Available online at http://scik.org
J. Math. Comput. Sci. 2024, 14:9
https://doi.org/10.28919/jmcs/8702
ISSN: 1927-5307

BRENT OIL PRICES FORECASTING USING MACHINE LEARNING

SOUMAYA BIDAH*, KHADIJA AKDIM, MEHDI ZAHID

Department of Mathematics, Faculty of Sciences and Technology, University Cadi Ayyad, Marrakesh 40.000, Morocco

Abstract. Machine Learning is one of the powrful tools which is used nowadays in prediction tasks. In this article, we will use machine learning models to predict Brent oil price, and compare their performance. Which would help investors to make the appropriate investments? decisions, especially under crisis circumstances. Several models were tested to confirm the appropriate one, starting from the very standard models, multiple linear regression, and finally try the recent developped ones, namely gradient boosting. We used as well a neural network model, Long short-term memory (LSTM) given that it approved their performance for financial series.

Keywords: time series; machine learning; prediction; LSTM; XGBoost; oil price; global energy crisis.
2020 AMS Subject Classification: 91G70.

1. INTRODUCTION

The global energy crisis of 2021-2023 emerged directly in the wake of the COVID-19 pandemic in 2021, with much of the world facing shortages and rising of the oil , gas and electricity markets. The crisis was triggered by a number of economic and geopolitical factors, including the rapid economic recovery after the COVID-19 pandemic, which outstripped energy supplies and turned into a widespread global energy crisis following the Russia's invasion of Ukraine.

*Corresponding author
E-mail address: soumaya.bidah22@gmail.com
Received June 13, 2024

Natural gas prices reached record highs, as did electricity prices in some markets. Oil prices have reached their highest levels since 2008 as in [1].

Several research articles in the literature were dedicated to analyse and predict the oil price series due to its importance in the commodities markets, and its impacts on the financial markets. The oil price could be impacted by several factors (inflation [2], fluctuation [3], crisis [7], etc.). Which make the modeling of its volatility one of the famous research subjects.

Hence, the importance of making the appropriate decision when it comes to the investments in energy in such crisis situation. Machine learning can play a significant role in oil price prediction, which can, in turn, have a substantial impact on decision-making within the energy industry and related sectors.

Generally in computer science, forecasting is an activity to predict future events by considering historic data. Several Forecasting models were subject to research studies and were used in various sectors [8], [9].

The specific objective of this article is to use machine learning models to predict the oil prices, compare their performance and showcase the shortcomings of each used model and focus on its added value to make relevant predictions in the case of Brent oil time series.

Several predictive models were tested to confirm the appropriate one, starting from the very standard models, multiple linear regression for instance, and finally try the most recent ones, namely gradient boosting models with two different variants (XGBoost and LightGBM). We used as well a neural network model, Long short-term memory (LSTM) given that it had approved its performance for various types of time series [10] and [11], and we aim at testing it for the Oil Price financial series. The comparison is carried out in terms of accuracy and training speed.

The added value of the article is mainly to confirm which one of the used models predict with high performance the crude oil prices. Therefore, help the investors to make the right oil investments' decisions. To do so we will try to compare the most recent advances in supervised machine learning (ML) and highdimensional models for time-series forecasting. We considered both linear and nonlinear models.

The paper is organized as follows: Section 2 describes the theoritical background of the used models for this study, emphasizing the diferent hyper-parameters that need to be tuned. Section 3 presents the results of the comparison per model and finally, the conclusions of the study are summarized in Section 4.

2. PRELIMINARIES

2.1. Multiple Linear Regression. The linear regression is one of the easiest and most common machine learning algorithms. It is a mathematical approach used to perform predictive analysis.

Multiple linear regression (MLR), is a statistical method that uses several explanatory variables to explain and forecast the outcome of a response variable. It aims at modeling the linear relationship between the explanatory (independent) variables and response (dependent) variables. Basically, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

The model can be expressed as:

$$Y_t = \beta_0 + \beta_1 X_1^t + \cdots + \beta_n X_n^t + \varepsilon_t$$

Where Y_t is the expected value of the dependent variable at the moment t, β_0 is the regression constant, $\beta_1 \cdots \beta_n$ represent the regression coefficient, $X_1 \cdots X_n$ are the explanatory variables and ε is the random error term at the moment t. More generally, the equation can be written as follows:

$$Y = X\beta + \varepsilon$$

The object of (MLR) is to model the linear relationship between the explanatory variables $X$and dependent variable Y.

The solution of this equation in a matrix format is as follow:

$$\widehat{\beta} = \left(X^T X\right)^{-1} X^T Y$$

Important part of the literature has revealed the relationship of correlation between oil price and other commodities price in financial markets, for instance [12], [13] and analyzed the accuracy

of linear models in predicting daily crude oil price [14]. Therefore the importance of having enough information about the future oil prices.

