DSU Spring 2021 Stock Prediction Project

Project Lead:
Louis Zhao
Project Members:
Ivan Guan
Ananya Achan
Tony Li

Introduction:

Due to its complex nature and over reliance on news driven changes, the US stock market has always had a notorious reputation for being extremely difficult to forecast. Movement in the stock market has often been described as Brownian motion, and while this is almost certainly not the case, the large amount of noise definitely makes creating an accurate model quite hard. While neural networks and quant based approaches have been proven to work to a decent degree in the past, for this project, we wanted to employ more traditional and explainable models as well as incorporate models specifically designed for time series analysis. Through this project, we used three different approaches (regression, classification, and time series modeling) to forecast changes in the stock market and report on our findings and results in this following write up.

Data Collection:

For this project, we aimed to build our ML models using two specific sources of data: the daily movements of the US stock market as well as financial ratios which are essential for conducting analysis on public companies. Because of the lack of computing power and in the interest of time, we decided to focus on only a subset of the US stock market and to maintain consistency and generalizability, we chose the S&P 500. For those unfamiliar with the US Stock Market, the S&P 500 is a collection of the 500 largest companies on the US stock market and because of its diverse nature, is a great center measure of the overall performance of the US stock market as a whole. Because of the time series nature of this project, there can be a lot of noise and variability coming from external circumstances such as current events and market trends. To try and eliminate as much of this noise as possible, we decided to only focus on the period from late 2008 to early 2020. During this time period, the US stock market was fairly bullish and there weren't that many extreme shocks shaking up the market such as the crash of 2008 or COVID-19 allowing for less variability and as you will see later, more accurate models. More specific information on how we collected the data can be found below:

A. **Stock Data**

Our daily stock data was collected with the help of a third party open source package in Python, yfinance. Yfinance is an API wrapper that gathers real time and historical data from yahoo finance. We used this to gather all historical stock data from 2000 to 2020 and later decided to cut it down to just 2008 to 2020. Because stock data updates real time during market hours, and we only gathered data at the daily level, we needed some way of measuring the average stock price during a given day. We decided to use two forms of measures: the closing price and the OHLC average to train our models. The closing price gives a good summary of how a stock performed on a given day while the OHLC average is a better value to calculate the moving average of a stock across a longer period of time.

