Biases of ML-Based Asset Allocation Methods

Introduction

"Diversification is protection against ignorance. It makes little sense if you know what you are doing." Warren Buffett once said.
Basically, the main point is not in the ignorance of the investor, and it can be reflected in the ignorance of the market! Imagine that the investor has discovered the right investment position, but the market is still moving in the opposite direction, in line with its mistake and current trend, as long as the market realizes its mistake and changes its trend and moves in the desired direction of the investor; Therefore, in any case, portfolio diversification in investment is the most logical act of an investor.
Portfolio management can be considered as an art, the art and science of selecting a group of assets, portfolios of assets or even different trading strategies or algorithms, which meet some short- and long-term financial goals according to the degrees of risks that customers take. Therefore, this process should be such that helps the portfolio manager to achieve maximum financial return and optimum level of risk.
At DFO's portfolio team, we want to present and challenge this important process, by comparing the latest asset allocation methods in the financial world, and present it in some articles, because many studies have proved it to us that methods such as modern portfolio theory and the Black Litterman model perform even worse than selecting and weighing by monkeys in out-sample data.
As you may know, Dr. López de Prado has presented two methods, HRP and NCO, for finding the optimal allocation of financial assets in an investment portfolio in 2016 and 2020, respectively. These methods made significant advances in proposing some solutions for major concerns of quadratic optimizers in general and Markowitz's CLA in particular (in Markowitz 's modern portfolio theory) such as: instability, concentration and underperformance in out-sample data. However, in this article, we examine the biases and weaknesses of these methods by providing some examples. In this regard, we will first briefly explain the implementation steps of each of these two methods by providing their codes in Python.

How These Methods Works?

Hierarchical Risk Parity (HRP) and Nested Clustered Optimization (NCO):

Hierarchical Risk Parity (HRP) introduced by Dr. López de Prado (2016) utilizes developments in machine learning and graph theory to bypass the need for inversion of the covariance matrix, a leading issue in estimation of weights in portfolio optimization. In this method, as you will see in the Python implementation section, if we want to explain it simply, it first sorts the assets based on their degree of similarity based on the correlation matrix. Then, in a few steps, each of these sorted lists divide the assets into two equal number clusters, and appropriate weight is assigned to each cluster, inverse proportional to the variance of each cluster.

The Nested Clustered Optimization (NCO) is the other ML-based portfolio allocation method designed by Dr. Lopez de Prado (2020), which wanted to tackle the source of Markowitz's curse. But first, before diving into details, it would be worth explaining this algorithm very simply, then the steps of the NCO algorithm and then finally, we will discuss the biases regarding the NCO algorithm. In this method, assets are first clustered based on their feature matrix, which in this method is a correlation matrix. Dr. de Prado has mentioned that if the number of historical observations of assets was much more than the number of assets, it would be better to first denoise the covariance matrix and then turn it into a correlation matrix. In the NCO method, Mr. de Prado uses the K-Mean method by applying some modifications for asset clustering. These modifications are to address two important limitations of this algorithm, i.e., the number of optimal clusters (K) and the initial values of the algorithm. Function `clusterKMeansBase ()` does this. Finally, in this function, the optimal number of clusters and the best starting points for the algorithm are determined by performing numerous iterations, which is extremely time-consuming. `clusterKMeansTop ()` Function is another version that double check the clusters. This function, after that first clusters are made, examines that if each of these resulting clusters has minimum appropriate score based on the average scores of the total clusters. This scoring is based on Silhouette method that will be discussed in detail in the Algorithm Description and Python implementation section. If this condition does not meet for some clusters, these clusters will be re-clustered separately and then will be attached to the other clusters of the previous step. After finding the appropriate clusters and arranging them, we see that the covariance matrix of the clusters as well as the covariance matrix of each cluster are almost in the form of a diagonal matrix, so we can say that we have some covariance matrix that are no longer ill-conditioned. Thus, we can find the weights by inversing the covariance matrix. This procedure is done in `optPort_nco ()` function.

Now, in the following section, we will discuss in some more detail the steps of the introduced methods and the biases that exist in some of these steps.
You can see the algorithm of these methods in the following diagrams, Figure 1 and Figure 2:

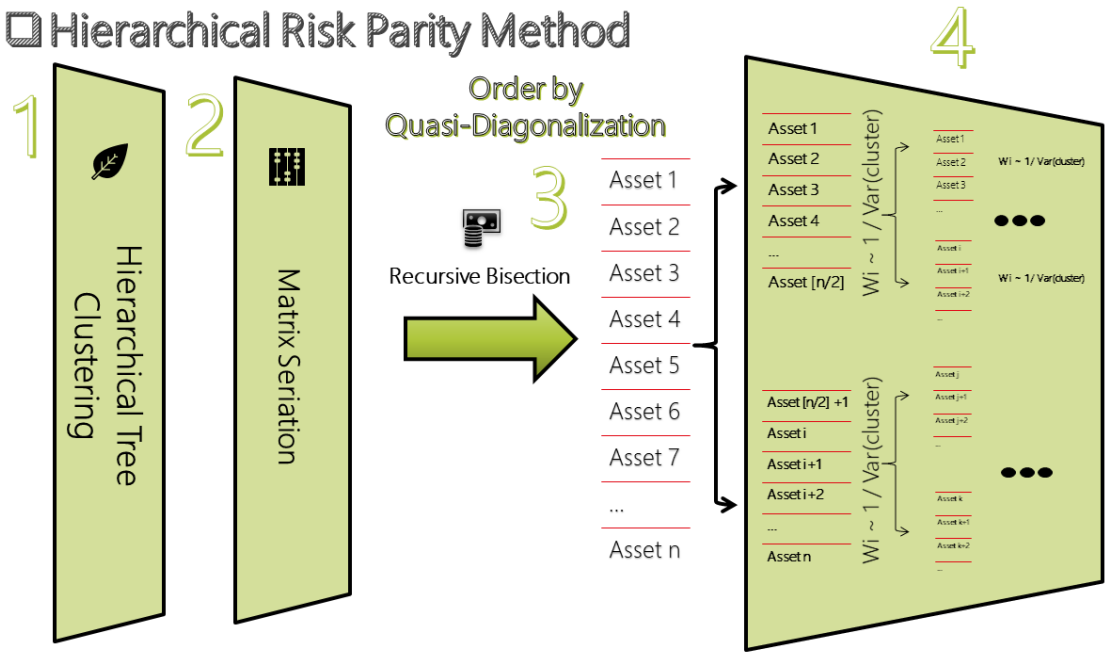


Figure 1- Hierarchical Risk Parity Method Explained

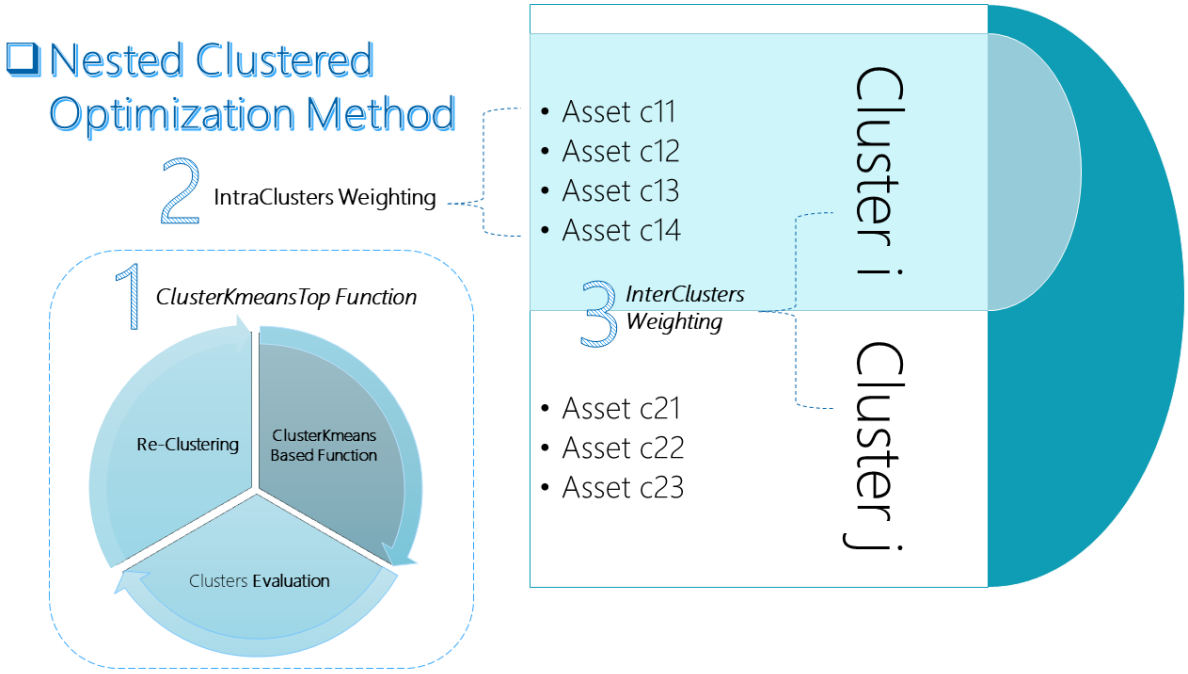


Figure 2- Nested Clustered Optimization Method Explained

A bit more detail about HRP and NCO

Actually, by relying on a more intricate algorithm, de Prado wanted HRP to avoid inverting the covariance matrix and be able to create efficient allocations for ill-conditioned and even singular covariance matrices (López de Prado, 2016) following a three main step process:

- Hierarchical clustering
- Matrix seriation
- Naive recursive bisection

In the first step we import required libraries for our algorithms and get some data from Yahoo Finance API. In our example we have got some different kind of tickers such as stocks: AMZN, F and cryptocurrencies: BTC-USD, ETH-USD and also some forex currency pairs like: EURUSD=X, GBPUSD=X and some other kinds of assets like: ^FVX, ^TNX, ^TYX, CL=F, ZC=F, CT=F, etc.

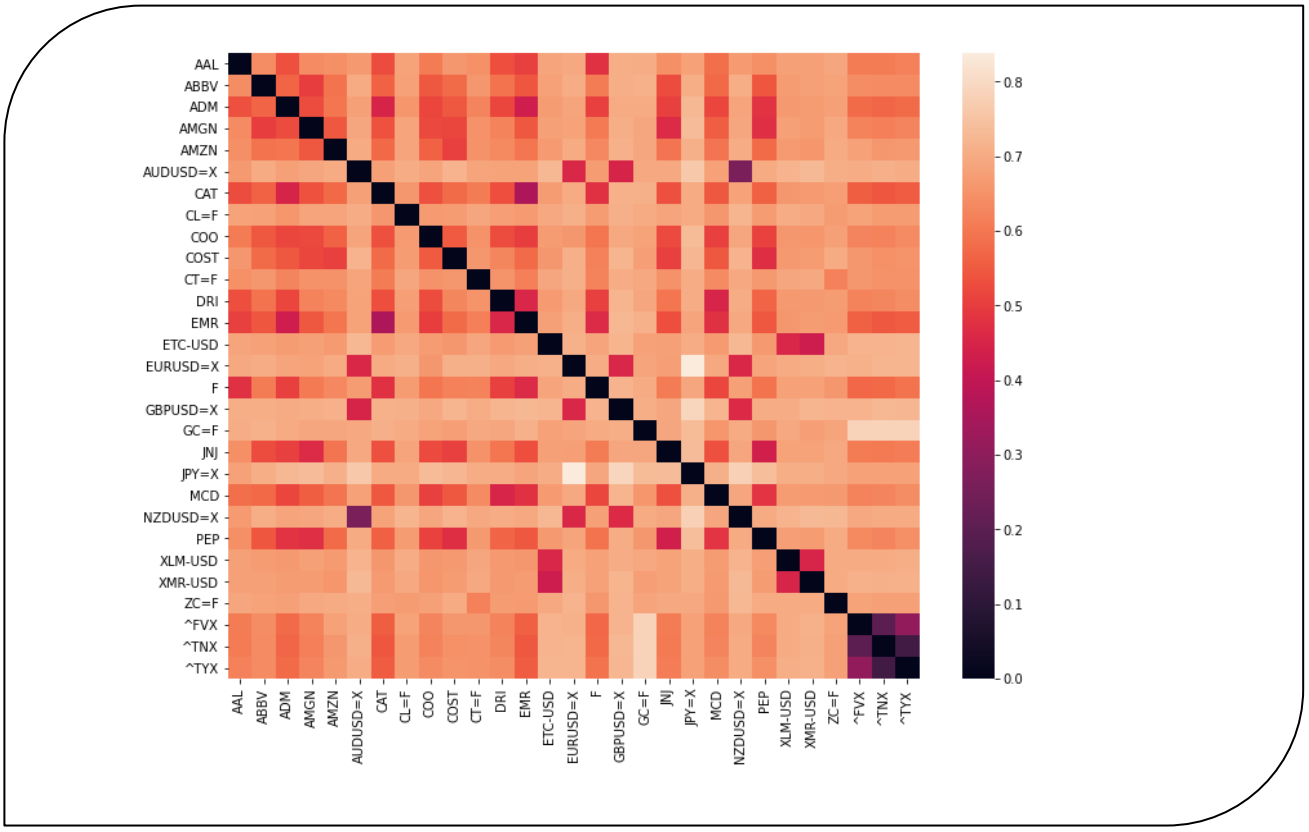
```
# Importing Libraries
import numpy as np , pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt , seaborn as sns
from scipy.cluster.hierarchy import linkage,dendrogram # For HRP
from sklearn.cluster import KMeans # For NCO
from sklearn.metrics import silhouette_samples # For NCO


import warnings
warnings.filterwarnings("ignore")
```

```
# Getting Data From YahooFinance
Tic = "AMZN MCD DRI F ADM COST PEP ABBV AMGN COO JNJ AAL CAT EMR XLM-
USD ETC-USD XMR-USD EURUSD=X GBPUSD=X JPY=X NZDUSD=X AUDUSD=X ^FVX
^TNX ^TYX GC=F CL=F ZC=F CT=F"
end_date = '2022-01-30'
start_date = '2016-01-01'
data = yf.download(tickers=Tic,
                   start=start_date,
                   end=end_date).Close.pct_change().dropna()
cov = data.cov()
Distance_corr = np.sqrt((1-data.corr())/2.)
del Tic , end_date , start_date , data
```

```
# First of all, Let's see a heatmap of this correlation matrix
plt.figure(figsize=(12,9))
sns.heatmap(Distance_corr)
plt.show()
```

