Machine Learning Engineer Nanodegree

Capstone Project

Zhenning Pei
July 26th, 2021

I. Definition

Project Overview

It has long been established in the world of capital markets that the value of the VIX index, which roughly tracks the market's perception of current stock market volatility, is an integral input to pricing models of various financial instruments. In the past, most models of the VIX index have been based on the heterogenous autoregressive (HAR) process, which captures linear relationships between different lags of log-transformed VIX values by fitting a linear regression model on the daily, weekly and monthly averages of past VIX values. However, the HAR model requires a log transformation on the raw VIX values before training a linear regression model on top of the transformed data, and the objective of this project is to determine whether more sophisticated models such as the LSTM neural networks would be able to pick up on more nuanced non-linear relationships in the data without prior log transformations.

The only required input to this project is the daily prices of the VIX index between 2005 and 2019. Although VIX data can be back-traced all the way to 1990, we only start using 2005 data since there was a change in the method of calculation of the VIX index in 2003. However, we will use different preprocessing for the benchmark model (HAR) and the proposed solution (LSTM) as the two models makes different assumptions about their inputs.

Problem Statement

This study aims to determine whether LSTM neural networks could be used to improve the accuracy of forecasting the VIX index over the traditional HAR model. This is a regression type problem, and we would compare the results of two approaches to determine whether the more sophisticated method offers better performance.

First, we analyze the autoregressive properties of the log-transformed VIX prices to establish the amount of lag we need to train our models by examining the specific days of lags that have statistically significant impact on the daily closing VIX prices.

Then, we perform the benchmark analysis by using the HAR model, which is a linear regression model on the VIX prices at 3 statistically significant lags identified by previous analysis.

Finally, we train an LSTM model using all historical VIX prices up until the longest lag as used by the benchmark model; this historical period is chosen so that the benchmark model and the LSTM model uses the same amount of information to generate their respective features. We then compare the chosen evaluation metrics (root mean squared error) of both models and decide if LSTM offers adequate performance improvement to justify its additional model complexity and cost of implementation.

Metrics

The performance of both the benchmark HAR model and the proposed LSTM model would be evaluated by the root mean squared error (RMSE). While we originally also wanted to explore the Akaike Information Criterion of both models, which offers additional insight into the model performance with consideration of model complexity, we found that the AIC calculation for neural networks are not well established, so we will be basing our judgement of the two models on RMSE only.

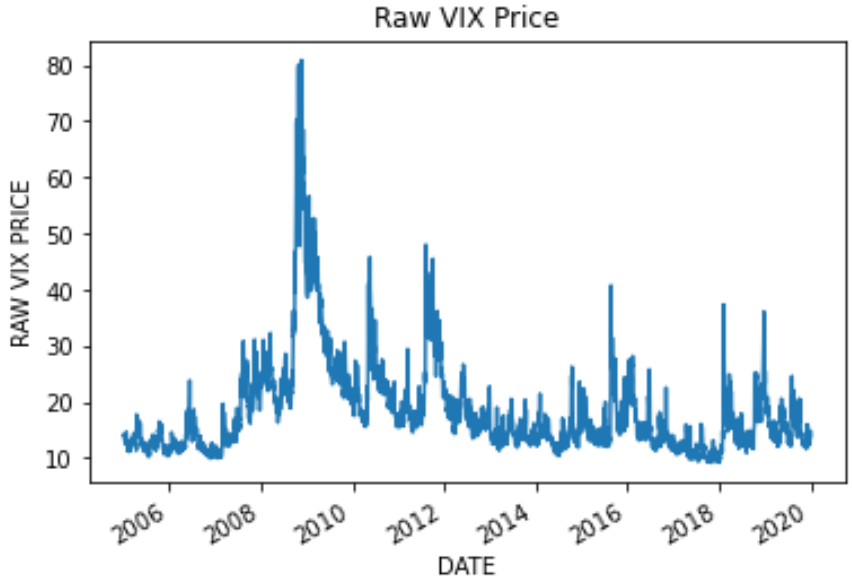
We primarily chose RMSE as a raw measure of the regression performance. The RMSE is the square root of the mean of the squared errors in the test set, so the lower this metric, the closer the model predictions are to the actual values, which is particularly suitable to our regression problem. However, the drawback of this metric is that it does not take into account model complexity, and it is up to the user to determine whether a small decrease in the RMSE is worth a dramatic increase in model complexity.

