SIADS 694/695 Milestone 2
Team #3 — Erik Lang, Liu Jason Tan, and Vlad Bardalez

S&P 500 Stock Performance Forecasting & Sector Clustering

As the markets enter yet another tumultuous period within a two-year period at the start of 2022, investors, asset managers, and the global economy gauge the current state of the market by tuning into the volatile swings of the largest stock market index by market cap, the S&P 500. Our project will use a single dataset with stock information from the constituents of the S&P 500 to examine (1) How can we use supervised learning techniques to predict the future performance of stocks in the S&P 500 given a set of commonly used features?, and (2) If the 11 economic sectors that the S&P 500 are divided into are appropriate, and whether unsupervised learning techniques suggest a different set of economic sectors?

Dataset [1]

The dataset is an aggregate of data manually collected from Bloomberg and FactSet quarterly since 2014, containing monthly data on a select constituent of stocks found on the S&P 500 index, with data ranging from December 2000 to June 2021, inclusively. Since the stock tickers were updated at the end of every quarter, retroactively, for the three-month period, the dataset does not take into account official rebalancing made by Standard & Poor's. The dataset is composed of financial data that is publicly accessible and can be commonly reported in a company's quarterly earnings reports or listed on stock exchanges. The aggregated dataset was composed in an Excel file and after some initial data cleaning and data manipulation the initial dataset is a 25 MB CSV file with more than 122,000 rows and 14 columns.

The initial features of the dataset contain columns that are needed to properly identify each stock's monthly datapoint (i.e., Date, Ticker, Name, Sector), data that was purely informational to maintain the integrity, and a set of 11 initial features. These initial features relate to characteristics that include a given stock's value, growth, price, momentum, volatility, and leverage at the end of every month. We further expanded our initial set of 11 features to 18 features, engineering them from the starting date. See **A1** and **A2** in the Appendix for definitions of the initial and generated features employed in our analysis. During our work on both portions of the project, we found a good reason to reduce our number of features based on their correlations, which we describe in the Supervised Learning section of this report. Applying this same treatment to both portions of the project, it was ultimately 14 features that were the focus of both the supervised and unsupervised learning components. See **A3** in the Appendix for a detailed description of our corresponding notebooks to each section of our report.

Data Cleaning & Exploratory Data Analysis [2]

While our initial extraction of the data from our original Excel file involved removing several unusable data points and ensuring that each feature in our initial CSV file was formatted correctly, the dataset still contained rows that contained missing information across our feature set. We applied two different procedures to appropriately fill in these missing data points. First, we found brief gaps in each stock's time series data, typically spanning between one and two months, and where valid data existed before and after the gap. In those instances, we would forward fill the missing data points with the most recent data point for each respective stock and only for features that we deemed reasonable after an in-depth assessment.

We aimed to visually explore our dataset to better understand the gaps and time periods in which we observed missing data across our feature set. This crucial step in our exploratory data

[1] See our notebook titled "Part 01 - XLSX to CSV" and "Part 02A - Additional Feature Creation"
[2] See the notebook titled "Part 02B - Feature Cleaning"

analysis yielded conclusions that we used when choosing how we represented our dependent variables in our supervised learning process. There were three features that had just over 10% of the months with missing data points; the features are: trailing EV to EBITDA, net debt to EBITDA, and trailing three-year average growth in earnings per share. We performed further visual exploration on those features to ensure we were not encountering any problems that would result in the feature's removal from our dataset. For instance, we would consider removal if the missing data points were all related to a single time period or if we were missing data for an entire sector. This visual exploration ensured that there was still a healthy percentage of valid data for each month and sector (see **Figure 1**). However, trailing EV to EBITDA and net debt to EBITDA, demonstrated that most of the missing values belonged to a single sector, Financials (see **Figure 2**).

This finding had implications in our choice of data representation for our dependent variable in our supervised learning analysis. We were originally considering two different methods of prediction in the supervised learning analysis: predicting the future performance of the stock relative to the overall index or predicting future performance relative to the stock's sector. The visual exploration of the data described above convinced us to focus only on predicting future performance related to the overall index, given the issue seen in the financial sector. After the gap-filling process, we then used a common investment industry practice of filling the remaining missing data points with the median data point of the given month and the sector the stock pertained to. This median replacement technique maintained some of the inherent structure of the features over time without needing to drop entire rows from the dataset.

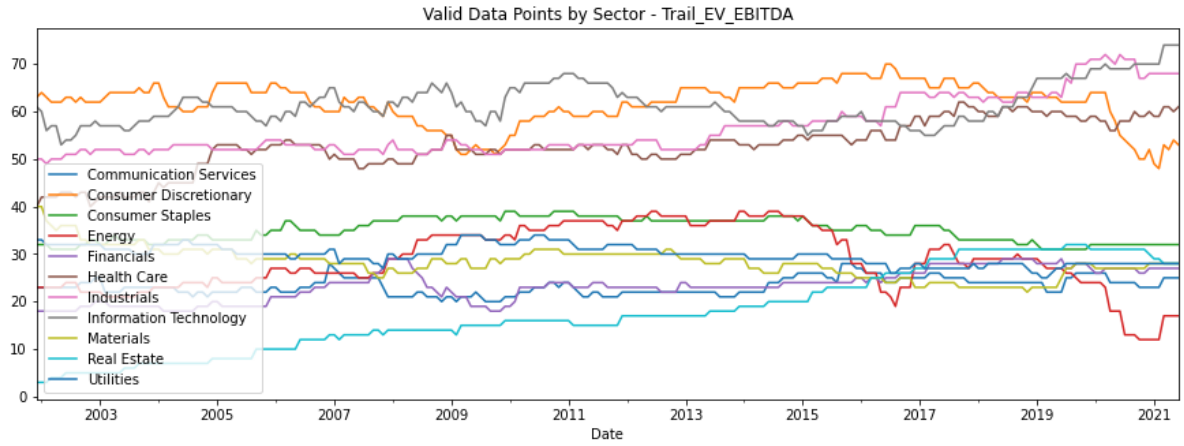


Figure 1 - Valid Data Counts Example for a Single Feature

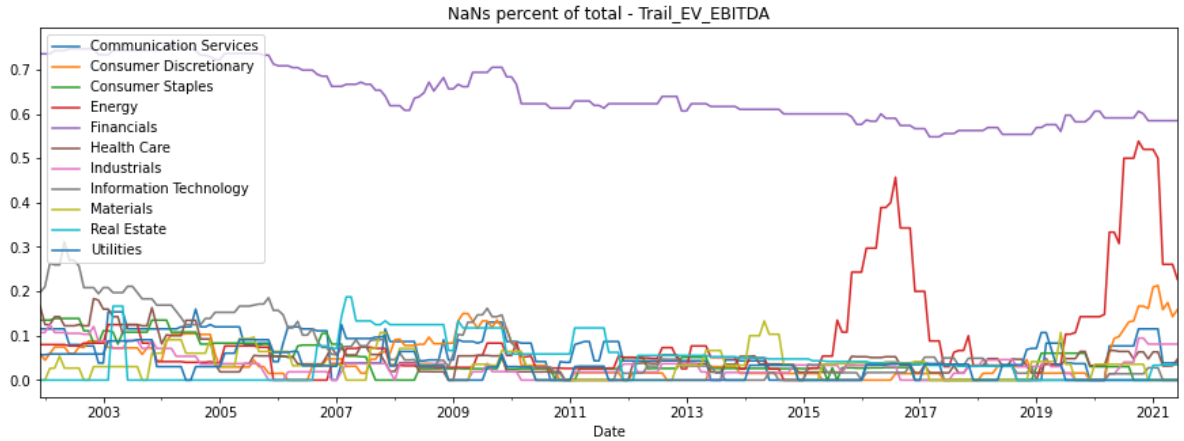


Figure 2 - Percentage of Missing Data Example for a Single Feature

Part A: Supervised Learning: Overview [3]

The motivation for the supervised portion of our project relates to a common investment industry challenge determining which stocks will perform the best and worst over a period of time. Some firms create rankings or recommendations on purely subjective opinions that are manually updated, while others try to create more systematic rankings or recommendations of stocks based on quantitative data. We are attempting to explore the latter approach, using our dataset along with supervised learning techniques to try to predict the best performing and worst-performing stocks monthly.

On top of the data cleaning we described earlier, we standardized our feature set using a custom min-max scaler that is applied on a monthly basis, and we use these standardized features in each prediction algorithm as our independent variables. Out of the numerous options[4] for prediction labels that we considered (Please see appendix **A11**), this paper will focus on the top 100 and bottom 100 performing stocks. For each stock ticker and month combination, if a stock is in the top 100 performing stocks for that month, we assign it a label of '4'. If the stock is in the bottom 100 performers for that month, the label will be '0'. All remaining stocks will be assigned a label of '2'. We favor these labels over the other alternatives we considered (Please see appendix **A10**) because it is most applicable to the finance industry; we are most interested in whether a stock will outperform or underperform other index constituents. This choice of our dependent variable also improves the performance of our machine learning models by simplifying our problem set to three distinct categories. We then combine our dependent variable with the 14 cleaned and normalized independent features mentioned above to begin our supervised learning analysis. We start by dividing our dataset into training and test sets at a 90%/10% split. We choose this particular split because our dataset has a sufficient number of observations. The test set has a total of 11,287 rows, which is more than enough data to get a representative sample across our 20 years of observations.

Supervised Learning: Methods and Evaluation

Following the train-test-split, we created a dummy classifier as our baseline. Our dummy classifier always predicts the most frequent label (in this case '2' since it contains the middle 300 stocks across the 20 years). This is a good baseline since a good machine learning model should outperform one that simply predicts the most frequent label. Our dummy classifier did quite well with an accuracy score of 0.60 and an f1 score of 0.25. However, since we have an imbalanced dataset, accuracy can be deceiving. A dummy classifier can have a high accuracy score just by predicting the most frequent label. Therefore, we also looked at the macro precision, recall, and f1. Since the macro average weights each class equally, we observe the full picture of the models' performance.

