	1.7	1.1	0.7	1.4	1.5	1.0	1.0	0.9	1.1
OPEC crude oil production (secondary sources)	25.2	27.7	27.0	26.5									
Non-OPEC DoC crude production	15.0	15.1	15.0	14.7									
DoC crude oil production	40.3	42.8	42.0	41.2									
Total liquids production	95.5	100.1	101.9	102.0									
Balance (stock change and miscellaneous)	-1.6	0.4	-0.3	-1.5									
OECD closing stock levels, mb													
Commercial	2,652	2,781	2,777	2,793									
SPR	1,484	1,214	1,207	1,212									
Total	4,136	3,995	3,983	4,005									
Oil-on-water	1,348	1,546	1,438	1,538									
Days of forward consumption in OECD, days													
Commercial onland stocks	58	61	60	61									
SPR	32	27	26	26									
Total	91	87	87	87									
Memo items													
(a) - (b)	41.9	42.4	42.3	42.7	42.8	43.7	43.6	43.2	43.1	43.4	44.8	44.5	44.0

Figure 1.2: Global Oil Market Trends
Source: [OPEC, 2024]

2.2.1. Price volatility and market speculation:

There have been two prevalent theories to explain oil price instability, a testament to earlier observed periods of oil price volatility during the American Civil War and both World Wars. The first theory stems from the economic theory of the competitive market price. Oil prices were said to have fluctuated in response to disequilibrium periods where OPEC was either earning far more or far less than its optimal expected revenue. Measures such as the percentage depletion allowance on discovering oil and natural gas wells, as well as the capital investment tax write-off in the Mining, Metallurgical, and Petroleum Engineering Industry, have encouraged exploration and production at the expense of the time involved in the search for higher resource quality. This, in turn, has created periodic excessive oil production in the United States. Simulation models using Markov Perfect Equilibrium strategies have shown how external supply shocks can significantly disturb the equilibrium and cause large price swings in the oil market. [Bornstein et al., 2023]

Can be attributed to an increase in global demand for oil indicates that the gap between oil supply and demand has narrowed. Stating that between 1950 and 1972, the average annual growth

rate in demand for OPEC oil exports has been close to 10%. This rate has been reduced to less than 1% in the 1990s, with future estimates indicating a gradual decline in demand from 28 million barrels a day at present to 24.8 million barrels a day in 2020. This decrease in demand growth has been met with a more erratic movement in the supply of oil caused predominantly by frequent changes in OPEC production levels and product strategy, as well as a greater degree of price flexibility in the long run from non-OPEC producers. These divergent movements in the supply and demand for oil have caused oil price instability. High and erratic changes in oil prices are said to be more damaging than the effect of low oil prices. Recent studies comparing the impact of oil price increases during the Gulf crisis in 1990 to the Iran-Iraq war in the 1980s consistently show that oil price instability has more detrimental effects on the global economy than the actual price level change. [Bakirtas and Akpolat, 2020]

Section 02: Algerian oil industry

1. Introduction to the Algerian oil industry:

1.1. Historical background and evolution:

Algeria was one of the first countries in Africa to discover oil on a large scale in the early 20th century. This discovery marked the beginning of oil and gas exploitation in the country, often under foreign control [[Chanderli and Zaimche, 2024](#)]. In fact, In 1925, France created the Office national des combustibles liquides (ONCL) to prospect and develop the oil industry in the country and its empire. Research resumed in 1945, and in 1946, it was transferred to Algeria to the Société nationale de recherche et d'exploitation des pétroles en Algérie. As early as the 1920s, studies had suggested the presence of oil reserves in the north of the Hoggar and the central Sahara. The research extended over approximately 600,000 km² by 1950 when Algeria became a significant oil producer with the majority of its fields located in the Sahara Desert. [[Kamen-Kaye, 1958](#)]

In January 1956, the discovery of oil at Edjelé and the commissioning of nine drilling wells at Hassi-Messaoud marked an important turning point. A city developed in this former caravan stop, entirely supplied by planes and trucks. This discovery would have a significant impact on the evolution of the Algerian conflict and influence the negotiations leading to the Évian Accords in 1962. [[Malti, 2012](#)]

After the independence in 1962, Algeria embarked on a process of nationalizing its oil industry starting from 1967 after the foundation of Sonatrach on December 31st, 1963, aiming to regain control over the country's natural resources. This process was marked by complex negotiations with foreign oil companies [[Mounecif, 2018](#)] with the Algerian government placing all American-run oil companies under temporary state control which lead to the nationalization of five American oil companies on August 30, 1967 [[CIA, 1970](#)]. This was followed by the nationalization of French oil and gas interests in Algeria was completed in 1971, with the establishment of ElfAlgerie, a new exploratory oil company. [[Times,](#)]

The post-nationalization period has been marked by economic and political challenges, including the management of oil revenues, economic diversification, and the pursuit of a balance between industrial development and environmental preservation. During the late 1970's, the country encountered significant economic crisis due to its heavy reliance on oil and gas exports. [[Mounecif, 2018](#)]

Over the decades, Algeria has implemented various reforms to modernize its oil sector, attract foreign investment, and stimulate economic growth, while seeking to preserve its sovereignty over its resources. [[Laframboise et al., 1998b](#)] [[Laframboise et al., 1998a](#)] Today, Algeria is facing new challenges, such as the fluctuation of oil prices on the global market, the need to diversify its economy, and the sustainable management of its natural resources to meet the needs of future generations. [[OilPrice.com,](#)]

1.2. Overview of oil reserves and production levels:

Algeria's first commercial oil discovery was Edjelleh in 1956, followed immediately by the Hassi Messaoud oil field the same year. Production began in 1958. [[OPEC, 2024](#)] The country holds substantial oil and gas reserves, ranking 16th globally with approximately 12,2 billion barrels of proven oil reserves as of 2016. This accounts for around 0.7% of the world's total oil reserves of

1.65058514 trillion barrels. Algeria's proven oil reserves are equivalent to 77.9 times its annual consumption, indicating that without net exports, there would be approximately 78 years of oil left at current consumption levels, excluding unproven reserves. This significant reserves-to-production ratio underscores Algeria's importance in the global energy landscape and highlights the country's potential to contribute to its economic development and energy security [alg,]. As of 2023, Algeria held an estimated 12.2 billion barrels of proved crude oil reserves. [U.S. Commercial Service, 2024]

Algeria's national oil company, Sonatrach, estimates that approximately two-thirds of the country's territory remains underdeveloped or unexplored, with an estimated 100 undeveloped discoveries waiting to be tapped. [eia,]

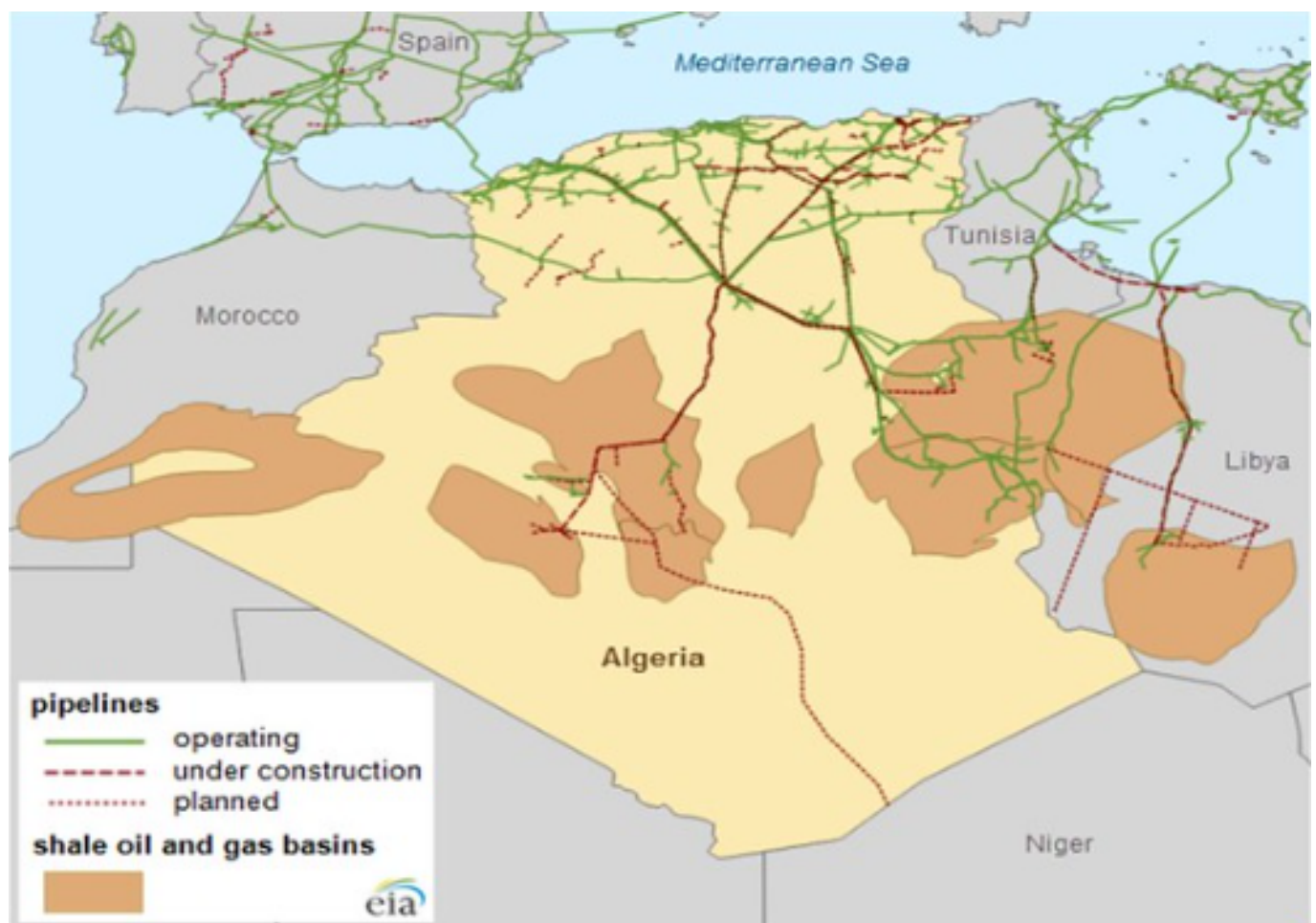


Figure 1.3: Oil and natural gas basins and pipeline infrastructure in Algeria
Source: [IEA, 2016]

2. Market dynamics and the economic impact:

2.1. Oil market trends and pricing dynamics:

In the decades following independence, Algeria focused on developing its oil and gas industries through state-led industrialization and investment [Hamzaoui, 2020a]. This led to rapid economic growth, with oil exports accounting for around 95% of total exports and two-thirds of government revenues by the 2000s. [OPEC, 2024]

2.2. Impact of oil revenues on the Algerian economy:

Algeria's economy is heavily dependent on its oil and natural gas exports, which account for around two-thirds of government revenues and 95% of total exports. The discovery of large oil and gas reserves in the 1950s and 1960s has been a major driver of Algeria's economic development, but has also presented significant challenges. [[of Commerce,](#)]

The volatility of oil prices has had a significant macroeconomic, financial, and policy impact on Algeria. Sharply lower oil prices can weaken the country's fiscal and external positions and reduce economic activity. This has led the government to assume flat oil and gas prices in its economic planning and provide sensitivity analysis based on higher and lower price scenarios. [[OPEC, 2023](#)]

During "oil boom" periods, such as the 1970s and early 2000s, the government increased public spending on social and infrastructure projects, but these investments often lacked strategic planning. Instead of focusing on diversifying the economy or boosting productivity, spending was directed towards unproductive areas, leading to inefficiencies in fiscal policy. This misallocation of resources contributed to Algeria's weak economic growth over time. [[Haouas et al., 2021](#)]

The reliance on oil revenues has also hindered economic diversification efforts. Despite gradual liberalization since the 1990s, Algeria's economy remains dominated by the state, and the government's efforts to attract foreign and domestic investment outside the energy sector have had limited success in reducing high poverty and youth unemployment rates [[OPEC, 2024](#)]. In 2020, Algeria's GDP per capita at purchasing power parity dropped from about US\$11,900 in 2016 to US\$10,800, and the economy contracted by 5.1% as COVID-19 deeply impacted the oil-reliant economy [[Chipanda, 2023](#)]. In fact, the heavy reliance on the oil sector and low-productivity sectors has made the economy more vulnerable to external shocks. [[Haouas et al., 2021](#)]

The government has attempted various reforms over the years, but these have often been dependent on the availability of oil revenues. These revenues have enabled the government to fund infrastructure projects, social services, and subsidies that benefit large segments of the population. Major public investments in housing, education, and healthcare have been supported by oil income. For example, social programs such as subsidies for basic goods like food and energy are financed by oil wealth, helping to alleviate poverty and provide stability [[Hamzaoui, 2020b](#)].

However, Algeria's over-reliance on oil has also exposed it to significant economic volatility, primarily due to fluctuations in global oil prices. When prices drop, as seen in recent years, the economy struggles, resulting in budget deficits, reduced foreign reserves, and economic contraction. This volatility makes long-term planning difficult, and despite efforts, Algeria has struggled to diversify its economy. The oil sector's dominance has hindered the development of other industries, such as agriculture and manufacturing, contributing to high unemployment, particularly among youth [[Hamzaoui, 2020b](#)].

The decline in oil prices has presented an opportunity for Algeria to reform fuel subsidies and energy taxes, as well as reinvest efforts to diversify its oil-reliant economy. But, the country faces significant challenges in diversifying its economy due to its heavy reliance on oil and gas exports. It has modest oil reserves compared to other OPEC members, estimated at 16-17 billion barrels, and may become an oil importing country by 2030 [[Abdellaoui, 2022](#)]. Algeria also has the world's third-largest untapped shale gas resources, but developing these resources will require significant investment and technological expertise. [[of Commerce, 2023](#)]

Chapter 2

Machine Learning in the oil industry

Section 1: Theoretical basis of Machine Learning

1. Introduction

1.1. Definitions

1.1.1. Data Science:

As mentioned in L. Cao's article, the first term for Data Science was "Datalogy", presented in P. Naur's book "The science of data and of data process" in 1968 [Naur, 1968]. The first appearance of the term "Data Science", was in 1974, when Naur has defined it as "the science of dealing with data, once they have been established, while the relation of the data to what they represent is delegated to other fields and sciences." in his other book "Concise survey of computer methods". [Naur, 1974]. [Cao, 2017]

According to the abstract description of Matthew Waller and Stanley Fawcett of Data Science: "Generally, Data Science is the application of quantitative and qualitative methods to solve relevant problems and predict outcomes.". [Waller and Fawcett, 2013]

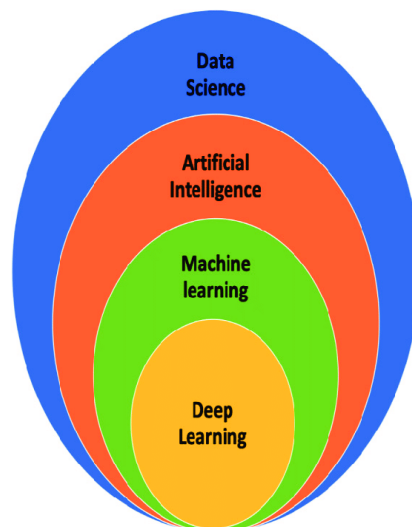


Figure 2.1: Relationship between data science, AI, and machine learning
Source: [Otokiti, 2020]

1.1.2. Data Mining:

The first use of the term “Data Mining” traces back to the late 1980’s within research community [Coenen, 2011]. According to Frawley’s definition of the Knowledge Discovery in Databases (KDD) “The non-trivial extraction of implicit, previously unknown, and potentially useful information from data” [Frawley et al., 1992], Data Mining can be defined as a step in the KDD process concerned with applying computational techniques [Džeroski, 2008], and so is a set of mechanisms and techniques, realized in software, to extract hidden information from data. [Coenen, 2011]

1.1.3. Machine Learning:

In 1959, Arthur Samuel described Machine Learning as the “field of study that gives computers the ability to learn without being explicitly programmed” [Samuel, 1959]. He concluded that programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.

According to Tom M. Mitchell’s definition : “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .” [Mitchell, 1997], [Awad and Khanna, 2015]. In other words, Machine learning is a set of mathematical techniques and computational algorithms that allow for the automated construction of a prediction function, also known as " f ", based on a collection of data points called the "training data". [Lemberger et al., 2015]

1.1.4. Deep Learning:

Rina Dechter was the first to ever use the term “Deep Learning” in the Machine Learning community in 1986 [Dechter, 1986], when Igor Aizenberd et al. introduced it in 2000 to ANN, specifically in the concept of Boolean threshold neurons [Aizenberg et al., 2000]. Oxford defines it as “a type of machine learning based on ANN in which multiple layers of processing are used to extract progressively higher-level features from data.” (<https://languages.oup.com/>)

1.2. Emergence and evolution of Machine Learning :

Machine learning, an integral part of artificial intelligence research, has evolved through various stages of development. Initially, in the 1950s and early 1970s, the focus was on logical reasoning, exemplified by programs like the Logic Theorist and the General Problem-Solving program. However, it became evident that logical reasoning alone was insufficient for achieving artificial intelligence [Zhou, 2021d]. This realization marked the transition to the "knowledge is power" era in the mid-1970s, where the emphasis shifted to the development of expert systems that found practical applications across different domains. [Carbonell et al., 1983]

E. A. Feigenbaum, credited as the pioneer of knowledge engineering, made significant contributions during this phase by creating the first expert system, DENDRAL, in 1965. Despite

these advancements, researchers encountered the "knowledge acquisition bottleneck," hindering the extraction and summarization of knowledge for machine learning. [Zhou, 2021d]

To address this challenge, researchers explored the concept of machine learning, as proposed by A. Turing in the 1950s. Early studies in the 1950s, such as A. Samuel's computer checkers program, laid the foundation for various machine learning approaches that emerged between the 1960s and 1970s. These included neural-network-based connectionism learning, logic-representation-based symbolism learning, decision-theory-based learning, and reinforcement learning, supported by statistical learning theory. [Carbonell et al., 1983]

By the 80s, machine learning had established itself as a distinct research field with diverse directions [Carbonell et al., 1983]. The First International Workshop on Machine Learning in 1980 marked a pivotal moment, leading to active research and publications like the book "Machine Learning: An Artificial Intelligence Approach." Machine learning methods were categorized into rote learning, learning from instruction, observation, discovery, analogy, and induction, with learning from examples gaining prominence through supervised and unsupervised learning. [Zhou, 2021d]

In the 80s, symbolism learning dominated learning from examples with decision trees and logic-based approaches. However, the complexity of hypothesis space led to a decline in its popularity by the late 1990s [Zhou, 2021d]. The mid-1990s saw the rise of neural-network-based connectionism learning, while statistical learning, exemplified by Support Vector Machine (SVM), emerged as a mainstream technique. [Carbonell et al., 1983]

In the early twenty-first century, deep learning, characterized by neural networks with multiple layers, demonstrated exceptional performance in complex data tasks like audio and image processing. This marked a resurgence of connectionism learning under the umbrella of deep learning, showcasing the continuous evolution and innovation within the field of machine learning. [Carbonell et al., 1983]

2. Types of Machine Learning algorithms:

Machine learning is a vast and varied field with widespread applications. The classification of machine learning methods can be approached from multiple perspectives. Generally, machine learning is divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning:

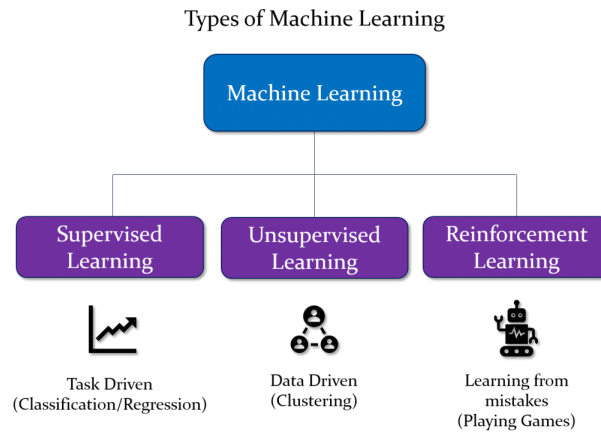


Figure 2.2: Machine Learning types
Source: [Dojo, 2020]

2.1. Supervised algorithms:

In the realm of machine learning, supervised learning is a crucial component where models are trained on labeled data to make predictions or classifications [Takahashi and Takahashi, 2024a]. Learners use labeled data, which is structured into a dataset despite the potential noise and missing details, with correct answers for each input to train models [Miikkulainen, 2024a]. This process involves an algorithm learning to map input data to corresponding output based on a set of training examples.

Supervised learning encompasses a range of models and algorithms, including linear regression for regression tasks, decision trees, random forests, and support vector machines for classification, as well as neural networks for more complex tasks [Takahashi and Takahashi, 2024a]. The main goal is to build a model that accurately predicts outcomes for new, unseen data. Success hinges on creating a model that generalizes effectively by grasping the underlying patterns and relationships within the training data. [Miikkulainen, 2024a]

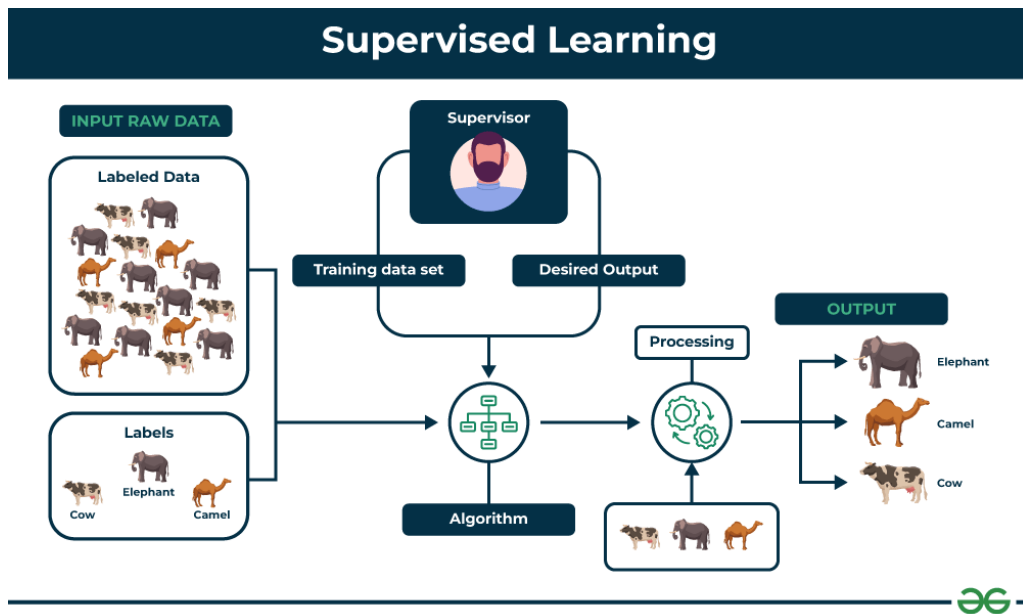


Figure 2.3: Supervised Machine learning
Source: [GeeksforGeeks, 2023]

2.1.1. Classification algorithms:

Machine learning classification is a supervised technique that assigns an instance or observation to a particular category or class based on a set of features or attributes [Nguyen et al., 2024]. The output variable has discrete values, turning prediction into a classification challenge, whereas input variable can be discrete or continuous. Supervised learning entails building a classifier from data for predicting outputs, known as classification. This classifier is used to predict outputs for new inputs, termed classification, with outcomes as classes.

In classification tasks, there are two key stages: learning and classification. During the learning phase, a robust learning technique is employed to train a classifier using a provided training dataset. Subsequently, in the classification phase, this trained classifier is utilized to categorize new input instances. [Li, 2024c]

- **K-Nearest Neighbour:** K-NN, also known as a lazy classification algorithm for not building any model [Biswas et al., 2018], is a non-parametric method that makes no assumptions about the underlying data, mostly used for classification but can also be used for regression analysis [Sarang, 2023j] created by Cover and Hart in 1968 [Cover and Hart, 1967]. It operates based on the principle that similar instances tend to be located in close proximity within the feature space. [Li, 2024e]

The algorithm takes an instance's feature vector as input, which corresponds to a point in the feature space. The output is the instance's class, which can be a single class or multiple classes [Li, 2024e]. In general, the performance of this algorithm relies on three key elements:

- ⇒ The choice of k , which determines the number of nearest neighbors to consider
- ⇒ The distance metric used to measure the similarity between instances
- ⇒ The decision rule used to determine the predicted class based on the classes of the k nearest neighbors [Biswas et al., 2018]

⇒ K-NN does not have an explicit learning process. Instead, it uses the training dataset as a reference to partition the feature vector space. This partitioning serves as a "model" for the classification task. [Li, 2024e]

- **Naive Bayes:** The Naïve Bayes technique is a classification approach that relies on the Bayes theorem and assumes conditional independence among features [Mitchell, 2005].

This algorithm assumes features are independent, which may not always be true, yet it often outperforms complex methods [Sarang, 2023k]. It's a probabilistic classifier that considers each feature's independent contribution to the target probability, disregarding correlations. This naive assumption allows maximum likelihood prediction, making it suitable for supervised learning. Bayesian inference, a statistical method updating hypothesis probabilities with new information, forms the basis for Bayesian methods, which excel in supervised learning scenarios. [Li, 2024g]

- **Decision Tree:** The decision tree is a highly adaptable machine learning algorithm that can effectively tackle both classification and regression problems. Its flexibility extends to handling both linear and non-linear relationships within datasets [Takahashi and Takahashi, 2024a]. In fact, they mimic the human decision-making process, which makes them relatively intuitive and accessible to understand. [Sarang, 2023f]

At its core, this algorithm operates by creating a hierarchical structure corresponding to a tree [Takahashi and Takahashi, 2024a] that represents the process of classifying instances according to their attributes within the classification task [Li, 2024a], where it learns to make informed decisions by meticulously examining the available data. [Takahashi and Takahashi, 2024a]

A standard decision tree structure includes a single root node, several internal nodes, and numerous leaf nodes. The leaf nodes represent the final decision outcomes, while all other nodes correspond to specific feature tests. Samples within each node are segregated into child nodes based on the feature splitting outcomes. Every route from the root node to a leaf node forms a decision sequence. [Zhou, 2021b]

- **Logistic Regression Model:** It is a classical classification method in machine learning mostly used for log-linear models [Li, 2024f] to make discrete predictions. [Sarang, 2023l]

The logistic regression type depends on the dependent variable used. If the output is a binary value, it is called a binary logistic regression. When the dependent variable is ordinal, it is an ordinal logistic regression. In cases where the output is multi-class variable, it is called a multi-nominal logistic regression. [Sarang, 2023l]

- **Support Vector Machine:** The SVM is a two-class classification model [Li, 2024h] (Li, H. 2024) that aims to build a hyperplane denoted by a weight vector [Takahashi and Takahashi, 2024a], where each element of the vector corresponds to an original feature, making the weight vector's dimensionality matches the number of features in the dataset (Huang, J., Hu, X., Yang, F. 2011). Once the hyperplane is determined, the placement of unseen data points into designated regions would be easier. [Sarang, 2023m]
- **Neural Networks:** In the first issue of Neural Network journal in 1988, Kohonen defined the ANN as: "Massively parallel interconnected networks of simple (usually adaptive) elements

and their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do”. [Kohonen, 1988]

ANN is a powerful and adaptable machine learning model designed to mimic the complex operations of the human brain. It has an input layer with nodes representing features, and an output layer with nodes indicating class probabilities. Hidden layers connect the input and output. The nodes are interconnected with weighted connections. [Sarang, 2023b]

Training an ANN involves optimizing the connection weights and biases to maximize classification accuracy, which depends on both the network architecture and the learned weights. [Nguyen et al., 2024]

2.1.2. Regression algorithms:

Regression analysis is employed to forecast the connection between the independent variable (input variable) and dependent variable (output variable), focusing on how the output variable’s value changes in response to variations in the input variable [Li, 2024c]. In machine learning, the input variables are commonly referred to as features or attributes. [Chen et al., 2024]

The regression problem is also divided into two processes: learning and prediction. [Li, 2024c]

- **Ridge regression:** Mainly used when multicollinearity is observed in the dataset, Ridge regression is a specific implementation of the general Tikhonov regularization technique [Sarang, 2023]. The algorithm incorporates regularization penalty, which is the sum of the squares of the coefficients multiplied by a constant, into the loss function [Takahashi and Takahashi, 2024a], so that it is equally applied to all coefficients in the model, teaching it how to reduce the magnitude of all coefficients, pulling them closer to zero by the same amount [Sarang, 2023] and allowing an improved regulation of the model’s behavior. [Takahashi and Takahashi, 2024a]
- **Lasso regression:** Like Ridge regression, Lasso also imposes a penalty on the magnitude of the regression coefficients. However, lasso differs in that it penalizes the absolute values of the coefficients rather than their squares.

By applying a penalty based on the absolute values, this type of regression can effectively reduce coefficients towards zero, potentially leading to the exclusion of certain variables. This characteristic aids in feature selection by promoting sparsity in the model.

- **XGBoost:** eXtreme Gradient Boosting, is a powerful machine learning framework designed for efficient and scalable implementation of gradient boosting algorithms. Developed by Tianqi Chen and Carlos Guestrin [Chen and Guestrin, 2016], XGBoost is an optimized implementation of gradient boosting that is designed to be highly efficient, flexible, and portable. It utilizes decision trees as base learners and is known for its speed and performance in predictive modeling tasks [Belyadi and Haghighat, 2021], also its ability to manage large datasets, handle missing values, and model non-linear relationships effectively. The Key features of the algorithm include regularization techniques to prevent overfitting, built-in cross-validation, and the ability to handle non-linear data patterns effectively. [Harrison, 2023]

To enhance model training and optimize performance, several advanced techniques can be employed, including hyperparameter tuning, early stopping, and ensemble methods.

A. Hyperparameter tuning: Hyperparameter tuning is crucial for improving the performance of XGBoost models. It involves adjusting parameters that govern the training process. Common techniques include:

- ⇒ **Grid Search:** This method systematically works through multiple combinations of parameter values, cross-validating as it goes to determine which combination produces the best model performance. For instance, parameters like *n_estimators*, *max_depth*, *learning_rate*, *subsample*, and *colsample_bytree* can be tuned using *GridSearchCV* from *scikit-learn*
- ⇒ **Random Search:** Unlike grid search, random search samples a fixed number of parameter settings from specified distributions. This can often yield good results with less computational cost compared to grid search.
- ⇒ **Bayesian Optimization:** This method builds a probabilistic model of the function mapping hyperparameters to a target objective and uses it to select the most promising hyperparameters to evaluate next. [XGBoost, 2024]

B. Early stopping: Early stopping is a crucial technique in machine learning, particularly when training complex models like those built with XGBoost. It helps prevent overfitting by halting the training process if the model's performance on a validation set does not improve after a specified number of iterations. [Baum and Gibbons, 2022]

C. Ensemble methods: Ensemble methods like bagging, boosting, and stacking play a crucial role in enhancing machine learning model performance. XGBoost exemplifies the power of boosting techniques, providing significant improvements in accuracy and efficiency across various applications. These methods not only improve predictive capabilities but also contribute to the robustness of models in real-world scenarios. [GitHub, 2023]

2.2. Unsupervised algorithms:

In the realm of machine learning, unsupervised learning plays a vital role in discovering patterns and relationships within unlabeled data [Takahashi and Takahashi, 2024b]. The key challenges in this approach involve clustering, dimensionality reduction, and association rule mining.

This type of learning aims to identify regularities in the data and to detect outliers and patterns [Santana, 2024a] of data without any labeled information [Li, 2024d]. It is particularly useful in data analysis and can also serve as a preprocessing step for supervised learning models, enhancing their performance by providing a more refined and organized dataset. [Takahashi and Takahashi, 2024b]

In fact, it enables the identification of variables that can be considered descriptors or possess independent significance, contributing to a deeper understanding of the data.