II. Analysis

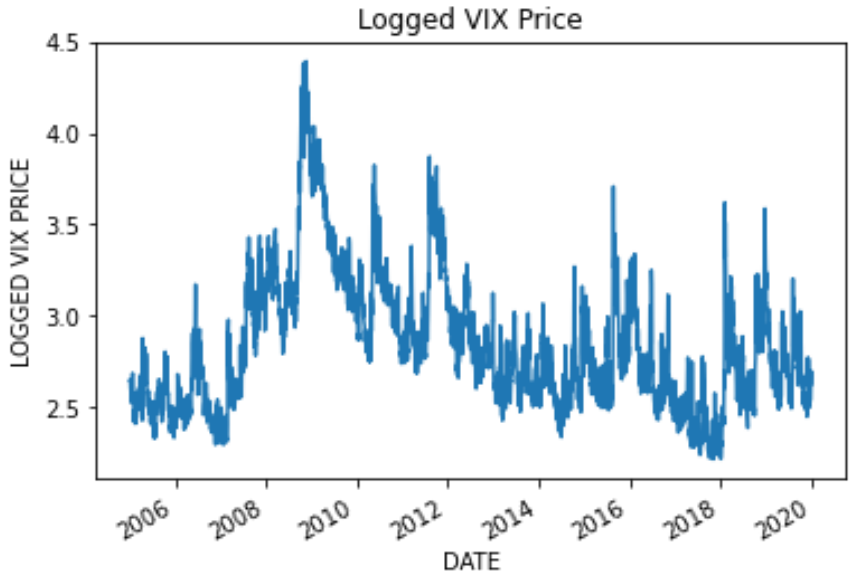
Data Exploration

The only input for this project is the VIX prices from 2005 to 2019. Since we are looking for autoregressive relationships within this dataset, we generate necessary features directly from the daily closing VIX prices within this selected period. We have chosen to use the last 2 years (2018 and 2019) as the test set and the remaining years as training set (2005-2017). That gives us 3,256 data points to train on and 503 data points to test.

In the plot of the raw data below, we see a fair amount of variability, which is typical of time series data. However, since the section containing sudden spikes in the data is

