Using Deep Learning to Find Stock Pricing Patterns in Exchange Traded Funds

Christopher Pranger

Anderson College of Business and Computing, Regis University

MSDS 696: Data Science Practicum II

John Koenig

December 12, 2021

Using Deep Learning to Find Stock Pricing Patterns in Exchange Traded Funds

The allure of potential riches has drawn investors into the world of stock trading for decades. The perceptions of high finance, the intricate puzzle of the world economy, the introduction to new technological and social trends, and the simple idea of having one's money 'work for you' all contribute to the massive draw of the stock market. Innovations in computing technology and data science methods have aided in the growth of the stock market and have even changed the entire ecosystem.

One change is the prevalence of algorithmic trading, or the buying and selling of stocks by computer programs. This strategy is used by major institutional investors such as banks and hedge funds all around the world. The spread of technology – as well as easier access to the markets via e-brokers – has set the stage for algorithmic trading to become even more prevalent. Some estimates show that algorithmic trading makes up 70% of the entire volume of the stock markets [(1)].

A second change is the growing number of Exchange Traded Funds (ETFs) available on the stock market. ETFs were introduced as a way to invest in index funds. An ETF is a stock that represents a share of a broader collection of stocks or other investments. For example, one can invest in an ETF that represents the S&P 500 index. The price of the ETF's stock goes up and down with the value of the index. Depending on the year, dozens or even hundreds of ETFs enter the market annually. ETFs offer diversification in the form of small-scale investments and represent everything from broad market indexes (like the S&P 500) to niche sectors and assets (real estate, semiconductors, legal marijuana, etc.) [(2)]. ETFs are used by individual investors as well as large institutions, who most likely trade ETFs algorithmically.

Data science innovations, especially in deep learning, seem ripe for application in the stock market. Time series analysis using Recurrent Neural Networks (RNNs) is one method that seems particularly appropriate for stock price time-series analyses and has experienced growing momentum recently.

Research Question

The rise of algorithmic trading, increasing prevalence of ETFs, and maturing methods in the field of deep learning all coalesce into one intriguing research domain. The purpose of my project is to investigate the following research question:

> In comparison to a passive investment strategy, can RNNs be used on the pricing data of a given set of high-volume ETFs to find profitable trading patterns within different market time windows?

Given the high-stakes nature of the stock market, any patterns found could be used to form the basis of lucrative trading strategies. Also, my question adds to the current domain in specific ways. First, focusing solely on ETFs is unique. Second, looking at pricing patterns of different intra-day time periods contrasts the more common approach of analyzing stock price changes over days or weeks. And finally, I structure the machine learning problem as a binary

classification one – making predictions whether the stock price goes up or down during a given time window. Many stock price analyses are regression problems, attempting to predict the stock price, not the movement specifically.

Data Overview

The stock price data for this project comes from financial data provider Alpha Vantage. Alpha Vantage provides many types of financial data, but for this project I use its time-series stock price data. For each stock, the pricing data contains a row for each minute, and that row contains the opening, closing, high, and low price for that minute. It also contains the volume, or number of shares traded during the minute. Alpha Vantage supplies data via API call and provides the most recent two years of historical pricing data, minute by minute. I downloaded pricing data for 54 ETFs (more on the ETF selection below).

Project Methodology

Data Collection

First, I defined a list of ETFs for which I would download the data. After a simple preliminary analysis of which ETFs were currently traded as well as traded two years ago, I selected 54 stocks from a list of high-volume ETFs [(3)]. Next, out of the 54 ETFs, I chose some ETFs that would serve as proxies for economic indicators. Stock market performance depends in large part to different factors of economic performance – such as economic growth, inflation, unemployment, business confidence, and housing. I chose the following ETFs to serve as proxies for these indicators: SPY (economic growth), TLT (inflation), VXX (unemployment), XLY (business confidence), and VNQ (housing). The pricing data of these proxies will be added as features to each dataset later. I wrote a function that communicated with Alpha Vantage's API to download the data for each stock.

Data Cleaning and Feature Engineering

Cleaning the data consisted of trimming the rows for pre-market and after-hours periods and market holidays. Investing just during standard trading hours offers higher liquidity and less risk compared to the higher volatility out of market periods. Trimming these periods mimics the strategy that I would use in real trading. Next, some of the datasets had missing values due to inactivity of the market or by collection error. I imputed the missing rows using the next available row and logged the number of rows missing.

Feature engineering included calculating three technical features and adding them to the dataset. Bollinger Bands, Relative Strength Index (RSI), and Moving Average Convergence/Divergence (MACD) are all commonly used by technical day traders [(4)]. For each stock I also added features in the form of the economic proxies' closing price and the same technical indicators as above for each proxy. Finally, I added features for time – such as minute, hour, and day of the week.

