Stock Price Predictions: Using Neural Networks in the Market for Commodities

Oakley Browning, Michael Kenny, Eva Pierre-Antoine
APMTH 115: Mathematical Modeling

Abstract

This paper investigates the optimal machine learning model, focusing on neural networks, to predict equity prices. We hypothesized that understanding the inputs, outputs, and future contracts of a commodities company could better map arbitrage in equity. To test this, we focused on ExxonMobil's operations in transforming crude oil into gasoline, initially using a standardized Long Short-Term Memory Network (LSTM). This model underperformed due to excess features and an inflexible architecture. Shifting to a simplified LSTM with fewer features and enhanced preprocessing, we observed significant accuracy gains. Finally, we pivoted to Transformer models and ensemble learning, which effectively captured long-range dependencies and feature interrelationships. Despite outperforming the traditional LSTM, this advanced model fell short of the simplified one-feature, one-output approach. These results reflect the efficient markets hypothesis, suggesting that factors like supply shocks and future expectations may already be priced in, leaving historical prices as the key driver of predictive power.

1 Introduction

Trying to systematically "beat" the stock market is an incredibly hard task; "the average investor does terribly, and professionals on average do okay but still underperform" [1]. We saw this first hand in our second mini-project where we attempted to use a pairs-trading model to profit off of the spreads between stock prices for stocks that tended to move in the same direction price-wise. Ultimately, we decided that this macroscopic view proved too difficult as potential idiosyncratic risk factors skew results, which is why we have turned to a more narrowed down approach in this project. Importantly, the problem that we are trying to solve is the same: how can an average investor make excess returns on the market? However, we will now try to accomplish this feat by focusing first on building an accurate stock price predictor for one equity in a specific market, with hopes that our general model will be generalizable and scalable.

Specifically, we now attempt to leverage commodities markets, which are markets for buying, selling, or trading natural resources [2]. Commodities markets offer great opportunities for investors, allowing trades on the stock utilizing the input, the input itself, the output, and associated futures or options across all levels. In this analysis, we focus on crude oil, gasoline, and ExxonMobil (a natural gas company that primarily uses crude oil as an input to manufacture gasoline). We begin by gathering data on historical and future prices of oil and gasoline, enabling the calculation of both spot (current) and future crack spreads. The crack spread (defined mathematically in Section 2) measures the profitability of the refinement process. Futures contracts, which are agreements to buy or sell a commodity at a predetermined future date [3], complement this analysis. Together, these spreads influence ExxonMobil's current and anticipated profitability.

Along with the crack spreads, we employ other macro and technical features, including: the implied and realized volatility for crude oil prices, lagged prices of the equity itself, statistical indicators of the equity, gauges on inflation [4], the strength of the dollar, and the energy sector as a whole. In simple terms, there are countless factors that influence the price of commodities, which ultimately dictates the value of companies trading in them. We hypothesize that with this combination of time series data, implementing a neural network known as the Long Short-Term Memory Network (LSTM) [5], will allow us to find patterns in the data that can be used to predict the future share price of ExxonMobil. We could then make trading decisions today that reflect our expectations of what the price will be in the future. As we demonstrate in this paper, this approach has

shortcomings, prompting us to explore a more advanced transformer model to better predict the equity's price movements.

In summary, actively investing in individual stocks is inherently challenging, and our attempts at a pairs trading model were hindered by excessive volatility. Here, we aim to leverage the detailed information available in commodities markets. If our model successfully forecasts the future share price of a specific stock, it could be generalized to other stocks within the same market and extended to additional commodities markets. While these extensions lie beyond the scope of this analysis, they highlight the vast potential of this approach, which begins with constructing a fundamentally sound model at the microscopic level.

2 Model Description

2.1 Data Sources and Preprocessing

We obtain crude oil prices from a Quandl-provided dataset of OPEC, an intergovernmental organization that coordinates and unifies petroleum policies among member countries. We select data starting from January, 2003, capturing daily crude oil prices in USD. We then retrieved data from the U.S. Energy Information Administration (EIA) for reformulated regulated gasoline - the standard gasoline used in the United States for cleaner combustion since the 1990 Clean Air Act amendments. We use the FOB (Free on Board - cost of gasoline at the point of delivery) spot prices as they are the price that the buyer and seller have agreed to buy and sell the commodity. We convert prices from dollars per gallon to dollars per barrel, as barrels are the standard unit for gasoline in the industry. The crack spread is calculated as:

$$\text{Crack Spread} = P_{\text{gasoline}} - P_{\text{crude}} - \text{Refining Costs}.$$

Using a rolling window algorithm, we normalize the crack spread with z scores using its mean and standard deviation (the number of standard deviations from the mean). This function is applied to the imported crude oil and gasoline prices to calculate the standardized crack spread. We repeat this process with futures data retrieved from Yahoo Finance, creating a dataframe with dates and prices. Futures prices for gasoline are adjusted by multiplying by 42 to convert from dollars per gallon to dollars per barrel. Using the same method, we calculate the crack spread for futures. We finally compute the spread between spot and futures prices for crude oil and gasoline, along with their rolling-window Z-scores. Additional technical and macroeconomic features, all normalized, include:

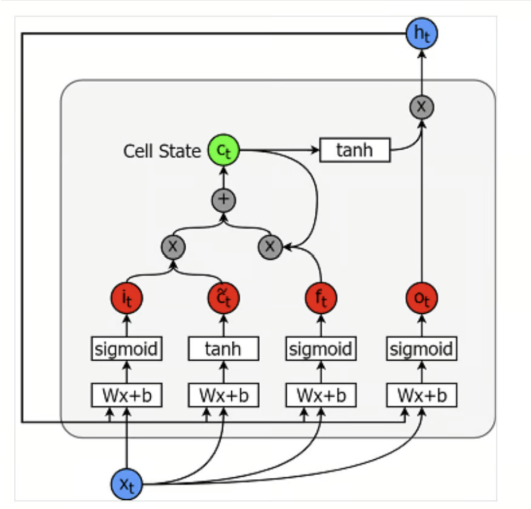
- Volatility Mismatch – The CBOE Crude Oil Volatility Index (OVX) database, representing implied volatility (expected future fluctuations in crude oil), is used to calculate the difference from realized volatility (actual fluctuations in crude oil prices) to compare market expectations with actual price movements. With rolling windows, we are able to calculate the volatility mismatch and its Z-score normalization.
- Relative Strength Index (RSI)– An indicator measuring the speed and change in price movements to evaluate underbought and oversold assets.
- Moving Average Convergence Divergence (MACD) – A momentum indicator showing the strength, direction, and momentum of a stock's price trend.
- Signal line – Calculated from the MACD to identify buy and sell signals in short and long-term price trends
- Inflation Proxy – The 10-year treasury note index, a measure of long-term stability backed by the government, is often used as a benchmark for overall market trends. From this, we calculate an inflation proxy by determining the daily percentage change in inflation rates.
- Energy Sector Proxy – From the Energy Select Sector SPDR ETF (an exchange-traded fund tracking the U.S. energy sector), we calculate the daily returns of the XLE ETF to find the percent change of the energy sector's performance.

- USD Strength Proxy – From the US Dollar Index we compute the day-to-day percentage changes in adjusted closing price. This provides a proxy for the strength of the USD relative to other currencies, which is crucial given the global nature of the oil and gas market.
- Lagged Closing Prices – We lag XOM's closing prices by a day and calculate their percentage change.

2.2 LSTM Architecture

To predict model stock price movements, we employed a Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN) designed for sequential data, inspired greatly by Datacamp's Stock Market Predictions with LSTM in Python [6].

Neural networks are trained using backpropagation, a process that adjusts internal weights by propagating errors backward through the network. In RNNs, backpropagation computes gradients across multiple time steps with the chain rule of integration. However, gradients often diminish exponentially as they move backward, a phenomenon called the vanishing gradient problem, which prevents the network from learning long-term dependencies. LSTMs address this with a cell state, which preserves long-term information across time steps, and a forget gate, which discards irrelevant information, allowing the model to effectively capture extended dependencies in financial time series.



The complete architecture of LSTM models consists of layers, each a stack of interconnected neurons. Each layer processes sequential data, maintaining a "memory" using five key components:

1. Cell State – a persistent memory element that stores long-term information
2. Hidden State – the output of the layer, influenced by both current input and the cell state.
3. Input Gate – regulates how much of the input updates the cell state.
4. Forget Gate – controls which parts of the cell state are discarded.
5. Output Gate – determines how much information flows to the hidden state.

We split our data using an 80/20 train-test ratio. We decided on a lookback window of 45 days, which is the number of past time steps the model used to predict future values. Next, we needed to define the model's hyperparameters, settings that define its structure and training process but are not learned during training. These include:

1. The number of neurons per layer defines how much information the model can store and process at each time step, directly influencing its capacity to learn patterns.
2. The number of layers specifies how many sequential LSTM cell stacks are used, enabling the model to extract hierarchical features.
3. The dropout rate prevents overfitting by randomly deactivating neurons during training.
4. The learning rate controls the magnitude of weight updates during backpropagation. Smaller rates allow finer adjustments but slow convergence, while larger rates risk overshooting optimal values
5. The batch size is the number of samples processed simultaneously in each iteration

Manually selecting these risks suboptimal performance due to the large search space and interdependencies. Instead, we used Optuna, which optimizes hyperparameters via Bayesian methods. Unlike grid search, which exhaustively tests fixed combinations, or random search, which samples configurations blindly, Bayesian methods prioritize exploring high-potential regions of the hyperparameter space. Through Optuna, we identified an optimal configuration: 178 LSTM units, 3 layers, 0.428 dropout rate, 0.0029 learning rate, and a batch size of 54.

The model was implemented using Keras, a framework that simplifies building and training neural networks. A sequential architecture was employed, meaning layers are stacked in order, with each layer feeding into the next. The layers were designed as follows. (1) Input Layer – configured to match the dimensionality of the input data (features and lookback window). (2) LSTM Layers – as according to the hyperparameters, three stacked LSTM layers each with 178 units were included to extract temporal patterns (3) Dropout Layer – placed after the LSTM layers to reduce overfitting. (4) Dense Output Layer – A single neuron was used to predict the normalized stock price, transforming the learned features into a single scalar output.