The limitations of linear regression are that it often explores a relation between the mean of the input variables (independent variable)and output variables. As the mean is not a full description of a single variable, linear regression is not a clear understanding of variable relationships. Therefore, an analysis of the various factors is done using Multiple Linear Regression (MLR) model. The dependent variable (target variable) is dependent on many independent variables, in this case.

In order to make sure that the prediction is accurate, we can calculate some error metrics.

Error metrics allow to quantify forecasting models' error.Hence, those metrics can help to compare the effectiveness of forcasting models. So many error metrics have been used in the literature, in this paper we will mainly use the common one defined in the following paragraph.

Suppose that Y_t is the actual value and $\tilde{Y}_t$ is the predicted value by the model. T is the sample size of observed time series data.

Root Mean Square Errors (RMSE):

RMSE can be defined as the standard deviation of the residuals values (differences between predicted values and the observed values). It is calculated as follows [15]:

$$RMSE = \sqrt{\sum_{i=1}^{T} \frac{\left(\tilde{Y}_t - Y_t\right)^2}{T}}$$

2.2. Gradient Boosting. Gradient Boosting (GB) is an efficient method for solving nonlinear classification and regression problems. The gradient boosting algorithm was proposed in 2016 and is a relatively new approach [16] Gradient Boosting is a machine learning technique used in regression and classification tasks. It creates a prediction model as a set of other weak prediction models, which are typically decision trees. Essentially, how boosting works is by adding new models to correct the errors that previous ones made.

The idea behind their using as interpretation models is that all features are sequentially considered in each iteration of boosting to learn shape function for all features [16].

In this section we will try to explain at a high-level th basic principles of Gradient Boosting and its implementation in XGBoost algorithm based on the paper by Chen and Guestrin [17]. Consider the data set $D = \{(X_i, y_i)\}$ $(i = 1 \cdots n, \quad x_i \in$ and $y_i \in \mathbb{R})$. That means as expalined in [15] that we have m features for each of n observation examples which correspond to the target (dependent) variable y. A tree ensemble prediction for a given observation i is produced as a sum of predictions from K additive functions:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in F$$

where $F = \left\{f(x) = \omega_{q(x)}\right\}$ $\left(q : \mathbb{R}^m \to T, \omega \in \mathbb{R}^T\right)$ is the space of regression trees, q represents the structure of each tree that maps an example to the corresponding leaf index, f_k is a regression tree predicting the value $f_k(x_i)$ for the i^{th} example. T is the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights ω. Unlike decision trees, each regression tree contains a continuous score on each of the leaf, we use w_i to represent score on i^{th} leaf [17].

By training an ensemble of regression trees, the model aims at minimizing the objective function with loss term (l) and regularization term Ω.

$$L(\phi) = \sum_i l\left(y_i, \hat{y}_i\right) + \sum_k \Omega\left(f_k\right)$$

With:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

where γ and λ are hyperparameters to penalize the model complexity defined by the number of leaves T and leaf weight values ω (the output scores of the leaves). The loss function (l) can be expressed in a form of the user?s interest, for instance, as the mean squared error for regression problems [15]. Higher values of γ result in simpler tree. The value of γ controls the minimum loss reduction gain needed to split an internal node.

The objective is minimizing the loss terms in an iterative manner by adding a regression tree at each iteration. This leads us to the following objective function at t_th iteration [18].

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_i)) + \Omega\left(f_t\right)$$

With $\hat{y}_i^t$ is the prediction of the $i-th$ instance at the $t-th$ iteration. As explained in [15], applying a second order Taylor expansion and removing the terms independent of f_t, it can be shown that the following approximation of L can be obtained as follows [18]:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

With g_i and h_i are the first and second order derivatives of $l(y_i, \hat{y}_i^{t-1})$ [12]. Defining $I_j = \{i|q(x_i) = j\}$ as the instance set of leaf j, a group of observations in the $j-th$ leaf in a particular tree structure and taking into account that the tree produces the same weights score for the observations in one leaf, we can compute the optimal leaf weights ω_j and the corresponding optimal value of the objective approximation $\tilde{L}^t$ as in [17] and [18].

$$\omega_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

$$\tilde{L}^t(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma Z$$

This latest equation experessing the value of $\tilde{L}^t(q)$ can be used as a scoring function to measure the quality of a tree structure q. This score is like the impurity score for evaluating decision trees, except that it is derived for a wider range of objective functions [15].

And as explained in the original article [14], it is impossible to enumerate all the possible tree structures q. A greedy algorithm that starts from a single leaf and iteratively adds branches to the tree is used instead. If we assume that I_L and I_R are the instance sets of left and right nodes after the split, and $I = I_L \cup I_R$, then the loss reduction after the split is given by the following equation:

$$L_{split} = \frac{1}{2} \left\{ \frac{\left(\sum_{i \in I_L} g_i \right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i \right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i \right)^2}{\sum_{i \in I} h_i + \lambda} \right\} - \gamma$$

This formula is used for evaluating the split candidates.

