SPY vs. SPY

OVERVIEW

The idea for this project started with a friendly argument over lunch with a coworker. We were talking about how to invest for retirement. He thought people should invest in the SPY and keep adding to it. I thought it was better to research individual stocks.

Both of us had a credit background so were used to looking at financial statements. So I wanted to see if there was a way to use financial data in a model to see if it was possible to do better than the SPY.

This project aims to determine to see if it is possible to programmatically determine on a quarter to quarter basis how well it can predict which stocks in the S&P 500 might be better to invest in using only quarterly financial data and average quarterly stock prices.

PROBLEM STATEMENT

This project works on creating an algorithm to pull in financial statement data from companies that make up the SPY to predict over a set of quarters which stocks might do better than the SPY itself.

Since we are looking at the companies that make up the SPY, naturally at least some of the stocks are driving the SPY. Can we find out which ones are doing so? Can we predict which ones using an algorithm?

The question we are trying to answer is:

Can we come up with a model or models that could be used to predict if any individual stocks would do better than the SPY from quarter to quarter?

We will be doing this by determining if the quarterly financial data of the tickers that make up the SPY, along with the average previous quarter's prices from those tickers, can predict the following quarter's prices for stocks.

Note the above is a very different question than "how much can we beat the SPY by"? For this project we're not trying to determine how much a stock might outperform the SPY, only we can come up with models that might be able to predict whether they outperform the SPY or not.

A major challenge in predicting the stock market is that you're training models on past data. For example, if you train a model on how stocks performed from say 2Q2022 to 3Q2022, then you predict how stocks might perform from 3Q2022 to 4Q2022 using the model trained on the previous quarter's data, there might be a large drop off in metrics between the test results from data as of 2Q2022, versus predictions based on data from 3Q2022

Since we're looking at past data, we can do back testing and look at how the predictions fared. We wouldn't have this luxury in real time of course until after the fact. This is perhaps the major challenge for market predictions.

We will be looking at a period of 6 quarters. In this time, the SPY clearly makes a down movement then an up movement. I'm deliberately picking this time period to see if we financial data can predict stock price movement from quarter to quarter.

Important note: before we use models to predict individual stocks, we need to see if the model has any value in the first place. This project is more concerned with the overall quality of models (ie, precision levels when predicting on the following quarter's data) rather than picking individual stocks. There is no point in predicting individual stocks if the overall precision is not particularly good.

METRICS

I am using classification models for this project and the most important metric would be precision. When a model predicts a stock will do better than the SPY in the next quarter, we want to increase the odds that this is in fact the case. It's more important for this project that stocks predicted to do better than the SPY actually do so. If a stock is not predicted to do better than the SPY and it actually does would be a missed opportunity, but we won't lose existing capital over that.

To determine which models to use, I am using precision.

DATA EXPLORATION

The first step was to find out where to get the list of tickers that currently make up the SPY.
I found a Wikipedia site that lists the ticker that make up the SPY and corroborated this with other websites.

As of the time of this project, there were 503 tickers that make up the SPY.

Why 503 tickers and not 500? Because 3 companies had 2 tickers each: Google, Fox News and News Corp. This is fine as it does not change the fact that each ticker makes up a piece of the SPY pie. Instead of 500 pie pieces, there are 503 (not necessarily the same size).

We will be looking at a range of 6 quarters to analyze the output of 4 of them. The time period we will be using is from the 2nd quarter in 2022 going through the 3rd quarter in 2023.

Aside from getting the list of tickers from wikipedia, and also a definition to scrape them thanks to this stack overflow website as the list changes over time:

Outside of the list of SPY tickers, we will be getting the bulk of our data from Alpha Vantage. Full disclosure: while there is a free tier, there is a cap of 25 pulls per day for the free tier. To replicate this project, a paid account is needed for at least 1 month, which costs $50/month. With the paid account you should be able to pull all the data needed for this project.

We are looking at 4 main pieces of the 503 tickers: time series data, balance sheets, cash flow statements, and income statements. All of the financial data is commonly used financial data fields.

Out of the 503 tickers, 9 of them started late, meaning they were added after the time period that we are looking at for our project so these will not be used in modeling.

While Alpha Vantage has both time series data and also financial data for all 503 tickers, there are some missing data in the financial statements. We will address this in more detail in the data preprocessing section.

Concerning time series, I am using the adjusted close price for all tickers. This accounts for 2 very important items: stock splits and dividends. If we simply used the close price, for some tickers it would display large movements up or down due to stock splits or reverse stock splits.

Example: if you owned a stock that was $100 a share and there was a 2 for 1 stock split, after the split, you'd own 2 shares each for $50. You have the same amount of money, but a different number of shares. So you have neither a gain or a loss. So it's not a movement that results in a gain or loss, simply an adjustment.