Output >



✓ **Correlation Bias**

The first bias in these kinds of portfolio asset allocation models presented by López de Prado is the use of correlation as a feature matrix. Correlation is a useful criterion for showing a linear relationship, but it has no other capability than the one mentioned.

"It has become quite understandable today that the existence of a high correlation between the occurrence of two phenomena does not necessarily mean the existence of a cause-and-effect relationship between them. But in this case, the issue is that if there is a high correlation between data of the occurrence of two phenomena, it does not even mean that there is a real and meaningful correlation between these two!" Nassim Talib said.

For example, take a look at the chart below. It studies the relationship between the US spending on science, space, and technology and the rate of suicide by hanging, strangulation and suffocation. It is interesting to know that the correlation between these two phenomena has been more than 99% since the beginning of the 21st century. But is there any true and meaningful relationship between these two?

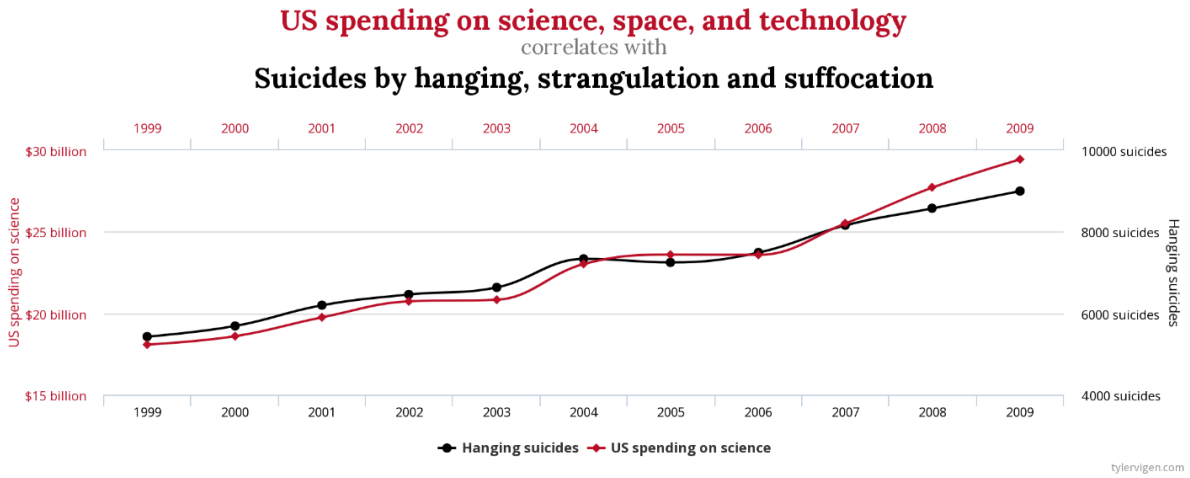


Figure 3

Anyway, it should be technically examined why correlation is not a good statistical measure to use as a tool to study the relationship between two variables. There are several technical limitations to correlation matrices that persuade us not to use them:

- The first point is that correlation is not essentially a metric, because it does not meet all three of its necessary main conditions, such as non-negativity, symmetry, and triangular inequality. As a result, comparing with non-metric measurements can lead us to completely inconsistent results.

- Correlation can be considered as a good tool only for measuring a very narrow linear relationship.

- And also, correlation is not a collectible measure. For example, correlation of 0.5 cannot be considered twice the correlation of 0.25, or correlation of 0.25 cannot be considered much more valuable than a correlation of 0. For example, take a look at These diagrams in figure 4:

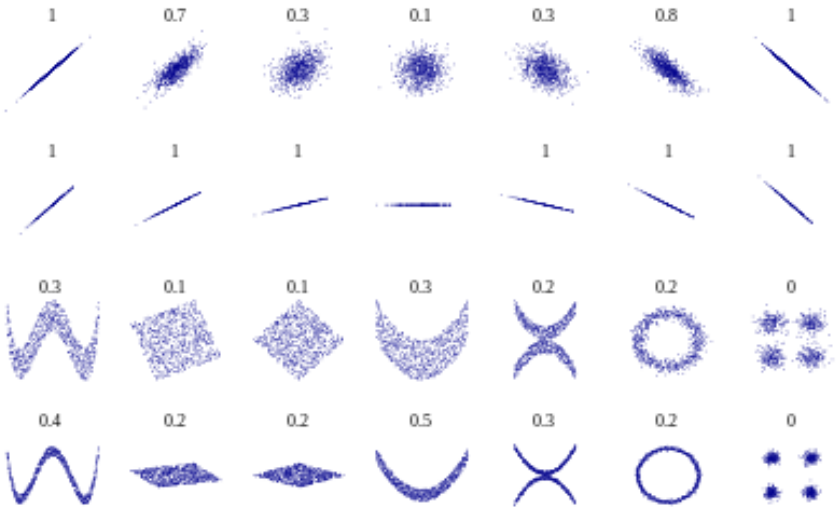


Figure 4- Some Example of correlations between two variables with its plot

Elsewhere, Nassim Talib once said about the information that can be inferred from the correlation between the two phenomena that:
“High correlation between two phenomena has no meaning but the diagram you see!”

