SIADS 694/695 (FALL/SUMMER 2021)

Stock Price Prediction

Michael Penrose and Allison Bergmann

Project Summary

This Milestone project builds on the Milestone I Project that Michael completed with Damon Clifford. The goal of this project is to study stock market data and predict future returns. We extend the data, data manipulation processes, and feature generation approach from the earlier Milestone project to create the dataset for this project. We also explore unsupervised learning to attempt to get a better understanding of the categories of stocks included in this project, and what useful business insights could be gained from those categories. Our dataset ultimately included 253 UK stocks with a timespan from 2017 to 2021.

Data Manipulation Methods and Process Analysis

Dataset

The input data consists of 3 types of features. They are 1) stock price histories including amount traded per day (volume), 2) fundamental data about the company such as revenue, earnings per share, and 3) technical features derived from price histories and volume. The fundamental data was retrieved from Stockopedia which is a small but rapidly growing UK-based financial data provider which operates with a paywall. Special permission was received from the CEO of Stockopedia to use their data for this project. Unfortunately, no API was available for this and historic data had to be downloaded as PDF files which were then parsed in Python using the tika library. In Milestone I Michael built a web-scraping tool to do this. In this project we reused and expanded on that work to collect the fundamental data needed. We retrieved price history data from the popular yfinance API. This data can be retrieved directly as a pandas dataframe. The technical Finance-domain specific features were generated as new features in pandas dataframes using the Finance-domain Python packages ta-lib and ta. These packages took the price data from finance as inputs. The Milestone 1 Project report is included in the final submission for this project for ease of reference and contains more details on how the data was originally collected and collated. This information is provided for background only as we consider this project to start with the dataset already prepared. This is because we do not want to retread too much of the ground covered in Milestone I and prefer to focus on the new work done here.

The final combined dataset contained over 1000 features, the vast majority of which were technical features. The technical features are generated with input parameters, and we experimented with a wide range of input parameters for each of the types of technical features. The target, y, is the price change from the date of the specific row of training data to the price of that same stock 20 trading days later. We call this feature ‘y_20d’. This is expressed as a factor. So, if the price rose by 10% in the 20 trading days after the date of a given row in the training data, then the target would be expressed as 1.1. Since stock market prediction is inherently a time-series prediction problem, the dataset was split to take account of that. Approximately the first 3.5 years of data was used for training data. Then a gap was left of 20 trading days to make sure that there was no leakage of information from train to test. Most of the remaining data was split randomly and equally into test and validation without any time sequence. The train set ran until July 2021 (and the target values until August 2021). A final additional validation dataset was generated from the last month of data in the last week of the project. So, this final validation set ran from July to August 2021 with the target values spanning August 2021 to September 2021 (since the target values on any given date were the change in price over the subsequent 20 trading day period). This was done to ensure that a final true out-of-sample validation could be run against data that was not available at the start of the project. The reason for this is that in time-series problems information leaks in settings that are chronologically incoherent are common and commonly undetected. Our final validation set collected only in the last week of the project contained data that had never previously been shown to the models and therefore leaks were avoided.

In working with this data there were some steps that were repeated frequently and there were some interim states of the data that we frequently used. As a result we built a pipeline of data preparation steps for the supervised portion of the project which we supported by setting up a utilities python file that we imported to the Jupyter notebooks both for supervised and unsupervised. The utilities took care of things like loading the data, data cleaning, data manipulation and also a data pipeline that did multiple steps that the supervised process was also going to need. After splitting the data into train, test and validation with X and y stored separately we saved each of those dataframes to their own pickle file to ensure that they were constant and did not get modified e.g., through an inadvertent change to the random_state variable in the data splitting process. These utilities proved very useful to create clean, reusable code, and have the benefit of making the Jupyter notebooks easier to read by allowing them to focus on the modelling and analysis stage of the project after the utilities took care of most data cleaning and manipulation.

Supervised Learning

We chose this project for supervised learning because we both have a background in Financial Services and an interest in trading, particularly algorithmic trading. Additionally, the signal to noise ratio in trading is known to be low and thus predicting future stock market returns presents a non-trivial machine learning challenge. The specific goal of the project was to predict stock prices 20 trading days into the future. Clearly, this goal impacted the way the dataset was constructed which is described above in more detail.

Evaluation

In selecting appropriate evaluation metrics, we considered the nature of the business problem. If an approach like this was used to inform trading decisions, and the goal of this project is to explore whether supervised learning approaches offer this type of promise, it would make sense to focus only on the stocks that are predicted to go up the most (assuming a long-only strategy). If this is run as a classification problem, then it would make sense to focus on stocks that are predicted to have the highest probability of going up. In either case an appropriate evaluation metric is precision. In trading, we don't have to trade every stock but can focus on the stocks we think are worth trading. Stocks that are predicted to go up the most are good candidates for stocks that are worth trading. Using the precision evaluation metric, we can evaluate only on stocks we think will be the best performers over the 20 trading-day horizon selected. We will do evaluation in 2 ways, firstly, precision for stocks that are predicted to go up and then precision for the top 5% of predictions. Our expectation was that, in general, the top 5% of predictions will achieve a higher precision than the precision for all stocks predicted to go up. We can consider precision as our technical metric. Of course, in trading a key consideration isn't just whether stocks go up but how much they go up or down. Our second approach to evaluation will be to measure the average return of all stocks that are predicted to go up. We will also measure the average return of the predicted top 5% of stocks. Again, our objective is to see the average return for the predicted top 5% of stocks being higher than the average return of all stocks and higher than the average return of the subset of all stocks that are predicted to go up. Here we assume that more than 5% of stocks will be predicted to go up. (This assumption may not always hold in exceptional market conditions.) We will consider the average return for the selected stocks as our business metric. As a naive benchmark we will calculate precision and average return in the case that we simply and unrealistically predict that all stocks will go up every day.