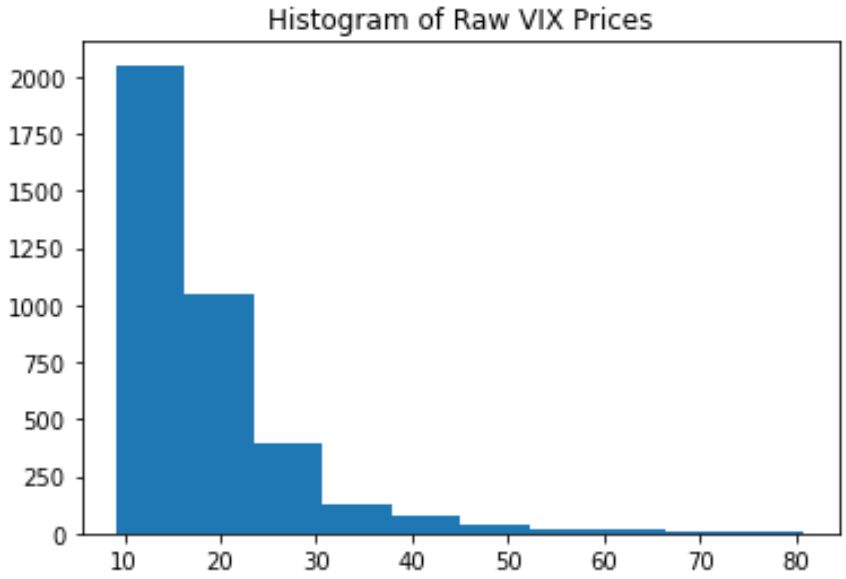


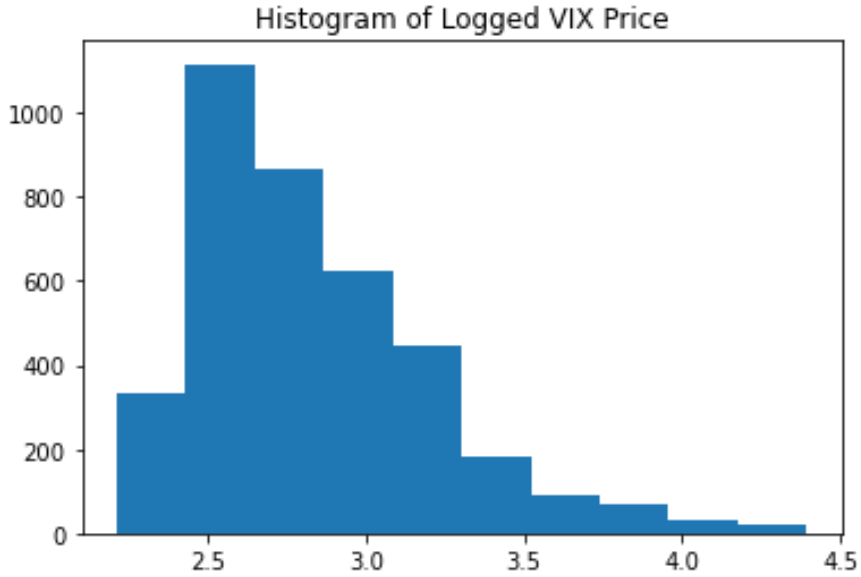
large, so such data spikes are not treated as outliers, we also believe that it is important for the final model to adapt quickly to such drastic changes, so we have decided to not provide further processing to this dataset when training the LSTM model.
In the log-transformed data required for the benchmark model, we see somewhat less variability than the raw data as a result of the scaling down of the independent variable, but the general trend of the data is preserved.



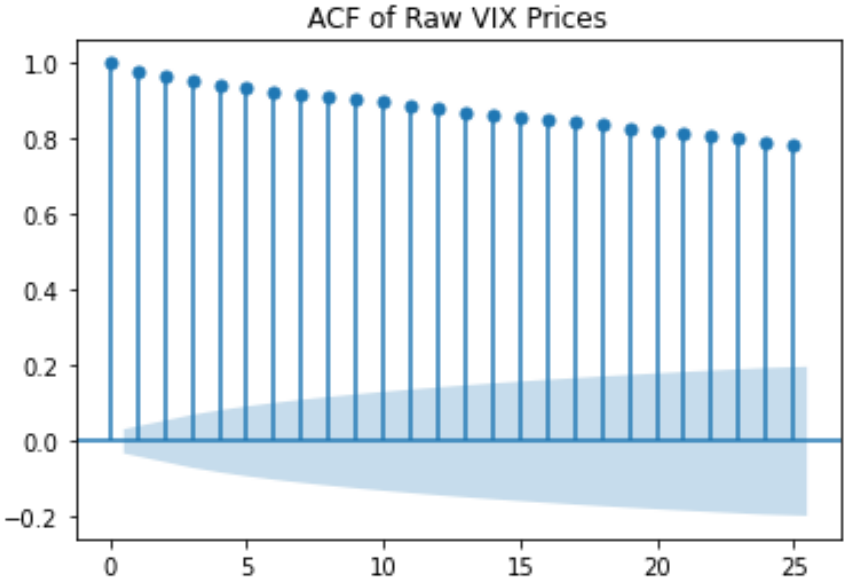
Since the data is a time series, we examine the three most prominent features for a time series for both the raw data and the log-transformed data: its distribution, its autocorrelation function (ACF), and its partial autocorrelation function (PACF).

Below are the histograms of both the raw and log-transformed datasets. We can see that both datasets are heavily skewed towards the right, with the log-transformed dataset skewed slightly less and slightly more resembling a normal distribution. This tells us that our data is clearly not centered around the mean, so when we choose our activation function for the LSTM approach, we needed to avoid activation functions that output centered outputs, such as the sigmoid and the tanh functions.

