# Predicting Stock Volatility with Time-Series Analysis

Spring 2019

Springboard Capstone Project 2

Milestone Report 2

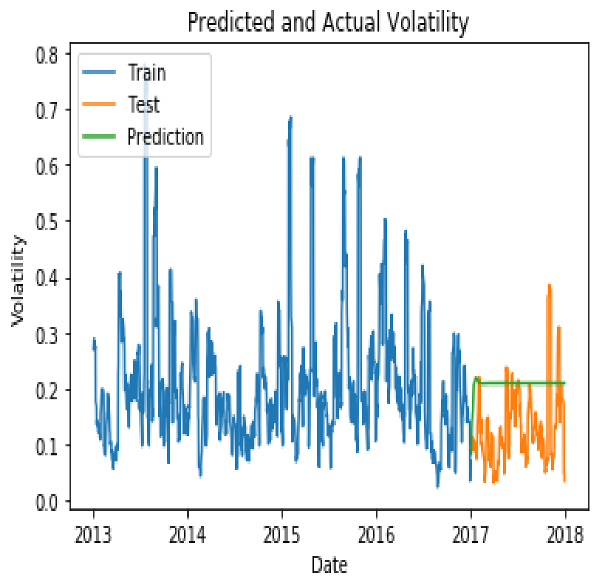
Brian J Zamkotowicz

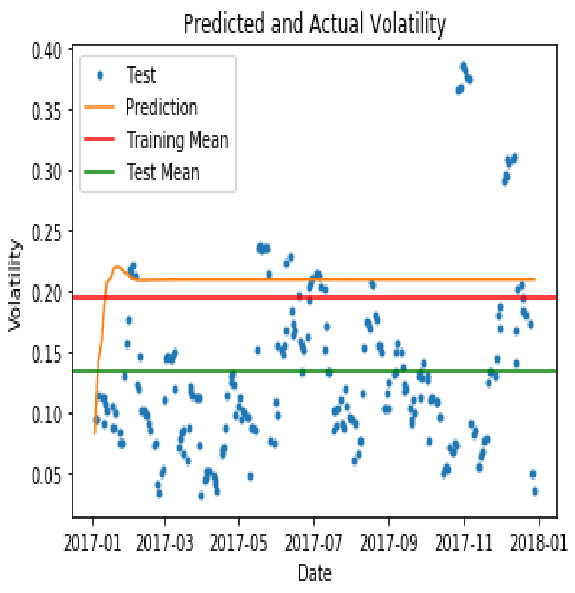
## Applications of Machine Learning to the Problem

The next phase of the project was to move on to model-building and actually attempting to predict historical volatility.  Since I had already determined that there were seasonal and trend components to the movement of the stock I decided to focus first on ARIMA (Autoregressive Integrated Moving Average), and then on Prophet--a model built by Facebook and recently made available to the public--both of which are capable of utilizing these factors.  Specifically, I would use auto-ARIMA, a model with built-in parameter tuning, which could automatically find the best parameters for the model using a methodology similar to grid search.

