Capstone Project 2
Forecasting S&P 500 Volatility: An Exploration of Predictive Models

1 Motivation

In the field of finance, asset managers and traders alike often analyze volatility for many practical purposes. For portfolio managers and risk departments, an understanding of volatility can help create accurate assessments and quantifiable measures of Value-at-Risk (VaR), which is essentially the probability adjusted notional value at risk of a portfolio given assumptions about moves in asset prices over a fixed period of time [1]. Thus, having better forecasts for volatility over fixed time intervals can help asset managers more efficiently allocate assets and help clients meet their investment goals. In other areas of finance, traders will trade volatility directly through options or other investment vehicles. The benefit of having more reliable volatility forecasts as an extra data point when making volatility trades is very straightforward for professionals working in this capacity.

2 Data Wrangling

The dataset for this project consisted of 5 years of daily historical price data from the S&P 500 Index ETF, SPY. The native dataset contained 1259 rows representing each date over the 5 years and 7 columns, which included the date as well as price data for each date. Outside of the date column, which was used as the index, all columns were numerical on a continuous scale (float values), except volume, which was an integer.

While the acquisition of the data was quite simple (it was imported from a downloaded CSV file from Yahoo Finance for free), there were some steps required to transform some of the columns into values that would be more useful for the project. In finance, it is common practice to view the change in stock price from day to day (or over a given time period) as the log difference in stock prices. For short periods of time, the log return is approximately equal to the discrete percentage return [2]. Once log returns were added as a column, daily realized volatility was then calculated, which was the target value to predict for this project. Since most practitioners in the field of finance typically view volatility as an annualized number, the daily realized volatility is then annualized to make numbers comparable to what portfolio managers and traders would see on a day-to-day basis [3]. While the initial dataset did not contain any null or missing values, transformation of the dataset throughout the project (via lag variables and calculating returns) did require handling and omitting rows. Restructuring the dataset to be a supervised learning problem was required to use machine learning algorithms for some of the modeling. This brought up the issue of not only missing values from lag variables, but also ensuring that the dataset was transformed in such a way that the target variable,

daily realized volatility, only had price data from previous days; this is necessary because in practice one would never have the advantage of seeing the current day's price history and be able to project volatility with those numbers.

3 Exploratory Data Analysis (EDA)

In order to gain some familiarity with the dataset, exploratory data analysis was performed. Since the goal of this study is to build a model to predict daily realized volatility, a good way to begin this process is to view price history data of the S&P 500 ETF to visualize what happens with volatility as price changes over time.

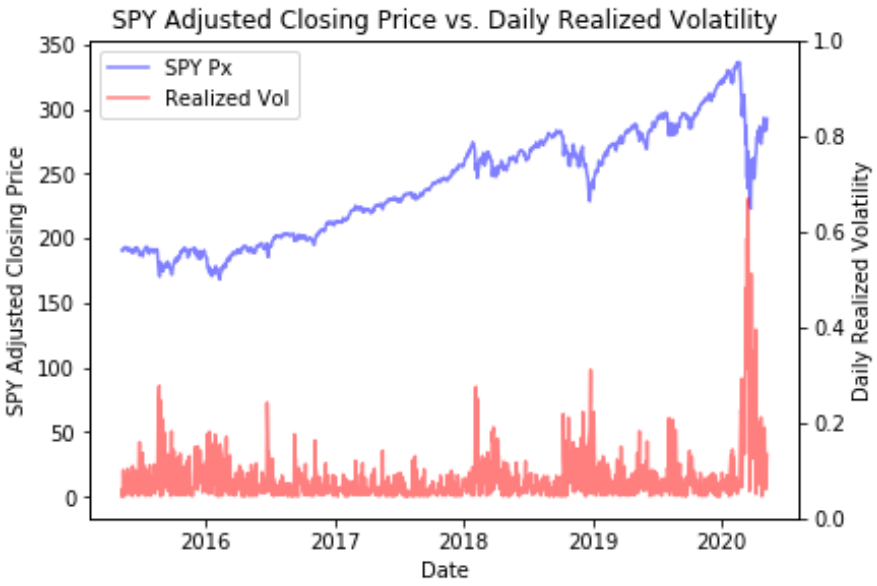

Figure 1: SPY Closing Price vs. Daily Realized Volatility

Figure 1 clearly shows what is commonplace knowledge in the field of finance, down moves in stocks, and especially equity indices, tend to be much more volatile than up moves. In addition, while spikes are somewhat short lived, there is some persistence, or a tendency for realized volatilities to stay elevated once they have risen. While the mean of the realized 1 day volatility over this 5 year sample is .1361, the standard deviation of .1905 reflects the fact that periods of low volatility tend to outweigh periods of high volatility over time and volatility tends to revert back to historically lower levels.

While stocks are not stationary processes over time, volatility is a stationary process over time, although it can be described as having different regimes in smaller time frames. The fact that volatility is a stationary process is another way of saying that it is a mean-reverting process. Coupled with the fact that quantifying risk from volatility is useful for many finance industry professionals, the mean reverting nature of realized volatility makes it an interesting candidate for forecasting problems.