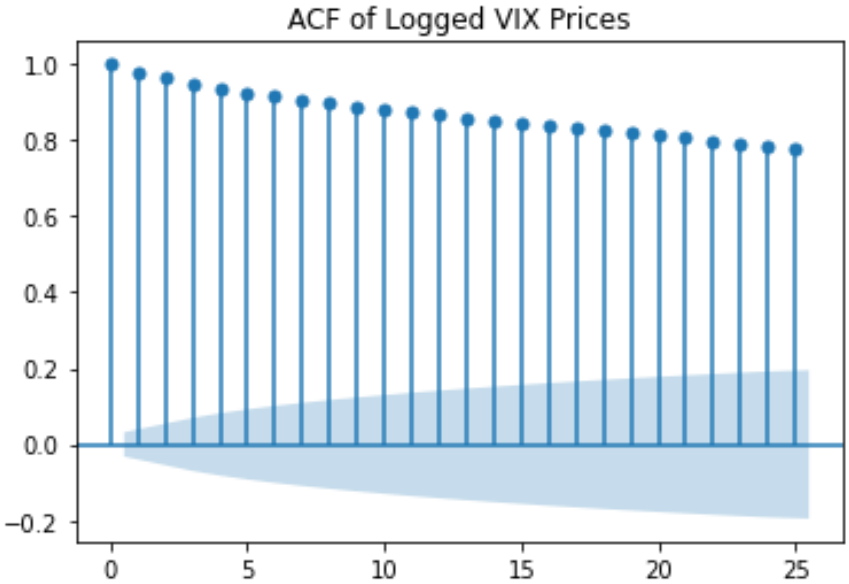




Next, we examine the ACF and the PACF of both datasets. We find that both data sets exhibit statistically significant autocorrelation for long periods of time (> 25 days in both cases), which suggests strong autoregressive properties in both datasets, thus providing theoretical basis for the HAR model, which is in essence an autoregressive model. We also plotted the PACFs of both datasets, which shows the autocorrelation of the datasets after adjusting for the propagated effect from previous days that have been already accounted for. From the PACF plots, we see that after adjusting for the effect of previous days, lags of day 1-5 and day 16 exhibits significant correlation to the current VIX price

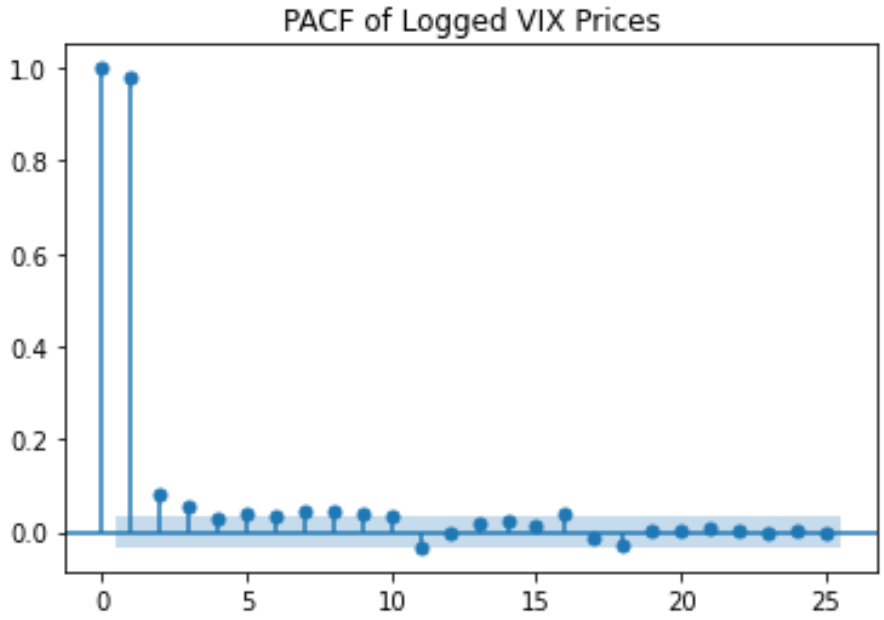


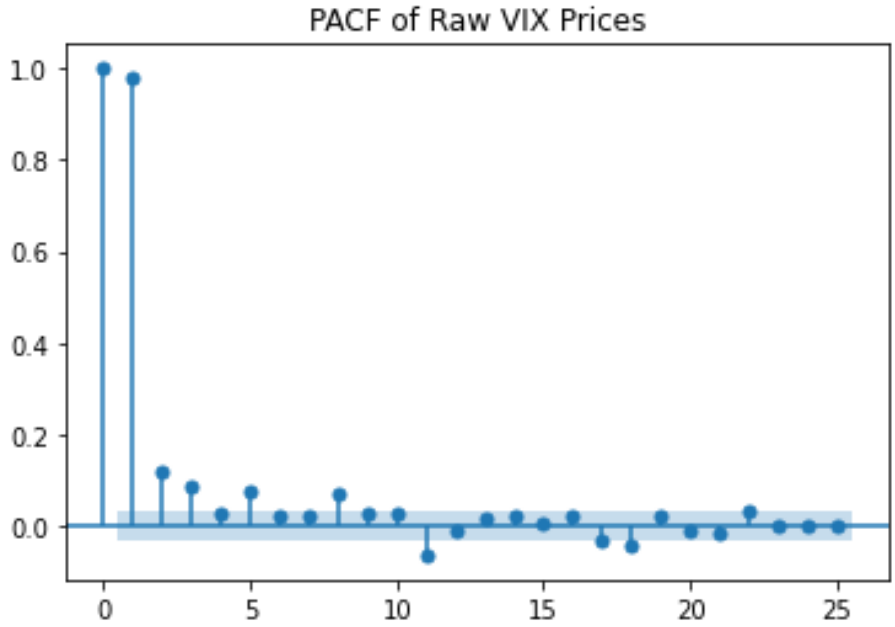
on both plots, which is critical for choosing the parameters for the HAR benchmark