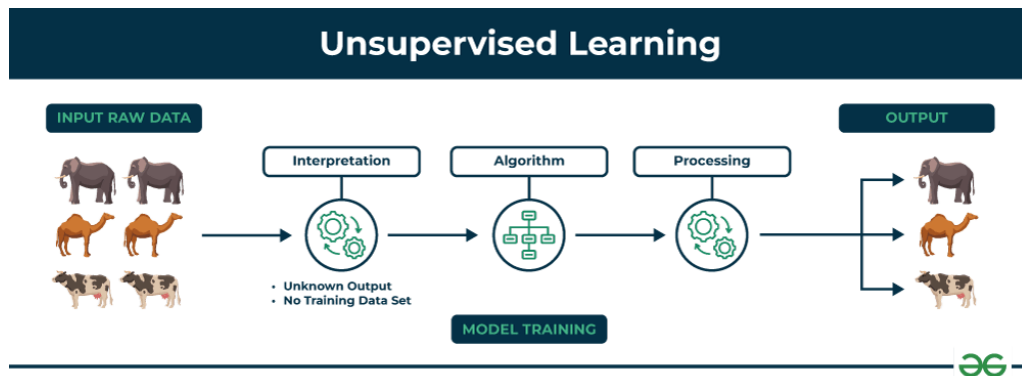


Figure 2.4: Unsupervised Machine Learning

2.2.1. Clustering Algorithms:

Clustering, the most important unsupervised learning [Sarang, 2023a], involves dividing a dataset into separate groups known as clusters, with each cluster representing a distinct concept [Zhou, 2021a]. This process aims to create partitions that align data points with relevant concepts [Handl et al., 2024]. The clustering process involves the following key stages:

- ⇒ **Feature Identification and Selection:** The identification the most informative features that best capture the essence of the data and select the most relevant ones for clustering.
- ⇒ **Algorithm Design or Selection:** The choice of an appropriate clustering algorithm, either by designing a new one or selecting a well-established method, based on the nature of the data and the clustering goals.
- ⇒ **Result Evaluation:** Assessment of the clustering results quality and validity to ensure they align with the expected outcomes and meet the customer's requirements.
- ⇒ **Result Interpretation and Presentation:** The explanation the clustering results to the customer, providing insights into the discovered patterns and their implications. If the customer is satisfied with the explanation, the clustering task is considered complete. [Sarang, 2023a]

A. Density-based:

It is a type of clustering algorithm that defines the clustering structure by the densities of the sample distributions [Sarang, 2023g]. These algorithms assess the connection between samples using a density-based approach and expand the clusters by incorporating samples that can be connected. [Zhou, 2021a]

- **Density-Based Spatial Clustering of Applications with Noise:** The DBSCAN algorithm defines neighbors as any point in the dataset with a distance less than or equal to Epsilon from a given point. Points are classified into three categories: core points, border points, and noise points. Core points have at least MinPts neighbors within a radius of Epsilon, border points have fewer than MinPts neighbors but are within a radius of Epsilon from a core point, and noise points are neither core nor border points (www.xlstat.com).

In this method, objects with a high locality density are grouped into a single cluster. Points with large number of nearby neighbors form a single group. Points whose nearest neighbors are too far away are treated as outliers. This algorithm is robust to outliers and does not require specifying the number of clusters in advance, making it capable of detecting nonlinearly separable clusters in such datasets. [Zhou, 2021a]

- **Ordering Points To Identify the Clustering Structure:** OPTICS is a density-based clustering algorithm that addresses a key limitation of DBSCAN by effectively handling datasets with varying densities.

Unlike DBSCAN, which struggles to form meaningful clusters in such scenarios, OPTICS computes an augmented cluster-ordering of database objects that is not constrained to a single global parameter setting [Sarang, 2023g]. This augmented cluster-ordering encapsulates information equivalent to density-based clustering across a wide range of parameter settings, making it a versatile foundation for both automatic and interactive cluster analysis. [Ankerst et al., 1999]

This clustering algorithm allows for standalone usage to gain insights into data distribution, with graphical representation for small datasets and appropriate visualization techniques for larger datasets. Additionally, it presents an efficient algorithm for automatically extracting not only traditional clustering information but also the intrinsic hierarchical clustering structure

B. Centroid-based:

Centroid-based clustering is a type of clustering where the algorithm separates data points based on multiple centroids in the data. Each data point is assigned to a cluster based on its squared distance from the centroid. [Sarang, 2023c]

- **K-means:** The k-means algorithm is a method used for clustering data points based on the center of gravity of the cluster. It is a simple algorithm that originated from signal processing for vector quantization.

The k-means algorithm is widely used in pattern recognition, document classification, and image processing, among others. The algorithm begins by selecting some objects as pivot objects. The number of clusters (k) is determined beforehand. The centroid of each pivot object is then calculated. Each data point is assigned to the cluster with the closest centroid.

The arithmetic mean is calculated separately for each dimension of the data points [Hartigan and Wong, 1979]. The k-means algorithm can be divided into two phases. In the first phase, k centroids are identified based on the chosen value of k. The distance between each data point and the centroids is calculated using the Euclidean distance metric. In the second phase, new centroids are determined based on the dissimilarity measures [Sarang, 2023c]. The process continues until the convergence property is met. [Kaufmann and Rousseeuw, 1987a]

- **K-medoids:** K-medoids clustering algorithm is a modified version of the K-means algorithm that is more resistant to noisy data and outliers. In K-medoids, the center of a cluster is represented by an actual data point within the cluster, rather than the mean point used in

K-means. This data point, known as the medoid, is the most centrally located object in the cluster, with the minimum total distance to all other points in the cluster.

By using a real data point instead of the mean, K-medoids is better able to handle the presence of outliers and noisy data, making it a more robust clustering algorithm in certain applications [Jin and Han, 2010], unlike the K-means clustering which is considered sensitive to outliers. [Sarang, 2023c]

C. Distribution-based:

Also known as the Probabilistic Clustering,

- **Mixture-of-Gaussian Clustering:** The algorithm plays with these two parameters to cluster the points. Depending on the values of these parameters, the cluster’s shape would differ [Sarang, 2023i]
 - **Expectation-Maximization:** This algorithm was originally proposed by Dempster, Laird, and Rubin in 1977 [Dempster et al., 1977]. It is a widely used method for finding the local optima of the parameters in probabilistic graphical models with latent variables. [Li, 2024b]
- The key purpose of the EM algorithm is to iteratively maximize the lower bound of the maximum likelihood estimation (MLE) of the model parameters. The algorithm achieves this by alternating between two steps:
- ⇒ **Expectation (E) Step:** In this step, the algorithm computes the expected value of the log-likelihood function, given the current estimates of the model parameters.
- ⇒ **Maximization (M) Step:** In this step, the algorithm updates the model parameters to maximize the expected log-likelihood computed in the E-step. [Jiang et al., 2023a]

D. Prototype-based:

A “prototype” refers to a representative data point in the sample space. Prototype clustering refers to a group of clustering algorithms that represent the clustering structure through a set of prototypes. These algorithms usually begin with some initial prototypes and then proceed to update and optimize them iteratively. Multiple algorithms have been created using various prototype representations and optimization techniques.

- **Self-Organized Maps:** The SOM, developed by Tuevo Kohonen in 1982, is a type of unsupervised learning algorithm that learns to represent input data in a way that preserves the essential characteristics of the original data [Kohonen, 1982]. The mapping process is nonlinear, ordered and smooth, ensuring that similar input data points are mapped to nearby locations on the output grid.

The SOM is a powerful tool for visualizing and analyzing high-dimensional data. They offer a unique approach to compressing information while preserving the essential topological and metric relationships of the input data.

By mapping high-dimensional data onto low-dimensional grid, typically a two-dimensional array, SOMs create a similarity graph that reveals the underlying structure and patterns within the data. [Kohonen, 2001]

- **Learning Vector Quantization:** Unlike the unsupervised clustering methods of Vector Quantization (VQ) and the basic self-organizing map (SOM), LVQ incorporates supervised learning by adjusting the position of the winning prototype based on whether it correctly classifies the data point [Kohonen, 1995]. This means that LVQ requires labeled data samples, as the clustering process is guided by supervised information. [Kohonen, 2001]

E. Connectivity-based:

Despite the advantages that K-means presents, like any other algorithm, it is limited to problems like where the clusters are convex or that we have to specify the value of K since the start [Sarang, 2023e], which requires the use of the connectivity-based clustering, also known as “hierarchical clustering”, algorithms for non-convex clusters [Kubat, 2021]. It is a technique used to create a tree-like structure of clusters from a dataset.

This clustering method aims to organize the data into a hierarchy, where smaller clusters are nested within larger ones. The hierarchical structure is formed by either a bottom-up or a top-down approach. [Zhou, 2021a]

- **Agglomerative Hierarchical Clustering:** Also referred to as a bottom-up or left-to-right clustering method where clusters have sub-clusters, which in turn have sub-clusters, etc. [Sasirekha and Baby, 2013], is a specific method within cluster analysis.

This technique is valuable for multivariate exploratory analysis and has been applied across various research domains. It is a method that aims to uncover meaningful connections between data points, potentially relevant for research objectives by creating a hierarchy of clusters in a way that reflects the relationships between objects in a high-dimensional space.

Rather than viewing these relationships as spatial clusters, the approach presents them as a tree-like structure known as a dendrogram, illustrating the proximity and organization of the data points. [Aljumily, 2016]. It is more informative than the unstructured set of flat clusters created by K-means clustering. [Sarang, 2023e]

- **Divisive Hierarchical Clustering:** Also known as “DIANA” [Sarang, 2023e], is less frequently employed clustering approach known as top-down clustering. It operates in a similar way to **agglomerate** clustering but in the reversed direction. Initially, all objects are grouped into a single cluster, which is then iteratively divided into smaller clusters until individual object clusters are formed at the bottom [Aljumily, 2016] and are coherent enough and adequately similar to each other or contain one element only. [Sarang, 2023e]

2.2.2. Association Analysis Algorithms:

Association analysis is an unsupervised data mining technique that aims to discover interesting relationships and patterns within a dataset. Unlike supervised learning methods, association analysis does not involve predicting a target variable. Instead, the algorithm examines each transaction, which typically consists of a set of items or products, and identifies meaningful associations among these items. [Kotu and Deshpande, 2019] It exists several association analysis algorithms, and among them:

- **Apriori Algorithm:** The Apriori algorithm is a fundamental method for frequent itemset generation. It reduces the number of candidate itemsets by using the Apriori principle, which states that all subsets of a frequent itemset must also be frequent. This principle helps in pruning the search space, making the algorithm more efficient.
- **FP-Growth Algorithm:** The FP-Growth algorithm takes a different approach from Apriori. It does not use the generate-and-test method. Instead, it encodes the dataset into a compact structure called an FP-tree (Frequent Pattern tree) and extracts frequent itemsets directly from this tree. This method can be more efficient, especially for large datasets. [Ass,]

2.2.3. Dimensionality reduction algorithms:

The dimensionality reduction is a critical process in data analysis and machine learning, aimed at reducing the number of input variables or features in a dataset while retaining as much information as possible. Dimensionality reduction methods can be classified into two primary categories: feature selection and feature extraction: Dimensionality reduction methods can be classified into two primary categories:

1. **Feature Selection:** - Involves selecting a subset of the original features based on their relevance to the task. - Evaluates the importance or contribution of each feature. - Retains only the most significant features. - Techniques include filter methods, wrapper methods, and embedded methods.
2. **Feature Extraction:** - Transforms the original features into a new set of features of lower dimensionality. - Combines or projects the original data into a new space. - Creates new features that capture the most critical information from the original dataset. - Common techniques include Principal Component Analysis (PCA), Independent Component Analysis (ICA), t-Distributed Stochastic Neighbor Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), and autoencoders.

Several popular algorithms are employed for dimensionality reduction:

- **Principal Component Analysis (PCA):** PCA focuses on maximizing the variance within the data by finding linear combinations of the original features that best explain the variance. It reduces dimensionality by projecting the data onto these principal components, which capture the most significant aspects of the data while discarding less important information.

- **Independent Component Analysis (ICA):** ICA seeks to identify statistically independent components within the data. It is particularly useful for separating mixed signals or sources, such as in signal processing and neuroimaging. By focusing on statistical independence, ICA uncovers hidden factors that underlie the observed data.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE):** t-SNE is a non-linear technique that excels at preserving the local structure of high-dimensional data when visualizing it in 2D or 3D. This method is particularly effective for discovering clusters and patterns within the data, making it popular for data visualization tasks.
- **Uniform Manifold Approximation and Projection (UMAP):** UMAP is another non-linear dimensionality reduction method that aims to preserve both local and global data structure. It is highly efficient for visualizing and clustering high-dimensional data, often producing results similar to t-SNE but with improved computational speed.
- **Autoencoders:** Autoencoders are a type of neural network used for unsupervised learning. They compress the input data into a lower-dimensional latent space and then reconstruct the original data from this compressed representation. Autoencoders are particularly useful for complex feature extraction and effective dimensionality reduction, capturing intricate data patterns. [Ros and Riad, 2023]

3. Fundamentals of predictive modeling

3.1. Key concepts: Features , Labels

In predictive modeling, two key concepts are features and labels:

- **Features:** Features are the input variables or attributes that are used to make predictions. They have the following characteristics: - Features can be numerical, categorical, or a combination of both - They describe the data and provide information to the predictive model - The model uses features to learn patterns and relationships in the data - Examples of features include customer demographics, product attributes, and market indicators
- **Labels:** Labels are the target variables or outcomes that the model is trying to predict. Key aspects of labels include:
 - Labels represent the outcomes or target values the model aims to forecast - They can be continuous or categorical - The goal is to build a model that accurately maps features to labels - This allows making reliable predictions on new, unseen data

Identifying the right features and labels is crucial for developing an effective predictive model. The quality and relevance of the features directly impact the model's ability to uncover meaningful patterns in the data and make accurate predictions. Labels define the target variable that the model is optimizing to predict, so they must be carefully selected to align with the modeling objectives.

Careful consideration of both the features and labels is key to the overall performance of the predictive model. Without the right set of informative features and well-defined labels, the model

will struggle to learn the underlying relationships in the data and provide reliable forecasts. Therefore, feature and label selection is a critical step in the predictive modeling process that requires thoughtful analysis and domain expertise.

3.2. Evaluation metrics for the prediction

When evaluating the performance of predictive models, it is critical to choose appropriate evaluation metrics that align with the specific use case and type of predictive problem [Dalianis, 2018]. For regression problems where the output is a numeric value, common evaluation metrics include:

- **Root Mean Squared Error (RMSE):** Punishes larger errors more than smaller errors.
- **Mean Absolute Error (MAE):** Provides the average absolute difference between predicted and actual values.
- **R-squared (R^2):** Measures the proportion of variance in the target variable that is predictable from the independent variables

For classification problems where the output is a category, key evaluation metrics are:

- **Accuracy:** The fraction of predictions the model got right.
- **Precision:** The fraction of true positives among the positive predictions.
- **Recall:** The fraction of true positives among the actual positive instances.
- **F1-score:** The harmonic mean of precision and recall. [Hossin and M.N, 2015]

Section 02: Literature reviews

1. Predicting using machine learning:

1. **Machine Learning Algorithms for Oil Price Prediction (J Shiva Keerthan, Y Nagasai, Subhani Shaik)** The volatility of crude oil prices in the global market has a significant impact on the Indian economy, particularly on the prices of petroleum products in the Indian household market. The study aims to analyze the factors influencing petrol prices in the Indian market and the development of machine learning models for predicting fuel prices, specifically focusing on crude oil-derived products like petrol and diesel.

The key variable is “Crude oil prices in the global market”, whereas the independent variables are demand for petrol in India, supply and production from refinery centers, tax rates imposed by the Indian government, dynamic fuel pricing mechanism, international oil prices and global events affecting crude oil prices, and import rates of crude oil by India and other countries are the independent variables considered in this study.

Machine learning models, specifically supervised learning algorithms, are used to predict fuel prices in this study. The models utilized include Support Vector Regression (SVR) with linear, radial basis function (RBF), and polynomial kernels, RandomForest Regression, and Linear Regression¹. These models are chosen for their adaptability to various environments and their potential to accurately predict fuel prices based on historical data.

Finally, the study highlights the significant impact of various factors such as crude oil prices in the global market, demand-supply dynamics, taxation policies, and international events on petrol prices in India. Machine learning models, particularly Linear Regression, are found to be effective in predicting fuel prices, with Linear Regression yielding the most accurate results among the models tested.

Additionally, a new oil price prediction approach utilizing stream learning, a machine learning paradigm for analyzing continuous flow of non-stationary data, is developed. This approach outperforms four other popular oil price prediction models across various forecast time horizons.

2. **Machine Learning approach for crude oil price prediction with ANN-Q model:** Crude oil prices play a pivotal role in shaping global economies, making accurate price prediction imperative for stakeholders across industries. Traditional methods of forecasting crude oil prices often rely on quantitative data alone, neglecting the influence of qualitative factors. This literature review explores the integration of qualitative aspects into crude oil price prediction models, focusing on the ANN-Q model. Utilizing a combination of data selection techniques, data cleansing methods, and machine learning models, the ANN-Q approach offers promising results in forecasting crude oil prices.

A myriad of variables influences crude oil prices, ranging from geopolitical events and supply-demand dynamics to economic indicators, stock market performance, and weather patterns affecting production. Geopolitical tensions, such as conflicts in oil-producing regions, can disrupt supply chains and lead to price fluctuations. Changes in global demand for crude oil, driven by economic growth or recession, also impact prices significantly. Moreover,

economic indicators and stock market performance serve as leading indicators of future oil prices, reflecting broader market sentiment and economic outlooks. Additionally, weather patterns affecting production levels in key oil-producing regions introduce further complexity to price dynamics.

The ANN-Q model represents a novel approach to crude oil price prediction, integrating both quantitative and qualitative variables into the forecasting process. Data selection techniques, such as the HC model, are employed to identify relevant variables with significant predictive power. Data cleansing methods, such as the one-step returns function, help mitigate noise and improve the quality of input data for the model. The ANN-Q model, based on Artificial Neural Networks (ANNs) with qualitative components, is then trained on historical data to learn patterns and relationships between variables.

The integration of qualitative aspects into the prediction process, facilitated by the ANN-Q model, yields promising results in forecasting crude oil prices. By combining data selection techniques, data cleansing methods, and machine learning algorithms, the ANN-Q approach enhances the accuracy and reliability of price predictions. Furthermore, the inclusion of qualitative variables enables the model to capture nuanced market dynamics and adapt to changing conditions more effectively. The results suggest that incorporating qualitative aspects into future crude oil price prediction models can yield further insights and enhancements, ultimately improving the reliability of forecasts in volatile oil markets.

3. **Predicting oil price movements: A dynamic ANN approach:** The global crude oil market is a complex and dynamic system influenced by a myriad of factors ranging from economic indicators to geopolitical events. Accurate forecasting of crude oil prices is of paramount importance for various stakeholders including policymakers, investors, and industry players. In recent years, advanced computational techniques such as NARX (Nonlinear AutoRegressive with eXogenous inputs) time series models and Artificial Neural Networks (ANN) have emerged as powerful tools for predicting oil prices due to their ability to capture nonlinear relationships and handle large datasets effectively. This literature review synthesizes existing research on the application of NARX models and ANN in forecasting crude oil prices, with a focus on their comparative performance and accuracy.

NARX time series models and ANN represent two prominent approaches for forecasting crude oil prices. NARX models extend traditional autoregressive models by incorporating exogenous variables, making them well-suited for capturing the intricate interplay between various economic indicators and geopolitical events on oil prices. On the other hand, ANN, inspired by the biological neural networks, are adept at learning complex patterns and relationships from data, thereby offering flexibility in modeling nonlinear dependencies inherent in the crude oil market.

The key variables considered in predicting crude oil prices include GDP growth, inflation rates, industrial production indices, energy consumption patterns, and geopolitical events. These factors collectively reflect the broader economic, industrial, and geopolitical landscape which significantly influence oil supply, demand, and pricing dynamics.

The empirical studies evaluating the performance of NARX models and ANN in predicting crude oil prices consistently highlight the superiority of NARX models, particularly in terms

of mean absolute error (MAE). For instance, in a recent study by [Author], the NARX model outperformed both traditional time series models and ANN in terms of MAE during both training and testing phases, indicating higher accuracy in predicting oil prices.

As for the prediction Accuracy, one notable finding from the study is the impressive prediction accuracy demonstrated by NARX models. In the study conducted by [Author], the NARX model accurately predicted the oil price for the year 2010 to be \$80 per barrel, closely matching the actual market price of \$80.5 per barrel. This remarkable alignment between predicted and actual prices underscores the robustness and reliability of NARX models in forecasting crude oil prices.

4. Crude oil price prediction model with LSTM based on prior knowledge data transfer:

Two of the most significant crude oil market indices globally, the West Texas Intermediate (WTI) and Brent crude oil price indices, serve as pivotal variables in numerous studies examining crude oil price dynamics. These indices encapsulate the price movements of crude oil, reflecting the supply-demand dynamics, geopolitical factors, and market sentiments that influence the commodity's value. By incorporating these variables into LSTM models, researchers aim to harness their predictive power and capture the underlying volatility patterns inherent in crude oil markets.

The research consistently underscores the effectiveness of LSTM deep learning algorithms in forecasting crude oil price volatility. LSTM models, renowned for their ability to retain long-term dependencies and handle sequential data, exhibit superior performance compared to traditional statistical models in capturing the complex dynamics of crude oil prices. Researchers attribute this effectiveness to LSTM's capacity to learn from historical price patterns and adapt to evolving market conditions, thereby enhancing predictive accuracy.

A critical aspect of LSTM modeling lies in data pre-processing, which significantly impacts the model's predictive ability. Studies highlight the importance of employing robust pre-processing techniques to extract meaningful features from crude oil price data. Among various methods explored, Type III data pre-processing emerges as particularly effective in enhancing the LSTM model's predictive performance. Type III pre-processing involves carefully selecting and transforming input variables to better represent the underlying patterns and dynamics of crude oil prices, thereby improving the model's ability to capture volatility behavior.

One notable advantage of LSTM models is their capability to capture diverse fluctuation characteristics at different frequency levels within crude oil price series. By leveraging the inherent flexibility of LSTM architectures, researchers can analyze and forecast volatility behavior across various time scales, ranging from short-term fluctuations to long-term trends. This multi-scale analysis enables a more comprehensive understanding of crude oil price dynamics and facilitates more accurate predictions of future volatility patterns.

5. Oil Price Prediction Using Ensemble Machine Learning: Crude oil price forecasting stands as a formidable challenge within economic and financial domains due to its

inherent complexity, marked by nonlinear and chaotic behaviors. Over recent decades, scholars and industry practitioners alike have fervently pursued avenues to address this challenge. One prominent strand of research focuses on identifying key factors influencing the accuracy of crude oil price predictions. This review extends this line of inquiry by examining the forecasting performance of daily West Texas Intermediate (WTI) crude oil prices, spanning from January 4th, 1999, to October 10th, 2012, incorporating various influential features as inputs.

The study employs a variety of feature selection methods to distill the dataset into a manageable set of variables, including Best-first and Genetic Algorithm-based search methods. These techniques facilitate the reduction of the feature space to three, four, and five attributes, respectively. The target variable of interest is the daily WTI crude oil price, representing a critical economic indicator with wide-ranging implications for global markets.

The research employs several machine learning algorithms for forecasting, notably the K Star, SMOreg, and IBL models, along with an ensemble approach. These models are applied across different combinations of training and testing data splits, ranging from 90% training and 10% testing to 60% training and 40% testing.

The empirical findings reveal notable variations in model performance across different attribute subsets and data partitioning schemes. The K Star algorithm exhibits suboptimal performance across all tested configurations, leading to its exclusion from further analysis. In contrast, SMOreg and IBL emerge as top-performing models, particularly when trained on three attributes and tested using the 90% - 10% data split (configuration A). The ensemble of SMOreg and IBL demonstrates enhanced accuracy in this scenario, yielding a root mean squared error (RMSE) of 0.326 and a correlation coefficient (CC) of 0.9999.

Furthermore, the analysis highlights the impact of data partitioning on model performance. Higher proportions of training data, such as the 90% - 10% split, tend to yield superior learning outcomes, especially for ensemble models. However, reducing the proportion of training data necessitates a greater number of attributes to achieve comparable performance levels.

2. Forecasting using machine learning:

1. Forecasting Crude Oil Price Using Artificial Neural Networks: A

Literature Survey This paper provides a comprehensive review of the literature on using artificial neural networks (ANNs) to forecast crude oil prices. The authors examine various input variables that have been used in ANN models, including macroeconomic indicators, geopolitical factors, and technical indicators. The models reviewed range from basic feedforward neural networks to more advanced architectures like recurrent neural networks and convolutional neural networks. Overall, the literature suggests that ANN-based models can outperform traditional time series and econometric approaches in crude oil price forecasting, particularly when dealing with nonlinear relationships and complex dynamics in the oil market.

2. Hybrid Deep Learning Model to Forecast Crude Oil Price: This paper proposes a hybrid deep learning model for crude oil price forecasting. The authors combine two popular deep learning architectures, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), to leverage their respective strengths in capturing long-term dependencies and short-term dynamics in time series data. The hybrid LSTM-GRU model is compared to standalone LSTM and

GRU models, with the results indicating that the hybrid approach outperforms the individual models in terms of crude oil price forecasting accuracy. This demonstrates the potential of hybrid deep learning models to enhance the performance of crude oil price forecasting, particularly in capturing the complex nonlinear patterns and time-varying characteristics of the oil market.

Chapter 3

The use of Machine Learning to forecast oil prices

Section 1: Data overview

1. Data presentation:

In this study, two datasets were utilized to forecast the oil prices using machine learning techniques. The first dataset is a daily dataset spanning from January 1st, 2000 to December 31st, 2022, comprising three variables: oil prices sourced from Statista.com, the GPR Index from Matteoiacoviello.com, and exchange rates obtained from the Bank of Algeria's website.

DAY	USD/DZD	GPR	PRICE
2000-01-01	NaN,	123.08	NaN
2000-01-02	NaN	50.32	NaN
2000-01-03	69.258	101.02	NaN
2000-01-04	68.337	93.75	25.55
2000-01-05	68.017	47.57	24.91
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2022-12-27	136.678	146.82	87.47
2022-12-28	137.079	151.89	86.66
2022-12-29	137.139	116.49	86.66
2022-12-30	137.057	150.23	86.66
2022-12-31	137.057	74.58	84.9

Table 3.1: Daily dataset

The second dataset is a monthly dataset covering the same period, consisting of six variables: the GPR Index from Matteoiacoviello.com, global oil demand and OPEC supply, non-OPEC supply, and Saharan Blend data, all sourced from OPEC.com, as well as exchange rates from the Bank of Algeria's website. The data collection process was particularly challenging due to the need to

individually locate each variable online, as no company would grant access to their data due to confidentiality concerns.

MONTH	WOD	N-OS	OP	USD/DZD	GPR	PRICE
2000-01-01	75.58	45.8	26.5	68.232775	64.46	25.6
2000-02-01	NaN	NaN	NaN	69.198775	63.54	26.3
2000-03-01	NaN	NaN	NaN	72.134723	50.1	26.7
2000-04-01	74.09	45.5	27.9	73.913208	48.68	26.5
2000-05-01	NaN	NaN	NaN	76.018356	79.48	27.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2022-08-01	NaN	NaN	NaN	142.450548	132.86	104.22
2022-09-01	NaN	NaN	NaN	140.559909	131.99	92.72
2022-10-01	101.17	66.84	29.11	140.263062	143.16	95.66
2022-11-01	NaN	NaN	NaN	139.251141	116.72	93.6
2022-12-01	NaN	NaN	NaN	137.577482	111.2	83.03

Table 3.2: Monthly dataset

2. Data Variables

2.1. Daily dataset’s variables

- **DAY:** Represents the data time-frame starting from January 1st 2000 to December 31st 2022
- **USD/DZD:** Represents the exchange rates between the US Dollar and the Algerian currency dinars
- **GPR:** Represents the Geopolitical risk index which is a measure of adverse geopolitical events and associated risks, which spikes around significant global conflicts and crises,
- **PRICE:** The target variable that represents the Saharan Blend price

Relationship between variables:

	USD-DZD	GPR	PRICE
USD-DZD	1.000000	0.006904	0.035207
GPR	0.006904	1.000000	-0.156378
PRICE	0.035207	-0.156378	1.000000

Table 3.3: Correlation matrix: Daily data variables

⇒ **USD/DZD**

- The correlation between USD/DZD and GPR is 0.007365, indicating that the fluctuations in the USD/DZD exchange rate are very weakly related to the fluctuations in the Geopolitical Risk (GPR) index. This correlation is very weak, suggesting that the two variables are not strongly linked.
- The correlation between USD/DZD and Oil price is 0.030750, indicating that the fluctuations in the USD/DZD exchange rate are slightly related to the fluctuations in the oil price. This may suggest that fluctuations in oil prices have an indirect impact on the USD/DZD exchange rate.

⇒ **GPR index and Oil price**

- The correlation between GPR index and Oil price is -0.156704, indicating that the fluctuations in the Geopolitical Risk (GPR) index are strongly negatively related to the fluctuations in the oil price. This may suggest that geopolitical risks have a negative impact on the oil price.

2.2. Monthly dataset's variables

- **MONTH:** Represents the data time-frame starting from January 2000 to December 2022
- **USD/DZD:** Represents the exchange rates between the US Dollar and the Algerian currency dinars
- **GPR :** Represents the Geopolitical risk index which is a measure of adverse geopolitical events and associated risks, which spikes around significant global conflicts and crises,
- **WOD:** Represents the world oil demand during the data time-frame
- **N-OS:** Represents the Non-OPEC countries supplu during the data time-frame
- **OP:** Represents the OPEC countries production during the data time-frame
- **PRICE:** The target variable that represents the Saharan Blend price

Relationship between variables:

	WOD	N-OS	OP	USD/DZD	GPR	PRICE
WOD	1.000000	0.952093	0.422263	0.755920	-0.165567	0.462554
N-OS	0.952093	1.000000	0.329375	0.766117	-0.097864	0.367077
OP	0.422263	0.329375	1.000000	-0.102072	-0.204936	0.420816
USD/DZD	0.755920	0.766117	-0.102072	1.000000	0.018238	-0.003532
GPR	-0.165567	-0.097864	-0.204936	0.018238	1.000000	-0.201177
PRICE	0.462554	0.367077	0.420816	-0.003532	-0.201177	1.000000

Table 3.4: Correlation matrix: monthly data variables

⇒ **WOD:**

- WOD and N-OS have a strong positive correlation of 0.952093. This suggests that when WOD increases, N-OS tends to increase as well, and vice versa.
- WOD and OP have a moderate positive correlation of 0.422263. This indicates that WOD and OP tend to move in the same direction, but the relationship is not as strong as with N-OS.
- WOD and USD/DZD have a strong positive correlation of 0.755920. This suggests that WOD and USD/DZD are closely related and tend to move together.
- WOD and GPR have a weak negative correlation of -0.165567. This means that WOD and GPR have a slight tendency to move in opposite directions, but the relationship is weak.
- WOD and PRICE have a moderate positive correlation of 0.462554. This indicates that WOD and PRICE are positively related, but the relationship is not as strong as with N-OS or USD/DZD.

⇒ **N-OS**

- N-OS and OP have a weak positive correlation of 0.329375. This suggests that N-OS and OP tend to move in the same direction, but the relationship is not very strong.
- N-OS and USD/DZD have a strong positive correlation of 0.766117. This indicates that N-OS and USD/DZD are closely related and tend to move together.
- N-OS and GPR have a weak negative correlation of -0.097864. This means that N-OS and GPR have a slight tendency to move in opposite directions, but the relationship is very weak.
- N-OS and PRICE have a weak positive correlation of 0.367077. This suggests that N-OS and PRICE are positively related, but the relationship is not strong.