After we created the dummy classifier, we trained other classifiers using our data. These include logistic regression, k-nearest neighbors, and random forest (see **Figure 3** for the scores for all the classifiers). We chose a wide variety of classifiers so we can see if a particular type of classifier is better. We used grid search as well as randomized search to tune the hyper-parameters, using f1 as the scoring metric to optimize. We found that the random forest model has the best overall performance. Although K-nearest neighbors had the highest F1 score, the accuracy of KNN is worse than the dummy classifier, so we decided against it. The random forest model has the second highest F1 score, and the highest accuracy score, so we choose it to be the focus of our report.

Some important trade-offs include the variance-bias trade-off. We noticed this trade-off in all of our models, where complex models have high variance and low bias, and a simple model has low variance but high bias. For example, in our random forest model, when we let the number of trees increase, we have a higher variance but lower bias. This means that it can capture more of the

[3] See the notebook titled "Part 03 - Supervised Learning"
[4] See the notebook titled "Part 02C - Supervised Y Variable Options"

training data but does not generalize well to the test data, also known as overfitting (Collins-Thompson). On the other hand, when we have a small number of trees, it has a much lower variance but higher bias. This causes the models to be too simple to capture the underlying pattern, a phenomenon known as underfitting (Collins-Thompson). The purpose of using grid search and randomized search is to tune the hyper-parameters in a way that balances this variance-bias trade-off, so the model is neither overfitting nor underfitting.

Model	Accuracy	Precision	Recall	F1
Dummy	0.60	0.20	0.33	0.25
Multinomial Naive Bayes	0.60	0.20	0.33	0.25
Gaussian Naive Bayes	0.60	0.49	0.41	0.41
Quadratic Discriminant Analysis	0.61	0.51	0.41	0.42
Logistic Regression	0.63	0.60	0.43	0.43
Ada Boost	0.64	0.61	0.43	0.44
Decision Tree	0.63	0.57	0.45	0.47
Random Forest	**0.65**	**0.62**	**0.48**	**0.50**
K-Nearest Neighbors	0.58	0.50	0.50	0.50

Figure 3 - Classifiers Scores, sorted by f1 score

Another trade-off that we noticed was in feature engineering. When we first started this task with 18 features, we noticed that several of them were highly correlated (see **A4** and **A5**). This caused our model to perform poorly. When we reduced our features to the 14 final features, we noticed a significant increase in model performance. These 14 features are semantically important in the finance industry and were significantly less correlated to one another. We also hypothesize that if we removed more features, we would not have enough features to build a good model. Similar to the variance-bias trade-off, we had to find an optimal number of features to optimize the models. See **A6** for the feature importance values for our 14 variables.

With these trade-offs, we can see that our results were sensitive to our choice of hyper-parameters and features. To get the best results, we needed to find the optimal features to use as well as the best hyper-parameters. We also noticed that our results were also sensitive to the dependent variable we selected. The less unique label an output had, the better the results. Our top/bottom 100 variable performed the best, followed by equally divided quintiles and deciles. This is related to having a simpler model since we tasked our model to predict three classes instead of five or ten classes.

Supervised Learning: Failure Analysis

We used the random forest model to find out which data points are particularly hard to predict. Since the random forest is our best-performing model overall, we chose this model to explore where our predictions were wrong and try to understand why. The most striking conclusion was that we found certain features, specifically the features that relate to a stock's valuation (cheap versus expensive) tend to have a trend of being very cheap when our model makes a bad prediction (See **A7** in the Appendix for the average value of wrong predictions). We discovered this anomaly when looking at the instances where our model was wrong and taking an average of the normalized feature. These low values indicate that the incorrect predictions are very cheap stocks, which tend to be companies that are either broken and will see a further decline in share price or could fix themselves and provide handsome returns to shareholders. Thus, this group of stocks that our

model is getting wrong have very binary outcomes, so it is not surprising that our model struggles to identify them. For additional charts and commentary on how we explored the incorrect predictions of our model, please refer to see Appendix **A9**.

Part B: Unsupervised Learning: Overview [5]

The motivation for the unsupervised portion of our project relates to the 11 sectors that the S&P 500 uses to classify its constituents. The reader might recognize sectors such as “Information Technology” or “Financials”, and others as terms that refer to the broad categories of the economy in which the individual stocks operate. These 11 categories influence many facets of a typical investment firm’s operations, such as how they measure sector exposure in their portfolios to the sector exposure of the S&P 500 or how they divide research responsibilities for the individual stocks. Our team wanted to explore if these 11 important divisions or “clusters” of the stock market would also be found naturally by a reasonable application of clustering techniques on our dataset. Having a reasonable framework to assign individual stocks (or other financial instruments) into similar groups could have many practical applications in a typical investment management firm. On top of the data cleaning we described earlier, we standardize our 14 features using a custom min-max scaler that is applied on a monthly basis, and we use these standardized features for each clustering algorithm for consistency.

Unsupervised Learning: Methods

The workflow of our unsupervised learning analysis begins with creating a baseline against which we could compare different unsupervised clustering techniques. Using three scoring methods for measuring the quality of clusters when the ground truth is unknown, we were able to create baseline metrics for our comparisons by feeding our features and the S&P sector classifications for each stock into their calculation. We used the Calinski-Harabasz score, the Davies-Bouldin score, and the Silhouette score in our notebook analyses. We think the Calinski-Harabasz score and Davies-Bouldin score offer an interesting contrast to each other because the former concerns both intra-cluster and inter-cluster relationships while the latter is mainly concerned with intra-cluster relationships (Collins-Thomson). We also like the Silhouette score because it is range-bound between -1.0 and +1.0 (bad to good), with clusters scoring near zero, meaning they have overlap, making it a very easy score to interpret (Rousseeuw). For this reason, in the charts below we will only be illustrating the Silhouette score. The charts for the Calinski-Harabasz score and Davies-Bouldin score are found in the Appendix.

After creating these baseline scores for the original 11 clusters of the S&P 500 Stocks over time, we proceeded to look at three different clustering algorithms, tuning the parameters of each algorithm to see if we can find the number of clusters that represent the best score on each of our scoring metrics. These three algorithms are a basic k-means clustering algorithm, the more intermediate agglomerative (or bottom-up hierarchical) algorithm, and a more complex affinity propagation algorithm. We chose these three algorithms because we wanted to examine clustering based on basic, intermediate, and advanced clustering techniques and we have witnessed each of these algorithms used before in investment industry research reports.

We tuned the most important parameter(s) for each of these three algorithms to return the number of clusters that have the best cluster score. In determining the range of parameters to search through, we considered the specifics of each algorithm. For the k-means algorithm, the primary parameter was the number of clusters, and we chose what we felt was a reasonable range from two to 20 clusters, inclusively. For the agglomerative algorithm, the main parameter we varied was the Ward distance threshold that would determine new clusters. Ward distance is the difference between the sum of the squared errors from within-cluster distances and the sum of squared errors from outside-cluster distances (Collins-Thompson). We were able to view several dendrograms for

[5] See the notebook titled “Part 04 - Unsupervised Learning”

individual months (see **Figure 4**) to determine a reasonable range of the Ward distance for our normalized data would range from 7.0 down to 0.5 in increments of 0.25. For the affinity propagation algorithm, there were two key parameters that we iterated through, the first being a damping factor and the second being a precision factor. Affinity propagation takes 'exemplars' as representative units of each cluster and we can control the extent to which the current value is maintained relative to new values through the damping parameter (Frey and Dueck). Similarly, the preference parameter assigns points values, with higher values dictating which points are more likely to be used as exemplars. We chose reasonable increments to iterate over for each of these two parameters, with the damping parameter range from 0.5 to 0.99 in increments of 0.1, and with the precision parameter ranging from -30 to -0.05 choosing 20 increments based on our preliminary testing.

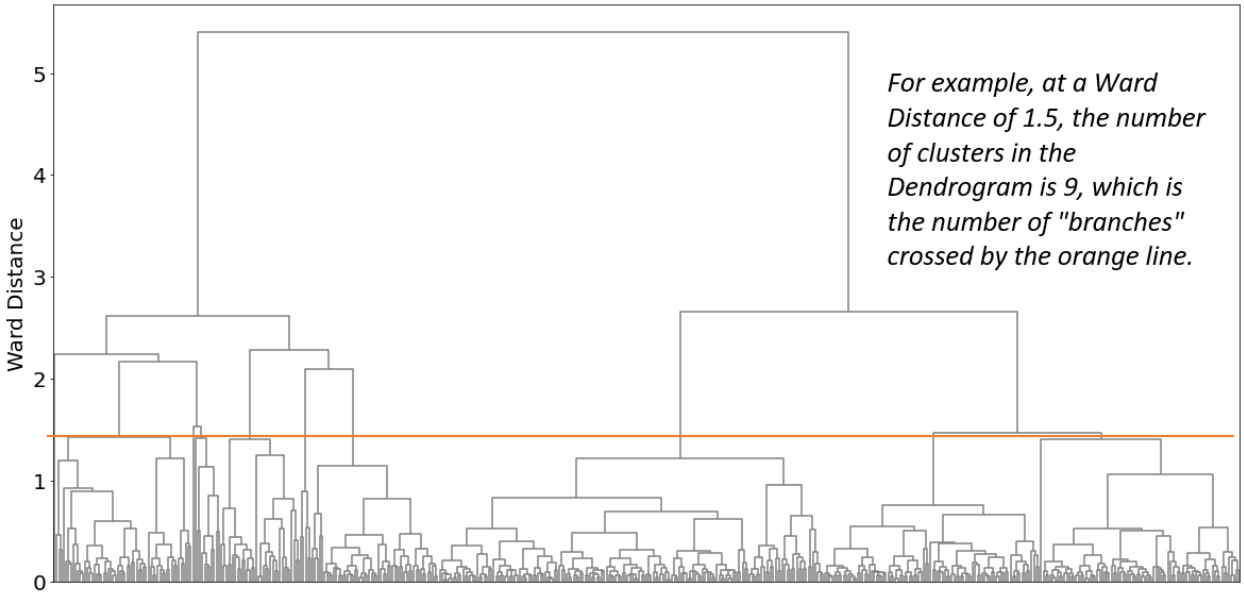


Figure 4 - Dendrogram for 2020-12-31 data

For the agglomerative algorithm, we also attempted a method presented to us in class that proposes that the optimal number of clusters is at the "elbow" of the curve formed by plotting the sum of squared errors (the distance between clusters) and the number of clusters (Collins-Thompson). The concept is that the 45-degree line tangent to this "elbow" is the point at which there are diminishing benefits to additional clusters. We chose to use the agglomerative algorithm to attempt this because we could very easily visualize the Ward distance in the dendrogram and compare it to the number of clusters being created to see if this method was providing reasonable results (see **Figure 5**). We created a function that would loop through the different distance threshold parameters that we described above, but this time returning the number of clusters at the "elbow" of the curve rather than the clusters with the highest score.