model. We also observe that the shape of the ACF and PACF of both the raw and log-

transformed datasets are extremely similar, which suggests that the autoregressive properties of the raw dataset have not been compromised by the log transformation.





Algorithms and Techniques

We have chosen to use the LSTM neural network for completing this project. LSTM is the ideal candidate to learn sequential data. After the LSTM layer, we have chosen two linear layers to reduce the dimension of the LSTM output.

For the LSTM layer, we have the following hyperparameters: timestep length (sequence-size), dimension of the hidden state (hidden-dim), learning rate (learning-rate), number of samples in a batch (batch-size), and the number of epochs (epochs).

To choose timestep sequence size, we decided to use the number of days of historical VIX price we expect to have a significant effect the current VIX price, which is chosen to be 16 and is also consistent with the number of days used to predict current VIX price in the benchmark HAR model.

Usually, the dimension of the hidden state is larger than the number of input features for LSTM to extract more information from the given inputs, so we decided to find a good hidden state dimension by trial and error, with the minimum hidden state dimension being 16.

For learning rate, batch size and number of epochs, we have decided to use the trial-and-error method with a few industry standards and adjust based on our experimental results.

Benchmark

Our benchmark model is a model based on the HAR process as laid out in the paper by Corsi (2009), which is still the most widely accepted way to model and forecast volatilities due to its high performance and ease of interpretation. In Corsi's paper, we generate a vector of 1, 7 and 22 day averages (corresponding to past daily, weekly and monthly VIX levels) of past log-transformed VIX prices, which is then fed into a linear regression model to forecasting future logged VIX values.

Through the PACF of the log-transformed VIX prices in our data exploratory analysis, we see that days 1, 5, and 16 seem to have more impact on current VIX prices than days 1, 7, and 22 as proposed on the paper, so we have built the same linear regression model based on the averages of the log-transformed VIX prices with lags of days 1, 5 and 16. It turns out this simple linear regression model offers surprisingly high performance.

III. Methodology

Data Preprocessing

The data preprocessing for the benchmark model and the proposed LSTM model are different due to different model assumptions, but we kept the period of data used in

models the same for comparability. Both models are trained using data from 2005 to 2017, and tested with data from 2018 and 2019.

For the benchmark HAR model, data is assumed to have close to normal distribution, and the original 2009 paper preprocessed their data with a log transformation. We followed suit and performed a log transformation on the raw VIX prices, and from the exploratory data analysis, we see that the log transformation indeed moved the distribution of the data closer to normal. Then, we created the 3 features used in the model by taking averages of past log-transformed VIX prices with lags of 1-day, 5-day and 16-day.

For the LSTM model, we are looking to avoid the log transformation done to the HAR model as we wanted to capture all possible nonlinear relationships within the data, so we are keeping preprocessing to the minimum. In the benchmark model, we aim to predict the current VIX levels with 16 days of past data, so we are using 16 days of raw VIX values as features. We also scaled the inputs to be within the range of 0 and 1 by dividing the inputs by 100 since LSTMs tend to perform better with small-valued inputs. We chose the simplest way to scale down the inputs also as an attempt to reduce the loss of nonlinear relationships in the inputs by more complicates, nonlinear methods of scaling.

Implementation

In order to implement the project, we need to create our own custom data loading functionalities suited for LSTM models, which takes in 3 dimensional inputs, determine a suitable activation function, and convert the predicted values back to the original untransformed VIX prices before evaluation. This is quite a complicated process, and various mistakes were made in the process which resulted in the model predicting constants for a long time before it was eventually fixed.