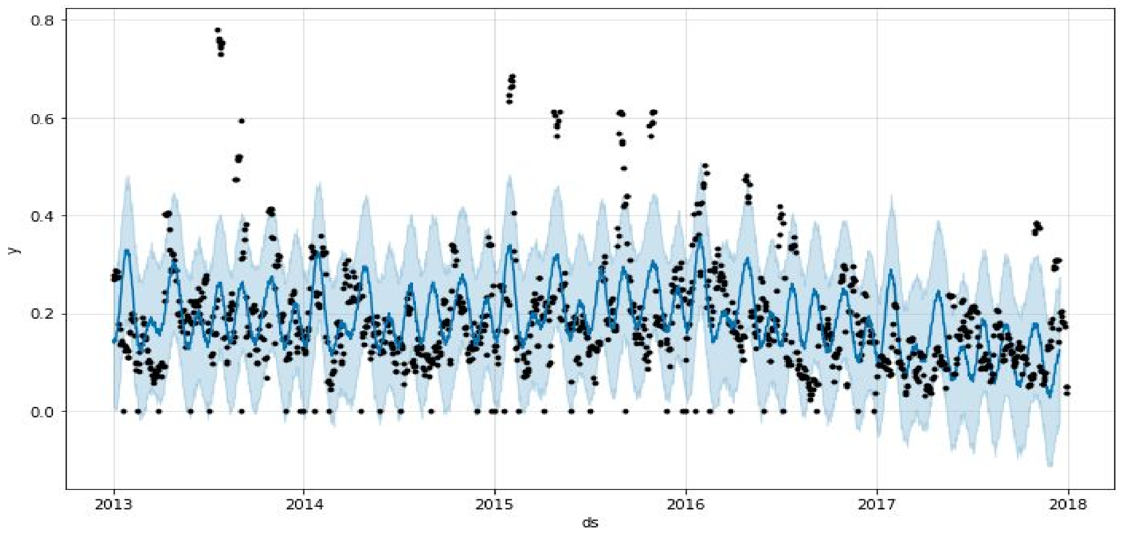
I began by running two ordinary least squares models using the statsmodel package.  One model focussed solely on historical volatility, while the other include several other features (such as implied volatility, volume and skew).  The model with additional features outperformed the more basic model (in terms of AIC, or Akaike Information Criteria), which led me to believe, that additional features might improve forecasts going forward.

In order to begin the search for the best predictive model I had to start with a baseline.  I chose an auto-ARIMA model as the starting point for attempting to predict Microsoft stock's 10-day historical volatility.  This initial model seemed to start off very close to the actual volatility but then quickly gravitated to a spot slightly above the mean  of the training set (2013-2016) where it stayed through most of 2017.  The straight line drew my attention to an issue that would be present in several of the models.  A forecast built in this fashion was attempting to predict a full year's data based on data for the previous 4 years, rather than taking into account the most recent measure of the stock's volatility.  For this reason the model performed managed to track volatility reasonably well for a few days, but quickly degraded.