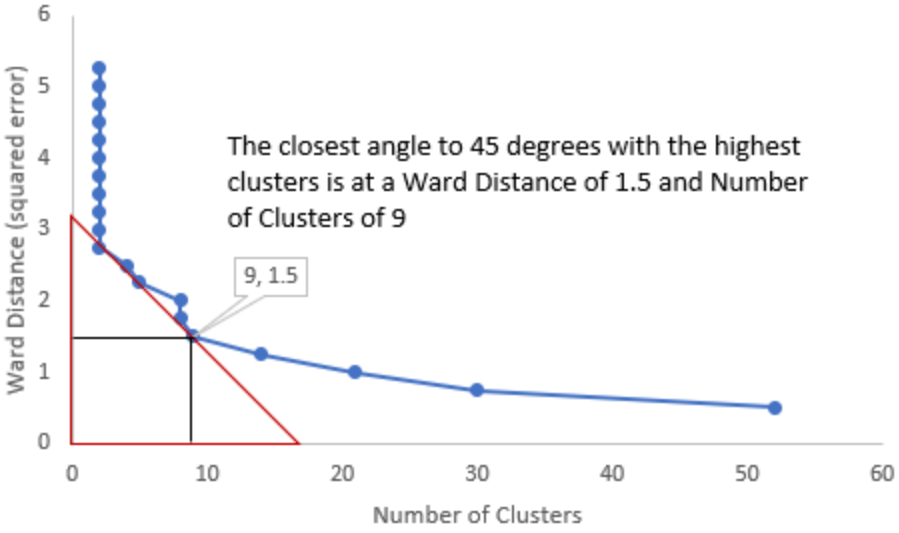


Figure 5 - Distance vs. Clusters for 2020-12-31 data

While the four methods described above were successfully executed, there were several challenges we encountered in our unsupervised analyses. Initially, we were trying to run our algorithms on our entire dataset of around 500 stocks covering over 240 months, which in many cases could not be completed with our computational resources available to our team. We solved that by iterating through individual months in our dataset, trying to find the optimal number of clusters for each date. We ran into issues with computational resources when looping over parameters, which we solved by reasonably reducing the number of increments we were using. Other methods we employed in our dataset to identify clusters were DBSCAN and OPTICS, a generalization of DBSCAN, due to the non-flat geometry of our data and the uneven cluster sizes we identified in our exploratory data analysis. OPTICS could also account for variable cluster density which was a characteristic we expected in our data. Additionally, DBSCAN and OPTICS can identify outliers, but our best cluster quality scores were achieved when the number of outliers were high, leaving us with only a fraction of the 500 stocks in a given month assigned to a cluster. Since the number of outliers were high, we abandoned DBSCAN and OPTICS in favor of the other three methods we discussed.

Unsupervised Learning: Evaluation

We found that all three algorithms tuned to maximize cluster quality scores could create clusters that produce better cluster quality scores when compared to the baseline score of the original 11 clusters. We illustrate this conclusion in **Figure 6**, which shows the cluster quality score from setting parameters with the highest Silhouette score. The baseline cluster score (the red line) lies between the -0.20 to 0.75 range, while all three of the algorithms show a higher (better) score, which varies from 0.15 to 0.80 over the 20-year period. While we show only the Silhouette score in this report, similar conclusions can be made with the Calinski-Harabasz score and Davies-Bouldin score about score improvement (see **Appendix A8**).

However, while all three score-maximized algorithms improved cluster quality scores compared to the baseline we considered, the chosen optimal number of clusters did not always seem to make intuitive sense to us. We illustrate this conclusion in **Figure 7**, which shows what each method determined to be the optimal number of clusters based on the parameters with the maximum Silhouette score. We see the number of optimal clusters ranging from 2 to 18, and most often the number of clusters was well below the 11 in the baseline, and also frequently in the low single digits. The Calinski-Harabasz score method had an even greater number of months where a very low number of clusters, between two to four clusters, was considered optimal. The Davies-Bouldin score

method had the opposite problem, very often suggesting the number of clusters to be in a much higher range, between 40 to 140 clusters. All of these optimal numbers of clusters generally did not feel intuitive to us, suggesting maximizing cluster quality scores alone may not be an ideal approach to clustering.

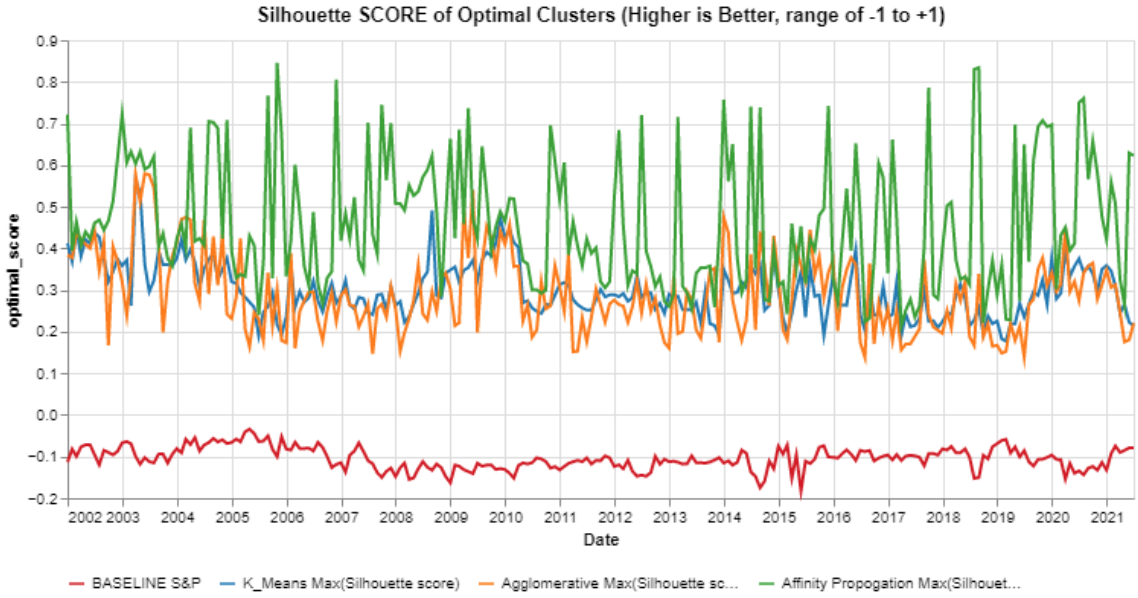


Figure 6 - Silhouette Score of Optimal Clusters

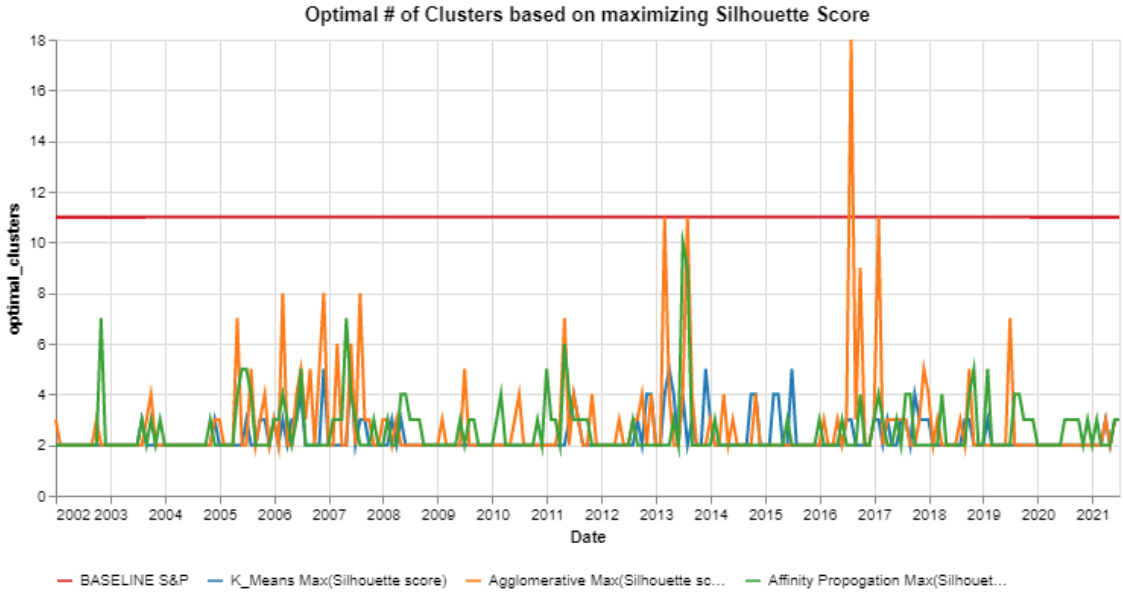


Figure 7 - Optimal Number of Clusters

Our initial attempts to determine the optimal number of clusters don't appear intuitive, or realistic, to us when tuning our parameters to only produce the best cluster quality score. The fourth method we tried seemed more promising. This is the method where we use the agglomerative algorithm and try to pinpoint the elbow of the curve illustrating where there are diminishing benefits to adding additional clusters. As illustrated in **Figure 8**, we see that the optimal number of clusters (the purple line) determined by this method still produces a score much higher than our baseline (the red line) and comparably lower to the score created by the first three methods (the light gray lines).

