CLUSTERING TECHNIQUES FOR ASSET ALLOCATION IN RISK PARITY STRATEGY

Anuraag Khandagle
Senior Quantitative Analyst,
OTC Derivatives Pricing, BNY Mellon,
Pune, India
anuraag.khandagle@bnymellon.com

Rupesh Mishra
Manager,
OTC Derivatives Pricing, BNY Mellon,
Pune, India
rupesh.mishra@bnymellon.com

December 23, 2021

ABSTRACT

Stock markets go through ups and downs across various points in time of the economic cycle. The conventional 60/40 allocation in equity and fixed income therefore need not be efficient across business cycles. Equities are volatile and are considered risky assets and a 60/40 portfolio would lead to a significant concentration in risk which may reap returns in bull run but can considerably hurt portfolios during recessionary periods. Considering this, risk parity portfolios make asset allocation based on the risk possessed by the underlying assets. Traditional risk parity portfolios make use of quadratic optimization to obtain weights which can be highly unstable due to the requirement of matrix invertibility. In this paper, we intend to overcome the requirement of matrix inversion by using these clustering algorithms for asset allocation and also compare them: Hierarchical Clustering and K-Medoids which make use of covariance matrix to form clusters and overcome the instability caused due to matrix inversion. The paper also suggests rebalancing strategy to adjust weights according to the updated market regime. We perform the tests on the Indian equity index, commodities index and government security index over a period of around two years. The performance of the algorithms is measured using Sharpe Ratio, Maximum Drawdown, Maximum Drawdown Duration and Standard Deviation. Our analysis shows that Hierarchical Clustering not only segregates the assets into respective sectors but also gives higher risk adjusted returns and least maximum drawdowns.

***K*eywords** Risk Parity · Clustering algorithms · Trading Strategies

1 Introduction

Asset allocation is one of the major challenges faced by asset managers while devising a strategy. For a strategy, its crucial to ensure that capital is not overly concentrated in a particular asset. One such example of concentration is the traditional 60/40 portfolio, which although budgets sixty percent of capital to high-risk equities but actually ends up allocating more than ninety percent based on the risk of underlying assets. This is where risk parity comes in, which makes allocation based on the risk of the assets in the portfolio. Traditional risk parity strategy obtains weights by solving a convex optimization problem [1]. The weights become unstable when the covariance matrix consists of correlated assets making them vulnerable to large errors even with minor shifts. The paper is intended to showcase how clustering algorithms overcome this problem and compare with each other. The paper is structured as follows. In section 2, we describe risk parity strategy along with the trading framework. Section 3 describes the mathematical background of the clustering algorithms used. In section 4, we present the data used. Section 5 describes the implemented trading strategies along with the results. Conclusions are provided in section 6.

2 Strategy Framework

2.1 What is Risk Parity?

Risk parity strategy is a risk-based diversification strategy which allocates weights to different asset classes based on the inherent risk possessed by a particular asset. Traditional 60/40 portfolio allocates equal dollar value to the constituent assets whereas risk parity portfolios allocate weights such that the risk contribution by different assets is equal.

For example, a typical risk parity strategy would allocate significant weight to fixed income since they have been less volatile historically. A higher concentration in equities leads to its domination in the entire portfolio. The result is that even though such portfolios may outperform during a bull run, but can also lead to huge losses during economic downturns. Since, various asset classes tend to behave differently across different asset classes, risk-parity strategy is bound to outperform across economic cycles. For instance, equities do well in high growth and low inflation environments, bonds do well in deflationary or recessionary environments, and commodities tend to perform best during inflationary[2].

The main advantage of risk parity portfolios is that the investors get better risk adjusted returns. In long term, such portfolios let the investor get paid equivalent returns while bearing less amount of risk. The following section describes our strategy framework.

2.2 Strategy Framework

The entire strategy is divided into the following four steps:

1. Data Preparation: Our clustering algorithms take asset correlation distance matrix as our input. This requires us to calculate correlation matrix and convert it into a distance matrix. Let $\sum$ represent the NxN correlation matrix of N assets.

$$\sum = \begin{bmatrix} \rho_{11} & \dots & \dots & \rho_{1N} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \rho_{N1} & \dots & \dots & \rho_{NN} \end{bmatrix} \tag{1}$$

   The, correlation distance matrix is given by D with

$$d_{i,j} = 0.5 * \sqrt{1 - \rho_{i,j}}, \forall i, j \in [1, N] \tag{2}$$

$$D = \begin{bmatrix} d_{11} & \dots & \dots & d_{1N} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ d_{N1} & \dots & \dots & d_{NN} \end{bmatrix} \tag{3}$$

2. Cluster formation: The data obtained in the above format is used as an input in the clustering algorithms to obtain clusters of assets.
3. Obtaining weights using inverse variance: In each of the above clusters, we use inverse variance method to obtain weights. Therefore, weight for an asset with n assets in a cluster is given by,

$$w_i = \frac{\frac{1}{\sigma_i}}{\sum_1^n \frac{1}{\sigma_i}} \tag{4}$$

   where σ is the volatility of the asset. Using the above formula, weight of asset 1 in a two asset cluster would be,

$$w_1 = \frac{\frac{1}{\sigma_1}}{\frac{1}{\sigma_1} + \frac{1}{\sigma_2}}$$

4. Portfolio rebalancing using volatility index data, VIX. Since the weights we obtain are dependent on volatility, we use VIX as an indicator to identify if the market has moved considerably.