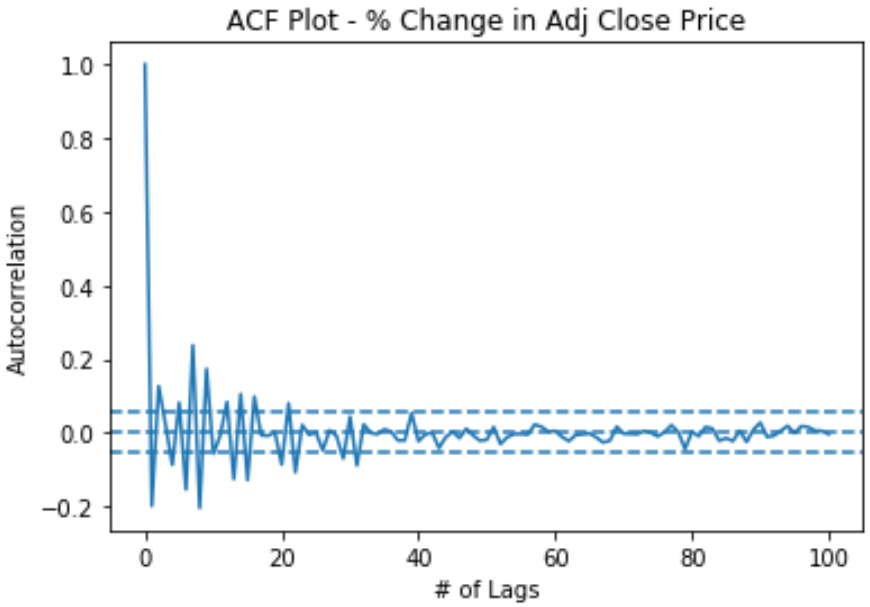

Figure 2: ACF Plot of % Change in SPY Closing Price

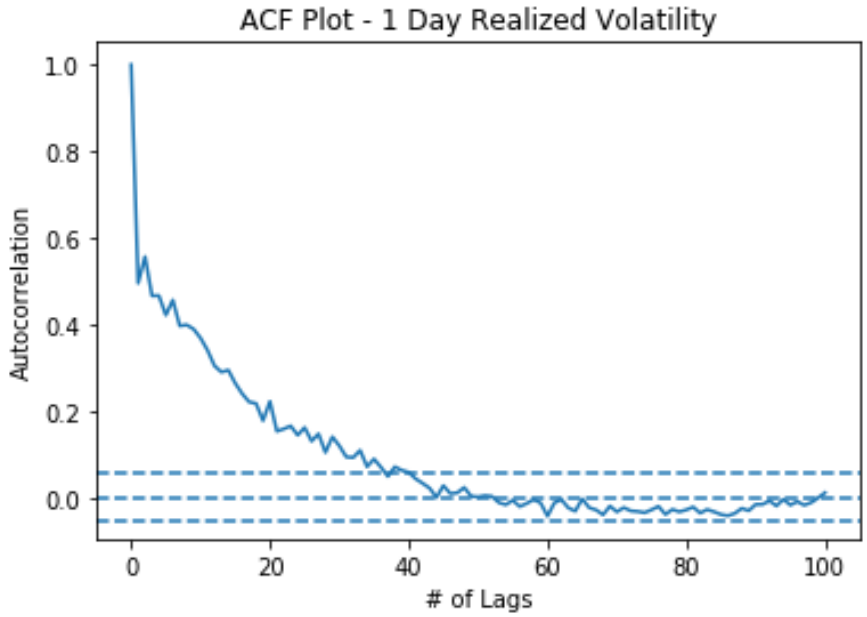

Figure 3: ACF Plot of Realized Volatility

Figures 2 and 3 are meant to highlight a specific characteristic about volatility. As inferred from the SPY daily price versus volatility chart, there was a hunch that volatility persisted over short periods of time. Figure 3 shows that indeed volatility is persistent in a statistically significant way for at least 30 lag periods, or days in this case. As such, it is a reasonable belief that past volatility is indeed predictive of future volatility to some extent. Contrasting this with Figure 2, which is an ACF plot of the percent change in adjusted closing prices, that series is predictably not persistent; if past daily returns were indeed predictive of future daily returns, this relationship would be exploitable by stock traders and investors until no such

correlation existed. Figure 3 also highlights the need for models that are specific to forecasting time series that are autoregressive in nature; these models will be discussed in more detail in the next section.

4 Models

The framework for modeling and analysis is discussed in detail below. In general, the focus of the modeling was on one-step-ahead forecasts, or forecasts for the next day in time. Multi-step-ahead forecasts have the potential issue of creating predictions with an error so large the predictions would be rendered useless. The first step was establishing a null model as a baseline point of comparison to see how all models performed versus each other and versus the null model. Standard ARIMA models for time series forecasting are not appropriate given the autoregressive nature of volatility referenced above. GARCH models are commonly used in finance for predicting volatility and will be the first models discussed below and compared to the null model for performance. The discussion will then move towards other models such as neural networks and other machine learning models and will conclude with a comprehensive discussion about which models would be best fit for use in practice and why.

4.1 Null Model

The null model was constructed using the previous day's realized volatility value as the projection for the next day. While very simple, this model is not trivial in the slightest since volatility is known to be an autoregressive property and empirically past volatility does to tend to be indicative of future volatility. The null model had a mean absolute error (MAE) of .1201. When looking at volatility, it can be easier to think in terms of points, so in practice volatility numbers are often multiplied by 100. In this case, we can interpret the MAE to mean the 1 day lagged prediction of realized volatility has an absolute error of 12.01 volatility points. With an average realized 1 day volatility of 13.61 over the sample period, this seems quite high. However, the standard deviation of the 1 day realized volatility is 19.05 volatility points, so the prediction is still within 1 standard deviation of the mean. Nevertheless, this is the benchmark to beat for models that were evaluated in this study.

4.2 GARCH (1,1) Fixed Rolling Window

For time series forecasting problems, there is an issue of getting suitable projections based on the work trying to be accomplished. To make relevant projections for the field of finance, it is desirable to have predictions for the next period in time (i.e. if daily predictions are the goal, it is important to have the prediction for tomorrow based off the past x days of data). There are 2 common ways to accomplish this: fixed rolling windows and expanding windows. The first model evaluated is a fixed, rolling window, which uses 21 days of historical data for predicting volatility. As shown in the EDA section, the volatility in this dataset shows significant persistence past 20 days and 21 trading days is significant because it represents one month of

daily data on average. For this fixed rolling window model, a standard GARCH (1,1) model is used, which means that 1 period of past residuals and 1 period of past variance is used in making forecasts. GARCH, which stands for generalized autoregressive conditional heteroskedasticity, is the generalized version of ARCH, and the distinguishing difference is that GARCH also looks at past periods of variance, or lagged variance, in addition to looking at lagged residuals (which is what the ARCH model looks at).