Process

For this project we have defined the top stocks as the top 5% of predictions. Stocks that go up the most may not be the same stocks that have the highest probability of going up. In Finance it is quite often observed that stocks that are perceived to be low-risk may rise gradually with a high probability that they will keep rising gradually whilst stocks that are higher-risk or higher-growth may go through much more dramatic periods of price rises. We decided to model this principally as a regression problem to look for the stocks that would go up the most, but to also monitor precision with a kind of informal constraint that precision should at least match the benchmark precision. Matching the naïve benchmark in this case on the test set was not trivial because the UK stock market rose dramatically in the second half of 2020 and

the first half of 2021. The benchmark precision on the test set for predicting that every stock would go up every time was 0.584 whereas on the train set it was 0.538.

Through analysis with this dataset, we have found that model performance using all or most of the features did not generalize well to the test set and failed to match the naïve benchmark. Therefore, Michael focused on a recursive feature selection and elimination approach to select a subset of features that would perform better on both the train and test sets. This recursive feature selection/elimination approach was a bespoke piece of coding for this project. Starting from a very small initial selection of features based on domain knowledge and experience, the process allows the possibility to automatically add or remove features (so long as removing doesn't mean zero features are selected). The initial selection is the default best subset of features. Iterating through all features in the data, each feature is added to the set if it isn't already there or removed if it is in the current best set from the current best set. Using this new trial feature subset, the model is run again. If the model scores better on the evaluation metric in train and test the new trial feature subset becomes the new best set, otherwise the current best set is unchanged. Either way the process moves on to the next feature and adds/removes it from the current best set to make a new trial set. The process can iterate through all the features more than once because a feature that was useful earlier may no longer be useful based on features later added to the best set of features, or a feature may become useful in conjunction with new features in the set where it wasn't originally useful to the model. This feature selection technique was different to what had been envisaged in Milestone I where the approach was to look for features that were correlated to the target with statistical significance of $p < 0.01$. The reason that this approach was not used in Milestone II was due to the high levels of collinearity among the features.

Results

Running models with all features we found that the results could not match benchmarks and were overfit on train in ways that did not generalize to test. This is likely due to the issue of collinearity mentioned above. More evidence of this can be seen in the unsupervised section which illustrates how the majority of variance in the entire dataset can be captured with just a few principal components from over 1000 original features. As a result of this collinearity of features we employed recursive feature selection and experimented with many different models. We predominantly treated this as a regression problem attempting to predict the percentage price change from the date of the row in the training data to the date of the future price. However, the relevant notebook was set up with the option to treat this as a classification problem instead. In that case, we were predicting whether the stock will go up over the next 20 trading days. Where we treated this as a classification problem, we did see precision in the test set consistently higher than the benchmark and generally tending to be higher than when we modelled this as a regression problem. However, ultimately, we are more interested in the average return of the top predictions than we are in the average precision. In the below chart we see a summary of the results achieved on the test set with various models and modelling approaches.

ID	Model Family	Model Type	Benchmark Return	Return	Top 5% Return	Benchmark Precision	Precision	Top 5% Precision	Count of Features
1	Regressor	RandomForest	2.21%	2.21%	9.83%	0.584	0.584	0.606	17
2	Regressor	BayesianRidge	2.21%	2.37%	7.48%	0.584	0.588	0.653	17
3	Regressor	Lars	2.21%	2.33%	6.69%	0.584	0.583	0.580	25
4	Regressor	RandomForest	2.21%	2.51%	6.56%	0.584	0.590	0.676	10
5	Regressor	RandomForest	2.21%	2.21%	6.55%	0.584	0.584	0.630	5
6	Regressor	Linear	2.21%	2.37%	5.94%	0.584	0.584	0.591	45
7	Regressor	PassiveAggressive	2.21%	2.75%	5.11%	0.584	0.573	0.529	6
8	Regressor	kNN	2.21%	2.61%	4.96%	0.584	0.600	0.604	16
9	Classifier	LogisticRegression	2.21%	2.84%	2.93%	0.584	0.634	0.650	12

10	Regressor	MLP	2.21%	2.25%	2.81%	0.584	0.590	0.565	8
11	Classifier	RandomForest	2.21%	2.23%	1.60%	0.584	0.592	0.588	13

This table is ordered by the return of the subset of stocks that had the highest 5% predicted returns. The purpose of this project is not to implement a full trading strategy but rather to establish whether and to what extent machine learning approaches are able to detect meaningful signal to predict future stock market returns. The random forest regressor model featureset with ID 5 in the table above was selected by hand based on domain knowledge and trial and error. This process led to the decision to automate a feature selection and elimination approach. Run on a variety of classification and regression model types this approach consistently only selected a small subset of the available features. The range across all the optimization runs was to select between 6 and 45 of the 1000+ available features. These features were selected where the addition (or removal) of a feature led to an improvement in train and test return for all stocks predicted to go up. We then experimented with the case where features were added (or removed) from the best feature set if they caused the average return to go up on train and test for just the top 5% of predictions. This is a more aggressive form of tuning because the size of the test set effectively becomes a lot smaller and as such the risk that the feature selection has been tuned to the test set (effectively overfit) increases. During this process Random Forest models appeared to be doing well so we compared the Random Forest results using these 2 types of automated tuning. Model ID 1 represents the more aggressive tuning where features are selected or eliminated based on the average return for the top 5%. This scored better on the test set than feature selection/elimination based on the average return of all stocks predicted to go up. Another model that did well on the test set was the BayesianRidge approach. However, we did not highlight that to the same extent in our analysis of the performance on train and test because it did not perform as well on the train set seeing a return of 3.59% compared to the random forest models which tended to perform better on train than test. Whilst this is a considerable improvement over the benchmark it is not on the same scale as the 7.59% seen on the test set and as such our expectations for how this would perform on the validation sets were lower than for the random forest regressors which tended to have train scores slightly higher than the test scores on this metric. We identified important features as features that were picked in the recursive feature selection approach by different models and on different settings within the feature selection / elimination approach. Across 9 different runs we found that the opening price of a stock was the most important feature being selected in 5 of the 9 selection runs. A technical feature ‘volume_cmf’ was the next most popular being selected 4 times. There was a tie for third with 5 features being selected three times each. These were ‘SR’ which is a proprietary feature from Stockopedia, it is mostly based on fundamental information but also partly on technical information. Two of the others selected three times were fundamental features ‘ROCE_TTM_pct’. This stands for Return on Capital Employed and is an indicator of how profitably the business can reinvest in itself which is a well-known indicator of Quality of earnings which is a widely used factor in investment decisions within the industry. Also selected three times was ‘Price_to_FCF_TTM’ which is a ratio of price to free cash flow. This measures the value of a stock. Lower price to free cash flow, other things being equal, is better. This is an example of the Value factor which is also used commonly in finance. Typically in the industry Price to Earnings is more widely quoted but earnings can be more easily manipulated than free cash flow and it is interesting that the model preferred the indicator that is harder to manipulate.

