A Work Project, presented as part of the requirements for the Award of a Master's degree in Finance from the Nova School of Business and Economics.

MACHINE LEARNING AND ASSET MANAGEMENT: CLUSTERING FOR PORTFOLIO CONSTRUCTION

GIACOMO VIALETTO

Work project carried out under the supervision of:

Daniele d'Arienzo

11/01/2023

Abstract (100 words maximum)
This research investigates how machine learning can be applied to portfolio management and the results of a related experimental asset allocation. Clustering aims at minimizing inter-clustering similarities, therefore translating in potentially higher diversification benefits, one of the goals of portfolio managers. The chosen allocation strategy of this research is K-Means clustering on prices with the 100 stocks with largest capitalization in the S&P500 index, fully backtested and measured as of performance. The strategy yields interesting out-of-sample explanatory power, with good results over the considered period, although confirming the difficulty for portfolio managers to consistently deliver high abnormal returns.

Keywords (minimum of four)

Machine learning, asset management, clustering, portfolio

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

1. Machine Learning for financial applications: an overview

As the need for larger data sets emerge over decades, an efficient answer can be provided by machine learning techniques. The need for more efficient techniques of data management stems from several reasons: among them, that of responding to complex (high-dimensional) problems and that of handling an increasingly large amount of data, whose connections are increasingly hard to spot by means of classical statistical properties such as simple correlation (as demonstrated later) or methods of linear regression. Machine learning techniques can therefore provide an answer to these issues as they define methods that leverage data in order to improve performance on some datasets (Mitchell 1997). As such, machine learning techniques identify as a highly interdisciplinary field, that links aspects of computer and data science, linear algebra, and statistical learning, among others. Applications to the financial sector are recent and mainly relate to sectors like portfolio management, price prediction, trade execution and credit rating. Hence, financial machine learning presents distinguishable features that are worth investigating, and next steps in research within this field can open opportunities to relevant applications. One of the fields where advances in financial machine learning are most tangible consists in portfolio management and construction. Most of the data sets available to portfolio managers are:

- High-dimensional (i.e., anywhere from a few dozens to many thousands of dimensions can be required to define a point or value within a dataset)
- Sparse (i.e., most of the elements can be equal to zero) (Di, Tao e Liu 2017)
- Hierarchical (i.e., the data can be organized into a tree-structure)

The necessity to deal with these features makes machine learning techniques particularly suitable, especially in that they can broaden simple traditional statistics or econometrics methods that might yield naïve results with those data. Machine learning can additionally be beneficial in backtesting with its promise of higher out-of-sample predictive power (Chu e Qureshi 2022).

Other than this, the second promise of machine learning in this regard is that is that of delivering and dealing with complex models (that involve several parameters) without overfitting, as detailed later. The ability of machine learning techniques to uncover hidden variables involved in a complex (i.e., high-dimensional, sparse or hierarchical, as defined above) scenario can produce theoretical frameworks with a wide range of testable implications and, within this process, machine learning plays the key role of decoupling the search for variables from the search for specification (i.e., determining the variables to be included and the functional form of the model) (Prado 2020). Other than this, the financial domain oftentimes exhibits hierarchical structures (Simon 1962): in other words, complex interactions that behave in a treelike scenario are common, and some machine learning methods, as detailed later in this research, can effectively overcome and provide solutions that traditional econometrics models might not grasp with such degree of complexity in terms of variables' interactions. As such, opportunities in the asset management industry from machine learning are several and can correct for some blanks left from traditional statistical techniques; even the research field, however, can benefit from such methods by handling large datasets with solid theoretical frameworks. Nevertheless, machine learning can present pitfalls that researchers will need to take into consideration. Approximate implementations of machine learning might easily lead to overfitting, as analyses that correspond too closely to a particular dataset or do not adjust to additional datasets in a reliable manner. Overfitting can be broadly defined in two separate ways, each of which can however be overcome by attentive machine learning applications (Prado 2020):

- Train set overfitting, i.e., where the specification explains not only the signal but even the noise in the data, which will lead to poor out-of-sample evidence
- Test set overfitting, i.e., where the model is fitted so as to perform well on the test set instead if the train set