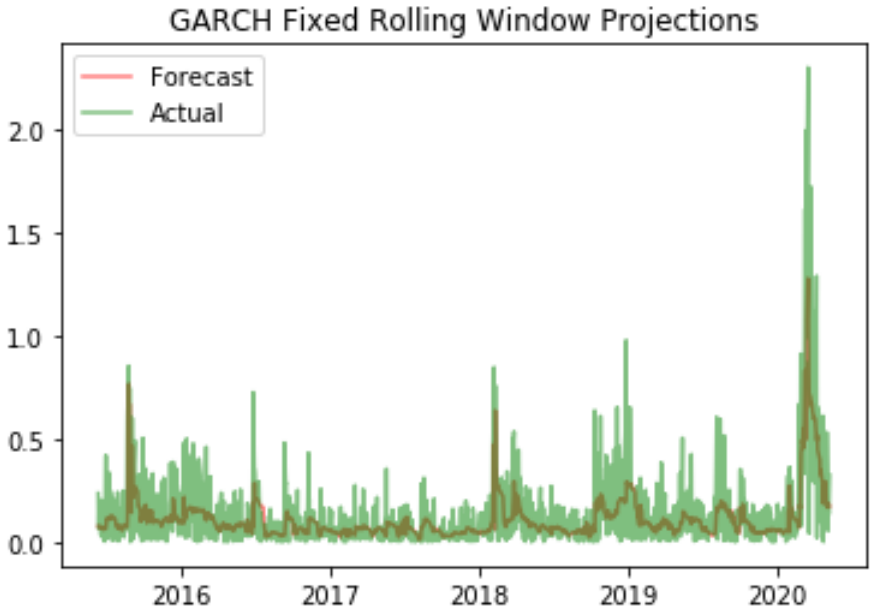

Figure 4: GARCH (1,1) Fixed Rolling Window Projections

The predictions for this relatively simple GARCH (1,1) model using a fixed, rolling window of size 21 days, actually performed quite well. The MAE was .0845, or just 8.45 volatility points, which is a significant improvement from the results of the null model. Again, while the 8.45 volatility points is reasonably large compared to the average volatility of 13.61, it seems to be a decent result when taking into consideration the standard deviation of 19.05, as the MAE lies well within 1 standard deviation of the realized volatility.

4.3 GARCH (1,1) Expanding Window

In contrast to the fixed rolling window used above, where a fixed number of observations is used and that window slides as time moves forward, this next model uses an expanding window. For comparison, GARCH (1,1) was used to model the underlying volatility process to give a comparison of the two methods of interpreting and predicting new data as it is made available. The expanding window utilizes all available data from the past to make projections about each new day in the future. This method allows for users of the model to simply add new data points to the dataset as they become available (in this case daily). While all data is incorporated into the model, newer data points still tend to have the biggest effect on model parameters at each step forecast.

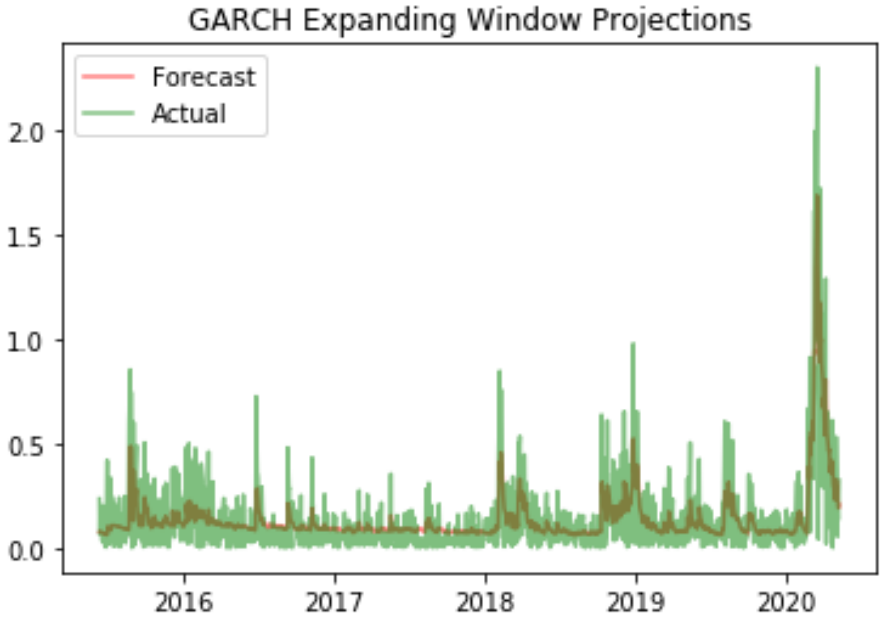

Figure 5: GARCH (1,1) Expanding Window Projections

Figure 5 shows that the expanding window framework using the same GARCH (1,1) model for predicting the underlying volatility process seems to, on average, do a better job of predicting the large spikes in volatility when compared with the fixed rolling window GARCH (1,1) model. Quantifying this observation by calculating the MAE of 8.16 confirms that this model is also more accurate across the whole forecasting window.

4.4 EGARCH (1,1) Expanding Window

Within the field of finance, equity volatility tends to exhibit a pattern of asymmetry, where heightened volatility leads to more volatility and periods of low volatility also tend to persist [4]. In order to help model this asymmetry, there are several different variations of the basic GARCH model to help better model this asymmetry. This project looks at two common variants of the GARCH model, EGARCH and GJR GARCH. In terms of parameter specification within the Python arch_model package, there is an additional parameter 'o' which accounts for the asymmetry in volatility when modeling; this is in addition to the 'p' and 'q' parameters, which allow specification of lagged residuals and lagged variance, respectively.