1- Clustering

Anyway, if we accept that the correlation matrix can be used as a feature matrix for this problem, let's see how the clusters are formed if we cluster this feature matrix hierarchically with the Single Method. We used this method because it’s mentioned by López de Prado (2016) in his paper for the HRP method.
As you can see in Figure 5, it’s obvious that there are some clusters based on assets correlations. But you will see in final results of HRP that this algorithm does not follow this clustering logic in its weighting step.

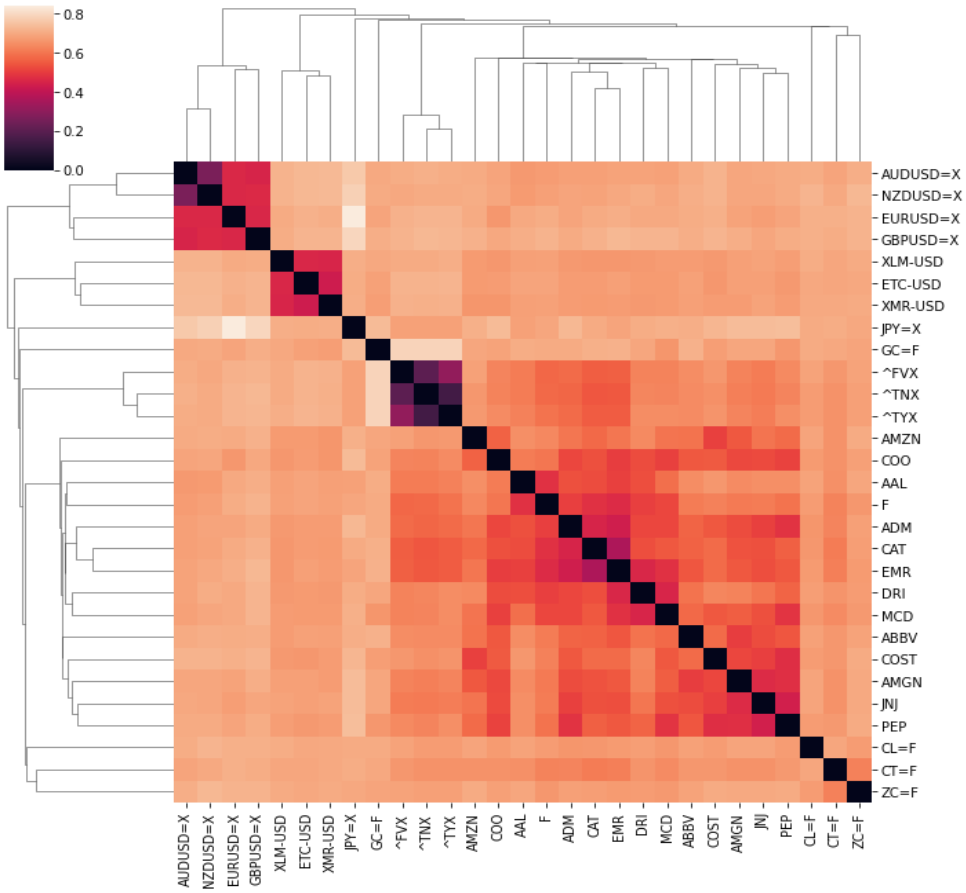


Figure 5- Hierarchical Clustering by Single Method

Dr. de Prado also uses K-mean Clustering method in his NCO algorithm and This is exactly where another bias, the clustering bias, originates from in these models.

- ✓ **Clustering Bias**

In the NCO algorithm, we cluster the correlation matrix by the K-means method. First of all, let's take a look at different types of clustering algorithms and also their strengths and weaknesses in different clustering environments. For instance, HDBSCAN clustering algorithms have strength in dense environments clustering; As density algorithms search for connected dense regions in the data space, they are better to use for 3D spaces.

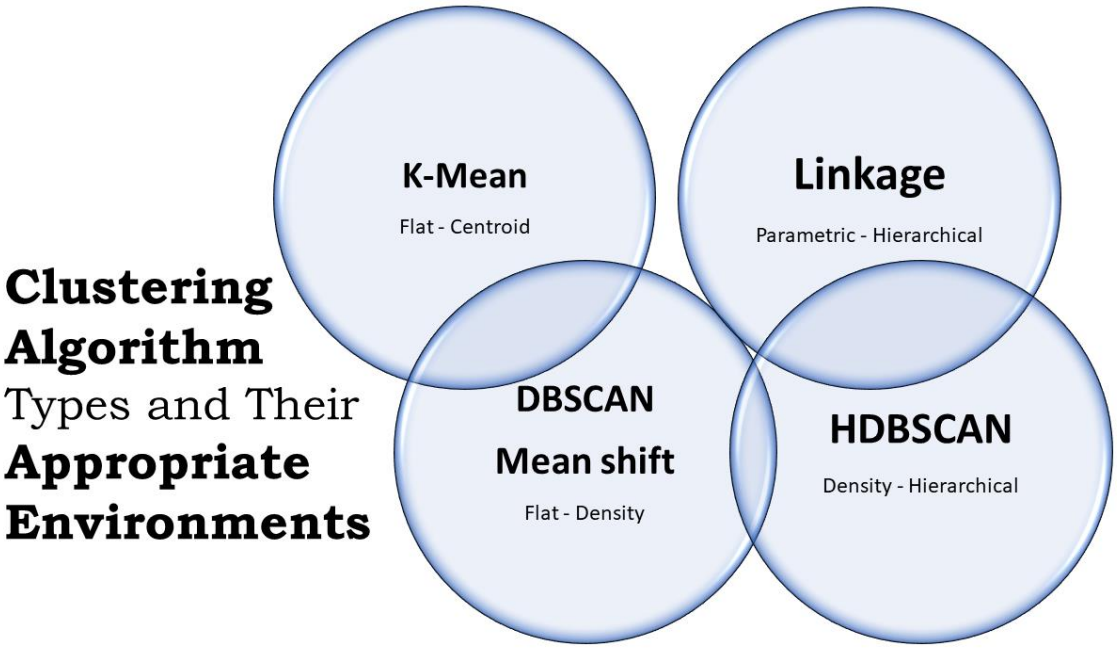


Figure 6- Different Types of Clustering Algorithms with Their Appropriate Environments

As it is illustrated in the image above, in Figure 6, the K-mean algorithm is better to use for Flat and Centroid types of data, and it is due to the K-mean algorithm notion and its steps of clustering. While on the other hand, the Correlation matrices are deemed as Parametric data types. This phenomenon is the reason why the NCO algorithm has inability to make proper clusters using correlation matrices.