And dividends are also important. Some companies pay dividends that are frequently better than a savings account interest. 2%, 3%, or even higher. We need to take this into account as these are clearly part of the overall return on the investment.

The class problem. One frequent problem exists when trying to predict the stock market: a class problem. Meaning, often, the market moves in one direction and most stocks tend to go in that same direction. So we either have many stocks that go up and few that go down, or vice versa.

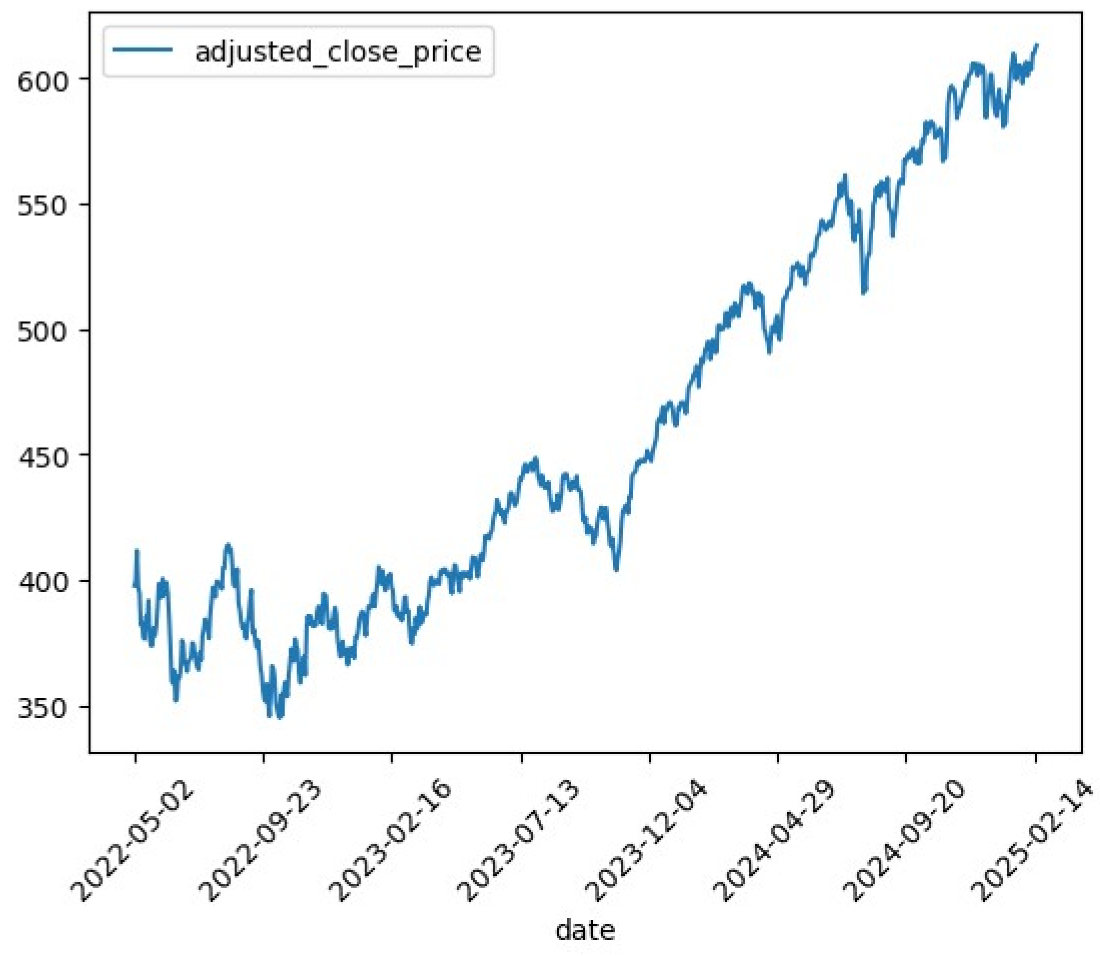
In an attempt to mitigate this issue, I created a target variable called “Better than SPY” which is a 1 or 0. If a stock’s return from one quarter to the next was 2.4% and the SPY return from that same quarter was 2.2% then this stock’s “Better than SPY” value would be a 1, meaning for that quarter’s movement it performed better than the SPY. This would also apply to downward movements. For example, if the SPY’s return for one quarter was -2.9% and a stock’s return in that same quarter was -1.5% then that stock would perform better than the SPY in that particular quarter.

Using a variable like “Better than SPY” doesn’t mean the classes are always 50/50, but it does mitigate some of the variance than if we simply used whether the stock return was positive or negative. Some quarters have a 2.5 to 1 ratio concerning classes, which is about the worst the class imbalance appears to get – at least during this time frame.