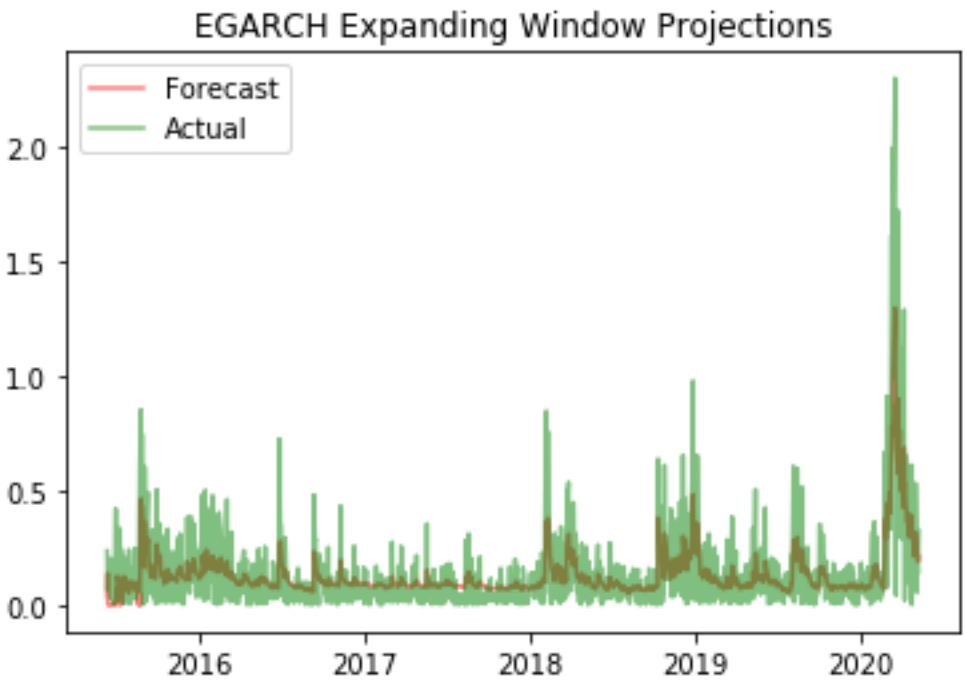

Figure 6: EGARCH (1,1) Expanding Window Projections

Visual inspection of Figure 6 shows that the forecasts missed the extremity of the big volatility spike in 2020, but perhaps did a better job of capturing some of the smaller spikes. The MAE for this model was the lowest of all models tested, coming in at 7.98. Although the MAE is a slight improvement from the GARCH (1,1) model, there could be an argument for using the GARCH(1,1) model in practice, as more accurate predictions on large volatility spikes may be more important than being accurate in quiet periods of low volatility.

4.5 GJR GARCH (1,1)

The next volatility model evaluated in this project was the GJR GARCH (1,1) model. This model is similar to the EGARCH model in that it accounts for asymmetry in volatility, but the formulas for each have slightly different parameterizations for how they capture that asymmetry [5].

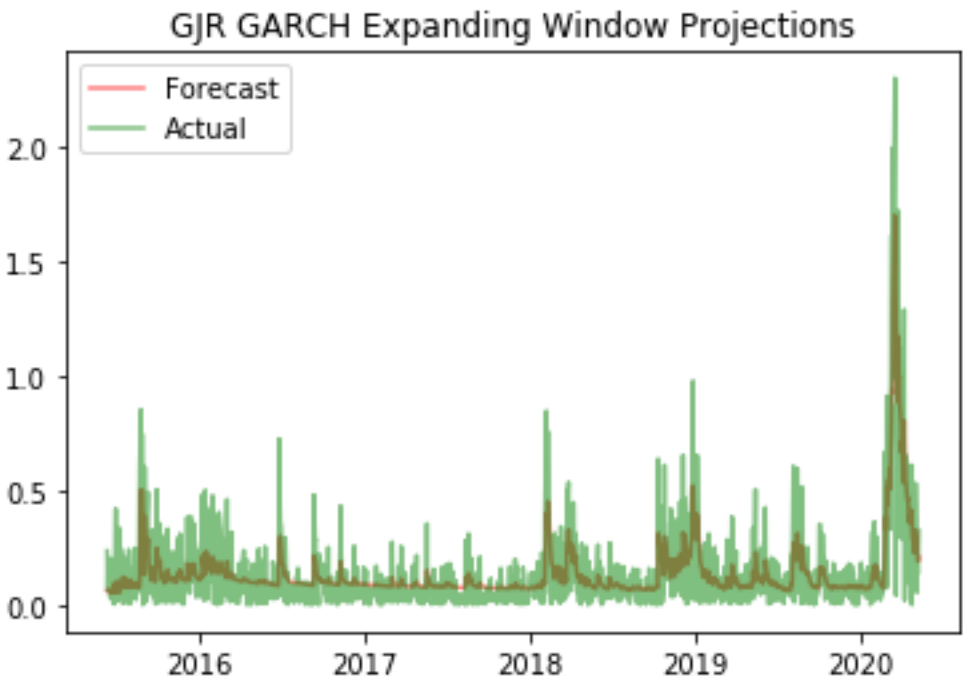

Figure 7: GJR GARCH (1,1) Expanding Window Projections

Analyzing the plots of forecasted versus realized volatility from the GJR GARCH (1,1) model, the predictions appear to match more closely to those generated by the GARCH (1,1) model in that the GJR GARCH model seems to do a decent job of capturing the magnitude of spikes in volatility relative to the EGARCH model. The MAE for the GJR GARCH predictions is 8.02, which is just .04 higher than the EGARCH model. Given the underlying ability of the GJR GARCH model to take into account asymmetries in returns in addition to better performance during times of high volatility, the difference in MAE between the two models is negligible. For practical and industry use cases, the GJR GARCH would be the recommended model due to its ability to more accurately capture large spikes in volatility as well as maintain a sufficiently low MAE.

4.6 Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN)

After exploring predictive volatility models, the next task this study tackled was implementing and testing the predictive power of an LSTM RNN as well as various machine learning models which will be discussed in future sections. When constructing the framework for this model and others, it was useful to first "reframe" the dataset to fit into the structure required by Python's many machine learning algorithms [6]. This was accomplished by shifting all the data back in time the by the appropriate amount (i.e. for a 5 day lag period, the data was shifted back 5 time periods so that the prediction variable of interest, in this case 1 day realized volatility, can only see data from the past, and not from the present or future). For the LSTM model, a simple structure of 1 day of lagged observations was used. In addition to the lagged realized volatility, this model also took as input the log difference of the lagged high and low price of the S&P 500 on the notion that there

might be information about future volatility embedded in the full range of prices from the previous day as opposed to just being limited to seeing the change in price on a close-to-close basis. The LSTM model was fit with 50 neurons in the first hidden layer and 1 neuron in the output layer [4]. Epochs control how much fitting is done with the test set, so this was an important number to determine when constructing the LSTM model. Epochs used in this model were 750, as it struck the right balance between computation time and MAE. Figure 5 below shows that the MAE actually started increasing again as the number of epochs went up.