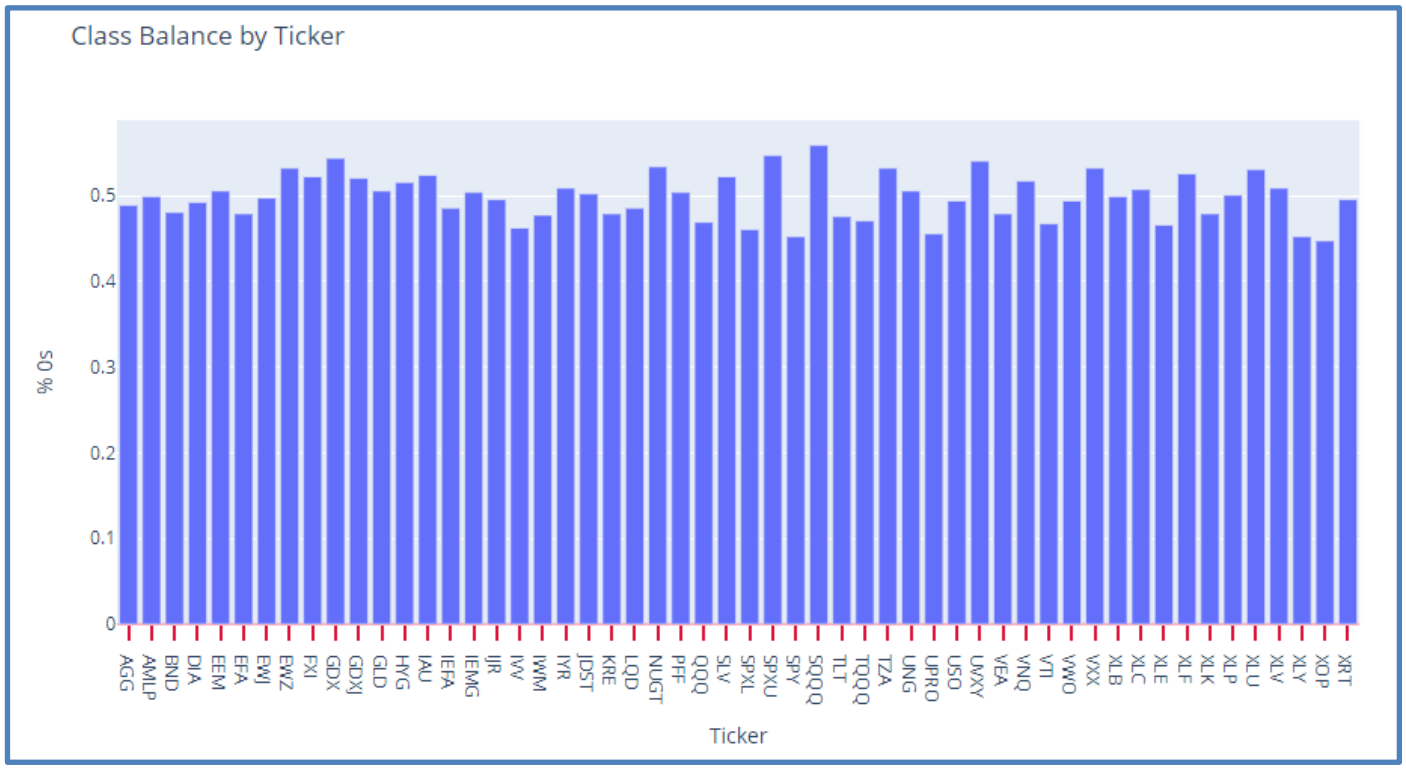
Exploratory Data Analysis

I first took a high-level view of Exploratory Data Analysis (EDA). I had 54 datasets, one for each ETF. Each dataset had 57 features of pricing data, proxy data, and time of day data. The timeframe that each dataset encompassed was 493 trading days, starting on 11-19-2019 and ending 11-05-2021. This represents 192,270 rows, or trading minutes, in each dataset.

I also checked the number of missing rows for each dataset. Greater than 40 of the 54 ETFs had less than 1% of missing data. The ETF with the highest missing data was only 6%.

My EDA included plotting prices for the ETFs, which varied greatly, but allowed me to see the trends in each one. I also wrote a function that found the average percent change of a stock's price over certain time windows. For example, the ETF that track's the Dow Jones Industrial Average (DIA) averaged a price increase of 0.00553% for each 30-minute period. Calculating these average returns forms a good baseline expectation for a given stock's passive performance. This baseline can later be compared to my models' performances.

Finally, I checked class balances. As a binary classification problem, I had two classes – '1' for a stock price increase, and '0' for no change or a decrease. The plot below shows the percent of each ETF's instances that the class was '0'. Class balances are close to even across ETFs. This balance was not surprising given the nature of the stock market but was nonetheless encouraging as the balance makes machine learning tasks easier to structure and evaluate.



Data Preparation

A few steps of data preparation were necessary to get my cleaned datasets ready for deep learning. First, I used a function to transform the raw cleaned data to be stationary. Deep learning models perform better on stationary data, which is where the mean, variance and autocorrelation do not change over time [(5)]. Stock prices experience large trends over time, so I implemented a function to transform the dataset to be stationary [(6)]. Second, I split the dataset into a training set, validation set, and test set. The training set included the first 70% of the original dataset. The validation set included the next 20% of the dataset, and the test dataset included the remaining 10% which was held out for model evaluation later. Then, I feature scaled the datasets based on the mean and standard deviation of the training dataset, applying that same scale to the validation and test sets.