DATA VISUALIZATION

Most people know that over the long term, the stock market goes up. In particular, the index funds like the S&P 500, the Nasdaq and the Dow Jones

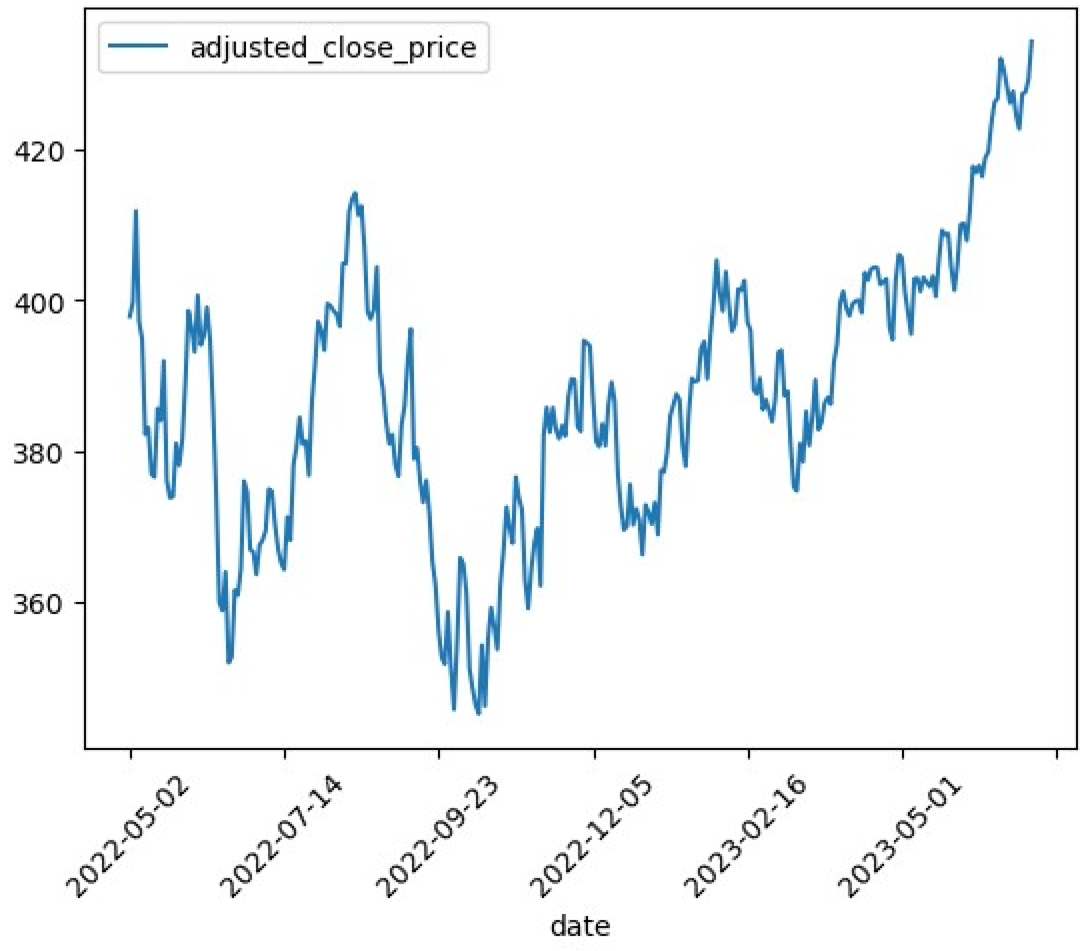
Here is a snapshot of the SPY from May 2022 through FEB 2025



