Long Short-Term Memory (LSTM) Oil and Energy Price Prediction Model

Roger IL Grande

SYS/EM-623 Final Project

5/5/2022

Table of Contents

Motivation

One interesting topic of interest in recent times has been: how does the world react to a crisis? The events surrounding COVID-19 have brought this question to the forefront of people's minds. More recently, a crisis has been developing between Russia and Ukraine. Russia has accused Ukraine of being against Russia's interests and they decided to take strong military action to invade Ukraine. This period of military action is considered to be one of the most significant periods on the European continent since the end of World War II. The full results of this crisis are currently unfolding, but it is clear that one effect of this invasion has been a global reaction in energy prices. Energy price changes are felt by everyone, and not just at the gas pump. All goods that people use every day around the world need to be manufactured and transported using some form of energy. I believe this topic also sheds light on the national security implications of unpredictable changes in energy prices.

My original idea for this project was to utilize supply data to train the model and relate the supply data to price fluctuations. This turned out to be extremely difficult since there was not much regularly updated global supply data or even regional supply data available online for free. Instead, I decided to select two datasets which individually represent global Brent crude oil prices and the Henry Hub Natural Gas Spot Price. The datasets were available from the U.S. Energy Information Administration.

Previous Work

My goal for this project was to practice building a simple neural network model using the best available data on global energy prices, and take a look into how this model performed. I found an interesting implementation of this type of exercise on Kaggle, in which

the model aimed to predict future crude oil prices using a Long Short-Term Memory (LSTM) network. This model used the Keras free open source Python library, which is powerful and easy-to-use for developing and evaluating deep learning models. The library also has excellent documentation online and is relatively simple to work with. I originally intended to build a model using a normal RNN, but after some more research I found that the best model to model energy prices with these types of historical price datasets was an LSTM. LSTMs typically retain a longer-term context which helps to overcome the long-term dependency problem faced by other models.

Data Description

Energy prices for one form of energy certainly have connections to prices for other sources of energy. Because of this, I made it part of my plan to take a look into multiple datasets for different energy sources and get a basic idea of how the model's results compare between the two datasets. The two datasets I chose were daily Brent crude oil prices from the U.S. Energy Information Administration, and the daily Henry Hub Natural Gas Spot Price also from the U.S. Energy Information Administration. The Brent crude dataset includes data from 1989 until now in USD, and the natural gas dataset includes data from 1997 until now in USD per million btu. I wanted to make sure I had large enough datasets for the model to be somewhat meaningful, and these were some of the largest free energy price datasets I could find online. Both datasets are comprised of dates and the corresponding energy prices on each date.

For the predictor and response variables, I chose to go with the recommended split on the original Kaggle model. The model uses 70% of the data for training/predictor

variables and the remaining 30% for the testing/response variables. The training set contains a known output and the model learns on this data, and the testing set is used to evaluate the models performance using what it learned from the training data. Before the training even began, I found it useful to know that the natural gas price data generally followed the crude oil price data at some periods in time with some similar price spikes and declines, while keeping in mind the two datasets do not capture exactly the same time period of data. There was a clear spike around the time period of the Russian invasion as well. I used Python to output simple plots of the two datasets which are shown here:

Figure 1 (above): Daily Brent Crude Oil Dataset

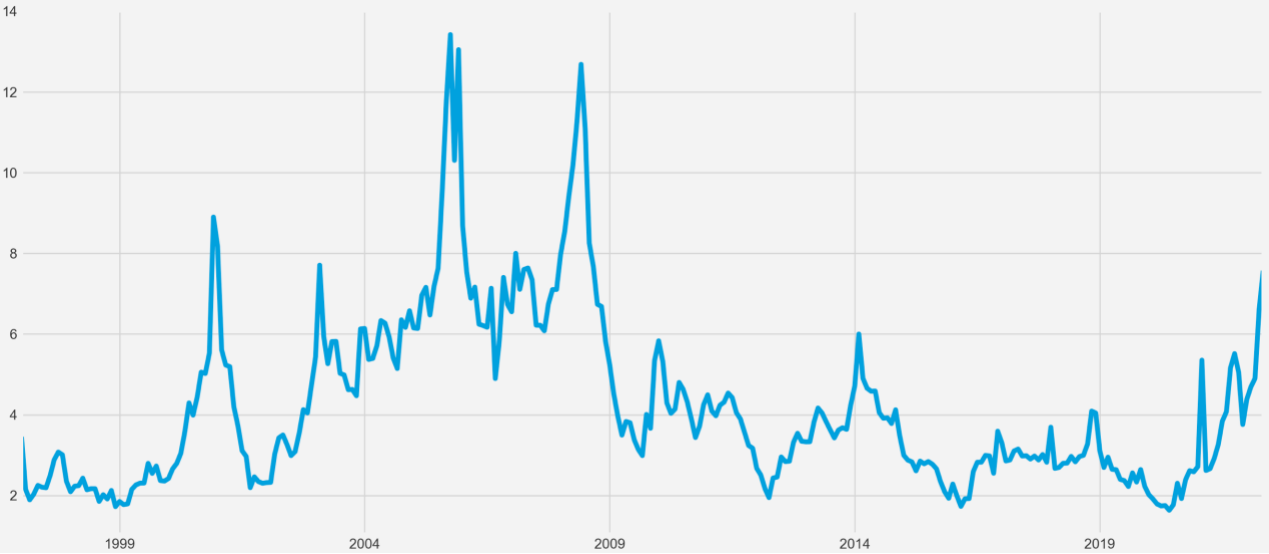

Figure 2 (above): Daily Henry Hub Natural Gas Spot Price Dataset

Data Pre-Processing

There were two minor steps for pre-processing required to ensure the data would be formatted correct for the model. The first step was to read the csv data into a dataframe, and parse the data using the "Data" column of each dataset. The parse_dates field of the pandas read_csv function serves to infer the format of the datetime strings in the columns, and if it can be inferred, switch to a faster method of parsing them. This is a known method to possibly increase the parsing speed. The second step in pre-processing the data was to sort the values by date and ensure the data was grouped by date and price. I then printed out a sample of the data and checked the datasets for null values.

Machine Learning Model & Performance

The model I chose to use for this project was an LSTM, which is a type of Recurrent Neural Network (RNN). There are several benefits to LSTM models. LSTMs process entire sequences of data without treating each point in the sequence independently of one another.

They instead retain useful information about previous data in the sequence to help with the processing of new data points. They are also technically more sophisticated than RNNs due to their gate structure, and can learn a pattern that exists every 12 periods in time. LSTMs retain a longer-term context which is beneficial in overcoming the long-term dependency problem faced by other models. As far as the inner workings of the model, LSTMs use a series of gates which control how the information in a sequence of data comes into, is stored in, and leaves the network. There are three gates in a typical LSTM: the forget gate, the input gate, and output gate. The forget gate decides which bits of the cell state (long term memory of the network) are useful given both the previous hidden state and new input data. The input gate determines what new information should be added to long-term memory (cell state), considering the previous hidden state and the new input data. The output gate decides the new hidden state using the newly-updated cell state, the previous hidden state, and the new input data.

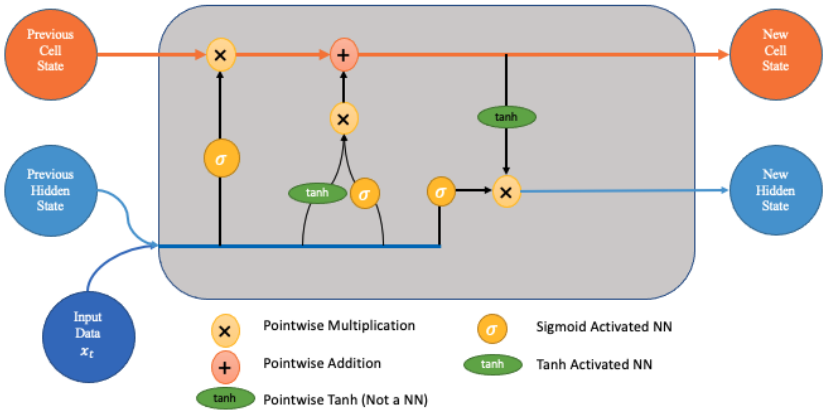

Figure 3 (above): LSTM Network Gates. Source: towardsdatascience

The performance of this model using the two input datasets was best evaluated through the model loss, and the model area under curve (AUC). I tried outputting the

accuracy metric from Keras and I was getting values of zero. I found through documentation that because the model uses loss = 'mean_squared_error', which indicates a regression setting, using accuracy as a metric is not appropriate. I set the model up to show the loss and AUC live during training through 20 epochs, and then output plots of the final results from the training when the model is finished running.

