Forecasting Commodity Prices

Matt Johnson
SUNet ID: mattj949
mattj949@stanford.edu

Bodhi Nguyen
SUNet ID: bodhin
bodhin@stanford.edu

Abstract

In this project, we explore forecasting commodities prices and returns using ARIMA and LSTM models. For each model, we test a simple trading strategy using historical data, and calculate both the Sharpe ratio and the directional accuracy of the forecasts over time. We start by applying ARIMA models to corn, wheat and crude oil prices. We found that crude oil prices were exceedingly unstructured, and as a result we failed to find a reasonable model. For corn and wheat, we found that MA(1) and AR(1) models resulted in the lowest AIC/BIC scores respectively. However, both models fail to predict directional price moves with an accuracy greater than 50%. To explore the usefulness of more complicated models, we forecast corn futures prices using a single feature LSTM model, a 13 feature LSTM model (with features related to exogenous price data for soybeans, oil, commodity indexes), and a 13 feature multi-task LSTM model which explicitly emphasizes the direction of returns in its loss function. Of the three LSTM models, the single feature model performs the best, however it still performs very poorly: it predicts the correct direction less than 50% of the time, and its associated trading strategy obtains an in sample Sharpe ratio less than 1. In conclusion, we find that commodity prices are exceedingly hard to forecast with few features; more informative, carefully constructed features likely are required.

Github: https://github.com/mattj949/stats207

1 Introduction

Financial asset price prediction is notoriously difficult, primarily due to an extremely low signal to noise ratio, and a lack of stationarity. There are two common schools of thought in quantitative finance on how to deal with extremely noisy data series: 1) keep the model simple, because more complicated models will just overfit to noise, and 2) go straight to the complicated/advanced models, since one needs a more powerful model to have any hope of deciphering any signal buried deep in the noise. We choose to examine both approaches, first by applying ARIMA models to corn, wheat and crude oil. Then, we delve deeper into a more complicated model, LSTM neural networks, to forecast corn prices.

2 Dataset

We chose to forecast corn, wheat, and crude oil prices. We first take the log of each price series (to suppress extreme values), and then take a first-order differencing; this yields a series of log returns (see figure 1). We note that the transformed data are still nonstationary. Particularly, there are exceedingly large price moves which exhibit clustering behavior. Higher order differencing did not improve the stationarity of the data, and as a result the log return transformation was the best we could do. We use data from 1990 through 2020 to study and evaluate different model parameterizations.

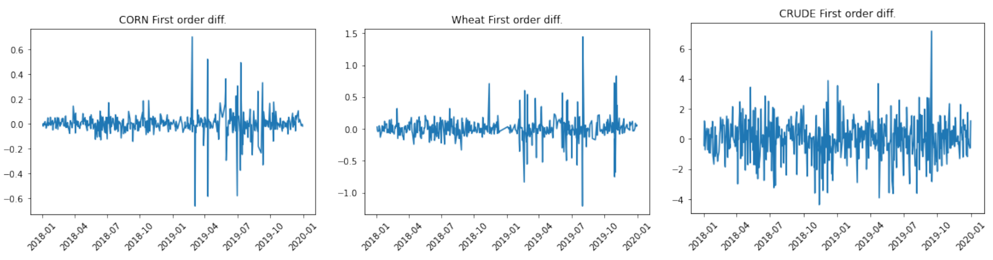


Figure 1: Data series are still "spiky" - indicative of nonstationarity, or heteroskedasticity. Plot from 2018-2020.

For the LSTM model, our partition of data between training, validation, and test sets is as follows: all data up to 2018 is used as training, 2018-2019 is used as validation, and 2019-2020 is used as test data. Sharpe ratios and directional accuracy statistics are calculated on this 2019-2020 period for both the LSTM models and for the ARIMA models to ensure consistency. For the 13-feature LSTM models, in addition to the log return price series we also include a set of twelve other exogenous variables as input. These features include data on other commodity indexes, regional corn sub-prices and volumes, oil prices and volumes, and soybean prices and volumes. A full list of these exogenous variables is available in the appendix.

All data was acquired through Global Financial Data, provided through Stanford's Graduate School of Business.