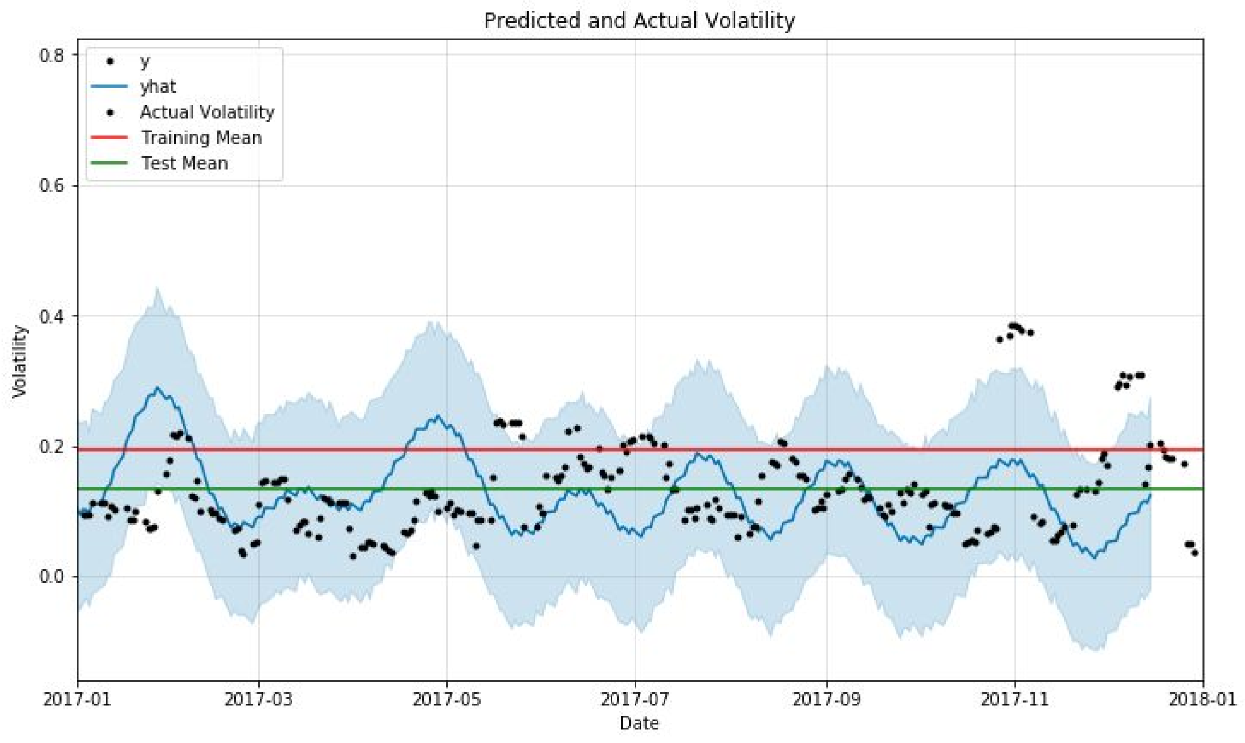




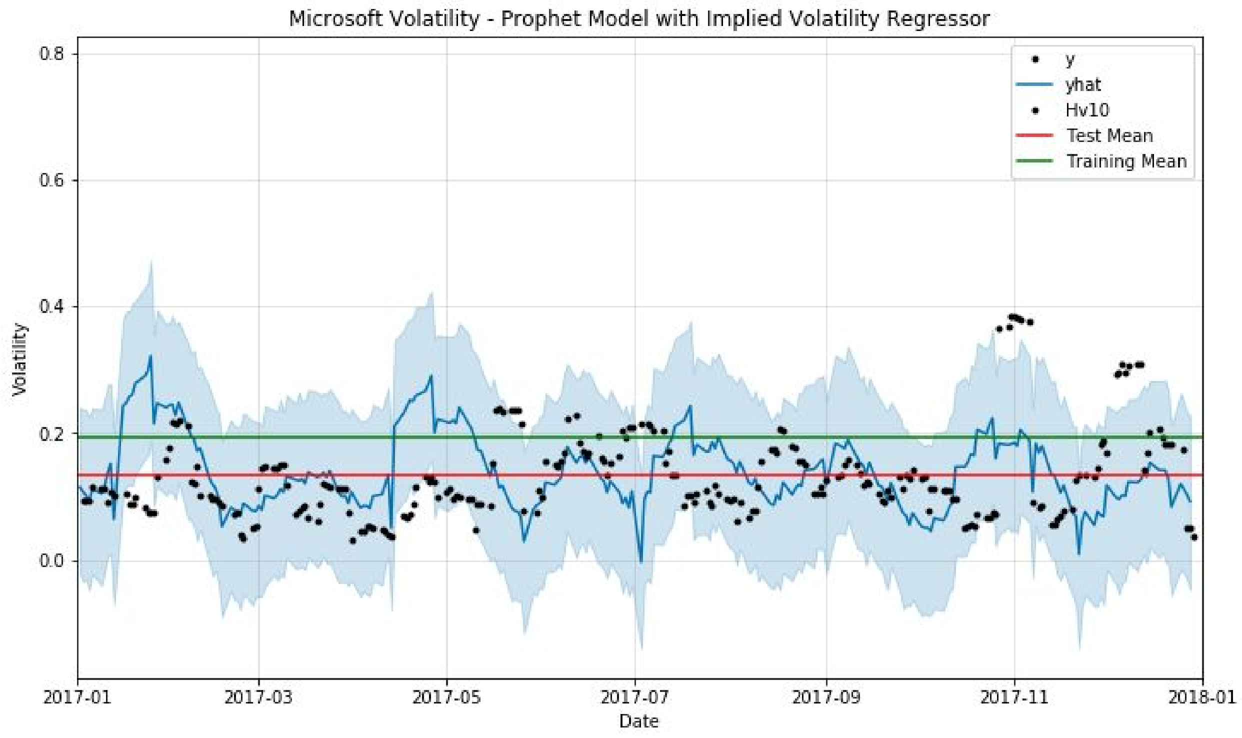
The next foray into modeling utilized Facebook Prophet (FBProphet) for Python. This model claims to outperm ARIMA, and I wanted to put that claim to the test. The first look at the model showed immediate improvement.