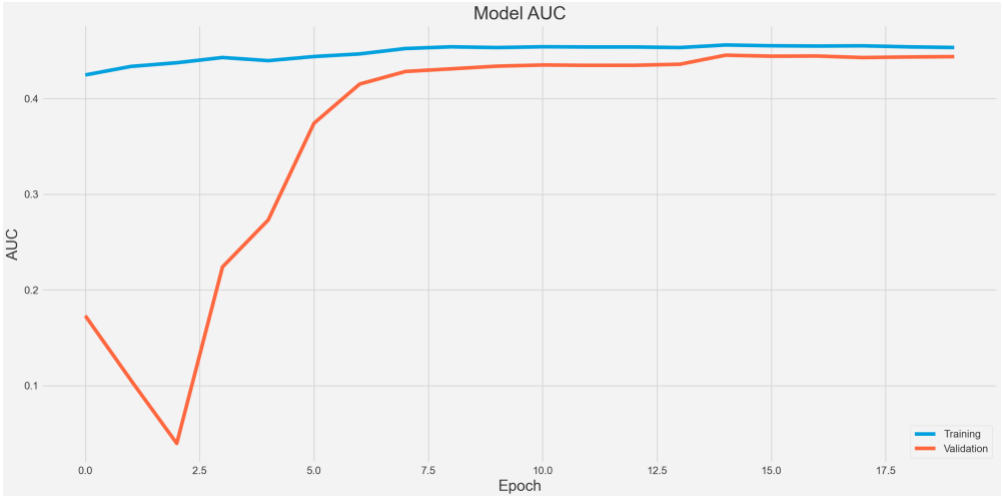

Figure 4 (above): Model Loss with Brent Oil Dataset

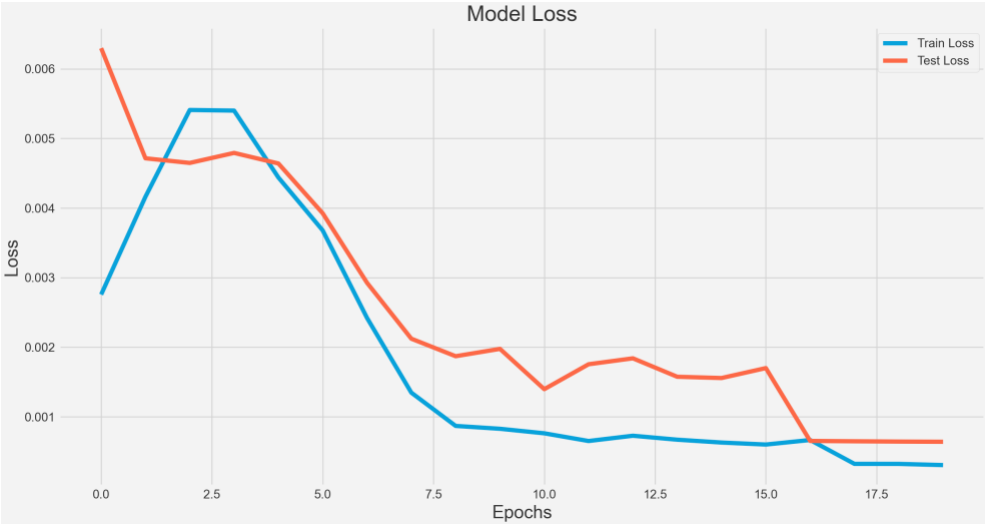

Figure 5 (above): Model Loss with Natural Gas Dataset

Loss in machine learning applications is defined as a number indicating how bad the model's prediction was on a single instance. It is a summation of the errors made for each example in training or validation sets. Ideally, I expect a reduction of loss as the number of iterations increases. A perfect prediction would indicate a zero loss, and then the loss goes up from their depending on the difference in the prediction compared to the actual value at that instance. The losses in the model when taking each dataset separately as an input appear to be fairly good in general. The closer to zero, the better and both datasets inputted into the model generally result in a reduction of loss from one epoch to the next.