3 Methods

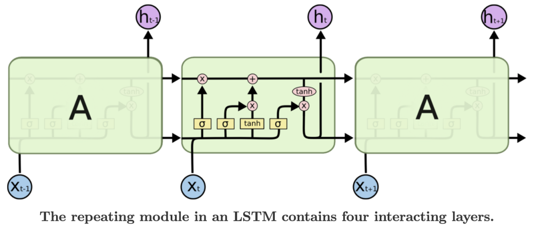
3.1 ARIMA

Our baseline model is the AutoRegressive Integrated Moving Average (ARIMA) model. As mentioned above, we performed a first order differencing on the log price data, hence the ARIMA model as opposed to ARMA. The biggest problem we ran into with ARIMA models was in choosing the parameters: What is the AR order? What is the MA order? Since our data series are so noisy, deciphering the best parameterization is very difficult, even when using tools such as Q-Q plots, Ljung-Box statistics, and plots of the residuals. The Box-Jenkins method for applying ARIMA models informs us that we need residuals which are stationary and uncorrelated. As we show in section 4, generating uncorrelated residuals is not difficult, however generating truly non-stationary residuals is quite difficult (Q-Q plots diverge from normality at the tails). Financial data is quite non-stationary and non-Gaussian, and this makes fitting parametric, stationary models such as ARIMA very difficult. Determining appropriate model parameters via a qualitative analysis of the diagnostic plots is too uninformative/inconclusive to be useful. Through numerous iterations of ARIMA parameterizations on all 3 commodities, no combination of parameters ever stands out as being superior, or even "good". After running in circles, we decided to follow a similar procedure to Modal(2014) [4] by prioritizing models with low AIC/BIC. By doing so, we implicitly are choosing models that balance explanatory power (log likelihood) with the number of parameters, however here lies an issue.

We are using the `statsmodels` package in python to perform our ARIMA forecasting. For parameter selection, the package uses Maximum Likelihood Estimation, and this became problematic in our testing. Inherently, MLE requires some method by which one can quantify "likelihood" (i.e. probability); this is done by assuming that the model residuals are Gaussian. However, as described above and as is shown in the results section below, the residuals of our models are by no means Gaussian. Hence, using the density function of a normal distribution to determine likelihood is nonsensical. A solution to this might be to use the empirical probability distribution as formulated in [2], however our modelling software does not include such an implementation. We proceeded to use MLE to fit our model parameters, with the caveat that while normality holds most of the time, it can blow up spectacularly in periods of financial stress.

3.2 LSTM - Neural Network

A LSTM neural network was chosen as our neural network forecasting model, predominantly because this model is advantageous in managing sequences. In the context of our problem, the input sequence is a set of daily log returns. The output/target is the next day's daily return. Long Short Term Memory networks improve upon traditional RNNs by learning long term dependencies. They do this by maintaining a cell state, where information is added or removed by three different types of "gates": a forget gate, input gate, and output gate. These gates control the flow of information, and determine how much information is carried over from the beginning of the sequence to the end. This is particularly useful for long sequences or time series when points at the beginning of the sequence are still important. A diagram of this structure is shown below in figure 11.



h_{t-1}
h_t
h_{t+1}
A
σ
tanh
A
X_{t-1}
X_t
X_{t+1}
The repeating module in an LSTM contains four interacting layers.

Figure 2: Structure of an LSTM. source: [5]

This is ideal for our problem because contextual information from previous time steps may be very useful for return prediction in the future, and we want to make sure we don't lose that information. For example, say a forward-looking crop health report came out 30 days ago. This information may still be relevant for the next day's return, so we want our neural network to encode information about it, although in terms of time it is very far away. Our LSTM is written in Pytorch, and the initial experiment uses the following architecture. Our network consists of one Pytorch LSTM layer, and a linear layer to case the output of the LSTM to a single prediction label. Our hidden layer in the LSTM only has 2 units. The lookback period is 30 trading days. The standard Adam optimizer was used with a learning rate of 0.001. The loss function was chosen as average mean squared error.

We build three separate LSTM models, focusing on just corn futures prices (CBOT Corn Futures Prices). The first model uses the just a single feature, the log return of corn prices. The second model uses corn prices and 12 other features related to commodity prices (soybean prices, oil prices, commodity indexes, various regional corn sub-prices).