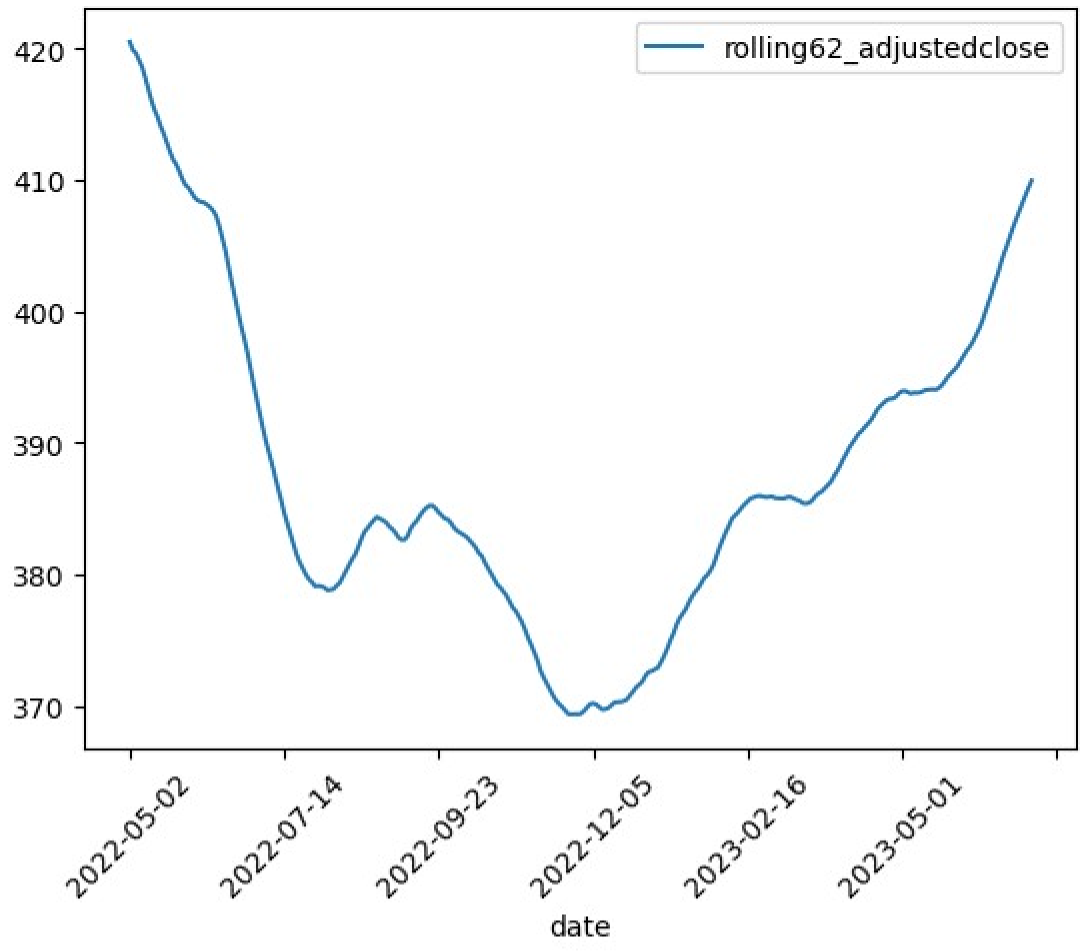
Clearly there is an upward trend overall.

However, if you look at the earlier part of this graph, it’s flatter than the later part.

If we take a closer look only looking from May 22 through SEPT 23 we can see that while the overall movement is up, there are clearly some up and down trends in the middle of the graph:

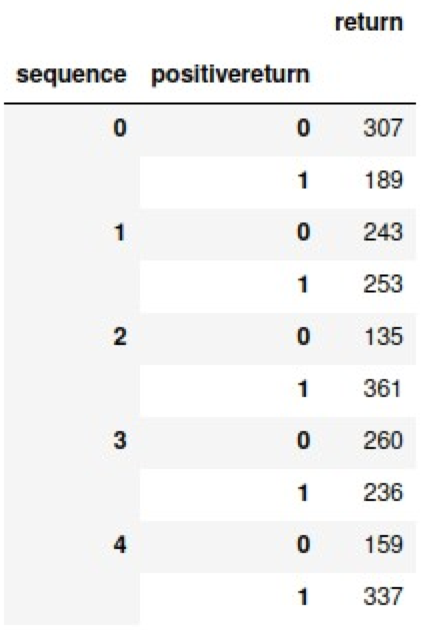


If instead of using the daily adjusted close price, we use a rolling 62 day adjusted close price, there is an even clearer picture of a downward trend in the first half, then an upward trend in the latter part.



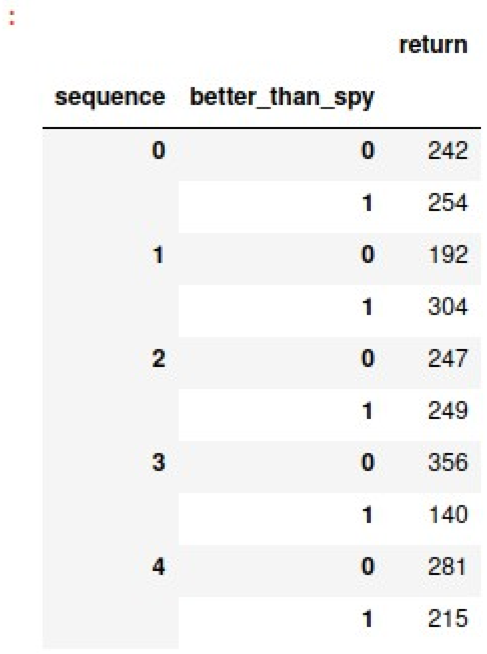
We will be using the 62 day rolling average adjusted close prices (above) from 6-30-22 throught 9-30-22, but we will only be using 6 days to create, evaluate and run our modeling. The 6 dates are: 6-30-22, 9-30-22, 12-30-22, 3-31-22, 6-30-22 and 9-29-22. Keep in mind that some months had their last days fall on a weekend.