The model was compiled with the Adam optimizer, which adaptively adjusts learning rates to improve convergence, and trained using the mean squared error (MSE) loss function. Training was conducted over 50 epochs, where one epoch represents a complete pass through the entire training dataset. We finish our model by using our predicted prices to make trades, by predicting the prices of XOM using lagged equity; if we expect XOM to rise, we buy the stock, and vice versa. Comparing the actual and predicted percentage changes, if this percentage change is positive or negative, we trade on its direction (long or short selling). Based on what is concluded, we calculate the daily profit or loss and update the cumulative trading profits

3 Analysis

3.1 Standardized LSTM Model

In our first attempt to predict the price movements of ExxonMobil, we use all of the features explained in the model description above. A DataFrame with all of these "predictive" features is shown below:

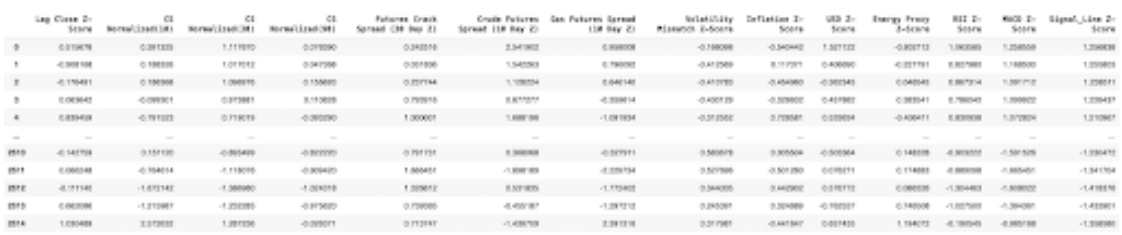


Figure 1: Normalized Features that are fed into the LSTM Training and Testing Model. All features are on the same scale and correspond to their historical values.

The question now is why do we believe that these features will have predictive power over the price of the equity? The lagged closing prices are obvious, but what about something like the crack spreads or inflation? Essentially, we believe that a wider crack spread would signal that there are more profits to be made for ExxonMobil as their profit margin is increasing. We would thus expect these higher profits to be factored into the stock price under the Efficient Markets Hypothesis. Figure 2 to the right shows the relationship between the spot (current) and future crack spreads in relation to the price of the equity. It is not definitive by any means, but over the course of this selected month (ignoring noise), there is some evidence that the price of XOM rises and falls in response to increases and decreases in the spot crack spread. Moreover, there seems to be a lag between the price of the equity increasing or decreasing as a response to the future crack rising or falling.

As for inflation, we know that "the revenues of energy stocks are naturally tied to energy prices, a key component of inflation indices. So by definition, they generally have performed well when inflation rises" [7]. The Nasdaq website itself states how a stock like ExxonMobil is about "as good as it gets if you're trying to beat inflationary pressure [8]. Thus, through our research, we believed that these proxies would be the best

features to feed into the neural network to predict price movements.

More specifically, we used feature data spanning from 2014 to 2024, and trained on the first 80% of the data, leaving the last 20% as our test sample. We normalized each feature by its corresponding z-score as well as the target variable, which is the price of the stock. Then with our optimal hyperparameters chosen computationally, we were able to predict the price of ExxonMobil (in terms of its z-score) across nearly 450 days, ranging from 1/19/2022 to 11/13/2023. Figure 3 below shows the results. The first observation is that the predicted normalized prices significantly differ from the actual normalized prices in magnitude. We attribute this discrepancy to the normalization process. While normalization posed a risk, we deemed it essential for the neural network to identify patterns within the feature data effectively. Thus, using these results, it would be impossible to trade based on precise estimates of what the stock price will be in the future. Nevertheless, the general direction of the predicted price fluctuations may suggest that there are opportunities to use these forecasts to make future trading decisions (see Figure 3).

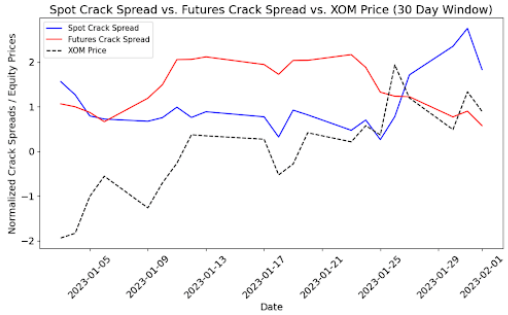


Figure 2: Relationship between Crack Spreads and ExxonMobil.

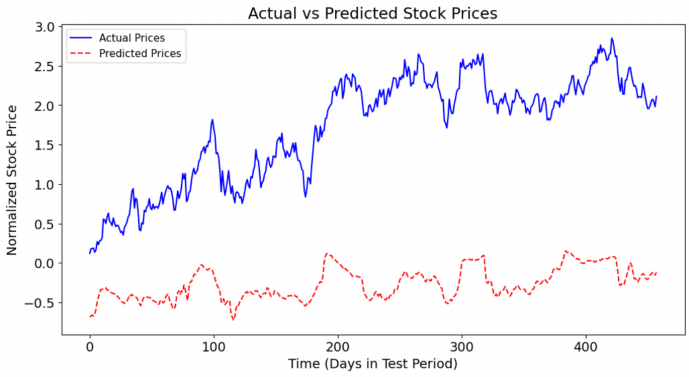


Figure 3: The relationship between the predicted and actual prices of ExxonMobil during the test period, with the prices being in terms of their normalized z-score. We note that the magnitudes of the z-scores are not aligned, potentially due to the normalization occurring separately to all of the features before the training of the LSTM model. Nevertheless, there does seem to be a slight qualitative correlation between the changes in prices that our model predicts and the actual price changes. We can then attempt to make daily trades based on the direction we anticipate the equity to move in. A major downfall is that only trading on direction does not indicate our confidence in that actual movement occurring.

