ECE 498 Project Progress Report 2
4-credit hours, Spring 2020

Title: Feature generation for portfolio diversification

Team: gowtham4, mananm2, somani4

Link to all our codes: https://github.com/gowthamkuntumalla/Quant_analysis_stock_market

Specifically look at codes with titles:

1. project_AS.ipynb, project_GK_generate_features_csv.ipynb, Project_GK_Algos.ipynb (run in order) - These files generate all the necessary features for further analysis. They also explore various approaches in clustering the data, which will be needed for portfolio diversification.
2. project_MM and experiments in other ipynb files. These files consist experiments on portfolio optimization, along with additional algorithms for diversification.

Feature Engineering:

Step1: We have finished modeling our custom features (around 25) for use in later machine learning models. We will use this in an unsupervised learning ML model to gather 'closeness' in different stocks.

Step 2: We do note that we will not require explicitly stated features for use in a deep learning model as it will internally generate those features in its hidden layers. Link to the overall feature studies performed: https://drive.google.com/file/d/1gQF84druyTsRTsv9BcEXGuzaKjpAT0_U/view?usp=sharing. We programmed a smaller set of features which we think may depict longer term correlation between stocks. Some of them are straightforward and linear in nature. Others are complex and may capture nonlinear interactions in between stocks.

ML model - Clustering:

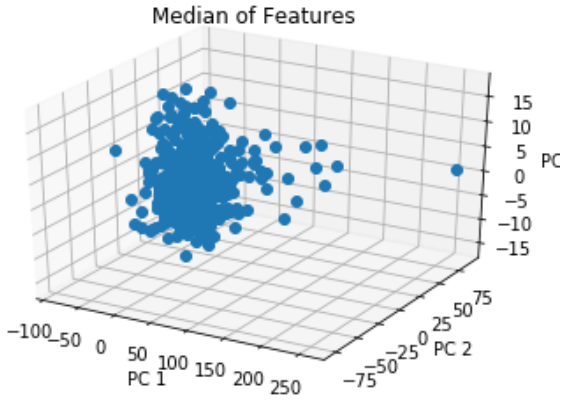
This exercise of gathering different stocks into clusters is to give us a general idea of how a cluster or industry reacts during a particular period of time. It is well known that prices of stocks are heavily dependent on prevailing world factors and news of the period. 'Close' stocks tend to react similarly in a systematic manner. Idiosyncratic or unsystematic risks are inherent in every company but they tend to even out over a long range of time.

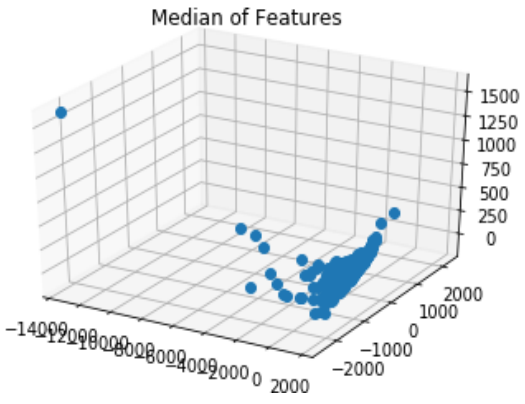
PCA + K-Means:

We are currently exploring K-Means clustering as the first step. Then we will proceed into kernel K-means if proved necessary. We tried t-SNE for 2D visualization but the algorithm is too slow for on our machines hence we will explore using Hierarchical clustering. We will employ a deep learning model (with target industries derived from already known classification on S&P 500 index and wikipedia website) to bolster confidence on our features.

One of the major challenges in clustering is the fact that in addition to having samples (time-series data) and features (stock features), we also have several instances (companies). Performing PCA on such data structure is difficult as it spans in 3 dimensions (i.e. samples, features, instances). As a preliminary effort, we condensed the time series for each feature to a single value (like the mean or median). In doing so, we

do realize losing the seasonal variations in the data, but performing PCA and KernelPCA becomes feasible in this scenario. The aim was to study any formation of clusters/groupings based on this approach. As shown below, we do not observe multiple clusters (as we would like to observe). Everything is clustered together. The surprising fact is that the PC components explain over 97 percent of the total variance. Hence, there is no need to include more PC components and data can be visualized in 3D.