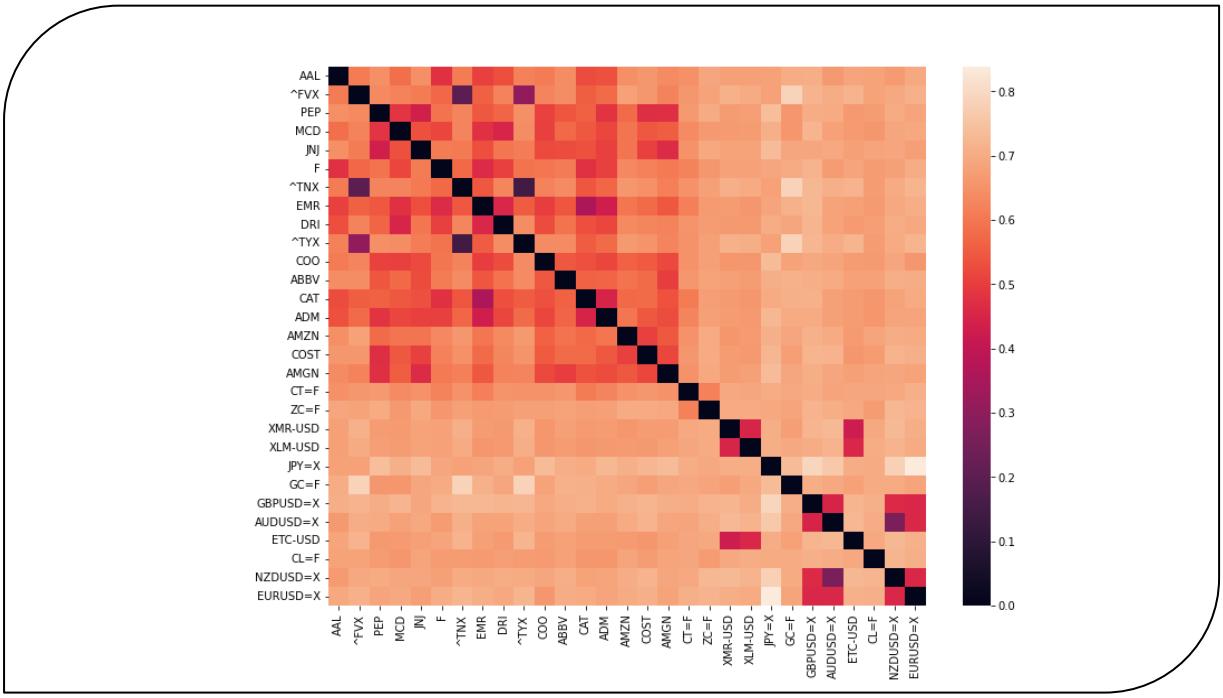
As mentioned before, NCO provides a strategy for addressing the effect of Markowitz's curse on an existing mean-variance allocation method. So, the first step of this algorithm is to cluster the correlation matrix using K-means with `clusterKMeansBase ()` function. This function involves finding the optimal number of clusters, appropriate algorithm initialization and also at the end evaluate if resulting clusters have minimum acceptable Silhouette scores or not. You can see the python implementation of this step, in code snippet bellow.

```
def clusterKMeansBase(corr0,maxNumClusters=10,n_init=10):
    x,silh=((1-corr0.fillna(0))/2.)**.5,pd.Series()
    for init in range(2,n_init+1):
      for i in range(2,maxNumClusters+1):
        kmeans_=KMeans(n_clusters=i,random_state=5,n_init=init)
        kmeans_=kmeans_.fit(x)
        silh_=silhouette_samples(x,kmeans_.labels_)
        stat=(silh_.mean()/silh_.std(),silh.mean()/silh.std())
        if np.isnan(stat[1]) or stat[0]>stat[1]:
          silh,kmeans=silh_,kmeans_
    newIdx=np.argsort(kmeans.labels_)
    corr1=corr0.iloc[newIdx]
    corr1=corr1.iloc[:,newIdx]
    clstrs={i:corr0.columns[np.where(kmeans.labels_==i)[0]].tolist() \
            for i in np.unique(kmeans.labels_) }
    silh=pd.Series(silh,index=x.index)
    return corr1,clstrs,silh
```

Then, resulting clustering is shown in the code snippet bellow.

```
corr_sort_KMeansBase,cluster_KMeansBase,_= \
                                        clusterKMeansBase(Distance_corr)

plt.figure(figsize=(12,9))
sns.heatmap(corr_sort_KMeansBase)
plt.show()
```



and also, the resulting clusters,

```
{
0: ['AAL', 'ABBV', 'ADM', 'AMGN', 'AMZN', 'CAT', 'COO', 'COST', 'DRI',
'EMR', 'F', 'JNJ', 'MCD', 'PEP', '^FVX', '^TNX', '^TYX'],
1: ['AUDUSD=X', 'CL=F', 'CT=F', 'ETC-USD', 'EURUSD=X', 'GBPUSD=X',
'GC=F', 'JPY=X', 'NZDUSD=X', 'XLM-USD', 'XMR-USD', 'ZC=F']
}
```

As it is shown in the resulting heatmap above, the result of the proposed function that uses K-Mean algorithm, the NCO is not able to cluster an extremely well-diversified portfolio consists of obviously different financial asset types with this clustering algorithm. To tackle this problem, we can use Linkage matrices, which are better for Parametric types. This method was used in HRP. But, in the end, even HRP algorithm did not use these clusters properly for weighting.

2 - Hierarchical Clustering and Matrix seriation (Quasi-Diagonalization) (For HRP)

Matrix seriation is a statistical method used to reorganize rows or columns of a matrix to enumerate them in an appropriate order. López de Prado (2016) uses this method as a way to rearrange the rows and columns of the covariance matrix into a more diagonalized representation by aligning the highest covariances along the diagonal. In this way, similar investments are placed closer together and dissimilar investments further apart.