XGBoost proposes a sparsity-aware algorithm for finding the best split. The sparsity of an attribute can be caused by the presence of many zero valued entries and/or missing values. XGBoost removes automatically these values from the computation of the gain for split candidates.

Light gradient boosting machine (LGBM) is a novel gradient boosting framework proposed by Ke et al. in 2017 to address the efficiency and scalability problems of GBDT and XGB when applied to problems with high-dimensional input features and large data volumes [19].

LightGBM implementation also proposes new features, which mainly are: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [20].

Gradient-based One Side Sampling (GOSS) is used for sampling the dataset in LightGBM model implementation. GOSS is a technique used to build the training sets parts for building the base trees in the ensemble. This technique aims at incrementing the importance of those instances with higher uncertainty in their classifications. Those instances are identified as ones with higher gradient [20]. When the GOSS option is set, the training sets for the base learners are composed of the top fraction of the instances with the highest gradients (a) plus a random sample fraction (b) retrieved from the instances with the lowest gradients. To compensate for the change of the original distribution, the instances in the low gradient group are weighted up by when computing the information gain [20].

Exclusive feature bundling (EFB) is a near-lossless method to reduce the number of effective features. In the same time, EFB technique bundles sparse features into a single feature. This can be done without losing any information when those features do not have non-zero values as mentioned in [20].

Both GOSS and EFB provide further training speed gains as in [20]. Which makes XGBoost models among the best performing.

2.3. Long-Short Term Memory. Long Short-Term Memory Networks is a deep learning, sequential neural network that allows information to persist. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by RNN [19]. LSTM was designed by Hochreiter and Schmidhuber that resolves the problem caused by traditional RNNs and machine learning algorithms.More specifically, RNNs remember the previous information and use it for processing the current input. The shortcoming of RNN is they cannot remember long-term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems. LSTM can be implemented in Python using the Keras library.

Moreover, an important benefit of recurrent neural networks is their ability to use contextual information when mapping between input and output sequences [21].

The vanishing gradient problem is illustrated schematically in Fig.1 as explained in [19].

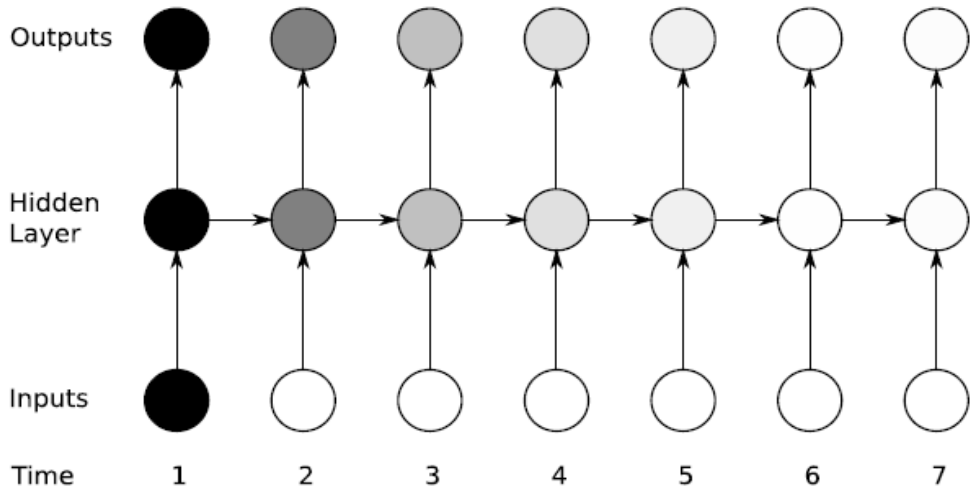

FIGURE 1. The vanishing gradient problem for RNN

As mentionned in [19] The shading of the nodes in the unfolded network indicates their sensitivity to the input data at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer (state), and the network forgets the first inputs.

Instead of neurons, LSTM networks model have memory blocks connected through layers.

The LSTM model is basically developed to overcome the drawback of the RNNs by adding a memory or cell state to the network. The added cell state is responsible for adding or removing past information conisdering its relevance and importance to make the prediction.

The LSTM model has S cell-blocks connected in series, where S is the total time-steps or length of input data.

Fig.2 presents the architecture of an LSTM with C features and D hidden units, the former C is equivalent to the number of neurons in the classical neural network.