We selected some of the top performing models to run on the validation set. The results were not quite as good as on the test set but were still very strong:

ID	Model Family	Model Type	Count of Features	Concurrent Validation Benchmark Return	Concurrent Validation Top 5% Return	Concurrent Validation Benchmark Precision	Concurrent Validation Top 5% Precision
1	Regressor	RandomForest	17	2.14%	5.97%	0.577	0.558
2	Regressor	BayesianRidge	17	2.14%	6.10%	0.577	0.630

3	Regressor	Lars	25	2.14%	5.30%	0.577	0.563
4	Regressor	RandomForest	10	2.14%	4.84%	0.577	0.600
5	Regressor	RandomForest	5	2.14%	5.82%	0.577	0.614

The validation process was split into 2 parts and neither of those were run until the end of the project in the final week after the results for train and test were collected and collated. The first validation set was chronologically concurrent with the test set. The second was chronologically subsequent to the test set and provided a true out-of-sample test for the model. For this out of sample testing we ran some of the models that performed best on the test and original validation sets. Unfortunately the results on the true out of sample test were less good.

ID	Model Family	Model Type	Count of Features	Final Validation Benchmark Return	Final Validation Top 5% Return	Final Validation Benchmark Precision	Final Validation Top 5% Precision
1	Regressor	RandomForest	17	1.89%	1.02%	0.643	0.547
2	Regressor	BayesianRidge	17	1.89%	1.27%	0.643	0.644
3	Regressor	Lars	25	1.89%	0.88%	0.643	0.506
4	Regressor	RandomForest	10	1.89%	-4.42%	0.643	0.000
5	Regressor	RandomForest	5	1.89%	-0.63%	0.643	0.625

There are a few observations we can draw from this as we conduct a failure analysis. Firstly, there is a long gap between the end of the training data in June 2020 and the final validation data from July-August 2021. Remember the time horizon we are interested in is forecasting stock performance 20 days into the future. Here all the training data is over a year old which makes the task more difficult. Retraining with stock data up to June 2021 to forecast July-August 2021 did lead to some improvement in the above results. Notably, the BayesianRidge model did just beat the benchmark on the top 5% return at 1.92% and beat the benchmark with precision at 0.672. The precision result here is more impressive because it is well ahead of the benchmark. However, we should not read too much into this because the return of 1.92% is only a very marginal improvement over the benchmark. We should also note that the final validation window is quite short so the number of records in this dataset was only just over 5,000 compared to over 30,000 in each of test and validation plus over 220,000 in the original train dataset rising to over 280,000 when retraining included data to June 2021. The smaller size of the final validation set reduces our confidence in the results. This again means we should not read too much into these results although clearly they do not match the results seen on the test set and the earlier validation set that was concurrent to the test set. Finally we note that model drift can be an issue in any type of time-series prediction. Combining this with the notoriously low signal to noise ratio in Finance may explain why the results on the final validation set did not replicate the earlier results. Indeed it would hardly be realistic to expect to achieve in practice a 6% return (as seen in the validation data) month after month for long periods. Even for top hedge funds that would represent an absolutely exceptional performance unlikely to be repeated since this would constitute a return of just over 100% in 1 year. Some further comments on failure analysis are included in the conclusion section.

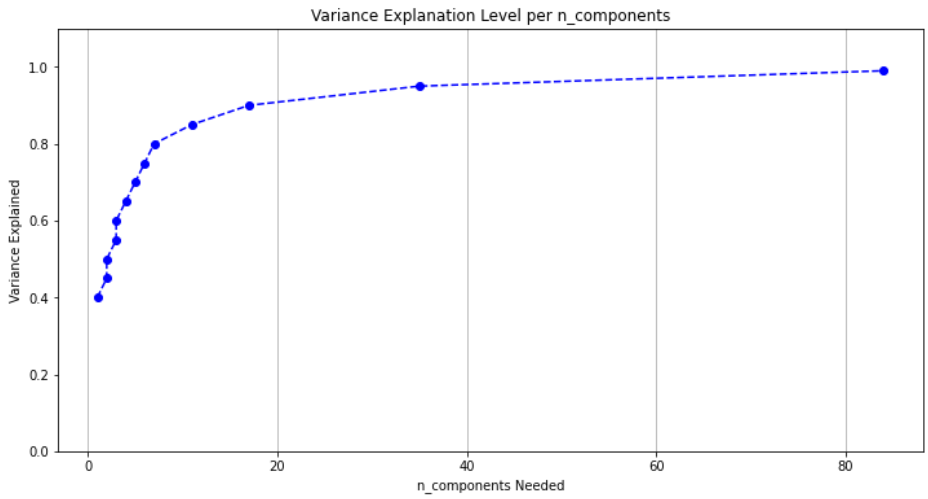
Unsupervised Learning

Principal Component Analysis

The unsupervised learning portion of this project specifically focuses on clustering the dataset into usable groupings that can eventually be re-run through the supervised learning portion to better predict stock movements. The dataset contains thousands of features, and while this should lend itself well to

describing stock movements, due to the nature of how the features were developed, we had a hypothesis that many would be highly correlated. Because of this, we decided to leverage Principal Components Analysis to reduce the featureset into something more concise, while still retaining meaningful explanatory value. This dataset will be using the most recent day of the X_train dataset only at this time. In future iterations of the project, we may explore the movement of the clusters over time.

