Connor Uzzo
Data 1030
Final Project Report

Machine Learning Assisted Portfolio Optimization

Introduction:

With the increased popularity of stock trading apps like Robinhood, more people are choosing to invest at least some of their money in the stock market. While some people enjoy the thrill of gambling with such investments, many people simply want a safe place to put their money that should steadily increase in value faster than their bank account. This long term investor would be well advised to hold a portfolio of multiple stocks in different industries, a process called diversification. My goal with this project is to develop a machine learning regression model which will take as input stock price data for a diversified portfolio from the last five years, and predict how much each stock will grow so that the portfolio can be optimized via a process called Post Modern Portfolio Theory (PMPT)[1]. This process is well documented in many finance textbooks and involves solving an optimization problem to minimize risk given predicted returns. However, this theory often relies on shotty, inaccurate methods to predict expected returns, which could potentially be improved upon by machine learning. This model would allow an investor who is interested in long term returns to put aside a large sum of money for a portfolio, and at the end of each year (or quarter, month, etc if desired) adjust their investments according to how the market is expected to behave.

Daily stock price data is easily available from Yahoo Finance, by searching for a publicly traded company and selecting "Historical Data". Each row of the feature matrix will include the name of the company, its industry, and its daily stock prices at opening, closing, high, and low. The target variable is not directly accessible from this website however, instead it will be calculated as the growth in a stock price one year in the future from the feature matrix data. While this does limit the amount of data that can be used, if this model works effectively it will be a wiser way to invest long term than trying to optimize our allocation of assets based on current data. This strategy interests me because usually the stock market is too random in nature due to unanticipated effects (such as the covid 19 pandemic, trade wars, etc.) to reliably predict its behavior in the future. However, regardless of whether or not the market is affected by unexpected events in a given year, there still will be some optimal allocation of assets that may not even change significantly in response. Therefore, I think machine learning will be able to predict long term stock market trends better than short term trends, even if the model has a relatively high MSE.

Exploratory Data Analysis:

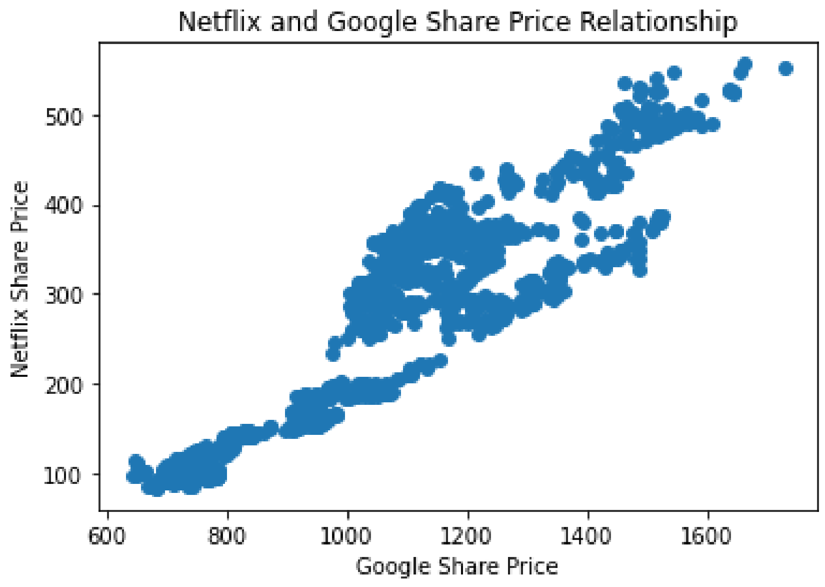

Figure 1: Scatter plot of share price data at closing for Netflix and Google. Note that there is a distinct linear-looking positive trend in this graph, which would suggest these two stocks are strongly positively correlated. Having too many positively correlated stocks will increase the risk since their fluctuations will tend to amplify each other, but it can also increase the expected return.