Based on the direction we expected the daily prices to move, we simulated the profits that we would have made if we invested $100,000 every day into this single stock, either buying or shorting it. We found that the results were mixed. Out of the 458 total trading days, we traded in the correct direction 226 times and 232 times in the incorrect direction. This suggests that the daily predictions are not fine-tuned enough to be a reliable source of investment decisions. Even though it appears that the general trend of the prices move similarly, on a day-to-day basis, we are opening trades that lose us money roughly half of the time. This is more clearly seen in Figure 4, which shows a histogram of the daily profits and losses throughout the time period. The distribution of profits appears to be somewhat normally distributed, with most trades profiting or losing roughly 1-2% of our daily bankroll. Therefore, it was rare for there to be massive swings in profits and losses, but it was also very tricky to be confident that we were opening winning positions every day. Figure 5 then graphs the cumulative profits made off of trading the stock over time. We see that at first, the profits are extremely high, which suggests that our model may have picked up on trends in the stock price. However, these profits are then followed by a series of losses, and gains, repeating throughout the time period. In the end, our total cumulative profits amounted to $27,218. Therefore, the profits that were made on correct trading decisions slightly outweighed the incorrect trades made.

With this in mind, there are three possible reasons as to why our predictions varied so widely in terms of their accuracy. First, is that the features we used, such as the crack spread, the technical indicators, and the other economic proxies are not strongly enough correlated to the price changes that occur with ExxonMobil. If

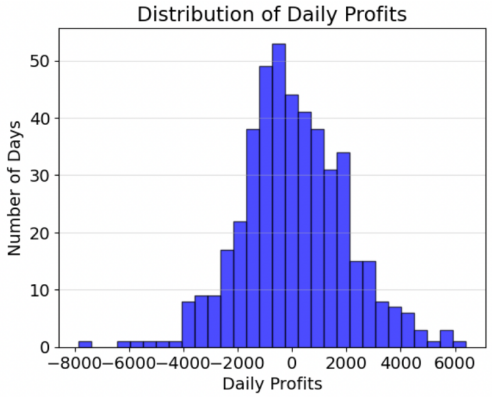


Figure 4: Profit gains and losses across the test period (roughly symmetric).

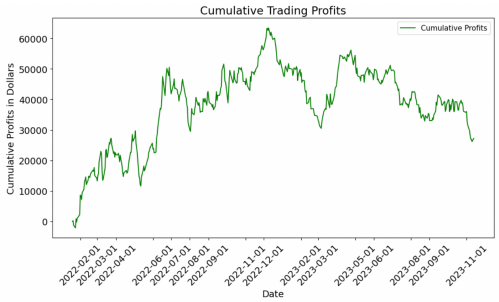


Figure 5: Cumulative trading profits vary sharply over time, but end up being positive.

this is the case, then regardless of the hyperparameters or normalization techniques we use, the predictions will inherently be flawed. This is plausible, but it also could be the case that our mixed results were due solely to the normalization of each feature individually, while also normalizing the target price. There is room for error here, but through countless iterations we performed with this model, we realized that without scaling the features and target variables, the results become catastrophically worse (usually resulting in a flat line price prediction). Lastly, it could be the case that the LSTM network, when fed so many features, "is not flexible enough and cannot model very complex relationships in sequence data" [9]. We first look to see if a simplified version of this model improves our performance before then attempting to turn to a more sophisticated Transformer model.

3.2 Simplified LSTM Model

Due to the varying nature of the results above, we pivoted to see if an extremely simplified version of the same model would yield better results. Instead of basing the future price movements on spreads and economic indicators, we decided to base the price predictions just on the percentage change in the lagged closing prices for ExxonMobil. Our target variable then became the percentage change in closing prices for the equity. This way we were avoiding the issue of having to normalize features individually as well as evading the potential for the features we chose to not be representative of the stock price movements. Figure 6 shows results that are much more in line with what we hypothesized would be the case when basing the predictions off of the entire set of indicators. We see that throughout the test period, the predicted direction of the percentage changes matches strikingly with the actual percent changes.