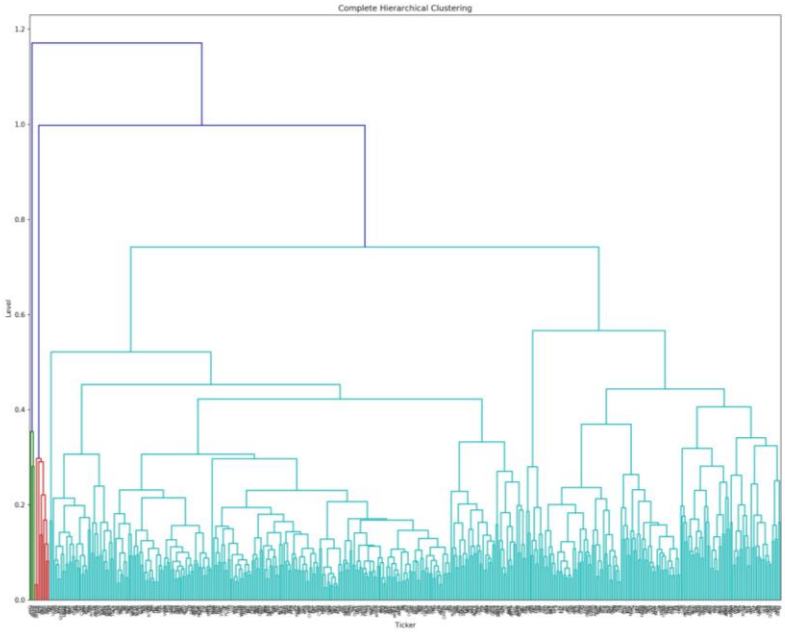


PCA **KernelPCA (2-degree polynomial)**

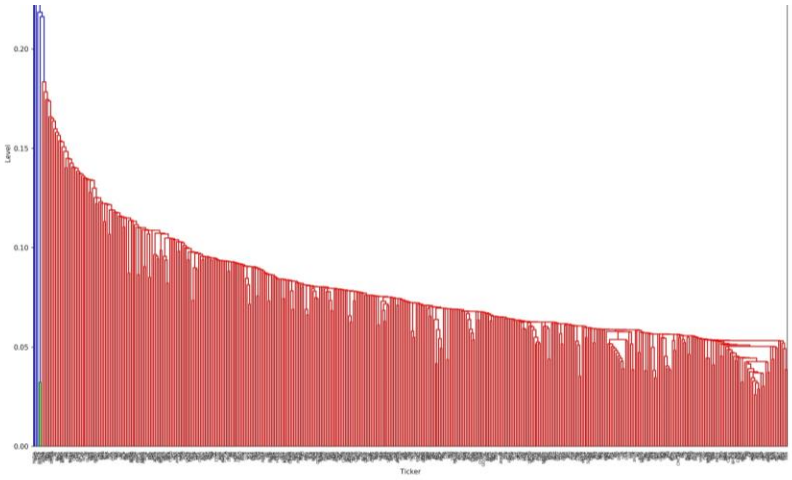
A similar result is observed for the case where the mean of the features is used for clustering analysis.

Hierarchical Clustering:

We explored this method of agglomerative clustering using two methods of linkage - complete link and single link. This was done using normalized close prices of all companies with data for our chosen time period. The matrix dimensions were 468x1235 indicating 468 companies and 1235-day close prices.