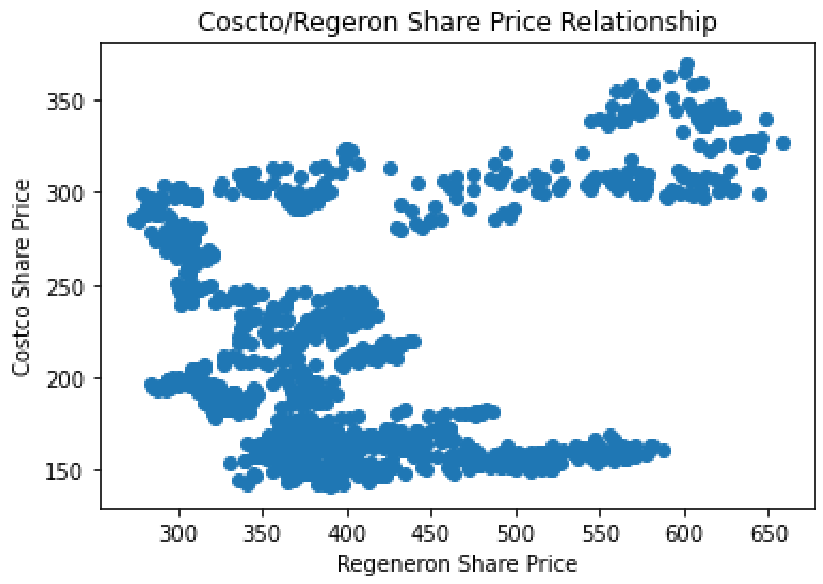

Figure 2: Scatter plot for Costco and Regeneron share prices. Note that there is no distinct linear trend to this graph, which makes these two stocks good for minimizing risk, although they will likely be doing well at different times which can make the portfolio seem like its growing in value slower.

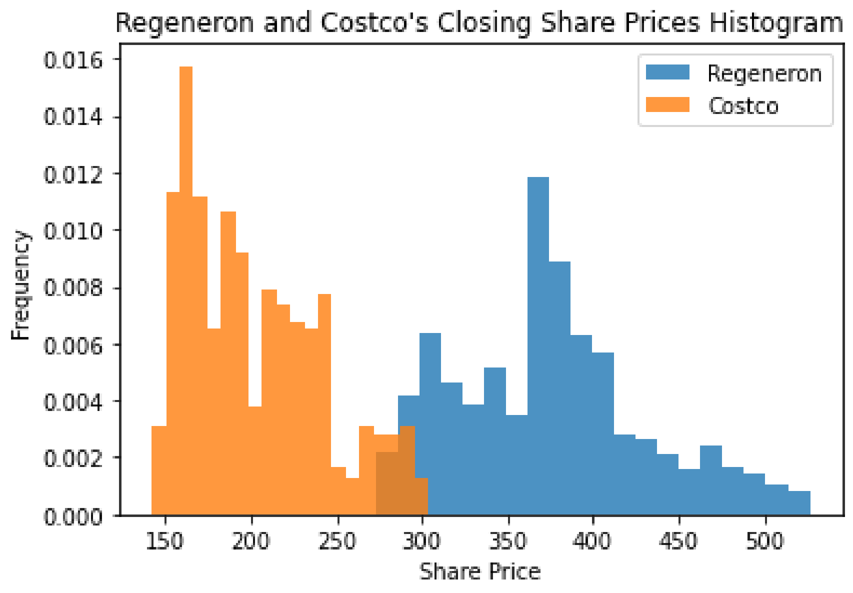

Figure 3: Share price histograms for Regeneron and Costco. While they appear to have a similar amount of variance to their histograms, the Costco histogram is heavily skewed to the right. This shows us that we should not assume that stock prices have symmetric distributions, and they may have heavy tails.

Methods:

One unfortunate reality about dealing with stock market data is the fact that it is not independant or identically distributed since the stock market is almost always growing in size. However, we can assume that it grows at approximately the same rate all of the time if we want to define a target variable that is not likely to increase or decrease with time. For any given stock, its growth in one year is equal to the change in its value over the last year divided by its value at the start of that year. Because our X values will still have some time dependence, however, we are dealing with time series data, and we must keep this in mind when we split it into different sets. The best way to do this is to ensure that our validation and test sets come chronologically after the training set, which can be accomplished using sklearn's TimeSeriesSplit function.

