# Capstone 1 Milestone

Objective

1. The initial objective of this project was to forecast prices in the biotech industry, a notoriously volatile industry. The scope of the project changed *from* prediction of price movement in the biotech industry using related industries *to* prediction of a single ticker in the biotech industry. A good lot of the impetus for this project came not only from price volatility but the very fact that the biotech industry is distinct from the healthcare industry.

   The question of price prediction in the stock market is not just an area of academic research but a constantly developing field in business and financial engineering. As it stands, it is relevant to any hedge fund that uses its computational expertise to analyze market behavior. Hedge fund entities may operate as lone stars or within financial, private or public agencies who, at the very least, may wish to entice their employees with the perks of investing their wages in stable securities. In addition, a company may wish to understand its market potential, its current profitability, and competitor price behavior. Accurate forecasting can potentially lead to gains or protect against systemic risk. Nevertheless, the first step in the process was how to choose a suitable ticker in an industry of several hundred tickers each whose price movements behave quite erratically.

Approach

2. The basic methodology of portfolio optimization appeared to be the only reasonable answer to the rather daunting question of how to select the most fit subset of tickers with some reproducible selection process. It is important to note that the use of portfolio analysis in this project was not strict. To be exact, analysis followed Data Camp's Course instructions on skew and kurtosis strictly and loosely on the steps toward calculating the Sharpe and Sortino Ratio.

   All of the python imports used in portfolio selection can be taken from DataCamp's Portfolio Analysis course. For normality testing, the analyst does not need portfolio optimization imports. She can do just fine with numpy. As a side note, DataCamp is a very useful educational platform for implementation but the analyst can achieve no more than a basic understanding of the more difficult tasks in statistics and machine learning. DataCamp by no means serves as a sufficient source of information on these matters. Mathematics is rarely explained, explained in depth or even covered. Implementation is often basic and mysterious. Finally, the analyst may find that it is the only source for learning how to implement concepts that may otherwise be too difficult to approach. For that reason, DataCamp is a great launching point and ultimately valuable.

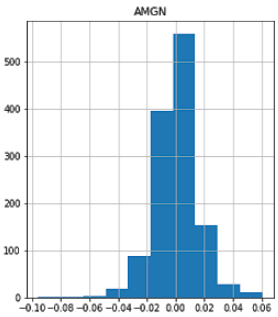
Data

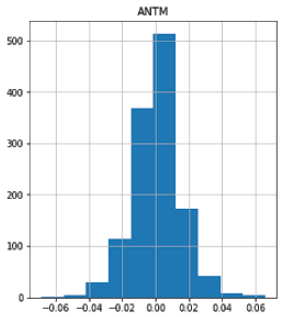
3. The dataset was taken from Yahoo Finance API. Pandas datareader method is able to read the data into a dataframe in just one line. An issue encountered was with the dataframe itself, which comes as a multi index dataframe. The columns are multi indexed, which means the dataframe has to be treated similar to a multi index dataframe indexed by row. As a beginner, multi indexing can be tricky but the trick appears to be the use of .loc and slice(None). Once the analyst becomes familiar with the multi index, she realizes lots of information can be held in a single dataframe object, a great advantage when it comes time to visualization and modeling.

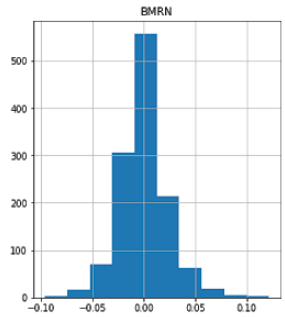
   Cleaning the yahoo dataframe was a simple task. It contains no string elements. In the past, string elements can come in a non normalized form and must be normalized. In this case, the date time indexing was correct and the only issues were that of nans and dropping levels in the column indexing.