Careful implementations can prevent such phenomena through, for instance, resampling or Monte Carlo simulation. A challenge for researchers is that financial datasets often exhibit low signal-to-noise ratio, meaning that the ratio between the desired information and the undesired statistical noise or meaningless data is low. This property is inherent to financial time series, which are non-stationary: this means they exhibit mean, variances and covariances that vary over time, something that (alongside with arbitrage forces) reduces their signal-to-noise ratio. An attentive handling of data is hence necessary to avoid working with spurious correlations or including high statistical noise. The challenges for research in financial machine learning will be, firstly, ascertaining the suitability of the datasets and, secondly, avoiding an approach devoid of theoretical foundations. Incorporating economic theory in machine learning significantly improves the model's forecasting ability, with improved models explaining almost twice as much as models devoid of theories (Baba Yara 2021). Machine learning therefore cannot be a substitute for theory, which instead remains indispensable in asset markets (Giglio, Kelly e Xiu 2021). In this research, one part will be descriptive and the other experimental: specifically, after an overview on recent applications and research (Chapter 1), concept of portfolio management (i.e., diversification, Chapter 2) and of machine learning (i.e., clustering methods, models and related features, Chapter 3) will be presented. After this, Chapter 4 will develop the experimental part, with the implementation of an asset allocation strategy of K-Means clustering of stocks on prices using Python (Chapter 4.1) and subsequent backtesting and discussion (Chapter 4.2). The research will be closed by a conclusion summarizing the results (Chapter 5).

2. Concept of diversification and pitfalls of modern portfolio theory

2.1 Benefits of diversification

Diversification, as a process to reduce exposure to any specific asset or associated risk, commonly implies investing in a variety of assets to reduce the idiosyncratic (specific) risk of an asset and the overall volatility. Diversification suggests not to consider assets as isolated

items, but to consider how they tend to reciprocally behave together (Ang 2014). Advantages of diversification affect realized returns, as the compound returns of a diversified portfolio is greater than the weighted average of the compound returns on the assets of the portfolio, with the incremental returns being due precisely to diversification (Booth e Fama 1992). The theory of diversification is part of the capital asset pricing model, based on concepts of mean-variance investing (where mean and variance define the utility of investment choices). This derives that an asset's risk premium is related to its lack of diversification benefits, which results to be the asset's beta (Ang 2014). Diversification benefits can be measured though covariances or correlations, where the variance of the returns of a two-assets portfolio is $var(r_p) = w_1^2 var(r_1) + w_2^2 var(r_2) + 2w_1 w_2 cov(r_1, r_2) = w_1^2 var(r_1) + w_2^2 var(r_2) + 2w_1 w_2 \rho_{1,2} \sigma_1 \sigma_2$. Hence, large diversification benefits correspond to low correlation. Although this framework only reflects a one-period scenario, international investments are proven to still offer significant diversification benefits (Ang e Bekaert, International Asset Allocation With Regime Shifts 2002). The research presented here will also investigate how correlations move over time. Overall, benefits from diversification are clear and it appears evident how this fits into the purpose of the research and how diversification has been the reason to choose clustering.

2.2 Markowitz's efficient frontier: main features and limitations

The efficient frontier is the set of all portfolios with the highest expected return (for a level of standard deviation) and was first formulated by Harry Markowitz in 1952 (Markowitz 1952). It emphasizes returns and risk, with mean and variance describing them. Variations of the Markowitz' efficient frontier often perform optimally in-sample, but relatively poorly out-of-sample due to computationally instabilities (i.e., exponential and often variable features unrelated to the original equation) involved in portfolio optimization problems (Prado 2020). Machine learning can overcome this by producing portfolios that perform well out-of-sample by recognizing sparse hierarchical relationships, as defined above (López de Prado 2016).

Shrinkage methods are a typical way to avoid such overfitting: these methods reduce the dimensionality of some elements by shrinking some parameters to zero (Tibshirani 1996). Clustering enters this framework as, similarly, it groups elements into clusters with simply few similar significant features being considered, as defined later. Another limitation of the Markowitz frontier is considering variance as the only measure of risk, which might lead to over-investment in risky securities, particularly when volatility is high (Cvitanić, Polimenis e Zapatero 2008); moreover, other measure of risk such as skewness might be relevant. In this research, skewness will be considered in the experimental part as well to yield a complete picture. The main mathematical assumption of modern portfolio theory is normal distribution of returns, which might not hold true, considering that correlations vary over time. This research will examine the rolling correlations of returns to demonstrate this, analyze whether and how this impacts the results and verify whether results align with global equity markets.

