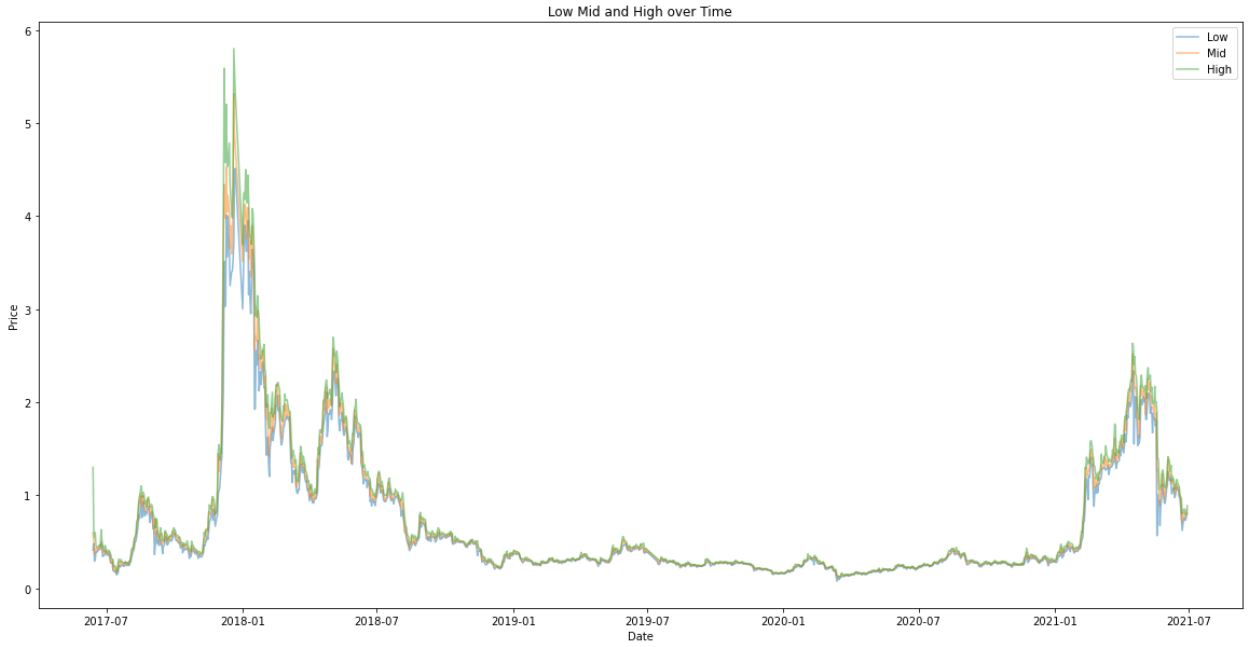
The Problem:

While not a new problem to the world of data science, I decided to try my hand at predicting future stock market prices in such a way that someone could confidently invest in a given stock with little to no risk of loss.

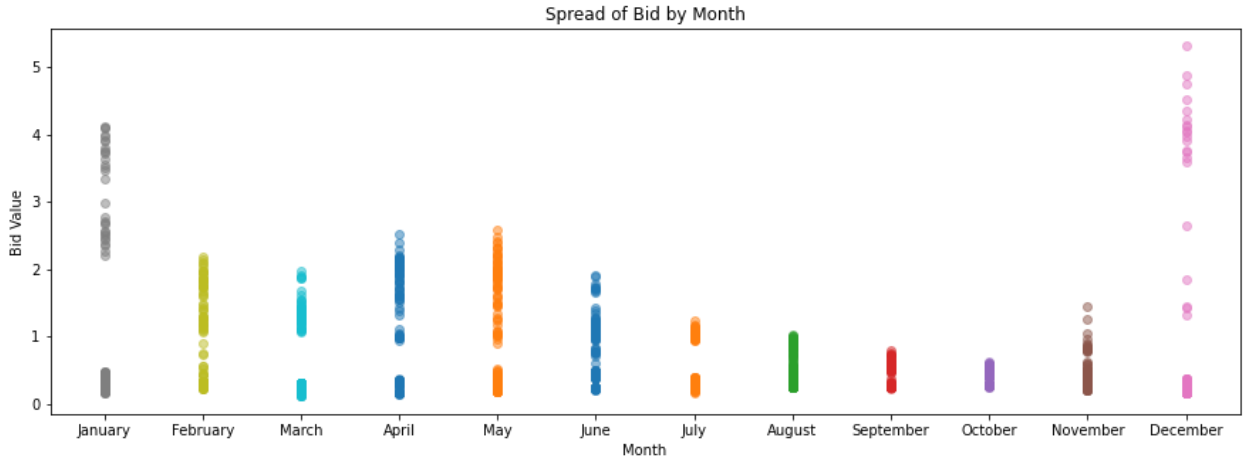
The Approach:

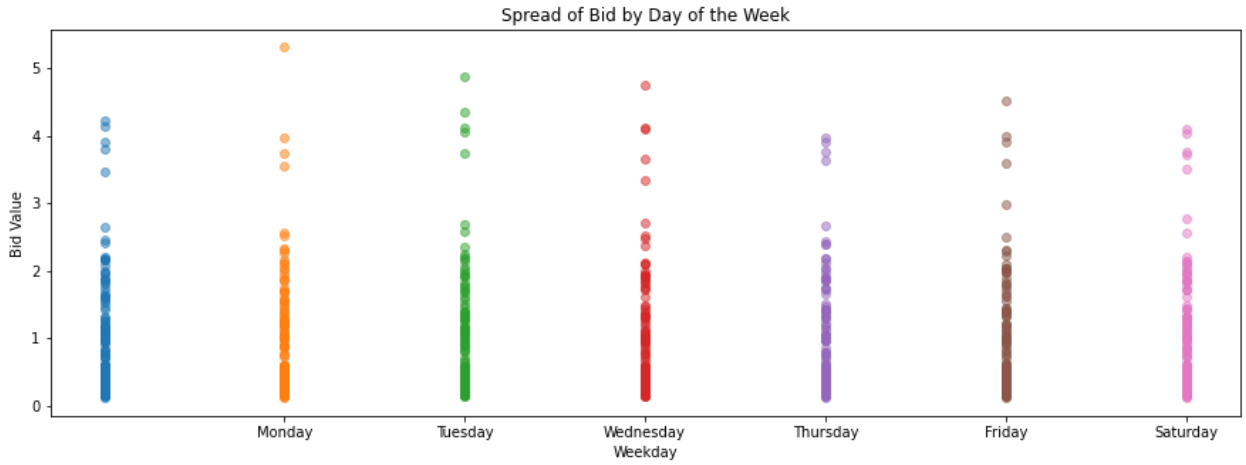
I started with a dataset of up-to-date stock market data from Quandl, specifically from the company iFinex, who uses different forms of virtual currency such as BitCoin. Looking at the data, I had access to each day's date, local high, mid, and low prices, as well as the standard bid and ask prices and volume. Since the data was non-continuous, I decided to predict the bid and ask prices, which are essentially the values deemed as safe buy and sell prices respectively, and to make sure that my predictions did not exceed the daily high and low. Below is a graph of the main prices over time. One thing that will make these predictions tricky is the level of precision needed, as the range of the prices is roughly from 0-6.



Exploring the Data:

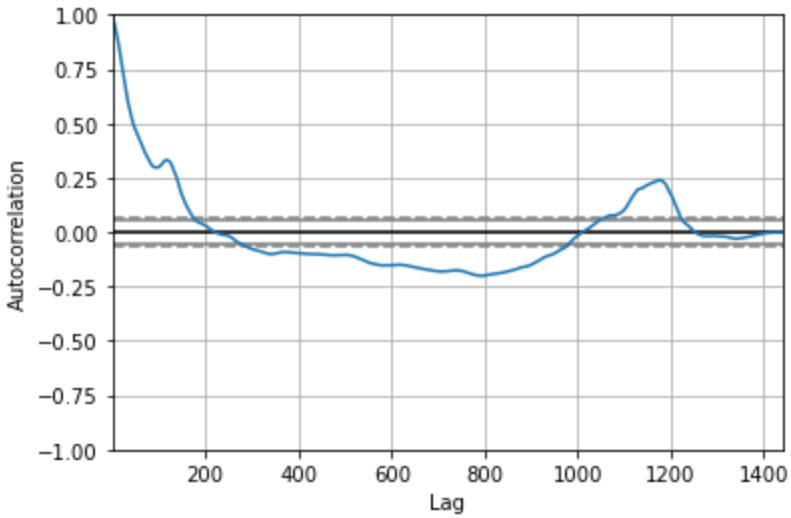
The first thing I checked was for null values, as those can throw off any machine learning algorithms. Having found none, I then added columns that elaborated on the date by including the year, month, day of the month, and day of the week. These helped me to check for seasonal patterns. For example, below are graphs for the spread of the Bid column by month and by day of the week.



