S&P 500 Equity Predictions With Gradient Boosting

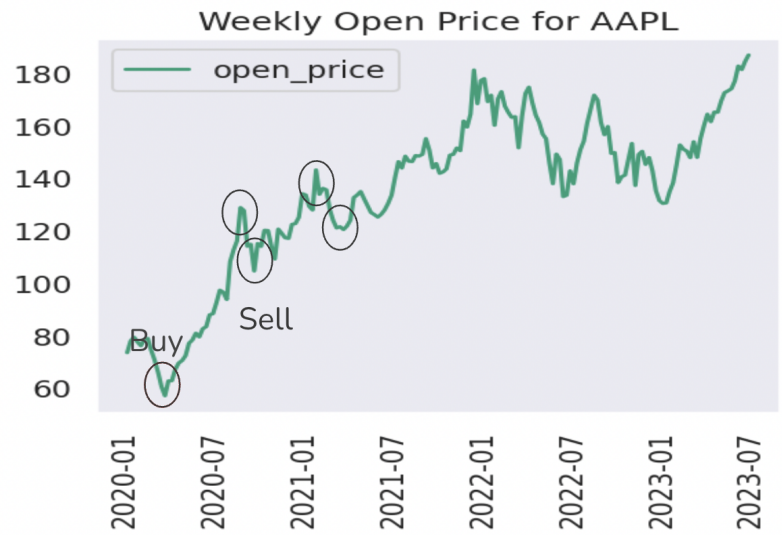
Andrew Koller
Github:

Regis MSDS 696
Practicum

Introduction

Buying low and selling high is the cornerstone of almost any trading strategy. However, timing the relative highs and lows remains elusive to even the most experienced traders. Further research by the brokerage company Charles Schwab demonstrates that time spent in the market is more important for total returns than timing the market[1]. Instead of trying to perfectly time the market, *momentum trading* assumes that short term trends hold over the short term. Equities that have been increasing in price are likely to keep increasing, while equities that are decreasing in price are likely to continue decreasing. This paper explores the possibility of using momentum based features in a Gradient Boosted model to predict whether or not next week's opening price will be higher or lower than the current week's opening price.

Figure 1: Weekly Opening Price of Apple Jan 2020 - June 2023



Data

The primary source of data for the project comes from the 'yfinance' python package, which is essentially an API for web scraping Yahoo Finance. From this package the user is provided with metrics such as opening price, closing price, and volume. The Beautiful Soup python package was used to scrape a list of S&P 500 ticker symbols, as well as their respective industry. From the 'yfinance' package weekly price and volume data was downloaded for S&P500 companies between January 2020 and June 2023.

Figure 2: raw data frame with industry indicators

	Date	Symbol	Adj Close	Close	High	Low	Open	Volume	ticker	industry
0	2020-01-06	A	85.549835	87.589996	88.239998	83.599998	84.000000	8855200.0	A	Health Care
1	2020-01-13	A	88.020920	90.120003	90.279999	86.699997	87.809998	10250000.0	A	Health Care
2	2020-01-20	A	86.204269	88.260002	90.639999	87.580002	89.800003	6756000.0	A	Health Care
3	2020-01-27	A	80.637001	82.559998	88.360001	82.339996	86.540001	10528800.0	A	Health Care
4	2020-02-03	A	81.232796	83.169998	85.500000	82.110001	83.290001	8224000.0	A	Health Care

Feature Engineering

All features used in modeling were calculated using the weekly open price, or equity volume provided by yfinance. The target variable is defined as a 1 or 0 depending on whether or not a given equity increased in price week-over-week. Ex. The target variable for ‘A’ on the week of 2020-01-06 would be 1 since the open price on the following week 2020-01-13 is higher than the previous week. The features considered for modeling include price moving averages, exponential moving averages, MACD, and volume based moving averages.

Features Considered:

Industry

Open price

Rolling 2,4,6 week open averages

2,4,6,12 week Exponential Moving Average (EMA)