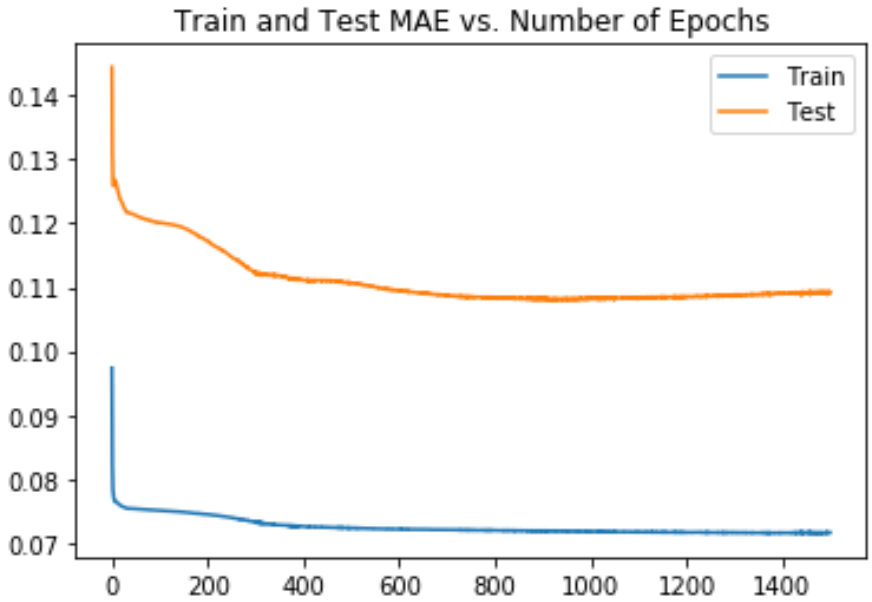

Figure 8: Train & Test MAE by Number of Epochs

One drawback of this model and the others discussed going forward is that they require a training and testing set. The first half of the data was used for training and the second half was used for testing for this model and all subsequent models.

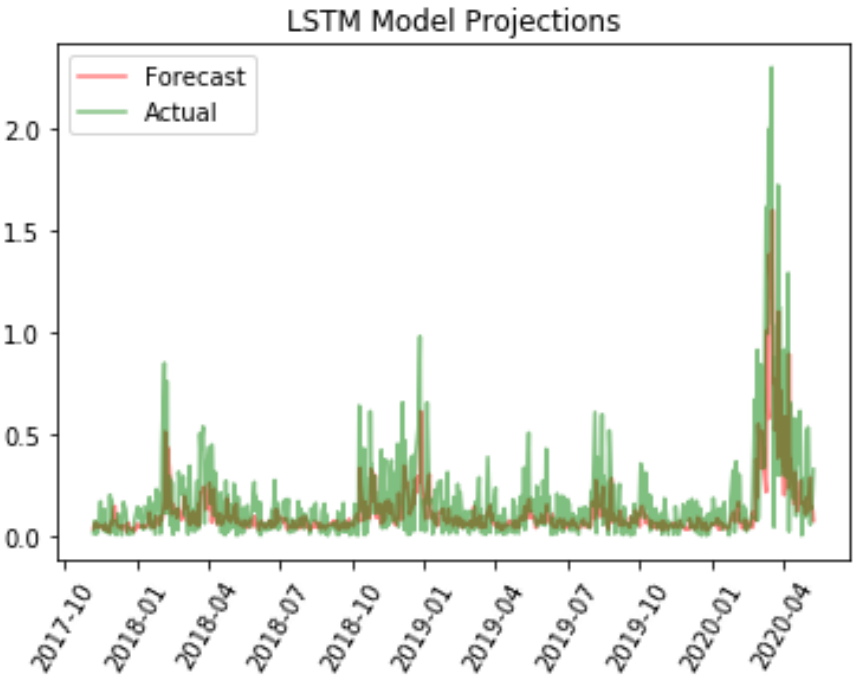

Figure 9: LSTM RNN Projections

At first glance, the LSTM RNN model seems to do a good job at capturing spikes in volatility. The MAE of 10.87 though tells us that this model did quite a bit worse than any of the GARCH models or variants. Upon closer examination, possible reasons for this are that while the big spike in March was captured and predicted quite well by the model, it does not perform as well in some of the other, smaller spikes. With an MAE still below the null model, this model could show merit in future work with different neural network specifications as well as potentially including other dependent variables.

4.7 Random Forest Regressor

Random forests are one of the most powerful and popular modern machine learning models used for prediction. Random forests build upon decision trees by specifying several trees with different, random parameters. With a collection of trees, the forest becomes a worthy prediction model, whereas a lone decision tree is usually a quite poor predictor. The random forests were tuned for several different hyperparameters and then this optimal set of parameters was used for predicting volatility with a feature set of 5, 10 and 21 days of lagged volatility data and high/low data. The industry significance for these periods of lagged data are 1 week, 2 weeks and 1 month of data in trading days.