First, we needed to define a custom Dataset class for pytorch to create batches of sequential windows for LSTM input. Given that the chosen time step is 16, every time the data loader selects one sample, we needed to return input features for 16 consecutive days, which required a custom Dataset class that returns a window of elements instead of a single row of the sample upon calling its __getitem__ method. The biggest mistake during this process was when the data loader run out of enough samples to create a timestep of length 16, we padded the features with 0s but mistakenly propagated the last available VIX price as the value for the independent variable, causing the model to produce a constant output since even all zeroes were matched to a non-zero output. There were many attempts to solve this problem, and eventually we found that the easiest way to fix it was to interpret the __len__ function to

the dataset class to refer to the maximum number of elements that could be extracted with the __getitem__ method, so we restricted the __len__ of the class to be the number of possible starting points for a window containing a full timestep (number of all samples – timestep length + 1). Thus, we eliminated the need for any padding, and thus overfitting, of any section of the training data.

The second big hurdle was had was to construct the LSTM model. Our model consisted of 1 LSTM layer and 2 linear layers to reduce the dimension of the LSTM output. When we first constructed the LSTM model in pytorch, we had problems with the outputs being constant for different inputs after every single layer. We rigorously examined and fixed a few bugs in the data loader which caused it to provide unbalanced, but the problem remained. Eventually, we found out that both the linear layers and the LSTM layer were initialized with 0 weights, and once we modified the initial condition to be randomly generated instead of all 0, we started seeing the model producing non-constant outputs. Eventually, we decided to use the kaiming_normal_ method in pytorch to initialize our layers, since it seems to be the best candidate for our ReLU activation function.

After the model was fixed, the last step was to transform the model outputs into actual VIX prices. For the log transformation done for the HAR model, we exponentiate the model outputs, and for the LSTM model, we simple scale the predictions back by multiplying by 100. Then, we computed the RMSEs for both models by comparing the predictions with the actual VIX values during the testing period.

Refinement

The refinement of the LSTM model took almost as much time as the implementation. We determined several hyperparameters by trial and error, the most important of which are the learning rate and the optimal number of epochs.

The hyperparameter that had the greatest effect on the model performance was the learning rate. After we fixed the problem with the model only predicting constants, we found that although there was starting to have variability in the model output, the variation was too small that the output still resembled constant. However, this problem went away as soon as we set the learning rate from 0.1, which was extremely high, to 0.01, which gave us much better results. Eventually we settled on a learning rate of 0.001 as it seems to still allow the model to train accurately while not drastically increasing the model training time.

We determined the optimal number of epochs to train by examining the convergence of the training error. We started with 30 epochs, and we gradually increased this value

since we saw that the training error was still on a decreasing trend when we stopped training. We chose the number of epochs to be 200 since we saw that the training error became steady around 200 epochs.

IV. Results

Model Evaluation and Validation

Given that our testing period contains two full years, we are confident that the RMSE reflects the true predictive power of the two specific models used to generate the predicted outputs. However, a big caveat is that the hyperparameters of the neural network were selected by trial and error (albeit careful as we could be), and this study could not conclude whether a better-tuned neural network could perform better, especially when the difference in the RMSE is low as we will see below.