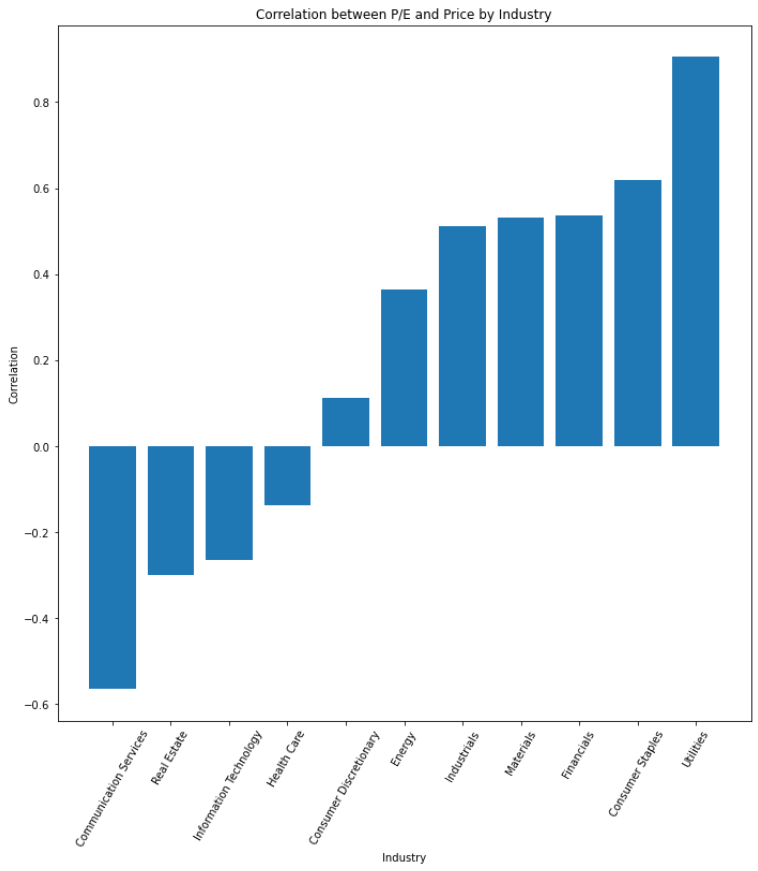
B. **Financial Data**

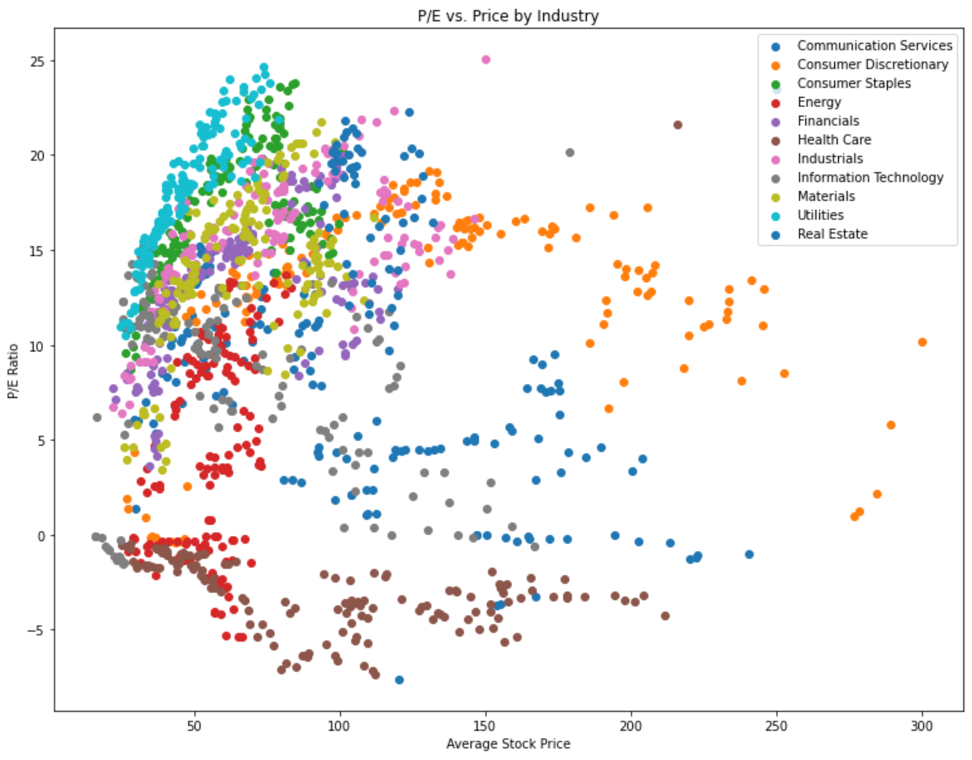
Every fiscal quarter, companies release their statement of earnings along with a myriad of other financial fundamentals that point to the health of their company for that quarter. While these fundamentals are a great indicator for how well a company is doing, the combination of these variables actually provide much more relevant information. This process is sort of like feature engineering in Machine Learning whereby crossing different features can help encode nonlinearity and improve accuracy by contextualizing certain features with others. For example, the P/E or Price to Earnings Ratio is one such crossing that works better than its individual parts (price and earnings). This is because by dividing a company's price with its quarterly profits, we are scaling their profits with the actual size of the company to help identify if the company is overvalued or undervalued. Because this data is fairly difficult to gather from open-access resources, we used WRDS (Wharton Research Data Services) to provide us with this information. Their databases had a total of 69 financial ratios which we incorporated into our analysis and models.

Exploratory Data Analysis:

A. Pearson Correlation between P/E Ratio and Price

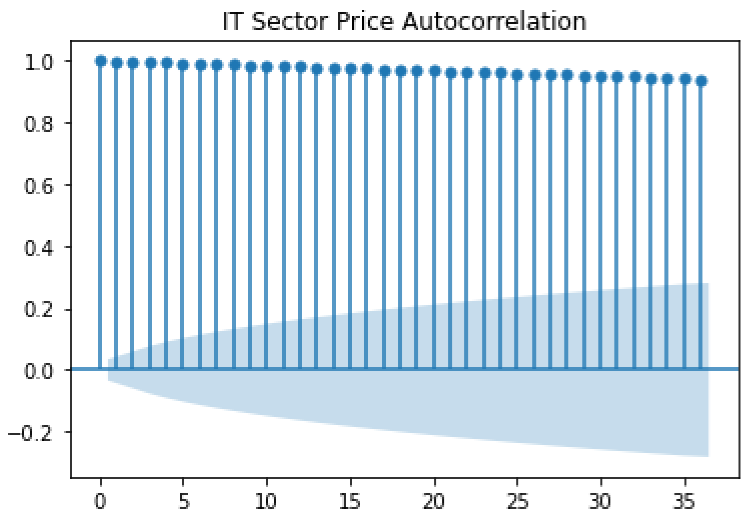
From our preliminary research, we knew that the P/E Ratio was one of the most important financial statistics when it came to the valuation and forecasting of a stock. Thus, we made it the focus of the EDA portion of our project. First, we separated our companies based on the specific industry they were in and examined the individual correlation between each industry's average P/E ratio and their average price across time. As we can see from the bar graph below, only a few specific industries have average stock price correlated with their Price to Earnings Ratios. Some industries have very low correlation and some have even negative correlation which was very surprising for us. A potential explanation for this phenomenon could be that some industries are inherently more speculative than others. For example, in the healthcare field, most companies don't actually have marketable products since it takes many years to actually develop a FDA approved drug. Thus their profits might be extremely low but their stock price could still be high due to their future prospects. This shows us that while P/E Ratio could be a good metric of evaluating some companies, it is definitely not the be all and end all when it comes to forecasting a company's financial wellbeing. A scatter plot can be found below which shows a more in depth look into the actual relationship between these two variables across different industries.