3 Background

3.1 Hierarchical Clustering

The hierarchical structure of complex financial systems was first investigated by the Nobel prize winner Herbert Simon in 1991 in his paper, The Architecture of Complexity [3]. In the paper, Simon describes how financial systems exhibit

hierarchical structures and how it makes sense to organise such systems into subgroups while finding solution to problems. Traditional method of asset allocation in a risk parity portfolio involves solving a convex optimisation problem. This approach requires matrix inversion and can cause significant estimation errors in case of correlated assets.

Marcos Lopez de Prado proposed the idea of asset allocation by assuming a hierarchical structure [4]. The method makes use of correlation distance matrix and avoids matrix inversion. The overall hierarchical risk parity framework is divided into three steps as described by Marcos[4] in his book: 1. Tree Clustering 2. Quasi Diagonlization and 3. Recursive Bisection.

1. **Tree Clustering**
   The python implementation of hierarchical clustering through the linkage function of the scipy library takes correlation distance matrix as input and allows clustering of assets belonging to same sector to be grouped as nodes of the same branch. The grouping of the assets can be best described using the following example. Consider the following correlation distance matrix, A, obtained using equation 2 with total 6 assets:

$A =$

	A1	A2	A3	A4	A5	A6
A1	0					
A2	0.23	0				
A3	0.22	0.15	0			
A4	0.34	0.20	0.15	0		
A5	0.34	0.14	0.28	0.29	0	
A6	0.23	0.25	0.11	0.22	0.39	0

Now follow the below given steps:
Step 1: Find the minimum distance between the assets. In our case A3 and A6 have the least distance of 0.11. A3 and A6, therefore form the first cluster.
Step 2: The next step is to update the distance matrix by recalculating the distance between the new cluster and the remaining points using the "single" linkage method.
MIN (dist(A3,A6), A1) = MIN(dist(A3,A1), dist(A6,A1))= MIN(0.22,0.23)= 0.22
Similarly, MIN(dist(A3,A6),A2)=0.15
MIN(dist(A3,A6),A4)=0.15
MIN(dist(A3,A6),A5)=0.28

The updated distance matrix A_1 after the 1^{st} iteration is as follows:

$A_1 =$

	A1	A2	A3,A6	A4	A5
A1	0				
A2	0.23	0			
A3,A6	0.22	0.15	0		
A4	0.37	0.20	0.15	0	
A5	0.34	0.14	0.28	0.29	0

Linkage function also enables user to obtain a Dendrogram. Dendrogram displays the hierarchical relationships between the assets using a tree like structure. The nodes at the bottom of a branch represent the cluster members, while the length of the branch represents the distance between two members of the cluster. Higher the length, less is the closeness between the members. In our case the dendrogram for the 2 member cluster is shown in figure 1.

Repeating the steps 1 and 2 gives the matrix A_n in the n^{th} iteration. The final dendrogram is given in figure 2.

$A_n =$

	A1	((P2,P5),(A3,A6),P4)
A1	0	
((P2,P5),(A3,A6),P4)	0.22	0

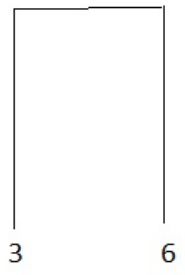

Figure 1: Cluster obtained after the first iteration.

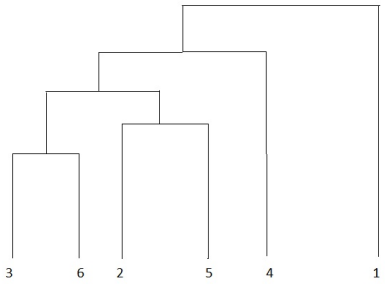

Figure 2: Cluster obtained after the final iteration.

2. **Quasi Diagonalization**

   The linkage matrix obtained in the previous step is a (N-1)x4 dimensional matrix. The first 2 columns consists of the cluster numbers, the third column represents the distance between the clusters and the last column represents the number of original assets in the cluster [5]. The output description of the linkage matrix is also give in the Scipy documentation. Quasi diagonalisation is the process of obtaining the sorted original unclustered assets in a given cluster. The process works according to the following pseudo code:

   (a) Let z be the linkage matrix. Store the first two columns of the last row of the linkage matrix in a variable, $sortedoutput = [z[N-1,0], z[N-1,1]]$

   (b) Get the indices which have entries greater than the total assets in our portfolio, $index = sortedoutput[sortedoutput > N].index$
   $values = sortedoutput[sortedoutput > N].values - N$

   (c) Update the $sortedoutput$ variable using the above obtained arrays in the following way.
   $sortedoutput[index] = sortedoutput[values, 0]$
   $sortedoutput[index] = sortedoutput[values, 1]$

   (d) Sort the above obtained array based on index.

   (e) Repeat step (b), (c) and (d) until there is no item remaining in $sortedoutput$ greater than N.

3. **Recursive Bisection**
   Recursive Bisection is the process of obtaining weights using the Inverse variance method mentioned in section 2.2. The following algorithm takes the covariance matrix and the $sortedoutput$ array obtained in the previous step and outputs the weights. The algorithm is as follows:

   (a) Take the $sortedoutput$ array and break each of its components into 2 halves, $c_1 1$ and c_2.