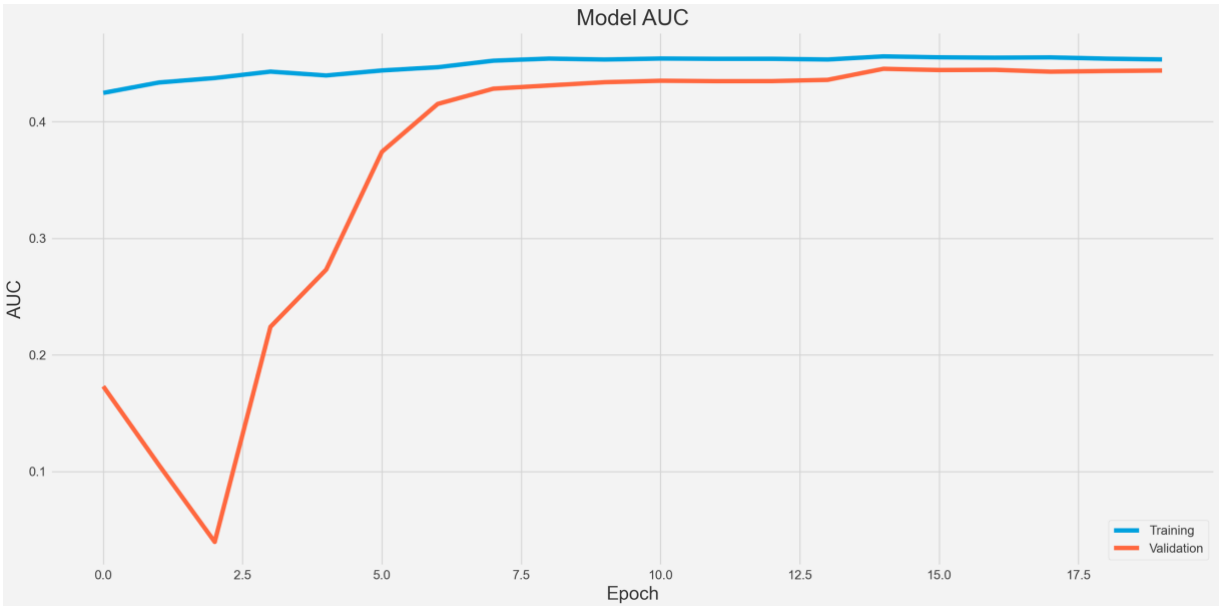

Figure 6 (above): Model AUC with Brent Oil Dataset

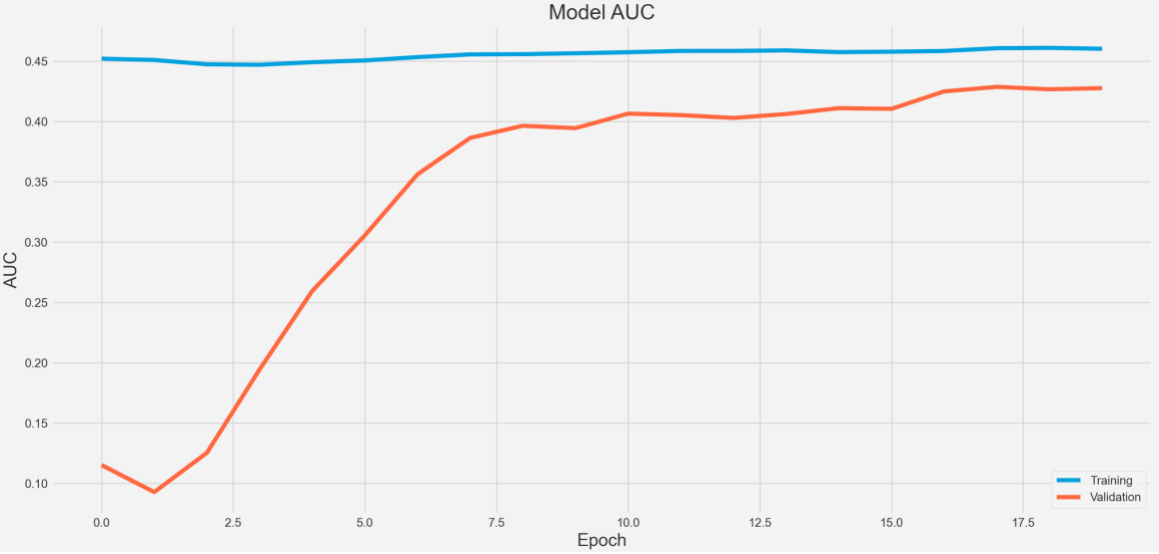

Figure 7 (above): Model AUC with Natural Gas Dataset

The next metric to look into is area under curve (AUC). The AUC measures the ability of a classifier to distinguish between classes and is a summary of the ROC curve. A higher AUC indicates better performance of the model at distinguishing between the positive and negative classes. A model with entirely incorrect predictions has an AUC of 0.0, and a model with 100% correct predictions has an AUC of 1.0. I was looking for an AUC of at least 0.7 for it to be considered acceptable. The results from this model for both of the input datasets shows a fairly low AUC. The reasons behind this in this specific model are something I will continue to investigate, but I suspect that even larger datasets would have improved the model's performance in this metric. Regression algorithms such as this one are used to predict continuous values such as price, and classification algorithms are used to predict and classify discrete values. Confusion matrices are useful in classification models, so it was not useful to produce a confusion matrix for this model.

Discussion

After the data pre-processing section of the code, there are then plots that output for the full input dataset, the trendline, seasonal spikes/dips, and residual values. The data is then normalized using MinMaxScaler, and then split into test (30%) and train (70%) sets. The lookback value defines how many previous timesteps are used in order to predict the subsequent timestep, and I set lookback to 90. To build the model, I used the Sequential() model from Keras. This is a good model for a plain stack of layers where each layer has exactly one input tensor and one output tensor, and to create it I passed a list of layers to the Sequential constructor. I then included some code to customize the calculated AUC value since I was having issues with the default value that Keras can produce. When compiling the model, I used the Adam optimizer. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. It is known to be computationally efficient, it has little memory requirement, it is invariant to diagonal rescaling of gradients, and is beneficial for problems that are large in terms of data or parameters. The loss is set to mean_squared_error, which computes the mean of squares of errors between labels and predictions. Loss functions compute the quantity that a model should seek to minimize during training. The model is trained with a epochs = 20 and batch_size = 15. I then outputted Train Mean Absolute Error, Train Root Mean Squared Error, Test Mean Absolute Error, and Test Root Mean Squared Error at the end of the run, as well as the plots for model loss, AUC, and the model's actual vs predicted values.