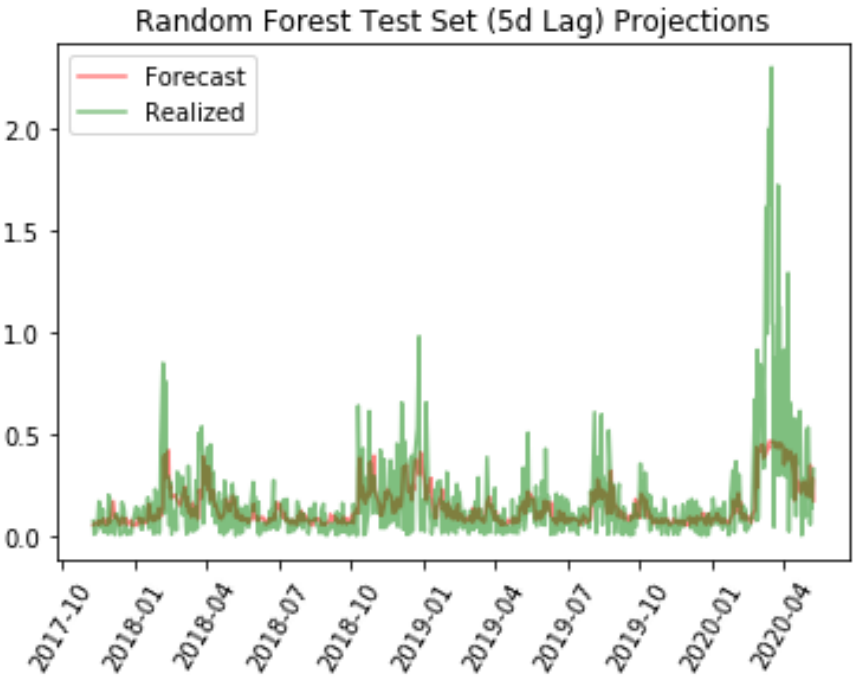

Figure 10: Random Forest (5d Lag) Projections

Performance among the 3 different random forests models was quite similar both in visual performance and when comparing MAEs. Since the 5 day lag is the simplest of the 3 models, this was used for the graphical depiction of predictions from these models. The MAE for the random forest models were 11.03, 11.02 and 11.10 for the 5, 10 and 21 day models, respectively. These numbers suggest what we could deduce visually: the random forest models are the weakest performers thus far and would not serve useful in further developments or applications of this project. Also worth mentioning is the poor performance in predicting the magnitude in periods of high volatility

4.8 Multiple Linear Regression

The next models used for evaluation were simple multiple linear regression models which used the same feature sets as the random forest models: 5, 10 and 21 days of lagged volatility and high/low price observations.

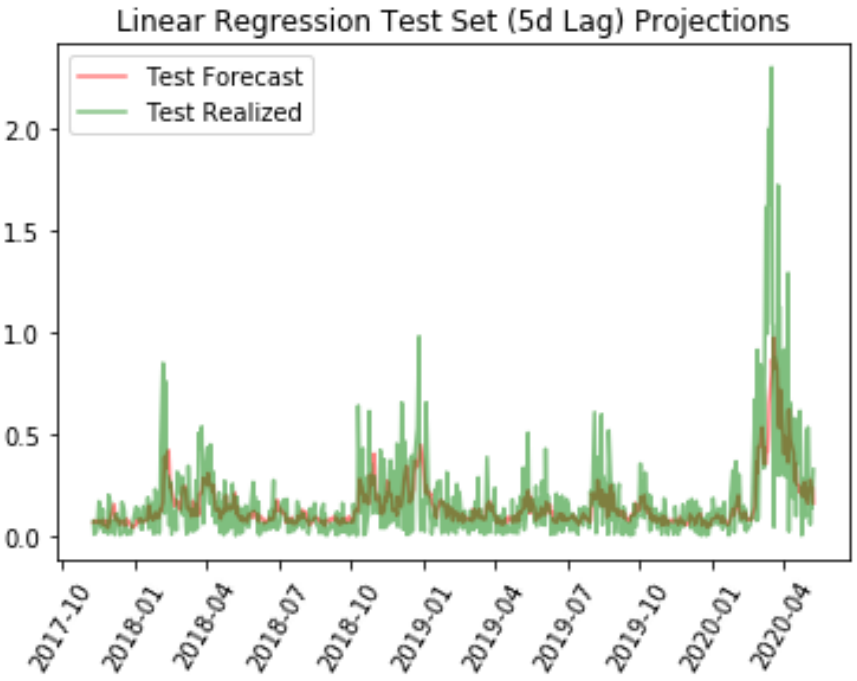

Figure 11: Linear Regression (5d Lag) Projections

Similar to the evaluation of the random forest models, just one of the three model prediction plots was shown here because all models produced similar results. The 5 day lag model was again the simplest model, but also performed slightly better than both the 10 and 21 day models. The MAEs were 10.72, 10.95 and 11.29 for the 5, 10 and 21 day models, respectively. Contrary to the random forest models, there was actually some differentiation in the MAEs of these 3 models, but even the best model still achieved an MAE of just 10.72. While that makes this the best non-GARCH model yet, it still has quite inferior predictive power relative to any of the GARCH models, but still much better than the null model. The MAE of 10.72 is just slightly better than the LSTM model, but the LSTM model did a better job of predicting the magnitude of spikes in volatility.

4.9 Symbolic Regression

The next model tested was symbolic regression, which uses the gplearn package in Python to perform Genetic Programming (GP) with symbolic regression. The main idea in symbolic regression is that "generations" of programs are evolved from ones that came before it by selecting the fittest individuals (or best predictors) [7][8]. As such, there are many different hyperparameters to choose values for, but one of the most important is number of generations. Increasing the number of generations can significantly increase computation time, as the 10 generation model implemented in this project took 33.6 seconds and another 50 generation model not reported here took 195.5 seconds to run.