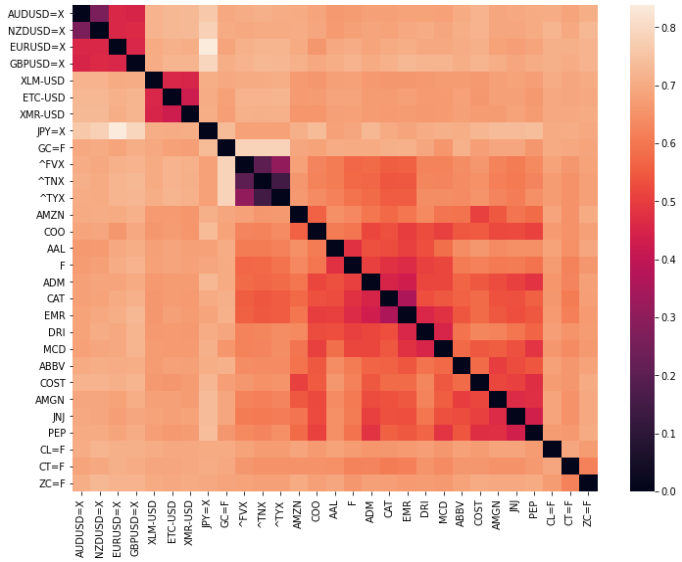
```
link = linkage(Distance_corr,'single') # for HRP

def getQuasiDiag(link): # for HRP
    link = link.astype(int)
    sortIx = pd.Series([link[-1, 0], link[-1, 1]])
    numItems = link[-1, 3]
    while sortIx.max() >= numItems:
        sortIx.index = range(0, sortIx.shape[0] * 2, 2)
        df0 = sortIx[sortIx >= numItems]
        i = df0.index
        j = df0.values - numItems
        sortIx[i] = link[j, 0]
        df0 = pd.Series(link[j, 1], index=i + 1)
        sortIx = sortIx.append(df0)
        sortIx = sortIx.sort_index()
        sortIx.index = range(sortIx.shape[0])
    return sortIx.tolist()

QuasiDiag_sort = getQuasiDiag(link)

# So we did this to see it visually
# Sorted Heatmap With QuasiDiag
plt.figure(figsize=(12,9))
sns.heatmap(
    Distance_corr.iloc[QuasiDiag_sort,QuasiDiag_sort]
    )
plt.show()
```

Output >



Weighting

3 - Naive recursive bisection (for HRP)

The recursive bisection algorithm uses information from the two previous steps to allocate weights to the assets in the portfolio. The naive recursive bisection method assumes a binary tree where clusters are recursively split into two equally-sized sub-clusters. And this causes the assets that are actually in the odd numbers, in different types, to be clustered and weighed in even numbers, and therefore one of the types of assets is divided into the rest of the clusters.

- ✓ **Using Standard Deviation as a Risk Measure Bias**

The second bias that can be mentioned in HRP Model is due to use of standard deviation as a risk measure. As you can see in the code snippet below, the variance of each cluster has been calculated using `getClusterVar ()` function. And then weights of the each cluster will be determined inverse proportional to the variance of them.

```
def getIVP(cov): # for hrp
    # Compute the inverse-variance portfolio
    ivp = 1. / np.diag(cov)
    ivp /= ivp.sum()
    return ivp

def getClusterVar(cov,cItems): # for hrp
    # Compute variance per cluster
    cov_=cov.iloc[cItems,cItems]
    w_=getIVP(cov_).reshape(-1,1)
    cVar=np.dot(np.dot(w_.T,cov_),w_)[0,0]
    return cVar
```

This bias can be clearly explained by a simple example. Assume that we have three types of assets with different return patterns. These assets are as the following:
Asset 1: The daily return of the first asset is +10% a day for half of the working days, and after 50 days of positive returns, it has -10% daily returns for the remaining days, another 50 days precisely.

Asset 2: The return pattern of the second asset is exactly the opposite of the Asset 1; It has daily return of -10% a day for the first 50 days, and +10% for the remaining half of the working days.
Asset 3: The third asset has a completely different behavior; it returns +10% in odd (2n+1) days and -10% in even(2n) days.
The return pattern of all three assets is shown in the figure below in Figure 7,