2,4,6,12 week rolling volume average

MACD 9-4 week, MACD signal 9-4

MACD 12-6, MACD signal 12-6

Industry 2,4,6 week open averages

Industry 2,4,6 week volume averages

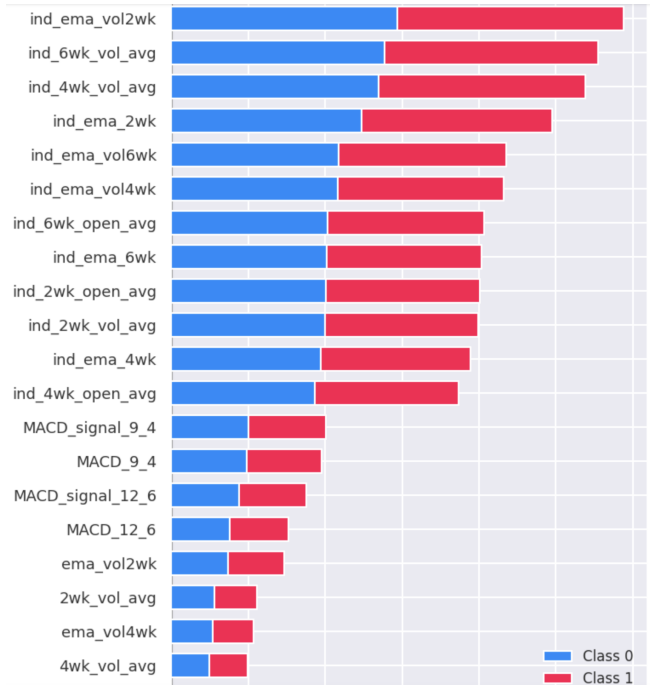
Industry 2,4,6 week EMA

Feature Selection

“Feature selection is a crapshoot, its best avoided” - Anonymous Redditor

Feature selection is cumbersome, and it is often difficult to know if the correct features are being included or excluded due to correlated variables. Since all features being considered are based off of the open price of an equity, or its weekly volume looking at a correlation matrix offers little insight. Rather than using a more standard approach to feature selection like a recursive elimination approach, the features in this project are compared against a random normal variable to measure usefulness. To accomplish this, a random normal variable was appended to the feature set. This feature set was then fed into a random forest classifier, which was then used to generate a list of SHAP values. Since the random normal variable is nothing but noise, any other feature that is shown to be of more importance than the random variable is to be included in the final model. After running the experiment, all features were determined to be more important than random noise, and thus should contribute to the predictive power of the final model.

Figure 3: SHAP Values



Model

XGBoost, or eXtreme Gradient Boosting was used to model the data. These types of models are known to perform well for a variety of regression or classification tasks. These models are an ensemble of weak learners that are combined into a learner that is much more powerful. Functionally, this works by building one weak learner, and using subsequent weak learners to model the residuals of the previous learner while optimizing for the loss function.

Baseline Model

A baseline model was established by training a XGBoost model with default parameters.

Figure 4: Baseline Model Results - Test Set

Metric	**Value**
Accuracy	48.4%
Precision	41.8%
Recall	59.7%

As far as accuracy is concerned, the baseline model performs about as good as flipping a coin. The precision metric suggests that out of all identified price increases, only 42% of the predicted increases were true increases. While the recall metric suggests that the baseline model is able to identify about 60% of positive cases. One does not need to evaluate the model further to determine that using the baseline model as a trading guide would be a great way to lose money.

Optimization

One of the benefits of using a baseline model is that this often represents the worse case scenario for modeling efforts. Models typically perform better when they are tuned to the task at hand. Since the baseline model used all default parameters, it is reasonable to think the predictions could be improved through optimizing the hyper parameters. To accomplish this, Bayesian Optimization was implemented across a set of possible parameter values. Bayesian Optimization is just one of many options to optimize hyper parameters, but offers distinct advantages in terms of speed and generalization over methods like grid search, or random search[2].

Figure 5: Bayesian Optimization Hyper Paramaters

Gamma = 6.5
learning_rate = 0.15
max_depth = 14
n_estimators = 80
reg_alpha = 0.8
reg_lambda = 0.1

Optimized Results - Test set

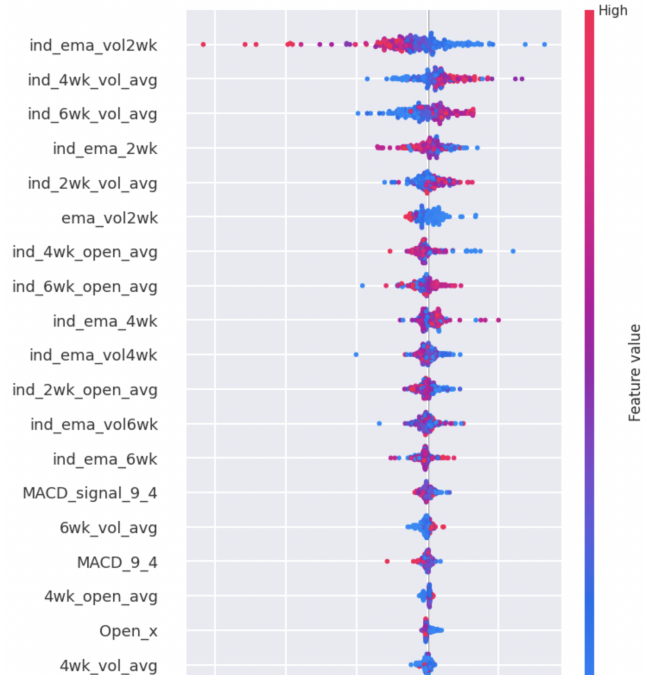
Metric	**Value**
Accuracy	45.5%
Precision	41%
Recall	65%

Building a model with the optimal hyper parameters as defined by the Bayesian Optimization yields an overall much less accurate model, but sees significant gains on the 'Recall' metric.

Discussion

While the optimized model does not perform well enough to gain my confidence in using it with actual money, that does not mean the model used in this experiment is useless. It appears that momentum based features alone are insufficient in predicting the direction of weekly stock price movements with any degree of accuracy. However, the model can still be used to assess what is important for weekly price movements. Looking at the SHAP values of the optimized model gives insight into this:

Figure 6. Optimized Model SHAP values



Interestingly, it appears that Industry specific metrics are more important than company specific momentum metrics. This is somewhat unsurprising as industry trends are less noisy than company specific trends. Nevertheless, this does support the notion that momentum is a real thing for equities. A high tide raises all ships. A better strategy might be trading industry index funds rather than individual equities with a model such as the one proposed.

References

1. Schwab.com. (2021, July 14). *Does market timing work?*. Schwab Brokerage. https://www.schwab.com/learn/story/does-market-timing-work#:~:text=Our%20research%20shows%20that%20the,benefit%20of%20even%20perfect%20timing.&text=And%20because%20timing%20the%20market,invest%20as%20soon%20as%20possible.
2. Koehrsen, W. (2018, July 2). *A conceptual explanation of Bayesian hyperparameter optimization for Machine Learning*. Medium. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f