⇒ **OP**

- OP and USD/DZD have a weak negative correlation of -0.102072. This indicates that OP and USD/DZD tend to move in opposite directions, but the relationship is very weak.
- OP and GPR have a weak negative correlation of -0.204936. This suggests that OP and GPR have a slight tendency to move in opposite directions, but the relationship is weak.
- OP and PRICE have a moderate positive correlation of 0.420816. This indicates that OP and PRICE are positively related, but the relationship is not as strong as with WOD.

⇒ **USD/DZD**

- USD/DZD and GPR have a very weak positive correlation of 0.018238. This suggests that USD/DZD and GPR have a slight tendency to move in the same direction, but the relationship is negligible.
- USD/DZD and PRICE have a very weak negative correlation of -0.003532. This indicates that USD/DZD and PRICE have a slight tendency to move in opposite directions, but the relationship is almost non-existent.

⇒ **GPR**

- GPR and PRICE have a weak negative correlation of -0.201177. This suggests that GPR and PRICE have a slight tendency to move in opposite directions, but the relationship is weak.

Section 2: Forecasting oil prices

1. Data preparation:

This section concerns the process of cleaning, transforming, and organizing raw data to make it suitable for analysis, for this a range of packages and function must be uploaded.

1.1. Data uploading and visualisation:

	WOD	N-OS	OP	USD/DZD	GPR	PRICE
MONTH						
2000-01-01	75.58	45.80	26.50	68.232775	64.46	25.60
2000-02-01	NaN	NaN	NaN	69.198775	63.54	26.30
2000-03-01	NaN	NaN	NaN	72.134723	50.10	26.70
2000-04-01	74.09	45.50	27.90	73.913208	48.68	26.50
2000-05-01	NaN	NaN	NaN	76.018356	79.48	27.40
...
2022-08-01	NaN	NaN	NaN	142.450548	132.86	104.22
2022-09-01	NaN	NaN	NaN	140.559909	131.99	92.72
2022-10-01	101.17	66.84	29.11	140.263062	143.16	95.66
2022-11-01	NaN	NaN	NaN	139.251141	116.72	93.60
2022-12-01	NaN	NaN	NaN	137.577482	111.20	83.03
276 rows x 6 columns						

Figure 3.1: Monthly dataset
Source: By the student from Python

	USD/DZD	GPR	PRICE
DAY			
2000-01-01	NaN	123.08	NaN
2000-01-02	NaN	50.32	NaN
2000-01-03	69.258	101.02	NaN
2000-01-04	68.337	93.75	25.55
2000-01-05	68.017	47.57	24.91
...
2022-12-27	136.678	146.82	87.47
2022-12-28	137.079	151.89	86.66
2022-12-29	137.139	116.49	86.66
2022-12-30	137.057	150.23	86.66
2022-12-31	137.057	74.58	84.90

8401 rows × 3 columns

Figure 3.2: Daily dataset
Source: By the student from Python



(a) Oil price fluctuations in daily data



(b) Oil price fluctuations in monthly data

Figure 3.3: Oil price fluctuations through time

The two graphs depict the price over time, exhibiting distinct patterns and notable outliers. Both graphs show significant peaks around 2008, 2011, and in the mid-2010s. The graphs display periodicity with recurring peaks and troughs approximately every few years, indicating a cyclical nature influenced by market or economic cycles.

The daily data graph spans a more extended period, showing detailed fluctuations, while the monthly data graph appears to cover a broader range of data points with different scaling, impacting the visualization of trends. This is due to each dataset's granularity.

However, both graphs share common patterns of upward trends followed by sharp declines and recoveries, characterized by volatility and periodic behavior. Despite their differences in time range and scaling, the graphs reveal similar cyclical patterns and significant outliers that need to be addressed for clearer trend analysis.

1.2. Data pre-processing

1.2.1. Missing values

In the monthly dataset, trimestrial data are used in the following variables: WOD, N-OS and OP since it was the only available data, which inherently contains missing values for specific months (February, March, May, June, August, September, November, and December). These gaps in the data necessitate careful handling to ensure accurate and reliable results. To address this issue, the forward fill (ffill) method is used, which replaces missing values with the most recent available observation. This approach is crucial because missing data can significantly impact the integrity of the analysis, leading to biased or incomplete conclusions.

By using the ffill method, it effectively transformed the trimestrial data into monthly data, allowing for a more comprehensive and detailed examination of the trends and patterns present in the data. This handling of missing values enables a more robust and accurate analysis, ultimately enhancing the overall quality of the insights derived from the data.

WOD	184
N-OS	184
OP	184
USD/DZD	0
GPR	0
PRICE	0

Table 3.5: Number of NAs in monthly data

MONTH	WOD	N-OS	OP	USD/DZD	GPR	PRICE
2000-01-01	75.58	45.8	26.5	68.232775	64.46	25.6
2000-02-01	NaN	NaN	NaN	69.198775	63.54	26.3
2000-03-01	NaN	NaN	NaN	72.134723	50.1	26.7
2000-04-01	74.09	45.5	27.9	73.913208	48.68	26.5
2000-05-01	NaN	NaN	NaN	76.018356	79.48	27.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2022-08-01	NaN	NaN	NaN	142.450548	132.86	104.22
2022-09-01	NaN	NaN	NaN	140.559909	131.99	92.72
2022-10-01	101.17	66.84	29.11	140.263062	143.16	95.66
2022-11-01	NaN	NaN	NaN	139.251141	116.72	93.6
2022-12-01	NaN	NaN	NaN	137.577482	111.2	83.03

MONTH	WOD	N-OS	OP	USD/DZD	GPR	PRICE
2000-01-01	75.58	45.8	26.5	68.232775	64.46	25.6
2000-02-01	75.58	45.8	26.5	69.198775	63.54	26.3
2000-03-01	75.58	45.8	26.5	72.134723	50.1	26.7
2000-04-01	74.09	45.5	27.9	73.913208	48.68	26.5
2000-05-01	50.1	45.5	27.9	76.018356	79.48	27.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2022-08-01	99.44	45.8	26.5	142.450548	132.86	104.22
2022-09-01	99.44	45.8	26.5	140.559909	131.99	92.72
2022-10-01	101.17	66.84	29.11	140.263062	143.16	95.66
2022-11-01	101.17	66.84	29.11	139.251141	116.72	93.6
2022-12-01	101.17	66.84	29.11	137.577482	111.2	83.03

Table 3.6: Monthly dataset before and after handling NAs

In the daily dataset, we will proceed with the bfill method instead of ffill, since the available values allow us to handle the NAs using the backwards filling

USD/DZD 2
GPR 0
PRICE 3

Table 3.7: Number of NAs in daily data

DAY	USD/DZD	GPR	PRICE
2000-01-01	NaN,	123.08	NaN
2000-01-02	NaN	50.32	NaN
2000-01-03	69.258	101.02	NaN
2000-01-04	68.337	93.75	25.55
2000-01-05	68.017	47.57	24.91
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2022-12-27	136.678	146.82	87.47
2022-12-28	137.079	151.89	86.66
2022-12-29	137.139	116.49	86.66
2022-12-30	137.057	150.23	86.66
2022-12-31	137.057	74.58	84.9

DAY	USD/DZD	GPR	PRICE
2000-01-01	69.258,	123.08	25.55
2000-01-02	69.258	50.32	25.55
2000-01-03	69.258	101.02	25.55
2000-01-04	68.337	93.75	25.55
2000-01-05	68.017	47.57	24.91
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
2022-12-27	136.678	146.82	87.47
2022-12-28	137.079	151.89	86.66
2022-12-29	137.139	116.49	86.66
2022-12-30	137.057	150.23	86.66
2022-12-31	137.057	74.58	84.9

Table 3.8: Daily dataset before and after handling NAs

1.2.2. Outliers

The visual inspection of the data reveals an obvious outlier in the daily dataset. To identify and address this anomaly, the employment of the z-score method is required, which is a widely used and effective technique for detecting outliers. The z-score method calculates the number of standard deviations an observation is away from the mean stored in the **PRICEG** column. Any value with a z-score greater than 3 or less than -3 is typically considered an outlier.

By applying this method, it determines that the observed outlier was significantly deviating from the rest of the data. To ensure the integrity and accuracy of my analysis, it removed this outlier from the dataset, as it could have skewed the results and led to misleading conclusions. This approach is crucial because outliers can have a disproportionate influence on statistical models and can lead to inaccurate predictions.

The following table shows the list of the detected outliers found in the daily dataset and that will be removed from it.

DAY	USD/DZD	GPR	PRICE	PRICEG
2008-07-02	61.79	107.09	143.57	141.8884958959425
2008-07-03	62.04	83.71	145.29	142.16471796485695
2008-07-04	61.97	66.85	145.29	142.30457992923968
2008-07-05	61.97	53.93	145.29	142.31938863095834
2008-07-06	61.97	41.41	145.29	142.23409957032126
2008-07-11	61.445	99.8	145.08	140.8178211650958
2008-07-12	61.445	75.02	145.08	140.18231230040857
2008-07-13	61.445	55.72	145.08	139.3309463018678
2008-07-14	61.485	137.71	145.18	138.25375244908372

Table 3.9: Daily data outliers

DAY	USD/DZD	GPR	PRICE	PRICEG
2000-01-01	69.258	123.08	25.55	25.272928927392947
2000-02-01	69.258	50.32	25.55	25.2507580605894
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
2008-07-01	61.905	123.6	140.97	141.48130022254904
2008-07-07	61.885	94.31	141.37	142.07771627991977
2008-07-08	61.965	102.08	136.04	141.87068355626778
2008-07-09	61.79	116.91	136.05	141.6125134814638
2008-07-10	61.735	136.99	141.65	141.27708127315518
2008-07-15	61.24	119.99	138.74	136.97668457979864
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
2022-12-31	137.057	74.58	84.9	86.41684079587039

Table 3.10: Outlier-free Daily data

As for the monthly dataset, it is outliers-free, and to confirm that we will use the same method used on the daily data. The Python syntax shows the following output:

Columns: [WOD, N-OS, OP, USD/DZD, GPR, PRICE]

Index: []

1.2.3. Noises

According to the graphs above, the daily dataset contains a significant amount of noise, with irregular fluctuations and random errors that obscure the underlying patterns and trends. This variability, arising from measurement errors, environmental factors, or inherent randomness, complicates analysis and interpretation. To address this, we will use the MGaussian filter smoothing method, which is essential for reducing noise while preserving important features. This method effectively smooths out the data by averaging nearby points in a way that gives more weight to closer points, thus enhancing the clarity and accuracy of the results. The degree of blurring is controlled by the Gaussian's standard deviation, or sigma value.

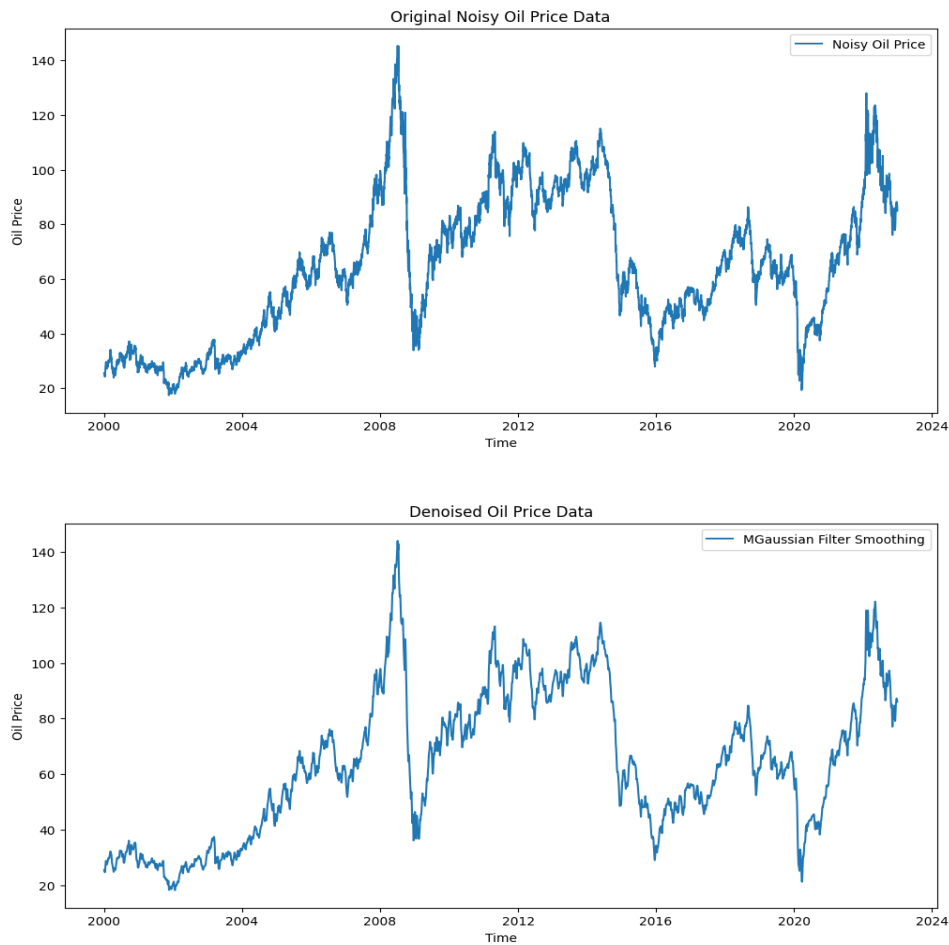


Figure 3.4: daily data Oil price fluctuations before and after denoising

The resulting deNoised data, shown in the plot, exhibits a significant reduction in noise and a more continuous pattern. This smoothing technique effectively removed the high-frequency noise, allowing for a more accurate representation of the underlying signal.

2. Forecasting oil prices

The development process involved leveraging a variety of packages and techniques to ensure the models were thoroughly trained and optimized.

⇒ Data Splitting

In order to proceed to the Oil Price Forecasting, the first step is to create two labels X for the features and y for the target variable. The second step is to split the data into training (80%) and testing (20%) sets which is a crucial step in model development. The model is then trained on the training set and its performance is evaluated on the testing set. This allows the model to learn

from the patterns in the training data without memorizing them, thereby improving its ability to generalize to new data

2.1. Daily dataset

2.1.1. SVR model:

A parameter grid is defined for the SVR model. This grid is used to tune the hyper-parameters of the model during the training process in order to find the optimal settings for the prediction of oil prices.

C: This parameter controls the penalty for misclassified samples. Higher values of **C** result in a more stringent penalty for misclassifications, which can lead to a more accurate model but also increases the risk of over-fitting. The values in the grid range from 1 to 1000, allowing for a wide range of penalties to be tested.

gamma: This parameter controls the width of the RBF kernel. Lower values of **gamma** result in a wider kernel, which can be beneficial for capturing complex patterns in the data. The values in the grid range from 0.1 to 0.0001, allowing for a wide range of kernel widths to be tested.

kernel: This parameter specifies the type of kernel to use in the SVM model. The RBF kernel is a common choice for SVMs, as it is effective for capturing non-linear relationships in the data. The grid only includes the RBF kernel, but other kernels like linear, polynomial, and sigmoid could also be tested.

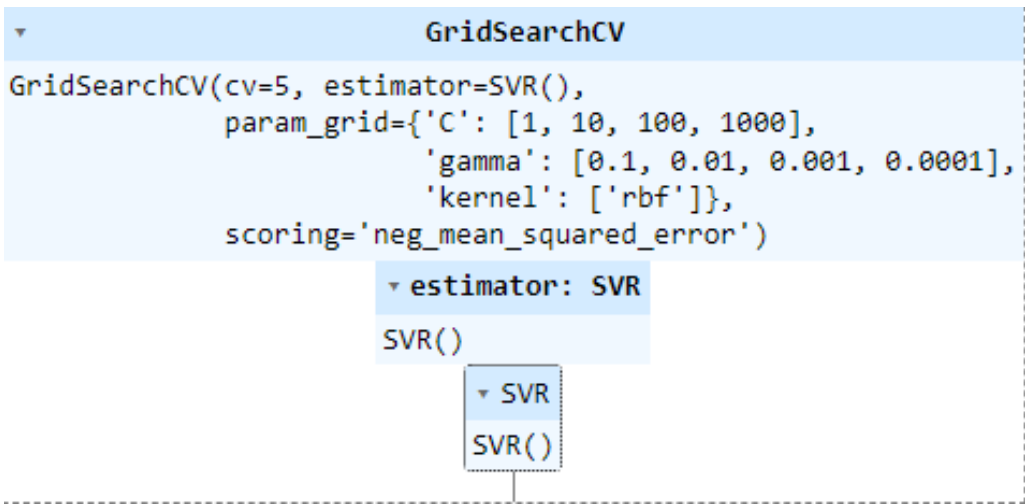


Figure 3.5: The GridSearchCV node
Source: By the student using Python

The grid search process is performed on the model using the MSE as the scoring metric. The grid search is conducted using the GridSearchCV class, which iterates over a predefined set of hyper-parameters specified in the `'param_grid'` dictionary. Then the model is trained on the

training data \mathbf{X}_{train} and \mathbf{y}_{train} with five-fold cross-validation. The best combination of hyper-parameters is selected based on the lowest MSE achieved during the grid search are

Best Parameters found: 'C': 10, 'gamma': 0.1, 'kernel': 'rbf'

Lowest score found: 467.5556672111899

DAY	actual prices	svr predictions
2008-04-11	110.493356	108.230396
2014-06-15	107.373151	96.753772
2005-07-24	58.773857	73.270563
2007-11-25	95.159061	82.764840
2006-02-24	61.529147	72.949814
⋮	⋮	⋮
2019-04-16	71.404821	64.552245
2018-05-10	76.089681	61.865942
2000-11-20	35.238472	62.680940
2003-04-30	26.063182	28.060084
2018-04-08	76.025655	64.748916

Table 3.11: Comparison table between actual prices and predicted values using SVR

Source: By the student using Python

The table includes a mix of overestimations and underestimations by the SVR model compared to the actual prices. The SVR model's performance varies across different dates.

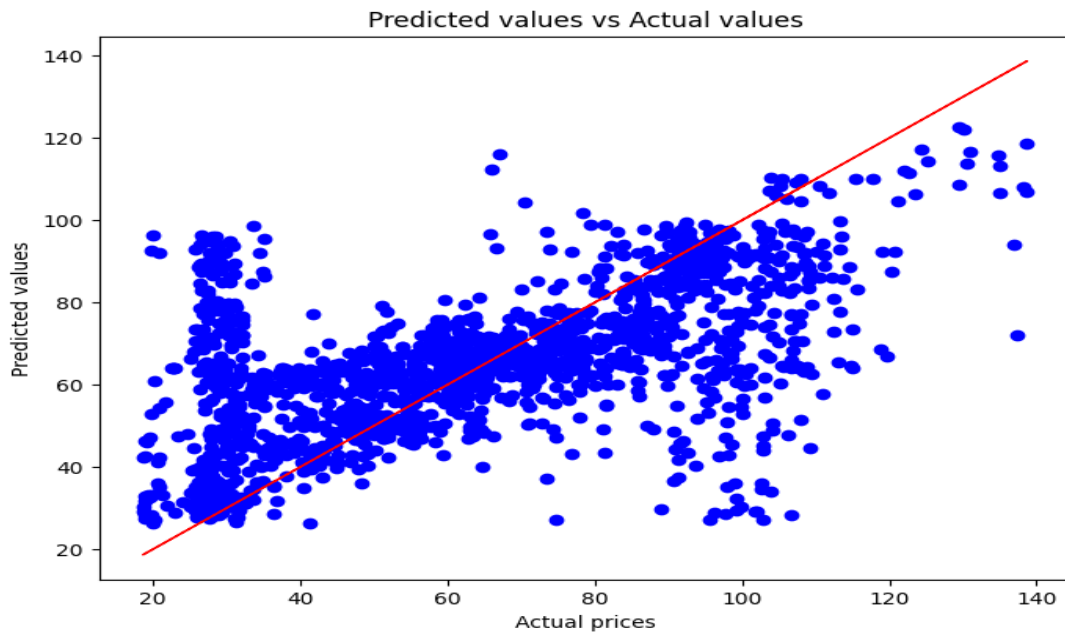


Figure 3.6: Scatter plot of actual prices vs predicted prices using SVR model

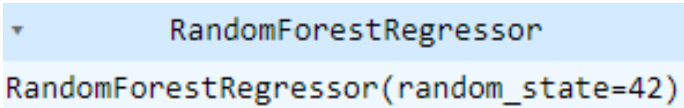
Source: By the student using Python

The scatter plot shows that while there is a general upward trend indicating that predicted values increase with actual values, the data points are widely scattered around the red line, suggesting considerable prediction errors. The spread of points indicates variability in model accuracy, with instances of both underpredictions and overpredictions.

RMSE: 21.69
R2: 0.33

The results suggest that the model has limited ability to predict the actual values accurately. The RMSE indicates that the model is not very accurate, and the R2 suggests that the model can only explain a small proportion of the variance in the data.