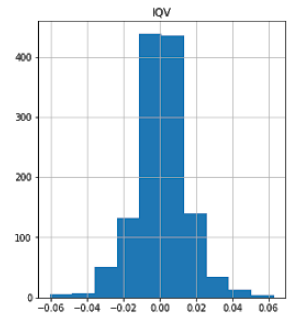
   For the issue of nans, the analyst needs to consider the problem of observations. If she decides to drop observations, does she have enough to do an analysis? Also, are there enough variables to do the analysis if she decides to drop some due to nans? For this project, nans were dealt with in a very simple manner. Many variables appeared not to have nans. Since an abundance of data was at our disposal, we only considered whether there would be enough variables to do our analysis for future projects. For the current project, we only needed one good

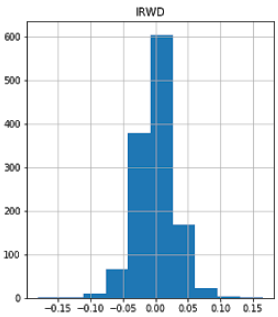
ticker. Upon inspection, nans appeared to arise from the company's lifeline. Some companies had not yet entered the market. As a result, tickers with nans in the selected time period were dropped. Following the removal of those tickers, an abundance of tickers remained and clearly needed to be reduced to a much smaller, more refined pool. *See image of refined pool below*.