The next item to look at would be the classes we will be using. At first I considering using whether the stock return over a quarter was positive or not. However there is a degree of variance from quarter to quarter in the classes. Training a model in one quarter with a certain distribution, then testing it on the following quarter with a potentially very different distribution might result in poor performance.



		return
sequence	positivereturn	
0	0	307
	1	189
1	0	243
	1	253
2	0	135
	1	361
3	0	260
	1	236
4	0	159
	1	337

So I created a new field for the target variable called better_than_spy. 1 for yes. 0 for no.



:

		return
sequence	better_than_spy	
0	0	242
	1	254
1	0	192
	1	304
2	0	247
	1	249
3	0	356
	1	140
4	0	281
	1	215

Considering that there are only about 500 rows, reducing the row count to make things more even is not a path I want to go down. 500 rows is on the lighter side already.

I also took a look at the features after filling in blanks to look at the correlations. Nothing stood out at more than about 0.6 so it seems reasonable to start with using all of them.

DATA PREPROCESSING

The time series data from Alpha Vantage is pretty clean. Using the adjusted close price eliminates the need to manually adjust for both stock splits and dividends, both saving some time and also reducing unnecessary complexity.

But in order to use the time series on a quarterly basis, I added a rolling 62 day adjusted close price field. 62 days is an average number of business days in a quarter.
Then, selected only 6 dates corresponding to the 6 end of quarter dates that we will be looking at. I did this for each ticker as we will then shortly combine these values with the financial data.

The financial statements however had a few things to deal with.

First there was some missing data for some companies in some quarters. Second, some companies had multiple submissions of financials in the same quarter. We will address these one at a time.

The data wrangling script goes through 1 ticker financial data piece at a time (balance sheets, cash flow statements and income statements each done separately but use a similar concept that will be described here)

First we order the data by fiscal date ending (all 3 types of financial data had this field so I was able to use 1 function for all 3).

Next we work on filling in any missing data. First, we use a linear interpolation to fill in missing data in between ordered rows that did have data. This makes more sense than using mean, median or mode as we're looking at a financial time series. Next, we use a back fill in case any values in the beginning we're missing, then finally we will use front fill for anything at the end that is missing. Fortunately, since we're looking at data in the middle periods, it would not be frequent for either a front or back fill to happen.

Next, we create a column called “quarter” using the Fiscal Date Ending, then sort by ticker, quarter, then Fiscal End Date. Technically we don’t need to sort by ticker since these are individual files at this stage, but, it helps me keep track of things.

Now we drop duplicates by ticker and quarter, keeping the latest one (since we have Fiscal Date Ending as the tertiary sort).

So far so good. We still have a few instances where a company has 2 submission rows on the same date. In this case, I’m using a function called “calculate non blanks in row” and keeping the row with the most non blanks. At this point, there should not be any but just in case, I’m adding the count of non nulls as a new column, then resorting the dataframe by ticker, fiscal end date, then by count of non nulls in the row (high to low), keeping the first one.

Also in the wrangling section, I added another definition that consolidates the time series data with the financial data and creates input data for our machine learning.

Then in a function to create machine learning input data, we pull in the consolidated data from the time series as well as the data from the 3 parts of the financial statements. We merge all 4 of these dataframes using an inner join , and using ticker and quarter for the join.

We drop some columns not particularly useful for machine learning such as CIK numbers (unique for all tickers), HQ location (with only 500 rows, this is too thin), currency (all are in dollars),

Additionally in this function, I added financial ratios to get additional interactive features using definitions from the corporate finance institute.
https://corporatefinanceinstitute.com/resources/accounting/financial-ratios/

Since many financial analysts use these ratios in determining credit amounts to loan to a company, it seems logical to add these to our code as – at least in theory – we might risk money buying a company’s stock.

Also, when analyst’s use financial ratios, they take industry type into account. Therefore it makes sense to use the industry type in the alpha vantage data and convert to dummy variables.

In the machine learning data, I’m replacing quarters with sequence numbers from 0 to 5 for convenience. Each quarter’s data has the y variable from the next for training purposes. So, we train on the first quarter (sequence 0) which uses the average rolling stock price from the next quarter (ie, sequence 1) to train.

IMPLEMENTATION

I went at this project's goal as an agnostic, not assuming any particular machine learning model would do better than any other, letting the results speak for themselves.

But I chose classification models over regression as I simply wanted to see if an algorithm could find stocks that perform better than the SPY, not necessarily by how much, which could be a project for another day.

Since each quarter only had about 500 rows, I thought "let's use that to our advantage". We can try mulitiple models as it takes less time to train on 500 rows than 100,000 rows. Therefore we could try out different kinds of models as well as experiment with feature reduction and also grid searching.

To investigate our project question I compared various models, collecting metrics of the test data from the set of quarters to analyze each model's performance. As previously stated, the main criteria for this project is precision. This would determine which model(s) to use.

First, I tried using 5 unmodified frequently used classification models: Random Forest Classifier, Gradient Booster, SVM, Logistic Regression and KNN. Comparing metrics in backtesting from quarter to quarter. The average precision of the 5 generic models (generic means no grid search or feature reduction) was in the low 60s. This is certainly quite a bit better than casino odds, but could we improve that?