The X_train dataset was leveraged and joined with the y_train set so that returns could be potentially used in groupings if they were significantly significant, since target variables are not used in this particular unsupervised model. The first step of the PCA included scaling the data, as per our standard preprocessing procedures. The first question to answer with PCA is how many PCA components we want to use, or, alternatively, how much variance do we want to be able to explain. So our first task was to get a rough understanding of how much marginal gain each additional component can get for us. We generated a PCA object that found the lowest amount of variance that can be explained, then incemented up 5% to 99%. We then re-ran the model on each variance increment to find the minimum number of components that would be needed to explain that amount of variance.

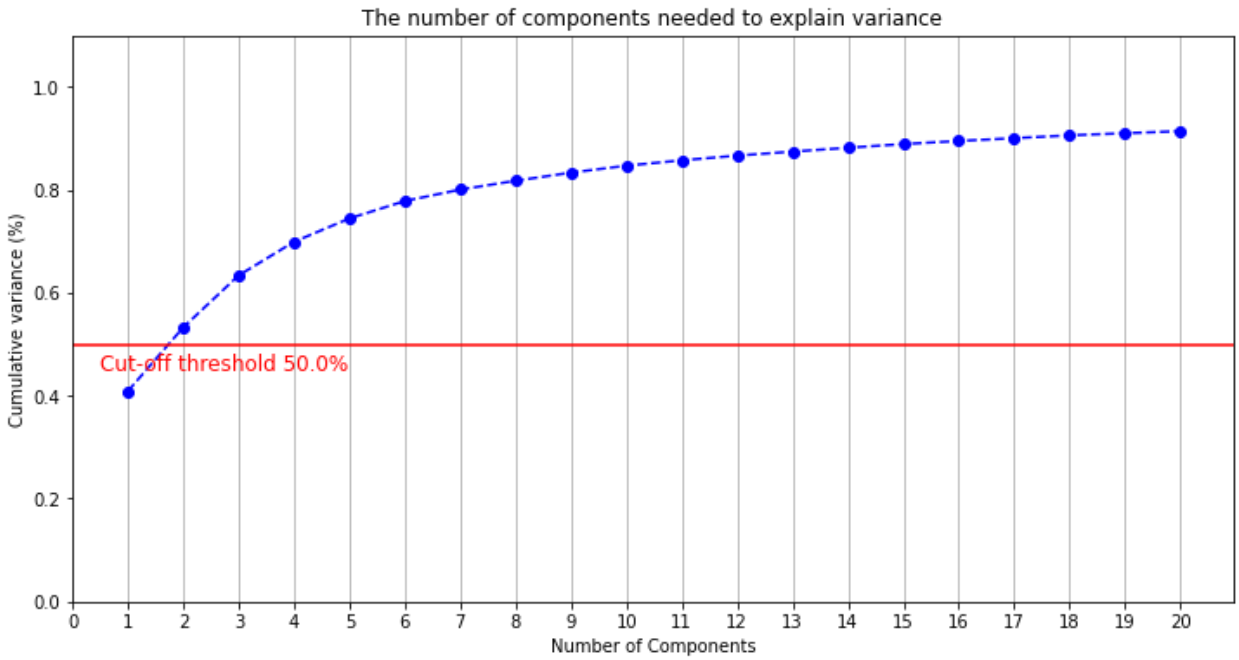


	variance	pca_components
0	0.40	1
1	0.45	2
2	0.50	2
3	0.55	3
4	0.60	3
5	0.65	4
6	0.70	5
7	0.75	6
8	0.80	7
9	0.85	11
10	0.90	17
11	0.95	35
12	0.99	84

So, as expected, to be able to explain more variance, we would need to include more PCs. That said, the marginal increase in informational value per PC decreases significantly as the variance increases. We can actually garner more than 80% of the variance explanation in less than 10 principal components. The amount of information added per PC after that is so small that it causes the PC count to begin to increase exponentially. So, including more information and PCs, we would increase complexity and dimensionality for very little informational return, or, in other words, the material information of the principal components have rapidly diminishing returns.

This indicates that there is quite a significant amount of correlation in the 1000+ features that the dataset contains. We suspected this, given Milestone I research, our familiarity with finance and trading, and the general nature of the features created (varying combinations of inputs for similar calculations), but this illustrates clearly that many of the similar features can likely be condensed into a single principal component.

Now, the next task is which variance or principal component number we want to use. There is no right answer here, but our business use case for this is specific to future projects in which we may wish to leverage this work. In finance, there is a significant risk of overfitting to a dataset, so we are comfortable in this project with sticking to explaining about 50% of the variance, which should result in two main principal components on which to focus. A future consideration may be adding more if, in a supervised learning re-run on the clusters, we are finding that accuracy and precision are lacking and may benefit from clustering on more PCs.