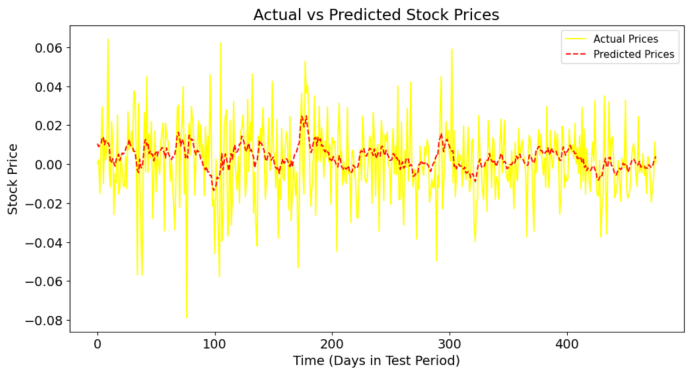


Figure 6: When training and testing the neural network on solely the percent change in lagged closing prices, our predictions are much more accurate and less volatile. The direction of the stock price changes moving mostly in unison suggests that the LSTM model was able to pick up on the underlying patterns in the time series. Compared to the previous model, we can now more confidently begin to make trades. This could potentially suggest that the LSTM model can only handle fewer time series or that the normalization of features, if not done one-hundred percent accurately, can significantly skew the overall results of the model's predictions.

Aside from hypothetical profits that can be made trading based on these predictions, one of the most crucial evaluation metrics is simply the percentage of the time the trades that were opened were in the right direction. In the more robust model above, our accuracy was around 50%. However, with this model, our accuracy is just under 65% (with 306 out of 476 trades being made in the right direction), which is a significant improvement. Moreover, we can then repeat the same process as we did above to analyze the profits that could have been

made if we traded based on our predictions. Our results show that not only are there more days where we correctly long/short the stock (see Figure 8 and 9), but also that we get the trades right when it matters the most. The cumulative profits increase approximately linearly with time, with losses on certain days not able to cause any significant reductions of profits across the test period as shown in Figure 7. Specifically, across this roughly 2-year period, we made cumulative gains of approximately $241,779. Compared to the cumulative profits under the previous assumptions, this is a nearly 788% improvement in overall performance.

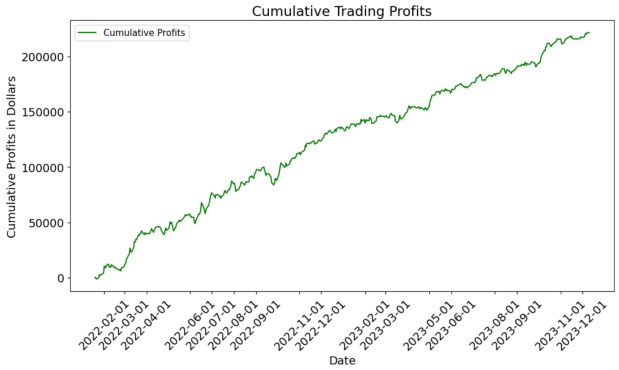


Figure 7: The cumulative profits increase approximately linearly, leading to massive returns.

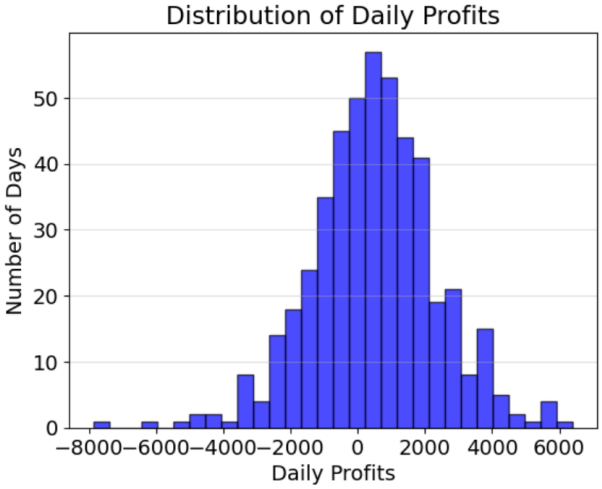


Figure 8: Mean and median of the daily profits both increase here.

Therefore, with this simplified approach, we are better able to predict the price of ExxonMobil days in advance. This is a major step toward accomplishing our goal of being able to earn excess returns in the stock market, but we have some lingering questions. Why is it the case that training our neural network on just one feature performs better than when we feed it a handful of predictors that should indicate the direction of the stock's price movements? Similar work has been done using LSTM models and approaches vary widely.

Similar to the results here, a network that used just the daily price data of Google to predict its price the next day performed very well [10]. This follows more of a technical analysis approach where estimates of future prices are based solely off of past and current prices [11]. Thus, aside from the potential profits that can be made from trading with this simplified LSTM model, this may also be evidence suggesting that the technical analysis approach can actually be a viable alternative to complicated methods that attempt to price an equity using other metrics aside from just the past prices. Almost like the other features we are testing are priced into the stock price – the efficient market hypothesis. Last project we attempted to trade based off of mean reversion assumptions for cointegrated stocks and here our initial thought was that the more indicators we use to trade on, the better our results will be. However, the results above suggest that simplicity may be the key to successful trading strategies, even when stocks are subject to multitudes of idiosyncratic and systematic risk factors every day. Nevertheless, we still wanted to gain a better understanding of if our initial model failed due to poor feature choices, bad normalization practices, or because the LSTM model could not capture all of the patterns present in the time series. As a result, we attempted to explore a more sophisticated, and challenging, deep learning model explained below.