Data Generator

One last step was required before I was ready to begin training my deep learning models. As a reminder, I structured the machine learning problem as a binary classification problem. The input to the model would include the most recent 3 days of data and the model would predict whether the stock price would be higher or lower at the end of the next 30-minute period. For this structure, I needed to feed the model data in sliding windows. For example, the trading day starts at 9:30 AM EST. So, the input to the model at this window would be the most recent 1,170 minutes – or 3 days – of data, and the model would predict whether the price would be higher or lower at 10:00 AM than it was at 9:30 AM. Then, the window would slide 30 minutes and the model would make a prediction for the next 30-minute period (10:30 AM) in the same manner, and so on for the rest of the dataset.

Training the models in this way required a lot of data windows. Simply saving so many windows of 1,170 rows in memory was neither feasible nor efficient. Therefore, I programmed a data generator function to split the data in real time as it inputs it into the model during training.

Model Building and Training

Finally, I was able to build the models and train them on my datasets. I started out by testing different model architectures as proposed in Chapter 6 of "Deep Learning with Python" by Francis Chollet [(7)]. I settled on a model that used two stacked LSTM layers, as was recommended. Each layer utilized both dropout and recurrent dropout. I tested different layer sizes, different values for dropout, and with adding additional layers before settling on the final model architecture – one which consistently produced greater than 50% accuracy. This accuracy threshold may seem low relative to other machine learning applications, but some of the most successful hedge funds are only profitable on slightly more than half of their trades [(8)].

The time it took to test the different architectures and parameters was not insignificant. Given more computing resources, I would have liked to test more options. But, at this point I was able to move on to training models for each ETF dataset. I wrote a function to take the

prepared dataset; split it into train, validation, and test sets; input them into the data generator; and finally, to train and save the models. I also added functionality to evaluate each model after training and to log the results. After much time processing, I had different models for each of the 54 ETFs trained, validated, evaluated, and saved.

Results

My research question asked whether deep learning could be used to find stock pricing patterns for ETFs in different time windows. The results of this project indicate that, yes, patterns can be found. This answer however comes with the large caveat that patterns in historical data do not guarantee patterns in future data. Another caveat is sample size. My overall dataset included 2 years' worth of data, and the test data – as 10% of the whole – represents only 10 trading weeks.

When viewed on all trading windows together, 18 of 54 models had an ROC score of over 50% on the test dataset. Said differently, only these 18 had more predictive power than random chance. However, when looking at the models more granularly, many models achieved higher scores for specific windows within the trading week. The interactive dashboard that I created with Plotly's Dash [(9)], helps to show model performance by ETF by trading window. For example, choosing all the models with an ROC score greater than or equal to 75% for the first trading period (9:30 AM) of each day of the week would provide an average return of 0.88% for the week. Investing in the same stocks passively (buying and holding for each 9:30 AM period, rather than buying only for the 9:30 AM periods that the model suggests), would return only 0.14%.

The models' weekly rate of 0.88% translates to around 57.5% annually, assuming compounding. This is great return, especially given that in this scenario, one would only be invested for the first 30-minute window of each day. This scenario is just one of several profitable examples revealed by slicing and dicing the data table on the dashboard [(9)].

Conclusion

Overall, my deep learning models achieved satisfactory accuracy scores on one third of the ETF datasets. Furthermore, when evaluated on individual trading windows throughout the week, the models were indeed able to find some patterns in the datasets with higher accuracy. Following the models' predictions – on the 10-week testing sample at least – would have provided positive returns well above a passive strategy. The answer to my research question, then, is that deep learning can use ETF stock pricing data to find patterns in different trading windows throughout the day. But as I wrote before, past performance is not a guarantee of future returns. Things can and will change quickly in today's fast paced stock market. It is impossible to know for how long discovered patterns will stay the same. The risk of patterns changing before the next investing decision will always require traders to tread carefully.

Notes and References

1.) https://www.youtube.com/watch?v=dyZUvP5sAw8&ab_channel=AlphaVantageCoin

2.) https://www.investopedia.com/terms/e/etf.asp

3.) List of ETFs: https://etfdb.com/compare/volume/

4.) Technical Indicators Definitions and Calculations

https://www.investopedia.com/terms/b/bollingerbands.asp

https://www.investopedia.com/terms/r/rsi.asp

https://www.investopedia.com/terms/m/macd.asp

5.) Stationarity Definition

https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm

6.) Stationarity Transformation Function

https://analyzingalpha.com/check-time-series-stationarity-python#augmented-dickey-fuller-adf

7.) Chollet, F. (2018). *Deep learning with Python*. Manning Publications.

8.) Zuckerman, G. (2020). *The man who solved the market: How Jim Simons launched the Quant Revolution*. Penguin Books.

9.) Plotly Dash Interactive Dashboard: http://chpr1410.pythonanywhere.com/