Hierarchical Clustering-Complete Link



Hierarchical Clustering-Single Link

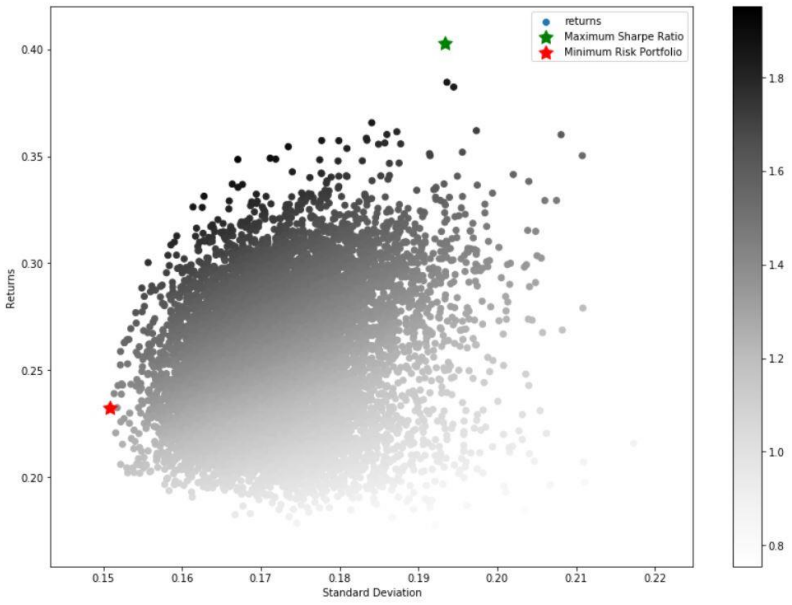
Since the images are not clear, we will give an example here. The very first cluster formed in NVDA, AMD in the single link Dendrogram, which we know are competitors in the market. But it didn't perform well as FB and GOOGL are far apart. Overall, it is only a primitive clustering algorithm. This can be made better by using additional information from other features.

PCA with time series:

Incorporating the time series data to compare two stocks requires a 3-dimensional PCA, algorithms for which are difficult to find and code. To get around this, we thought of a selection algorithm to pick the most 'diverse' stocks together. The algorithm combines the idea of a K-Means++ clustering algorithm with PCA correlation minimization for a given initial "seed stock". As a particular example, on giving Nvidia as the seed stock, the algorithm gives the five next 'most diverse' stocks to be General Dynamics (Defence), Wynn Resorts (Hotels), Key Corp (Banking), Hasbro (Toymaker) and Aetna (Healthcare). Clearly, the algorithm identifies different industries (based on PC1 of the time series PCA). The details of the algorithm will be explained in the final report and can be viewed in the project_MM.ipynb file.

Portfolio Optimization:

Based on Markovitz's Mean-Portfolio theory, we have written functions to calculate the Sharpe Ratio, Treynor Ratio and the Sortino Ratios. This is the logical next step after combining our clustering methods. The importance of optimizing individual stock weights in the portfolio can be seen in the figure below. The figure uses randomized 10,000 trials for a 10-stock portfolio. The portfolio is generated using the Nvidia seed in the selection algorithm as mentioned in the 'PCA with time series' section. As seen, just weighing stocks differently in a portfolio leads to drastically different risk-return pairs. We have highlighted the best Sharpe Ratio portfolio and the least risky portfolio. The figure also renders a clear visualization of the Efficient Frontier as discussed in the modern portfolio theory. We see that just by balancing weights correctly, we get a Sharpe Ratio of ~ 2 (colorbar) and 40% annual returns for the Nvidia portfolio discussed above, for a standard deviation of ~ 19%.



Next Steps:

Currently we are facing issues with using time series data as inputs to ML algorithms built on Scikit Learn. We will try to utilize smarter ways of using our features to obtain meaningful results.

Condensed Data Visualization: t-SNE - too slow, may have to look at other options.

Data Decomposition: Better ways of extracting meaningful information from time-series PCA

Unsupervised Clustering: Hierarchical Clustering, K-Means, Ward HC, Spectral Clustering- good for Time Series Data, GMMs

*** (if time permits) Supervised Classification:** Deep Learning using partial target data from Wikipedia. We call it partial because though S&P agency divides companies into industrial sectors, there is a lot of intercompany interaction which may be ignored at a higher level. Hence, we think of it as partially labeled data.