The third model builds upon the second model by turning the LSTM's loss function into a multi-task problem. We train the LSTM by explicitly penalizing directional mismatches between the predicted return and actual return; this is done by defining our loss function to be the mean squared error of predictions, plus the binary cross entropy loss of the sign of the prediction (ie the direction of the return). Ideally, this will help our strategy, as having the correct direction in our prediction can be just as important as how far off in magnitude it is. Additionally, all features are Z-scored by the historical data preceding it, to avoid look-ahead bias and to avoid exploding or vanishing gradients, which helps ensure more efficient convergence.

3.3 Evaluation and Backtesting

As we mentioned before, model selection and evaluation is particularly difficult in this project as we are generally comparing "bad" models to "worse" models. Further, traditional loss metrics like mean-squared-error don't capture the nonsymmetric impact of errors in finance; for instance, in a problem where we buy a stock who's price we expect to go up, upside error is different than downside error. If we predicted too high of a price, then we might lose money. If we predicted too low of a price, then we might make more money than we anticipated; hence a downside error is not as bad as an upside error.

We decided to evaluate the usefulness of our models by performing a backtest. A backtest is performed by running a trading strategy on historical data as if we had traded using the strategy in the past. Then, we can evaluate how well the trading strategy "would have performed". For a given model, our strategy is simple: buy if the forecast says that the price will go up, and sell short otherwise. We pitted our ARIMA forecasts against the financial industry's standard forecast for this type of strategy: the mean price over the last two months, or 40 trading days. This is akin to an AR(40) model, where each AR coefficient is $\frac{1}{40}$. We refer to this industry standard strategy as the "reference" strategy.

We assess each backtest both visually (by looking at the portfolio value over time) and by by computing a common financial statistic known as the Sharpe ratio. Higher Sharpe ratios are favorable, and this metric is calculated as follows: annual Sharpe ratio = square root of (252) * (mean daily return / standard deviation of daily returns), where 252 is derived from there being 252 trading days in a year. We also compute each model's directional accuracy: the percentage of days the model predicted the correct direction of price movement.

4 Results

4.1 ARIMA

The models and methods mentioned below were applied to the first order differenced log price series of the commodity in focus.

4.1.1 Corn

We start off by illustrating our decision making process in detail for Corn. We start off by observing the ACF and PACF:

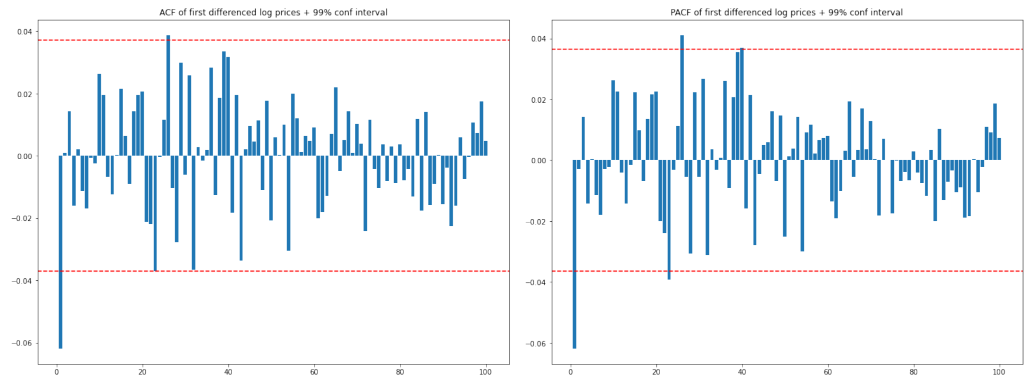


Figure 3: Minuscule coefficients, even if statistically significant. Run on data from 01/01/2000 - 01/01/2020

Unfortunately, the ACF and PACF plots are not very informative. It is not very promising to observe that all coefficients are minuscule in magnitude. One interpretation of figure 3 could be that both ACF and PACF coefficients cut off after lag 1, yet the magnitude of the lag-1 coefficients is still only around 0.06. Another interpretation/observation could be that both tail-off as the lag increases, yet this behavior is still slight. Overall, the ACF and PACF seem to be consistent with non-stationary, random time series. Nevertheless we tested three models:

- ACF cuts off after lag 1. Try MA(1)
- PACF cuts off after lag 1. Try AR(1)
- Both series appear to tail off over time. Try ARIMA(p,1,q), but we cannot immediately know p or q.