   (b) Calculate variance for c_1 and c_2 as v_1 and v_2.

   (c) Calculate the weight for each cluster as $w_1 = \frac{\frac{1}{v_1}}{\frac{1}{v_1}+\frac{1}{v_2}}$.

   (d) Repeat steps (b) and (c) for each cluster in $sortedoutput$

Table 1: Distance of non-medoids from cluster centers.

Points	**Distance from Cluster C1**	**Distance from Cluster C2**
A1	0.23	0.22
A2	-	-
A3	-	-
A4	0.20	0.15
A5	0.14	0.28
A6	0.25	0.11

(e) Repeat (a) until the array $sortedoutput$ is broken into an array of the size equal to the number of total assets.

3.2 K-Medoids Clustering

3.2.1 K-Medoids algorithm

The k-medoids algorithm is a clustering algorithm related to the k-means algorithm and the medoidshift algorithm. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups). K-means attempts to minimize the total squared error, while k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars). Another advantage of k-medoids algorithm is that is takes pairwaise distance or correlation distance matrix as input unlike k-means algorithm. The clusters are obtained using the famous algorithm, Partitioning Around Medoid (PAM) proposed by Kaufman and Rousseeuw [6]. For our strategy we have used the K-medoids implementation of the pyclustering library in Python. The algorithm works in the following way:

1. Pass an array of points to initialize medoids.
2. Assign the remaining points to the medoids defined in the previous step based on the pairwise distances obtained from the correlation distance matrix.
3. For each medoid do the following.
   (a) For each non medoid, swap the medoid with the non medoid point and compute the cost.
   (b) Keep the swap if the cost is less, else revert the swap.

The following example will illustrate the working of K-medoid algorithm. Consider the same correlation distance matrix given in section 3.1

$A =$

	A1	A2	A3	A4	A5	A6
A1	0					
A2	0.23	0				
A3	0.22	0.15	0			
A4	0.34	0.20	0.15	0		
A5	0.34	0.14	0.28	0.29	0	
A6	0.23	0.25	0.11	0.22	0.39	0

Let $A2$ and $A3$ be the initialized medoids. The points assigned to the cluster will be as follows:

Looking at the table above we can see that points $A1$, $A4$ and $A6$ belong to cluster $C2$ and $A5$ belongs to cluster $C1$. Total cost is given by $P = 0.22 + 0.15 + 0.14 + 0.11 = 0.62$. Refer table 1.
As per the algorithm steps, we will now fix one medoid and swap the other medoid with the non-medoid points. Lets fix $A3$ and swap $A2$ with $A1$. The updated matrix can be seen in table 2.

The updated Table 2 assigns all the points to cluster 2. However, the cost P gets updated to 0.69. The updated costs is greater than the cost obtained earlier, i.e. 0.62. We will thus revert the swap.

The process goes on until we get the minimum cost for set number of clusters.

Table 2: Distance of non-medoids from cluster centers after the first swap.

Points	**Distance from Cluster C1**	**Distance from Cluster C2**
A1	-	-
A2	0.23	0.15
A3	-	-
A4	0.34	0.15
A5	0.34	0.28
A6	0.23	0.11

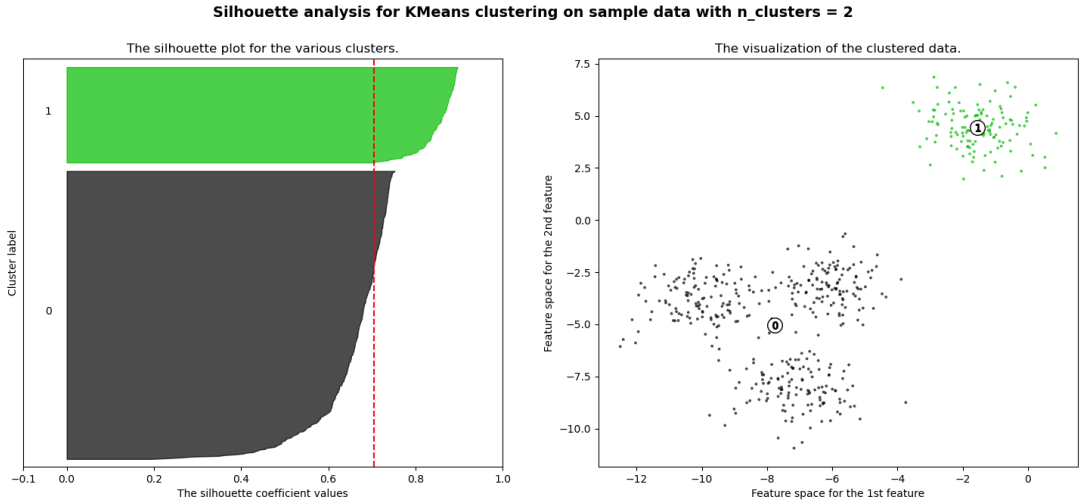

Figure 3: Silhouette score plot for a sample data with k=2.