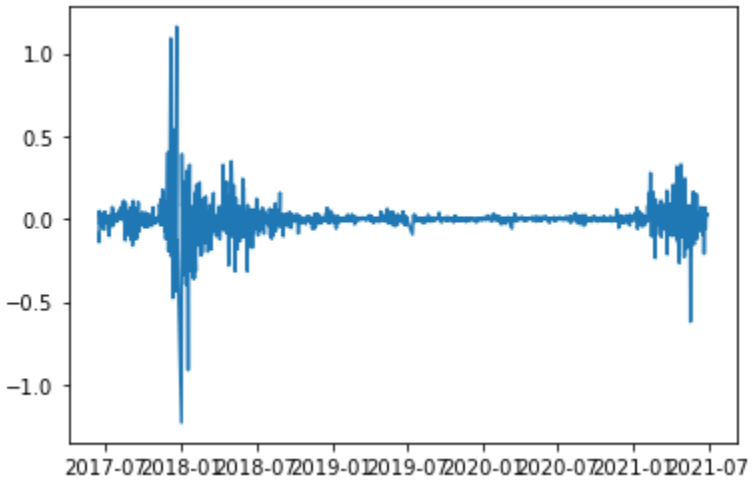


Looking at the data like this, it's clear to see that early winter and late spring generally have the highest Bid values, with fall having the lowest Bid rates. On a weekly basis, the spread is more even but is slightly higher at the beginning of the week and slightly lower on weekends.

The next thing I looked for is autocorrelation, or how strongly the data is correlated to itself, as that can make predicting easier for models like ARIMA. Looking at the autocorrelation plot below though, the data doesn't seem to have a strong correlation (measured by how much of the data stays between the dashed lines). As a result, a model that takes the other columns into account might be better suited for this task.



The last thing I want to take note of is whether or not the data is stationary, as that will have an impact on the mean of the data, as well as the linearity. While the original data itself is not very stationary, the autocorrelation of the data is.



Preprocessing:

Since I have non-continuous data to work with, I'm going to need to run predictions based off of the previous day's values. In order to do this, I added columns that indicated the previous day's High, Low, Mid, Bid, Ask, and Volume. This process added null values to the first and added a last entry with nulls, since the first day does not have previous values to copy and the last doesn't have it's own day's values. I dropped the first entry, and saved the last as the day I want to run predictions on. Another preprocessing step I did was to create dummy values for the Weekday column. While this column was originally made of integer values when it was created, what day of the week it is falls under categorical data rather than continuous. The final step in preprocessing my data was to split the training and testing data. Since this is time series data, I ran a TimeSeriesSplit, rather than the usual randomized TrainTestSplit, as chronological order matters. In the X data I used the previous day's price values and Volume as well as the