We chose a number of ARIMA parameterizations up to order 10 for both p and q; we quickly learned that lower p and q parameterizations receive lower AIC and BIC. In fact, an ARIMA(0,1,1) model, a.k.a. an MA(1) model, on the first-order-differenced log price series scored the lowest on AIC and BIC, beating the AR(1) model as well as all higher-order combinations of parameters for the ARIMA(p,1,q) model. The Q-Q plot and Ljung-Box p-stat show that the residuals for the MA(1) model are somewhat normal (albeit with fat tails), and that the residuals have no statistically significant autocorrelation. See figure 4. However, we already knew that the data series had little autocorrelation; further, the Q-Q plot of the model residuals looks almost identical to the Q-Q plot of the transformed corn price series. See figure 5

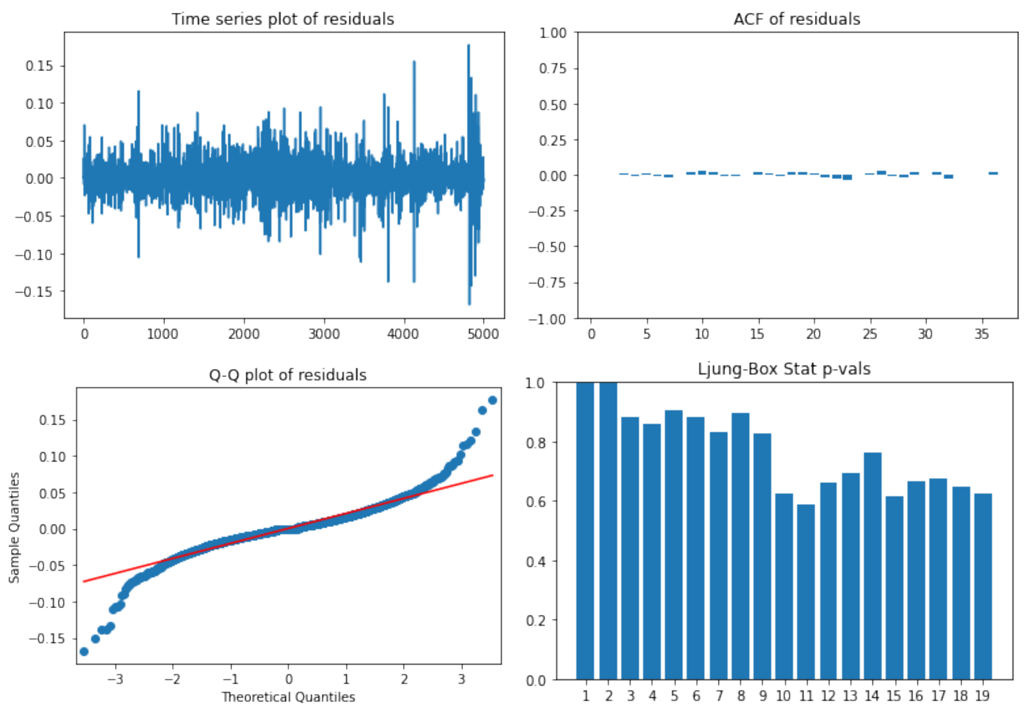


Figure 4: MA(1) Model diagnostics for Corn

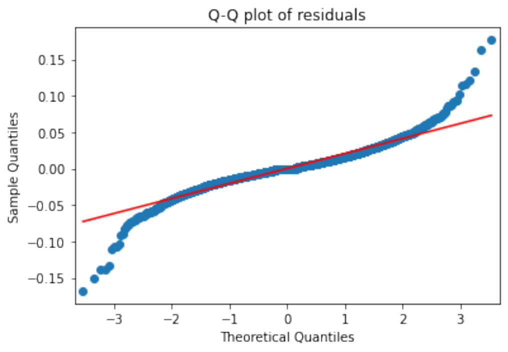


Figure 5: Q-Q Plot of transformed corn price series

What we wish to underscore with the provided model diagnostics is how deceiving they can be in isolation. Applying an MA(1) model to corn might appear reasonable if one just observed the model diagnostics: the residuals appear somewhat stationary, the residual autocorrelation is almost zero and insignificant, and the Q-Q plot is somewhat acceptable except for the tails. Nevertheless, we know that the model fits exceedingly poorly. One way to illustrate this is to note that the model only predicts the correct direction of price movement around 46% of the time. In other words, it's akin to a random coin flip. Another way to visualize model performance is by observing the performance of the model in a backtest. See figure 6, where we observe that the MA(1)-model-based trading strategy underperforms the reference strategy.