We also observe that the number of clusters suggested by this elbow method (see **Figure 9**) seems more intuitive than the number of clusters from maximizing scores. Here we observe the number of clusters ranging from 3 to 19 over time, but mostly the number of clusters stays in the mid-to-high single digit range, which we view as an improvement over the three prior score maximization methods. If we were to look at the same charts for the Calinski-Harabasz score and the Davies-Bouldin score, we would see this agglomerative elbow method as slightly preferable to score maximization alone.

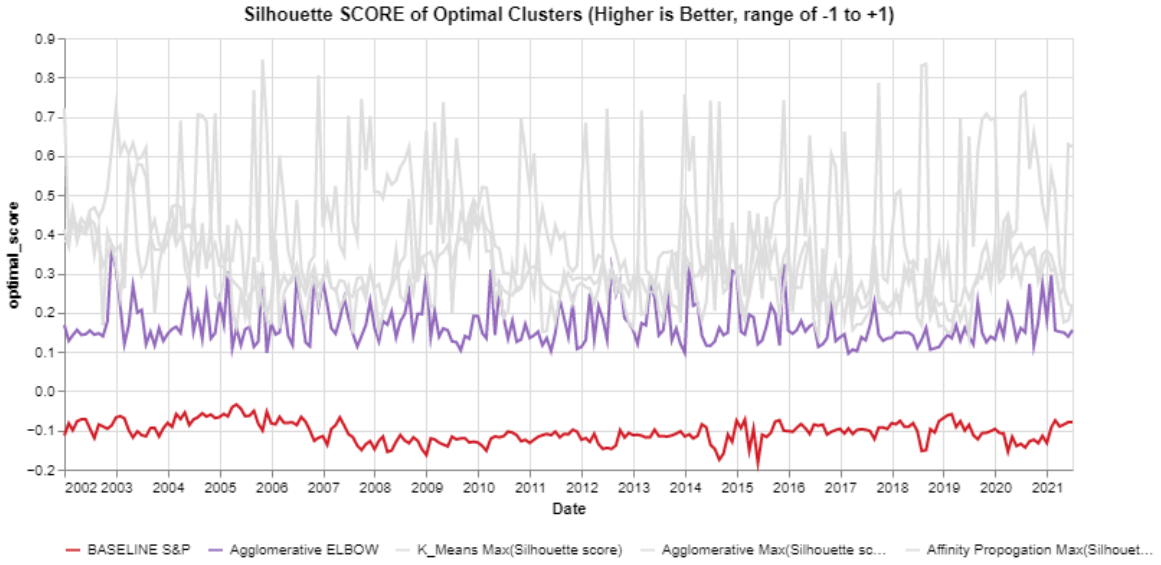


Figure 8 - Silhouette Score of Optimal Clusters with Agglomerative Elbow Method

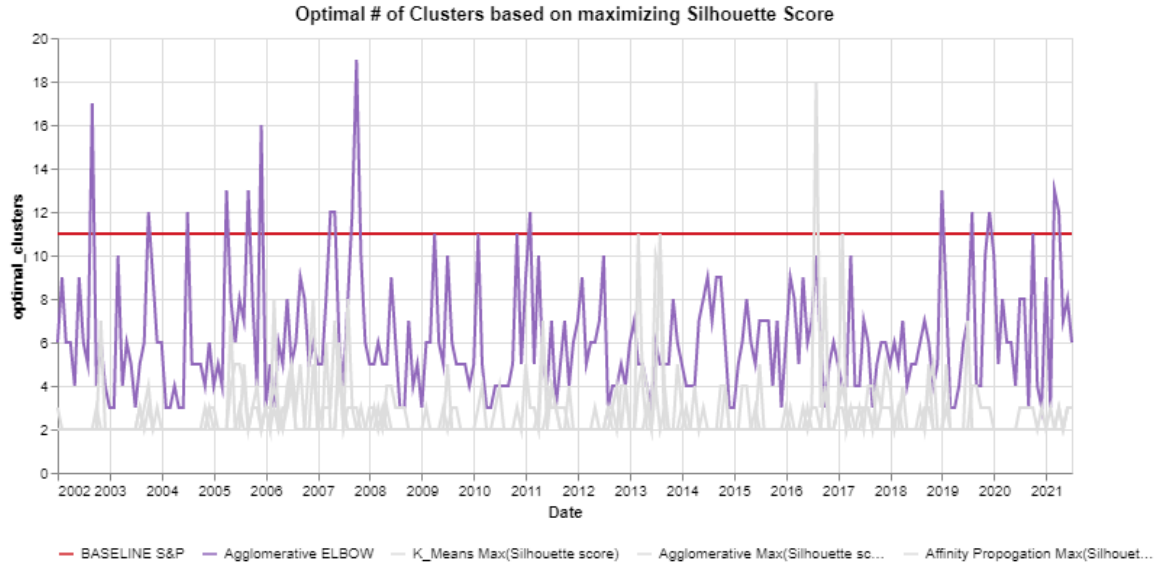


Figure 9 - Optimal Number of Clusters with Agglomerative Elbow Method

In summary, our unsupervised clustering analysis showed that regardless of the algorithm, tuning our parameters to maximize any of the three cluster quality scores produced results that we were not intuitively comfortable with. Using agglomerative clustering and tuning the parameters based on the concept of finding the elbow as brought up in previous classes gave us a more intuitive number of clusters that we were more comfortable applying to the domain of finance.

Discussion

In the supervised learning portion of our project, we learned that not all classifiers were suitable for our dataset. We found that our classes are not linearly separable as they did not do well with linear classifiers. We believe that the data points are positioned in a way that requires jagged decision boundaries. That is why tree-based models (random forests, decision trees and AdaBoost) tend to perform well. We were surprised by some classifiers that would not run because the model did not scale well with our dataset. For example, gradient boosted trees and support-vector machines did not complete training within a reasonable time frame and we decided against using these models. With more time and resources, we would like to do some more feature engineering to make our data optimal for the problems we aimed to solve using the different machine learning approaches we proposed. We would also explore other models, especially those beyond the sklearn library. Traditional time series models, such as those under the ARIMA and GARCH classes of models, would be suitable candidates for further investigation given the time series structure of our data**.**

In the unsupervised learning portion of our project, we learned that achieving reasonable results from clustering requires careful tuning of the parameters, which may not always be achieved systematically. What surprised us was that having the highest score of cluster quality was no guarantee of achieving reasonable results, and there are trade-offs between having a systematic clustering process (highest quality score) and having intuitive results. We found that we preferred a method that did *not* maximize cluster quality but instead chose parameters based on where the benefits of an additional cluster started to diminish relative to the error term (or ward distance). We successfully implemented this elbow method with the agglomerative clustering algorithm. With more time and resources, we would want to further explore the elbow method that we explored and see if we could apply something similar to other algorithms. We could also use our knowledge of the three cluster quality scores to similarly apply this elbow concept to find the point of reduced improvement in the score per additional cluster, which might yield better outcomes than maximizing the score alone.

Regarding ethical concerns, while our dataset does not include any individual's personal information, it does contain information about companies that employ individuals. We would want to be aware of any unintentional harm or unforeseen consequences of either our supervised or unsupervised analyses. For instance, if one of our supervised learning models was applied to encourage investment in top-ranked stocks, we would want to ensure that it was not accidentally penalizing minority-owned businesses. Similarly, if an unsupervised clustering algorithm was accidentally providing groupings that were used to assign business responsibilities according to race or gender, we would find it unethical. While our dataset does not have some of the more obvious ethical concerns, we can still imagine ways in which we might unintentionally cause harm and would need to be able to appropriately identify those instances and mitigate them before deploying our models or results.

Statement of Work

Overall, the workload was well balanced amongst the team. Erik performed the initial data cleaning, which merges the original Excel file with multiple tabs into a single CSV file. We gathered in weekly meetings to determine the best approach to manipulate the data. Erik developed this pipeline to prepare the dataset for our supervised and unsupervised learning methods. Next, Jason used the cleaned and normalized data against several classifiers. Vlad has also worked on an extreme gradient boosted model for the supervised learning section that did not make it into the report. All parties worked on a portion of the unsupervised learning task, with Jason taking the lead; Jason worked on k-means clustering, Erik worked on agglomerative clustering, and Vlad worked on affinity propagation. Erik wrote most of the introduction, dataset, and unsupervised learning, while Jason wrote most of the supervised portion. Vlad completed the remainder of the report and thoroughly edited the commentary.

Works Cited

Collins-Thompson, Kevyn. *Evaluating and Labeling Clusters*. *Coursera,* https://www.coursera.org/learn/siads543/lecture/UV04f/evaluating-and-labeling-clusters.

Collins-Thompson, Kevyn. *Hierarchical clustering*. *Coursera,* https://www.coursera.org/learn/siads543/lecture/UV04f/evaluating-and-labeling-clusters.

Collins-Thompson, Kevyn. *Overfitting and Underfitting*. *Coursera*, https://www.coursera.org/learn/siads542/lecture/fVStr/overfitting-and-underfitting.

Frey, Brendan J., and Delbert Dueck. "Clustering by passing messages between data points." *science*, vol. 315.5814, 2007, pp. 972-976.

Rousseeuw, Peter J. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics*, vol. 20, 1987, pp. 53-65. 0377-0427.