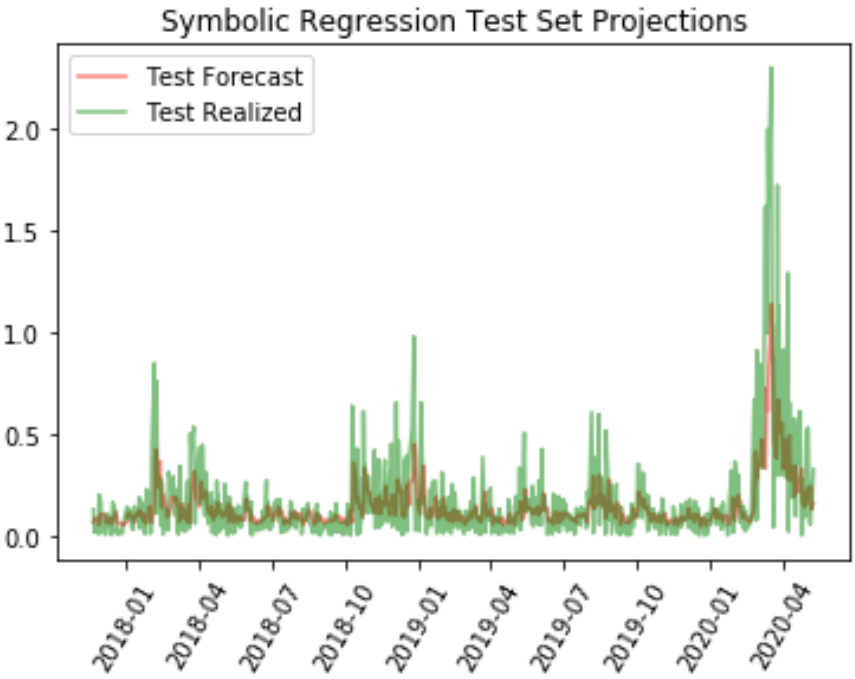

Figure 12: Symbolic Regression Projections

Examining the plot of forecasted volatility versus realized, there does not seem to be any notable areas of outperformance or underperformance compared to the linear regression model with 5 days lag. The MAE of 10.95 confirms that the symbolic regression model performs just a bit worse than the best linear regression model. Overall, it seems to not capture the magnitude of spikes in volatility very well. As with the linear regression and random forest models, this would likely not be suitable for real-world use cases.

4.10 Extreme Learning Machine (ELM)

An ELM is a single-layer feed-forward neural network (SLFN). An important takeaway from ELM theory is that the hidden layer does not need to be tuned and is independent of the training data [9].

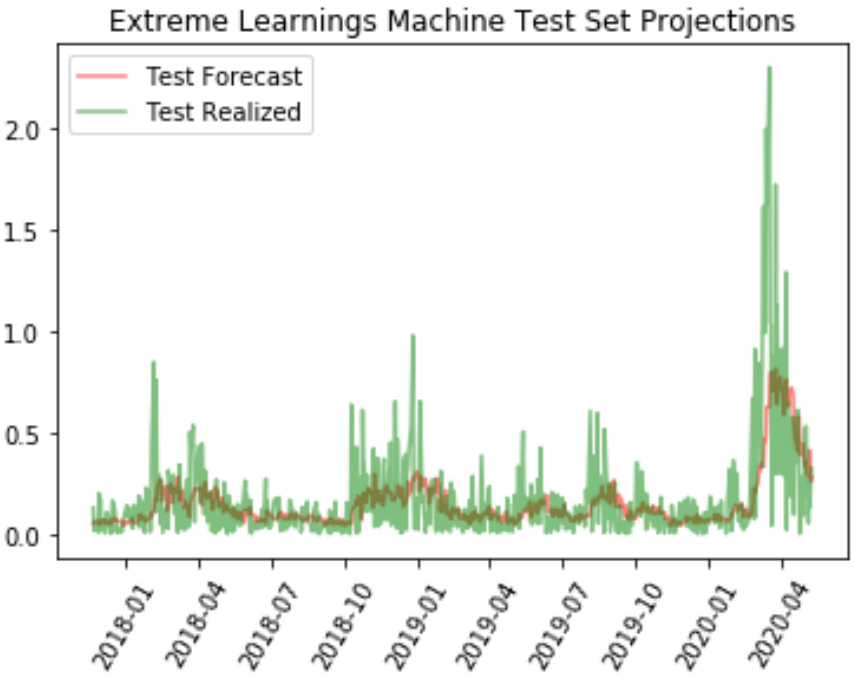

Figure 13: Extreme Learning Machine Projections

The ELM projections definitely do not appear as accurate as either the symbolic regression or linear regression predictions, but appear to be similar, if not better than the random forest projections. Calculating the MAE yields 11.34, which actually means the ELM model did even a bit worse than the random forest models. Either way, it is safe to say that this model is also not extremely useful for practical applications.

4.11 AdaBoost Regressor

The last method used in predicting realized volatility for this project was a popular boosting method, AdaBoost. Boosting is similar to random forests in that it was designed to build on the individual weakness of a given decision tree. However, it is different in that boosting typically produces relatively weaker trees than random forests, but the idea is that the weaknesses of each tree are used to strengthen the overall predictive power of the final model. The main parameters of interest here are the number of estimators (or trees) and the learning rate. For this model, 500 trees were used with a learning rate of .0001.

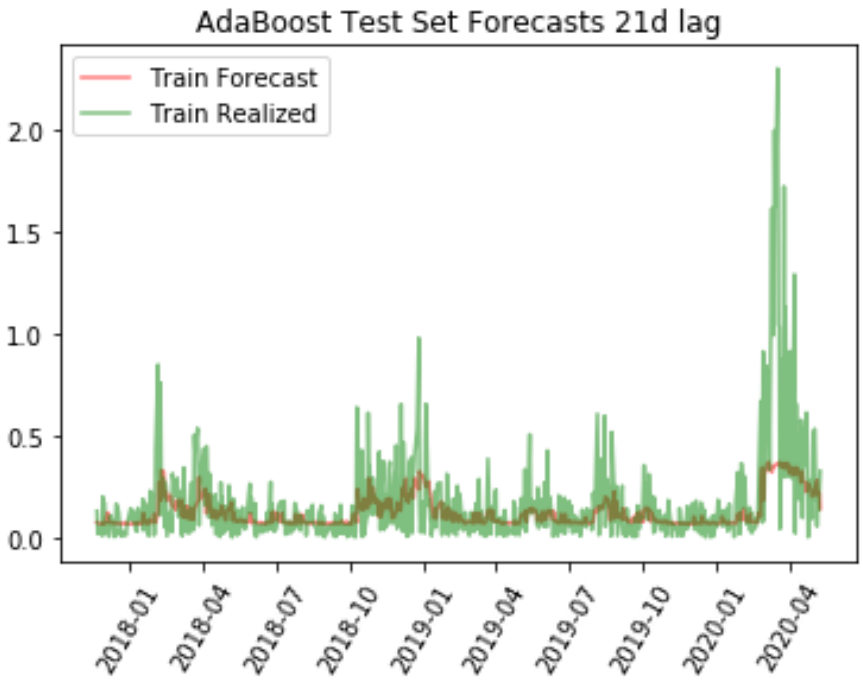

Figure 14: AdaBoost Projections

The AdaBoost predicted realized volatility graph shows that, similar to the random forest models, it failed to capture the large volatility spikes in early 2020 with any reasonable degree of accuracy in regards to magnitude. Although it has an MAE of 11.12, which is among the worst of all models, it is also visually one of the worst due to its poor performance in periods of high volatility rendering this model not fit for use in practical applications.