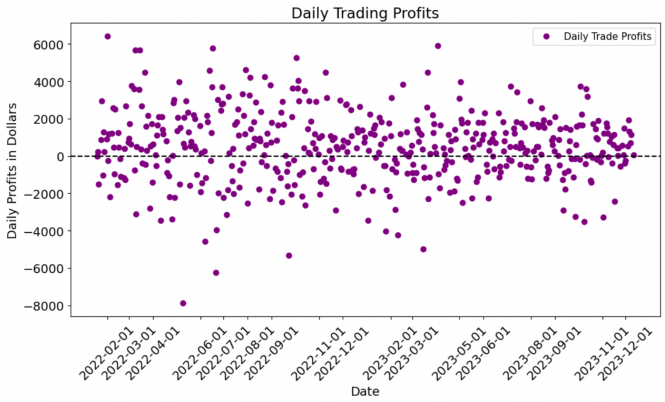


Figure 9: Profits significantly more likely to be positive, regardless of time period.

4 An Extension of our Project – The Transformer Model

While the simplified LSTM model demonstrated improved performance by concentrating solely on the percentage change in lagged closing prices, it fell short of leveraging the broader range of economic indicators, technical proxies, and volatility measures available in the dataset. To address this, we shifted our focus to the Transformer architecture, which is better suited to interpret high-dimensional datasets and modeling interdependencies between features.

Introduced by Vaswani et al. in 2017, the Transformer model was originally developed for natural language tasks like machine translation [12]. Its self-attention mechanism allows it to process entire sequences at once and identify long-range patterns, making it faster and more effective than models reliant on sequential processing. This adaptability inspired us to apply the Transformer model to our situation, interpreting relationships between the multiple economic and market factors that influence price movements.

4.1 Technical Description of Transformer Model

Due to page count restrictions, an explanation of the architecture of our Transformer model is attached as supplementary information at the end of the report. Of note, we purposely omit the decoder layer for financial prediction because it is designed for sequence generation tasks, whereas our focus is on sequence-to-value prediction, which only requires the encoder component.

4.2 Pipeline and Parameters

As a disclaimer, due to the highly complex nature of this model, we relied on model architecture from various research papers and code assistance from Chat GPT [9] [12] [13].

We collected our data using the same methods as our LSTM model. Because financial data is so noisy and complex, it is often better to employ weak learners and combine them into an ensemble [9]. To facilitate this, we organized the features into distinct clusters based on their interrelationships. Pearson and Spearman correlation matrices were computed to capture linear and monotonic relationships, while mutual information scores were calculated to identify non-linear dependencies. Hierarchical clustering was then performed using the Pearson correlation matrix with the Ward linkage method, which groups features by minimizing within-cluster variance. A predetermined number of four clusters were established, and each feature was assigned to one of these clusters based on the clustering results. We also established a target group with adjusted closing prices of XOM. Each feature group then undergoes preprocessing where log percentage changes are computed to normalize the data and capture relative movements. Next, we define a custom TimeSeriesDataset class to create sequences of input features (X) and corresponding targets (y) based on a 60 day sequence length (L) and 5 day prediction horizon.

We employed Optuna, a Bayesian optimization framework, to systematically explore and identify the optimal set of hyperparameters. The embedding dimension was set to 128 to balance representation capacity with computational efficiency. The number of attention heads was set to 8, enabling the model to capture diverse feature interactions. The number of encoder layers was chosen as 3 to provide sufficient depth for hierarchical feature extraction. The dimension of the feedforward network was set to 512 to allow the model to perform complex transformations on the aggregated attention outputs. The dropout rate came in at 10% to prevent overfitting by randomly deactivating neurons during training. For optimization, the Adam optimizer was chosen with a learning rate of 1e-4 and a weight decay of 1e-5 to ensure stable and regularized updates. A ReduceLROnPlateau scheduler was implemented to dynamically adjust the learning rate based on validation loss, reducing it by half if no improvement was observed over 2 epochs. We trained the model for 12 epochs (number of iterations through the data) using 5-fold cross-validation, which involved splitting the data into five parts, training on four parts and validating on the fifth each time. A batch size of 32 was employed, meaning the model processed 32 samples before updating its parameters.

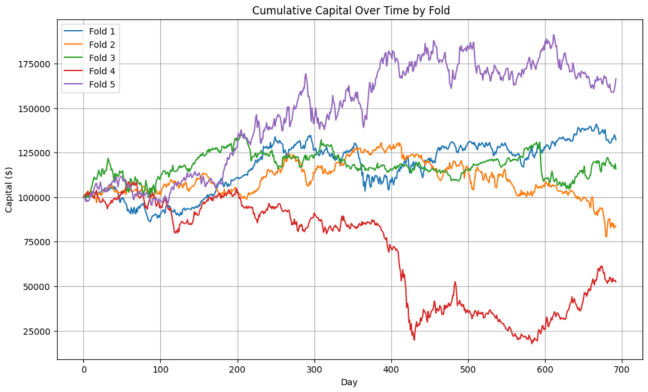
The Mean Squared Error (MSE) loss function was used to measure prediction accuracy, optimizing the model to minimize the average squared differences between predicted and actual values. Throughout training, the model parameters were regularly evaluated on the validation set, and the best-performing model based on the lowest validation loss was retained.