3.2.2 Finding the optimal value of K

In order for us to cluster the assets into K groups, its important to determine the value of K. Silhouette score is one of the ways to identify the optimal number of clusters present in the data. Silhouette score lies in the range of -1 to 1 with 1 representing the most appropriate clustering configuration. The score tells us how closely an asset is to a cluster when compared to other clusters.
In order to calculate this value, we measure two distances for a point. First, the average distance of a point from its cluster members, a_p and second, the average distance of a point from the member of the other clusters, b_p.
For a point p in cluster C_i, a_p is given by,

$$a_p = \frac{1}{N_i - 1} \sum_{k \in C_i} d_p(k) \tag{5}$$

where N_i is the number of points in a cluster C_i and $d_p(k)$ is the distance between a fixed point p and point k. Similarly, consider a point $p \in$C$_i$, then b_p is given by,

$$b_p = \underset{C_k \neq C_i}{min} \frac{1}{N_k} \sum_{k \in C_k} d_p(k) \tag{6}$$

Silhouette score for a point p in cluster C_i is given by,

$$s_p = \frac{b_p - a_p}{max(b_p, a_p)} \tag{7}$$

The final silhouette score is the average of the scores obtained across the entire dataset. The plots in figure 3 and figure 4 taken from the sklearn documentation [7] is an illustration of how the silhouette scores vary upon changing the value of k. The dotted red line represent the average score across the dataset.

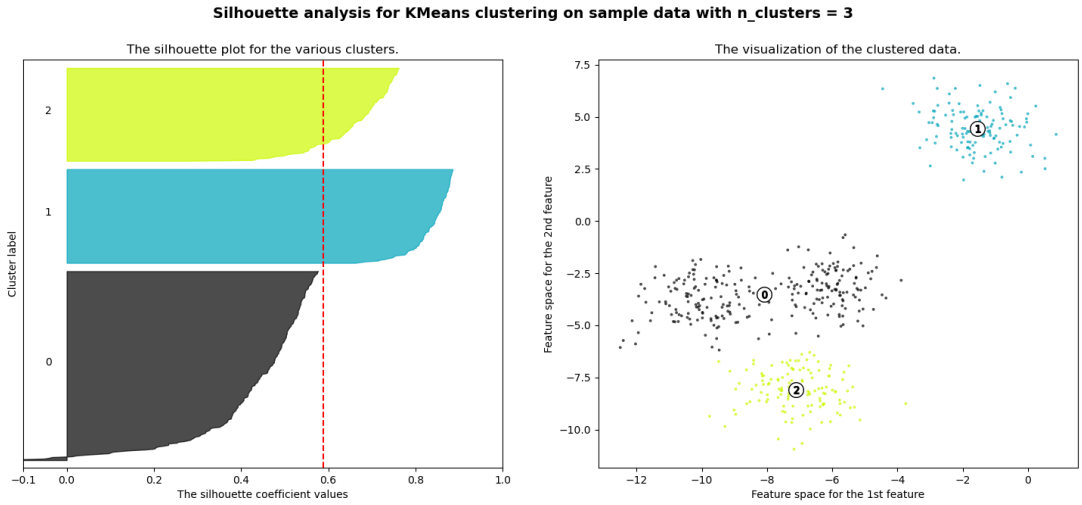

Figure 4: Silhouette score plot for a sample data with k=3.

3.2.3 Weight calculation

The calculation of weights in K-medoids is similar to the method we saw in hierarchical clustering. In hierarchical clustering, the weights are obtained recursively by dividing the $sortedoutput$ array into 2 halves. This approach is intuitive since in hierarchical clustering the clusters are obtained in pair, represent the nodes of a branch. However, in K-medoids a cluster can have more than 2 members. Let C_i represent the i^{th} cluster with N_i assets. The weights of the assets within the cluster is given by:

$$w_{C_i} = \begin{bmatrix} \frac{\frac{1}{\sigma_1}}{\sum_1^{N_i} \frac{1}{\sigma_i}} \\ \frac{\frac{1}{\sigma_2}}{\sum_1^{N_i} \frac{1}{\sigma_i}} \\ .. \\ .. \\ .. \\ \frac{\frac{1}{\sigma_{N_i}}}{\sum_1^{N_i} \frac{1}{\sigma_i}} \end{bmatrix} \tag{8}$$

Cluster variance is given by:

$$\sigma_{C_i} = w_{C_i}^T \sum_{C_i} w_{C_i} \tag{9}$$

where $\sum_{C_i}$ is the covariance matrix of the C_i cluster. The final weights of assets in a cluster is given by:

$$W_{C_i} = \alpha_{C_i} * w_{C_i} \tag{10}$$

where α_{C_i} is $\frac{\frac{1}{\sigma_{C_i}}}{\sum_1^k \frac{1}{\sigma_{C_k}}}$.

3.3 Performance Metrics

The strategies are assessed based on the following three performance metrics:

1. **Sharpe Ratio**
   Sharpe ratio measures the risk adjusted return of an investment strategy, where risk is represented in the form of standard deviation of the asset excess returns. Sharpe ratio can also be interpreted as the slope of the Capital Allocation Line (CAL). [8].