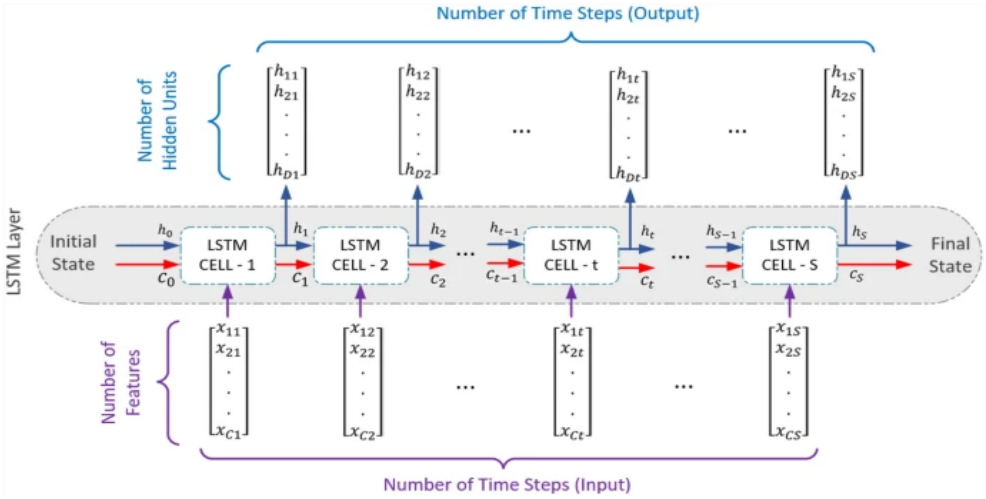

FIGURE 2. An LSTM layer with multi-inputs and multi-outputs

Each LSTM cell consists of three adjusting-gate blocks to describe its state. The gates are simple neural networks composed of weights, biases, and activation functions. The LSTM gates can be described as follows [20]:

(1) Forget gate: It determines what information from the cell state c_{t-1} (as in Figure 3) should not be considered based on information from the previous hidden state h_{t-1} and the current input x_t. To get the output vector, the current cell input x_t is multiplied by the weight matrix W_f while the previous hidden-state h_{t-1} is multiplied by the recurrent weight matrix R_f. And the resulting output are added to a bias vector b_f. Finally, a sigmoid function σ_g is activated to get the output vector f_t that has values varying between 0 and 1. The value 0 means no information from the previous time-step of cell state is allowed to flow (not important information), while the value 1 means all previous information of the memory is allowed to flow (extremely important). If the information is partially relevant, the function outputs a value between 0 and 1. The following formula describes mathematically this process as in [20]:

$$f_t = \sigma_g \left(W_f x_t + R_f h_{t-1} + b_f \right)$$

(2) Update gate: It is used to update the cell state or memory (output of the forget gate in the previous step). It contains two parts of neural networks: input gate i_t and candidate cell g_t, and are fed with the same inputs used for the forget gate $(h_{t-1}$ and $x_t)$. However, the

weights, biases, and activation functions of i_t and g_t branches are different. For the input gate branch i_t, we have the input variables x_t and h_{t-1} weighted by the matrices W_i and R_i, respectively and biased with b_i, and finally activated using a sigmoid function σ_g. The same thing for the candidate state branch g_t using the denoting letter i instead of g, and replacing the sigmoid function σ_g with a tan hyperbolic ($tanh$ or σ_s) to squishes the data between -1 and 1. The input branch is used to control the output of the squished data (the candidate state). Finally, the outputs of these two neural networks f_t and g_t are multiplied to produce the output of the update gate. Mathematically, the two networks can be written as follows [20]:

$$i_t = \sigma_g\left(W_i x_t + R_i h_{t-1} + b_i\right)$$

$$g_t = \sigma_s\left(W_g x_t + R_g h_{t-1} + b_g\right)$$

where σ_s denotes the state activation function.

(3) Output gate: It is used to get the current hidden state h_t. We apply the same for the output gate. We combined input (x_t and h_{t-1}) to a sigmoid function σ_g after multiplied with the respective weights W_o and R_o, and added to the bias b_o. The resulting output o_t is multiplied with the current cell state c_t after squished t_o the range $[-1,1]$ using the tanh function σ_s. As in [20] mathematically, the output gate can be written as follow:

$$o_t = \sigma_g\left(W_o x_t + R_o h_{t-1} + b_o\right)$$

And the new cell-state c_t and hidden-state h_t are:

$$c_t = f_t * c_{t-1} + i_t * g_t$$

$$h_t = o_s * \sigma_s\left(c_t\right)$$

The operator $*$ refers to the Hadamard multiplication (element-wise or pointwise operation.

So the forget gate allows the model to determine what information from the past memory is relevant to keep and forget the irrelevant ones. The input gate allows to update the relevant memory from the past block and generate the current memory used in the next one. And finaly,the output gate allows to compute the output of the current block and the next hidden-state.

To be noted that the three gates have the same inputs consisting of three copies of the previous hidden state and the current input combined. The LSTM memory or cell state that is used by the network to learn about the sequence order of input data.