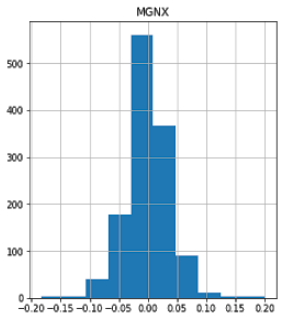

AMGN

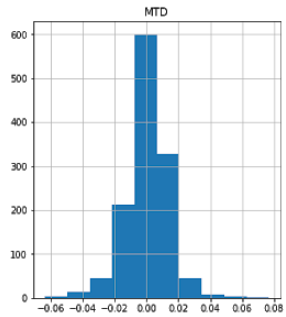

ANTM

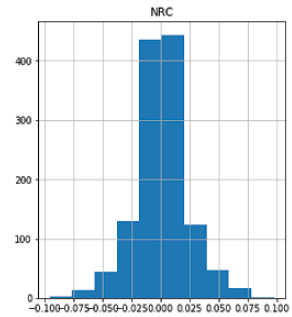

BMRN

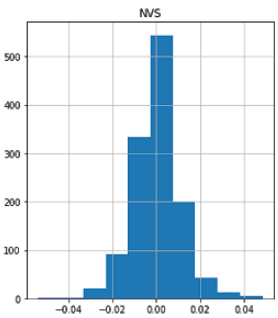

IQV

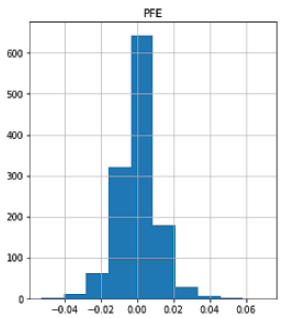

IRWD

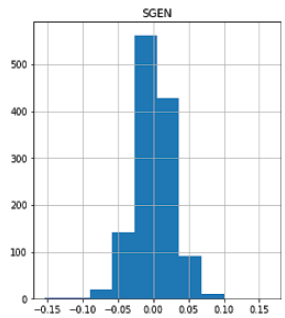

MGNX

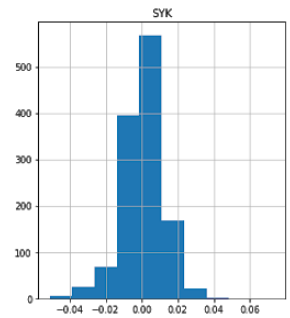

MTD

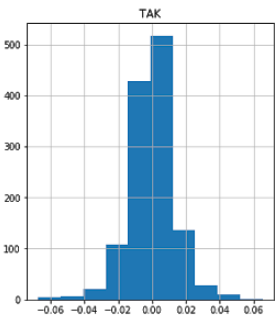

NRC

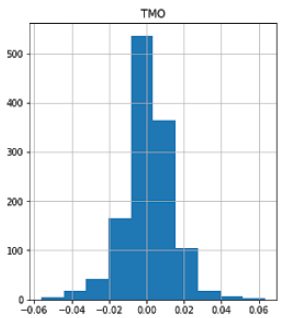

NVS

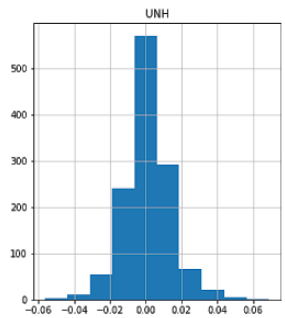

PFE

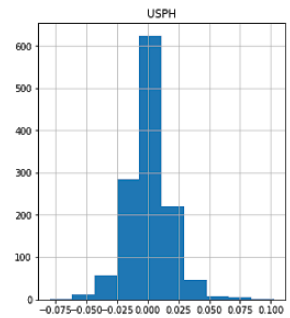

SGEN

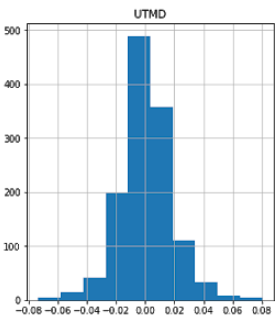

SYK


TAK


TMO


UNH


USPH


UTMD

Attempting to read the full set of 245 tickers with get_data_yahoo() caused a particular error, which was resolved by an exception method. The method overlooks the error and moves on to the next ticker in the list. The ultimate meaning of the error is still obscure because it could not have been solely based on nonexistent values in the selection period. The get_data method still produced a dataframe with ticker/columns containing nans. In other words, some tickers still did not satisfy the entire selection period.

Aside from multi index manipulations, nan and/or column removal, the dataset was easy to clean, especially since merging data sets wasn't necessary. However, if the choice to use macroeconomic variables such as federal interest rates or consumer price indices had been pursued, more cleaning would have been pertinent. Generating duplicate values on a low frequency time series would have been integrated into the process to match the high frequency of a daily time series. Generally, macroeconomic variables do not come as high frequency series. According to portfolio optimization techniques, prediction accuracy may improve when macro variables are used. The use of exogenous variables was a part of the instruction guidelines not followed. Comparison of outcomes on this matter should be verified in future projects since the process is simple to implement.

To revisit the multi index dataframe generated from get_data_yahoo(), the analyst needs to decide on which of the price variables to keep. Among them are high, low, open, close, adjusted close. The analyst can arbitrarily choose among

them as I chose adjusted close. This choice for a single price variable was driven by the intended use of autoregression models. The analyst could have just as well take the average of all price variables to get a univariate series.

The approach to generate a model based on related industries with the aid of exogenous variables is still a powerful path to pursue for discovering relationships across certain industries that would inevitably exhibit dependencies. The final approach to analyze a solitary time series was a decision based on a simplified objective to research classical statistical univariate models. In this instance, a single time series selected out of the portfolio analysis process was determined to be sufficient for objective goals. As pertaining to the strict use of portfolio optimization, the use of exogenous variables will be used when time constraints are not as limiting. Beginning with a simplified model has proven to be an informative approach. The previously anticipated approach will be the approach to compare against the current simplified model. If predictions solely based on a single adjusted price series can compare well with predictions using exogenous variables and an aggregated industry approach, the results will certainly inspire discussions regarding the ability to generalize these new methods for accurate predictions.

Statistics

4. In order to employ the use of standard autoregressive models with a firm understanding, the analyst should become familiar with parameters such as p(lags),d(difference lags),q(error lags).