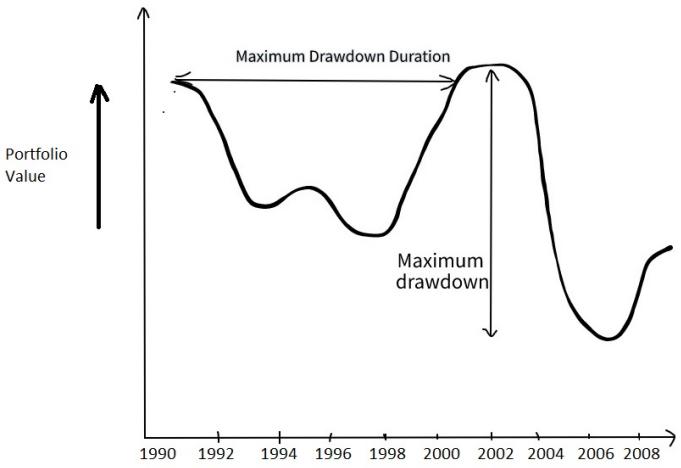

Figure 5: Illustration of difference between Maximum drawdown and Maximum drawdown duration.

$$SharpeRatio, S = \frac{E[R - r_f]}{\sqrt{var(R - r_f)}} \tag{11}$$

In our analysis, we first calculate the average of daily returns and divide it by the standard deviation of returns. We multiple this value by $\sqrt{252}$ to obtain annaualized Sharpe Ratio.

2. **Maximum drawdown**
Maximum drawdown is the maximum loss incurred with respect to highest value observed before achieving the next peak value. Following is the formula for maximum drawdown(MDD):

$$MDD = \frac{TroughValue - PeakValue}{PeakValue} \tag{12}$$

3. **Maximum drawdown duration:**
Maximum drawdown duration is the longest of all the worst times observed between peak values. An illustration of the maximum drawdown and maximum drawdown duration is given in figure 5. The calculation of Maximum drawdown duration and Maximum drawdown is well explained in the famous book by Ernest P. Chan [9].

4 Data description

We have tested the strategies on End of Day data from Indian market across three asset classes, Equity, Fixed income and Commodities. We have used the companies listed on the National Stock Exhchange(NSE). Apart from this, the portfolio is re-balanced using Volatility Index (VIX) data as a signal. The data used in our backtesting ranges from 6^{th} January 2017 to 31^{st} December 2019.

1. **Equity:**
For equity we have chosen the individual stocks of the benchmark index NIFTY 50 consisting of 50 Indian large cap stocks. The stocks belong to a wide range of sectors. Table 3 lists out the constituent stocks included in our portfolio along with their sectors. The sectoral distribution in the NIFTY 50 index is given in figure 6. The data has been download from the following NIFTY website.

`https://www1.nseindia.com/products/content/equities/indices/historical_index_data.htm`

2. **Fixed Income:** For fixed income we have used the NIFTY Composite G-Sec index.NIFTY Composite G-Sec index is constructed using the prices of top 10 (in terms of traded value) liquid GOI bonds with residual maturity greater than 1 year and having outstanding issuance of over Rs.5000 crores. The individual bonds

Table 3: Constituent companies in NIFTY 50 index along with their weights.

Sector	**Company**	**Weight(%)**
Automobile	Bajaj Auto Ltd.	0.74
	Hero MotoCorp Ltd.	0.58
	Eicher Motors Ltd.	0.51
	Mahindra and Mahindra Ltd.	0.76
	Maruti Suzuki India Ltd.	1.59
	Tata Motors Ltd.	0.36
Cement	BajajGrasim Industries Ltd.	0.52
	Shree Cement Ltd. Ltd.	0.65
	UltraTech Cement Ltd.	1.05
Cigarettes	ITC Ltd.	4.18
Consumer Goods	Hindustan Unilever Ltd.	4.59
	Britannia Industries Ltd.	0.88
	Nestle India Ltd.	1.62
	Titan Company Ltd.	1.09
	Asian Paints Ltd.	2.10
Energy	Oil and Natural Gas Corporation Ltd.	0.70
	NTPC Ltd.	1.14
	Power Grid Corporation of India Ltd.	1.14
	Bharat Petroleum Corporation Ltd.	0.71
	Indian Oil Corporation Ltd.	0.58
	Reliance Industries Ltd.	10.06
	GAIL (India) Ltd.	0.4
Financial Services	Axis Bank Ltd.	2.39
	HDFC Bank Ltd.	10.42
	ICICI Bank Ltd.	5.85
	IndusInd Bank Ltd.	0.59
	Kotak Mahindra Bank Ltd.	4.85
	State Bank of India	2.11
	Bajaj Finance Ltd.	1.64
	Bajaj Finserv Ltd.	0.78
	Housing Development Finance Corporation Ltd.	7.88
Engineering	Larsen and Toubro Ltd.	2.79
Fertilizer	UPL Ltd.	0.50
Media and Entertainment	Zee Entertainment Enterprises Ltd.	0.32
Shipping	Adani Ports and Special Economic Zone Ltd.	0.54
Information Technology	HCL Technologies Ltd.	1.32
	Infosys Ltd.	6.56
	Tata Consultancy Services Ltd.	5.36
	Tech Mahindra Ltd.	0.98
	Wipro Ltd.	0.82
Metals and Mining	Hindalco Industries Ltd.	0.39
	Vedanta Ltd.	0.33
	JSW Steel Ltd.	0.41
	Tata Steel Ltd.	0.57
	Coal India Ltd.	0.82
Pharma	Cipla Ltd.	0.60
	Dr. Reddy's Laboratories Ltd.	1.06
	Sun Pharmaceutical Industries Ltd.	1.06
Telecom	Bharti Infratel Ltd.	0.38
	Bharti Airtel Ltd.	2.75