Appendix

A1. Original Feature Definitions

Feature	Variable Name	Definition
Trailing 12-month performance	Price_Returns_12m	The trailing 12-month performance of the stock. A measure of long-term momentum.
Trailing one-month performance	Price_Returns_1m	The trailing one-month performance of the stock. A measure of short-term momentum.
Trailing dividend yield	Trail_DivYld	The trailing dividend yield of the stock, based on the last four quarters reported dividend.
Price-to-book ratio	PB	The ratio of the stock price to the most recent quarter's book value of the stock (a term for the "on paper" value of the company according to its balance sheet)
Trailing EV to EBITDA	Trail_EV_EBITDA	The trailing ratio of Enterprise Value to EBITDA of the stock, based on the share price, shares outstanding, and debt on the balance sheet (the numerator) divided by the trailing four quarters of Earnings Before Interest Taxes Depreciation & Amortization on the income statement (the denominator. The metric is supposed to reflect the value of the company overall (both stock & debt) rather than just the stock alone.
Trailing price over earnings per share	Trail_PE	The trailing ratio of stock price over earning per share of the stock, based on the last four quarters reported earnings per share.
Trailing three-year average growth in dividends per share	Trail3yrAvg_DPSgro	The trailing three-year average growth in dividends per share based on the most recent 36 quarters.
Trailing three-year average growth in earnings per share	Trail3yrAvg_EPSgro	The trailing three-year average growth in earnings per share based on the most recent 36 quarters.
Volatility	Volatility	The volatility of the share price over the past year.
Total debt-to-capitali	Debt_to_MktCap	The most recently reported total debt of the company divided by the market capitalization. A measure of the

zation ratio		quality of the company's balance sheet (too high = bad, and yet too low also = bad!)
Net debt to EBITDA	NetDebt_EBITDA	The most recently reported total net debt (debt minus cash on hand) of the company divided by the last four quarters of Earnings Before Interest Taxes Depreciation & Amortization on the income statement. A measure of the quality of the company's balance sheet (too high = bad, and yet too low also = bad!)

A2. Additional Feature Definitions

Feature	Variable Name	Definition
Trailing three-month performance	Price_Returns_3m	We use the already existing one-month performance to calculate this three-month momentum measure.
Trailing six-month performance	Price_Returns_6m	We use the already existing one-month performance to calculate this six-month momentum measure.
Trailing three-month moving average	Moving_Avg_Returns_3m	We use the already existing one-month performance to calculate the three-month moving average of the stock's performance.
Trailing six-month moving average	Moving_Avg_Returns_6m	We use the already existing one-month performance to calculate the six-month moving average of the stock's performance.
Trailing twelve-month moving average	Moving_Avg_Returns_12m	We use the already existing one-month performance to calculate the 12-month moving average of the stock's performance.
Trailing one-year earnings per share growth	Trail1yr_EPSgro	We use existing data we had on hand for Trailing four-quarter Earnings Per Share (EPS) to estimate a year-on-year trailing growth rate for earnings.
Trailing one-year dividend per share growth	Trail1yr_DPSgro	We use existing data we had on hand for Trailing four-quarter Dividend Yield and the share price to estimate a year-on-year trailing growth rate for earnings.

A3. Brief Description of Notebooks

Jupyter Notebook	Report Section	Description
Part 01 - XLSX Data to CSV Data with Some Cleaning	Dataset	Notebook used to convert original excel file of multiple sheets into a single csv file: SP500_Index_Data_partial_clean.csv
Part 02A - Additional Feature Creation	Dataset	Notebook used to read SP500_Index_Data_partial_clean.csv to generate additional features and outputs SP500_Index_Data_with_Added_Features.csv
Part 02B - Feature Cleaning	Data Cleaning & Exploratory Data Analysis	Notebook used to read SP500_Index_Data_with_Added_Features.csv, eliminate the missing values (by dropping, forward filling, and median filling), and normalize the data. Outputs SP500_Index_Normalized_Features.csv
Part 02C - Supervised Y Variable Options	Supervised Learning	Notebook used to read SP500_Index_Data_partial_clean.csv to generate the different possible y variables for supervised learning. Outputs SP500_Supervised_Y_Variable_Options.csv
Part 03 - Supervised Learning	Supervised Learning	Notebook that reads SP500_Index_Normalized_Features.csv and SP500_Supervised_Y_Variable_Options.csv. Generates classification models mentioned in this report
Part 04 - Unsupervised Learning	Unsupervised Learning	Notebook that reads SP500_Index_Normalized_Features.csv and generates clustering models mentioned in this report

A4. Feature Correlation of Performance-only features

Performance-only Feature Correlations	Price_Returns_12m	Price_Returns_6m	Price_Returns_3m	Price_Returns_1m	Moving_Avg_Returns_12m	Moving_Avg_Returns_6m	Moving_Avg_Returns_3m
Price_Returns_12m	1.00	0.59	0.39	0.22	0.93	0.58	0.39
Price_Returns_6m	0.59	1.00	0.66	0.37	0.67	0.96	0.65
Price_Returns_3m	0.39	0.66	1.00	0.53	0.47	0.70	0.98
Price_Returns_1m	0.22	0.37	0.53	1.00	0.28	0.42	0.56
Moving_Avg_Returns_12m	0.93	0.67	0.47	0.28	1.00	0.69	0.48
Moving_Avg_Returns_6m	0.58	0.96	0.70	0.42	0.69	1.00	0.71
Moving_Avg_Returns_3m	0.39	0.65	0.98	0.56	0.48	0.71	1.00

A5. Feature Correlation of Non-Performance features

Non-Performance Feature Correlations	Trail_DivYld	PB	Trail_EV_EBITDA	Trail_PE	Trail3yrAvg_EPSgro	Trail3yrAvg_DPSgro	Volatility	Debt_to_MktCap	NetDebt_EBITDA	Trail1yr_EPSgro	Trail1yr_DPSgro
Trail_DivYld	1.00	0.01	-0.01	-0.02	-0.01	0.13	0.05	0.15	0.03	0.00	0.19
PB	0.01	1.00	0.00	0.00	0.00	0.01	-0.01	-0.01	0.00	0.00	0.00
Trail_EV_EBITDA	-0.01	0.00	1.00	0.08	0.00	-0.01	0.01	0.00	-0.38	0.00	-0.01
Trail_PE	-0.02	0.00	0.08	1.00	0.00	-0.02	0.03	-0.01	-0.05	0.00	-0.01
Trail3yrAvg_EPSgro	-0.01	0.00	0.00	0.00	1.00	0.02	0.01	-0.01	0.00	0.00	0.05
Trail3yrAvg_DPSgro	0.13	0.01	-0.01	-0.02	0.02	1.00	-0.14	-0.04	-0.01	0.01	0.39
Volatility	0.05	-0.01	0.01	0.03	0.01	-0.14	1.00	0.21	0.02	-0.01	-0.04
Debt_to_MktCap	0.15	-0.01	0.00	-0.01	-0.01	-0.04	0.21	1.00	0.07	0.00	-0.02
NetDebt_EBITDA	0.03	0.00	-0.38	-0.05	0.00	-0.01	0.02	0.07	1.00	0.00	-0.01
Trail1yr_EPSgro	0.00	0.00	0.00	0.00	0.00	0.01	-0.01	0.00	0.00	1.00	0.00
Trail1yr_DPSgro	0.19	0.00	-0.01	-0.01	0.05	0.39	-0.04	-0.02	-0.01	0.00	1.00

A6. Feature importance for each feature in the dataset, ranked in ascending order

Feature	Importance
Trail_DivYld	0.055
Trail3yrAvg_DPSgro	0.059
Trail1yr_DPSgro	0.061
Debt_to_MktCap	0.063
Trail_EV_EBITDA	0.063
NetDebt_EBITDA	0.065
PB	0.065
Trail_PE	0.066
Trail3yrAvg_EPSgro	0.067
Trail1yr_EPSgro	0.067
Price_Returns_12m	0.074
Price_Returns_1m	0.084
Volatility	0.090
Price_Returns_3m	0.125

A7. Average value for each feature where the model is incorrect, ranked in ascending order

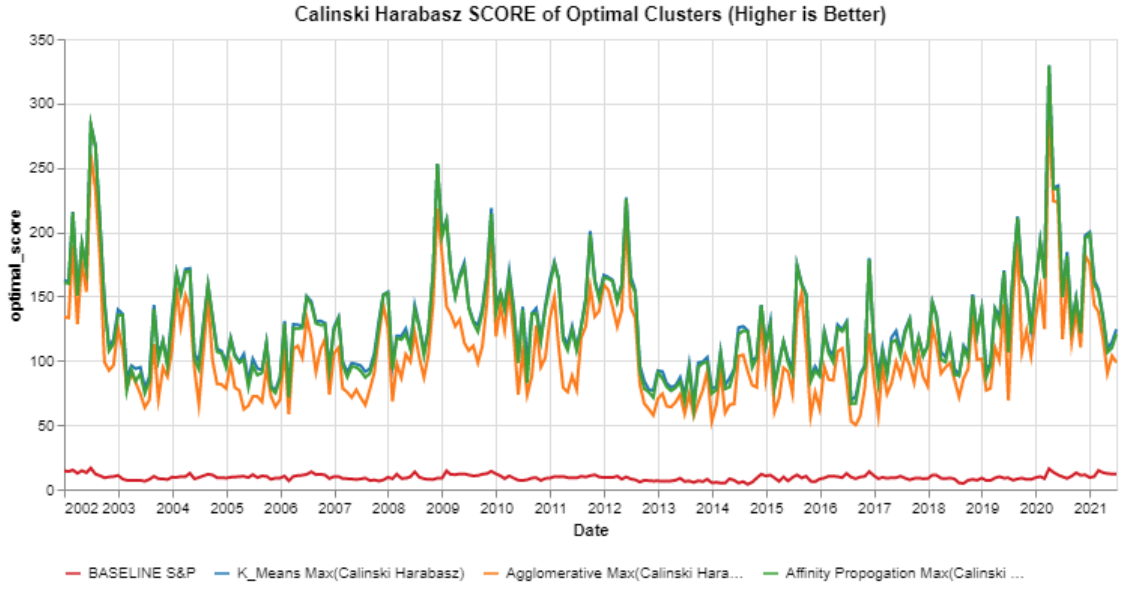
Feature	Average Value
PB	**0.021**
Trail_PE	**0.035**
Debt_to_MktCap	0.036
Trail_EV_EBITDA	**0.048**
Trail_DivYld	**0.077**
Trail1yr_DPSgro	0.135
Trail3yrAvg_EPSgro	0.210
Volatility	0.260
Trail3yrAvg_DPSgro	0.293
Price_Returns_12m	0.322
NetDebt_EBITDA	0.347
Price_Returns_3m	0.418
Price_Returns_1m	0.448
Trail1yr_EPSgro	0.537