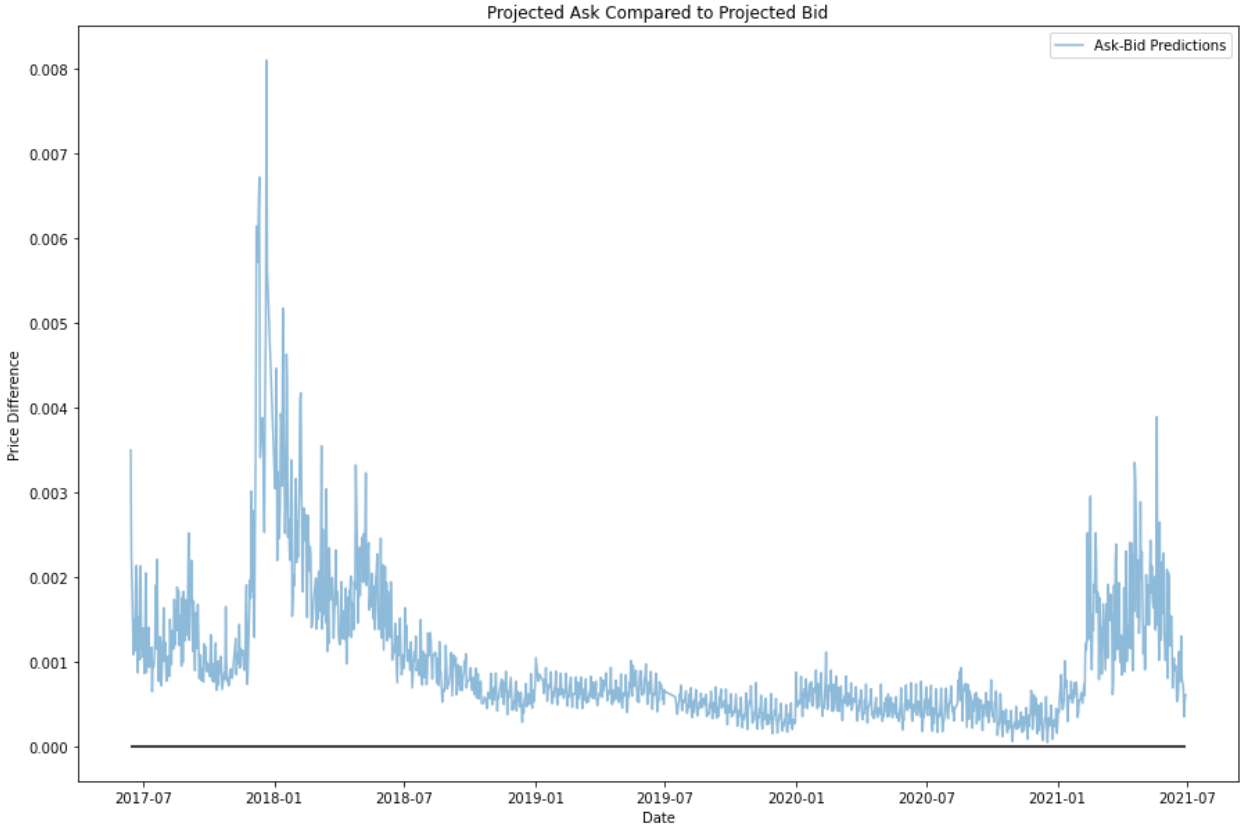
time data such as Year, Month, Day, and Weekday, and for y I used that day's Bid and Ask, as well as the High, Mid, and Low for comparisons in the modeling stage.

Modeling:

Since I'm working with time series data, the two models I decided to go with were a Linear Regression model and an ARIMA model. The former's strength lies in being able to predict off of the provided X columns without need for hyperparameter tuning, while the ARIMA model only requires the Y column to predict itself, but requires a lot of tuning. Since I am using two prediction variables, I ended up making two of each model.

Findings:

The metric I used for both of these model's was a mean squared error formula to gauge how close each was to correctly predicting that day's Bid and Ask values. Overall both models did a fairly good job. The ARIMA model was able to get a MSE score of (0.00951, 0.00953) for Bid and Ask respectively, while the Linear Regression model did slightly better at (0.00863, 0.00864). While ARIMA did have the advantage of being able to be tuned for better predictive power, ultimately it lost out due to the lack of a strong autocorrelation. The Linear Regression model was successfully able to find patterns in the date and previous values to predict accurately. Below is a graph of the difference between the predicted Bid and Ask values from the linear model, which essentially show the profit per stock if someone were to buy at the Bid then sell at the Ask, assuming the Bid precedes the Ask. Since this is a measure of potential profit, it is reassuring that it never dips into the negatives, although on a stock whose price values vary on a small scale, the potential profits are not very high.



Ideas for the Future:

The linear model might be able to be better optimized if I analyze the coefficients of each of the columns to see how relevant they are to predictive power. The ARIMA model could be optimized with a wider range of tested hyperparameter values, although the time it would take to optimize it further might not be worth the improvement if it still underperforms compared to the other model. Lastly, creating a pipeline to automate these steps will make it easier for a client to use in the future.

Client Recommendations:

While this model is able to fairly accurately predict a day's Bid and Ask values, it is not yet automated to tell you when to buy and when to sell. As a result, you will need to wait to buy until the current price is below the Bid, and wait to sell until the price is above the Ask. This model is also tuned to one specific stock, so investors of other stocks will need to have the models retrained on the stocks they wish to invest in.