Following the split operations, we want to apply the one hot encoder to our company column as it is our only categorical variable and it has no ordinal structure to it. We apply the fit only to the training data, so as to not bleed statistics into the validation and test sets. For the remaining variables, all of which are continuous, we apply the standard scaler to give them mean zero and standard deviation 1. While the min-max scaler may also work in this application, stock prices can follow distributions with heavy tails when volatility is high, so the standard scaler is probably safer. Altogether we have 6 features in our X matrix, in order they are Company, Open, High, Low, Close, and Volume.

The ML algorithms I tried deploying on the dataset were linear regression with Ridge regularization, linear regression with Lasso regularization, random forest regression, and kernelized support vector machine regression. For both linear regression algorithms, the only parameter that needs to be tuned is the regularization parameter. Both are fed a parameter grid of log-spaced points from 1e-3 to 1e2. For the random forest regressor, the “max_depth” and “max_features” parameters both must be tuned. I chose possible max_depth values of 2, 5, and 10 (this value must be greater than 1 but quickly increases the run time of the fit) and max_features values of 2, 3, 5, and 10 (this value must be less than or equal to the number of columns in the transformed X matrix). Lastly, the kernelized support vector machine regression requires a kernel shape and a regularization parameter as input. The kernels I included in the parameter grid were “linear”, “rbf”, “poly”, and “sigmoid”, which are all built in for the sklearn SVR function. I also included a set of 6 log spaced points from 1e-3 to 1e2 for the regularization parameter, much like on the linear regression algorithms. Conveniently, my dataset has no missing values since stock data is taken every day the market is open, so no extra measures had to be taken to fill in or work around missing data.

Results:

The optimal algorithm and parameters were found by maximizing the negative RMSE between the predicted stock growth and the actual stock growth a year later. It was found that the regularized linear regression algorithms worked the best for predicting this data, as seen below.

Algorithm	**Mean MSE**	**Standard Deviation of MSE**	**Mean R2**	**Standard Deviation of R2**
L1-Linear Regression	0.268	0.00564	-8.16	0.284
L2-Linear Regression	0.373	0.00648	-6.09	0.189
Random Forest Regression	0.516	0.0182	-12.70	1.83
SVM Regression	2.32	0.0235	-4.09	0.188
Baseline	1.249	-	-0.766	-

Figure 4: Table of RMSE and R2 values for various ML algorithms

The R2 values were all negative, so RMSE was used as the primary metric. Since the L1 linear regression had the lowest RMSE, it was chosen to be the best algorithm for this model. There is no single baseline RMSE or R2 for this problem, since different financiers use different methods to estimate stock growth. Some fit a linear or exponential curve to the data to predict what it should be in a year, while others simply estimate a stock’s growth for the year to be the same as its growth for the previous year. I considered the latter method to be a baseline since it is the simplest and makes the fewest assumptions. This method was found to have a root mean RMSE of 1.249, which is better than the SVM regressor but worse than all other algorithms. No algorithm had an R2 value above baseline, which indicates that none of these algorithms work very well universally for this dataset.