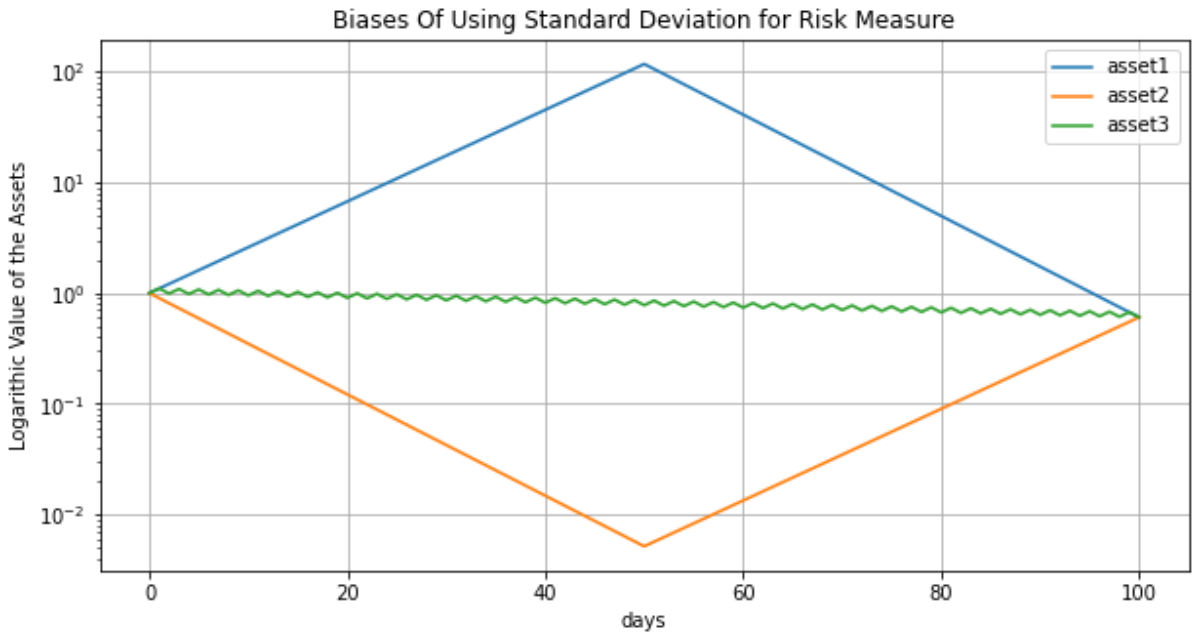


Figure 7- Logarithmic Value of the Assets in this Example

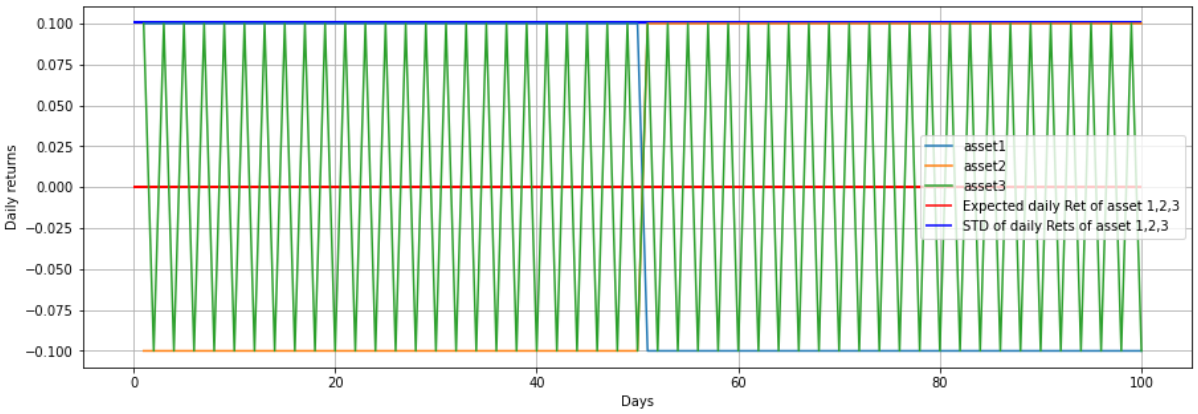


Figure 8- Daily Returns of these assets with their Expected daily return and standard deviation of these daily returns

Despite that these three assets have different behavior and return patterns, Standard deviation of daily returns and expected daily return of all three assets are the same, 0.1 and 0 respectively. As a result, according to the standard deviation as risk measure, there is no difference between these three assets,

and they are all classified as identical assets and will be weighted the same. Thus, Standard deviation may not be the best way to use as the risk measure for our portfolio optimization.
In addition, the Standard deviation is calculated from the historical data; but it violates the IID (Independent and Identically Distributed) assumption, which means that the probability of an outcome does not depend on any of the previous outcomes.
Finally, in HRP, each cluster is weighting in `getRecBipart ()` function:

```
def getRecBipart(cov, sortIx): # for hrp
    # Compute HRP allocation
    w = pd.Series(1, index=sortIx)
    cItems = [sortIx]
    while len(cItems) > 0:
        cItems = [i[j:k] for i in cItems for j, k in ((0, len(i)// 2),
        (len(i) // 2, len(i))) if len(i) > 1]
        for i in range(0, len(cItems), 2):
            cItems0 = cItems[i]
            cItems1 = cItems[i + 1]
            cVar0 = getClusterVar(cov, cItems0)
            cVar1 = getClusterVar(cov, cItems1)
            alpha = 1 - cVar0 / (cVar0 + cVar1)
            w[cItems0] *= alpha
            w[cItems1] *= 1 - alpha

    return w
```

Weighting Bias

✓ Identifying the Optimal Number of Clusters

The hierarchical clustering, that is used in HRP, steps begin by dividing the big cluster into two new clusters, which it repeats at every step; it will culminate in each asset being in its own individual cluster. This approach triggers some problems:

- The hierarchical tree is too big that algorithm computation is heavily time-consuming for very large datasets.
- The overfitting problem will occur if we let the hierarchical tree grow completely.

This will lead to very small inaccuracies in the data, which result in large estimation errors in the portfolio allocations.
Hence, enforcing a stopping measure at an early stage of the tree and a way to calculate the exact number of optimal clusters is needed.

✓ **Not Following the Dendrogram Structure**

The hierarchical clustering algorithm separates the assets in the portfolio into different clusters. Then, a tree remains that can be visualized as a graph called dendrogram, as shown below in Figure 9:

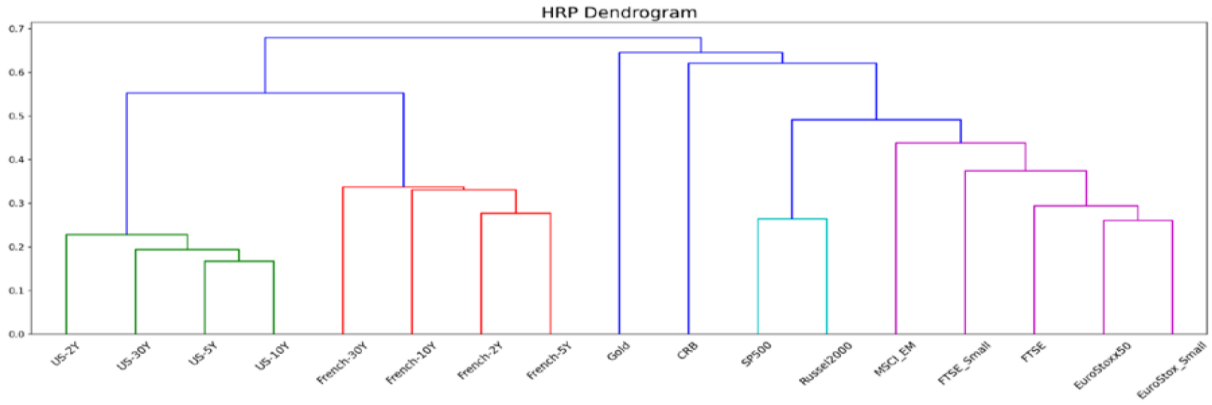


Figure 9

At first, each asset is considered a single cluster, then the clustering algorithm uses linkage matrix to determine how the clusters are combined. It is introduced different linking methods in the literature such as single, complete, average and Ward. But HRP mainly uses single linkage method in clustering, which forms the tree based on the distance between the two nearest points in these clusters. This leads to a chain effect, which makes the tree very deep and wide and prevents the formation of any dense clusters. Dr. Papenbrook has explained this issue in detail in his dissertation through the following image:

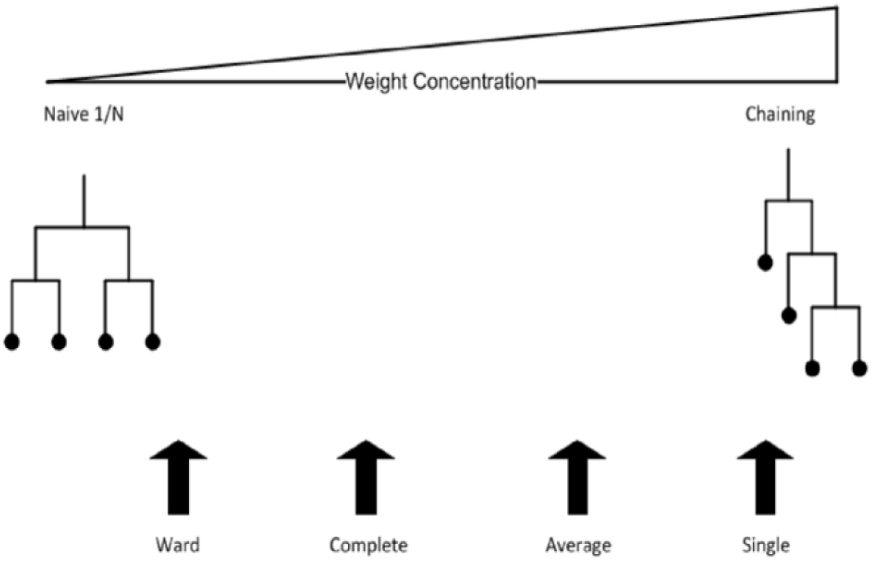


Figure 10

The chain effect on a single linkage method leads to a very large weights asset allocation to a small number of assets and the unequal distribution of the portfolio.
During recursive bisection, in HRP algorithm, the hierarchical tree grows until all the assets at the bottom level become allocated to individual weights. The issue is that HRP does not follow the dendrogram structure, instead it halves the tree according to number of assets.
The resulting hierarchy that is made in the weighting process in HRP in our example is shown below. Each of elements of this python dictionary are for levels of hierarchy structure.

```
{
1: [['AUDUSD=X', 'NZDUSD=X', 'EURUSD=X', 'GBPUSD=X', 'XLM-USD', 'ETC-USD', 'XMR-USD',
 'JPY=X', 'GC=F', '^FVX', '^TNX', '^TYX', 'AMZN', 'COO'],
 ['AAL', 'F', 'ADM', 'CAT', 'EMR', 'DRI', 'MCD', 'ABBV', 'COST', 'AMGN', 'JNJ', 'PEP', 'CL=F',
 'CT=F', 'ZC=F']],

 2: [['AUDUSD=X', 'NZDUSD=X', 'EURUSD=X', 'GBPUSD=X', 'XLM-USD', 'ETC-USD', 'XMR-USD'],
 ['JPY=X', 'GC=F', '^FVX', '^TNX', '^TYX', 'AMZN', 'COO'],
 ['AAL', 'F', 'ADM', 'CAT', 'EMR', 'DRI', 'MCD'],
 ['ABBV', 'COST', 'AMGN', 'JNJ', 'PEP', 'CL=F', 'CT=F', 'ZC=F']],

 3: [['AUDUSD=X', 'NZDUSD=X', 'EURUSD=X'], ['GBPUSD=X', 'XLM-USD', 'ETC-USD', 'XMR-USD'],
 ['JPY=X', 'GC=F', '^FVX'], ['^TNX', '^TYX', 'AMZN', 'COO'], ['AAL', 'F', 'ADM'], ['CAT', 'EMR', 'DRI', 'MCD'],
 ['ABBV', 'COST', 'AMGN', 'JNJ'], ['PEP', 'CL=F', 'CT=F', 'ZC=F']],

 4: [['AUDUSD=X'], ['NZDUSD=X', 'EURUSD=X'], ['GBPUSD=X', 'XLM-USD'], ['ETC-USD', 'XMR-USD'],
 ['JPY=X'], ['GC=F', '^FVX'], ['^TNX', '^TYX'], ['AMZN', 'COO'], ['AAL'], ['F', 'ADM'], ['CAT', 'EMR'],
 ['DRI', 'MCD'], ['ABBV', 'COST'], ['AMGN', 'JNJ'], ['PEP', 'CL=F'], ['CT=F', 'ZC=F']],

 5: [['NZDUSD=X'], ['EURUSD=X'], ['GBPUSD=X'], ['XLM-USD'], ['ETC-USD'], ['XMR-USD'], ['GC=F'],
 ['^FVX'], ['^TNX'], ['^TYX'], ['AMZN'], ['COO'], ['F'], ['ADM'], ['CAT'], ['EMR'], ['DRI'], ['MCD'],
 ['ABBV'], ['COST'], ['AMGN'], ['JNJ'], ['PEP'], ['CL=F'], ['CT=F'], ['ZC=F']]
}
```

And, Figure 11 is taken from HERC paper and illustrates the importance of following the tree structure. If you look closely, dividing the tree based on number of assets wrongly separates assets {1, 2} and {3, 4} while the correct cluster composition should have been {1} and {2, 3, 4}. You can also see these bias in our above resulting clusters.

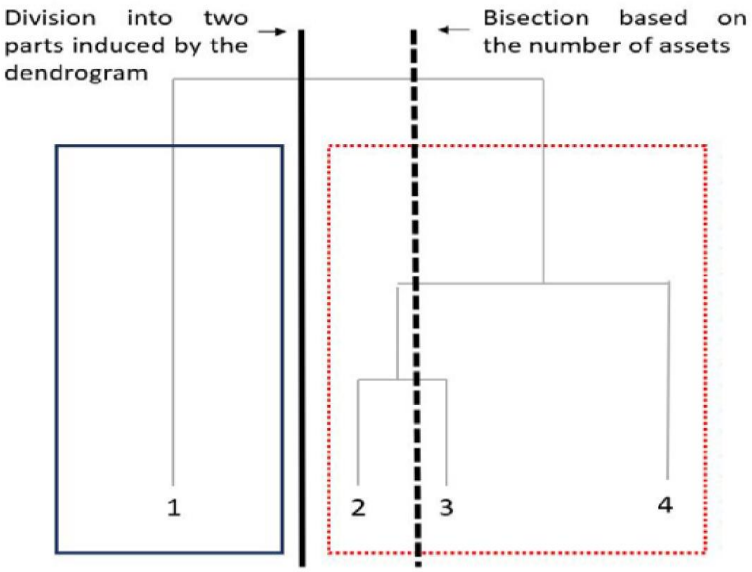


Figure 11

Conclusion

We, Quantitative Research and Development team of Deep Finance Organization, are about to introduce innovations to this field of Machine learning in finance; thus, evaluation of previous outstanding findings, Lopez de Prado's achievements for instance, is necessary to evolve through this cutting-edge technology. Therefore, two of portfolio optimization methods by Dr. de Prado have been discussed in this article, which the biases have been justified by mentioning various examples.

Each individual method, HRP and NCO, is facing three main biases. Correlation and Standard deviation Biases are common among these methods. In addition to these biases, the HRP's third bias is that in the weighing and allocation phase it does not follow the Hierarchical clustering tree. Besides, the NCO's third bias is the inability to weight the assets properly, especially when the portfolio itself contains different asset types.