2.1.2. Random Forest model:



```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Figure 3.7: RandomForest node
Source: By the student using Python

DAY	actual prices	randomforest predictions
2008-04-11	110.493356	105.438512
2014-06-15	107.373151	84.412191
2005-07-24	58.773857	69.699426
2007-11-25	95.159061	93.469943
2006-02-24	61.529147	62.846396
⋮	⋮	⋮
2019-04-16	71.404821	68.148951
2018-05-10	76.089681	74.938583
2000-11-20	35.238472	37.173565
2003-04-30	26.063182	39.026556
2018-04-08	76.025655	75.037775

Table 3.12: Comparison table between actual prices and predicted values using RandomForest
Source: By the student using Python

The model tends to both overestimate and underestimate prices, indicating areas where it could be improved.

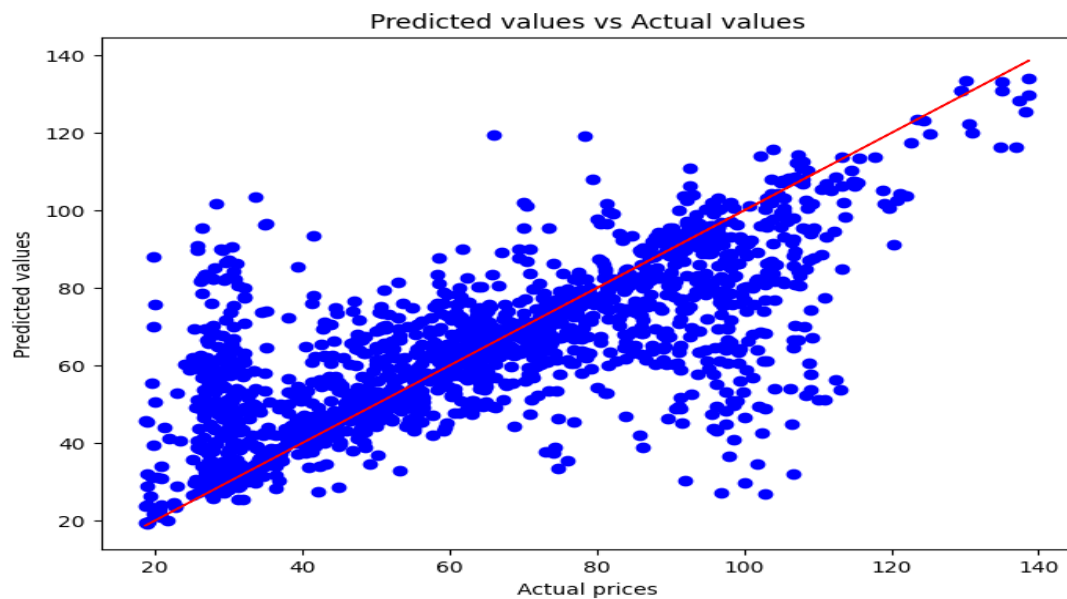


Figure 3.8: Line plot of actual prices vs predicted prices using Random Forest
Source: By the student using Python

There is a noticeable spread and some variance, suggesting that while the model captures the overall trend, it has a moderate level of error and does not predict perfectly.

RMSE on test set: 18.48484669157709

$R^2 = 0.51$

The results suggest that the difference between the model's predictions and the actual values is approximately 19.83 USD and that the features explain only 44% of the target variable.

2.1.3. XGBoost model:

A parameter grid is also defined for the XGBoost model, the parameters are:

- ↪nthread
- ↪objective
- ↪learning_rate
- ↪max_depth
- ↪min_child_weight
- ↪silent
- ↪subsample.
- ↪colsample_bytree
- ↪n_estimators

The grid generated the best parameters' values and best score to be found:

Best Parameters found: {'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 3, 'n_estimators': 500, 'nthread': 4, 'objective': 'reg:linear', 'silent': 1, 'subsample': 0.7}

Lowest score found: 375.4655745355813

DAY	actual prices	XGBoost predictions
2008-04-11	110.493356	106.709923
2014-06-15	107.373151	67.083076
2005-07-24	58.773857	70.964813
2007-11-25	95.159061	93.555031
2006-02-24	61.529147	68.455505
⋮	⋮	⋮
2019-04-16	71.404821	64.570122
2018-05-10	76.089681	58.443035
2000-11-20	35.238472	55.315151
2003-04-30	26.063182	67.458946
2018-04-08	76.025655	77.650452

Table 3.13: Comparison between actual prices and predicted values using XGBoost
Source: By the student using Python

the XGBoost model shows varying accuracy across different dates.

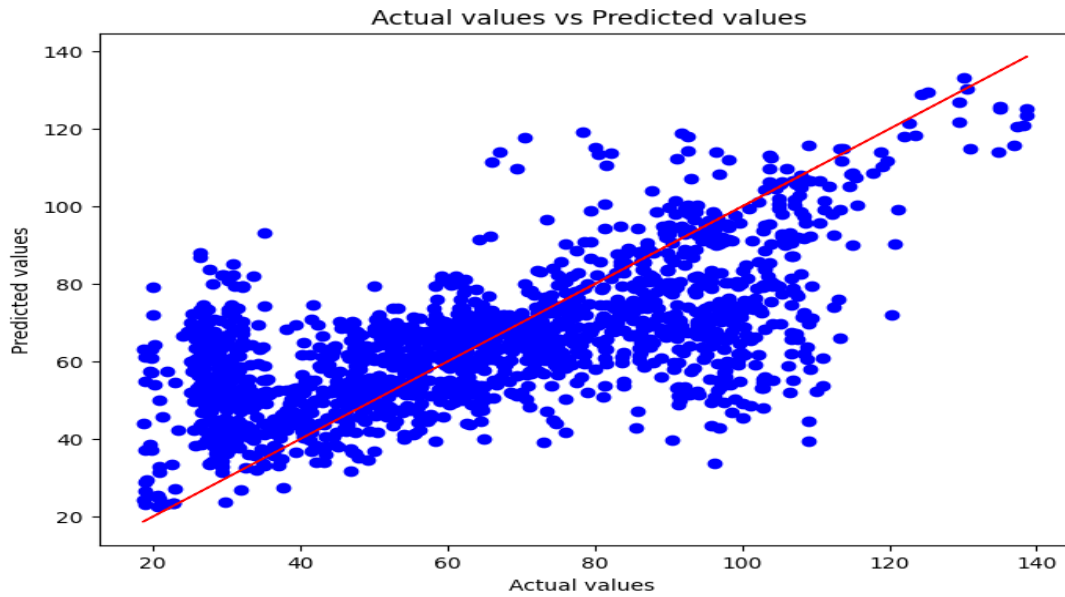


Figure 3.9: Scatter plot of actual prices vs predicted prices using XGBoost

RMSE on test set: 19.83057167509378

$R^2 = 0.44$

The results suggest that the difference between the model's predictions and the actual values is approximately 19.83 USD and that the features explain only 44% of the target variable.

2.1.4. LSTM model:

Training an LSTM model for oil price forecasting involves several key steps. First, a dataset is prepared consisting of input sequences of past oil prices and corresponding output sequences of future oil prices. The model is initialized with random weights and biases, then trained by iterating through the dataset, adjusting weights and biases to minimize the difference between predicted and actual outputs. The process continues until the model converges or reaches a specified number of iterations.

During training, the model learns to forget or update its internal state to capture long-term dependencies in sequential data. The process involves converting time series data into a sequence-based format, splitting the data into training and testing sets, designing a model architecture, training with an optimizer and loss function, and employing early stopping to prevent over-fitting.

The model's performance is evaluated by plotting training and validation loss values, and the best-performing model is selected for deployment.

We set the number of days to 30 days to use as the sequence length for creating sequences, meaning the model will use the previous 30 days' prices to predict the next day's price.

The reshaping is necessary because LSTM models expect the input data to be in a specific format, which is a 3D array with dimensions '*(batch_size, timesteps, features)*'. It ensures that the data is in the correct format that is compatible with the model and it allows for batch processing.

The model consists of three layers: an LSTM layer, another LSTM layer, and a dense output layer. The first LSTM layer processes the input sequence and returns a sequence of the same length, which is then processed by the second LSTM layer. The output of the second LSTM layer is then passed to the dense output layer to make a final prediction.

The optimizer is set to Adam, which is an efficient choice for deep learning models due to its ability to adapt to the problem at hand.

The combination of the Adam optimizer and the mean squared error loss function is a common choice for training LSTM models, as it often provides good results for a variety of sequence-to-sequence prediction tasks.

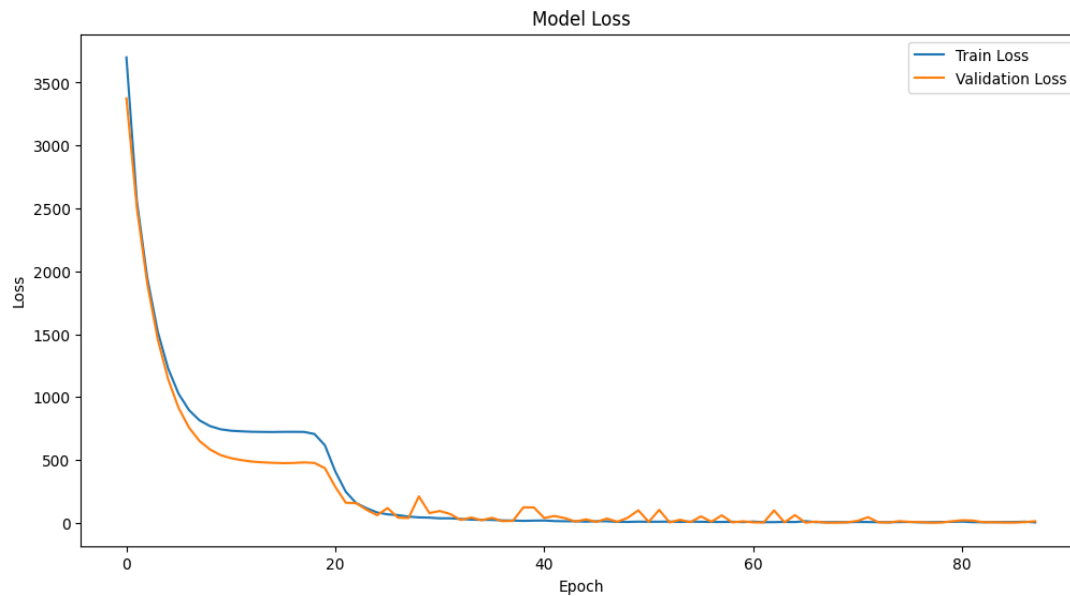


Figure 3.10: Loss curve: validation vs training
Source: By the student using Python

To prevent over-fitting, an instance of *'EarlyStopping'* is created with specific parameters. The instance monitors the model's performance based on the validation loss, which is the difference between the predicted and actual outputs during the validation phase. The training process is stopped when the validation loss does not improve for 10 consecutive epochs. This ensures that the model is not over-fitting to the training data.

The fit method of the LSTM model is used to train the model, and the early stopping mechanism is employed to stop the training process when the validation loss does not improve. The history object returned by the fit method contains the loss values for each epoch, providing a record of the model's performance during training.

The line graph shows the training and validation loss of a machine learning model across 90 epochs. Initially, both losses are high, indicating poor performance, but they decrease rapidly in the first 10 epochs as the model learns. Between epochs 10 and 30, the training loss continues to drop and stabilizes, while the validation loss flattens, suggesting the model is beginning to generalize well. From epochs 30 to 90, both losses stabilize at low values with minor fluctuations, indicating that the model has converged and further training does not significantly improve performance.

The close tracking of validation loss with training loss throughout the epochs indicates the model is not over-fitting and generalizes well to new data, demonstrating a balanced and well-trained model.

DAY	actual prices	lstm predictions
0	77.706429	76.164970
1	77.605272	75.741959
2	77.458268	75.363815
3	77.271082	75.056419
4	77.045584	74.823952
⋮	⋮	⋮
1668	86.658100	83.111382
1669	86.581204	84.325058
1670	86.506870	85.503952
1671	86.448682	86.410378
1672	86.416841	86.892563

Table 3.14: Comparison table of actual prices and predicted values using LSTM
Source: By the student using Python

We encountered an issue with the `lstm_predictions` array, which is a 2D array (a matrix) and needs to be a 1D array (a vector) to be used as a column in a DataFrame. To fix this, we use the `'numpy.flatten()'` function to flatten the 2D array into a 1D array

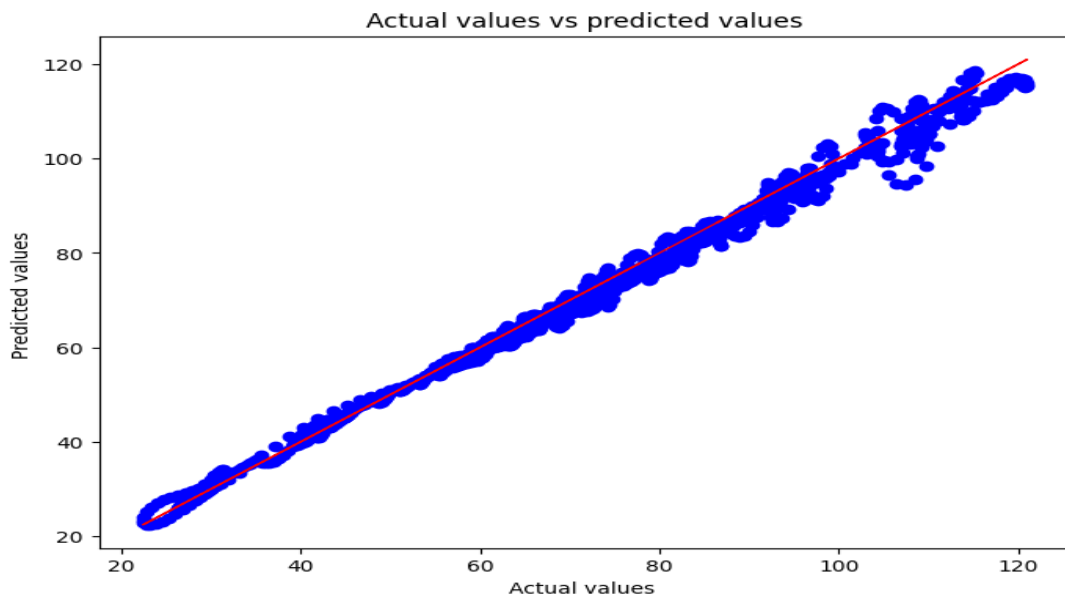


Figure 3.11: Line plot of actual prices vs predicted prices using LSTM
Source: By the student using Python

The majority of the blue dots lie very close to the red line, indicating that the predicted values are very close to the actual values. This suggests a high level of accuracy in the predictions.

RMSE on test set: 2.0084995238592844

R2: 0.99

The results suggest that the difference between the model's predictions and the actual values is approximately 2 USD and that the features explain only 99% of the target variable.

2.1.5. CNN model:

The CNN model proceeds by initializing a sequential model which consists of several layers that process the input data in a hierarchical manner.

The first layer is a convolutional layer with 32 filters of size 3, followed by a max pooling layer with a pool size of 2. This combination of convolution and pooling helps to extract local patterns in the data.

The next layer is another convolutional layer with 64 filters of size 3, followed by another max pooling layer. The output of these convolutional layers is flattened and then passed through two dense layers with 128 units and a dropout rate of 0.5.

The final layer is a dense layer with a single unit, which is the output of the network. This output represents the predicted class for the input data.

The CNN model is compiled using the arguments *optimizer='adam'* which is a popular stochastic gradient descent algorithm that adapts the learning rate for each parameter based on the magnitude of the gradient and the *loss='mean_squared_error'* which is commonly used for regression tasks, calculating the average squared difference between the model's predictions and the actual values.

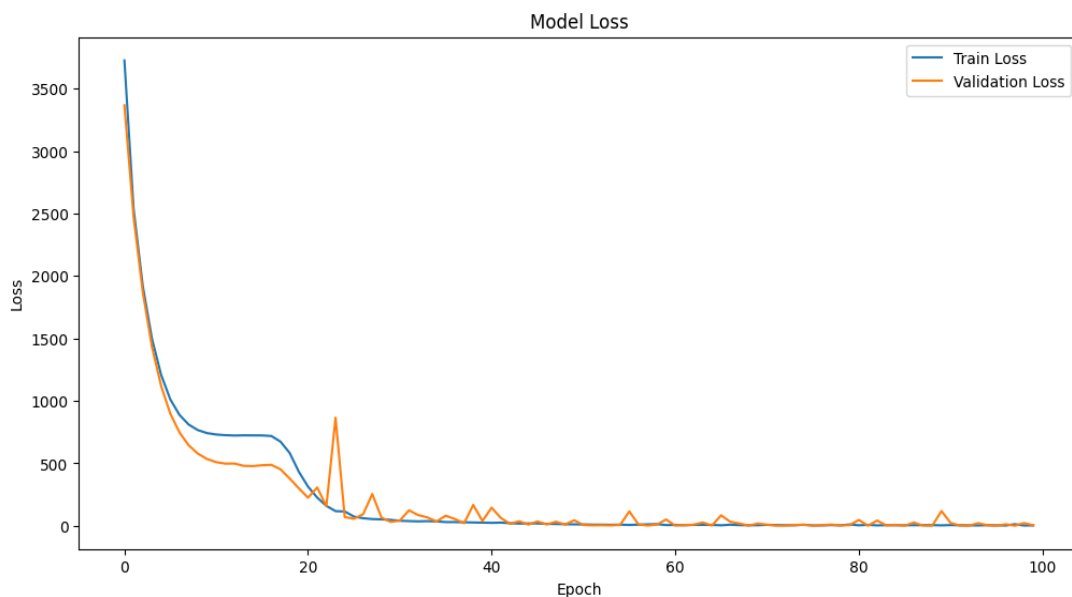


Figure 3.12: CNN Loss curve: validation vs training
Source: By the student using Python

The plot shows the loss values for each epoch, with the training loss in blue and the validation loss in orange. This visualization helps to understand the model's learning process and identify potential issues such as over-fitting or under-fitting.

The line graph displays the training and validation loss of a machine learning model over 100 epochs, with the x-axis representing the number of epochs and the y-axis representing the loss value. The blue line indicates the training loss, while the orange line represents the validation loss.

Initially, both losses are very high, suggesting poor model performance, but they rapidly decrease as training progresses, indicating the model is learning effectively. Around epoch 20, there is a

noticeable spike in the validation loss, showing a temporary degradation in performance on the validation set. However, after this point, both losses stabilize and continue to decrease gradually, demonstrating consistent performance with minimal over-fitting. By the end of the training process, the convergence of both losses to low values suggests the model is well-trained and generalizes effectively to new data.

The overall trend indicates successful training with the model performing well on both the training and validation datasets.

DAY	actual prices	cnn predictions
0	94.699287	98.974457
1	65.138100	65.826500
2	38.852539	40.268875
3	46.560364	46.273506
4	64.242432	65.566330
⋮	⋮	⋮
1668	91.275772	88.589073
1669	88.688652	86.948769
1670	112.065720	111.054153
1671	18.977600	19.057690
1672	46.964287	46.585861

Table 3.15: Comparison table of actual prices and predicted values using CNN
Source: By the student using Python

Encountering an issue with the `cnn_predictions` array, which is a 2D array (a matrix) and needs to be a 1D array (a vector) to be used as a column in a DataFrame. To fix this, the use of the `'numpy.flatten()'` function to flatten the 2D array into a 1D array is required.

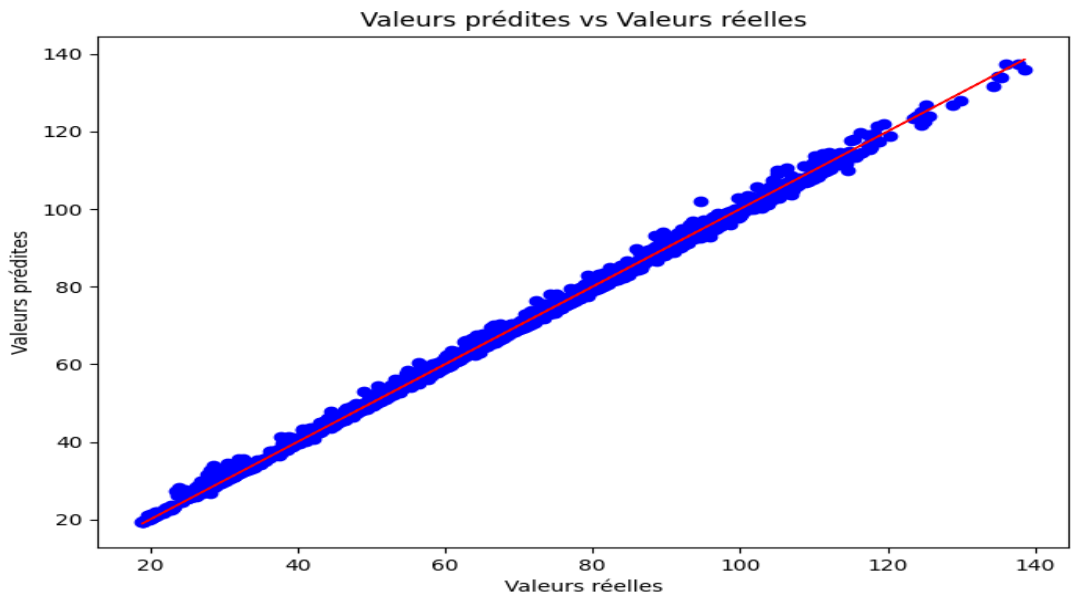


Figure 3.13: Scatter plot of actual prices vs predicted prices using CNN
Source: By the student using Python

The majority of blue dots lie very close to the red line, indicating that the predicted values are very close to the actual values which suggests a high level of accuracy in the predictions.

RMSE on test set: 1.5853550434112549

R2: 1.00

The results suggest that the difference between the model’s predictions and the actual values is approximately 1.6 USD and that the features explain only 100% of the target variable.

2.1.6. Hybrid RandomForest-XGBoost model:

A RandomForestRegressor and an XGBRegressor are trained on the training data ($\mathbf{X_train}$, $\mathbf{y_train}$).

The predictions made by these individual models on the training data are combined horizontally using `'np.column_stack()'` to create a new training set ($\mathbf{X_train_hybrid}$). A similar process is applied to the test data ($\mathbf{X_test}$) to create $\mathbf{X_test_hybrid}$. Finally, a LinearRegression model is trained on the hybrid training set ($\mathbf{X_train_hybrid}$, $\mathbf{y_train}$).

DAY	actual prices	rf-xgb predictions
2008-04-11	110.493356	107.848324
2014-06-15	107.373151	92.792676
2005-07-24	58.773857	72.265286
2007-11-25	95.159061	103.044101
2006-02-24	61.529147	59.568219
⋮	⋮	⋮
2019-04-16	71.404821	67.597092
2018-05-10	76.089681	73.166232
2000-11-20	35.238472	35.740393
2003-04-30	26.063182	39.267417
2018-04-08	76.025655	76.710758

Table 3.16: Comparison table of actual prices and predicted values using Hybrid RandomForest-XGBoost

Source: By the student using Python

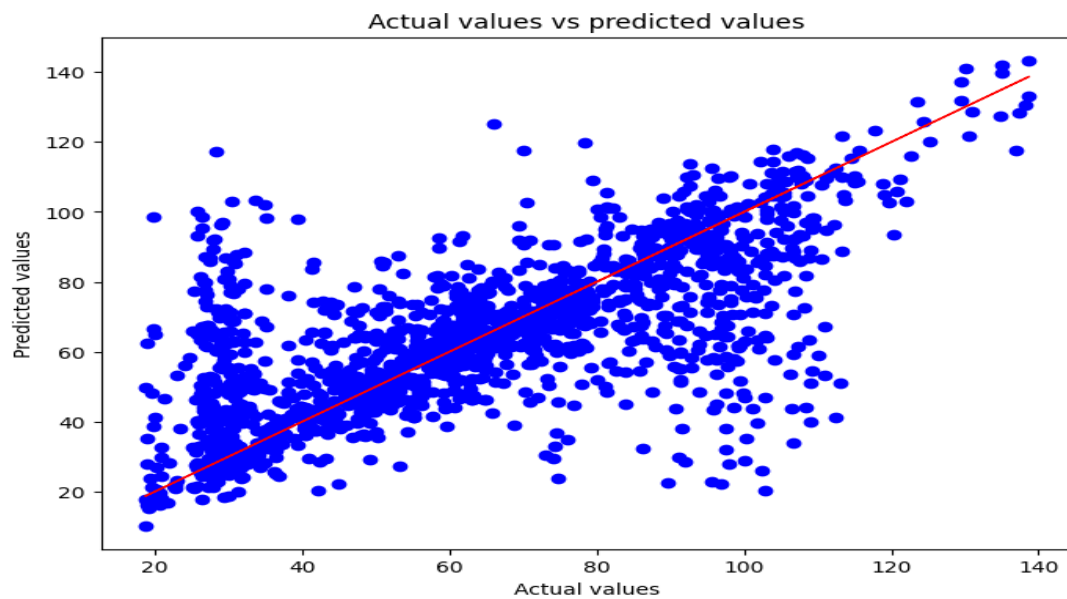


Figure 3.14: Scatter plot of actual prices vs predicted prices using hybrid RandomForest-XGBoost
Source: By the student using Python

RMSE on test set: 19.44310522690441

R2: 0.46

The results suggest that the difference between the model's predictions and the actual values is approximately 19.44 USD and that the features explain only 46% of the target variable.

2.1.7. Hybrid CNN-LSTM model:

The oil price data is likely contained in a pandas DataFrame or Series. The underlying numpy array is extracted from this DataFrame or Series. Subsequently, the array is reshaped into a 2D array with a single column. This reshaping is necessary for the Min-Max Scaler, which requires the input data to be in a 2D array format for proper scaling.

The data normalization process is initiated by importing the MinMaxScaler class from the *'sklearn.preprocessing'* module. MinMaxScaler is then initialized with a feature range of 0 to 1, ensuring that the normalized values will fall within this range. The *'fit_transform'* method is applied to the oil price data, which both fits the scaler to the data by calculating its minimum and maximum values, and transforms the data accordingly, scaling all values to the specified range of 0 to 1.

The processed data is split into training and testing sets using the *'train_test_split'* function from the *'sklearn.model_selection'* module, with 20% of the data allocated for testing and the remaining 80% for training, ensuring reproducibility with a *'random_state'* of 42.

Model: "model_1"			
Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 10, 1)]	0	[]
conv1d_5 (Conv1D)	(None, 9, 64)	192	['input_2[0][0]']
max_pooling1d_5 (MaxPooling1D)	(None, 4, 64)	0	['conv1d_5[0][0]']
lstm_63 (LSTM)	(None, 10, 50)	10400	['input_2[0][0]']
flatten_3 (Flatten)	(None, 256)	0	['max_pooling1d_5[0][0]']
lstm_64 (LSTM)	(None, 50)	20200	['lstm_63[0][0]']
concatenate_1 (Concatenate)	(None, 306)	0	['flatten_3[0][0]', 'lstm_64[0][0]']
dense_43 (Dense)	(None, 50)	15350	['concatenate_1[0][0]']
dense_44 (Dense)	(None, 1)	51	['dense_43[0][0]']
Total params: 46193 (180.44 KB)			
Trainable params: 46193 (180.44 KB)			
Non-trainable params: 0 (0.00 Byte)			

Table 3.17: Hybrid CNN-LSTM model architecture
Source: By the student using Python

This model is designed to handle sequence prediction by combining the strengths of CNN and Long LSTM networks. The input layer accepts sequences of a specified length with one feature per time step. The CNN component processes this input with a convolutional layer that uses 64 filters and a kernel size of 2 to detect local patterns, followed by max-pooling to downsample the feature map and flattening to prepare it for dense layer integration.

Simultaneously, the LSTM component processes the input through two stacked LSTM layers with 50 units each, where the first LSTM layer outputs the full sequence to the second LSTM for further processing.

The outputs from the CNN and LSTM branches are then concatenated, combining their extracted features. This combined feature set is passed through a dense layer with 50 neurons and ReLU activation to learn high-level representations, followed by an output layer with a single neuron for the final prediction. The model uses the Adam optimizer and mean squared error loss function, suitable for regression tasks, and a summary of the model provides a detailed view of its architecture and parameters.

The Hybrid CNN-LSTM model is trained on the training data X_train and corresponding labels y_train for 50 epochs. Each epoch processes 32 samples at a time. The model's performance is evaluated on the test data X_test and y_test after each epoch, and the results are printed to the console. The verbose=1 parameter ensures that the training process is displayed in the console, providing updates on the model's performance at each epoch.

DAY	actual prices	cnn-lstm predictions
0	94.962974	94.554398
1	107.981214	107.988449
2	48.530286	48.743153
3	95.626969	95.699036
4	60.090281	59.846828
⋮	⋮	⋮
1672	35.962855	36.125256
1673	61.824139	61.999016
1674	52.135276	52.329590
1675	49.358847	49.350048
1676	115.551182	115.778160

Table 3.18: Comparison table of actual prices and predicted values using Hybrid CNN-LSTM
Source: By the student using Python

The predictions are transformed back into the original scale using the scaler object, which was used to scale the data during training. Then, the training and test labels are also transformed back into their original scale for comparison with the predictions.

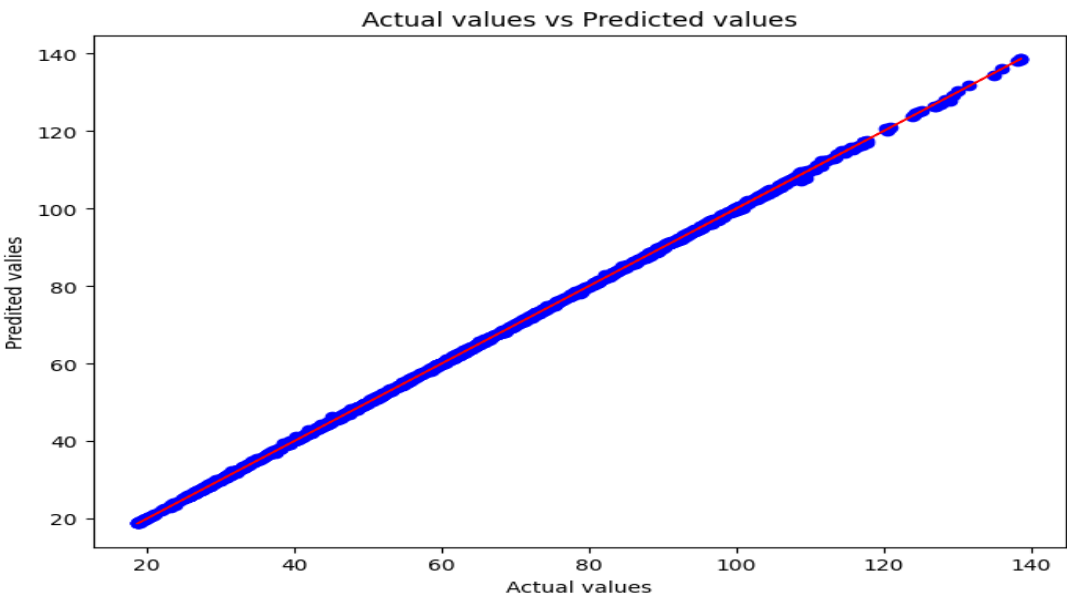


Figure 3.15: Scatter plot of actual prices vs predicted prices using Hybrid CNN-LSTM model
Source: By the student using Python

RMSE on test set: 0.21752225893275765

R²= 1.00

The results indicate that the typical difference between the model's predictions and the actual values is 0.22 USD, and the features explain 100% of the target variable.

2.2. Monthly dataset

2.2.1. SVR model:

The *GridSearchCV* function is used to find the best hyperparameters combination for the SVR model and evaluate them using 5-fold cross-validation by optimizing for the negative mean squared error.

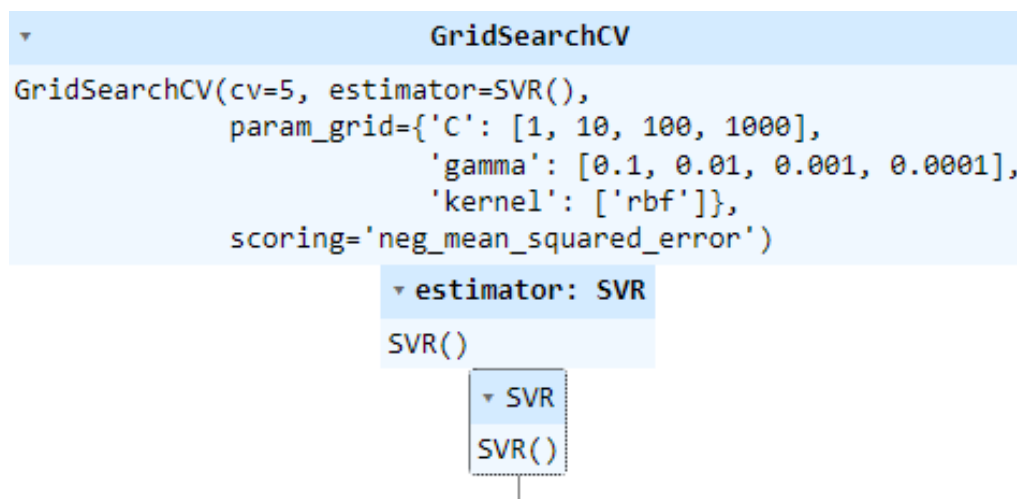


Figure 3.16: GridSearchCV node for SVR
Source: By the student using python

Best Parameters found: 'C': 10, 'gamma': 0.1, 'kernel': 'rbf'

Lowest score found: 467.5556672111899

MONTH	actual prices	SVR predictions
2002-07-01	25.790001	25.940591
2010-05-01	75.669998	73.813121
2016-05-01	47.730000	45.244139
2010-08-01	78.220001	91.083027
2018-01-01	69.930000	57.603674
2019-07-01	63.919998	61.818479
2012-01-01	111.430000	103.483559
2016-12-01	53.820000	50.903867
2021-03-01	65.760002	62.988246
2011-12-01	108.559998	106.420429
⋮	⋮	⋮
2001-08-01	25.959999	26.138550
2016-10-01	49.790001	52.604283
2014-01-01	109.959999	115.746639
2017-01-01	54.840000	50.297045
2007-03-01	64.300003	68.705367
2012-11-01	109.360001	104.027111
2013-04-01	102.639999	99.468270
2001-04-01	25.650000	28.440753
2013-03-01	108.870003	103.270676
2000-11-01	33.060001	28.947848

Table 3.19: Comparison between actual prices and predicted values using SVR
Source: By the student using Python

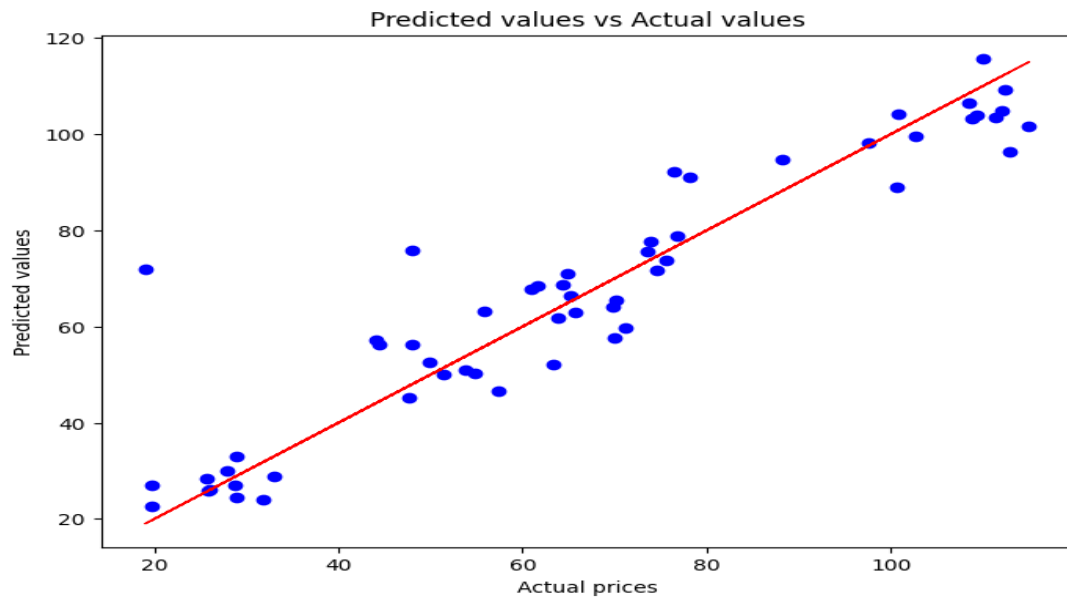


Figure 3.17: Comparison of actual prices vs predicted prices using SVR
Source: By the student using Python

RMSE: 10.64

R2: 0.86

The results indicate that the typical difference between the model's predictions and the actual values is 10.64 USD, and the features explain 86% of the target variable.

2.2.2. Random Forest model:

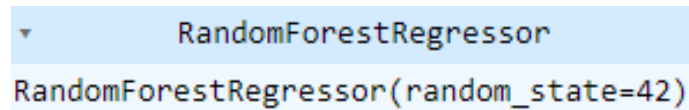


Figure 3.18: RandomForest node
Source: By the student using Python

MONTH	actual prices	RandomForest predictions
2002-07-01	25.790001	26.252701
2010-05-01	75.669998	76.912800
2016-05-01	47.730000	46.968400
2010-08-01	78.220001	88.267298
2018-01-01	69.930000	67.651401
2019-07-01	63.919998	63.347100
2012-01-01	111.430000	114.290600
2016-12-01	53.820000	47.083800
2021-03-01	65.760002	56.765501
2011-12-01	108.559998	110.380140
⋮	⋮	⋮
2001-08-01	25.959999	26.138700
2016-10-01	49.790001	46.766901
2014-01-01	109.959999	110.596098
2017-01-01	54.840000	56.356301
2007-03-01	64.300003	59.529501
2012-11-01	109.360001	109.501702
2013-04-01	102.639999	105.118580
2001-04-01	25.650000	27.216300
2013-03-01	108.870003	112.577199
2000-11-01	33.060001	34.647200

Table 3.20: Comparison table between actual prices and predicted values using Random Forest
Source: By the student using Python

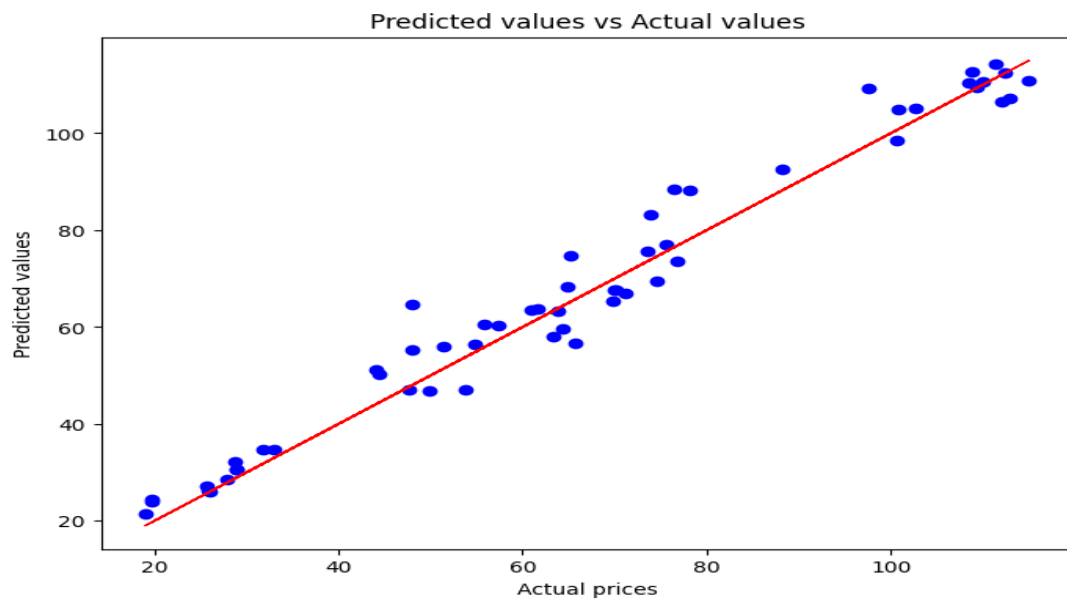


Figure 3.19: Scatter plot of actual prices vs predicted prices using RandomForest
Source: By the student using Python

RMSE on test set: 5.288514497411396

R2 = 0.97

The results indicate that the typical difference between the model's predictions and the actual values is 5.3 USD, and the features explain 97% of the target variable.

2.2.3. XGBoost model:

Using the parameters grid, the output shows that:

Best Parameters found: {'colsample_bytree': 0.7, 'learning_rate': 0.05, 'max_depth': 6, 'min_child_weight': 3, 'n_estimators': 500, 'nthread': 4, 'objective': 'reg:linear', 'reg_alpha': 0, 'reg_lambda': 0.1, 'silent': 1, 'subsample': 0.7}

Lowest Score found: 93.18370666503907

The XGBoost model must be initialized with the best parameters found in order to obtain the lowest score

MONTH	actual prices	XGBoost predictions
2002-07-01	25.790001	26.502102
2010-05-01	75.669998	80.959953
2016-05-01	47.730000	43.282009
2010-08-01	78.220001	86.731949
2018-01-01	69.930000	67.943268
2019-07-01	63.919998	64.661362
2012-01-01	111.430000	106.733650
2016-12-01	53.820000	49.741158
2021-03-01	65.760002	59.787117
2011-12-01	108.559998	111.440636
⋮	⋮	⋮
2001-08-01	25.959999	24.823780
2016-10-01	49.790001	45.509907
2014-01-01	109.959999	110.144463
2017-01-01	54.840000	56.170647
2007-03-01	64.300003	61.504082
2012-11-01	109.360001	109.126648
2013-04-01	102.639999	106.374275
2001-04-01	25.650000	28.048033
2013-03-01	108.870003	115.224854
2000-11-01	33.060001	32.656548

Table 3.21: Comparison table between actual prices and predicted values using XGBoost
Source: By the student using Python

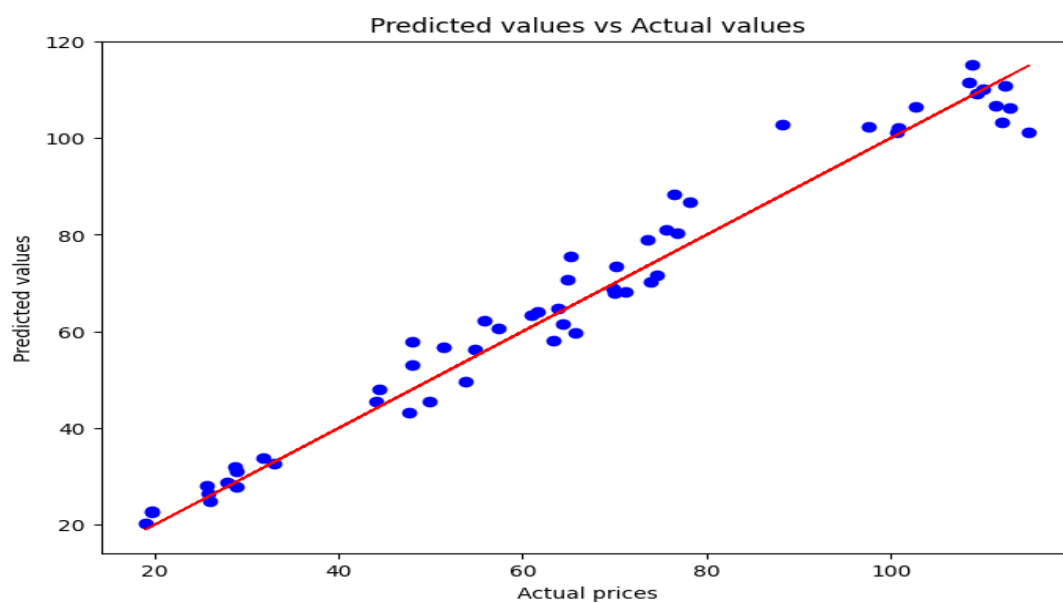


Figure 3.20: Scatter plot of actual prices vs predicted prices using XGBoost
Source: By the student using Python

RMSE on test set: 5.185967445373535

R² = 0.97

The results indicate that the typical difference between the model's predictions and the actual values is approximately 5.2 USD, and the features explain 97% of the target variable.

2.2.4. LSTM model:

The first two lines reshape the input data \mathbf{X}_{train} and \mathbf{X}_{test} to have the correct format for an LSTM model, which expects a 3D input with dimensions (*samples, time steps, features*). The LSTM model is then defined using the Keras Sequential API. It has a single LSTM layer with 50 units, which takes the reshaped 3D input. This is followed by a Dense layer with a single output unit, as this is a regression task.

MONTH	actual prices	LSTM predictions
2002-07-01	25.79	49.345692
2010-05-01	75.67	81.221527
2016-05-01	47.73	48.605762
2010-08-01	78.22	79.615944
2018-01-01	69.93	50.314857
2019-07-01	63.92	53.332748
2012-01-01	111.43	75.894691
2016-12-01	53.82	59.846378
2021-03-01	65.76	46.587234
2011-12-01	108.56	80.755470
⋮	⋮	⋮
2001-08-01	25.96	52.993580
2016-10-01	49.79	47.229893
2014-01-01	109.96	75.520668
2017-01-01	54.84	65.829224
2007-03-01	64.30	81.579224
2012-11-01	109.36	76.122276
2013-04-01	102.64	80.824951
2001-04-01	25.65	44.138321
2013-03-01	108.87	69.887230
2000-11-01	33.06	42.903893

Table 3.22: Comparison table between actual prices and predicted values using LSTM

Source: By the student using Python

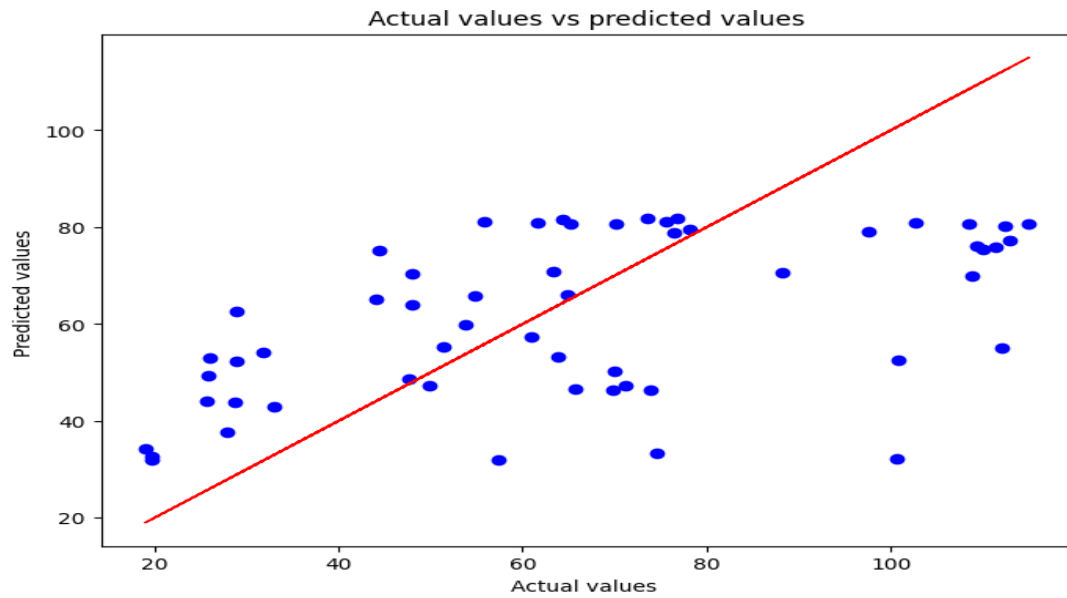


Figure 3.21: Scatter plot of actual prices vs predicted prices using LSTM
Source: By the student using Python

RMSE on test set: 24.92502374778355

R2: 0.25

The results indicate that the typical difference between the model's predictions and the actual values is 25 USD, and the features explain 25% of the target variable.

2.2.5. CNN model:

The input data for the CNN is reshaped by expanding its dimensions along the second axis. A Sequential model is defined for the CNN, and a 1D convolutional layer with 32 filters, a kernel size of 3, and ReLU activation is added. The input shape is specified as `(X_train.shape, 1)`. A Flatten layer is added to convert the feature maps into a 1D vector, followed by a Dense layer with a single unit and linear activation. The model is compiled using the Adam optimizer and Mean Squared Error loss. The model is then trained for 50 epochs with a batch size of 32, using the validation data (`X_test, y_test`). Finally, the model's performance is evaluated on the test data, and the test loss is printed.

MONTH	actual prices	CNN predictions
0	25.79	55.579205
1	75.67	67.604691
2	47.73	70.827286
3	78.22	68.739288
4	69.93	73.621117
5	63.92	73.664268
6	111.43	70.400681
7	53.82	73.401100
8	65.76	67.180168
9	108.56	70.247986
⋮	⋮	⋮
46	25.96	59.041523
47	49.79	72.090767
48	109.96	71.478241
49	54.84	73.685638
50	64.30	67.347794
51	109.36	70.665451
52	102.64	70.549316
53	25.65	57.238789
54	108.87	70.058250
55	33.06	56.820164

Table 3.23: Comparison table between actual prices and predicted values using CNN
Source: By the student using Python

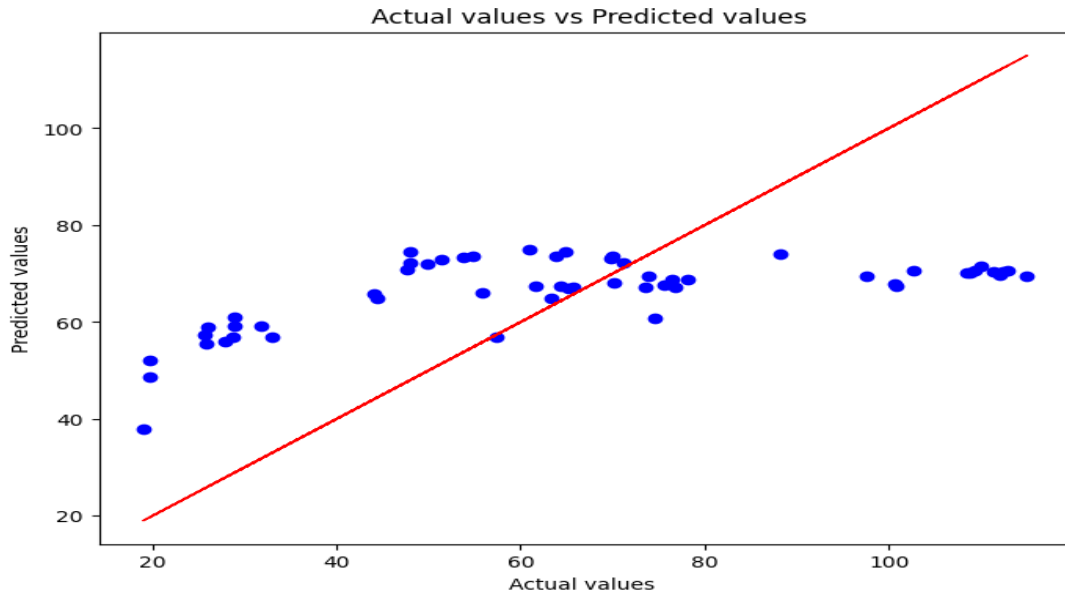


Figure 3.22: Scatter plot of actual prices vs predicted prices using CNN

RMSE on test set: 24.964631879726404

R2: 0.25

The results indicate that the typical difference between the model's predictions and the actual values is 25 USD, and the features explain 25% of the target variable

2.2.6 Hybrid RandomForest-XGBoost model:

MONTH	actual prices	CNN predictions
2002-07-01	25.79	26.842595
2010-05-01	75.67	80.065343
2016-05-01	47.73	47.480311
2010-08-01	78.22	73.808443
2018-01-01	69.93	66.250194
2019-07-01	63.92	63.296957
2012-01-01	111.43	103.919041
2016-12-01	53.82	46.836895
2021-03-01	65.76	60.450505
2011-12-01	108.56	108.524754
⋮	⋮	⋮
2001-08-01	25.96	25.036821
2016-10-01	49.79	45.568396
2014-01-01	109.96	108.673046
2017-01-01	54.84	55.712970
2007-03-01	64.30	59.564602
2012-11-01	109.36	110.651260
2013-04-01	102.64	105.685163
2001-04-01	25.65	28.302076
2013-03-01	108.87	114.343919
2000-11-01	33.06	43.598378

Table 3.24: Comparison table between actual prices and predicted values using Hybrid RandomForest-XGBoost

Source: By the student using Python

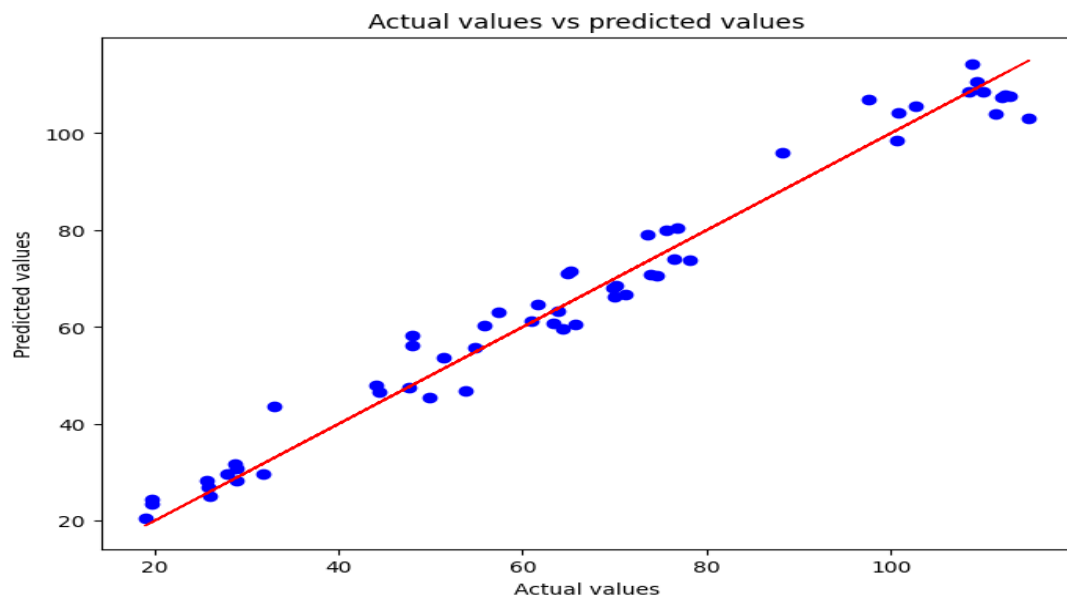


Figure 3.23: Scatter plot of actual prices vs predicted prices using Hybrid RandomForest-XGBoost

RMSE on test set: 4.771283658930129

R2: 0.97

The results indicate that the typical difference between the model's predictions and the actual values is approximately 5 USD, and the features explain 97% of the target variable

2.2.7 Hybrid CNN-LSTM model:

MONTH	actual prices	Hybrid cnn-lstm predictions
2002-07-01	25.79	15.130135
2010-05-01	75.67	15.130106
2016-05-01	47.73	15.130121
2010-08-01	78.22	15.130079
2018-01-01	69.93	15.130136
2019-07-01	63.92	15.130138
2012-01-01	111.43	15.130079
2016-12-01	53.82	15.130146
2021-03-01	65.76	15.130075
2011-12-01	108.56	15.130102
⋮	⋮	⋮
2001-08-01	25.96	15.129831
2016-10-01	49.79	15.130119
2014-01-01	109.96	15.130075
2017-01-01	54.84	15.130148
2007-03-01	64.30	15.130100
2012-11-01	109.36	15.130089
2013-04-01	102.64	15.130129
2001-04-01	25.65	15.129330
2013-03-01	108.87	15.130059
2000-11-01	33.06	15.129141

Table 3.25: Comparison table between actual prices and predicted values using Hybrid CNN-LSTM
Source: By the student using Python

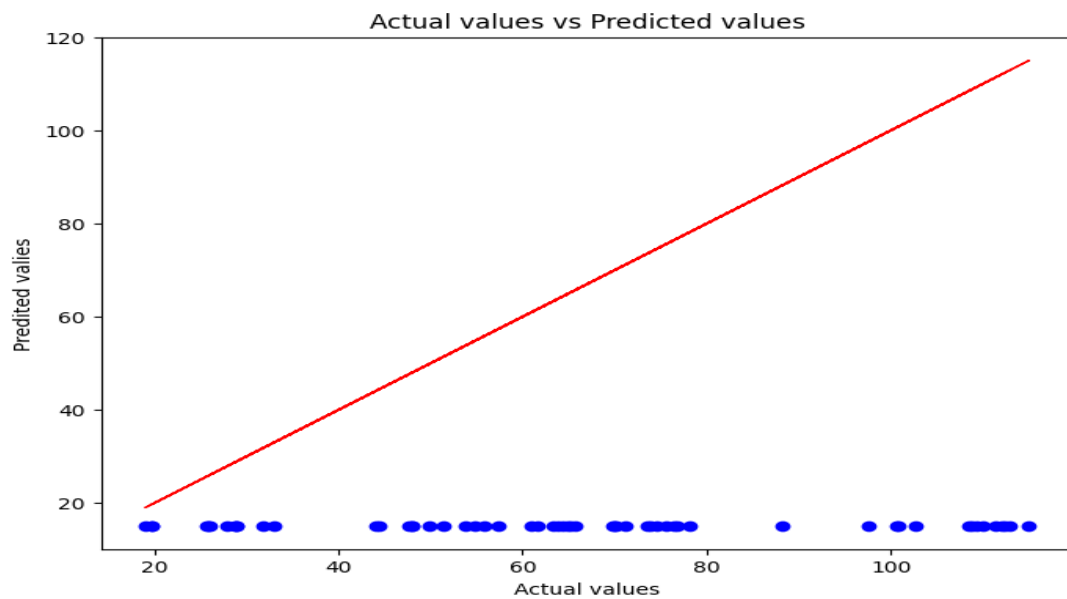


Figure 3.24: Scatter plot of actual prices vs predicted prices using Hybrid CNN-LSTM

The graph indicates that the model is not performing well in predicting the actual values, showing a tendency to underpredict consistently

RMSE on test set: 58.226776759552

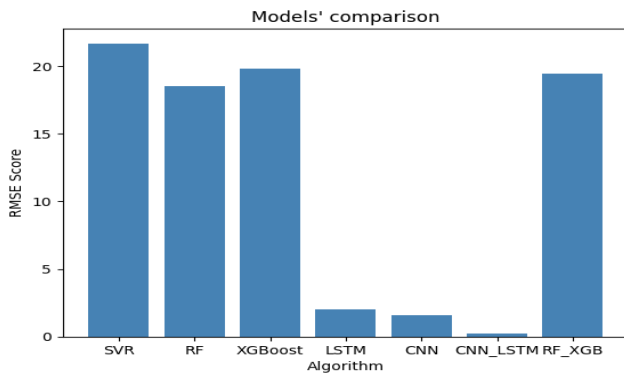
R2: 3.08

The results indicate that the typical difference between the model's predictions and the actual values is approximately 58 USD, while the R2 shows a model overfitting which is due to the use of many predictors in the model, leading to an inflated R2.

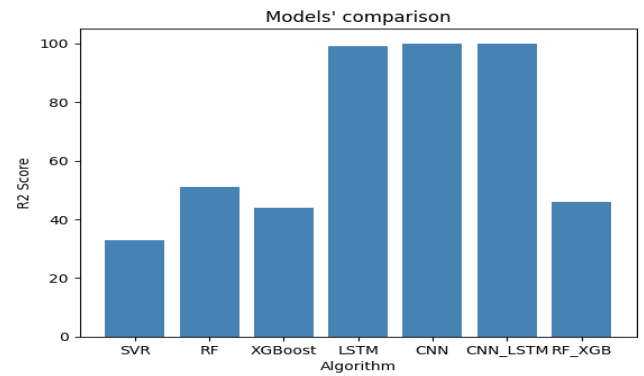
Section 3: Results and Discussion

1. Models comparison

⇒ **Daily data models' performance comparison:**



(a) Bar chart comparison of models' RMSE scores



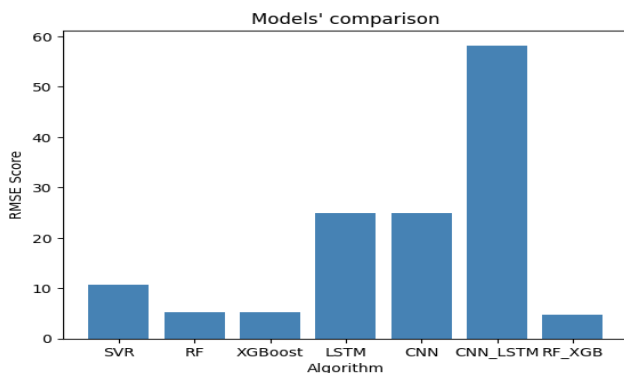
(b) Bar chart comparison of models' R2 scores

Figure 3.25: Models' performance on daily data

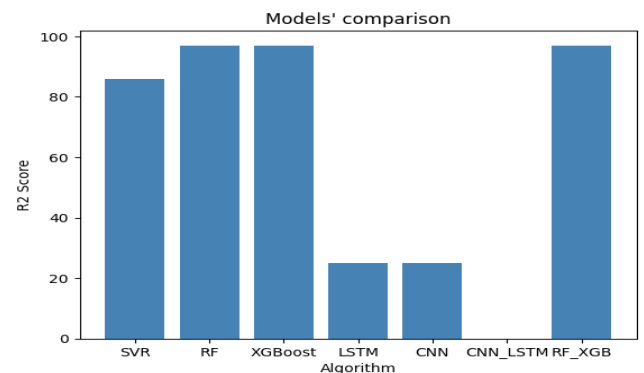
LSTM, CNN, and the CNN_LSTM combination demonstrate the highest performance with R2 scores close to a 100% accuracy and a very low RMSE, indicating excellent model accuracy with no error. The hybrid CNN_LSTM capture both temporal dependencies (via LSTM) and local patterns (via CNN), giving it an edge in scenarios with many data points and simpler feature sets allowing for less error than when using each model separately.

In contrast, SVR has the lowest R2 score, followed by XGBoost and RF_XGB, showing relatively poorer performance. The RandomForest model shows a better performance. Once again, it shows the power of combining forecasting models into hybrid models to assure a better performance.

⇒ Monthly data models' performance comparison:



(a) Bar chart comparison of models' RMSE scores



(b) Bar chart comparison of models' R2 scores

Figure 3.26: Models' performance on monthly data

The hybrid CNN_LSTM model exhibits the highest RMSE score (lowest R2), indicating the poorest performance in terms of prediction accuracy. In contrast, the RandomForest and XGBoost

models show the lowest RMSE scores, suggesting they are the most accurate among the evaluated algorithms. The SVR model also performs well, with a high R2 score.