Then I used a voting model of the 5 to see if that did better. Using a shuffle data approach, the voting model typically had a precision between the best and worst of the 5 individual generic models.

REFINEMENT

Next I looked at feature reduction using a Random Forest model to do so, trying a range of number of features from 50 to 100 with a space of 10 between them (ie, 50, 60, etc.). Then ran the reduced number of features through all 5 generic models.

Again, using a shuffle parameter for the models, 90 seemed to have the best precision compared to other features, but was below the generic model average precision.

Similarly I tried a voting model on the outcome of the reduced features (same features for each model). The precision here was again between the best and worst of the individual models.

Next I used a grid search on each of the 5 models listed above. I did not use every single combination of parameters but some of the more commonly used ones, considering time. Even though there are only about 500 rows per input data, we’re still training 15 models for each of 5 quarters, so 15*5 = 75 trainings.

The average of the grid search results was actually slightly less than just using the generic models.

Then I looked at a voting model on the outcome of the grid searches.
As with the other voting models, the precision seems to fall within the best and worst of the 5 individual models.

Ultimately, I ended up using the 5 individual models as the final test for 2 reasons: 1) the precision was slightly higher and 2) why use more computer computational power than necessary if there is little to no benefit in doing so.

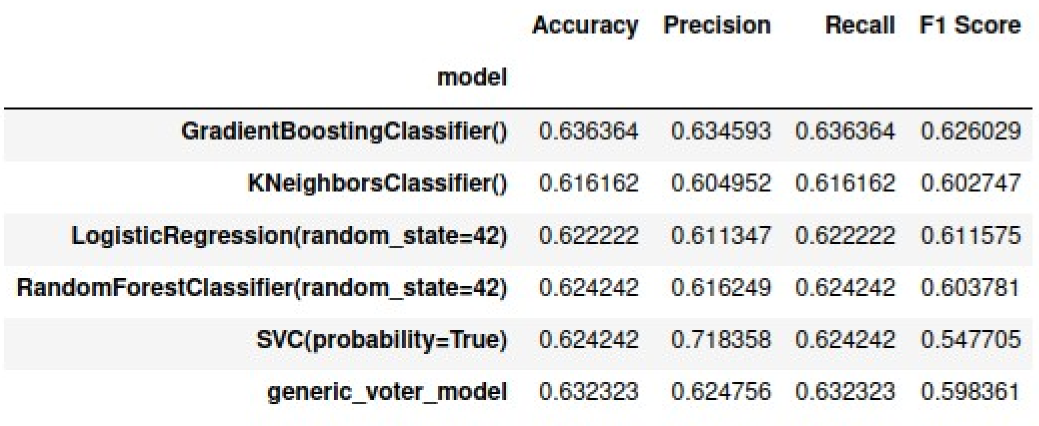
For picking individual stocks to see how they performed against the SPY, I used the average probabilities of the 5 generic models, since higher probabilities means the model is more confident that it’s choice is correct. If the average probabilities of 5 models suggested a pick, then the algorithm in the final script picked that ticker.

MODEL EVALUATION AND VALIDATION

I evaluated each model set over multiple sequences (the generic models, the reduced feature models and the grid search models), paying particular attention to the precision, while not ignoring other metrics. For the record, the accuracy and recall were close to the values of precision in every case, looking across multiple sequences for a more robust metric.

Here are some tables, summarizing the results of 3 groups of models: the generic set (no changes made either with number of features nor doing any grid searching), the set with reduced number of features and then the set using grid search. Then we’ll look at a table of averages for the 3 groups side by side.

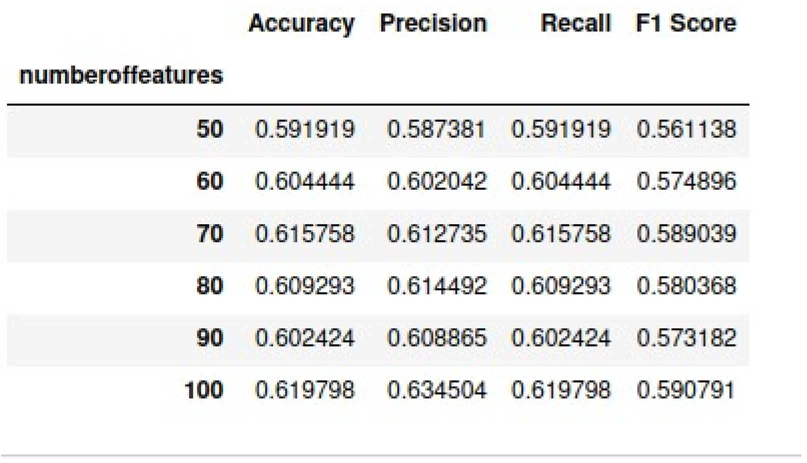
Here is a table of average test scores for the 5 generic models, including a voter model to see how that might do compared to individual models.