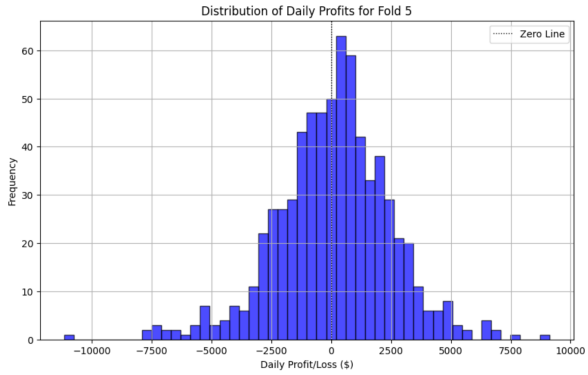
We then decided to test the profitability of the model's predictions. Starting with an initial capital of $100,000, the function iterates through each fold's test data, making long or short positions based on the forecasted returns and updating the capital accordingly. The performance is summarized by metrics such as total return and average profit and loss of each trade.

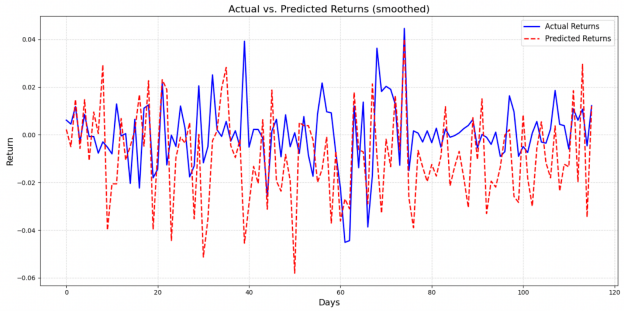
4.3 Results for Transformer

The simulation concluded with a final capital of $145,936, marking a 45.94% increase from the initial investment. Over the simulation period, 3,465 trades were executed, each yielding an average profit of $13.26, highlighting consistent performance. Despite an accuracy rate of only 48% – how often our model correctly predicted the direction of the stock price 5 days ahead based on its most recent data – it excelled by capitalizing on key opportunities where the signals aligned strongly, leading to impressive returns.

Interestingly, the Transformer model with all inputs underperformed compared to the simpler LSTM model, which relied solely on the log percent change of prices. This suggests that markets efficiently price in diverse signals, and adding complex inputs like future spot spread and crack spreads may introduce noise. For example, stock prices can reflect simultaneous influences from various market factors, such as broader economic trends or supply shocks, making it difficult for the model to isolate actionable signals for the specific trade horizon.







5 Discussion

At the beginning of this project, we built out our LSTM model based on the assumption that the more data we feed the model, the better its prediction of the stock price would be. We soon realized that there are issues with this theory. First, if the features training the model are not predictive of the stock price, then the predictions will be flawed as a result of noise. Second, the LSTM model requires that features and target variables are normalized on the same scale, which can be tricky when working with price data, input-output spreads, technicals, and economic proxies, some of which are normally nominal variables, percent changes, or are represented in terms of their z-score. With this in mind, the relatively poor performance our of normalized LSTM model makes sense. Essentially, it was having trouble determining which patterns in the data were relevant and indicative of the price of ExxonMobil. As a result, we used the same model, but now trained solely on the past price movements of the stock. The performance was substantially improved, but we still sought to understand why this was the case.

Fortunately, the more sophisticated Transformer model, being able to better model the relationship between the input features, offered us some insight into our overall findings. We saw that while the model's predictions led to profits significantly higher than the complex LSTM model, it fell short of beating the LSTM model

reliant solely on lagged price data. Overall, this suggests that the optimal way to predict the price of an equity is to rely on a technical analysis approach, and base your future price predictions on simply the past prices of the stock. Introducing feature data, that while still might be relevant for the price of the equity, can confuse the deep learning models by introducing noise that makes price patterns harder to decipher. Additionally, it could be the case that information such as spot and future crack spreads, inflation, and momentum is already factored into the price of the equity in accordance with the Efficient Markets Hypothesis.

Nevertheless, we were not afraid to take risks in this analysis, and learned that sometimes even with the most complex models, it is the case that keeping inputs and assumptions simple is the key to optimal performance. Our goal at the outset of this project was to keep things on a microscopic level and get an accurate prediction for the stock price of one commodities company. Fortunately, we were able to make significant progress towards this goal, while also understanding the factors that are most predictive of the price. Moving forward we would repeat this analysis for other equities in the oil and gas market, while also branching out to other commodities markets. From there, if we were confident in our projections, we could build an optimal portfolio of stocks to predict prices for and trade on day-by-day. If done properly, we have newfound hope that it is in fact possible to beat the market.

6 Summary

As mentioned above, trying to make excess returns on the market is incredibly difficult, even for "professional" investors. As a result, we took a deep dive into predicting the price of just one equity, ExxonMobil, by leveraging deep learning techniques such as the LSTM model and the Transformer model. Our results show that manufacturing and economic factors are potentially already included in the price of the equity, so the optimal way to predict the price is to base future predictions on the past prices of the stock themselves. Through this technique, we were able to make substantial hypothetical trading profits, making the correct trading decision on nearly two-thirds of the days in the testing period. Fortunately, the model we built is scalable and can be applied to other stocks in the same market or beyond. Future work would repeat this analysis for other equities and extensively review the results. Depending on the accuracy of these results, we could then use this framework to construct an optimal portfolio of daily stocks to be traded based on our predictions. Our hope is that by doing this, we see that through careful modeling decisions and proper testing, there are significant profits to be made in the stock market.