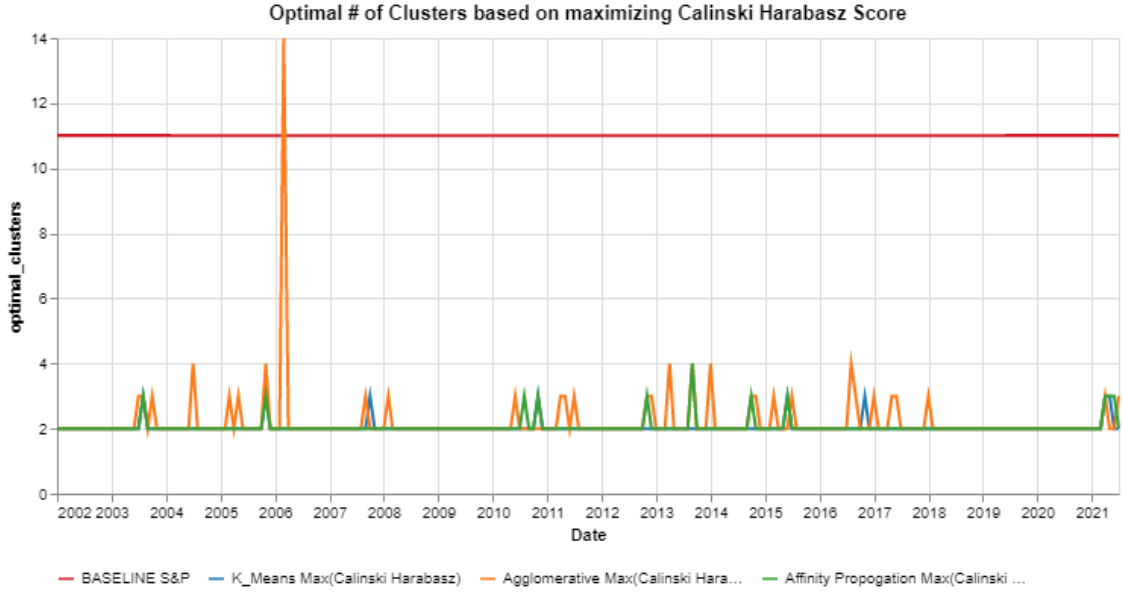
A8. Unsupervised Learning Evaluation of other Cluster Quality Scores

In the report above, we mention that while we looked at three ways of scoring cluster quality, we only illustrated our evaluation with the Silhouette score and only briefly summarized the difference in output by the Calinski-Harabasz score and Davies-Bouldin score. In this section of the appendix, we replicate the same charts that we showed for the Silhouette score for the latter two quality scores.

<u>Calinski-Harabasz: Finding the highest score by tuning the parameters</u>

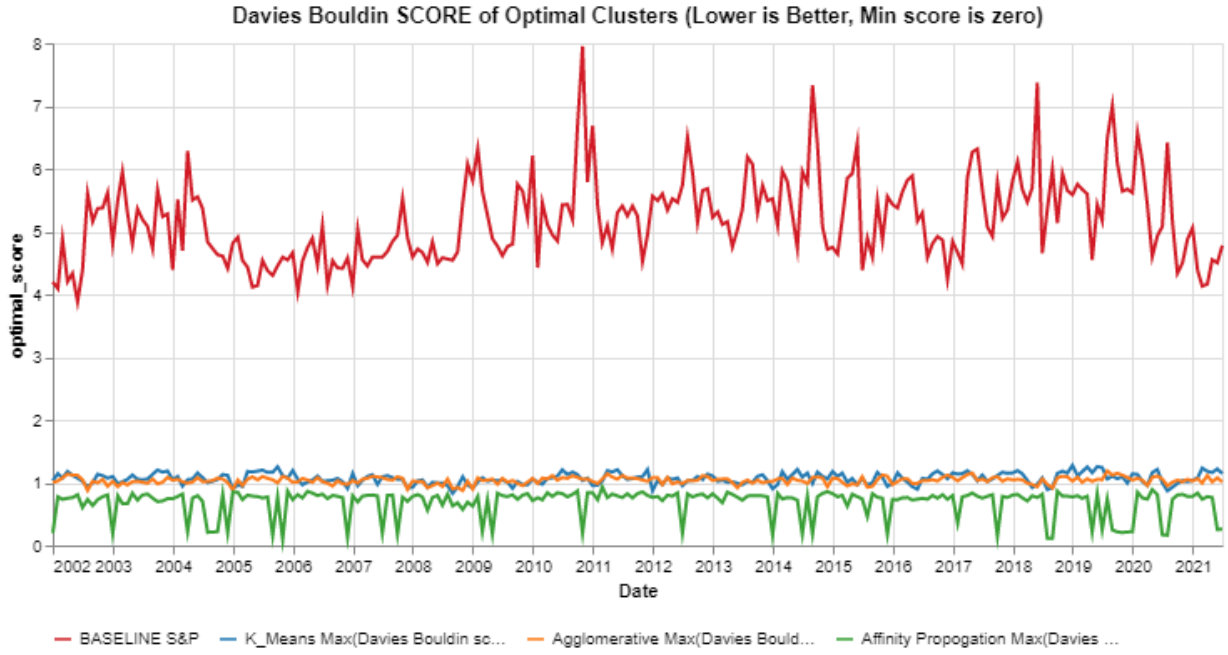
As we noted above, when tuning the parameters of the k-means, agglomerative, and affinity propagation cluster algorithms to maximize based on the Calinski-Harabasz score, we find we can certainly improve on the Calinski-Harabasz score of the baseline S&P sector classifications. However, the optimal number of clusters did not always make intuitive sense to us. In the case of Calinski-Harabasz seemed to stay in the very low-single digits of two to four.

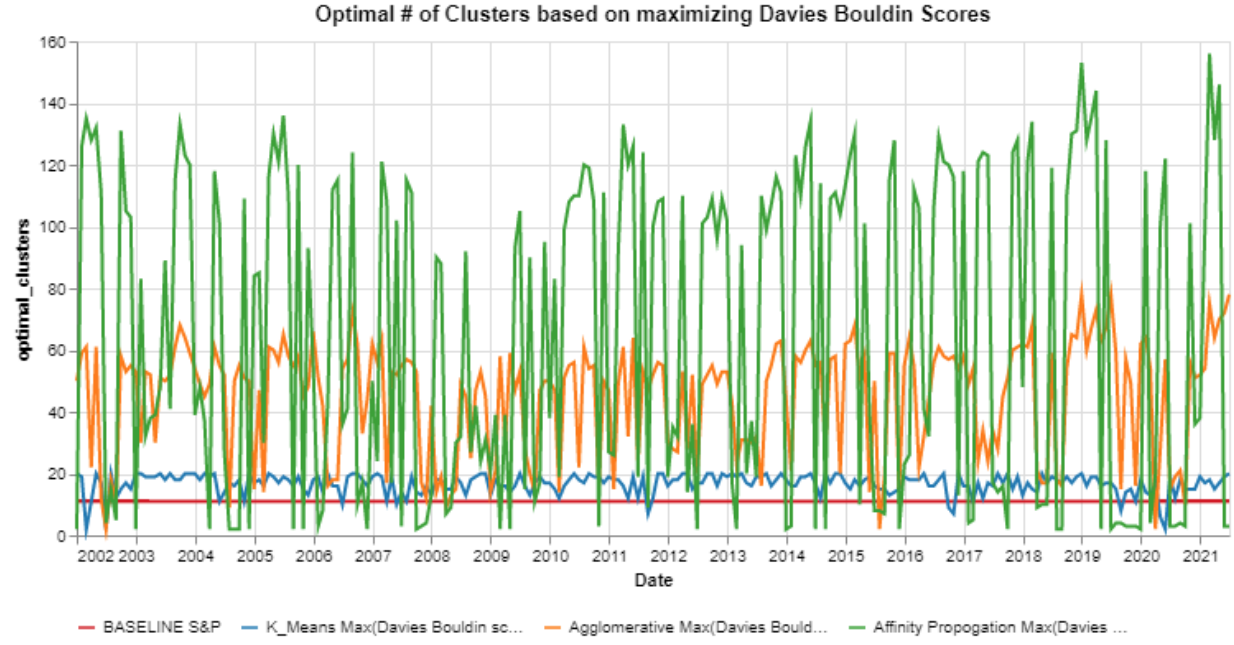




Davies-Bouldin: Finding the lowest score by tuning the parameters

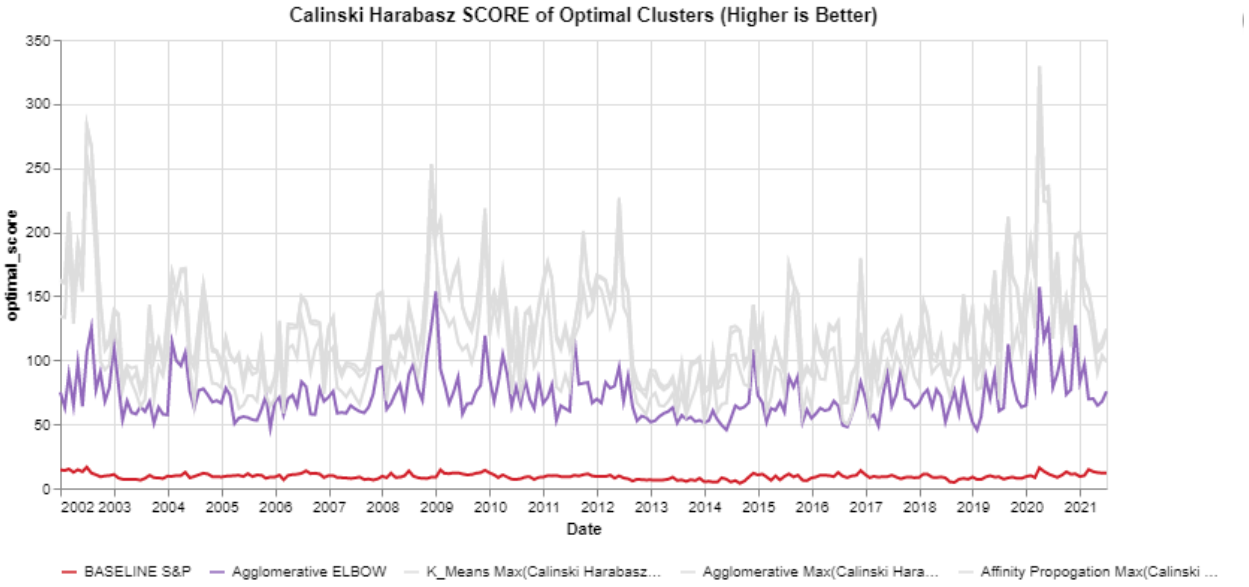
As we noted above, when tuning the parameters of the k-means, agglomerative, and affinity propagation cluster algorithms to maximize based on the Calinski-Harabasz score, we find we can certainly improve on the Davies-Bouldin score of the baseline S&P sector classifications. However, the optimal number of clusters did not make intuitive sense to us. In the case of Davies-Bouldin, we generated an optimal number of clusters in a much higher range than we would consider realistic between a range of 40 to 140 clusters.