Other models such as LSTM, and CNN demonstrate intermediate performance, with RMSE scores falling between these extremes. The combined RF_XGB model also shows a relatively low RMSE score, comparable to the individual performance of RF and XGBoost.

This comparison highlights the superior performance of ensemble methods like RF and XGBoost in reducing prediction errors, but most importantly when they are combined.

2. Data comparison

The LSTM, CNN, as well as their hybrid counterparts, have been observed to perform better on the *daily dataset* with a larger number of samples and fewer features (02). In fact, the large number of samples allows deep learning models (LSTM, CNN) to train effectively without overfitting, as deep models benefit from having more data to capture underlying patterns, when fewer features reduce the risk of overfitting, so the models can focus on learning the data's structure, making them ideal for this type of dataset.

In contrast, SVR, RandomForest, XGBoost and the hybrid RF-XGBoost, have been found to outperform the former models on the *monthly dataset* that has less samples and more features (05). In fact, these models are more suited to smaller datasets, as they do not require as much data to perform well. They also have mechanisms like regularization and ensemble learning to avoid overfitting while handling the complexity of a high-dimensional feature space.

Conclusion

The findings of this thesis provide valuable insights into the performance of various machine learning models in forecasting crude oil prices. The results demonstrate that hybrid models, which combine multiple individual algorithms, consistently outperform standalone models across all cases examined.

The performance of the machine learning models is found to be influenced by the characteristics of the dataset used for training and evaluation. Long Short-Term Memory and Convolutional Neural Network models exhibit superior forecasting accuracy when applied to datasets with larger samples size and fewer features. In contrast, Support Vector Regression, Random Forest, and XGBoost models perform better on datasets with fewer time frames but a larger number of input features.

These results highlight the importance of considering the specific characteristics of the data when selecting and optimizing machine learning models for crude oil price forecasting. For datasets with more samples and fewer features, deep learning models like LSTM and CNN tend to excel. However, for datasets with fewer samples but more features, traditional machine learning models such as SVR, RandomForest and XGBoost are more effective due to their ability to generalize better and avoid overfitting in scenarios with limited data.

Also, the superior performance of hybrid models suggests that leveraging the strengths of multiple algorithms can lead to improved forecasting accuracy compared to individual models.

Additionally, incorporating other factors like environmental variables, macroeconomic indicators, and foreign market behavior into the forecasting models could provide a more comprehensive representation of the multifaceted drivers influencing crude oil prices.

The purpose of this thesis was to investigate the performance of various machine learning models in forecasting crude oil prices, but also the suitable data characteristics for each algorithm. This research allowed us to answer the questions asked previously:

- ✓ The hybrid machine learning models outperform individual models in forecasting crude oil prices
- ✓ The use of datasets with larger sample size can enhance the accuracy of traditional models, but not for the neural network algorithms such as LSTM and CNN which tend to perform better of larger datasets.
- ✓ The number of input features impact the forecasting accuracy depending on which model is trained. The traditional models such as SVR, RandomForest and XGBoost tend to perform better on datasets with a larger number of input features, in contrast of neural network algorithms.

In conclusion, this thesis contributes to the growing body of knowledge on the application of machine learning techniques in crude oil price forecasting. The findings underscore the potential of

hybrid models and the significance of aligning model selection with data characteristics to enhance forecasting accuracy. The insights gained from this research can guide stakeholders in making informed decisions and developing effective investment strategies in the oil market.

Bibliography

- [eia,] Algeria. Technical report. Accessed on 2024-05-20.
- [alg,] Algeriaoil. Technical report. Accessed on 2024-05-20.
- [Ass,] Chapter 5: Association analysis: Basic concepts and algorithms.
- [Abdellaoui, 2022] Abdellaoui, S. (2022). The algerian economy between oil dependence and the inevitability of economic diversification. *International Journal of Economic Performance*, 5(2). Accessed on April 05, 2024.
- [Abdullah and Zeng, 2010] Abdullah, S. N. and Zeng, X. (2010). Machine learning approach for crude oil price prediction with artificial neural networks-quantitative (ann-q) model. In *Proceedings of the International Joint Conference on Neural Networks/Proc Int Jt Conf Neural Networks*, pages 1–8. IEEE.
- [Aboelkheir, 2022] Aboelkheir, I. M. M. (2022). Who are nocs and iocs and what their public role is in the international gas industry and oil? *Cognizance Journal of Multidisciplinary Studies*, 2:37–49.
- [Aizenberg et al., 2000] Aizenberg, I. N., Aizenberg, N. N., and Vandewalle, J. (2000). *Learning Algorithms*, pages 139–167. Springer US, Boston, MA.
- [Al-Fattah, 2014] Al-Fattah, S. (2014). National oil companies: Business models, challenges, and emerging trends. *Alternative Investment Analyst Review Journal*, 2:11–28.
- [Aldabagh et al., 2023] Aldabagh, H., Zheng, X., and Mukkamala, R. (2023). A hybrid deep learning approach for crude oil price prediction. *Journal of Risk and Financial Management*, 16(12).
- [Aljumily, 2016] Aljumily, R. (2016). Agglomerative hierarchical clustering: An introduction to essentials. (3) standardization, normalization, and dimensionality reduction of a data matrix. *Global Journal of Human Social Science*, 16.
- [Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99*, page 49–60, New York, NY, USA. Association for Computing Machinery.
- [Awad and Khanna, 2015] Awad, M. and Khanna, R. (2015). *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. ApressOpen.

- [Awan, 2019] Awan, O. A. (2019). Price discovery or noise: The role of arbitrage and speculation in explaining crude oil price behaviour. *Journal of Commodity Markets*, 16:100086.
- [Bakirtas and Akpolat, 2020] Bakirtas, T. and Akpolat, A. G. (2020). The relationship between crude oil exports, crude oil prices and military expenditures in some opec countries. *Resources Policy*. academia.edu.
- [Bank, 2023] Bank, W. (2023). The world bank in algeria. Accessed on 2024-04-11.
- [Baum and Gibbons, 2022] Baum, A. and Gibbons, G. (2022). Three ways to speed up xgboost model training. Accessed on May 12, 2024.
- [Belyadi and Haghighat, 2021] Belyadi, H. and Haghighat, A. (2021). *Chapter 5 - Supervised learning*, pages 169–295. Gulf Professional Publishing.
- [Biswas et al., 2018] Biswas, N., Chakraborty, S., Mullick, S., and Das, S. (2018). A parameter independent fuzzy weighted k -nearest neighbor classifier. *Pattern Recognition Letters*, 101:80–87.
- [Blokhina et al., 2016] Blokhina, T., Karpenko, O., and Andrey, G. (2016). The relationship between oil prices and exchange rate in russia. *International Journal of Energy Economics and Policy*, 6:721–726.
- [Bornstein et al., 2023] Bornstein, G., Krusell, P., and Rebelo, S. (2023). A world equilibrium model of the oil market. *The Review of Economic Studies*, 2023. nber.org.
- [Bouoiyour et al., 2014] Bouoiyour, J., Selmi, R., and Shahbaz, M. (2014). The nexus between oil price and russia’s real exchange rate: Better paths via unconditional vs conditional analysis.
- [Breitenfellner et al., 2009] Breitenfellner, A., Cuaresma, J. C., and Keppel, C. (2009). Determinants of Crude Oil Prices: Supply, Demand, Cartel or Speculation? *Monetary Policy & the Economy*, pages 111–136.
- [Bret-Rouzaut and Favennec, 2011] Bret-Rouzaut, N. and Favennec, J.-P. (2011). *Oil and Gas Exploration and Production: Reserves, Costs, Contracts*. TECHNIP.
- [Cao, 2017] Cao, L. (2017). Data science: A comprehensive overview. *ACM Computing Surveys*, 50:1–42.
- [Carbonell et al., 1983] Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. *Machine Learning*, pages 3–23.
- [Carlsson, 2023] Carlsson, A. (2023). Predictive regression model evaluation: Evaluating predictive machine learning models to reduce food waste in the dairy industry. Master’s thesis, School of Electrical Engineering and Computer Science, Stockholm.
- [Chanderli and Zaimeche, 2024] Chanderli, A. and Zaimeche, S. (2024). Economy of algeria. Technical report. Accessed on 2024-05-12.
- [Chen et al., 2016] Chen, H., Liu, L., Wang, Y., and Zhu, Y. (2016). Oil price shocks and u.s. dollar exchange rates. *Energy*, 112:1036–1048.
- [Chen et al., 2024] Chen, Q., Xue, B., Browne, W., and Zhang, M. (2024). *Evolutionary Regression and Modelling*, pages 121–149. Springer Nature Singapore, Singapore.

- [Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pages 785–794. ACM.
- [Chipanda, 2023] Chipanda, B. (2023). A sustainable analysis of algeria. *ISS African Futures*. Accessed on April 04, 2024.
- [CIA, 1970] CIA (1970). Nationalization of esso and mobil facilities in algeria. Reference 18.bd2d2d17.1712956237.6feb44e.
- [Coenen, 2011] Coenen, F. (2011). Data mining: Past, present and future. *Knowledge Eng. Review*, 26:25–29.
- [Cologni et al., 2015] Cologni, A., Scarpa, E., and Sitzia, F. G. (2015). Big Fish: Oil Markets and Speculation. Energy: Resources and Markets 206220, Fondazione Eni Enrico Mattei (FEEM). Accessed on 2024-04-22.
- [Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.
- [Dalianis, 2018] Dalianis, H. (2018). *Evaluation Metrics and Evaluation*, pages 45–53.
- [Dechter, 1986] Dechter, R. (1986). Learning while searching in constraint-satisfaction-problems. volume 01, pages 178–185.
- [Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39:1–22.
- [Desorgues and AFP, 2021] Desorgues, P. and AFP (2021). Coronavirus, chute des prix du pétrole : l’algérie dans une situation de "vulnérabilité économique. *TV5 Monde*. Accessed on April 04, 2024.
- [Dojo, 2020] Dojo, N. T. (2020). 3 types of machine learning. Accessed on June 22, 2024.
- [Džeroski, 2008] Džeroski, S. (2008). *Data Mining*, pages 821–830.
- [Feizi et al., 2022] Feizi, R., Ahmadzadeh, K., and Javaheri, B. (2022). The impact of exchange rate fluctuations and the oil price shocks on government budget: Cge model approach. *Iranian Economic Review*, 26(2):343–368.
- [Fernandois and Medel, 2020] Fernandois, A. and Medel, C. A. (2020). Geopolitical tensions, opec news, and the oil price: A Granger causality analysis. *Revista de análise econ text three superiormico*, 35:57 – 90.
- [Ferrari et al., 2023] Ferrari, M., Lappe, M.-S., and Röbber, D. (2023). Geopolitical risk and oil prices. Technical report. Accessed on 2024-04-20.
- [Frawley et al., 1992] Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):1–27.

- [Gallo et al., 2010] Gallo, A., Mason, P., Shapiro, S., and Fabritius, M. (2010). What is behind the increase in oil prices? Analyzing oil consumption and supply relationship with oil price. *Energy*, 35(10):4126–4141.
- [GeeksforGeeks, 2023] GeeksforGeeks (2023). Types of machine learning. Accessed on June 22, 2024.
- [GitHub, 2023] GitHub (2023). Lecture 14 - ensemble methods. Accessed on May 12, 2024.
- [Gomez and Schmidhuber, 2005] Gomez, F. and Schmidhuber, J. (2005). Co-evolving recurrent neurons learn deep memory pomdps.
- [Gränitz, 2020] Gränitz, M. (2020). Long-term investment trends: the crude oil boom in the 2000s. Technical report. Accessed on 2024-05-16.
- [Guechari, 2017] Guechari, Y. (2017). *The effect of oil price shocks on the Algerian economy*. PhD thesis, Mohamed Khider Biskra University.
- [Guillemoles, 2019] Guillemoles, A. (2019). La sonatrach, un État dans l’État algérien. Technical report.
- [Hamzaoui, 2020a] Hamzaoui, A. (2020a). The impact of oil revenues on the path of reforms in algeria: Whom to blame government or people? *Algerian Journal of Political Economy*, 02.
- [Hamzaoui, 2020b] Hamzaoui, A. (2020b). The impact of oil revenues on the path of reforms in algeria: Whom to blame government or people? *Algerian Journal of Political Economy*, 2(1):01–2020.
- [Handl et al., 2024] Handl, J., Garza-Fabre, M., and José-García, A. (2024). *Evolutionary Clustering and Community Detection*, pages 151–169. Springer Nature Singapore, Singapore.
- [Haouas et al., 2021] Haouas, A., Ochi, A., and Labidi, M. A. (2021). Sources of algeria’s economic growth, 1979–2019: Augmented growth accounting framework and growth regression method. *Review of Social Economy*, 79(3):445–472.
- [Harrison, 2023] Harrison, M. (2023). *Effective XGBoost: Optimizing, Tuning, Understanding, and Deploying*. MetaSnake, United States.
- [Hartigan and Wong, 1979] Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society*, 28:100–108.
- [Hossin and M.N, 2015] Hossin, M. and M.N, S. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining and Knowledge Management Process*, 5:01–11.
- [Huang et al., 2011] Huang, J., Hu, X., and Yang, F. (2011). Support vector machine with genetic algorithm for machinery fault diagnosis of high voltage circuit breaker. *Measurement*, 44:1018–1027.
- [IEA, 2016] IEA (2016). Oil and natural gas basins and pipeline infrastructure in algeria. Technical report, U.S. Energy Information Administration. Accessed on 2024-06-22.

- [IEA, 2023] IEA (2023). Oil market report - december 2023.
- [IEA, 2024] IEA (2024). Oil market report - april 2024. Technical report, Paris. Accessed on 2024-04-30.
- [Jiang et al., 2023a] Jiang, D., Zhang, C., and Song, Y. (2023a). *Expectation Maximization*, pages 53–62.
- [Jiang et al., 2023b] Jiang, D., Zhang, C., and Song, Y. (2023b). *Probabilistic Topic Models: Foundation and Application*. Springer Nature Singapore.
- [Jin and Han, 2010] Jin, X. and Han, J. (2010). *K-Medoids Clustering*, pages 564–565. Springer US, Boston, MA.
- [Kamen-Kaye, 1958] Kamen-Kaye, M. (1958). Petroleum development in algeria. *Geographical Review*, 4(4):463–473.
- [Kaufmann and Rousseeuw, 1987a] Kaufmann, L. and Rousseeuw, P. (1987a). Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416.
- [Kaufmann and Rousseeuw, 1987b] Kaufmann, L. and Rousseeuw, P. (1987b). Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416.
- [Kilian and Vigfusson, 2011] Kilian, L. and Vigfusson, R. J. (2011). Are the responses of the u.s. economy asymmetric in energy price increases and decreases? *Quantitative Economics*, 2:419–453.
- [Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59–69.
- [Kohonen, 1988] Kohonen, T. (1988). An introduction to neural computing. *Neural Networks*, 1:3–16.
- [Kohonen, 1995] Kohonen, T. (1995). *Learning Vector Quantization*, pages 175–189. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Kohonen, 2001] Kohonen, T. (2001). *Learning Vector Quantization*, pages 245–261. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Kotu and Deshpande, 2019] Kotu, V. and Deshpande, B. (2019). *Chapter 6 - Association Analysis*, pages 199–220. Morgan Kaufmann, second edition edition.
- [Kubat, 2021] Kubat, M. (2021). *Unsupervised Learning*, pages 297–325. Springer International Publishing, Cham.
- [Laframboise et al., 1998a] Laframboise, N., Alonso-Gamo, P., Feler, A., Bazzoni, S., Nashashibi, K. A., and Horvitz, S. (1998a). *Algeria: Stabilization and Transition to Market*. International Monetary Fund, USA.
- [Laframboise et al., 1998b] Laframboise, N., Alonso-Gamo, P., Feler, A., Bazzoni, S., Nashashibi, K. A., and Horvitz, S. P. (1998b). *II The Setting of Economic Reform*, chapter 2. International Monetary Fund, USA.

- [Lemberger et al., 2015] Lemberger, P., Batty, M., Morel, M., Delattre, M., and Raffaëlli, J. (2015). *Big data et machine learning: manuel du data scientist*. InfoPro. Management des systèmes d’information. Dunod.
- [Lemberger et al., 2023] Lemberger, P., Batty, M., Morel, M., and Raffaëlli, J.-L. (2023). *Big Data et Machine Learning: les concepts et les outils de la data science*. Dunod, 3rd edition.
- [Li et al., 2020] Li, B., Chang, C.-P., Chu, Y., and Sui, B. (2020). Oil prices and geopolitical risks: What implications are offered via multi-domain investigations? *Energy and Environment*, 31:492–516.
- [Li, 2024a] Li, H. (2024a). *Decision Tree*, pages 77–102. Springer Nature Singapore, Singapore.
- [Li, 2024b] Li, H. (2024b). *EM Algorithm and Its Extensions*, pages 201–220. Springer Nature Singapore, Singapore.
- [Li, 2024c] Li, H. (2024c). *Introduction to Machine Learning and Supervised Learning*, pages 1–37. Springer Nature Singapore, Singapore.
- [Li, 2024d] Li, H. (2024d). *Introduction to Unsupervised Learning*, pages 281–292. Springer Nature Singapore, Singapore.
- [Li, 2024e] Li, H. (2024e). *K-Nearest Neighbor*, pages 55–66. Springer Nature Singapore, Singapore.
- [Li, 2024f] Li, H. (2024f). *Logistic Regression and Maximum Entropy Model*, pages 103–125. Springer Nature Singapore, Singapore.
- [Li, 2024g] Li, H. (2024g). *The Naïve Bayes Method*, pages 67–75. Springer Nature Singapore, Singapore.
- [Li, 2024h] Li, H. (2024h). *Support Vector Machine*, pages 127–177. Springer Nature Singapore, Singapore.
- [Lizardo and Mollick, 2010] Lizardo, R. A. and Mollick, A. (2010). Oil price fluctuations and u.s. dollar exchange rates. *Energy Economics*, 32(2):399–408.
- [MacFadyen and Watkins, 2014] MacFadyen, A. J. and Watkins, G. C. (2014). *Petroleum and the Petroleum Industry: What Are They?*, pages 3–18. University of Calgary Press.
- [Malti, 2012] Malti, H. (2012). *Histoire secrète du pétrole algérien*. La Découverte.
- [McClay, Rebecca, 2022] McClay, Rebecca (2022). Unraveling the oil and gas industry: A comprehensive guide to upstream, midstream, and downstream segments. *Investopedia*. Accessed on 2024-04-21.
- [Mignon and Saadaoui, 2023] Mignon, V. and Saadaoui, J. (2023). How Political Tensions and Geopolitical Risks Impact Oil Prices? Working Papers of BETA 2023-15, Bureau d’Economie Théorique et Appliquée, UDS, Strasbourg. Accessed on 2024-04-20.
- [Miikkulainen, 2024a] Miikkulainen, R. (2024a). *Evolutionary Supervised Machine Learning*, pages 29–57. Springer Nature Singapore, Singapore.

- [Miikkulainen, 2024b] Miikkulainen, R. (2024b). *Evolutionary Supervised Machine Learning*, pages 29–57. Springer Nature Singapore, Singapore.
- [Mitchell, 2005] Mitchell, T. (2005). *Generative and discriminative classifiers: Naive Bayes and logistic regression*.
- [Mitchell, 1997] Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.
- [Monge et al., 2016a] Monge, M., Gil-Alana, L., and Perez De Gracia, F. (2016a). Crude oil price behaviour before and after military conflicts and geopolitical events. *Energy*, 120.
- [Monge et al., 2016b] Monge, M., Gil-Alana, L., and Perez De Gracia, F. (2016b). Crude oil price behaviour before and after military conflicts and geopolitical events. *Energy*, 120.
- [Mounecif, 2018] Mounecif, R. A. (2018). *La prudence et la prise de décision : la Compagnie Française des Pétroles Algérie et l’algérianisation du personnel (1962-1971)*, volume 92, pages 43–58. ESKA.
- [Musa, 2023] Musa, A. (2023). Review paper revolutionizing oil and gas industries with artificial intelligence technology. *International Journal of Computer Sciences and Engineering*. Accessed on 2024-06-22.
- [Nakhle, 2022] Nakhle, C. (2022). Peak oil demand will change global market dynamics. Technical report. Accessed on 2024-05-01.
- [Naur, 1968] Naur, P. (1968). ‘datalogy’, the science of data and data processes. In *IFIP Congress*.
- [Naur, 1974] Naur, P. (1974). *Concise Survey of Computer Methods*. Petrocelli Books.
- [Nguyen et al., 2024] Nguyen, B., Xue, B., Zhang, M., and Browne, W. (2024). *Evolutionary Classification*, pages 171–204.
- [of Commerce,] of Commerce, U. D. Algeria - oil and gas - hydrocarbons. Technical report.
- [of Commerce, 2023] of Commerce, U. D. (2023). Algeria - oil and gas - hydrocarbons. Accessed on May 05, 2024.
- [OilPrice.com,] OilPrice.com. Crude oil prices today | oilprice.com. Technical report. Accessed on 2024-06-20.
- [Olcott, 2022] Olcott, M. B. (2022). The changing role of national oil companies in international energy markets. *Carnegie Endowment for International Peace*.
- [OPEC, 2023] OPEC (2023). Algeria.
- [OPEC, 2024] OPEC (2024). Algeria. Technical report. Accessed on 2024-06-21.
- [OPEC, 2024] OPEC (2024). Opec monthly oil report: May 2024. Technical report, OPEC. Accessed on 2024-06-22.
- [Otokiti, 2020] Otokiti, A. (2020). *Digital Health and Healthcare Quality: A Primer on the Evolving 4th Industrial Revolution*. Intech open.

- [Raymond and Leffler, 2017] Raymond, M. S. and Leffler, W. L. (2017). *Oil & Gas Production in Nontechnical Language*. PennWell, Tulsa, OK, 2nd edition.
- [Ros and Riad, 2023] Ros, F. and Riad, R. (2023). *Feature and Dimensionality Reduction for Clustering with Deep Learning*. Unsupervised and Semi-Supervised Learning. Springer Nature Switzerland.
- [Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.
- [Santana, 2024a] Santana, R. (2024a). *EML for Unsupervised Learning*, pages 59–78. Springer Nature Singapore, Singapore.
- [Santana, 2024b] Santana, R. (2024b). *EML for Unsupervised Learning*, pages 59–78. Springer Nature Singapore, Singapore.
- [Sarang, 2023a] Sarang, P. (2023a). chapter Centroid-Based Clustering. The Springer Series in Applied Machine Learning. Springer, Cham.
- [Sarang, 2023b] Sarang, P. (2023b). *Artificial Neural Networks*, pages 261–287. Springer International Publishing, Cham.
- [Sarang, 2023c] Sarang, P. (2023c). *Centroid-Based Clustering*, pages 171–183. Springer International Publishing, Cham.
- [Sarang, 2023d] Sarang, P. (2023d). *Clustering: Overview*, page 181. why k-medoids.
- [Sarang, 2023e] Sarang, P. (2023e). *Connectivity-Based Clustering*, pages 185–195. Springer International Publishing, Cham.
- [Sarang, 2023f] Sarang, P. (2023f). *Decision Tree*, pages 75–96. Springer International Publishing, Cham.
- [Sarang, 2023g] Sarang, P. (2023g). *Density-Based Clustering*, pages 209–228. Springer International Publishing, Cham.
- [Sarang, 2023h] Sarang, P. (2023h). *Dimensionality Reduction*, pages 19–52. Springer International Publishing, Cham.
- [Sarang, 2023i] Sarang, P. (2023i). *Gaussian Mixture Model*, pages 197–207. Springer International Publishing, Cham.
- [Sarang, 2023j] Sarang, P. (2023j). *K-Nearest Neighbors*, pages 131–141. Springer International Publishing, Cham.
- [Sarang, 2023k] Sarang, P. (2023k). *Naive Bayes*, pages 143–152. Springer International Publishing, Cham.
- [Sarang, 2023l] Sarang, P. (2023l). *Regression Analysis*, pages 55–73. Springer International Publishing, Cham.
- [Sarang, 2023m] Sarang, P. (2023m). *Support Vector Machines*, pages 153–165. Springer International Publishing, Cham.

- [Sasirekha and Baby, 2013] Sasirekha, K. and Baby, P. (2013). Agglomerative hierarchical clustering algorithm- a review. *International Journal of Scientific and Research Publications*, 3.
- [Takahashi and Takahashi, 2024a] Takahashi, K. and Takahashi, L. (2024a). *Supervised Machine Learning*, pages 191–226. Springer Nature Singapore, Singapore.
- [Takahashi and Takahashi, 2024b] Takahashi, K. and Takahashi, L. (2024b). *Unsupervised Machine Learning and Beyond Machine Learning*, pages 227–244. Springer Nature Singapore, Singapore.
- [Takroosta et al., 2019] Takroosta, A., Mohajeri, P., Mohammadi, T., Shakeri, A., and Ghasemi, A. a. (2019). An analysis of oil prices considering the political risk of opec. *Journal of Economic Modeling Research*, 10(37).
- [Times,] Times, T. N. Y. Algeria and france settle bitter dispute on oil. *The New York Times*.
- [Turgeon and Morse, 2023] Turgeon, A. and Morse, E. (2023). Petroleum. Technical report. Accessed on 2024-04-18.
- [Tusiani and Shearer, 2007] Tusiani, M. D. and Shearer, G. (2007). *LNG: A Nontechnical Guide*. PennWell, Tulsa, OK.
- [U.S. Commercial Service, 2024] U.S. Commercial Service (2024). Algeria - oil and gas - hydrocarbons. Technical report. Accessed on 2024-06-20.
- [Waller and Fawcett, 2013] Waller, M. and Fawcett, S. (2013). Data science, predictive analytics, and big data: A revolution that will transform supply chain design and management. *Journal of Business Logistics*, 34.
- [Wang and Sun, 2017] Wang, Q. and Sun, X. (2017). Crude oil price: Demand, supply, economic activity, economic policy uncertainty and wars – from the perspective of structural equation modelling (sem). *Energy*, 133:483–490.
- [XGBoost, 2024] XGBoost (2024). Xgboost documentation. Accessed: 2024-09-22.
- [Yan et al., 2021a] Yan, C., Huang, Z., Sanjuán Martínez, O., Fenza, G., and Gonzalez Crespo, R. (2021a). Monetary factors of commodity prices using fuzzy binomial approach. *J. Intell. Fuzzy Syst.*, 40(4):8345–8357.
- [Yan et al., 2021b] Yan, L., Zhu, Y., and Wang, H. (2021b). Selection of machine learning models for oil price forecasting: Based on the dual attributes of oil. *Discrete Dynamics in Nature and Society*, 2021(1):1566093.
- [Yergin, 1991] Yergin, D. (1991). *The Prize: The Epic Quest for Oil, Money & Power*. Simon & Schuster, New York.
- [Yergin, 2011] Yergin, D. (2011). *The Prize: The Epic Quest for Oil, Money & Power*. Free Press.
- [Yoshino and Alekhina, 2019] Yoshino, N. and Alekhina, V. (2019). Empirical Analysis of Global Oil Price Determinants at the Disaggregated Level Over the Last Two Decades. ADBI Working Papers 982, Asian Development Bank Institute.

- [Zhou, 2021a] Zhou, Z.-H. (2021a). *Clustering*, pages 211–240. Springer Singapore, Singapore.
- [Zhou, 2021b] Zhou, Z.-H. (2021b). *Decision Trees*, pages 79–102. Springer Singapore, Singapore.
- [Zhou, 2021c] Zhou, Z.-H. (2021c). *Dimensionality Reduction and Metric Learning*, pages 241–264. Springer Singapore, Singapore.
- [Zhou, 2021d] Zhou, Z.-H. (2021d). *Machine Learning*. Springer Singapore, 1 edition. Chapter 1: Brief History.

Abstract

Algeria's economy is heavily dependent on revenues from the hydrocarbon sector, which accounted for 19% of GDP, 93% of exports, and 75% of budget revenues between 2016-2021. This reliance on oil and gas makes Algeria vulnerable to price fluctuations in global energy markets. The research systematically evaluated the performance of different machine learning models in forecasting crude oil prices, assessing the impact of dataset characteristics on model effectiveness and also the combination of the models into a hybrid model. The results demonstrate that hybrid machine learning models outperform standalone models, but model performance is influenced by dataset characteristics. Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models exhibit superior forecasting accuracy on datasets with larger dataframes and fewer explanatory variables, while Support Vector Regression (SVR), Random Forest, and XGBoost models perform better on datasets with fewer time frames but a larger number of input features. These insights can guide stakeholders in making informed decisions and developing effective investment strategies in the oil market, which is critical for Algeria given its heavy reliance on hydrocarbon revenues.

Keywords—

Hybrid model, LSTM, CNN, SVR, XGBoost, RandomForest, Crude oil, Petroleum, Oil industry, Prediction

Résumé

L'économie algérienne dépend fortement des revenus du secteur des hydrocarbures, qui ont représenté 19% du PIB, 93% des exportations et 75% des recettes budgétaires entre 2016 et 2021. Cette dépendance au pétrole et au gaz rend l'Algérie vulnérable aux fluctuations des prix sur les marchés mondiaux de l'énergie. La recherche a systématiquement évalué la performance de différents modèles d'apprentissage automatique dans la prévision des prix du pétrole brut, en évaluant l'impact des caractéristiques de l'ensemble des données sur l'efficacité du modèle et aussi la combinaison des modèles dans un modèle hybride. Les résultats démontrent que les modèles hybrides d'apprentissage automatique sont plus performants que les modèles autonomes, mais que la performance du modèle est influencée par les caractéristiques de l'ensemble des données. Les modèles de mémoire à long terme (LSTM) et de réseau neuronal convolutif (CNN) affichent une précision de prévision supérieure sur les ensembles de données comportant des périodes plus longues et moins de variables explicatives, tandis que les modèles de régression vectorielle de soutien (SVR), de forêt aléatoire et de XGBoost sont plus performants sur les ensembles de données comportant moins de périodes, mais un plus grand nombre de caractéristiques d'entrée. Ces informations peuvent aider les parties prenantes à prendre des décisions éclairées et à élaborer des stratégies d'investissement efficaces sur le marché pétrolier, ce qui est essentiel pour l'Algérie compte tenu de sa forte dépendance à l'égard des revenus des hydrocarbures.

Mots clés— Modèle hybride, LSTM, CNN, SVR, XGBoost, Forêts aléatoires, Pétrole, Industrie pétrolière, Prédiction

Appendix A

Daily data

[Direct link to the Python syntax on google colab](#)

A.1 The SVR model