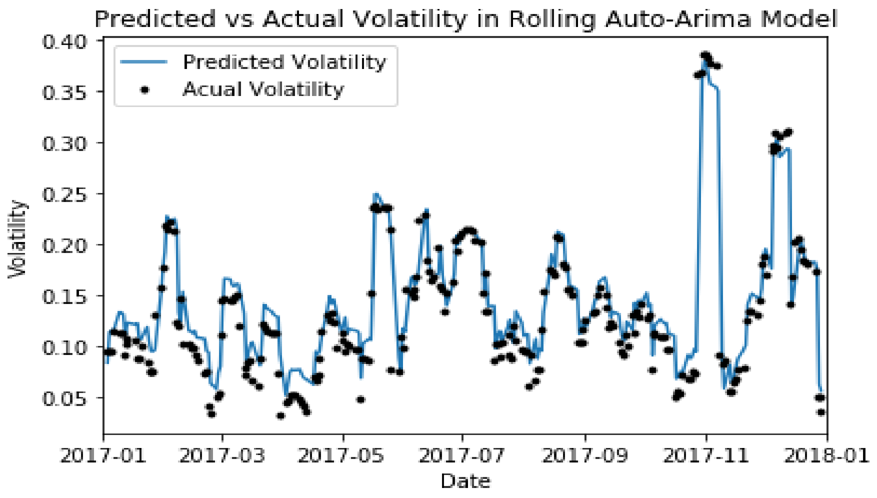
At the very least, the model above shows an attempt at modeling based on current volatility as well as seasonal patterns and trends. Close examination of 2017 shows that the model captured the market’s move towards decreased volatility over the period (a general downtrend in 2017 predictions). A closer look at the 2017 forecast shows it to be significantly better than the baseline model. In addition to what can be seen on the chart, this model showed improvement in both root mean-squared error (rmse) and r-squared when compared with the baseline model.



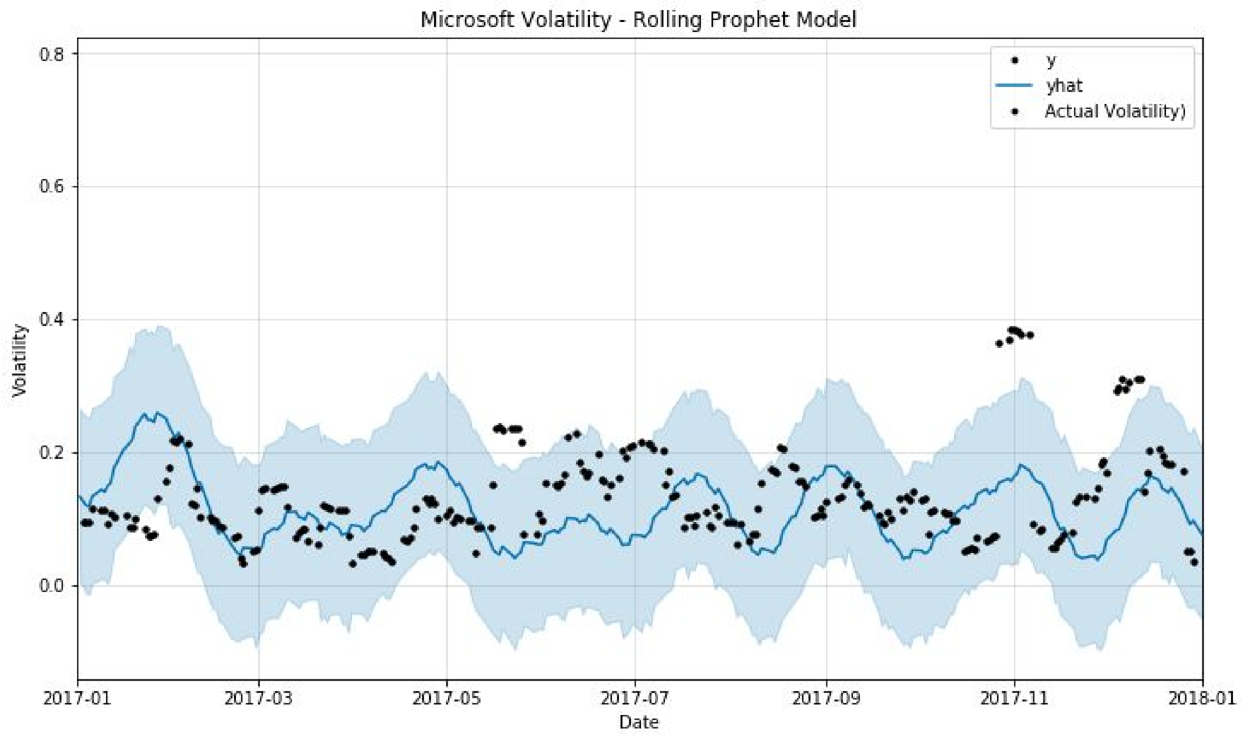
After proving that the initial Prophet model was more effective than the original auto-ARIMA baseline, I wanted to see if the Prophet model could be improved even further. I thought that by adding additional features, the forecasting accuracy of the this model might be improved. Prophet allows added regressors to be built into the model. Since the heatmap presented earlier showed a reasonable correlation between implied volatility and historical volatility I chose implied volatility as the feature to add. This attempt showed marked improvement over each of the first two in terms of R-squared and RMSE, meaning it was the most accurate model constructed so far. Once again though. R-squared value remained negative.