7 Attribution of Effort

For our last project, we once again worked really well as a team, as we approached an unforeseen amount of hurdles with these more challenging models. This time we focused a lot on iteratively overcoming disappointments we had reached, unafraid to take risks and confront these topics' difficulties. All three of us took turns working on the Colab notebooks, with Eva and Michael focused more on the LSTM model and Oakley tackling the very difficult Transformer model. Thus, we are able to report back on our progress as we tackled these aforementioned stumbles. We used Generative AI, particularly GPT o1, for help with brainstorming and coding, particularly our Transformer model, as this model was immensely difficult and above our coding capabilities/expectations. As for the report, we split up the writing comparably, focusing on writing areas that corresponded to what we had coded. We have enjoyed working together as a team, as well as the focus and determination we have felt while tackling this project!

8 Supplementary Information – Transformer Architecture

1. Input Embedding – Each token in the input sequence is converted into a dense vector via embedding layers. These vectors are high-dimensional (e.g. 768) with nonzero values distributed across all dimensions. This contrasts with the more compact representations of LSTM models

2. Position Encoding – Since Transformers do not inherently process data sequentially, positional encodings

are added to provide time-series context. Mathematically, this is achieved by adding sinusoidal functions of different frequencies to the embeddings.

3. Self-Attention – Determines how much focus each part of the input sequence should have on other parts. The model generates three vectors for each input token:

   - Queries (Q): Represent what the model is looking for.
   - Keys (K): Represent what each token contains that might match the queries.
   - Values (V): Hold the actual information to extract.

   The Queries and Keys are compared using a dot product to measure similarity, producing raw attention scores. These scores are then scaled (to stabilize training) and passed through a softmax function, which normalizes them into a probability distribution. The resulting weights are applied to the Values, producing a weighted sum that emphasizes the most relevant information for each token.

4. Multi-head Attention – Extends self-attention by running multiple attention "heads" in parallel. The input is split into smaller subspaces, and each head computes its own Queries, Keys, and Values, focusing on different parts of the sequence. This process enables the model to capture both short and long-range dependencies that a LSTM might struggle with.

5. Feedforward Neural Networks – Applies two linear transformations with a rectified linear activation function (ReLU) activation in between. The first transformation projects the input into a higher-dimensional space. The ReLU is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. This introduces non-linearity to the model by breaking the linear relationship between inputs and outputs. The second transformation maps the data back to its original dimension.

6. Residual Connections – Helps maintain smooth gradient flow during training and prevents the vanishing gradient problem. Instead of learning the full mapping, the network learns a residual between the input and output. This is achieved by adding the input of a sub-layer directly to its output.

7. Layer Normalization – Stabilizes training by normalizing the output across the features of each sub-layer. This ensures the output has zero mean and unit variance

8. The Transformer model comprises multiple identical encoder layers, each containing the multi-head self-attention and feedforward neural network components described above. Stacking these layers allows the model to build hierarchical feature representations, enhancing its capacity to understand complex patterns within the data.

References

1. Alden, Lyn. *Should Investors Try to Beat the Market?* Available at: `https://www.lynalden.com/beat-the-market`.

2. Hayes, Adam. *Commodity Market: Definition, Types, Example, and How It Works.* Investopedia.

3. *Basics of Futures Trading* — CFTC. `www.cftc.gov/LearnAndProtect/AdvisoriesAndArticles/FuturesMarketB index.htm`.

4. *Why And How to Invest in Commodities* — U.S. Bank. 21 Aug. 2020, `www.usbank.com/investing/financial-perspectives/investing-insights/why-and-how-to-invest-in-commodities.html`.

5. Siddharth. *Stock Price Prediction Using LSTM and Its Implementation.* Analytics Vidhya.

6. Thushan Ganegedara. (n.d.). *LSTM in Python for stock market prediction.* DataCamp, 10 Dec 2024. `https://www.datacamp.com/tutorial/lstm-python-stock-market`

7. Lamont, Duncan and Schroders Investment Management. *Insight. Vol. 1, 2024,* www.hartfordfunds.com/dam/en/docs/pub/whitepapers/WP597.pdf.

8. InvestorPlace. *"Exxon Mobil Stock Is an Inflation Beater."* Nasdaq, 4 Aug. 2021, www.nasdaq.com/articles/exxon-mobil-stock-is-an-inflation-beater-2021-08-04.

9. Jia, Hengjian. *Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction.* doi:10.48550/arXiv.1603.07893.

10. Vipond, Tim. *"Technical Analysis - a Beginner's Guide."* Corporate Finance Institute, 15 Dec. 2023, corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/technical-analysis

11. Vaswani et al., (2017). *Attention is All You Need. Advances in Neural Information Processing Systems.*

12. OpenAI. (2024). ChatGPT (GPT-o1) [Large language model]. Retrieved from https://chat.openai.com