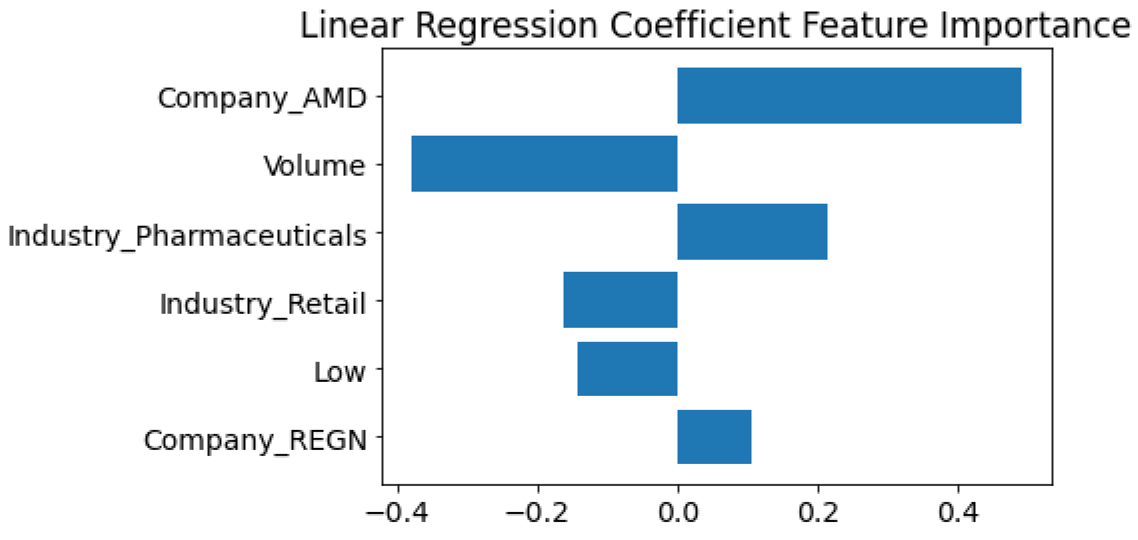

Figure 5: Scaled linear regression coefficients for the best fitting model. These coefficients were calculated on an X matrix with every column having mean 0 and standard deviation 1, so their magnitudes are representative of the global importances of different features. Only six features are shown because all others had 0 coefficients.

Surprisingly, the most important feature was whether or not the company was AMD, and ‘Volume’ and ‘Low’ were the only important continuous features. This is likely due to the fact that all 4 stock price columns (High, Open, Low, and Close) are relatively similar since stocks should not change too drastically in a single day, and so only one of these four features would be needed to make an almost equally effective model. The categorical variables therefore say more about how the stock price will behave than the continuous ones.

Outlook:

Originally, I was hoping to complete this project by including an optimization portion to find an ideal allocation of assets to minimize risk in a portfolio. Unfortunately, because I used free stock data from Yahoo Finance that is taken daily, I was not able to properly capture the volatility (normalized variance) of different stock prices. If the volatility of each stock in the portfolio were known, this ML pipeline could be used to optimize a portfolio for an ideal risk/return tradeoff, based off of the estimated growths of each stock. Additionally, there would be a risk tolerance parameter [2] that the user could tune to increase or decrease their portfolio risk, though there would be a corresponding dip or rise in expected return as well. Some investors may even prefer to keep two separate portfolios with different risk tolerances, one that is high risk but potentially high in return and one that is relatively safe but expected to return less profit.

Also, adding a new stock to this portfolio would be a very difficult and tedious process that would require rewriting much of the project code. This could be improved in a future model by adding a user interface that allows for a selection of several stocks to build a portfolio with. This interface would greatly improve the usefulness and convenience of the model, particularly if it allowed the user to buy or sell stock as well.

It is also worth saying that this model might not handle stock splits very well. A stock split occurs when a company divides its stock price by some value but increases the amount of shares each shareholder owns to make up for it. This action does not add any additional value to a portfolio, but it does change the stock price displayed on finance websites, and could heavily affect regression results. This could be solved via some custom transformations on the stock data, though the user would need to know exactly when the stock splits occur.

References

Ban, Gah-Yi, et al. *Machine Learning and Portfolio Optimization*. Management Science Articles, 21 Nov. 2016, pubsonline.informs.org/doi/10.1287/mnsc.2016.2644.

"Markowitz Model." Farlex Financial Dictionary. 2009. Farlex 13 Oct. 2020 https://financial-dictionary.thefreedictionary.com/Markowitz+Model

Yahoo Finance for Stock Data

Github Repository at https://github.com/connoruzzo/Connor-Uzzo-Data1030-Project