One thing I noticed about the models is that they seemed to build-in seasonal and trend patterns but not apply as much weight to recent values as I thought would be appropriate. After contemplating this issue I realized that this was a function of the way the previous models had been structured. The models essentially used data from 2013 through 2016 to build a forecast for the entirety of 2017. Only the regressor model had any 2017 data to work with (daily implied volatility) when making prediction. I speculated that fitting the model on all the data up to the day of the prediction might remedy this. To accomplish this I built a loop that would instantiate a new auto-ARIMA model for each day and output the result to a new dataframe.

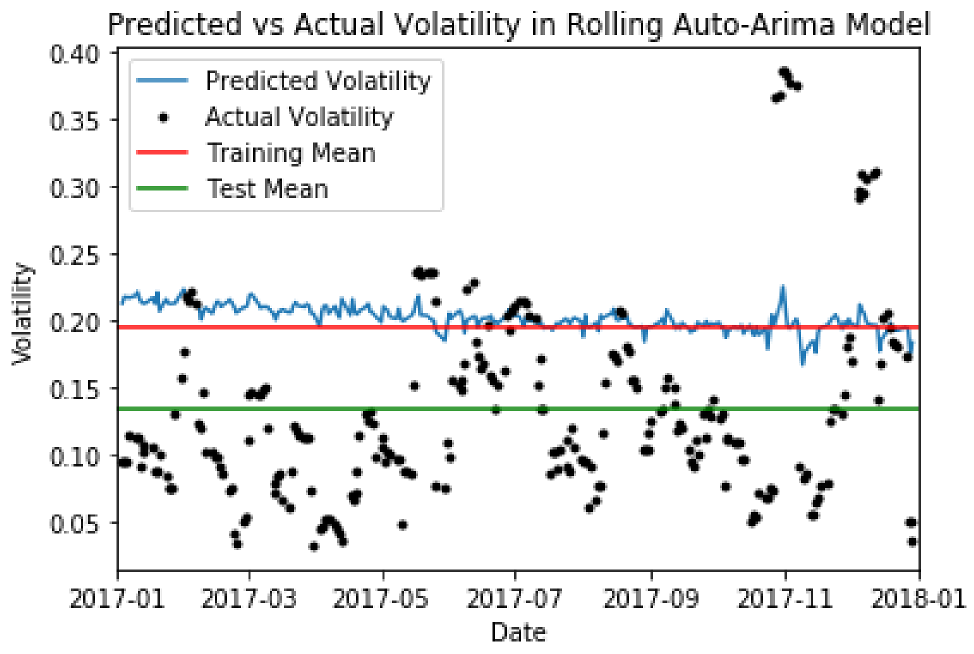


As seen above, the new rolling model was wildly successful in comparison to the previous three. It showed great improvement in RMSE, in addition to raising the R-squared value to .718 from a a value of -.688 in the next closest model. I then went ahead and created a new version of the prophet model to see how it would function if built in a similar, rolling, fashion.