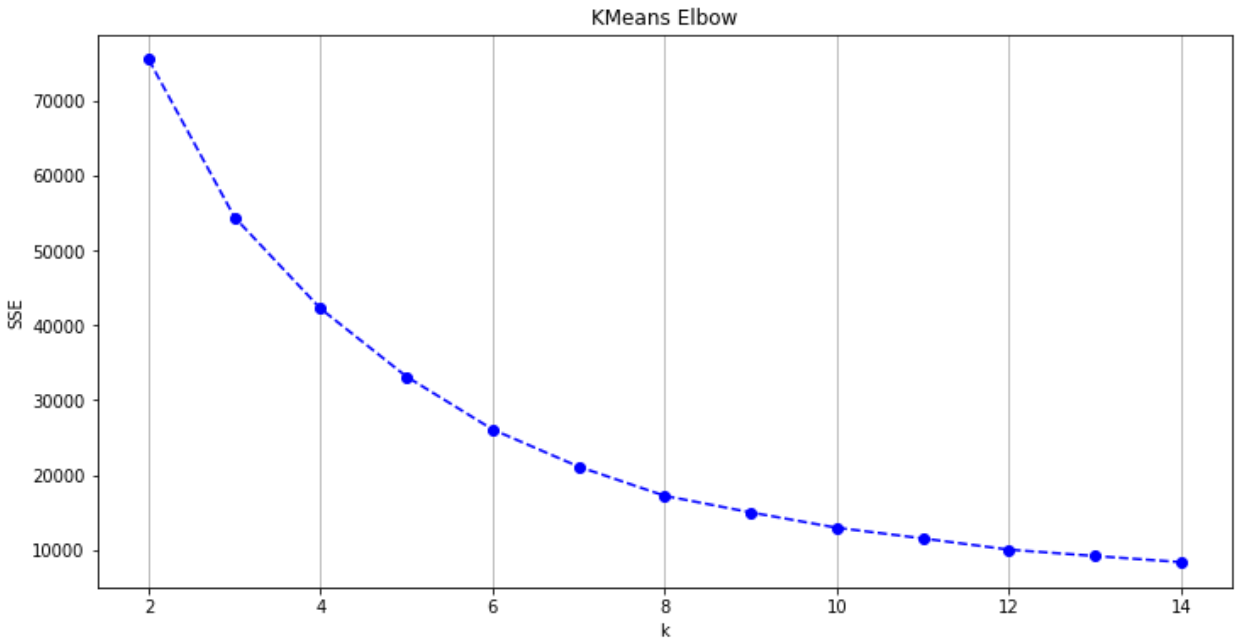


With a defined number of PCs to work with, we run the PCA model on our X_train_numeric_scaled with our n_components=2. We find that we can explain 53% of our variance with just these two principal components. We can now determine that the two original features that have the largest coefficients in the linear combination, or had the strongest effect on PCs 1 and 2, are 'APO_f2_s55', 'MACDEXT_macdhist_f2_s21_sig55', respectively.

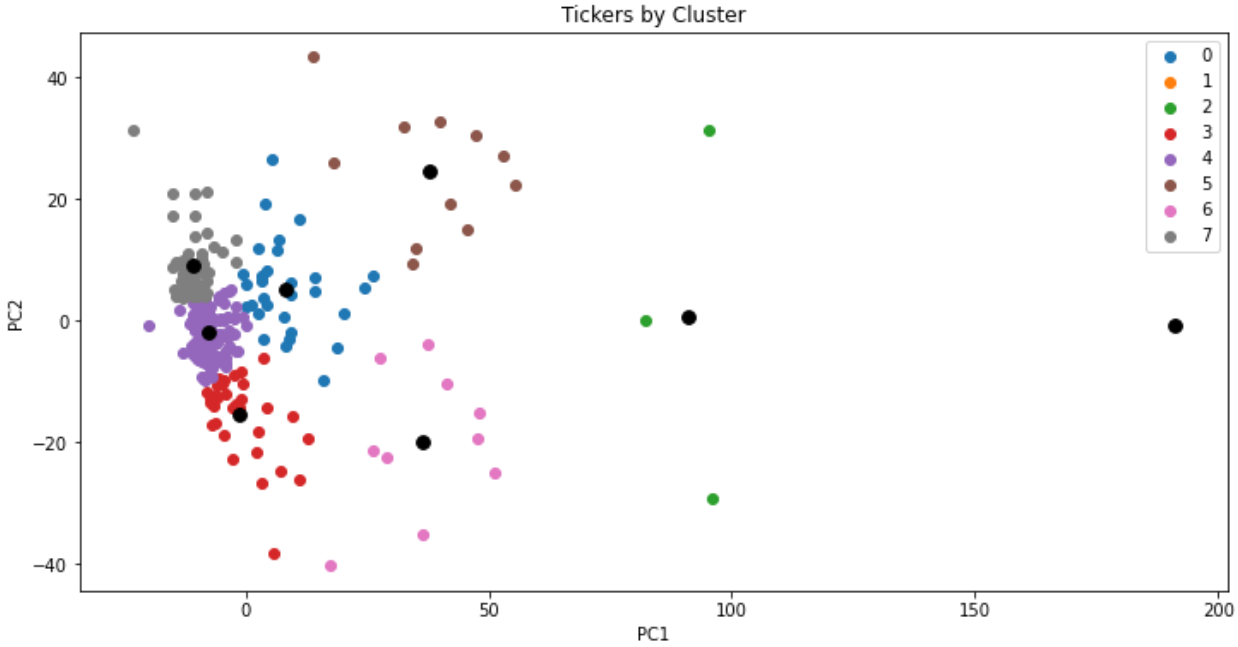
APO is our Absolute Price Oscillator, and MACD is our Moving Average Convergence Divergence. The “f” represents the fast lookback window, and “s” represents the slow. So, for 'APO_f2_s55', there is a rolling calculation looking back two observations, and a slow one looking back 55 observations. When the fast one crosses, it is an indication that the price momentum is changing. If it goes up, that indicates positive momentum, if it goes down that indicates negative momentum. Same with MACD, however there is one more derivative of moving average lookback with a “signal” (sig) parameter. The interesting thing about this insight is that these are both momentum signals, so it appears that most variance can be explained by technical indicators, rather than fundamental ones, which we also confirmed by pulling the next few PCs and discovering that three of the top 7 PCs’ heaviest loadings (80% of variance explanation) were all technical indicators, and three of them were MACD-related. Interestingly, this is corroborated by many algorithmic trading strategies in our professional practice that successfully leverage MACD.