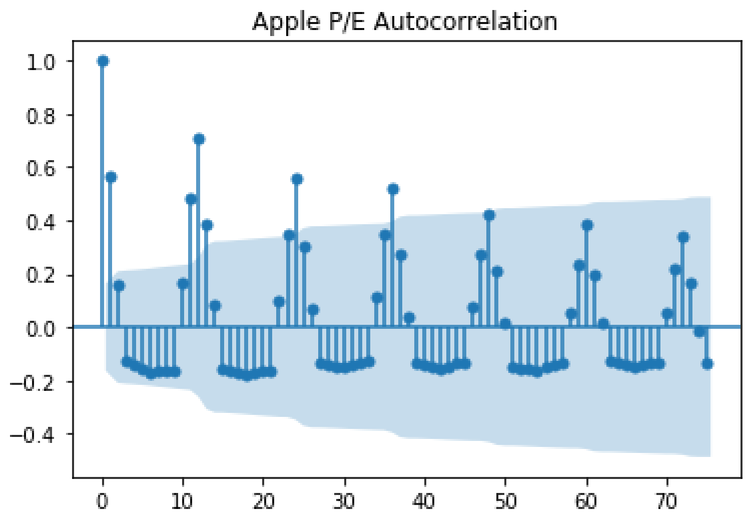



P/E vs. Price by Industry
Communication Services
Consumer Discretionary
Consumer Staples
Energy
Financials
Health Care
Industrials
Information Technology
Materials
Utilities
Real Estate
P/E Ratio
Average Stock Price
25
20
15
10
5
0
−5
50
100
150
200
250
300

B. Autocorrelation of P/E Ratio and Price:

The Autocorrelation of a time series variable is the mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive intervals. It is essentially the correlation of a variable with itself sometime in the past. The autocorrelation for a variable across time is an important tool when it comes to financial analysis as it can show seasonality and trend. Below are the autocorrelation plots of our P/E ratio and Price for the IT sector and Apple, the largest company in this sector. From the figure on the right, we can see that there is a strong linear trend across time for the price of a given stock. This is because we focused our analysis on a steady bull market which saw consistent growth in the market. Thus, it makes sense that different intervals of time would be positively correlated with each other. However, if we instead focus our autocorrelation plot on the P/E ratio instead, the trend is a lot less uniform. In fact, the trend is seasonal, which is probably due to the cyclical spending habits of consumers. For example, spending increases a lot during fall because of the Holidays, causing the earnings of companies to drastically increase during this period of time every year. This autocorrelation plot shows us the existence and extent of this seasonality.





Machine Learning Models:

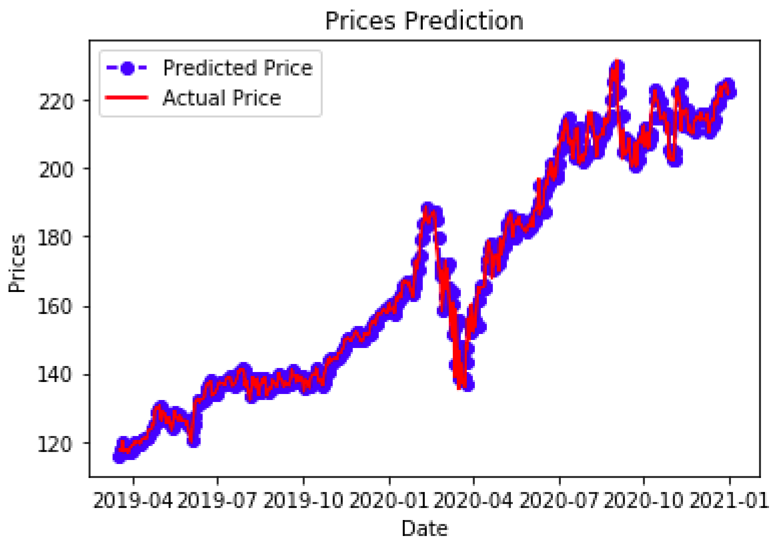
A. Time Series (Ananya)