model	Accuracy	Precision	Recall	F1 Score
GradientBoostingClassifier()	0.636364	0.634593	0.636364	0.626029
KNeighborsClassifier()	0.616162	0.604952	0.616162	0.602747
LogisticRegression(random_state=42)	0.622222	0.611347	0.622222	0.611575
RandomForestClassifier(random_state=42)	0.624242	0.616249	0.624242	0.603781
SVC(probability=True)	0.624242	0.718358	0.624242	0.547705
generic_voter_model	0.632323	0.624756	0.632323	0.598361

We can see that all precisions were over 0.60, with one model over 0.7. The voter model fell in the middle.

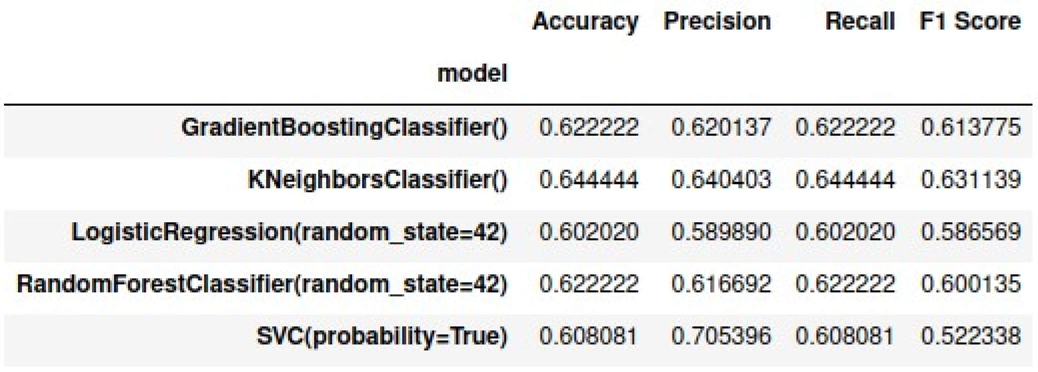
Next, is a table looking at averages of different numbers of feature reductions between 50 to 100 (with steps of 10).



numberoffeatures	Accuracy	Precision	Recall	F1 Score
50	0.591919	0.587381	0.591919	0.561138
60	0.604444	0.602042	0.604444	0.574896
70	0.615758	0.612735	0.615758	0.589039
80	0.609293	0.614492	0.609293	0.580368
90	0.602424	0.608865	0.602424	0.573182
100	0.619798	0.634504	0.619798	0.590791

using 100 features, based on precision looks like it works the best. However, this is a very minor drop in the number of features compared to using all of them.

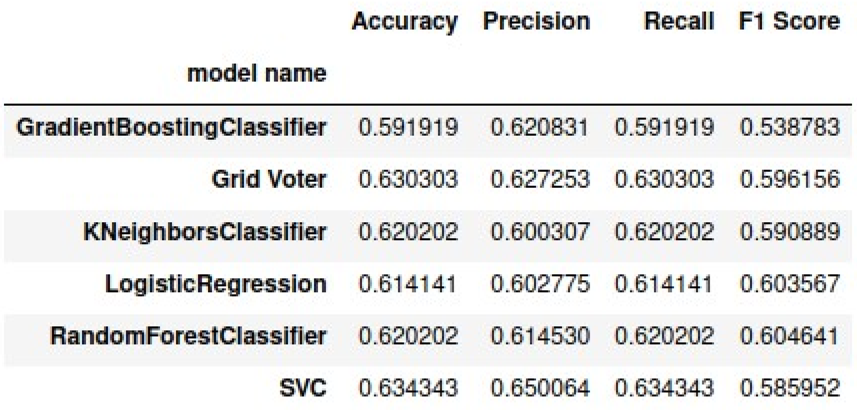
How do the individual models score using 100 features?



model	Accuracy	Precision	Recall	F1 Score
GradientBoostingClassifier()	0.622222	0.620137	0.622222	0.613775
KNeighborsClassifier()	0.644444	0.640403	0.644444	0.631139
LogisticRegression(random_state=42)	0.602020	0.589890	0.602020	0.586569
RandomForestClassifier(random_state=42)	0.622222	0.616692	0.622222	0.600135
SVC(probability=True)	0.608081	0.705396	0.608081	0.522338

While a few models performed a little better than using the generic model most scored slightly lower.