Figure 8 (above): Model Actual vs Predicted Values for Brent Oil Dataset

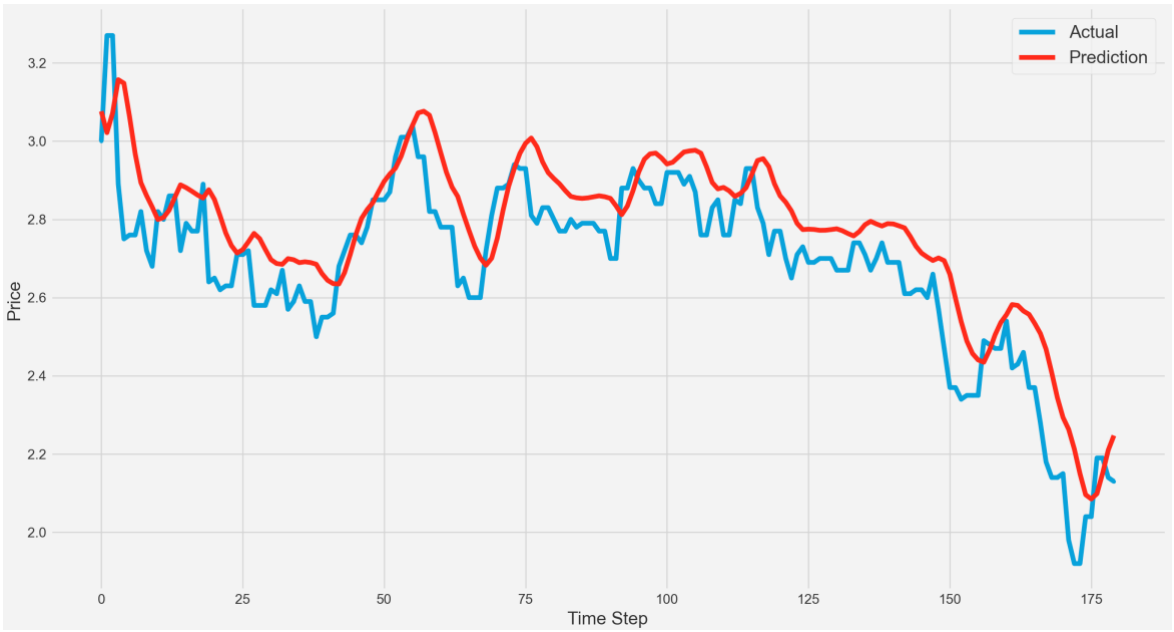

Figure 9 (above): Model Actual vs Predicted Values for Brent Oil Dataset

Brent Oil Price Prediction

Train Mean Absolute Error: 3.302000776302794

Train Root Mean Squared Error: 3.971374126681731

Test Mean Absolute Error: 3.397215876181023

Test Root Mean Squared Error: 7.363063619294272

Henry Hub Natural Gas Price Prediction

Train Mean Absolute Error: 0.2034399219842049

Train Root Mean Squared Error: 0.37406855160327845

Test Mean Absolute Error: 0.17819570561828948

Test Root Mean Squared Error: 0.608944490277766

There is definitely room for improvement in the model even though it appeared to predict the prices fairly accurately when comparing the actual and predicted values. I will continue to optimize this model further by adjusting all of the different parameters in different combinations, and also with different datasets (and larger datasets if I can locate them). However, I was very happy with the actual vs predicted results, as the prediction clearly tracked fairly close to the actual data in this sample time period. I'm still learning how to put these models together, but I believe this was a valuable project overall to start exploring this relevant topic of energy prices.

Summary

During this course, I learned from a starting point of having very little knowledge of the actual workings of machine learning algorithms. Before this course I knew of many of their applications, but the technical implementations of these models is very different from what I imagined that they would be. Extracting features and patterns from data is a very important skill that I will continue to work on, as I believe the skills introduced in this class are essential for me as an engineer working into the future. There were many examples in this course that proved how useful it is to use a software tool like Python to simplify large datasets, which are meaningless to look at without processing through algorithms. There was so much information covered in this course, and I know I have to keep studying it on my own in order to keep improving my skills. It is important to not only be familiar with data mining and machine learning concepts, but also be familiar applying them in useful scenarios in the real world. The best learning for these topics definitely comes through application. I am happy that I was able to learn and apply some basic machine learning methods in this final project, and I look forward to expanding on it and improving it more.

References

Dolphin, R. (2021, December 12). LSTM networks: A detailed explanation. Medium. Retrieved May 5, 2022, from https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

Europe brent spot price fob (dollars per barrel). (n.d.). Retrieved May 5, 2022, from https://www.eia.gov/dnav/pet/hist/rbrteD.htm

Henry Hub Natural Gas Spot Price (dollars per million btu). (n.d.). Retrieved May 5, 2022, from https://www.eia.gov/dnav/ng/hist/rngwhhdm.htm

mahmoud87hassan. (2019, October 23). Predict future crude oil prices using LSTM network. Kaggle. Retrieved May 5, 2022, from https://www.kaggle.com/code/mahmoud87hassan/predict-future-crude-oil-prices-using-lstm-network/notebook

Team, K. (n.d.). Keras documentation: Optimizers. Keras. Retrieved May 5, 2022, from https://keras.io/api/optimizers/