A time series is a sequence where a metric is recorded over regular intervals. In our case, this metric is the stock price of a company. Time series forecasting is predicting the next values in this sequence based on past values. For creating a model for time series forecasting, I used the AutoRegressive Integrated Moving Average (ARIMA) model.We'll only be using the Close price to predict the future values (univariate time series). In order to fit an ARIMA model, we need a stationary time series. Stationary means that the series is mean reverting. The model is autoregressive so it is a linear regression that uses its own lags as predictors. Linear regression works best when features are not dependent on each other. So it's important for predictors to not be correlated, which is why our series needs to be close to stationary. But price series are usually non stationary, so we must do differencing.

We need three parameters for the ARIMA model:
p - the order of the autoregressive term
q - the order of the moving average term
d - order of differencing in order to make the time series stationary

We can use an example of microsoft stock prices to see how the ARIMA model performs. We need to subtract the previous price from the current price (this means differencing once). So we can keep doing this multiple times, this is the multiple orders of differencing i.e. the d term in our model. If time series is stationary then d = 0. We need to find the order of differencing needed to make microsoft stock prices time series stationary. We can create autocorrelation plots to see when the series becomes over-differenced i.e. the lag goes into the far negative (by plotting for different orders of differencing), and then we can choose the d value accordingly. Instead of doing this manually, we could find d by using the pmdarima package. To find d, we can use the function ndiffs, in the package. So we don't have to manually look at plots to decide the value for d in our model. Next, p is the number of lags to be used as predictors. We can find this by looking at partial autocorrelation plots (i.e. the correlation between the series and the lags). The q term is the order of the moving average term i.e. the number of lagged forecast errors for the model.

For predicting microsoft stocks, we split the data for closing values using the S&P 500 dataset from 2015 to 2020 into 70% train and 30% test data, and we find that we should set p = 4, d=1, q = 0. Since we want them to be in a series, we don't use the train_test_split, but we manually split the data so that they remain in order. After training the model, we can test the model using the testing data. We got that our testing MSE is 13.14. In my opinion, this seems lower than expected, and ARIMA seems to work well!



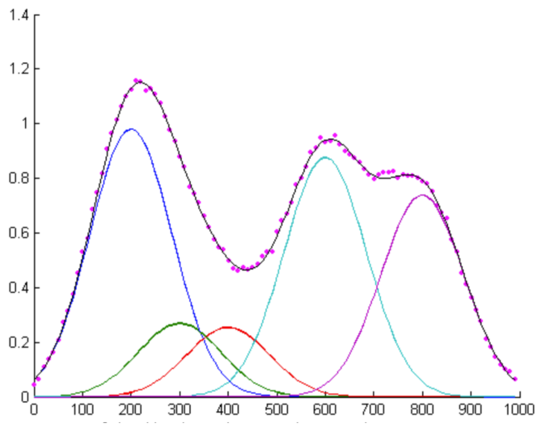
B. Support Vector Regression (Ivan)

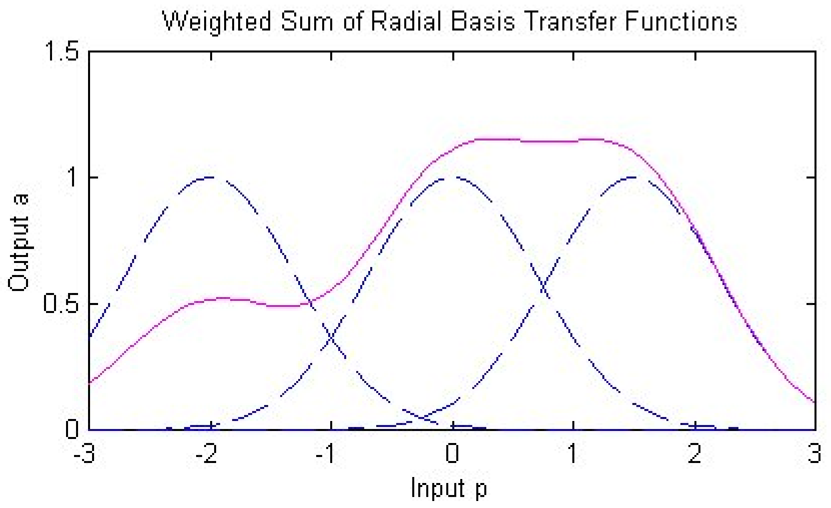
a. For modeling stock data using support vector regression, I relied on a radial basis function over linear and quadratic functions. Radial basis function is typically used for modeling non-linear trends which most stocks pertain to during a straight bull market. Stocks, on a daily basis, clearly do not fit a linear model or a quadratic model.