KMeans Clustering

With two principal components that meaningfully explain more than half of the variance in our data, we used these two PCs to run KMeans clustering on the dataset. The next question we had to answer was how many clusters do we want to select. To determine this, we ran the KMeans algorithm on all n_clusters from 2 to 15, and returned the sum of squared errors. The resulting elbow looked as follows:



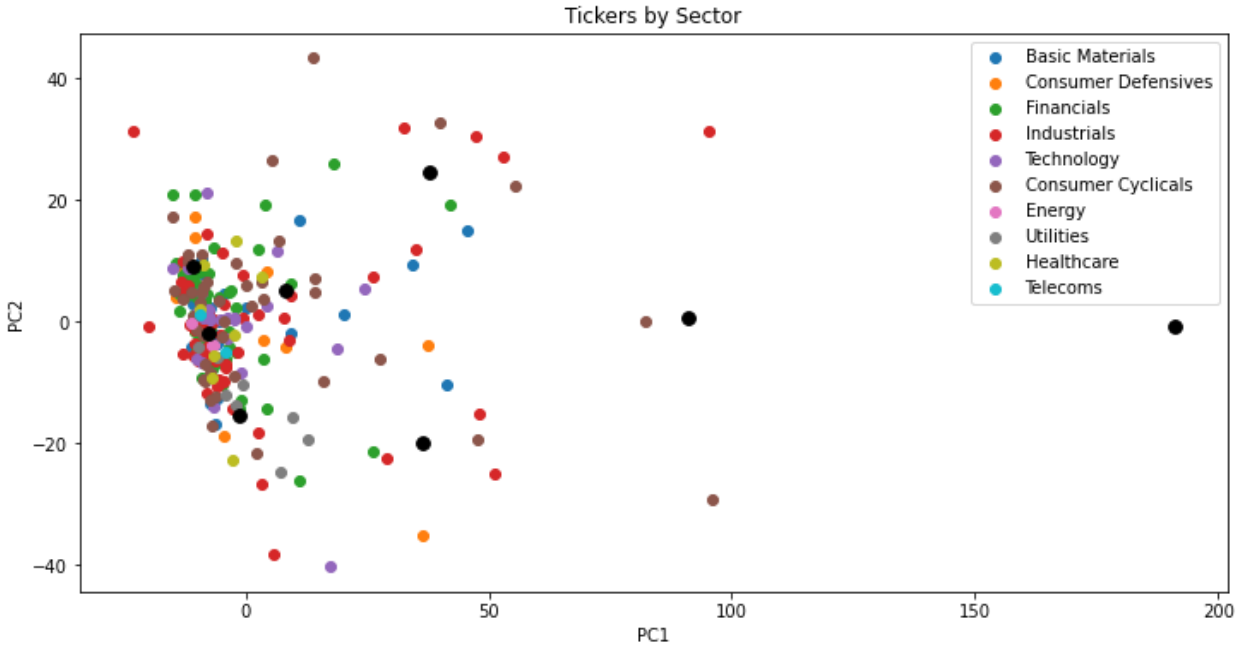
One could argue that the marginal decrease in SSE tapers off anywhere between 6 to 8 clusters. The “elbow” is more of a steady slope here. Due to the recursive nature of this particular clustering method, we tried out clusters 6 to 8 recursively until we found that 6 resulted in heavy concentration into two clusters, but we were able to break that up a little more into more clusters with more meaningful insights with 8. With the business case for these clusters being leveraging the clusters for future supervised learning projects and algorithms, it is more useful to have smaller, more specialized clusters as opposed to larger, more generic ones. Smaller/more similar clusters allow for more tailored algorithms to be applied.



(Note here that Cluster 1, furthest to the right, is its own cluster, as it is an outlier.)

When compared to traditional sectors, we can see that there is a clearly defined difference here between commonly defined sectors and the clusters we generated. We do not see one particular cluster that is

exceedingly concentrated with one particular sector, nor any particular sector that has similar variance explanation in the principal components. This indicates that much of the trading variance may actually be completely independent of what line of business a company is grouped in.



There are two “outlier” clusters here. In traditional clustering, one may choose to eliminate outliers. We chose to leave them in, noting that they were still their own clusters with k=6 and k=8 (the additional two broke up the more concentrated left portion of the plot). This is because in Finance, we often find that the outliers can hold more meaning when it comes to solving for maximizing alpha.

As we dove into Cluster 1, the standalone ticker to the right of the plot, the ticker is GAW, which upon further research has been a huge momentum stock and seen a 2000%+ increase in the last five years. The three tickers in Cluster 2 also indicate significant momentum stocks, with abnormally high returns ranging from 55-365% in the last five years. This is an example of how outliers in finance can occasionally indicate a point of interest as opposed to noise.

Use Cases and Applications

The purpose of clustering stocks is to refine specific trading strategies to different clusters, based on their attributes. So, once we have our clusters, we can highlight some aggregations, or overlays, that may better describe these clusters.

If we take a look at something meaningful to a strategy, say, a value stock strategy, we may want to look and see if any particular cluster lends itself well to that. One way to find undervalued stocks is to look for Price-to-Book values. A high P/B indicates that the price of the stock is worth more than the actual value of the firm, for example, a P/B of 3

	ticker	Price_to_Book_Latest			
	count	min	max	mean	std
cluster					
0	30	1.110352	16.500000	5.187500	4.004576
1	1	11.296875	11.296875	11.296875	NaN
2	3	4.031250	8.421875	5.699219	2.376880
3	33	0.000000	8.507812	2.521484	1.836484
4	111	0.000000	132.625000	4.468750	13.670880
5	11	0.000000	17.500000	6.667969	5.560518
6	10	1.589844	16.093750	5.832031	4.332329
7	54	0.000000	34.187500	4.457031	5.954329

indicates that a stock is trading at 3x the firm's book value. A value less than 1 indicates that it is trading at a discount. If we want to find a cluster that may be good for value stocks, we will want to find the cluster with the lowest distribution of P/B, which in our dataset is Price_to_Book_Latest.