```
# Define the parameter grid
param_grid = {
    'C': [1, 10, 100, 1000],
    'gamma': [0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf']
}
```

```
svr=SVR()
```

```
# Perform grid search with MSE as the scoring metric
grid_search_svr = GridSearchCV(svr, param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search_svr.fit(X_train, y_train)
```

```
> GridSearchCV
> estimator: SVR
  > SVR
```

```
# Get the best parameters and the corresponding model
best_params_svr = grid_search_svr.best_params_
best_model_svr = grid_search_svr.best_estimator_
```

```
# Print best parameters and best score
print("Best Parameters found: ", grid_search_svr.best_params_)
print("Lowest score found: ", -grid_search_svr.best_score_)

Best Parameters found: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
Lowest score found: 467.5556672111899
```

```
# Make predictions using the best model
svr_predictions = best_model_svr.predict(X_test)
svr_results = pd.DataFrame({'actual prices': y_test, 'svr predictions': svr_predictions})
print(svr_results)
```

```

      actual prices  svr predictions
DAY
2008-04-11    110.493356    108.230396
2014-06-15    107.373151    96.753772
2005-07-24     58.773857    73.270563
2007-11-25     95.159061    82.764840
2006-02-24     61.529147    72.949814
...
2019-04-16     71.404821    64.552245
2018-05-10     76.089681    61.865942
2000-11-20     35.238472    62.680940
2003-04-30     26.063182    28.060084
2018-04-08     76.025655    64.748916

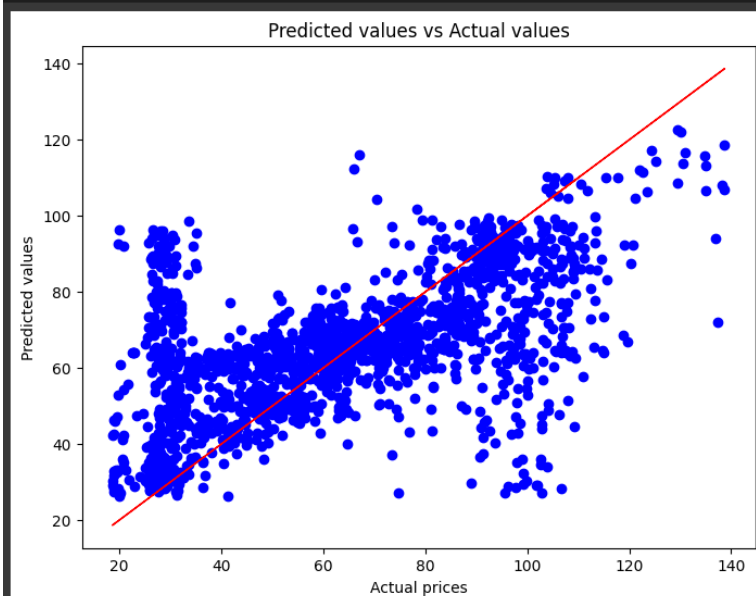
```

```
[1679 rows x 2 columns]
```

```

plt.figure(figsize=(8, 6))
plt.scatter(y_test, svr_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual prices')
plt.ylabel('Predicted values')
plt.title('Predicted values vs Actual values')
plt.show()

```



```
# Calculate RMSE and R2
svr_rmse = np.sqrt(mean_squared_error(y_test, svr_predictions))
svr_r2 = r2_score(y_test, svr_predictions)
print(f"RMSE: {svr_rmse:.2f}")
print(f"R2: {svr_r2:.2f}")
```

```

RMSE: 21.69
R2: 0.33

```

A.2 The RandomForest model

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
seed=42
rf = RandomForestRegressor(random_state=seed)
rf.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
rf_predictions = rf.predict(X_test)
```

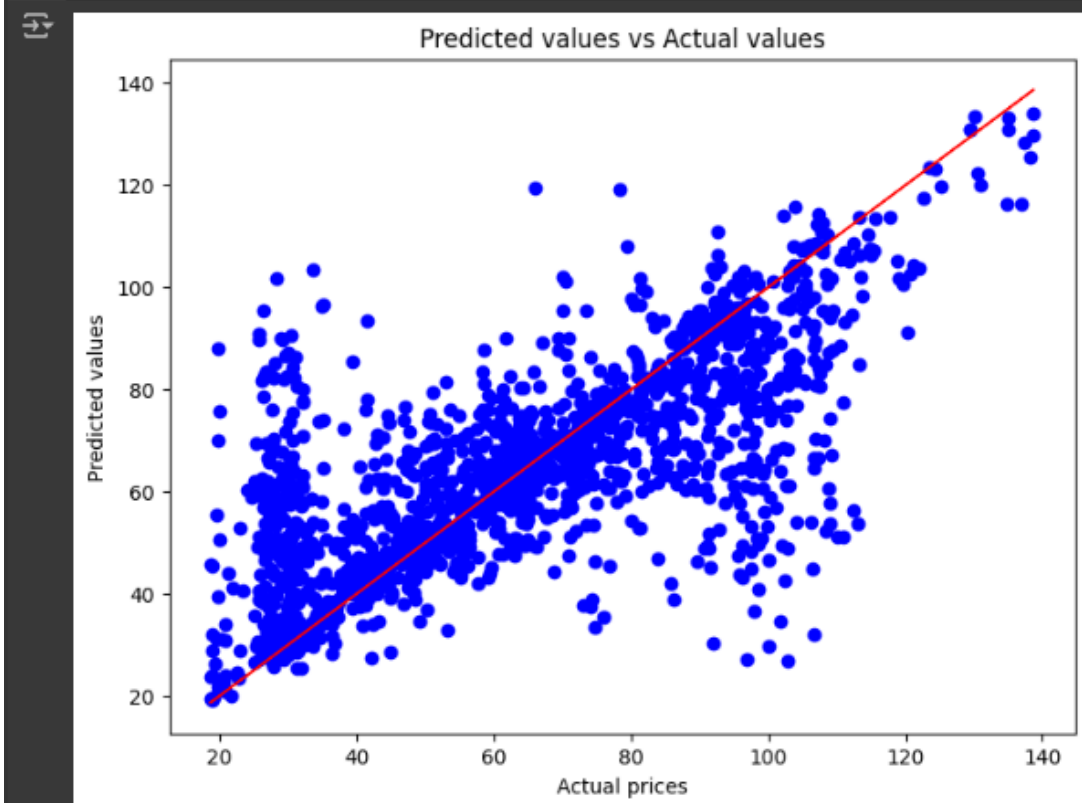
```
resultsRF = pd.DataFrame({'actual prices': y_test, 'randomforest predictionds': rf_predictions})
print(resultsRF)
```

	actual prices	randomforest predictionds
DAY		
2008-04-11	110.493356	105.438512
2014-06-15	107.373151	84.412191
2005-07-24	58.773857	69.699426
2007-11-25	95.159061	93.469943
2006-02-24	61.529147	62.846396
...
2019-04-16	71.404821	68.148951
2018-05-10	76.089681	74.938583
2000-11-20	35.238472	37.173565
2003-04-30	26.063182	39.026556
2018-04-08	76.025655	75.037775

```
[1679 rows x 2 columns]
```



```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='-', linewidth=1)
plt.xlabel('Actual prices')
plt.ylabel('Predicted values')
plt.title('Predicted values vs Actual values')
plt.show()
```



```
from sklearn.metrics import mean_squared_error
rf_mse = mean_squared_error(y_test, rf_predictions, squared=False )
print(f"Root Mean Squared Error: {rf_mse}")
```

Root Mean Squared Error: 18.48484669157709

```
rf_r2 = r2_score(y_test, rf_predictions)
print(f"R2 = {rf_r2:.2f}")
```

R² = 0.51

A.3 The XGBoost model

```
import pandas as pd
from xgboost import XGBRegressor
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'nthread':[4],
    'objective':['reg:linear'],
    'learning_rate': [.03, 0.05, .07],
    'max_depth': [5, 6, 7],
    'min_child_weight': [3, 4],
    'silent': [1],
    'subsample': [0.7],
    'colsample_bytree': [0.7],
    'n_estimators': [500]
}

# Initialize XGBRegressor
xgb = XGBRegressor(objective='reg:squarederror', random_state=42, eval_metric='rmse')

# Initialize GridSearchCV
grid_search = GridSearchCV(xgb, param_grid, cv=5, scoring='neg_root_mean_squared_error', verbose=2, n_jobs=-1)
```

```
# Fit GridSearchCV
grid_search.fit(X_train, y_train)

# Print best parameters and best score
print("Best Parameters found: ", grid_search.best_params_)
print("Lowest RMSE found: ", -grid_search.best_score_)
```

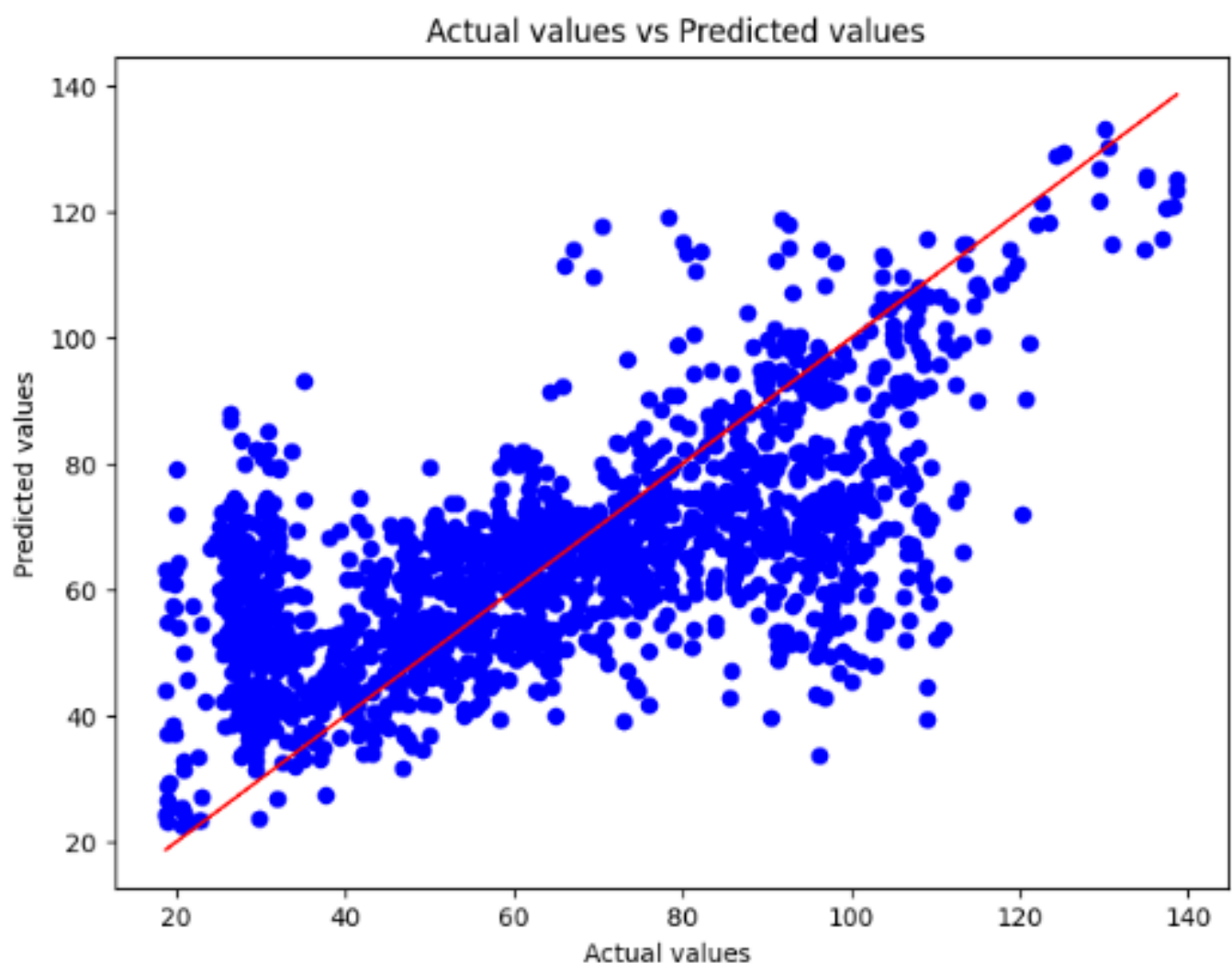
```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/backend/fork_exec.py:38: RuntimeWarning: os.fork() was called
  pid = os.fork()
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [16:45:19] WARNING: /workspace/src/objective/reg
  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [16:45:19] WARNING: /workspace/src/learner.cc:74:
  Parameters: { "silent" } are not used.

  warnings.warn(smsg, UserWarning)
Best Parameters found: {'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 3, 'n_estimators': 500}
Lowest RMSE found: 19.374275884871913
```

```
'n_estimators': 500, 'nthread': 4, 'objective': 'reg:linear', 'silent': 1, 'subsample': 0.7}
```

```
# Use the best estimator to predict on test data
best_model = grid_search.best_estimator_
xgb_predictions = best_model.predict(X_test)
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, xgb_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual values ')
plt.ylabel('Predicted values')
plt.title(' Actual values vs Predicted values')
plt.show()
```



```
# Calculate RMSE on test set
xgb_rmse = mean_squared_error(y_test, xgb_predictions, squared=False)
print(f"RMSE on test set: {xgb_rmse}")
```

```
RMSE on test set: 19.83057167509378
```

```
xgb_r2 = r2_score(y_test, xgb_predictions)
print(f"R2 = {xgb_r2:.2f}")
```

```
R2 = 0.44
```