The Fig. 3 is illustrating the different gates of an LSTM layer with multiple inputs as explained in [20].

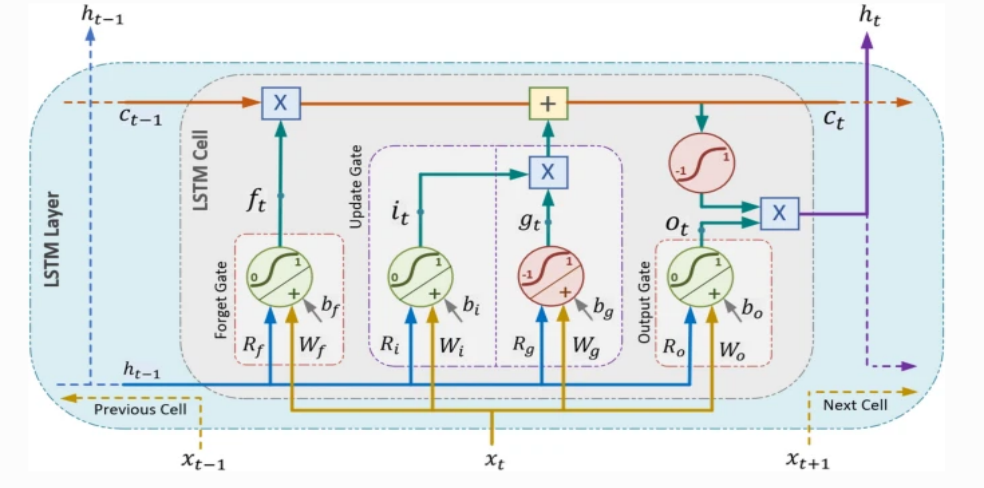

FIGURE 3. An LSTM layer with multi-inputs and multi-outputs

LSTMs can be used to model univariate time series forecasting problems (i.e. problems comprised of a single series of observations and a model is required to learn from the series of past observations to predict the next value in the sequence).

3. MAIN RESULTS

Having the right dataset is undoubtedly one of the most important condition of any prediction process. In this case scenario, the dataset we need is an archive of oil prices for the last years.

We are going to use more than twenty years of Crude Oil - Brent Europe prices starting from January 1st, 2000 up to December 20th, 2022. Please note that the used data is available via Python API: Brent-Europe ”FRED/DCOILBRENTEU”.

We are now going to proceed by visually plotting our data in the form of a graph for better analysis. Therefore the Fig. 4 prsents the historical prices of Brent crude oil for the observed period.

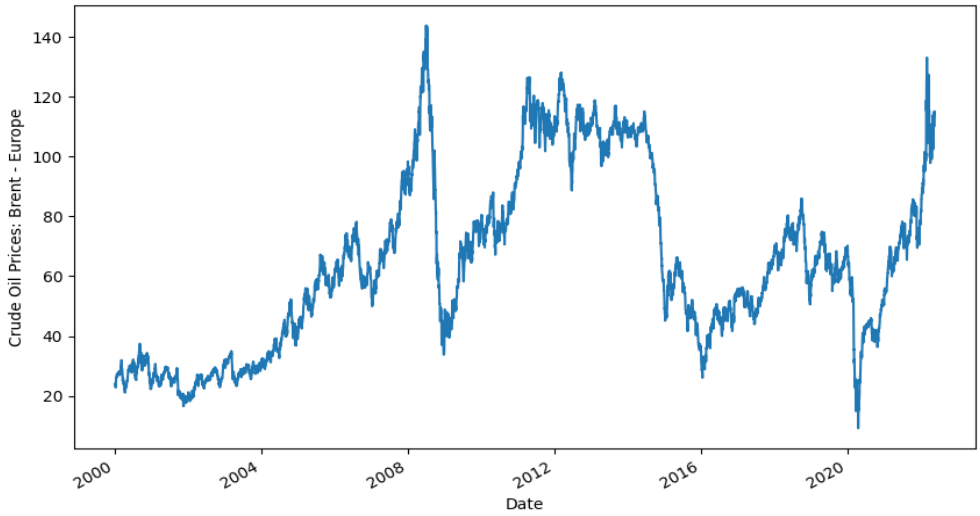

FIGURE 4. Crude Oil Prices: Brent - Europe

We can see clearaly from the graph in Fig.4 that in periods of 2008 and 2020 which coincide with the crisis times, oil prices have fallen drastically.

3.1. Multiple Linear Regression. Linear regression is the first model that we will use. We choose to use the moving averages for the past three and five days as models features. To be noted that a feature is a measurable property of an object we are analyzing. In a dataset, features appear as columns and are the different characteristics of an object. Which means that the moving average will be the explanatory variables.