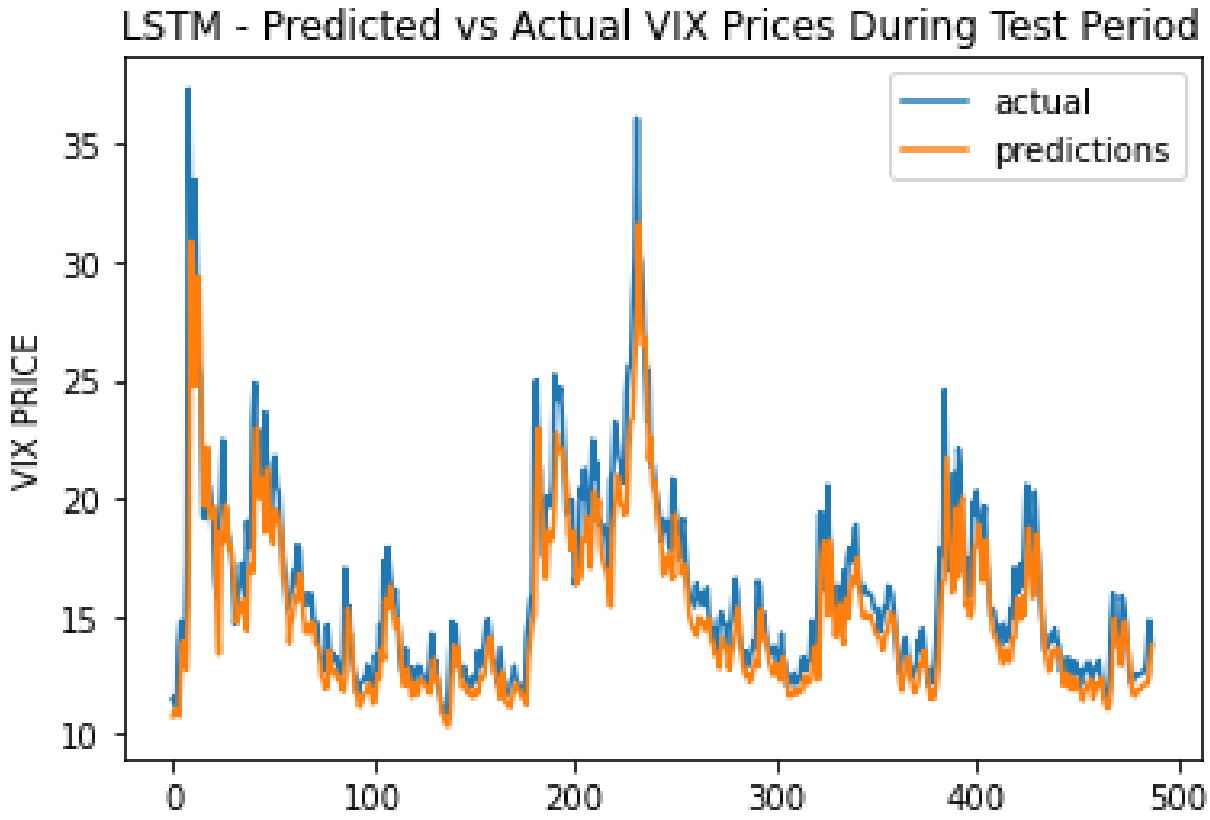
Justification

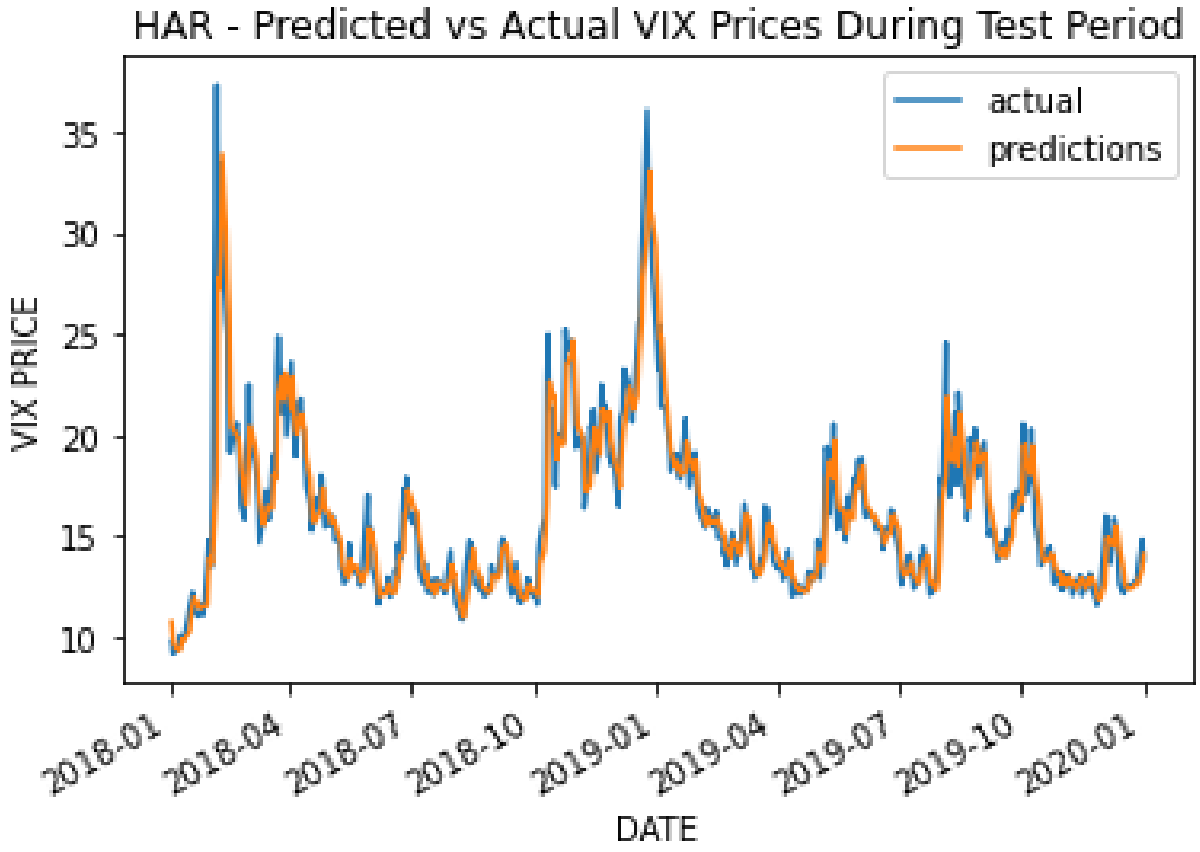
The metric we decided to use for the project was the root mean squared error (RMSE), which was calculated using both actual raw VIX prices during the testing period and the model predictions that have been transformed back to the scale of raw VIX prices.