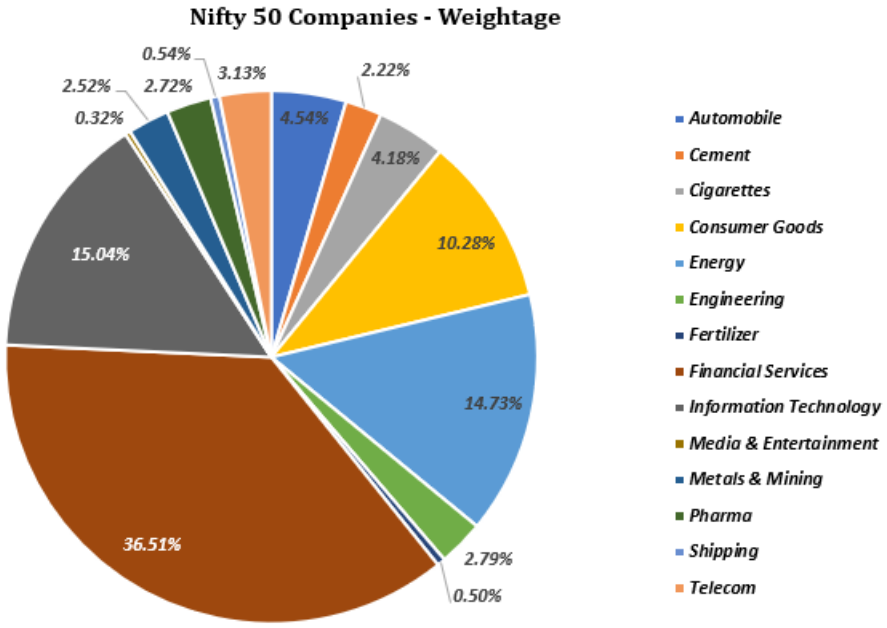

Figure 6: Sectoral distribution of companies in NIFTY 50 index.

are assigned weights considering the traded value and outstanding issuance in the ratio of 40:60. The index measures the changes in the dirty prices of the bond basket. The methodology and the index constituents are well explained in this link.

`https://www1.nseindia.com/products/content/equities/indices/GSEC_3.htm`

3. **Commodities:**
   The NIFTY Commodities Index is designed to reflect the behaviour and performance of a diversified portfolio of companies representing the commodities segment which includes sectors like Oil, Petroleum Products, Cement, Power, Chemical, Sugar, Metals and Mining. The NIFTY Commodities Index comprises of 30 companies that are listed on the National Stock Exchange (NSE). The sectoral distribution of the NIFTY commodities index constituents is given in table 4
4. **Volatility Index (India VIX)**
   India VIX is a volatility index computed by NSE based on the order book of NIFTY Options. For this, the best bid-ask quotes of near and next-month NIFTY options contracts which are traded on the FO segment of NSE are used. India VIX indicates the investors perception of the markets volatility in the near term i.e. it depicts the expected market volatility over the next 30 calendar days. Higher the India VIX values, higher the expected volatility and vice-versa. The methodology to obtain volatility is described in the following link:

   `https://www1.nseindia.com/content/indices/white_paper_IndiaVIX.pdf`

   The Day-over-Day change values of VIX data is used as a signal for rebalancing the portfolio. Figure 7 shows the histogram of the VIX day-over-day change values. We have used the change values corresponding to 2.5th and 97.5th percentile as a criteria for rebalancing the portfolio. The percentile values correspond to around 1.5% change which suggests that our portfolio will get rebalanced whenever the VIX values changes by 1.5%.

5 Results

We have conducted backtesting on data ranging from 6^{th} January 2017 to 31^{st} December 2019 covering 734 market days. In this section we will present the clusters formed and their performance metrics over the period mentioned above.

1. **Hierarchical Clustering**
   Figure 8 represents the clusters formed using the Hierarchical clustering. The nodes representing the various clusters are highly distinguished and classified into their respective sectors. Fixed income index has clearly