After setting up the dataset, we will split our dataset into two distinct parts. We will use 80% of our data as a training set, responsible for training the model. The rest 20% will be used as testing set, used to estimate the accuracy of the model. This way and by connecting the input from the training dataset with the expected result from the testing dataset, we will be able to create a linear regression model.

The last step will be fitting the dependent and explanatory variables and create a constant and a coefficient for the regression. So to check if our model works we will plot the price that our model predicted, as well as the actual price on the same graph for comparison.

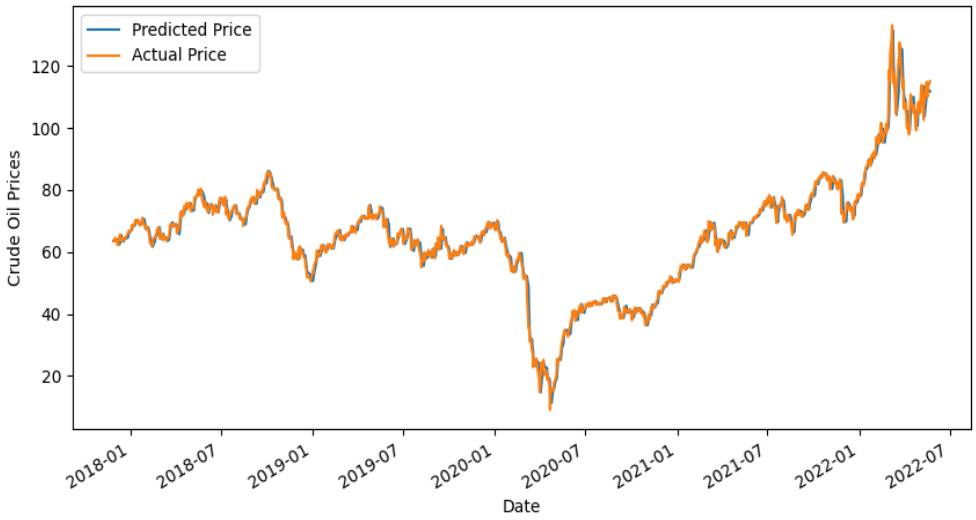

FIGURE 5. Crude Oil Predicted Prices (LR): Brent - Europe

Having a look on the graph on Fig. 5, what immediately stands out is that the linear regression based model did very well and that the predicted values are very close from the actual ones. However to confirm the visual observation, we calculated the the Root Mean Square Error (RMSE) of the model, we got a value of 1.49. Knowing that the RMSE measures the average difference between the statistical model?s predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. So the RMSE quantifies how dispersed these residuals are, revealing how tightly the observed data clusters around the predicted values.

Therefore, the linear regression model did very well in predicting the crude brent oil prices using moving averages features. Let's see how the other models will perform in predicting oil prices.

3.2. Long-Short Term Memory. In this section we will apply the Long short-term memory (LSTM) to our dataset and compare it with the previous (MLR) model. We applied the LSTM model to the same dataset that we used in the previous sections. The figure 6 below present the predicted values for used data.

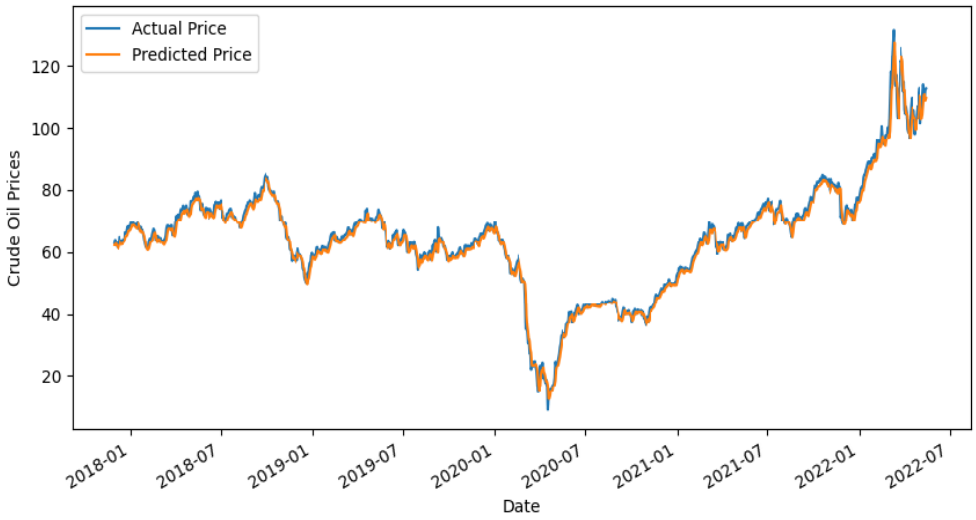

FIGURE 6. Crude Oil Predicted Prices (LSTM): Brent - Europe

Having a look on the figure 6, we can see obviously the accuracy of the predicting model. The predicted values matches well with the test data. To confirm this observation we calculated the Root Mean Square Error (RMSE) of the used model, we got 1.61 as RMSE score calculated on test data, which is a very good score for a forecasting model.