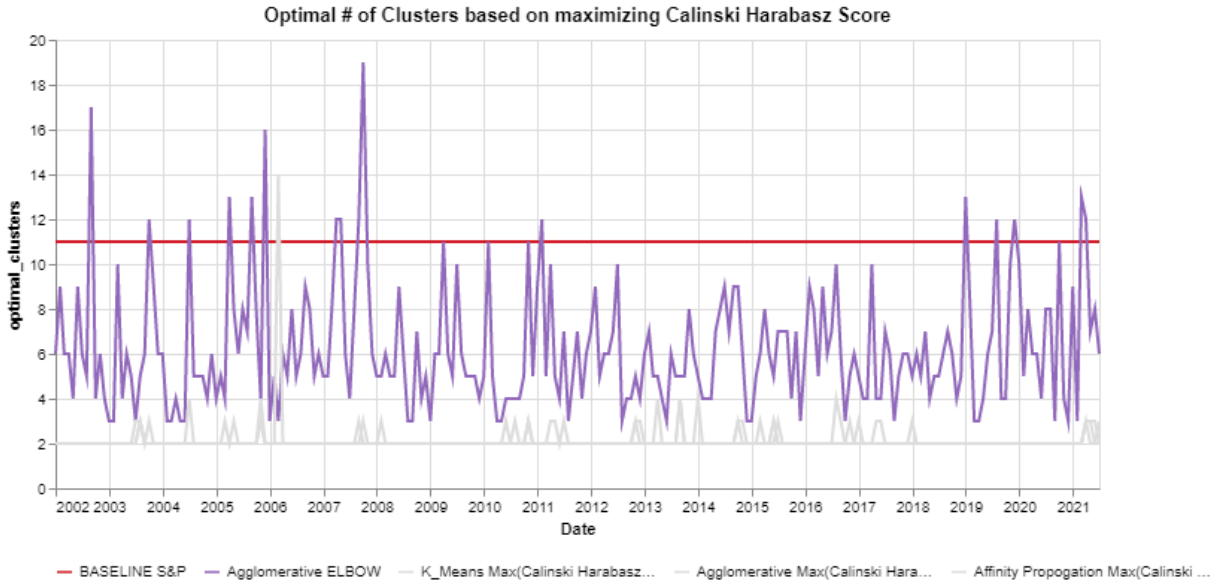




Calinski-Harabasz: Comparing to the elbow method of parameter tuning

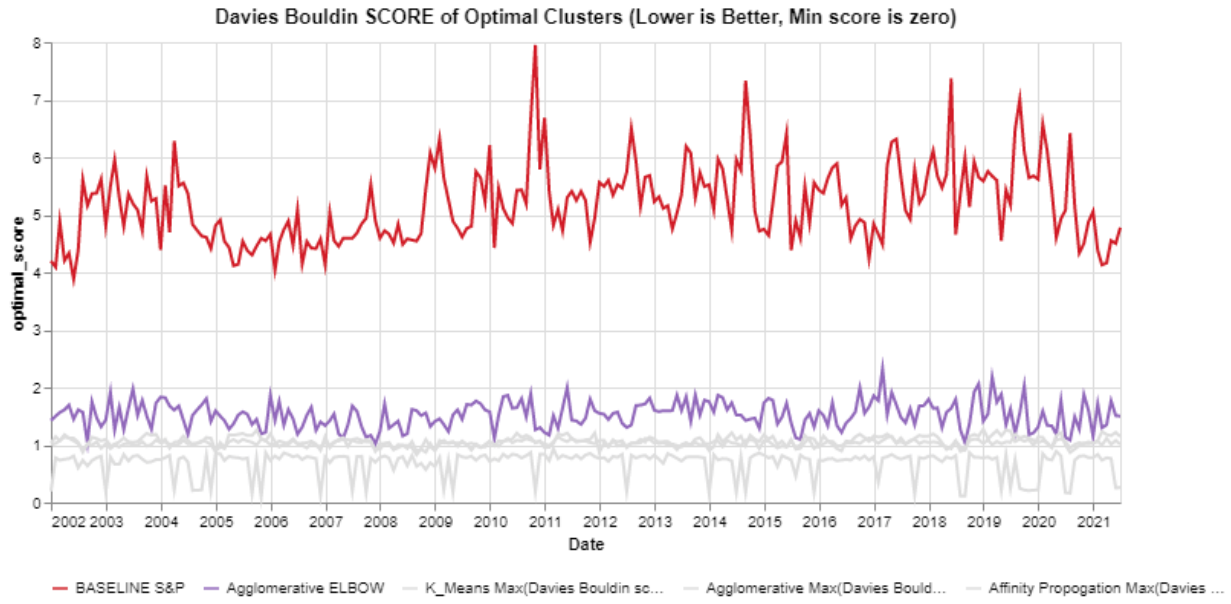
Below we reproduce the same charts for our more preferred elbow method of tuning the parameters that we describe in the report but now compare to the same results from maximizing the Calinski-Harabasz score rather than the Silhouette score. We prefer this elbow method to the score-maximization method and would like to further explore this and other parameter tuning choices with more time and resources.

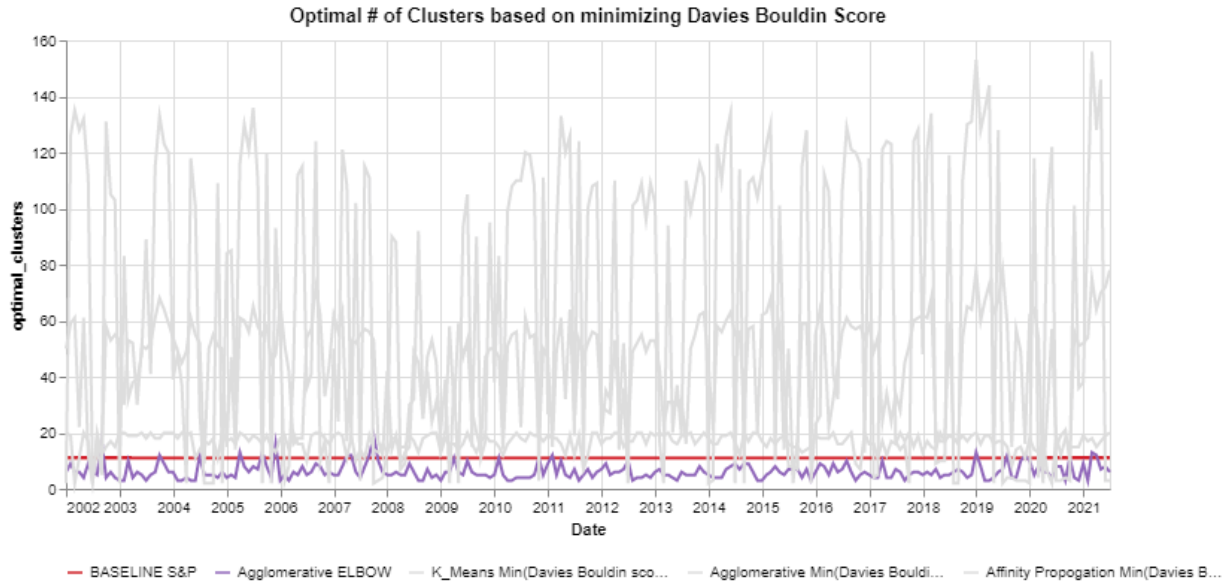




Davies Bouldin: Comparing to the elbow method of parameter tuning

Below we reproduce the same charts for our more preferred elbow method of tuning the parameters that we describe in the report but now compare to the same results from minimizing the Davies-Bouldin score rather than the Silhouette score. We prefer this elbow method to the score-maximization method, and we would like to further explore this and other parameter tuning choices with more time and resources.