b. Radial Basis Function

   i. A radial basis function is a simple neural network. It has three parts- an input, a translation based on a gaussian function, and an output for predictive purposes. The input that we feed into the network is the historical stock price. The translation simply computes the euclidean distance of the input from a center and spread of the gaussian function in that specific neuron. The center and spread are determined by training the model (by feeding in input). A weight associated with each neuron is then multiplied to that distance which forms the output. The sum of these outputs begin to form a smooth curve that similarly resembles a gaussian bell curve. More simply, we define this as the linear combination of gaussian curves (linear algebra).

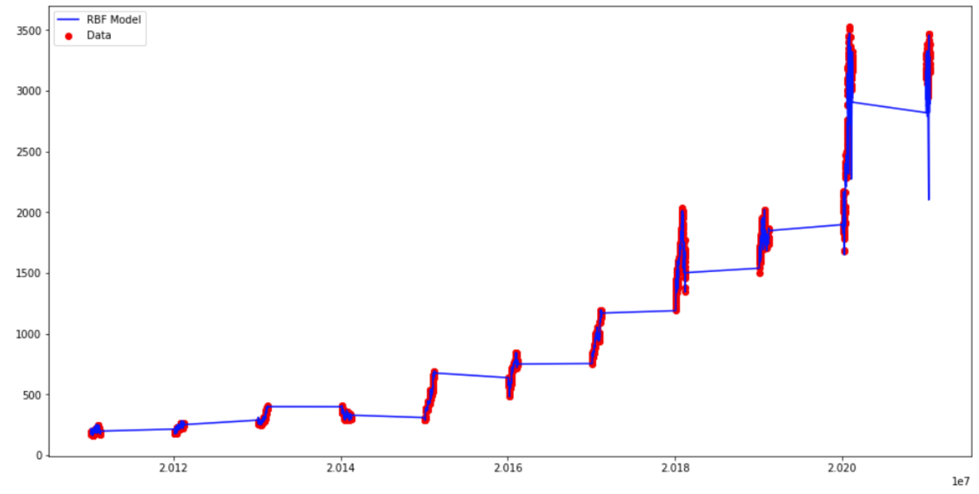
   ii. Predictably, the radial basis function cannot adjust to rapid drops or quick changes which will affect stock predictions from 2020.



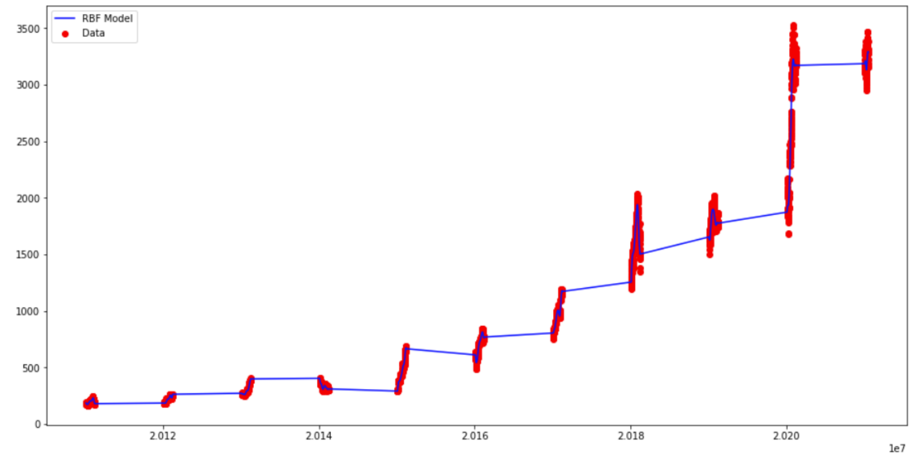


c. Results- Analyzing Amazon
   i. We fed in data in three different time frames with differing gamma values. The gamma value, in simple terms, states how much a single stock price can influence the regression curve. We trained the model up to 5/17, which we then used to predict the 5/17 price. We compared the predicted price to the actual closing price and calculated a percent error.
   ii. Results
      1. 5/2/2020- 5/17/2021- 1.09% error (gamma 0.1)
      2. 1/03/2011 - 5/17/2021 - 0.3606% error (gamma 1E-5)
      3. 1/03/2011- 5/17/2021 - 59.7% error (gamma 0.1)
   iii. Conclusions
      1. For a bull market run, we can see that from 5/2/2020 - 5/17/2021 has acceptable accuracy with a normal gamma value.
      2. However, as stated above, the model cannot handle rapid drops that come with the stock market. If we include the drops in 2020 in our model, with a normal gamma value, the model simply fails with a near 60% failure.