So, if we want to compare the regression models tested so far, namely the Multiple Linear Regression and the Long Short term Memory based models, we can say that both of them can forcast the crude oil very well, more specifically, the linear regression based model did better if we consider the RMSE as metric.

3.3. Gradient Boosting. We will apply the Gradient Boosting model to our dataset. The Fig. 7 below presents the result.

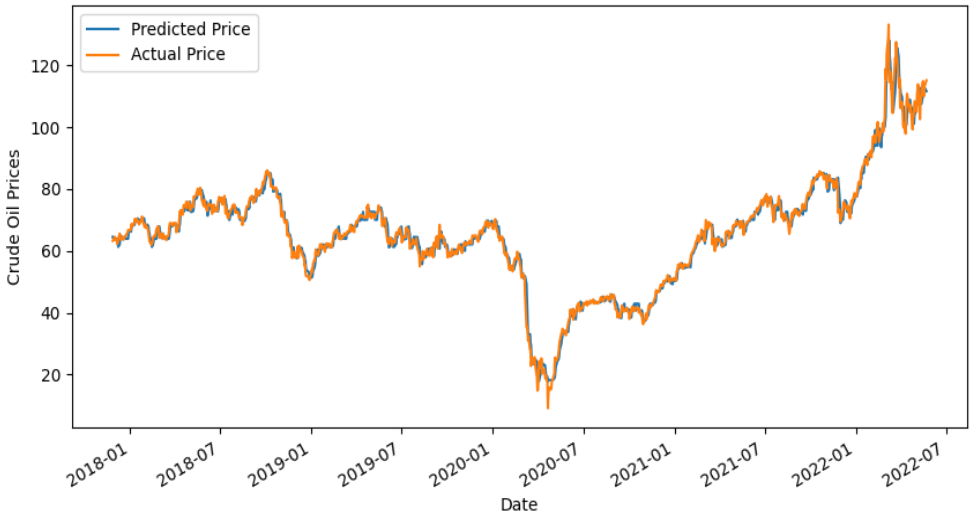

FIGURE 7. Crude Oil Predicted Prices (XGB): Brent - Europe

To be noted that XGBoost (XGB) is a type of gradient boosting model that uses regression tree-building techniques (Gradient Boosting Decision Tree - GBDT) to predict its final value. It usually requires extra tuning to reach peak performance.

The Fig.7 allows us to conclude that the XGBoost model works fine also to predict the brent oil price. To confirm this statement, we calculated the accuracy of the model which coorepond to R^2, coeffecient of determination, we get a value of 98.44%. To be noted that the coefficient of determination is a number between 0 and 1 that measures how well a statistical model predicts an outcome.

Another gradient boosting model is the light gradient boosting machine algorithm ? also known as LGBM or LightGBM. It is quite similar to XGBoost as it uses too decision trees to classify data.

LightGBM is an improved version of XGBoost with four aspects improved as mentionned in [21].

- LightGBM?s algorithm incorporates GOSS (Gradient-based One-Side Sampling) algorithm which strikes a good balance between the number of samples and precision for the decision tree. In training phase of the model, more importance to the samples with larger gradients, which also have more influence on the gain.
- LightGBM uses a histogram to identify the optimal segmentation point.

- LightGBM reduces the feature dimension to a certain extent by means of Exclusive Feature Bundling (EFB).
- LightGBM uses a leaf-wise algorithm with depth limitation instead of the traditional level-wise, which improves accuracy and prevents overfitting.

As we did for the previous models, we plotted the predicted and actual prices for the test dataset. Fig. 8 shows the results obtained.

FIGURE 8. Crude Oil Predicted Prices (LGBM): Brent - Europe

The Fig. 8 allows us to conclude that the LGBM model works fine also to predict the brent oil price. To confirm it we calculated the score of the model, we get a value of 98.56%. We can also apply the cross-validation method to evaluate the training score, we get a value of 0.94. Which it a very good score of the model.

So, from the obtained results, the LGBM model was more performant and efficient comparing to the XGB.

4. CONCLUSION

Based on the obtained results of the implemented models, we can say that the tested models, namely mulitple linear regression based model, long short term memory and gradient boosting ones have proven to be efficient and accurate to predict the Brent crude oil price.

And, looking to the performance indicators, the Light Gradient Boosting Machine algorithm (LGBM) were more performant than the XGBoost one with 98.56% accruacy and it was more efficient than the LSTM in terms of time consuming.This result was expected as LGBM is one of the trending techniques nowadays, so it comes as no surprise that this algorithm is favored in time series prediction and the machine learning community in general.