A.4 The LSTM model

```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM, Dense, Flatten
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
```

```
def create_sequences(data, sequence_length):
    X = []
    y = []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i + sequence_length])
        y.append(data[i + sequence_length])
    return np.array(X), np.array(y)
```

```
sequence_length = 30 # Using 30 days of data to predict the next day's price
X, y = create_sequences(data['PRICE'], sequence_length)
```

```
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

```
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
```

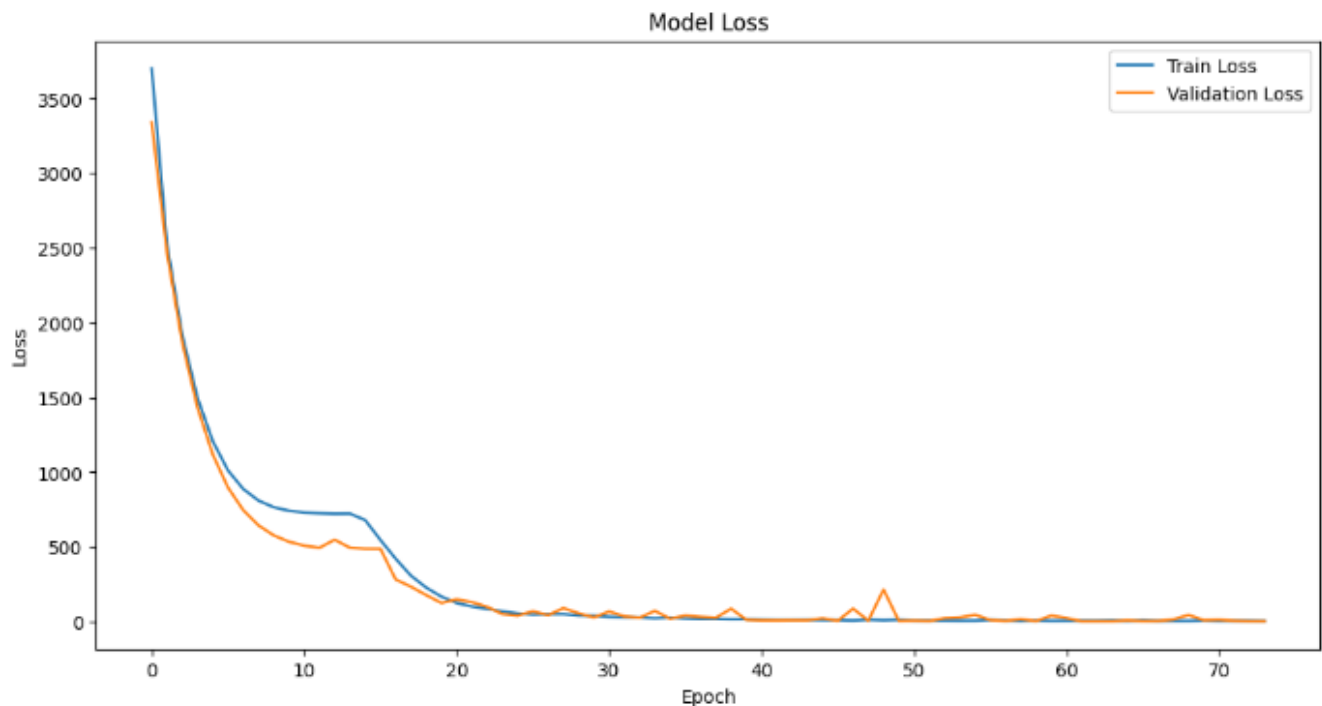
```
lstm = Sequential()
lstm.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
lstm.add(LSTM(50))
lstm.add(Dense(1))
```

```
early_stop = EarlyStopping(monitor='val_loss', patience=10)

# Train the model
history = lstm.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test), callbacks=[early_stop])

# Plot training & validation loss values
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

210/210 [=====] - 1s 6ms/step - loss: 4.2096 - val_loss: 4.0341



```
lstm_predictions = lstm.predict(X_test)
```

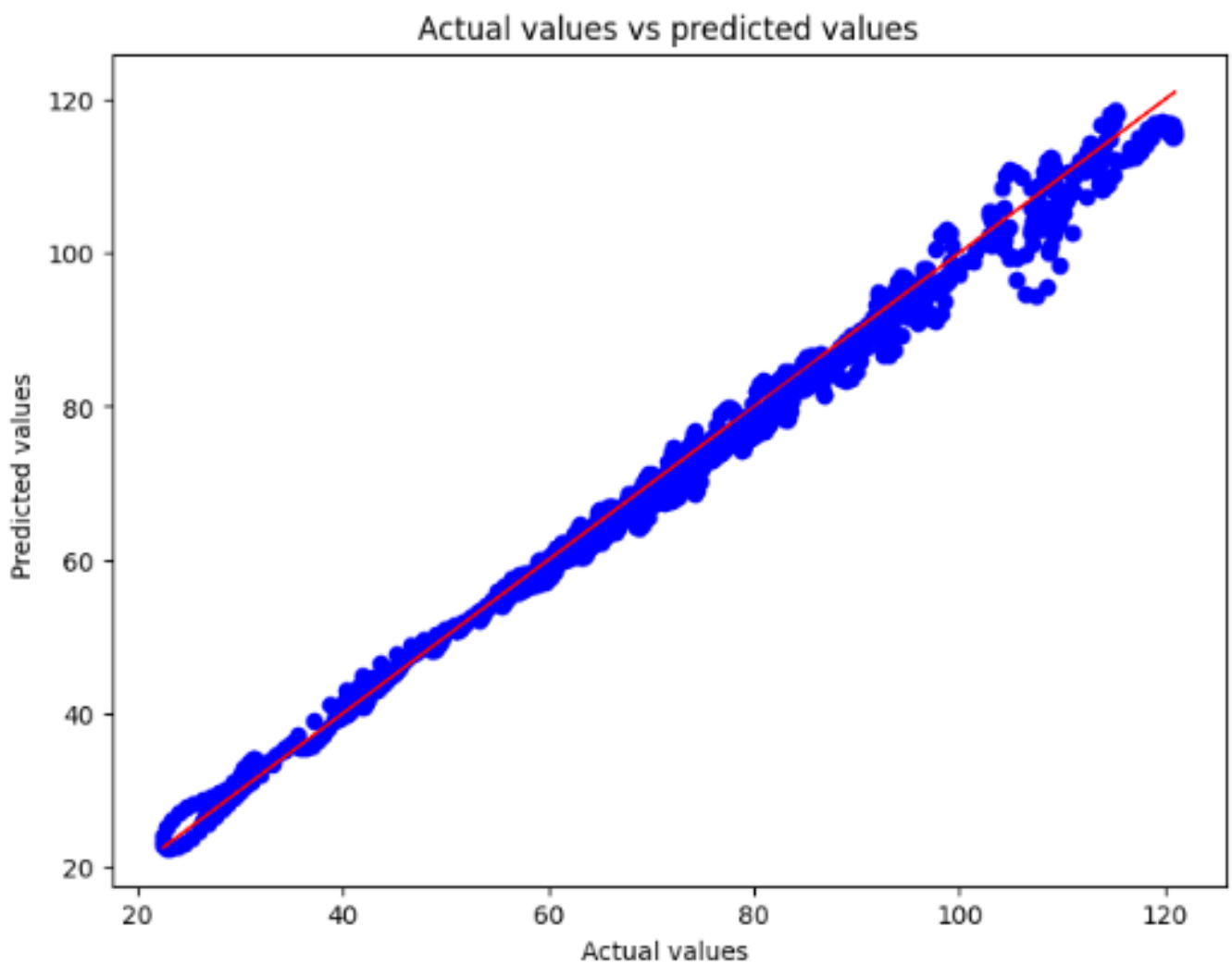
53/53 [=====] - 1s 3ms/step

```
resultsLSTM = pd.DataFrame({'actual prices': y_test, 'lstm_predictions': lstm_predictions.flatten()})  
print(resultsLSTM)
```

	actual prices	lstm_predictions
0	77.706429	76.164970
1	77.605272	75.741959
2	77.458268	75.363815
3	77.271082	75.056419
4	77.045584	74.823952
...
1668	86.658100	83.111382
1669	86.581204	84.325058
1670	86.506870	85.503952
1671	86.448682	86.410378
1672	86.416841	86.892563

```
[1673 rows x 2 columns]
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, lstm_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual values ')
plt.ylabel('Predicted values ')
plt.title('Actual values vs predicted values')
plt.show()
```



```
# Calculate RMSE on test set
lstm_rmse = mean_squared_error(y_test, lstm_predictions, squared=False)
print(f"RMSE on test set: {lstm_rmse}")
```

```
RMSE on test set: 2.0084995238592844
```

```
lstm_r2 = r2_score(y_test, lstm_predictions)
print(f"R² = {lstm_r2:.2f}")
```

```
R² = 0.99
```

A.5 The CNN model

```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dropout, Dense
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
X = X.reshape(X.shape[0], X.shape[1], 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
cnn = Sequential()
cnn.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
cnn.add(MaxPooling1D(pool_size=2))
cnn.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
cnn.add(MaxPooling1D(pool_size=2))
cnn.add(Flatten())
cnn.add(Dense(units=128, activation='relu'))
cnn.add(Dropout(rate=0.5))
cnn.add(Dense(units=1))
```

```
cnn.compile(optimizer='adam', loss='mean_squared_error')
```

```
cnn.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
```

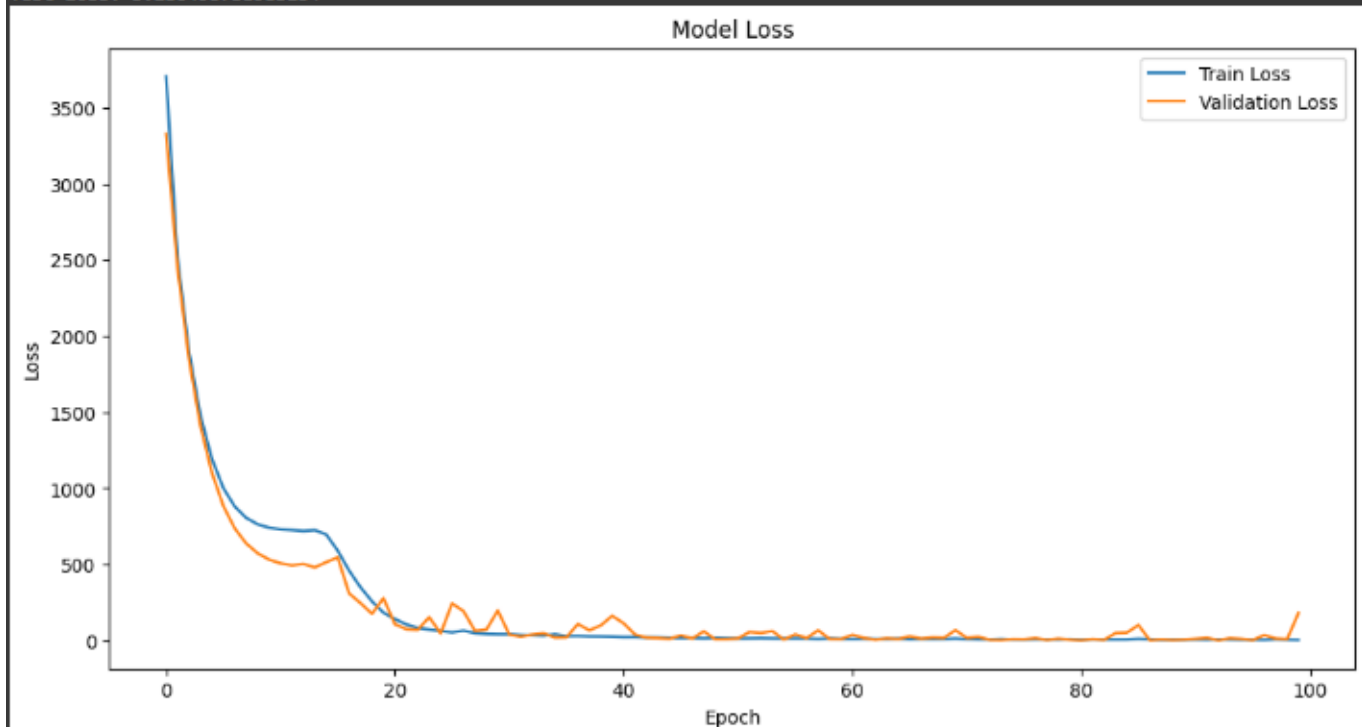


```

loss = cnn.evaluate(X_test, y_test)
print('Test loss:', loss)
# Plot training & validation loss values
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

```

Test loss: 5.180495738983154



```

cnn_predictions = cnn.predict(X_test)

53/53 [=====] - 0s 2ms/step

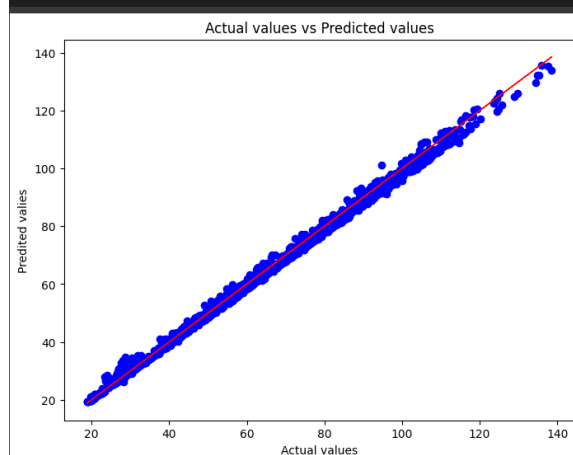
```

```
resultsCNN = pd.DataFrame({'actual prices': y_test, 'cnn_predictions': cnn_predictions.flatten()})
print(resultsCNN)
```

	actual prices	cnn_predictions
0	94.699287	98.974457
1	65.138100	65.826500
2	38.852539	40.268875
3	46.560364	46.273506
4	64.242432	65.566330
...
1668	91.275772	88.589073
1669	88.688652	86.948769
1670	112.065720	111.054153
1671	18.977600	19.057690
1672	46.964287	46.585861

[1673 rows x 2 columns]

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, cnn_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='-', linewidth=1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title('Actual values vs Predicted values')
plt.show()
```



```
# Calculate RMSE on test set
cnn_rmse = mean_squared_error(y_test, cnn_predictions, squared=False)
print(f"RMSE on test set: {cnn_rmse}")
```

RMSE on test set: 1.5853550434112549

```
cnn_r2 = r2_score(y_test, cnn_predictions)
print(f"R² = {cnn_r2:.2f}")
```

R² = 1.00

A.6 Hybrid RandomForest-XGBoost model

```
# Train individual models
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

xgb = XGBRegressor()
xgb.fit(X_train, y_train)

# Combine predictions from individual models
X_train_hybrid = np.column_stack((rf.predict(X_train), xgb.predict(X_train)))
X_test_hybrid = np.column_stack((rf.predict(X_test), xgb.predict(X_test)))

# Train hybrid model using Linear Regression
rf_xgb = LinearRegression()
rf_xgb.fit(X_train_hybrid, y_train)
```



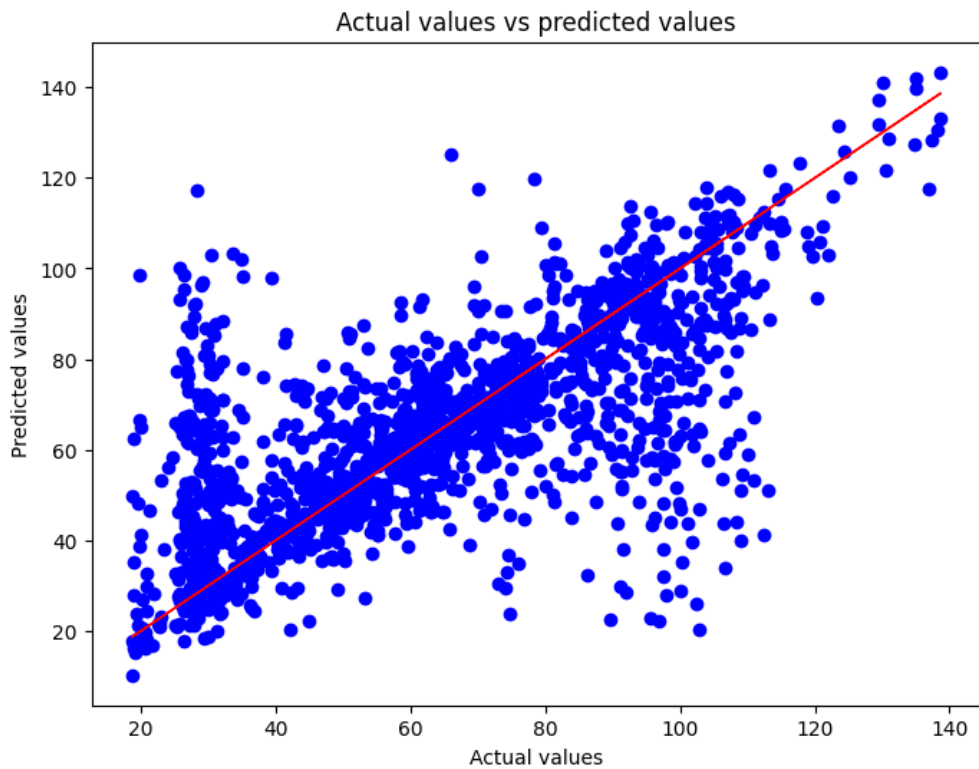
LinearRegression
LinearRegression()

```
# Evaluate hybrid model on test set
rf_xgb_predictions = rf_xgb.predict(X_test_hybrid)
resultsrf_xgb = pd.DataFrame({'actual prices': y_test, 'rf_xgb_predictions': rf_xgb_predictions.flatten()})
print(resultsrf_xgb)
```

	actual prices	rf_xgb_predictions
DAY		
2008-04-11	110.493356	107.848324
2014-06-15	107.373151	92.792676
2005-07-24	58.773857	72.265286
2007-11-25	95.159061	103.044101
2006-02-24	61.529147	59.568219
...
2019-04-16	71.404821	67.597092
2018-05-10	76.089681	73.166232
2000-11-20	35.238472	35.740393
2003-04-30	26.063182	39.267417
2018-04-08	76.025655	76.710758

[1679 rows x 2 columns]

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_xgb_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title('Actual values vs predicted values')
plt.show()
```



```
rf_xgb_rmse = mean_squared_error(y_test, rf_xgb_predictions, squared=False)
print(f"RMSE on test set: {rf_xgb_rmse}")
rf_xgb_r2 = r2_score(y_test, rf_xgb_predictions)
print(f"R² = {rf_xgb_r2:.2f}")
```

```
RMSE on test set: 19.44310522690441
R² = 0.46
```

A.7 Hybrid CNN-LSTM model

```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM, Dense, Conv1D, MaxPooling1D, Flatten, Concatenate, Input
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
from keras.models import Model
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```
oil_price = data['PRICE'].values.reshape(-1, 1)

# Normalize the data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_prices = scaler.fit_transform(oil_price)
```

```
def create_dataset(data, time_step=1):
    X, y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = 10
X, y = create_dataset(scaled_prices, time_step)
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Define the LSTM-CNN hybrid model
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv1D, MaxPooling1D, Flatten, Dense, LSTM, Concatenate

input_layer = Input(shape=(time_step, 1))

# CNN part
conv1 = Conv1D(filters=64, kernel_size=2, activation='relu')(input_layer)
maxpool1 = MaxPooling1D(pool_size=2)(conv1)
flatten = Flatten()(maxpool1)

# LSTM part
lstm = LSTM(50, return_sequences=True)(input_layer)
lstm = LSTM(50)(lstm)

# Concatenate CNN and LSTM parts
concat = Concatenate()([flatten, lstm])

# Fully connected layers
dense1 = Dense(50, activation='relu')(concat)
output = Dense(1)(dense1)

# Create the model
cnn_lstm = Model(inputs=input_layer, outputs=output)

# Compile the model
cnn_lstm.compile(optimizer='adam', loss='mean_squared_error')

# Summary of the model
cnn_lstm.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 10, 1)]	0	[]
conv1d_8 (Conv1D)	(None, 9, 64)	192	['input_1[0][0]']
max_pooling1d_8 (MaxPooling1D)	(None, 4, 64)	0	['conv1d_8[0][0]']
lstm_10 (LSTM)	(None, 10, 50)	10400	['input_1[0][0]']
flatten_4 (Flatten)	(None, 256)	0	['max_pooling1d_8[0][0]']
lstm_11 (LSTM)	(None, 50)	20200	['lstm_10[0][0]']
concatenate (Concatenate)	(None, 306)	0	['flatten_4[0][0]', 'lstm_11[0][0]']
dense_13 (Dense)	(None, 50)	15350	['concatenate[0][0]']
dense_14 (Dense)	(None, 1)	51	['dense_13[0][0]']
Total params: 46193 (180.44 KB)			
Trainable params: 46193 (180.44 KB)			
Non-trainable params: 0 (0.00 Byte)			

```
# Train the model
history = cnn_lstm.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test), verbose=1)
```

```
# Evaluate the model
train_loss = cnn_lstm.evaluate(X_train, y_train, verbose=0)
test_loss = cnn_lstm.evaluate(X_test, y_test, verbose=0)
print(f'Train Loss: {train_loss}, Test Loss: {test_loss}')
```

```
# Make predictions
cnn_lstm_predictions = cnn_lstm.predict(X_test)

# Inverse transform the predictions to get the actual prices
cnn_lstm_predictions = scaler.inverse_transform(cnn_lstm_predictions)

y_train = scaler.inverse_transform(y_train.reshape(-1, 1))
y_test = scaler.inverse_transform(y_test.reshape(-1, 1))
```

```
# Ensure y_test and cnn_lstm_predictions are numpy arrays
y_test = np.array(y_test)
cnn_lstm_predictions = np.array(cnn_lstm_predictions)

# Check the shape of y_test and cnn_lstm_predictions
print("Shape of y_test:", y_test.shape)
print("Shape of cnn_lstm_predictions:", cnn_lstm_predictions.shape)
```

```
Shape of y_test: (1677, 1)
Shape of cnn_lstm_predictions: (1677, 1)
```

```
if y_test.ndim > 1:
    y_test = y_test.reshape(-1)

if cnn_lstm_predictions.ndim > 1:
    cnn_lstm_predictions = cnn_lstm_predictions.reshape(-1)
```

```
resultsCNNlstm = pd.DataFrame({'actual prices': y_test, 'cnn lstm predictions': cnn_lstm_predictions})
print(resultsCNNlstm)
```

	actual prices	cnn lstm predictions
0	94.962974	94.251717
1	107.981214	107.950630
2	48.530286	48.738785
3	95.626969	95.488693
4	60.090281	59.825432
...
1672	35.962855	35.856506
1673	61.824139	61.903336
1674	52.135276	52.160564
1675	49.358847	49.253010
1676	115.551182	115.333572

```
[1677 rows x 2 columns]
```

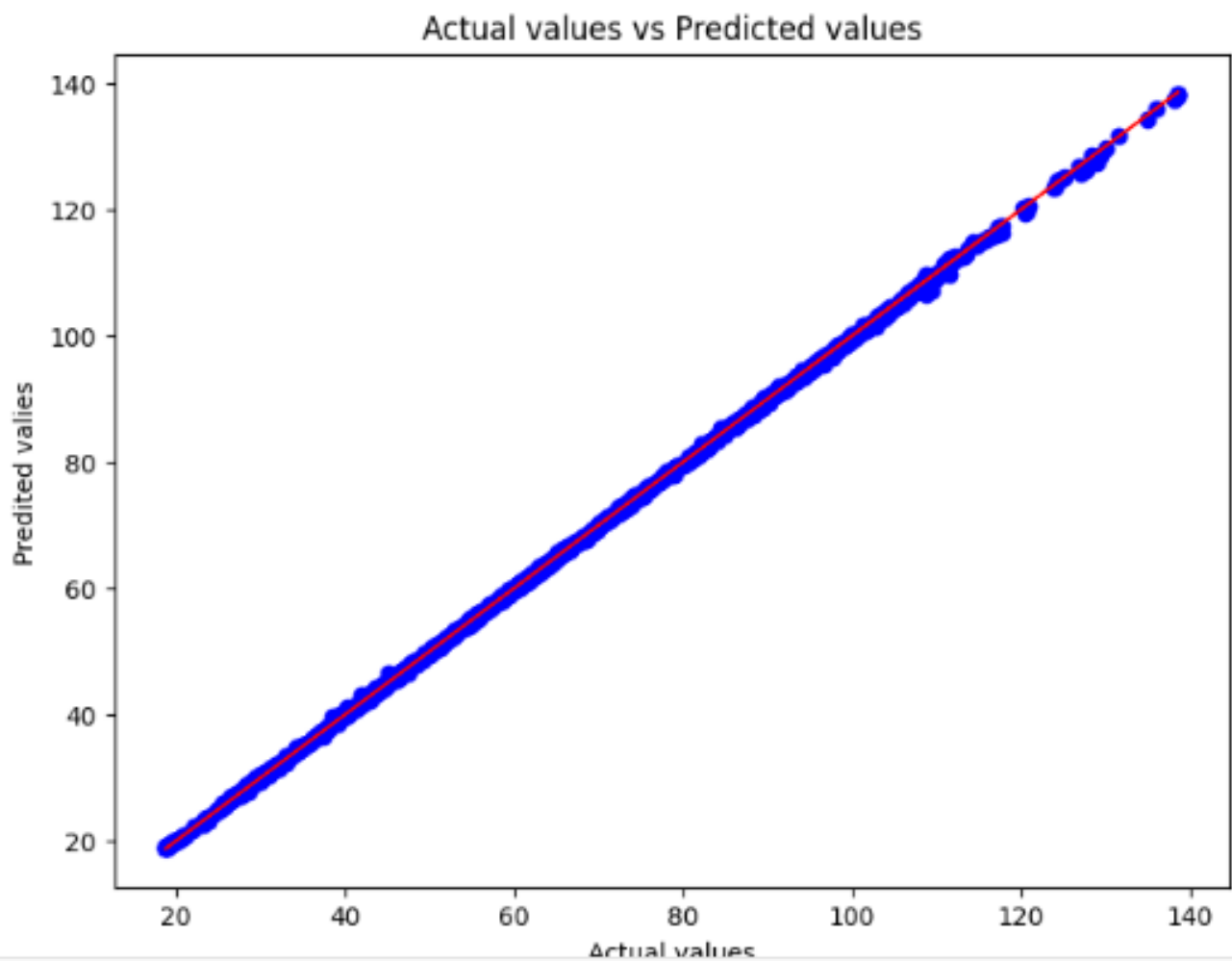
```
# Calculate RMSE on test set
cnn_lstm_rmse = mean_squared_error(y_test, cnn_lstm_predictions, squared=False)
print(f"RMSE on test set: {cnn_lstm_rmse}")
```

RMSE on test set: 0.21752225893275765

```
cnn_lstm_r2 = r2_score(y_test, cnn_lstm_predictions)
print(f"R² = {cnn_lstm_r2:.2f}")
```

R² = 1.00

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, cnn_lstm_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='-', linewidth=1)
plt.xlabel('Actual values')
plt.ylabel('Predited valies')
plt.title('Actual values vs Predicted values')
plt.show()
```



Appendix B

Monthly data

[Direct link to the Python syntax on google colab](#)

B.1 The SVR model

```
# Define the parameter grid
param_grid = {
    'C': [1, 10, 100, 1000],
    'gamma': [0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf']
}

svr=SVR()

# Perform grid search with MSE as the scoring metric
grid_search_svr = GridSearchCV(svr, param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search_svr.fit(X_train, y_train)

# Get the best parameters and the corresponding model
best_params_svr = grid_search_svr.best_params_
best_model_svr = grid_search_svr.best_estimator_

# Print best parameters and best score
print("Best Parameters found: ", grid_search_svr.best_params_)
print("Lowest score found: ", -grid_search_svr.best_score_)

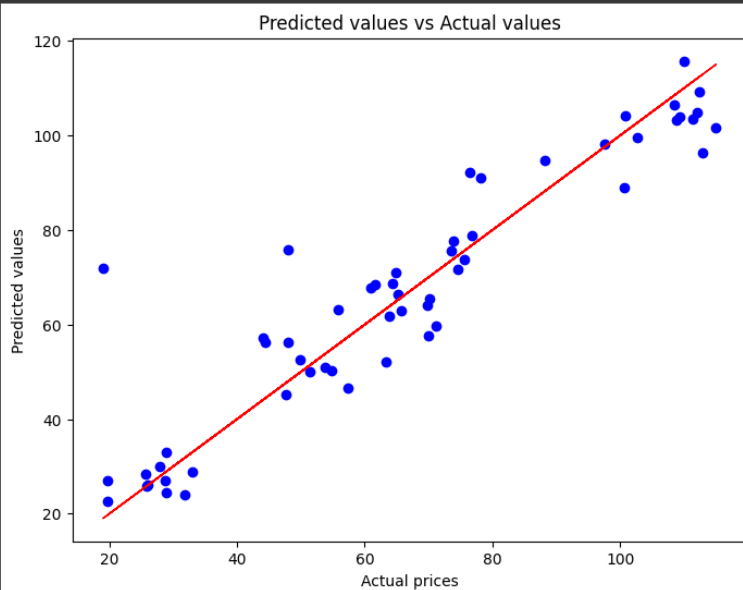
Best Parameters found: {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
Lowest score found: 223.0205716738946

# Make predictions using the best model
svr_predictions = best_model_svr.predict(X_test)
resultsSVR = pd.DataFrame({'actual prices': y_test, 'SVR predictions': svr_predictions})
print(resultsSVR)
```

	actual prices	SVR predictionds
MONTH		
2002-07-01	25.790001	25.940591
2010-05-01	75.669998	73.813121
2016-05-01	47.730000	45.244139
2010-08-01	78.220001	91.083027
2018-01-01	69.930000	57.603674
2019-07-01	63.919998	61.818479
2012-01-01	111.430000	103.483559
2016-12-01	53.820000	50.903867
2021-03-01	65.760002	62.988246
2011-12-01	108.559998	106.420429
2006-08-01	74.500000	71.790033
2011-01-01	97.500000	98.171206
2019-10-01	60.860001	67.857091
2022-02-01	100.709999	88.966499
2012-09-01	112.059998	104.821510
2011-11-01	112.410004	109.153214
2021-09-01	73.849998	77.759998
2005-01-01	44.389999	56.302307
2007-01-01	55.779999	63.202666
2019-12-01	69.739998	64.075073
2003-10-01	28.940001	24.423186
2006-02-01	61.590000	68.501907
2015-01-01	47.910000	75.866537
2013-09-01	112.949997	96.326662
2022-01-01	88.209999	94.746197
2017-07-01	47.959999	56.158300
2009-10-01	73.629997	75.510734
2003-07-01	27.910000	30.013509
2000-10-01	31.799999	24.014253
2011-09-01	115.029999	101.728818
2001-11-01	19.000000	72.013460

2009-02-01	44.070000	57.198172
2002-01-01	19.639999	27.069326
2010-01-01	76.790001	78.745079
2014-08-01	100.860001	104.273105
2006-04-01	70.209999	65.462336
2019-04-01	71.150002	59.649633
2019-06-01	64.830002	70.887256
2000-07-01	28.799999	26.931365
2005-09-01	63.299999	52.067187
2003-11-01	28.940001	32.921577
2010-07-01	76.489998	92.145134
2005-07-01	57.299999	46.524679
2002-02-01	19.730000	22.735160
2009-07-01	65.209999	66.486406
2017-03-01	51.400002	50.090439
2001-08-01	25.959999	26.138550
2016-10-01	49.790001	52.604283
2014-01-01	109.959999	115.746639
2017-01-01	54.840000	50.297045
2007-03-01	64.300003	68.705367
2012-11-01	109.360001	104.027111
2013-04-01	102.639999	99.468270
2001-04-01	25.650000	28.440753
2013-03-01	108.870003	103.270676
2000-11-01	33.060001	28.947848