3. From a similar time frame that included 2020, if we minimized gamma value to an extreme range very near zero, the model only has a 0.36% error. However, this is to be taken with a "grain of salt." The model, with that extreme of a gamma value, will overfit a regression line to that data which makes predictions rather one-dimensional.



4. 1/03/2011- 5/17/2021 - 59.7% error (gamma 0.1)



5. 1/03/2011 - 5/17/2021 - 0.3606% error (gamma 1E-5)

iv. Broader Market Analysis
   1. Same Procedure as above, trained up to 5/17/2021 and compared predicted price against actual closing price
   2. Focused on three sectors
      a. Consumer Discretionary
      b. Information Technology
      c. Financials
   3. Results

a. 5/02/2021 - Now (Gamma 0.1)
   i. Consumer Discretionary - 7.45%
   ii. Information Technology - 4.18%
   iii. Financials - 14.84%

b. 2012 - Now (Gamma 0.1)
   i. Information Technology - 28.26%
   ii. Consumer Discretionary - 34.95%

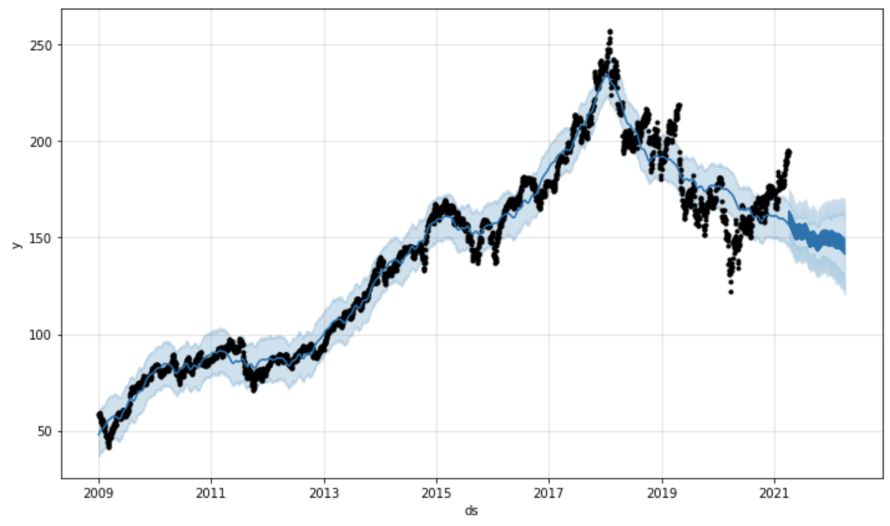
c. 2012- Now (Gamma 1E-5)
   i. Information Technology- 0.31%
   ii. Consumer Discretionary - 1.74%

4. Higher percentage error can be attributed to random events from a large sample size of companies

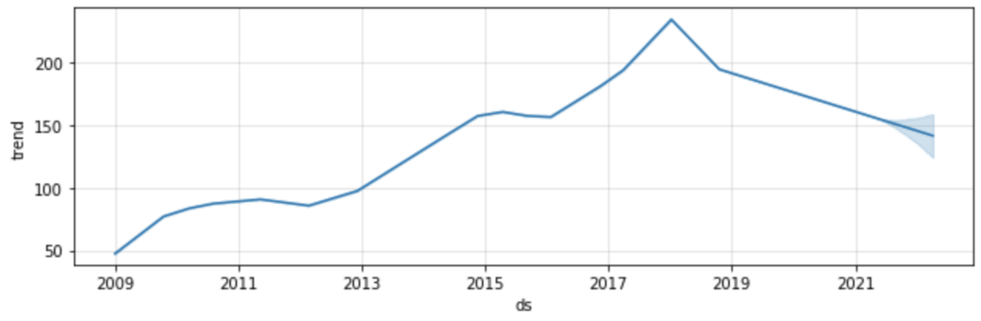
v. Overall Conclusions
   1. Regression models for stocks is very one dimensional
      a. Does not account for random events (stock splits) or predictable events (holidays)
   2. Gamma 1E-5, while it looks accurate, is very much a bandaid.
   3. The rbf model works well for companies that are stable and that do not sway quickly due to market bearings or news.
   4. As stated above, rapid drops (such as 2020) simply skews the data heavily, affecting predictions if a typical gamma value is used.