This comparative exercice allow us to confirm that there are severale machine leraning mod-oels that allow to predict the brent crude oil prices with hight accruacy.However, LGBM was the most accurate and efficient one that allows to predict perfectly the future oil prices even under crisis circumstances. Therefore, this model could be used by investors to hedge their positions on Brent oil, and to make appropriate decisions when it comes to this commodity.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests.

REFERENCES

[1] IEA Global Energy Crisis, International energy agency, Retrieved 2022-12-06, October 2022.

[2] M. Bikourne, K. Akdim, A. Khellaf, A. Ez-Zetouni, Investigating stochastic volatility and jumps in inflation dynamics: an empirical evidence with oil price effect, Eur. Phys. J. Plus 138 (2023), 1142. https://doi.org/10.1140/epjp/s13360-023-04778-5.

[3] M. Bikourne, K. Akdim, A. Ez-Zetouni, L'effet des fluctuations des prix du pétrole sur la dynamique de l'inflation au Maroc: Une approche de modélisation stochastique, Les Cahiers du Plan 56 (2023), 40–48. https://doi.org/10.34874/PRSM.CAHIERS-DU-PLAN-I56.42537.

[4] L. Maugeri, Understanding oil price behavior through an analysis of a crisis, Rev. Environ. Econ. Policy, 3 (2009), 147–166. https://doi.org/10.1093/reep/rep007.

[5] H. Zhang, J. Wang, Q. Li, Nonlinear fuzzy forecasting system for wind speed interval forecasting based on self-adaption feature selecting and Bi-LSTM, Signal Image Video Process. 18 (2023), 1249–1258. https://doi.org/10.1007/s11760-023-02759-w.

[6] Z. Liu, P. Li, D. Wei, et al. Forecasting system with sub-model selection strategy for photovoltaic power output forecasting, Earth. Sci. Inform. 16 (2023), 287–313. https://doi.org/10.1007/s12145-023-00938-4.

[7] F. Altche, A. de La Fortelle, An LSTM network for highway trajectory prediction, in: IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 2017.

[8] R. Fu, Z. Zhang, L. Li, Using LSTM and GRU neural network methods for traffic flow prediction, in: 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 2016.

[9] M.A. Naeem, M. Hasan, M. Arif, et al. Time and frequency domain quantile coherence of emerging stock markets with gold and oil prices, Physica A: Stat. Mech. Appl. 553 (2020), 124235. https://doi.org/10.1016/j.physa.2020.124235.

[10] W. Dbouk, I. Jamali, Predicting daily oil prices: Linear and non-linear models, Res. Int. Bus. Finance 46 (2018), 149–165. https://doi.org/10.1016/j.ribaf.2018.01.003.

[11] B. Lin, T. Su, Does oil price have similar effects on the exchange rates of BRICS?, Int. Rev. Financial Anal. 69 (2020), 101461. https://doi.org/10.1016/j.irfa.2020.101461.

[12] T. Chai, R.R. Draxler, Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature, Geosci. Model Dev. 7 (2014), 1247–1250. https://doi.org/10.5194/gmd-7-1247-2014.

[13] A.V. Konstantinov, L.V. Utkin, Interpretable machine learning with an ensemble of gradient boosting machines, Knowl.-Based Syst. 222 (2021), 106993. https://doi.org/10.1016/j.knosys.2021.106993.

[14] T. Chen, C. Guestrin, XGB: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, August 2016.

[15] B. Timur, J. Johannes, The idea behind their using as interpretation models is that all features are sequentially considered in each iteration of boosting to learn shape function for all features, 2019.

[16] G. Ke, Q. Meng, T. Finley, et al. LGBM: A highly efficient gradient boosting decision tree, in: Proceedings of the 31st Conference on Neural Information Processing Systems , Long Beach USA, December 2017.

[17] C. Bentejac, A. Csorgo, G. Martinez-Munoz, A comparative analysis of gradient boosting algorithms, Artif. Intell. Rev. 54 (2021), 1937–1967. https://doi.org/10.1007/s10462-020-09896-5.

[18] S. Hochreiter, Recurrent neural net learning and vanishing gradient, Int. J. Uncertain. Fuzz. Knowl.-Based Syst. 6 (1998), 107–116.

[19] Graves A., Supervised Sequence Labelling with Recurrent Neural Networks - Long Short-Term Memory, Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2012. https://doi.org/10.1007/978-3-642-24797-2.

[20] Z. Hamad, I. Abdulrahman, Deep learning-based load forecasting considering data reshaping using MATLAB, Int. J. Energy Environ. Eng. 13 (2022), 853–869. https://doi.org/10.1007/s40095-022-00480-x.

[21] D. Zhang, Y. Gong, The comparison of LightGBM and XGBoost coupling factor analysis and prediagnosis of acute liver failure, IEEE Access 8 (2020), 220990–221003. https://doi.org/10.1109/access.2020.3042848.