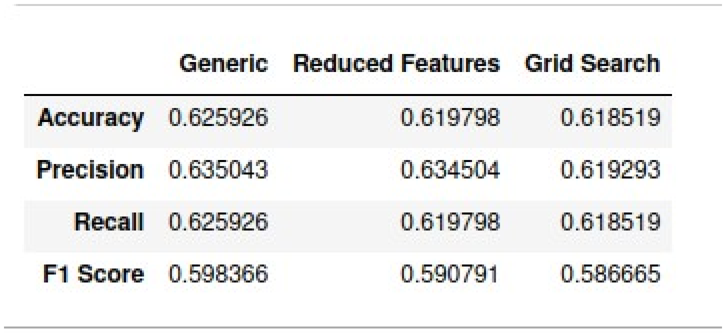
What about using a grid search? Here are the averages from going that route. This includes a voter model just in case that does better.



model name	Accuracy	Precision	Recall	F1 Score
GradientBoostingClassifier	0.591919	0.620831	0.591919	0.538783
Grid Voter	0.630303	0.627253	0.630303	0.596156
KNeighborsClassifier	0.620202	0.600307	0.620202	0.590889
LogisticRegression	0.614141	0.602775	0.614141	0.603567
RandomForestClassifier	0.620202	0.614530	0.620202	0.604641
SVC	0.634343	0.650064	0.634343	0.585952

Most models, including the voter model, score in the low .60s, with one in the mid .60s.

For the final visual from test results, here is a side by side average of the 3 groups:



	Generic	Reduced Features	Grid Search
Accuracy	0.625926	0.619798	0.618519
Precision	0.635043	0.634504	0.619293
Recall	0.625926	0.619798	0.618519
F1 Score	0.598366	0.590791	0.586665

We can see in the above that just using generic models performs better overall (judging by precision) vs using either feature reduction or a grid search. In the above, the best feature reduction only pulled out a few features and there was a slight drop off in precision. For the grid search, the average drop off was higher by more than 0.015.

Also when voter models were used, they tend to perform within the range of the best and worst performing models in the set.

Based on this, for the final script, back testing on future data, we're using the 5 generic models since these had the highest precision.

Overall the generic models had about 0.635 precision using test data from the same quarter's data, then an average of about 0.57 using the next quarter's data as test data.

Again, 0.57 is certainly better than a casino, but leaves room for error.

The final evaluation step was to do actual picks to see how they would have performed against the SPY over the period we were looking at. To do this, I used average probabilities of the 5 generic models and for each quarter, picked the top 5 stocks, doing this over multiple sequences.
I also added a shuffle parameter and ran this script multiple times. Generally the results performed better than the SPY.

However, I would like to add a cautionary statement as their could be potential data leakage issues.

While I was very careful not to train models on future data, there could still be at least 2 sources of data leakage in this project.

The first potential source of data leakage is that, if we were doing this in real time, it's possible that not all companies would have submitted their financial statements by the end of any given quarter. In other words, they may have filed for an extension (or more than 1). That means we might have a reduced set of financial statements for more recent quarters, some of which might affect training models.

A second potential source of data leakage could be that there may be errors in financial statements that were later corrected. There might be a typo or miscalculation somewhere that was later corrected and resubmitted. It's possible that some errors, particularly more recent ones, may not yet be corrected, which could also potentially affecting training models as the errors could be in the training data.

JUSTIFICATION

What is the justification? I think given inflation over time and the unpredictability of life events, this is a question worth asking. Is it possible to get better returns to help prepare for retirement?

But really where the justification could be argues is how did the set of chosen models do with back testing on future data.

As mentioned previously, I added a separate script, using a shuffle data approach that picks 5 stocks per quarter, then evaluates how those picks actually performed against the SPY. Most of the time, the picks did better than the SPY. For testing purposes the reason I used 5 picks was to spread out the risk.

Based on running this last script numerous times, it seems there is some possibility in this approach as it usually did better than the SPY.

CONCLUSION

This was an interesting project and I'm glad I was able to explore a concept from a friendly argument from several years ago.

Based on the outcome of primarily using financial data to predict stock market price movements, I would default to investing in the SPY. While I had fun exploring this possibility, it's much easier to simply invest in the SPY as over the long term it tends to go up, even though there can be large pullbacks.

REFLECTION

Disclosure, this was a theoretical model. I have not personally used this.

In order to consider using a model like this I’d want to do additional testing. While it seems to have worked for this period, I’d prefer to see it over a longer period of time through different economic seasons.

Also, it needs to be said. Even assuming this approach worked over very long periods of time (which remains to be seen), if the SPY goes down say, 5% in a quarter and the model picks only went down 4%, while that – in theory – is better than SPY returns, it still results in a loss. In other words, there is no guarantee of having positive returns.

Investing has risk that simply cannot be eliminated.

IMPROVEMENT

Some potential improvements could be using other types of features, not just financial data. I mean things like technical analysis data like the RSI or MACD. One might also consider incorporating sentimental analysis, or macro economic data such as inflation rates, GDP, employment rates, etc.

Or perhaps broaden the study and use a larger set of stocks that might be more conducive to a grid search with less risk of over fitting.