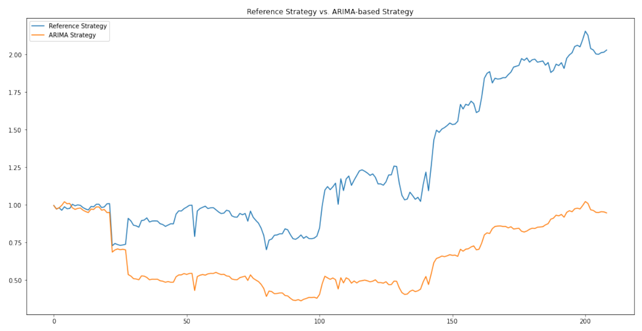


Figure 6: Trading strategy backtest using MA(1) model on corn

4.1.2 Wheat

We now perform a similar analysis on Wheat. We again provide the ACF and PACF of the first order differenced log price series:

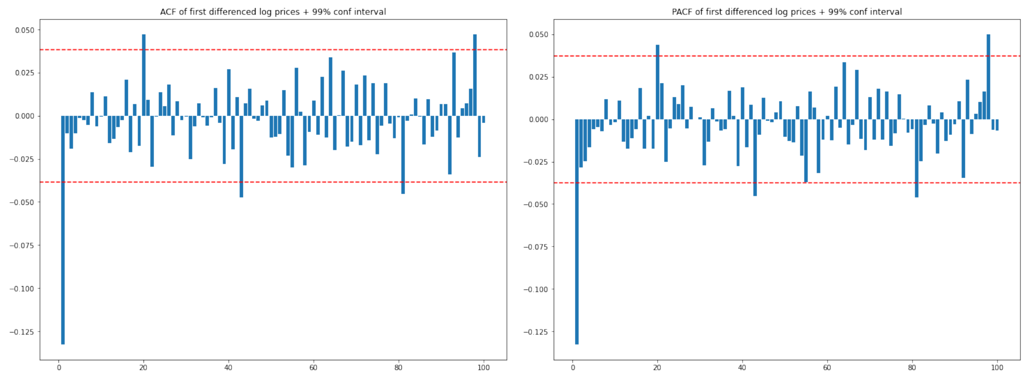


Figure 7: Wheat ACF and PACF Run on data from 01/01/2000 - 01/01/2020

As is the case for Corn, there are few statistically significant autocorrelation coefficients, but the most notable are those at lag 1. We fail to observe any tailing off behavior in the ACF or PACF, however slight. As a result, we forgo testing an ARIMA model and focus just on an AR(1) and MA(1) models as both the ACF and PACF observe noticeable cut-offs after lag 1. We found that an AR(1) model has the lowest AIC and BIC, and the model diagnostics look very similar to those shown above for Corn. As was the case before, the model superficially appears to fit the data, and we provide further performance evaluation via a backtest:

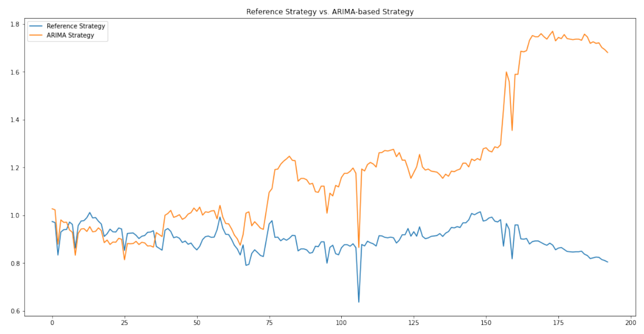


Figure 8: Trading strategy backtest using AR(1) model on wheat

Surprisingly, the AR(1) based trading strategy performs better than the reference strategy. This is despite the fact that the directional accuracy of the AR(1) model was only 44%, while the directional accuracy for the reference strategy was 42%. While we are tempted to credit the AR(1) model for it's superior forecasting ability, it is just as likely that this is a spurious byproduct of testing on a small sample.