   Autoregressive models use lags, which are unique types of variables. These are time shifted variables that are exact duplicates of the original time series down shifted in a dataframe.They are missing future values where the number of missing values depends upon how many periods the analyst wishes to shift the series by. *See image of lag variables(t-1, t-2) below*.

| | population | | |
|---|---|---|---|
| **lag** | **0** | **1** | **2** |
| **time** | | | |
| **1** | 5 | NaN | NaN |
| **2** | 4 | 5.0 | NaN |
| **3** | 3 | 4.0 | 5.0 |
| **4** | 2 | 3.0 | 4.0 |
| **5** | 1 | 2.0 | 3.0 |

   If the analyst wishes to create a (t-1) lag variable from an original time series with a daily frequency, he would duplicate the series and lose the last data point or the last day in the time series and get a nan where the first data point would be

in the original time series. Similarly, if he wanted to create a (t-2) lag variable, the last two data points would be lost from the original series and two nans would appear where the first two data points would be in the original series. All else, remains the same. The original series is just shifted down. This is a convenient matrix structure for the task of differencing. In time series, differencing may be necessary as this may have been the reason for creating lags in the first place. Differencing is necessary for non stationary data, especially if you wish to do modeling. Accurate modeling will be difficult without a stationary time series and that is often the major drawback of time series. Working with time series often requires methods of detrending and that is the purpose of differencing.

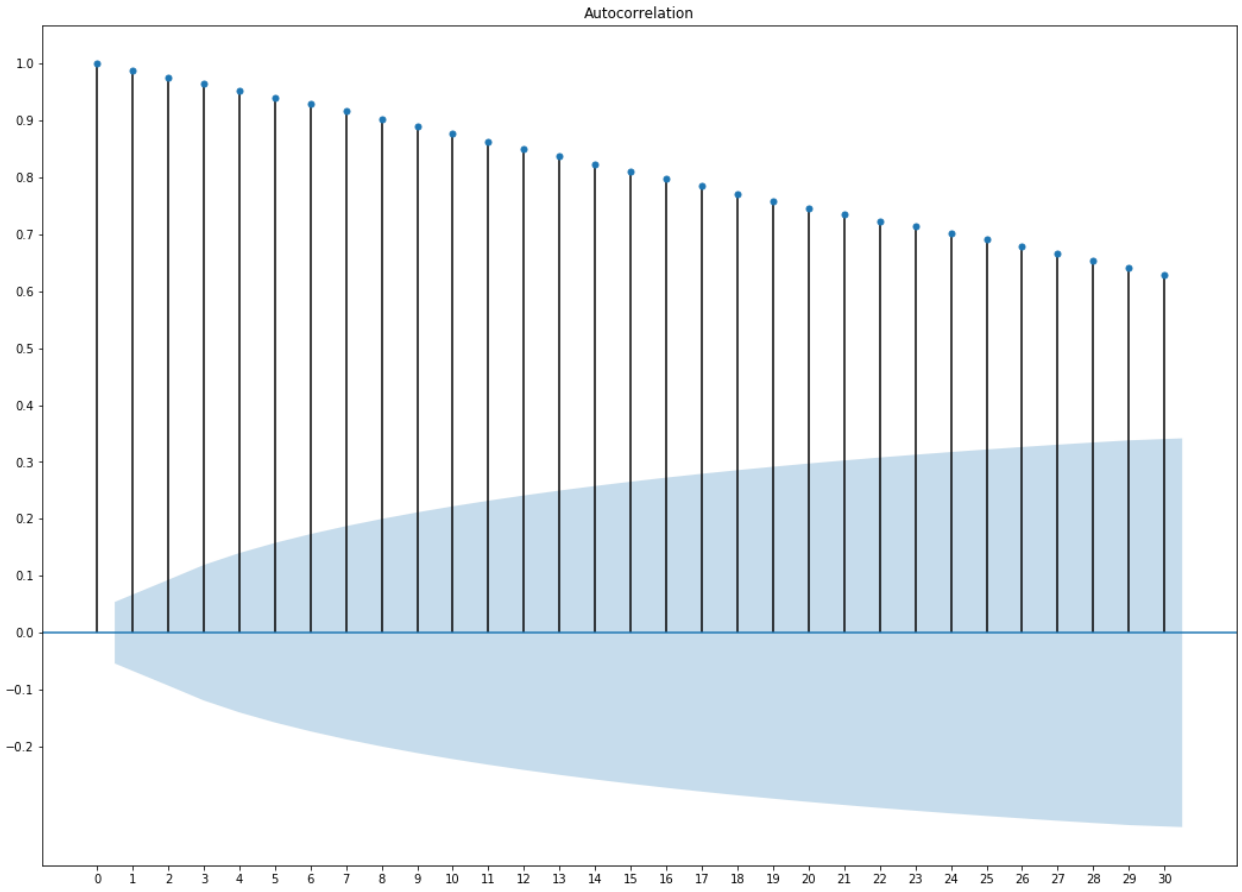
Returning to parameter p and d, p stands for the number of lags and d for the number of lags to difference. The analyst can use as many lags as he wishes and assess whether they are useful through autocorrelation and partial autocorrelation methods, which are, in many cases, already built into the autoregressive models found in packages like statsmodels. In particular, the arima function already implements these methods in order to optimize the set of parameters for prediction forecasting. The last parameter qin the arima model delivers the number of lag error terms on which to perform the moving average calculation. In most cases, the order of any of the three parameters will not exceed 3.