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, svr_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual prices')
plt.ylabel('Predicted values')
plt.title('Predicted values vs Actual values')
plt.show()
```



```
# Calculate RMSE and R2
svr_rmse = np.sqrt(mean_squared_error(y_test, svr_predictions))
svr_r2 = r2_score(y_test, svr_predictions)

print(f"Best parameters: {best_params_svr}")
print(f"RMSE: {svr_rmse:.2f}")
print(f"R2: {svr_r2:.2f}")

Best parameters: {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
RMSE: 10.64
R2: 0.86
```

B.2 The Random Forest model

```
rf = RandomForestRegressor(random_state=seed)
rf.fit(X_train, y_train)
```

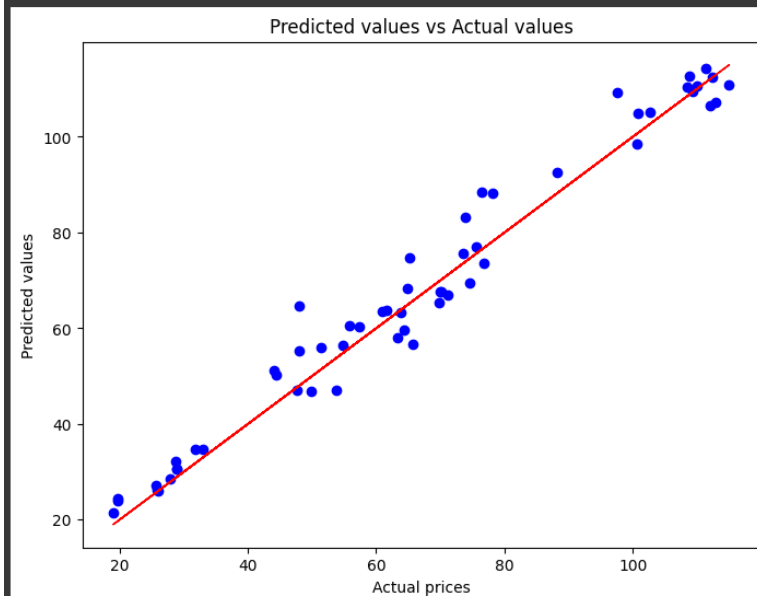
RandomForestRegressor
RandomForestRegressor(random_state=42)

```
rf_predictions = rf.predict(X_test)
rf_results = pd.DataFrame({'actual prices': y_test, 'randomforest predictions': rf_predictions})
print(rf_results)
```

	actual prices	randomforest predictions
MONTH		
2002-07-01	25.790001	26.252701
2010-05-01	75.669998	76.912800
2016-05-01	47.730000	46.968400
2010-08-01	78.220001	88.267298
2018-01-01	69.930000	67.651401
2019-07-01	63.919998	63.347100
2012-01-01	111.430000	114.290600
2016-12-01	53.820000	47.003800
2021-03-01	65.760002	56.765501
2011-12-01	108.559998	110.380140
2006-08-01	74.500000	69.502501
2011-01-01	97.500000	109.285400
2019-10-01	60.860001	63.504601
2022-02-01	100.709999	98.515102
2012-09-01	112.059998	106.514802
2011-11-01	112.410004	112.385139
2021-09-01	73.849998	83.169601
2005-01-01	44.389999	50.158799
2007-01-01	55.779999	60.451501
2019-12-01	69.739998	65.447000
2003-10-01	28.940001	30.545900
2006-02-01	61.590000	63.693299
2015-01-01	47.910000	64.652600
2013-09-01	112.949997	107.127361

2017-07-01	47.959999	55.257399
2009-10-01	73.629997	75.742702
2003-07-01	27.910000	28.635400
2000-10-01	31.799999	34.704900
2011-09-01	115.029999	110.776101
2001-11-01	19.000000	21.532400
2009-02-01	44.070000	51.090900
2002-01-01	19.639999	24.441800
2010-01-01	76.790001	73.478801
2014-08-01	100.860001	104.922082
2006-04-01	70.209999	67.731800
2019-04-01	71.150002	67.024398
2019-06-01	64.830002	68.415898
2000-07-01	28.799999	32.257700
2005-09-01	63.299999	57.962499
2003-11-01	28.940001	30.606000
2010-07-01	76.489998	88.450298
2005-07-01	57.299999	60.426299
2002-02-01	19.730000	23.907700
2009-07-01	65.209999	74.755300
2017-03-01	51.400002	56.087001
2001-08-01	25.959999	26.138700
2016-10-01	49.790001	46.766901
2014-01-01	109.959999	110.596098
2017-01-01	54.840000	56.356301
2007-03-01	64.300003	59.529501
2012-11-01	109.360001	109.501702
2013-04-01	102.639999	105.118580
2001-04-01	25.650000	27.216300
2013-03-01	108.870003	112.577199
2000-11-01	33.060001	34.647200

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual prices')
plt.ylabel('Predicted values')
plt.title('Predicted values vs Actual values')
plt.show()
```



```
# Calculate RMSE on test set
rf_rmse = mean_squared_error(y_test, rf_predictions, squared=False)
print(f"RMSE on test set: {rf_rmse}")
rf_r2 = r2_score(y_test, rf_predictions)
print(f"R² = {rf_r2:.2f}")
```

```
RMSE on test set: 5.288514497411396
R² = 0.97
```

B.3 The XGBoost model

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'nthread': [4],
    'objective': ['reg:linear'],
    'learning_rate': [.03, 0.05, .07],
    'max_depth': [5, 6, 7],
    'min_child_weight': [3, 4],
    'silent': [1],
    'reg_lambda': [0.1],
    'reg_alpha': [0],
    'subsample': [0.7],
    'colsample_bytree': [0.7],
    'n_estimators': [500]
}

# Initialize XGBRegressor
xgb = XGBRegressor(objective='reg:squarederror', random_state=42, eval_metric='rmse')
```

```
# Initialize GridSearchCV
grid_search_xgb = GridSearchCV(xgb, param_grid, cv=5, scoring='neg_mean_squared_error', verbose=2, n_jobs=-1)
grid_search_xgb.fit(X_train, y_train)

Fitting 5 folds for each of 18 candidates, totalling 90 fits
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/backend/fork_exec.py:38: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multi-
pid = os.fork()
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [20:52:19] WARNING: /workspace/src/objective/regression_obj.cu:209: reg:linear is now de
warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [20:52:19] WARNING: /workspace/src/learner.cc:742:
Parameters: { "silent" } are not used.

warnings.warn(msg, UserWarning)
> GridSearchCV
> estimator: XGBRegressor
  > XGBRegressor

best_params_xgb = grid_search_xgb.best_params_
best_model_xgb = grid_search_xgb.best_estimator_
# Print best parameters and best score
print("Best Parameters found: ", grid_search_xgb.best_params_)
print("Lowest Score found: ", -grid_search_xgb.best_score_)

Best Parameters found: {'colsample_bytree': 0.7, 'learning_rate': 0.05, 'max_depth': 6, 'min_child_weight': 3, 'n_estimators': 500, 'nthread': 4, 'objective': 'r
Lowest Score found: 93.18370666503907
```

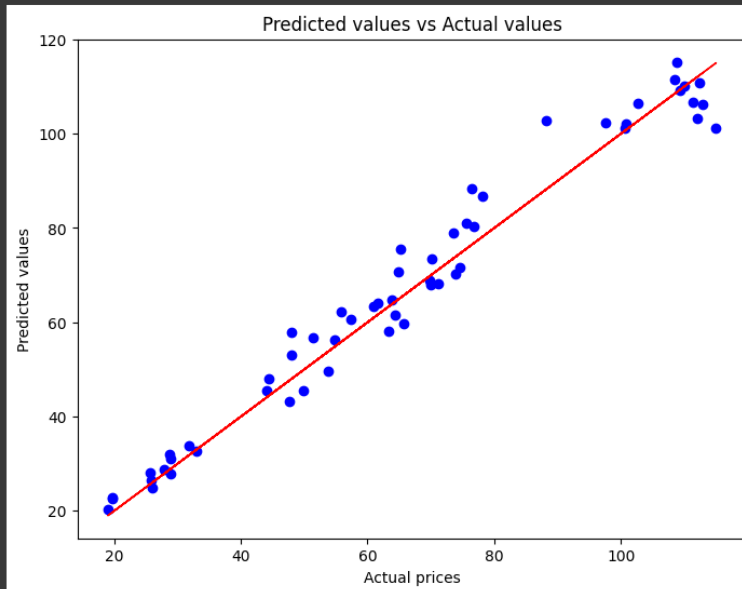
```
'objective': 'reg:linear', 'reg_alpha': 0, 'reg_lambda': 0.1, 'silent': 1, 'subsample': 0.7}
```

```
# Use the best estimator to predict on test data
xgb_predictions = best_model_xgb.predict(X_test)
resultsXGB = pd.DataFrame({'actual prices': y_test, 'XGBoost predictionds': xgb_predictions})
print(resultsXGB)
```

MONTH	actual prices	XGBoost predictionds
2002-07-01	25.790001	26.502102
2010-05-01	75.669998	80.959953
2016-05-01	47.730000	43.282009
2010-08-01	78.220001	86.731949
2018-01-01	69.930000	67.943268
2019-07-01	63.919998	64.661362
2012-01-01	111.430000	106.733650
2016-12-01	53.820000	49.741158
2021-03-01	65.760002	59.787117
2011-12-01	108.559998	111.440636
2006-08-01	74.500000	71.732094
2011-01-01	97.500000	102.295090
2019-10-01	60.860001	63.412384
2022-02-01	100.709999	101.104103
2012-09-01	112.059998	103.347450
2011-11-01	112.410004	110.743187
2021-09-01	73.849998	70.146339
2005-01-01	44.389999	48.054611
2007-01-01	55.779999	62.218315
2019-12-01	69.739998	68.879158
2003-10-01	28.940001	27.978697
2006-02-01	61.590000	64.145180
2015-01-01	47.910000	57.793003
2013-09-01	112.949997	106.257263
2022-01-01	88.209999	102.743042

2009-10-01	73.629997	78.926758
2003-07-01	27.910000	28.847242
2000-10-01	31.799999	33.733986
2011-09-01	115.029999	101.267067
2001-11-01	19.000000	20.269539
2009-02-01	44.070000	45.566788
2002-01-01	19.639999	22.523502
2010-01-01	76.790001	80.289024
2014-08-01	100.860001	102.018311
2006-04-01	70.209999	73.402954
2019-04-01	71.150002	68.112358
2019-06-01	64.830002	70.638321
2000-07-01	28.799999	31.889742
2005-09-01	63.299999	58.205200
2003-11-01	28.940001	31.035912
2010-07-01	76.489998	88.318138
2005-07-01	57.299999	60.717449
2002-02-01	19.730000	22.940817
2009-07-01	65.209999	75.459251
2017-03-01	51.400002	56.795307
2001-08-01	25.959999	24.823780
2016-10-01	49.790001	45.509907
2014-01-01	109.959999	110.144463
2017-01-01	54.840000	56.170647
2007-03-01	64.300003	61.504082
2012-11-01	109.360001	109.126648
2013-04-01	102.639999	106.374275
2001-04-01	25.650000	28.048033
2013-03-01	108.870003	115.224854
2000-11-01	33.060001	32.656548

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, xgb_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual prices')
plt.ylabel('Predicted values')
plt.title('Predicted values vs Actual values')
plt.show()
```



```
# Calculate RMSE on test set
xgb_rmse = mean_squared_error(y_test, xgb_predictions, squared=False)
print(f"RMSE on test set: {xgb_rmse}")

RMSE on test set: 5.185967445373535

from sklearn.metrics import r2_score

# Supposons que vous ayez des données réelles y_true et des prédictions y_pred
xgb_r2 = r2_score(y_test, xgb_predictions)
print(f"R² = {xgb_r2:.2f}")

R² = 0.97
```

B.4 The LSTM model

```
# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

# Define LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(1, X_train.shape[2])))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# Fit model
model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=2)
```



```
# Make predictions
lstm_predictions = model.predict(X_test)
resultsLSTM = pd.DataFrame({'actual prices': y_test, 'lstm_predictions': lstm_predictions.flatten()})
print(resultsLSTM)
```

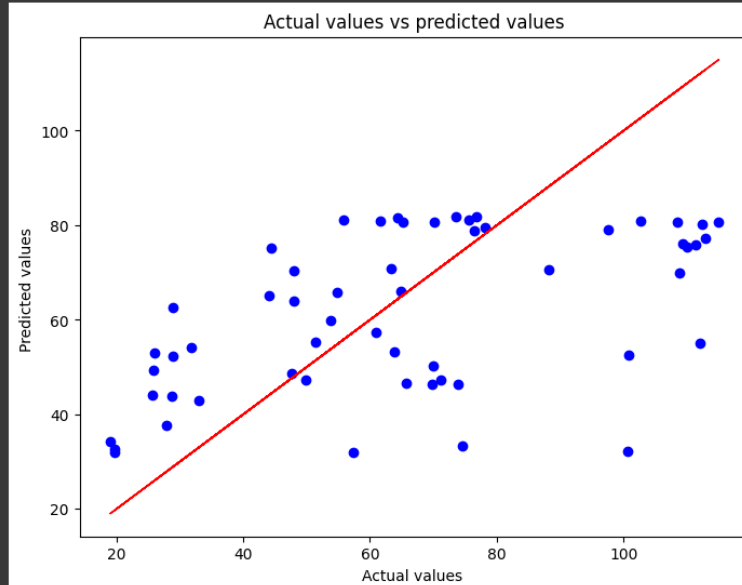
MONTH	actual prices	lstm_predictions
-------	---------------	------------------

2002-07-01	25.79	49.345692
2010-05-01	75.67	81.221527
2016-05-01	47.73	48.605762
2010-08-01	78.22	79.615944
2018-01-01	69.93	50.314857
2019-07-01	63.92	53.332748
2012-01-01	111.43	75.894691
2016-12-01	53.82	59.846378
2021-03-01	65.76	46.587234
2011-12-01	108.56	80.755470
2006-08-01	74.50	33.443420
2011-01-01	97.50	79.005180
2019-10-01	60.86	57.295475
2022-02-01	100.71	32.204712
2012-09-01	112.06	55.014729
2011-11-01	112.41	80.230644
2021-09-01	73.85	46.405907
2005-01-01	44.39	75.212700
2007-01-01	55.78	81.070000
2019-12-01	69.74	46.475052
2003-10-01	28.94	62.665527
2006-02-01	61.59	80.803368

2015-01-01	47.91	70.451866
2013-09-01	112.95	77.174126
2022-01-01	88.21	70.541237
2017-07-01	47.96	63.891052
2009-10-01	73.63	81.843185
2003-07-01	27.91	37.784138
2000-10-01	31.80	54.164253
2011-09-01	115.03	80.632378
2001-11-01	19.00	34.159466
2009-02-01	44.07	65.038086
2002-01-01	19.64	31.961689
2010-01-01	76.79	81.853638
2014-08-01	100.86	52.459259
2006-04-01	70.21	80.624420
2019-04-01	71.15	47.211975
2019-06-01	64.83	66.062782
2000-07-01	28.80	43.808369
2005-09-01	63.30	70.879333
2003-11-01	28.94	52.255360
2010-07-01	76.49	78.850098
2005-07-01	57.30	31.955612
2002-02-01	19.73	32.594440
2009-07-01	65.21	80.709503
2017-03-01	51.40	55.278381
2001-08-01	25.96	52.993580
2016-10-01	49.79	47.229893
2014-01-01	109.96	75.520668
2017-01-01	54.84	65.829224
2007-03-01	64.30	81.579224



```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, lstm_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual values ')
plt.ylabel('Predicted values ')
plt.title('Actual values vs predicted values')
plt.show()
```



```
# Calculate RMSE and R2 on test set
lstm_rmse = mean_squared_error(y_test, lstm_predictions, squared=False)
print(f"RMSE on test set: {lstm_rmse}")
lstm_r2 = r2_score(y_test, lstm_predictions)
print(f"R² = {lstm_r2:.2f}")
```

```
RMSE on test set: 24.92502374778355
R² = 0.25
```

B.5 The CNN model

```
# Reshape the input data for the CNN
X_train = np.expand_dims(X_train, axis=2)
X_test = np.expand_dims(X_test, axis=2)
# Define the CNN model
cnn = Sequential()
cnn.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
cnn.add(Flatten())
cnn.add(Dense(1, activation='linear'))
# Compile the model
cnn.compile(optimizer='adam', loss='mean_squared_error')
# Train the model
cnn.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
# Evaluate the model
loss = model.evaluate(X_test, y_test)
print('Test loss:', loss)
```

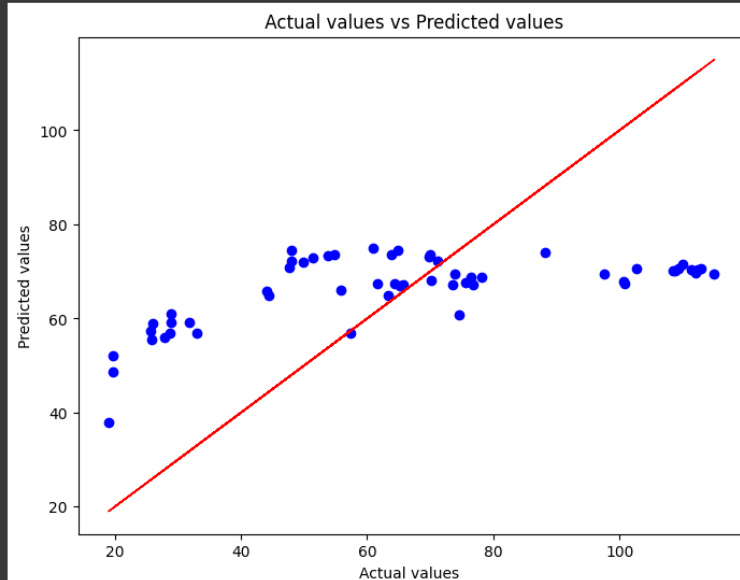
```
# Make predictions
cnn_predictions = cnn.predict(X_test)
# Reshape the predictions to be a 1D array
cnn_predictions = cnn_predictions.flatten()
resultscnn = pd.DataFrame({'actual prices': y_test, 'CNN predictionds': cnn_predictions})
print(resultscnn)
```

```
2/2 [=====] - 0s 6ms/step
```

	actual prices	CNN predictionds
0	25.79	55.579205
1	75.67	67.604691
2	47.73	70.827286
3	78.22	68.739288
4	69.93	73.621117
5	63.92	73.664268
6	111.43	70.400681
7	53.82	73.401100
8	65.76	67.180168
9	108.56	70.247986
10	74.50	60.762421
11	97.50	69.386078
12	60.86	75.063202
13	100.71	67.848305
14	112.06	69.712883
15	112.41	70.328423
16	73.85	69.363350
17	44.39	64.910248
18	55.78	65.962189
19	69.74	73.097252
20	28.94	60.896694
21	61.59	67.445358
22	47.91	72.246315
23	112.95	70.585510

24	88.21	74.142914
25	47.96	74.608887
26	73.63	67.108276
27	27.91	56.047710
28	31.80	59.104362
29	115.03	69.560158
30	19.00	38.014881
31	44.07	65.725975
32	19.64	48.766266
33	76.79	67.258995
34	100.86	67.341049
35	70.21	67.996109
36	71.15	72.236786
37	64.83	74.563477
38	28.80	56.841076
39	63.30	64.782166
40	28.94	59.145245
41	76.49	68.721428
42	57.30	56.882626
43	19.73	52.050499
44	65.21	66.917656
45	51.40	72.970490
46	25.96	59.041523
47	49.79	72.090767
48	109.96	71.478241
49	54.84	73.685638
50	64.30	67.347794
51	109.36	70.665451
52	102.64	70.549316
53	25.65	57.238789
54	108.87	70.058250
55	33.06	56.820164

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, cnn_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='--', linewidth=1)
plt.xlabel('Actual values ')
plt.ylabel('Predicted values')
plt.title(' Actual values vs Predicted values')
plt.show()
```



```
# Calculate RMSE on test set
cnn_rmse = mean_squared_error(y_test, cnn_predictions, squared=False)
print(f"RMSE on test set: {cnn_rmse}")
cnn_r2 = r2_score(y_test, cnn_predictions)
print(f"R² = {cnn_r2:.2f}")
```

```
RMSE on test set: 24.964631879726404
R² = 0.25
```

B.6 The Hybrid RandomForest-XGBoost model

```
# Train individual models
rf = RandomForestRegressor()
rf.fit(X_train, y_train)

xgb = XGBRegressor()
xgb.fit(X_train, y_train)

# Combine predictions from individual models
X_train_hybrid = np.column_stack((rf.predict(X_train), xgb.predict(X_train)))
X_test_hybrid = np.column_stack((rf.predict(X_test), xgb.predict(X_test)))

# Train hybrid model using Linear Regression
rf_xgb = LinearRegression()
rf_xgb.fit(X_train_hybrid, y_train)
```

LinearRegression

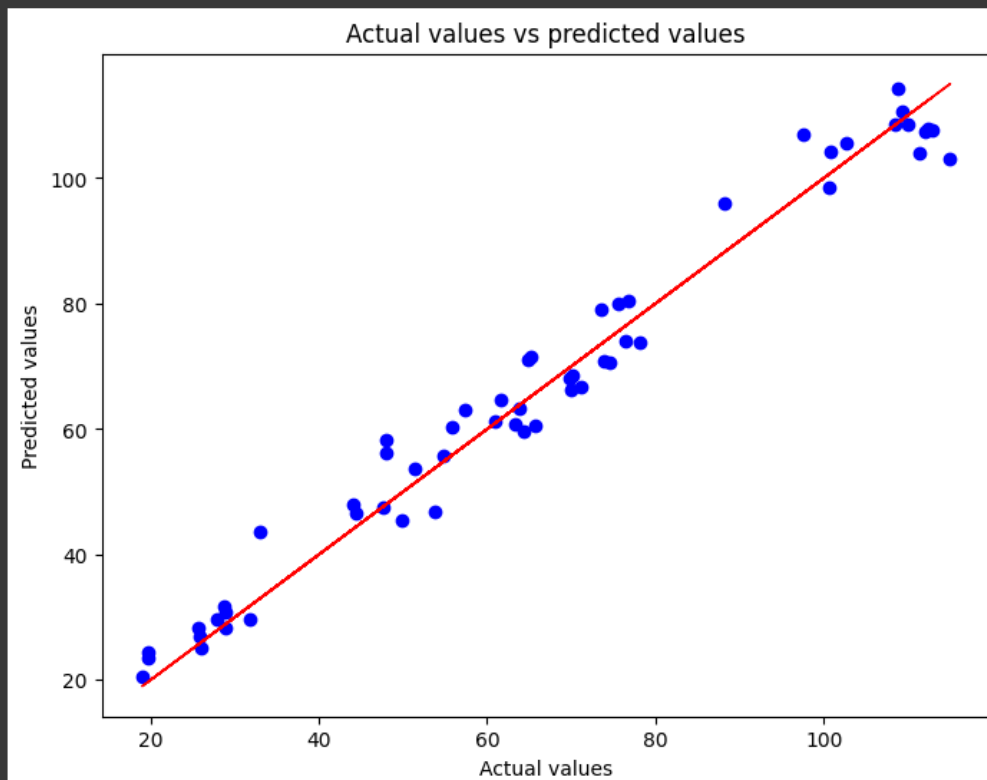
LinearRegression()

```
# Evaluate hybrid model on test set
rf_xgb_predictions = rf_xgb.predict(X_test_hybrid)
resultsrf_xgb = pd.DataFrame({'actual_prices': y_test, 'rf_xgb_predictions': rf_xgb_predictions.flatten()})
print(resultsrf_xgb)
```

	actual prices	rf_xgb_predictions
MONTH		
2002-07-01	25.79	26.842595
2010-05-01	75.67	80.065343
2016-05-01	47.73	47.480311
2010-08-01	78.22	73.808443
2018-01-01	69.93	66.250194
2019-07-01	63.92	63.296957
2012-01-01	111.43	103.919041
2016-12-01	53.82	46.836895
2021-03-01	65.76	60.450505
2011-12-01	108.56	108.524754
2006-08-01	74.50	70.600611
2011-01-01	97.50	106.977470
2019-10-01	60.86	61.235025
2022-02-01	100.71	98.570006
2012-09-01	112.06	107.323938
2011-11-01	112.41	107.973823
2021-09-01	73.85	70.855256
2005-01-01	44.39	46.580929
2007-01-01	55.78	60.380696
2019-12-01	69.74	68.020888
2003-10-01	28.94	28.313847
2006-02-01	61.59	64.585023
2015-01-01	47.91	58.175121
2013-09-01	112.95	107.655956
2022-01-01	88.21	96.088067
2017-07-01	47.96	56.286951

2009-10-01	73.63	79.118150
2003-07-01	27.91	29.593901
2000-10-01	31.80	29.714279
2011-09-01	115.03	103.092800
2001-11-01	19.00	20.514430
2009-02-01	44.07	47.863917
2002-01-01	19.64	24.393980
2010-01-01	76.79	80.443566
2014-08-01	100.86	104.210386
2006-04-01	70.21	68.614211
2019-04-01	71.15	66.755504
2019-06-01	64.83	71.172189
2000-07-01	28.80	31.647953
2005-09-01	63.30	60.688524
2003-11-01	28.94	30.721500
2010-07-01	76.49	74.091396
2005-07-01	57.30	62.962026
2002-02-01	19.73	23.419561
2009-07-01	65.21	71.621690
2017-03-01	51.40	53.588630
2001-08-01	25.96	25.036821
2016-10-01	49.79	45.568396
2014-01-01	109.96	108.673046
2017-01-01	54.84	55.712970
2007-03-01	64.30	59.564602
2012-11-01	109.36	110.651260
2013-04-01	102.64	105.685163
2001-04-01	25.65	28.302076
2013-03-01	108.87	114.343919
2000-11-01	33.06	43.598378

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_xgb_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='-', linewidth=1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title('Actual values vs predicted values')
plt.show()
```



```
rf_xgb_rmse = mean_squared_error(y_test, rf_xgb_predictions, squared=False)
print(f"RMSE on test set: {rf_xgb_rmse}")
rf_xgb_r2 = r2_score(y_test, rf_xgb_predictions)
print(f"R² = {rf_xgb_r2:.2f}")
```

```
RMSE on test set: 4.770687298720523
R² = 0.97
```

B.7 The Hybrid CNN-LSTM model

```
# Define the hybrid LSTM-CNN model
cnn_lstm = Sequential()
cnn_lstm.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
cnn_lstm.add(MaxPooling1D(pool_size=2))
cnn_lstm.add(LSTM(64, return_sequences=True))
cnn_lstm.add(Dropout(0.2))
cnn_lstm.add(LSTM(32))
cnn_lstm.add(Dropout(0.2))
cnn_lstm.add(Dense(1, activation='linear'))

# Compile the model
cnn_lstm.compile(optimizer='adam', loss='mean_squared_error')
```

```
[80] # Train the model
cnn_lstm.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)

7/7 [=====] - 0s 10ms/step - loss: 4129.0586
Epoch 23/50
7/7 [=====] - 0s 11ms/step - loss: 4108.3267
Epoch 24/50
7/7 [=====] - 0s 8ms/step - loss: 4080.2869
Epoch 25/50
7/7 [=====] - 0s 10ms/step - loss: 4055.4270
```

```
# Evaluate the model
loss = cnn_lstm.evaluate(X_train, y_train)
print('Loss:', loss)
```

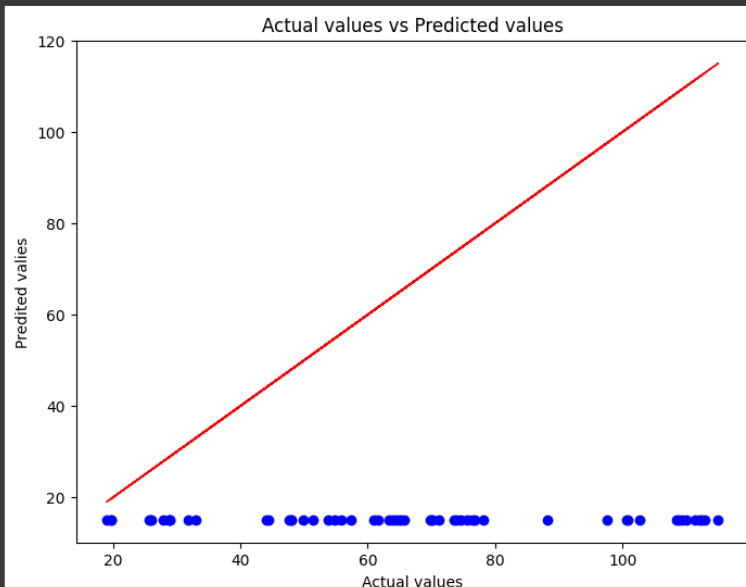
```
7/7 [=====] - 1s 4ms/step - loss: 3525.3997
Loss: 3525.399658203125
```

```
# Make predictions
cnn_lstm_predictions = cnn_lstm.predict(X_test)
cnn_lstm_predictions = cnn_lstm_predictions.flatten() # Flatten the predictions array
resultscnn_lstm = pd.DataFrame({'actual prices': y_test, 'cnn_lstm_predictions': cnn_lstm_predictions})
print(resultscnn_lstm)
```

	actual prices	cnn_lstm_predictions
MONTH		
2002-07-01	25.79	15.130135
2010-05-01	75.67	15.130106
2016-05-01	47.73	15.130121
2010-08-01	78.22	15.130079
2018-01-01	69.93	15.130136
2019-07-01	63.92	15.130138
2012-01-01	111.43	15.130079
2016-12-01	53.82	15.130146
2021-03-01	65.76	15.130075
2011-12-01	108.56	15.130102
2006-08-01	74.50	15.130157
2011-01-01	97.50	15.130074
2019-10-01	60.86	15.130145

2003-10-01	28.94	15.130127
2006-02-01	61.59	15.130122
2015-01-01	47.91	15.130152
2013-09-01	112.95	15.130138
2022-01-01	88.21	15.130163
2017-07-01	47.96	15.130148
2009-10-01	73.63	15.130110
2003-07-01	27.91	15.130145
2000-10-01	31.80	15.129995
2011-09-01	115.03	15.130093
2001-11-01	19.00	15.130167
2009-02-01	44.07	15.129953
2002-01-01	19.64	15.130157
2010-01-01	76.79	15.130112
2014-08-01	100.86	15.130159
2006-04-01	70.21	15.130104
2019-04-01	71.15	15.130099
2019-06-01	64.83	15.130153
2000-07-01	28.80	15.129469
2005-09-01	63.30	15.130130
2003-11-01	28.94	15.130136
2010-07-01	76.49	15.130072
2005-07-01	57.30	15.130160
2002-02-01	19.73	15.130150
2009-07-01	65.21	15.130083
2017-03-01	51.40	15.130140
2001-08-01	25.96	15.129831
2016-10-01	49.79	15.130119
2014-01-01	109.96	15.130075
2017-01-01	54.84	15.130148
2007-03-01	64.30	15.130100
2012-11-01	109.36	15.130089
2013-04-01	102.64	15.130129

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, cnn_lstm_predictions, color='blue')
plt.plot(y_test, y_test, color='red', linestyle='-', linewidth=1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title('Actual values vs Predicted values')
plt.show()
```



```
# Calculate RMSE on test set
cnn_lstm_rmse = mean_squared_error(y_test, cnn_lstm_predictions, squared=False)
print(f"RMSE on test set: {cnn_lstm_rmse}")
cnn_lstm_r2 = r2_score(y_test, cnn_lstm_predictions)
print(f"R² = {cnn_lstm_r2:.2f}")
```

```
RMSE on test set: 58.226776759552
R² = 3.08
```