4.1.3 Crude Oil

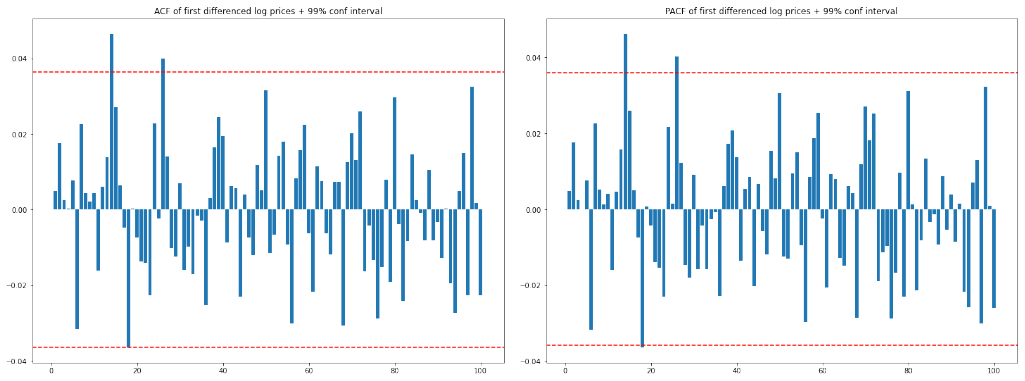


Figure 9: Crude ACF and PACF Run on data from 01/01/2000 - 01/01/2020

In contrast to Corn and Wheat, the lag 1 correlation coefficients in the ACF and PACF were not statistically significant. Further, while the lag-1 coefficients for Corn and Wheat generously exceeded the threshold for significance at the 99% level, the few significant lags in the ACF and PACF just barely meet the threshold for significance. We observe two things: all coefficients are small in magnitude, which indicates that the data are essentially random. Further, neither series decays to zero, which is indicative of nonstationarity (although since the coefficients are already so close to zero, this conclusion might be dubious). We tried increasing orders of differencing in attempts to generate a stationary price series, but to no avail. As a result, we did not apply an ARIMA model to forecast crude oil prices.

4.2 LSTM

The backtest statistics from our three LSTM models are shown below. Again, we show annualized Sharpe ratios, the percentage of time the direction of returns was predicted correctly, and a z-score from a regression of the prediction on actual returns (with Newey-West standard errors). The loss of the first two models (LSTM and LSTM with exogenous features) were arbitrarily low and about the same. The third model, the multi-task model predicting both magnitude and direction, learned from the regression loss quickly, but unsurprisingly failed to learn the direction of the return well.

Model Name	Split	Sharpe	% Direction Correct	% Time Direction + (-)	Z-stat
LSTM (1 Feature)	Train	**0.23**	**46.6%**	45.8% (54.2%)	1.23
LSTM (13 Features)	Train	0.08	45.0%	45.8% (54.2%)	0.49
LSTM (13 Feat + Dir.)	Train	0.12	45.0%	45.8% (54.2%)	0.63
LSTM (1 Feature)	Test	1.2	42.9%	41.1% (58.9%)	**1.32**
LSTM (13 Features	Test	0.107	41.4%	41.1% (58.9%)	0.09
LSTM (13 Feat + Dir.)	Test	0.22	41.1%	41.1% (58.9%)	0.28

We see that the single-feature LSTM model performed the best. However, this is a low bar, as it's Sharpe ratio is abysmal. Additionally, we see that the model fails to learn the directionality of returns. The model could have achieved greater than 50 percent accuracy on direction by predicting a negative log return every day! This is an indication that the LSTM model is not a good model for returns. Additionally, adding features seemed to hurt the model significantly: possibly by adding noise but not a lot of signal. Below, we plot a sample portfolio for the single feature LSTM, and the predicted log returns vs. the actual log returns.