There is no shortage of resources for these classical type autoregressive models. The decision to implement a univariate model was due to the fact that

these autoregressive models best stand alone as univariate models but can be applied one at a time to many time series variables, which is ultimately not computationally efficient.

The equation for an arima model is linear and the weights are like the weights of a linear regression where the weight of each lag predictor contributes some amount to the response variable. Given some arbitrary value for p, the ACF will give us an idea of how many lag variables in the set of generated lags are statistically significant and if there ultimately exists a significant correlation between lags and the original time series. Positive lags mean stickiness between sequential prices. High prices will be followed by high prices. Negative lags mean swings. So high prices could be followed by low prices. In addition, the series may contain seasonality, especially if lag variables show patterns in multiples of 30 days, 60 days, or 6 months, etc. The ACF can pick up this information. What it ultimately means is that correlation is a useful property of time series. It means there is more information with which we can improve our forecast.

*See image of ACF (w/ 30 lags) below*.




Models

5. The models used in this study were Simple Average, AutoArima, TBATS, FB Prophet, and LSTM. The definitive metric for scoring accuracy was Mean Absolute Percentage Error.

   While we explored the terrain of classical stats models, my mentor proposed it would be efficient to create a general function where we could implement all of our models.

   Aspects of the modeling process became illuminated in a magnificent way. The structures within our rolling_cross_validation method was able to handle a

majority of the workload. It is debatable whether current python methods have dependable implementations for cross validation and/or splitting time series data. This cv method can break down a dataframe ticker by ticker and populate a sample column based on date time index conditions for the purpose of splitting data into training, testing and validation sets. The structure of the method was not only designed to split but implement any chosen model by training on a portion of the set and predicting one point at a time based on historical values. This process continues until all future points are predicted, one at a time. More clearly, the training data will also grow one point at a time until the categorical limit is reached and that last training point gets incorporated in the prediction method. This a great idea and will certainly be a method to further develop for time series.

The method is easy to understand and manipulate. Errors encountered with the implementation of this method were mostly generated from the model. These errors gave us the opportunity to become acquainted with the structure and the model being implemented inside the method. Some models required an additional parameter, and in all cases, modifications were simple.

Results

6. We reduced 20 years of data to 3 months to cut down on model run time. The results demonstrate the difference in model performance on a subset of the selected time period.

The order of performance from best to worst using MAPE as an error metric: LSTM (0.099), AutoArima (2.38), Simple Average(4.37), TBATS (5.97), FB Prophet (29.7).

Observations

7. The one step rolling window works well enough with autoregressive models. But the performance of AR models compares to closely to the performance of the simple average, which outperforms TBATS. As for FB Prophet's use in the rolling cv method, it performs much worse than it could without the rolling method.The effects of a one step rolling cv on AR models is a research topic to revisit.

In summary, the LSTM outperforms all AR models and should be a standard model to improve upon and use for comparison in stock price prediction.

https://github.com/ce-r/Capstone-1-Biotech-Stock-Prediction-Model