4.12 Modeling Summary

General Model Classification	Model	Mean Absolute Error (MAE)
Null	Null Model	0.1201
GARCH	GARCH (1,1) Fixed Rolling Window	0.0845
GARCH	GARCH (1,1) Expanding Window	0.0816
GARCH	EGARCH (1,1) Expanding Window	0.0798
GARCH	GJR GARCH (1,1) Expanding Window	0.0802
Neural Network (Recurrent)	LSTM RNN	0.1087
Machine Learning - Random Forest	Random Forest Regressor (5d Lag)	0.1103
Machine Learning -	Random Forest Regressor	0.1102

Random Forest	(10d Lag)	
Machine Learning - Random Forest	Random Forest Regressor (21d Lag)	0.1110
Machine Learning - Linear Regression	Linear Regression (5d Lag)	0.1072
Machine Learning - Linear Regression	Linear Regression (10d Lag)	0.1095
Machine Learning - Linear Regression	Linear Regression (21d Lag)	0.1129
Symbolic Regression	Symbolic Regression (21d Lag)	0.1095
Neural Network (Single-Layer Feed-Forward)	Extreme Learning Machine (ELM)	0.1134
Machine Learning - Boosting	AdaBoost Regressor (21d Lag)	0.1112

Table 1: Model Summary

The best performing volatility prediction models were the GARCH and its model variants. Fixed and expanding window frameworks provided similar performance, with expanding window models providing slightly better performance. The best models as measured by MAE were the EGARCH and GJR GARCH expanding window models. Although the GJR GARCH had a slightly higher MAE, it may make more sense to use this model in practice as it did a better job of capturing the magnitude in large volatility spikes.

While all models presented in this project ultimately achieved a lower MAE than the null model, the non-GARCH based models performed much poorer. The MAE of the LSTM model was much higher than any of the GARCH models at 10.85, but it did show a decent ability to capture the magnitude of spikes in volatility when compared to other non-GARCH models, especially the random forest and AdaBoost models, which did the worst job of capturing these spikes.

5 Future Work

5.1 Modeling

While the scope of this project was intended to cover various methods for forecasting realized volatility, there is certainly more work to be done within each of the modeling categories. With the GARCH and related models, more parameters could have been tweaked, such as changing the underlying distribution assumption of the volatility process (i.e. a Student's t-distribution). In addition, the neural network models just scratched the surface of working with such models and there is certainly a lot of exploration to be done to see if model accuracy could benefit from further tuning of hyperparameters and model inputs. Lastly, the random forest and AdaBoost models were just two machine learning models of many that might be

appropriate, or even more appropriate, for this project; thus, testing a broader swath of machine learning models would also be included in future work.

5.2 Forecasting

This study focused on the simplest and most practical use case of volatility forecasts, which was one-step ahead forecasts. Future versions of this work could look into multi-step forecasts, perhaps a week or two weeks out into the future. As seen from the results of this project, the MAE was quite significant when compared to the average realized volatility, so multi-step forecasts were not a part of this work because they would likely come with such a large MAE it would render the predictions useless from a practical standpoint. Additionally, realized volatilities could have been averaged over the course of a week or two weeks. This would have the impact of making each one-step ahead forecast less significant and therefore provide more stability to forecasts. Using a model in this way, it would be beneficial to pay attention to the trend in 5 and 10 day predicted volatilities. For example, if the 5 or 10 day predicted volatility starts to trend up, it could be indicative of a regime shift in volatility and portfolio managers and traders could position accordingly.

5.3 Distribution of Realized Volatility Predictions

Something not covered in this project that would certainly be useful from an industry perspective would be to include some kind of visualization and analysis on not only realized volatility forecasts, but also the distribution associated with each point estimate. This would be another potential way to identify volatility regime shifts. The idea is that if the distribution of volatility outcomes widens, then perhaps realized volatility in the near future will be higher and vice versa.

5.4 Breadth of Securities Analyzed

One last obvious area of future work for this study would be to include analysis on a large list of securities. This could include more broad indexes, sector index ETFs, and individual securities. Individual equities come with the issue of filtering out earnings data, as these moves are often significantly higher than a stock's daily realized volatility. This would allow portfolio managers and traders to have updated volatility forecasts before the trading day begins for any and all securities of interest.

6 Conclusions

The GARCH, EGARCH and GJR GARCH expanding window models performed the best of all models tested. Although the EGARCH model had the best MAE by a marginal amount compared to the GJR GARCH model, the GJR GARCH model is most likely to be preferred by industry practitioners due to its outperformance in capturing the magnitude of realized volatility spikes. Of the non-GARCH models, the LSTM model showed the most promise, as it not only had the lowest MAE of the remaining models, but also did a reasonable job capturing the magnitude of volatility spikes. As alluded to in the future work section, this analysis would be best

suited for industry professionals by expanding on the work to include all securities of interest and further fine-tuning of the top models to get the best predictive performance.

7 References

[1]: https://en.wikipedia.org/wiki/Value_at_risk#:~:text=Value%20at%20risk%20(VaR)%20is,period%20such%20as%20a%20day.
[2]: http://mktlssns.blogspot.com/2010/04/why-we-use-log-returns.html
[3]: https://en.m.wikipedia.org/wiki/Volatility_(finance)
[4]: https://www.nber.org/papers/w11188.pdf, pg. 21
[5]: https://www.nber.org/papers/w11188.pdf, pg. 21-22
[6]: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/
[7]: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/
[8]: https://gplearn.readthedocs.io/en/stable/intro.html#representation
[9]: https://towardsdatascience.com/ml-approaches-for-time-series-4d44722e48fe