C. Facebook Prophet (Time series forecasting)

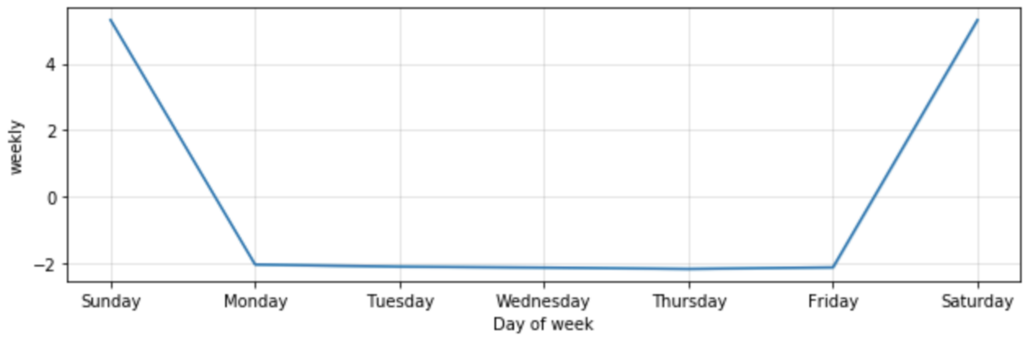
The data that we used for this model includes the daily price movements in the S&P 500 from 2009 to 2021. By feeding in historical data from this source, we created dataframes that include our values of interest. We were then able to extrapolate prices for 365 days after the last historical data entry that was collected, so our forecasted dataframe extends well into 2022. As a side note, Prophet was built around a generalized additive model (GAM), which is more powerful than a traditional linear model as it uses custom functions. We can subset the "forecast" matrix (dataframe) to extract the variables we want to consider (since the comprehensive dataframe contains more information than we wish to analyze) and then populate the dates in the dataframe with numeric values.

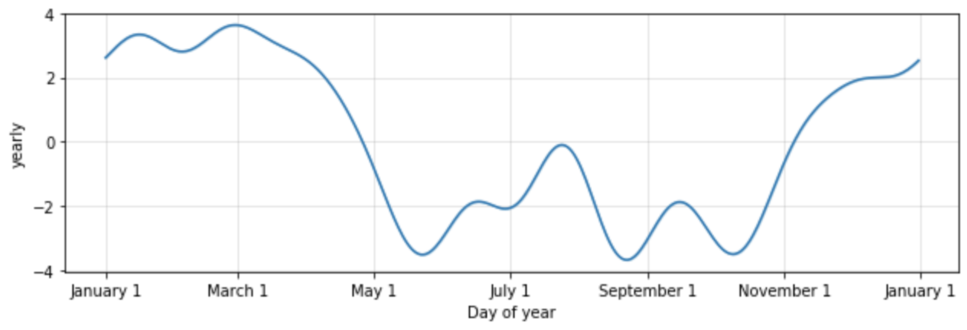


We can break down raw data into the subcomponents of time series (trend and seasonal factors).



We also consider seasonal factors, which we segment by week and by year.





We see that prices are higher at the beginning and at the end of a given year, thus indicating a bullish market during the holidays. This is most likely due to increased spending during these months.

D. Classification (Louis)

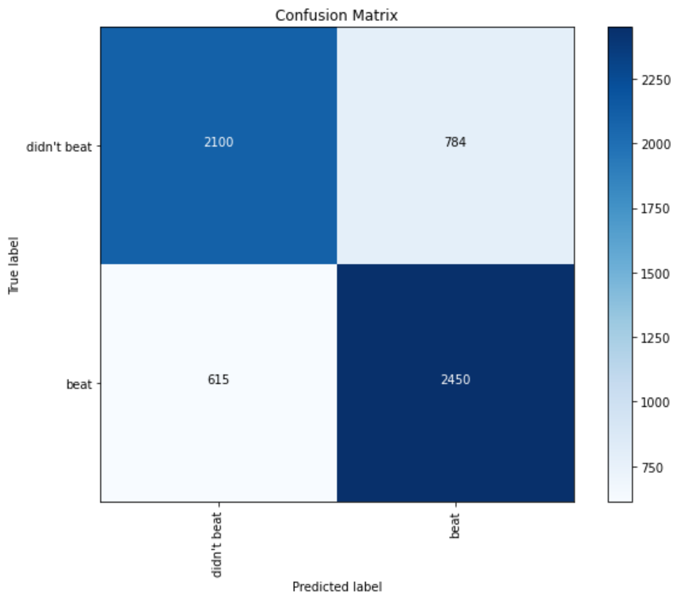
One major issue we discovered for our prediction problem is taking into account how the state of the overall market might affect the changes in price of a certain stock. Each company in the stock market does not exist in a vacuum and is constantly being affected by the conditions of its competitors as well as the general market. For example, if there is a massive sell off in one sector of the S&P 500 on a given day, then there is a very high chance other sectors might also experience a sell off. To take this into account, we wanted to create a classification problem focused on predicting whether a given stock might do better than the overall market for a given quarter. To accomplish this, we feature engineered a new binary variable which is labeled True if a given stock had a larger increase than the S&P 500 Index for a given quarter and False if it did not. We then took an iterative approach towards training and fine tuning a Random Forest Classifier Model in Python through sklearn. The iterative had three overall steps in which we engineered more and more complex variables with each step:

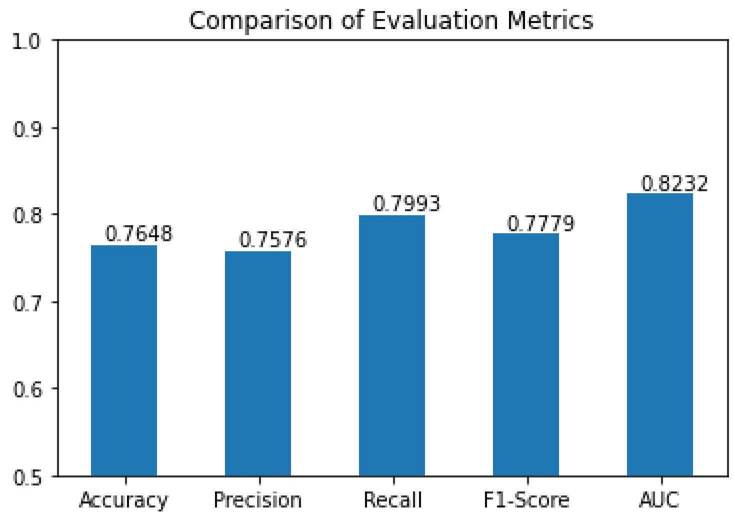
1. First, we only trained the model on our dataset of quarterly financial ratios for the S&P 500 companies. This gave an accuracy measure of only **58.5%** which means that the model was only doing a little bit better than random. This showed that just the financial ratios themselves didn't really mean anything when it came to predictive power, probably due to the drastic differences that exist between large and small companies as well as companies in different industries. This model also didn't account for the time series component of our dataset since each observation had no indicator of time.
2. Next, we wanted to implement a way for the model to recognize previous relevant trends. To accomplish this, we added two new binary variables for each financial ratio, QTQ and YTY. These variables would be labeled True if the corresponding financial ratio did better than the previous quarter and the previous year respectively. We also added

another binary variable which kept track of whether or not the observation beat the market the quarter before. After incorporating these new variables, our model's accuracy increased to **72.4%**, which was quite a remarkable jump from before.

3. Lastly, we wanted to add more in depth categorizations for the companies themselves. To do this, we created two new variables. The first one was a label encoder for each specific company that ordered them based on their total market cap. Number 1 would have the largest market cap while number 484 (the last company) would have the smallest market cap. The second categorization was a historical tracker that kept track of how often the company beats the market. Companies that beat the market more than average were labeled with a 1 while companies that didn't were labeled with a zero.
   With the inclusion of these two new variables, our model's accuracy improved to **76.5%.**

After evaluating our model using a stratified 10-fold cross validation, we generated the following visuals to gauge the predictive power of our model across different metrics.





From the bar graph above, we saw very consistent values across the different metrics tested. Because we are creating a model to generate investment returns, false positives means creating losses while false negatives means losing potential gains. Thus, a more wary investor should take into account the model's lower precision before using it to make decisions. This evidence is supported by our confusion matrix.

The three most important features in our model were: QTQ Price/Sales, QTQ Price/Book Value and QTQ Cyclically adjusted P/E Ratio. This shows that quarter to quarter trends are by far the most important predictor for the success of a stock, which is not surprising considering the time series nature of our data. However, what is surprising is the fact that YTY values did not have a large impact on our model predictions. Perhaps a reason for this is because we were in a bull market, most companies usually did better than the previous year. Maybe if we used a percentage change indicator instead of a binary indicator, we can capture this nuance and improve our model. This will be left to anyone interested in building off our model for future work.

Conclusion:

The four different models all gave us crucial insight into how the US stock market operates and the complex relationships that drive its constant movement. We discovered that using a regression based approach should be only used as a reference point. It is not completely accurate as it does not account for withstanding news or rapid changes to the market. Time series models aim to fix some of these shortcomings by using more rigorous and proven statistical methods that can not only accurately track a stock's movement but also make confident predictions about its future. However, these models are very susceptible to drastic movements in the overall market that are harder to foresee. Thus the classification model aims to fix this problem by using company fundamentals to make more generalized predictions that are more resistant to sudden and unexpected changes.