The RMSE for the benchmark HAR model is 1.6468, while the RMSE for the proposed LSTM model is 2.0859. While both RMSEs are relatively low, an observation also backed by the plot of both models' predictions against actual values during the test period as shown below, we see that from the perspective of RMSE, the simple benchmark HAR model with only three features seems to outperform the 3 layer LSTM neural network!

On another note, it is interesting to note the similarities of the outputs for the LSTM and the HAR models as exhibited in the plots below. First of all, prediction from both models seem to follow the shape of the actual VIX values very closely. We also see that when there's a sudden spike in VIX price, both models underestimates the magnitude of the spike, which is not surprising since the majority of the data points are much smaller in magnitude. It is also quite clear that the plot of outputs predicted by HAR more closely overlap the actual VIX prices, confirming the results of the RMSE analysis above.

However, it is important to remember that unfortunately, due to the limitation in the hyperparameter selection, this study single-handedly could not conclude that the class of LSTM neural network would definitely underperform the much simpler HAR model.


LSTM - Predicted vs Actual VIX Prices During Test Period
actual
predictions
VIX PRICE
35
30
25
20
15
10
0
100
200
300
400
500


HAR - Predicted vs Actual VIX Prices During Test Period
actual
predictions
VIX PRICE
35
30
25
20
15
10
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
DATE

Conclusion

To conclude, this study has determined that the benchmark HAR model is indeed an extremely effective model for volatility, slightly outperforming the much more complex and resource-consuming LSTM model trained on the same amount of data while offering much more model interpretability. This is not entirely surprising, since volatility forecasting had been a decade old problem in quantitative finance, and experts had been on the search for a better model than the simple HAR model ever since its conception. It seemed that very few models had outperformed the original HAR model, which remains the most popular method of calculating volatilities today.

Due to time constraint, one potential improvement we did not have the opportunity to explore was cross validation. Many of the hyperparameters we chose for both the benchmark HAR model and the proposed LSTM model were based on empirical evidence, which was haphazard and time-consuming. Although in our experiment, the LSTM model slightly underperformed the HAR model, we still have hope that if we had more time to tune the model better with different activation functions and other training parameters, it might eventually surpass the benchmark.

However, this project had taken us on an extremely rewarding journey in helping us learn about neural networks and modelling. The most important learning we had during this project was the workflow to debug a project involving a neural network. During our project, we had trouble figuring out what to do after seeing our neural network predicted a straight line for all training data and all test data. We tried changing the batch size, adding more epochs, using more training data and much more, but none of the methods worked. Eventually, we had to create a smaller set of data with a single batch size of 2 just for debug purposes, and we printed out the output of each step ranging from the data loader to the loss calculation line by line, which not only narrowed down the problem to the model layers but also helped us identify other unrelated bugs within the data loader. We then tried to narrow down the problem further to trying different activation functions, hidden state dimensions and learning rates, eliminating the effect of each one until we found that the problem lay with all 0 initial neural network weights assumed by the model. Once we initialized the neural network weights with non-zero values, the model started to exhibit variation in its outputs. We have always thought of neural networks as a black box, but it turns out that if we have patience, it can be debugged in a systematic and efficient way.

IV. References

Fulvio Corsi, A Simple Approximate Long-Memory Model of Realized Volatility, *Journal of Financial Econometrics*, Volume 7, Issue 2, Spring 2009, Pages 174–196, https://doi.org/10.1093/jjfinec/nbp001

Marcelo Fernandes, Marcelo C. Medeiros, Marcel Scharth, Modeling and predicting the CBOE market volatility index, Journal of Banking & Finance, Volume 40, 2014, Pages 1-10, ISSN 0378-4266, https://doi.org/10.1016/j.jbankfin.2013.11.004.