A cluster that lends itself well to this may be Cluster 3. There are a handful of tickers in this cluster, and it maintains a low mean P/B, and its max P/B is not concerningly high. Conversely, Clusters 0, 5, and 6 have higher P/B ratios and reasonably low standard deviations, indicating higher prices and possible concentration of growth/momentum stocks. We will ignore Clusters 1 and 2 for now as they only have one and three data points, respectively.

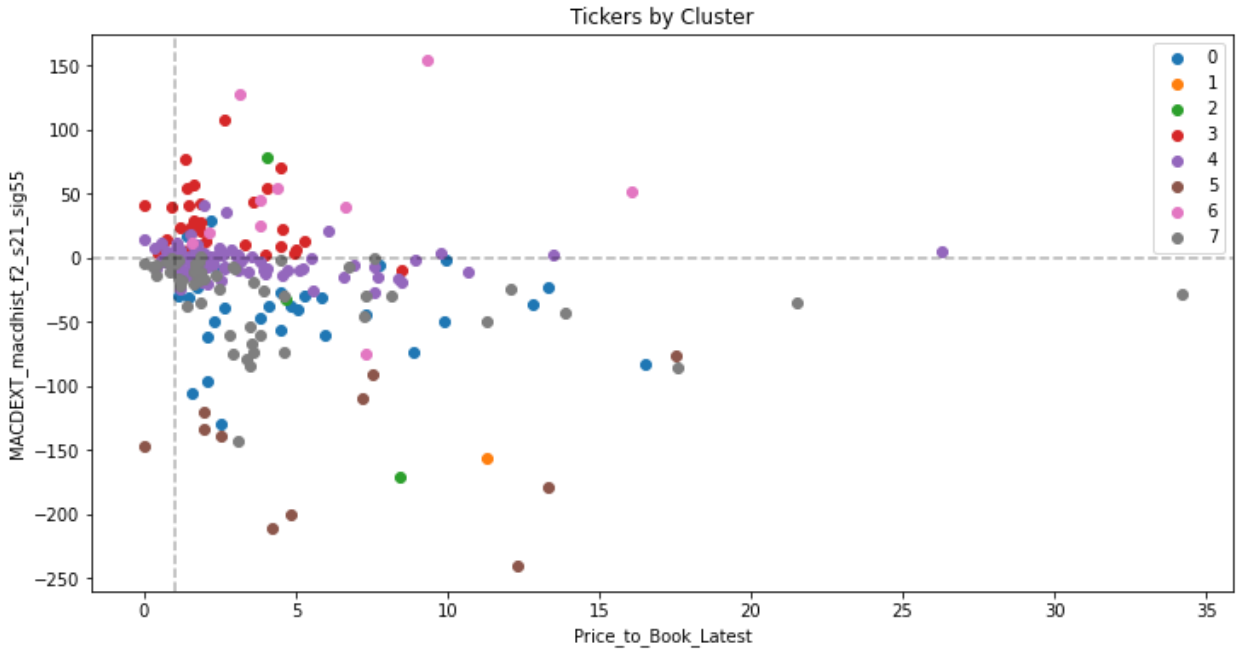
	ticker	20d_returns			
	count	min	max	mean	std
cluster					
0	30	0.757320	1.110207	0.972106	0.080002
1	1	1.021752	1.021752	1.021752	NaN
2	3	0.877183	0.995377	0.955388	0.067733
3	33	0.847154	1.391380	0.964712	0.102883
4	111	0.667883	1.224324	0.953622	0.095279
5	11	0.896605	1.082527	1.000751	0.058104
6	10	0.884060	1.102776	0.971033	0.064400
7	54	0.687085	1.172728	0.956906	0.095221

We could also look at returns to identify growth stocks using a similar method. If we find a cluster with a high mean of return and low standard deviation, that may indicate a cluster that has more consistent returns over 20 days across the cluster, and the higher the returns the more potential for growth stock strategies. When analyzing the data, it looks like Cluster 5 fits this bill, and Cluster 6 could be another possibility. They both have some of the higher means, as well as lower standard deviations, indicating that the cluster may have a propensity to return similar alpha.

Another possible way to look at the clusters is seeing their sector distribution. Even though we determined from our sector plot that it didn't look as though sectors clustered naturally anywhere, it is still possible that certain clusters may have a heavier, material weighting in a certain sector that lends itself to news screening algorithms, or some sort of strategy that would leverage an influential sector's movements. For example, if Cluster 7 is heavy in financials, we may want to make more considerations for interest rate announcements.

	sector_pcts									
	Basic Materials	Consumer Cyclicals	Consumer Defensives	Energy	Financials	Healthcare	Industrials	Technology	Telecoms	Utilities
cluster										
0	13.3%	30.0%	10.0%	0.0%	10.0%	3.3%	20.0%	13.3%	0.0%	0.0%
1	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2	0.0%	66.7%	0.0%	0.0%	0.0%	0.0%	33.3%	0.0%	0.0%	0.0%
3	9.1%	15.2%	3.0%	0.0%	18.2%	3.0%	27.3%	6.1%	0.0%	18.2%
4	10.8%	10.8%	3.6%	2.7%	28.8%	3.6%	23.4%	12.6%	1.8%	1.8%
5	18.2%	27.3%	0.0%	0.0%	18.2%	0.0%	36.4%	0.0%	0.0%	0.0%
6	10.0%	20.0%	20.0%	0.0%	10.0%	0.0%	30.0%	10.0%	0.0%	0.0%
7	13.0%	20.4%	9.3%	0.0%	33.3%	3.7%	13.0%	7.4%	0.0%	0.0%

We can also take a look at other indicators by plot. For example, the following plot looks at MACDEXT_macdhist_f2_s21_sig55 against Price_to_Book_Latest, a technical indicator and a fundamental indicator. A negative MACD Histogram score indicates that the stock is trending down, thus possibly alluding to a bearish signal. High MACD Histogram scores indicate bullish signals. High P/B indicates growth stocks, and a P/B less than 1 indicates a possible value stock.