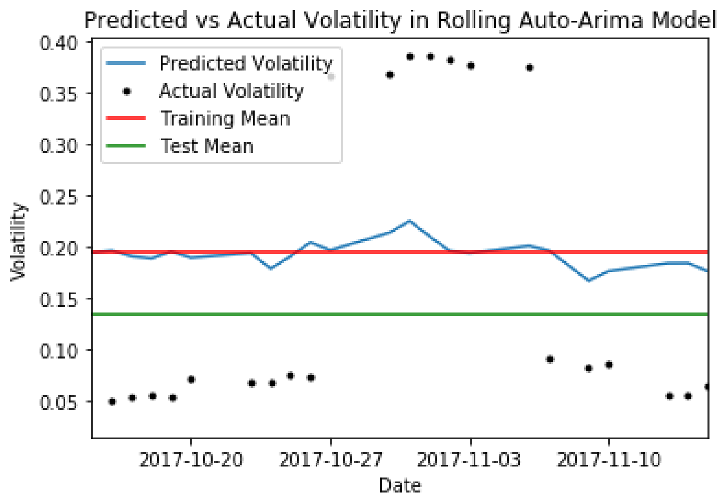


This Prophet model was actually less effective at forecasting than the rolling auto-ARIMA model. A glance at the plot above shows that this model seems to apply more weight to the trend and seasonality and less weight on the previous day’s volatility than the Auto-ARIMA model.

The amount of improvement from the first auto-ARIMA model to the rolling one was a bit startling. I began this project wondering if I would be able to predict volatility in a way that would generate a positive R-squared value at all. To see it jump from -0.6 to .71 in one iteration of the model was definitely unexpected. After spending some time pondering the reasons for the dramatic turnaround, I uncovered a flaw in the premise of the rolling auto-ARIMA. The model was attempting to predict a 10-day average of historical volatility. By fitting the model to the last day before the prediction, it was given access to 9 of the 10 days included in that average. The model was essentially able to fit itself to 90% of the test set before making a prediction. To remedy this situation I built a new version that fit a model and then predicted 10 days out each time.

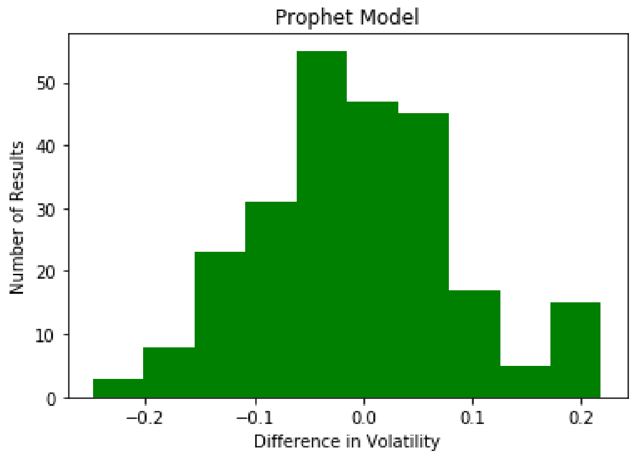


The resulting model is pictured above. This model is a large step back in terms of R-squared (-.948 the second worst model) but it actually has many positive aspects to it. First off, the jagged line shows that this model is actually taking the daily data into account and using it in the short term prediction. Secondly the overall downtrend of the prediction eventually moves the forecast below mean of the training set. In fact, in exploring the data for the last quarter of 2017, the model actually outperforms the mean of the training set and returns a positive R-squared for the period. Essentially the model is applying too much early weight to the previous period of higher volatility, but ‘learning’ to predict better as the model moves forward. There are also noticeable instances where the model makes accurate predictions about the direction of volatility, but is unable to accurately predict the scale of those moves, as in the diagram below.