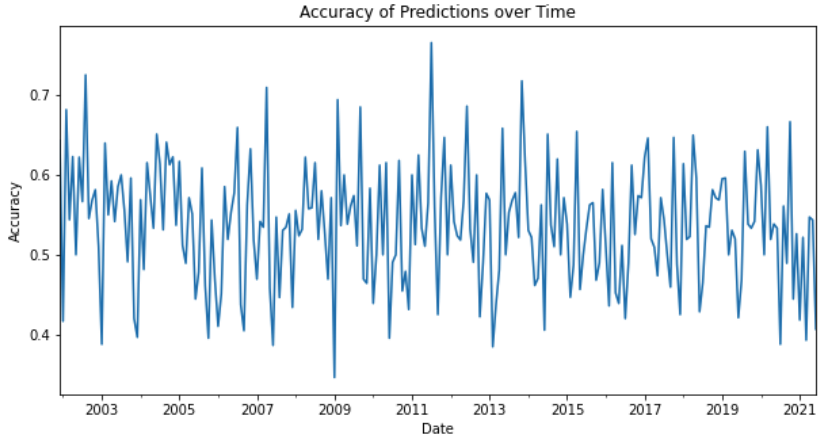


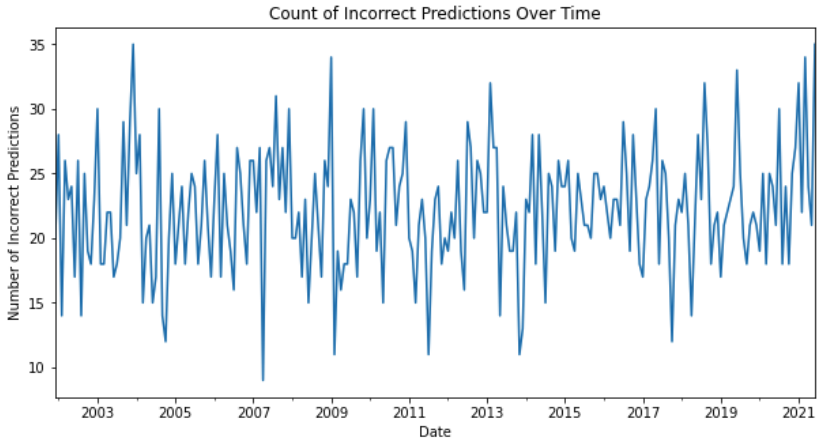
A9. Failure Analyses of Supervised Learning

These charts illustrate the statements made in the section of our supervised learning portion of the project that deals with failure analysis. As stated in the report, we used the best-performing random forest model to try to address this. We were able to isolate the incorrect predictions in our test dataset to help with this analysis. As referenced in the report above, it was the features related to valuation that had the most consistent trend among the data points we analyzed where the random forest model got it wrong. The table below illustrates that, showing that the normalized features (which were normalized on a min-max scale from zero to one) have a very low average value. The other features for incorrect predictions had an average that would be considered normal.

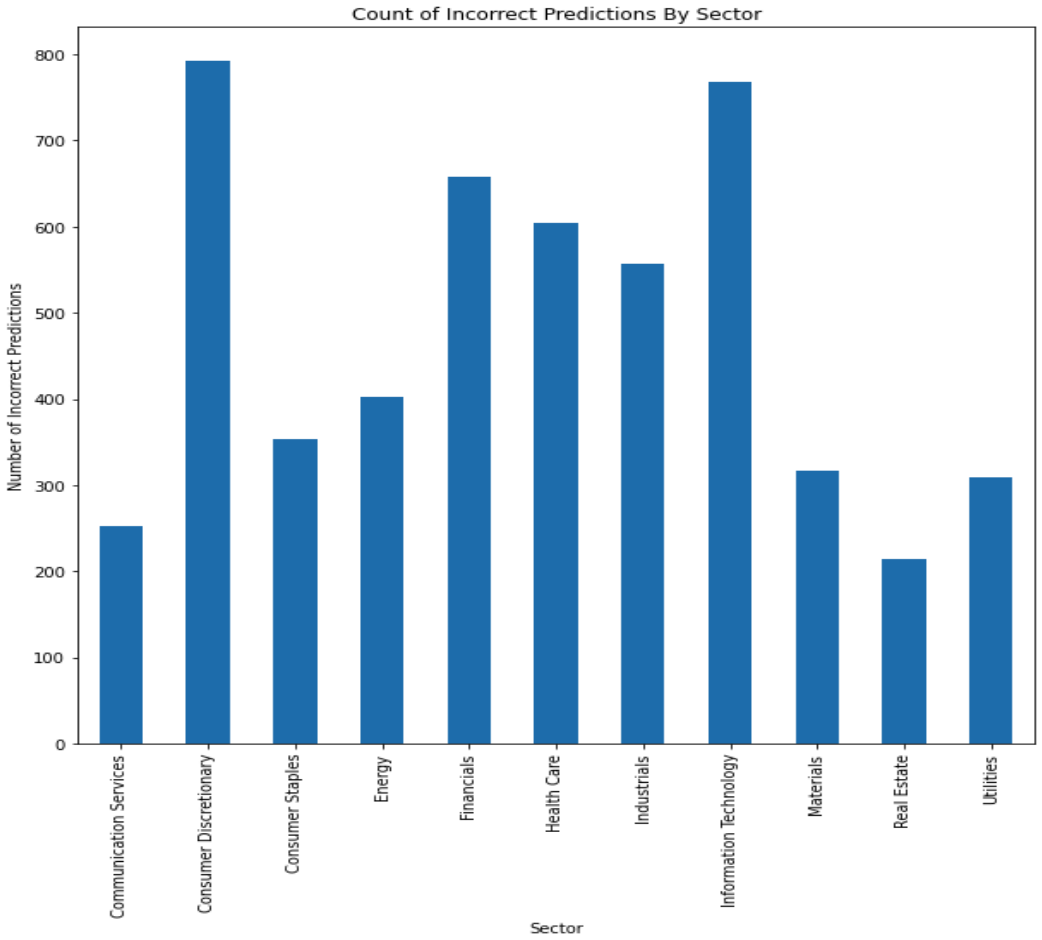
Feature	Variable Name	Average Value
Price-to-book ratio	PB	0.021
Trailing price over earnings per share	Trail_PE	0.035
Trailing EV to EBITDA	Trail_EV_EBITDA	0.048
Trailing dividend yield	Trail_DivYld	0.077

In these next two charts, we were trying to identify if there were time periods that were responsible for the model's inaccuracies. As illustrated, while there are months where the accuracy touched low points, the model's accuracy ranged up and down across the history of our test set.





We also tried to identify where this random forest model was performing worse or better by the individual S&P sectors. The below bar chart illustrates that our most problematic sectors were Consumer Discretionary, Financials, and Technology. This is good to know if we were to implement such a model, but also understandable given that these three sectors are very sensitive to the overall economy and to any quick changes in that economy. Therefore, these findings are not unexpected.



A10. Other methods and variables we considered

We tried several methods that did not ultimately make it into the report. This includes principal component analysis (PCA), using a GroupBy on our dataset, DBSCAN/OPTICS, support-vector machines, and XGBoost. We also tried using other y-variables, such as full-ranking, deciles, and quintiles.

When performing the clustering methods, we hypothesized that correlated features would lead to bad performance. That is why we choose to use PCA to reduce the dimension of the dataset from 18 to 9 while retaining 90% of the variance. While the PCA method worked to reduce the dimension and retain most of the data, it did not make a significant impact on the quality of the clusters or the number of clusters. Since the new 9 variables are hard to interpret, we decided not to use PCA in the interest of keeping our columns interpretable. We ultimately chose to drop some highly correlated columns, which is how we ended with the 14 columns for both the supervised and unsupervised portion of our analysis.

Our dataset started as a multi-dimensional time series. One approach we tried was to use a GroupBy method to group all tickers using the mean of the values. This is a way to have each ticker as a separate variable, which would be easier to do the clustering. However, this grouping lost a lot of information, so we decided against this method. We simply looped through each date to generate the unsupervised visualizations above.

Besides the three unsupervised methods that we tried (k-mean, agglomerative, and affinity propagation), we also explored DBSCAN and OPTICS. We decided not to use these methods because our dataset was not structured for these methods. DBSCAN and OPTICS require clear distance in between clusters while our dataset was densely clustered with little room for distinction. This yielded a high number of outliers, resulting in only a fraction of the S&P 500 constituents assigned to a cluster and leaving a lot of uncertainty in how outlier stocks would perform in a future period. With a high number of outliers, which had no intuitive way in which we could apply this category to other financial objectives, we pursued other methods which clustered all stocks in a given month.

For the supervised learning component, we also developed a gradient boosted model using XGBoost (Extreme Gradient Boosting) which provides regularization unlike standard gradient boosted machines. Since our random forest model proved to yield strong results, gradient boosted trees were a suitable candidate to explore. We applied the framework to both classification and regression tasks but performed subpar to the random forest model in both regression and classification tasks. While XGBoost may yield better results with careful hyper-parameter tuning, gradient boosting underperforms when there is a lot of noise in the data, which causes the model to overfit and perform poorly. We also created a support vector machine, but this took a long time to train, and decided not to pursue this model further.

A11. Y-variable Selection

We had a choice of 12 possible y-variable to use for the supervised learning portion of this report.

Y-variable	**Variable Name**
Normalized forward 1 month performance (using a standard score, or z-score)	FUTURE_Returns_zscore_overall
Normalized forward 1 month performance (using a standard score, or z-score), grouped by Sector	FUTURE_Returns_zscore_bySector
Normalized forward 1 month performance (using a Min Max method)	FUTURE_Returns_MinMax_overall
Normalized forward 1 month performance (using a Min Max method), grouped by Sector	FUTURE_Returns_MinMax_bySector
Forward 1 month performance, binned by quintiles	FUTURE_Returns_quintile_overall
Forward 1 month performance, binned by quintiles, grouped by Sector	FUTURE_Returns_quintile_bySector
Forward 1 month performance, binned into three groups: top 100, bottom 100, and then everything else in the middle	FUTURE_Returns_top_bot_100_overall

Forward 1 month performance, binned into three groups grouped by Sector: top 100, bottom 100, and then everything else in the middle	FUTURE_Returns_top_bot_100_bySector
Forward 1 month performance, binned by deciles	FUTURE_Returns_decile_overall
Forward 1 month performance, binned by deciles, grouped by Sector	FUTURE_Returns_decile_bySector
Forward 1 month performance, binned into three groups: top 50, bottom 50, and then everything else in the middle	FUTURE_Returns_top_bot_50_overall
Forward 1 month performance, binned into three groups grouped by Sector: top 50, bottom 50, and then everything else in the middle	FUTURE_Returns_top_bot_50_bySector

Half of these variables are by sector and the other half is overall. We ultimately choose to use the overall score because of some problematic sectors in our exploratory data analysis. The first 4 variables are continuous, meaning we needed to run a regression. We choose to use the min-max scaler since it puts all values in the range of zero to one, which makes it easier to understand. The regression performed poorly, which shows the difficulty of the problem; it is hard to predict the exact values of stocks.

Next, we tried to use the deciles and quintiles but only got accuracy scores of around 35%. We ultimately choose to use the top and bottom 100 stocks because it is most similar to how the finance industry ranks stocks. Financial analysts are interested in whether a stock is performing in the top or bottom 100. We also noticed that this is the y-variable with the best performance, so we decided to make it the focus of our report.