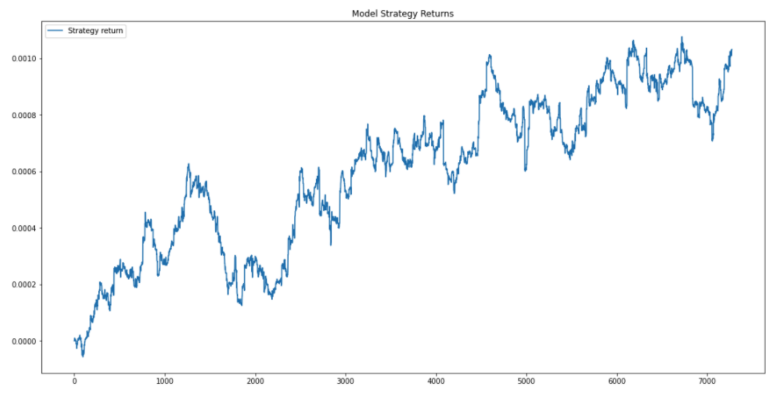


Figure 10: Sample training portfolio returns, single feature LSTM, unscaled

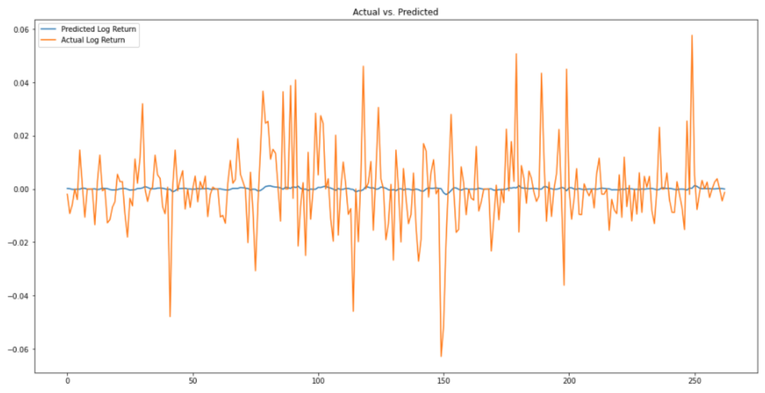


Figure 11: Predicted vs. Actual Log Returns, Testing

Our backtest seems to perform well, however the effect is quite weak, and hence the forecasts are unlikely to be useful in practice. If we look closer at the actual vs. predicted values, we see that the model simply learns a dampened version of the most recent log returns. Similar graphs for the other two models show very poor backtest performance and their predicted returns are close to random. Altogether, while neural networks succeed in other domains, the simple LSTM models presented here are not expressive enough to filter out the noise in our time series. In particular, the exogenous variables included only added to the noise; perhaps other choices of variables, outside of price data, may have added more predictive value to the model.

5 Future Work

Altogether, our models performed poorly at predicting commodities prices. The ARIMA-based models did not fit corn, wheat or especially crude oil very well, despite what the backtests might indicate (such as outperformance of the AR(1) backtest for wheat). Financial time series prices are non-stationary and non-gaussian, and the requirement of stationarity for ARIMA models is not met. Moving into the realm of more complicated models, we originally chose LSTM neural networks in an effort to get the most expressive model possible. However, given the noise and nonlinearities in financial data, the LSTM model failed to fit the data well. We believe that future work could include using other exogenous variables, or different neural network architectures. Otherwise, an accurate statement would be that financial time series data is extremely noisy, and therefore extremely difficult to forecast.

References

[1] Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. *Stock Price Prediction Using the ARIMA Model*. 2014. DOI: 10.1109/UKSim.2014.67.

[2] C. G. Constable. *Parameter estimation in non-Gaussian noise*. 1988.

[3] Hiransha M et al. *NSE Stock Market Prediction Using Deep-Learning Models*. International Conference on Computational Intelligence and Data Science. 2018. DOI: https://doi.org/10.1016/j.procs.2018.05.050.

[4] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. "Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices". In: *International Journal of Computer Science, Engineering and Applications* 4.2 (2014), p. 13.

[5] Christopher Olah. *Understanding LSTM Networks – colah's blog*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 11/29/2021).

[6] Jiayu Qiu, Bin Wang, and Changjun Zhou. *Forecasting stock prices with long-short term memory neural network based on attention mechanism*. 2020. DOI: https://doi.org/10.1371/journal.pone.0227222.

6 Appendix

Exogenous variable names:

Additional Variable Categories: Bloomberg Commodity Index, Economist Food Commodity Dollar Index, CBOT Corn Futures Volumes (USD per Bushel), CBOT Soybean Futures Prices (USD per Bushel), Chicago Yellow Corn No. 2 Spot Price (US$/Bushel), West Texas Intermediate Oil Price (US$/Barrel), Soybean Oil Cash Price (Cents/Pound), Soybeans Cash Price (US Dollars/Bushel), Brent Crude Oil (USD per Barrel), Corn, Average Price to Farmers (USD/Bushel), Sweet Corn, Price to Farmers (Cents/CWT)