Note: We have removed two data points from Cluster 4 that are significant outliers on the x-axis to get a better visualization of the cluster distribution.

This plot indicates that Cluster 5 likely contains stocks that are priced at a premium and exhibiting possible bearish signals, and thus may be good for a shorting strategy. Cluster 0 is less bearish, but similar sentiment. Cluster 3 seems to be concentrated more in bullish signal territory, as well as being more fairly priced. Clusters 4 and 7, while leaning slightly bullish and bearish, respectively, are harder to decipher given their large size, and in further analysis could potentially be broken up into value and growth stock strategies within the clusters.

It may also be useful to leverage the .describe() function on a single cluster to figure out what strategies may lend themselves best to a particular cluster based on which features have unique or prominent values. The direction of the research can really be leveraged both ways (cluster to strategy, or strategy to cluster) depending on the algorithms to be built.

Conclusions and Possible Next Steps

Supervised Learning

The single most instructive thing about this project from a supervised learning perspective was that simply feeding a model over 1000 potentially relevant features does not guarantee a good result. In fact, given that many features the model may not know what to do with all of them. This led us to employ recursive feature selection strategies that improved performance on train and test sets as well as a validation set that was concurrent with the test set. These results were not replicated on a final validation set from a later time period. Potentially the method of selecting features was slightly prone to overfitting, however, since the results were fairly robust on the original validation data, the main issue here appears to be model drift with changing conditions in the market in the final validation set. In Finance the signal to noise ratio is known to be low and it is well known that market conditions change over time. This was particularly exacerbated by the pandemic which led to huge upheaval in financial markets. Remember, this is the era where meme stocks behaved vastly differently to how the models of some of the most sophisticated hedge funds predicted they would. This led to billions of dollars of losses for some of those firms. Taking some of the time series nature of the problem away (by having the test and validation sets concurrent) we found substantial improvements over benchmarks through recursive feature selection but when we include the full time series nature of the problem in the final validation set, those models become less effective. Michael has previously worked on Facebook Prophet and VARIMA time-series models for

stock prediction and did not feel it was beneficial to repeat that work in this Milestone project, as the aim is to do something new. That is why some more specific time-series approaches were not attempted. Possible next steps would be to implement reinforcement learning approaches to the supervised learning research and development aspect of this project. There would also be scope to experiment with implementing some more of the findings from the unsupervised learning portion of this project to a future supervised learning stock market prediction project. Another thing to consider for future work is to look at different prediction windows, for example over 60 trading days or 90 trading days rather than the 20 we selected for this project. The 20 day window was a legacy of Milestone I where it had worked well for finding features highly correlated with the target. To make the validation approach more robust it may be useful in future projects to use time series split in sklearn and repeat the validation over multiple time windows (whilst still keeping a holdout validation set until the end). These models are not making decisions directly about individuals in the same way that some models decide who will get sent to prison and who won't, so the ethical issues are not as direct. However, there could be ethical concerns if AI solutions are touted as being reliable ways to make money and mis-sold with inadequate explainability, or inadequate explanations that the future may not be like the past. Based on the results in the original test and validation data it is possible that investors could be convinced to part with money and end up being disappointed that the portfolio returns in practice do not match what was advertised based on validation results.

Unsupervised Learning

From the Unsupervised Learning perspective, we can conclude that sectors may not be the best or only way to group stocks, as it seems there is significant reason to believe that movement is not necessarily determined only by sector. With this information, we can better cater our strategies to alternative clusters. The next steps with this clustering, which is out of scope of this project at this time, would be to re-run our clusters through the Supervised Learning portion of the project and determine if on a smaller scale the y_20d target can be more precisely calculated, given that the sets have more relation and possibly less noise than the entire dataset itself.

Additionally, future iterations of this work would likely include the movement of clusters, or possibly tickers within clusters, over time at different dates. We started with just the most recent date of the test set, but as companies report and prices change, clusters may shift as well. We found that the analytical methodologies and considerations that would go into that may be an entire ad hoc project on their own and were out of scope of this project. We would also like to test out other clustering methods, such as hierarchical clustering, however due to time constraints, the recursive nature of the project, and refactoring, were unable to do so on this iteration.

Statement of Work

Our team utilized a joint effort, capitalizing on each member's skillset and strengths, and communicating steps throughout. Michael Penrose, with the background of the Milestone I work that acted as a springboard for this project and experience working in investment banks, led the Supervised Learning effort as well as collated the dataset which included data scraping from the Stockopedia website, loading that data to a database, reading relevant information from the database and feature engineering to generate over 1000 features. Allison Bergmann, with a background in trading, led the Unsupervised Learning effort. The utils library was generated by both team members, and code specific to each project section was generated by the lead team member. All analytical methodologies and decisions were discussed, as well as all final results reviewed, by both parties, and the final written report was written and reviewed by both parties.

Disclaimer: This report is for educational and research purposes only and is not intended to be used as an official investment recommendation. All investment/financial opinions expressed in this document are from the personal research and experience of the project owners and are intended as educational material. Although best efforts are made to ensure that all information is accurate and up-to-date, occasionally unintended errors or misprints may occur in the data. We make no guarantees of financial returns based on this analysis. It is important to do your own analysis before making any investment based on your own personal circumstances. All financial decisions should be made with the help of a qualified financial professional. At this time, Michael Penrose owns shares of GAW as of 09/21/2021, and neither author has any known conflict of interest.

References

Current Stock Information and Future Stock Prices: Investigating the Relationship, University of Michigan Masters in Applied Data Science Milestone 1 Project, Jan 2021, Michael Penrose and Damon Clifford. (*For ease of reference this report will be included with the final submission*).