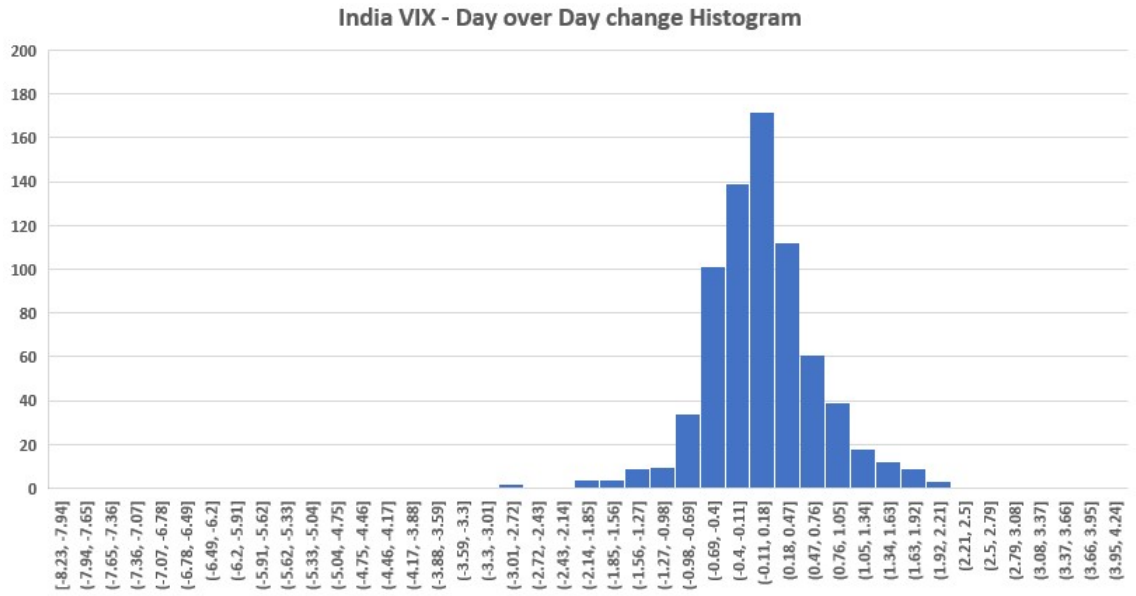

Figure 7: Distribution of India VIX Day-over-Day change values.

Table 4: Constituent sectors in NIFTY Commodities index along with their weights.

Sector	**Weight(%)**
Metals	29.09
Cement	21.91
Oil and Gas	21.35
Power	12.21
Chemicals	9.12
Fertilizers	6.31

been classified into a separate node from all the equity stocks and commodities. Further, we can also see how Pharma stocks like Sun Pharma, Cipla and Dr. Reddy are clustered together under the same branch. In table 4 we can see that the NIFTY Commodities index is majorly concentrated with metal stocks. Our dendrogram validates this by clustering the metal stocks with the commodities index.

2. **K Medoids Clustering**
   As earlier discussed, we have used the silhouette score to find the optimal value of K. As can be seen in figure 9, the maximum average silhouette score appears is 0.236 when k=5. The clusters are well distinguished when the score is close to 1. The low silhouette score is expected considering the presence of several correlated stocks in our portfolio. In table 5 we can see that K-Medoids has not been successful to segregate the stocks for all the sectors separately. However, it has been able to differentiate between the Tech, Pharma, Bank and Fixed income assets.

In table 6 we can see that the highest sharpe ratio is obtained with Hierarchical clustering technique.

6 Conclusion and future work

We can see through our analysis that clustering algorithms can provide a better solution to the problem of asset allocation. Hierarchical clustering not only segregates the individual companies as per their sectors but also helps in getting better risk adjusted returns. We also saw that how portfolio rebalancing can help the investor reduce drawdowns as well as maximum drawdown duration. Further, we observed how the traditional 60/40 portfolio also fails to provide diversification across asset classes and rather ends up allocating a significant capital to risky asset. K-Medoids clustering method is able to to classify certain sectors. However, it ends up combining 2 different industries together such as FMCG and Banks. The strategy to buy and hold NIFTY index can wipe out the portfolio value as can be seen from the obtained Maximum Drawdown of around 15%.

Table 5: Clusters obtained using K-Medoid clustering.

Cluster Number	Stocks
0	DRREDDY
1	HCLTECH, INFY, TCS, TECHM, WIPRO
2	HDFCBANK, HDFC, ITC, INDUSINDBANK, KOTAKBANK, HINDUNILVR
3	GSec Index
4	ADANIPORTS, ASIANPAINT, AXISBANK, BAJAJ-AUTO, BAJFINANCE, BAJAJFINSV, BPCL, BHARTIARTL, BRITANNIA, CIPLA, DIVISLAB,EICHERMOT, GRASIM, HEROMOTOCO, HINDALCO, ICICIBANK, IOC, JSWSTEEL, LT, M and M, MARUTI, NTPC, ONGC, POWERGRID, RELIANCE, SHREECEM, SBIN, SUNPHARMA, TATACONSUM, TATAMOTORS, TATASTEEL, TITAN, UPL, ULTRACEMCO, NIFTYCOMM, COALINDIA, NESTLEIND