## Extended Analysis

While this final auto-ARIMA model is far from perfect, I believe it represents a major step in the right direction.  Although I was unable to improve the metrics in the way I had hoped, the model does provide a positive R-squared in the final quarter of 2017.  It also shows a number of instances of accurately predicting the direction of volatility if not the scale of those moves.  The chart below shows the various models graded by how they performed in terms of RMSE and R-Squared.  It also includes the upper and lower bounds for 90% density, meaning only 5 % of predictions fell below the lower bound of the test set and 5% fell above the upper bound. Prior to that is a histogram of the residuals of the best fitting (Prophet with regressor) model.



### Model Performance

| Model | RMSE | R-Squared | Lower Bound | Upper Bound |
|---|---|---|---|---|
| | | | | |
| Auto-Arima (baseline model) | 0.103 | -1.143 | -0.16 | 0.084 |
| Auto Arima rolling (10 day window) | 0.098 | -0.948 | -0.154 | 0.087 |
| Prophet (Base) | 0.097 | -0.88 | -0.122 | 0.198 |
| Prophet with Implied Volatility Regressor | 0.091 | -0.688 | -0.147 | 0.183 |
| Prophet rolling | 0.087 | -0.501 | -0.083 | 0.191 |
| Auto Arima rolling (1 day window) | 0.038 | 0.718 | -0.053 | 0.028 |

While the above table shows the Prophet models to have significant statistical advantages, I do believe that the final auto-ARIMA model showed improved flexibility in adapting to market conditions.  I believe that both models could have useful applications in the further examination of this data set.

## Conclusions

The obvious takeaway from this exploration is that forecasting stock volatility is not a simple task.  While I was unable to obtain the degree of forecasting accuracy I had initially hoped for, many improvements were made from the initial baseline model.  Shortening the forecast period from a full year to 10 days was a major improvement in the quality of the model.  Adding regressors was also helpful and is something that could potentially be expanded on.

## Future Work

While I am uncertain as to whether 10-day historical volatility, specifically, can ever be accurately predicted, I believe these initial models to be an excellent jumping off point for attempting to predict moves in stock volatility. I believe there are a number of explorations that could be made that might significantly improve the forecasting accuracy of these initial models.

1. **Regressors**: Adding the implied volatility of the 10-day options to the prophet model provided significant improvement over the previous Prophet model. I believe that adding additional regressors could improve the predictive accuracy of these forecasts going forward. By adding additional regressors the model is directed to focus more on current market conditions than on longer term data which I suspect improves forecast quality.

2. **Shorter Term Training Set**: The final auto-ARIMA model was obviously hampered by the 4-year training set that took place in an era of higher volatility. This led to a model that performed significantly better at the end of the test period than at the beginning of it. The four-year test period was appropriate when an entire year was modeled at once. A shorter training period of say 40 days (for an 80/20 split) might be more appropriate when only predicting 10 days. The downside of that approach is that some seasonality in the model may be lost with a training period that short.

3. **Shorter Term Prediction**:- A general comment that can be made about predicting future events is that the further in the future they are, the harder they are to predict. In this project the length of the predictive period may have too large a hurdle. While 10-day historical volatility was a nice “out of the box” metric provided by Quandl, I’m not sure predicting what is essentially two full weeks of market activity is the easiest or most appropriate time period for the model. I believe that a shorter time period, perhaps 3 days, might be easier to forecast, while providing just as much, if not more, useful information from a business perspective.

## Customer Recommendations

The goal of this exploration was to provide a ten day historical volatility prediction that would be actionable in the trading arena for a client. While I believe the project fell short in that regard, the modeling did provide some useful insight into the market. I was able to identify yearly and weekly patterns as well as overall trends in volatility that could provide guidance for a client. Also, while the final auto-ARIMA model did not provide the accuracy I had hoped for, I do believe that it made useful predictions about the direction of market volatility, and that these predictions might be of use when making decisions. To be useful, the model need not necessarily predict the scale of moves. Even a model that could predict the direction of volatility at a better success rate than a coin flip would be useful, and if it were to approach 60% accuracy, it could definitely be implemented into profitable trading strategies. Conversely, a model that could predict large moves without predicting direction could also be considered a success, and would be very useful for providing 'smoother' returns, something money managers are often measured by.