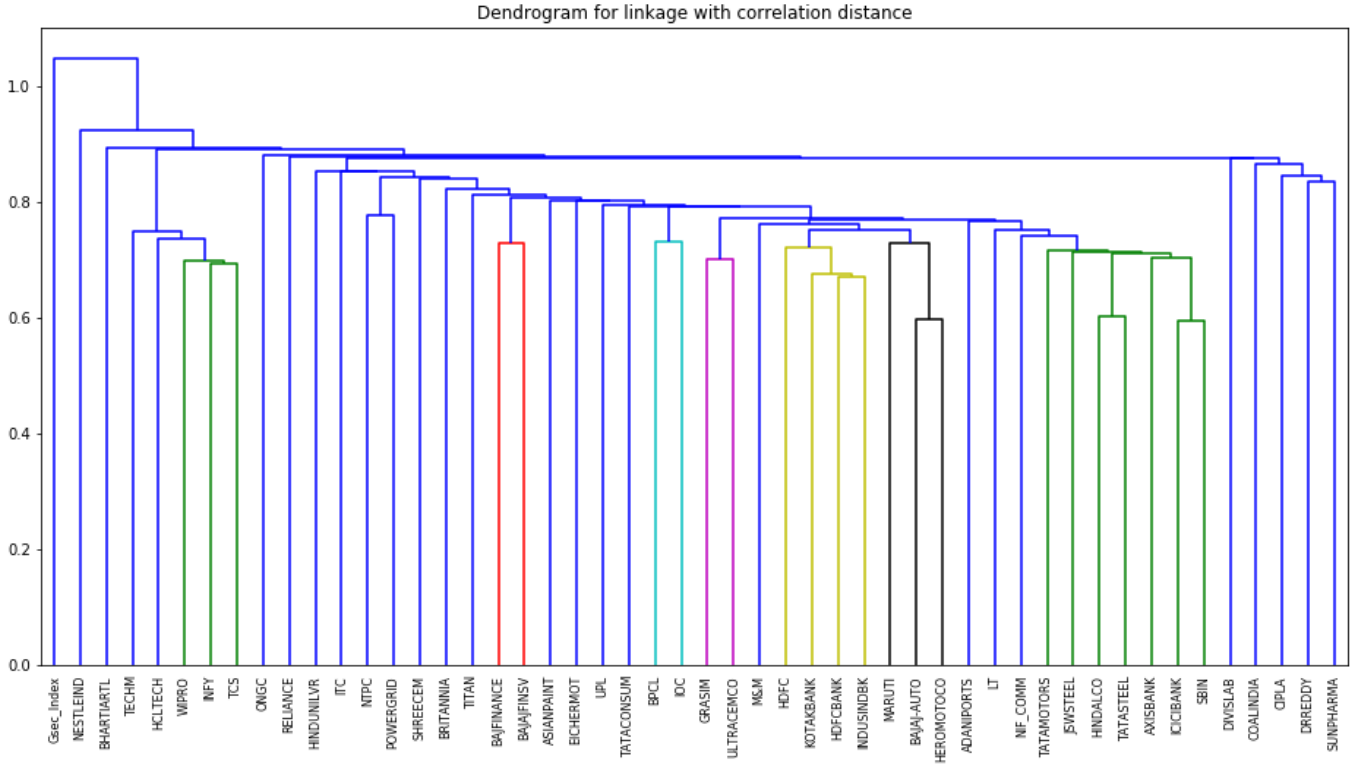

Figure 8: Cluster formation using Hierarchical clustering.

Table 6: Strategy comparison.

Metrics	Hierarchical Clustering	K-Medoid Clustering	60/40 Portfolio	NIFTY 50
Sharpe Ratio	2.36	2.28	1.64	1.16
Standard Deviation	5.37%	4.45%	10.01%	12.08%
Maxmimum Drawdown	-4.08%	-4.47%	-10.41%	-14.55%
Maximum Drawdown Duration	66	68	86	154

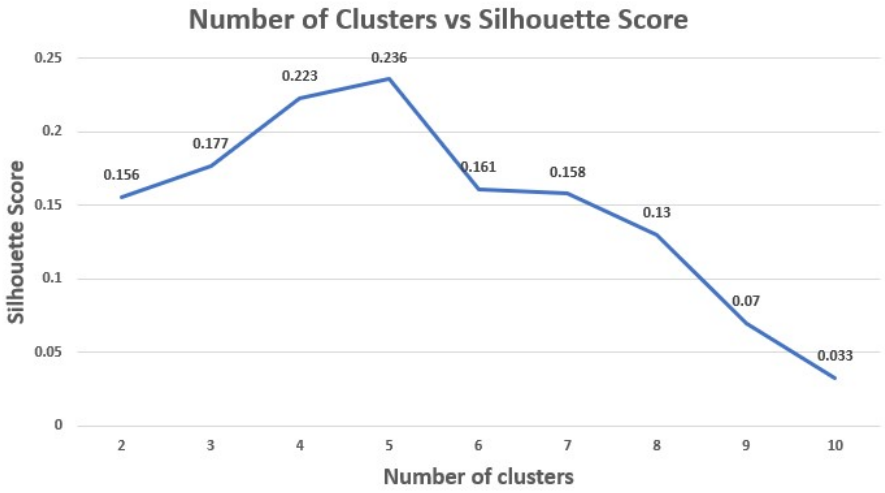

Figure 9: Number of clusters vs average Silhouette score

The current study doesn't really focus on the weakness of solving an optimisation problem for allocating assets. Our further study will therefore concentrate on asset allocation using traditional methods on Indian data and perform a stress test to assess how sensitive the weights are. We also aim to devise better portfolio rebalancing times to improvise drawdowns.

References

[1] Thierry Michel Emmanuel Jurczenko and Jerome Teiletche. Generalized risk-based investing. *SSRN Electronic Journal*, 2013.

[2] Brian K. HurstBryan JohnsonYao Hua Ooi. Understanding risk parity, 2010.

[3] Herbert A. Simon. The architecture of complexity. 1991.

[4] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley, 2018.

[5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[6] Leonard Kaufmann and Peter Rousseeuw. Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416, 01 1987.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] William F. Sharpe. The sharpe ratio. *The Journal of Portfolio Management*, 21(1):49–58, 1994.

[9] *Backtesting*, chapter 3, pages 31–67. John Wiley Sons, Ltd, 2012.