3. Unsupervised learning methods: clustering algorithms

A broad distinction between machine learning techniques can be drawn between supervised and unsupervised learning algorithms. With the former, data are tagged ('labeled' or categorized) as a user input, while the latter instead use unlabeled examples as available data; in this scenario, the algorithm can exhibit self-organization that captures ('learns') patterns of the data (Hinton e Sejnowski 1999). As such, supervised learning has training sample's data containing information (the labels) that are missing in the test sample; in unsupervised learning, instead, there is no distinction between training and test data, as the goal is that of inputting data in order to yield a compressed or summarized version of those data (Shalev-Shwartz e Ben-David 2014). Clustering, the technique chosen to develop this research, is an unsupervised method, as it describes the data by grouping them into subsets of similar objects, as further deepened later. Some of the main reasons that led to this choice have been outlined above: among them, its ability to yield coherent results with complex and multi-dimensional datasets, and its feature to

spot patterns and trends among the data themselves within large datasets, which results to be particularly suitable with historical financial data sets, as in this case. Clustering is also useful for its straightforward implementation; additionally, unlike other machine learning mechanism, it requires a specified number of clusters as user inputs, which will be explored in this research. A clustering problem starts with a set of objects and some associated features; the goal is that of separating those objects into groups ('clusters') according to those features, so that intragroup similarities are maximized, and intergroup similarities are minimized (Prado 2020). Different classes of clustering algorithms have different implementations: the choice of the algorithm depends on several parameters such as the type of data set available, the intended results and other parameters such as the distance function. The main parameter to be chosen is the proximity matrix (as the algorithm often relies on a distance function mathematically defined): this research will therefore clarify this aspect, which will also introduce the proper functioning of algorithms.

3.1 Proximity matrix and concept of distance

Quantifying the extent to which data differ in terms of some features is the starting problem of clustering. Measuring similarity among elements can take the form of correlation (i.e., the linear relation between two variables or data) or mutual information (i.e., the amount of information in probabilistic terms that can be obtained on one variable by observing the other, which includes higher order correlations). Measuring dissimilarity, instead, is typically done through a form of distance metric, which is a function such that (Burago, Burago e Ivanov 2001):

- The distance of an element of the dataset to itself is 0 ($d(x, x) = 0$)
- The distance between two distinct elements is always positive ($d(x, y) > 0$)
- The distance from *x* to *y* is the same as the distance from *y* to *x* ($d(x, y) = d(y, x)$)
- The triangle inequality holds true ($d(x, y) \leq d(x, y) + d(y, z)$)

With clustering algorithms, a dataset usually takes the form of a matrix composed of a number *N* of elements, each described by a number *F* of associated features: proximity can represent either similarity (correlation or mutual information, as defined above) or dissimilarity (a distance metric) (Prado 2020). With clustering algorithms, the specific distance metric representing dissimilarity is the Euclidean distance, the geometric length of a segment connecting two points in a two-dimensional space. Mathematically, the distance between two points p and q is given by (Cohen 2004) $d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$ which, for a higher-dimensional space, can be intuitively generalized as $d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$ which responds to higher dimensionality. Here, despite being usually convenient, it is not strictly necessary that dissimilarity measures satisfy the characteristics of a distance metric outlined above (Kraskov, Stoegbauer e Grassberger 2008). The choice of distance measure is a critical step of clustering techniques, which can influence results. Framing the general purpose of clustering in this way also shows why it was worth clarifying some concepts on distance measures beforehand.

3.2 Partitional clustering and hierarchical clustering

Clustering techniques are described by two broad classes: partitional and hierarchical. In partitional clustering, each element of the dataset belongs to one cluster only to create a one-level unnested partitioning of the data; instead, in hierarchical clustering the output consists in a nested sequence of partitions, with a single comprehensive cluster at the top and single clusters of individual points at the bottom (Prado 2020).

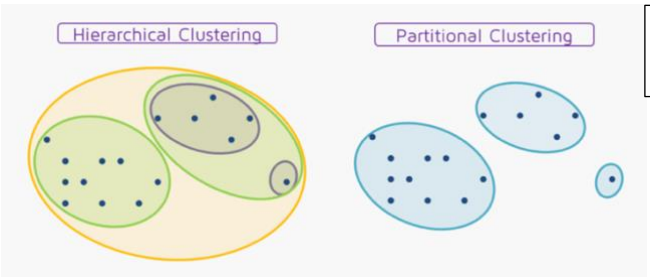

Figure 1: Hierarchical vs partitional clustering (Fuertes 2016)

The approach of hierarchical clustering described above is specifically top-down (additionally defined as 'divisive', as one all-inclusive cluster is recursively split into individual components) and can be separated from the opposite, bottom-up approach, which instead results as an 'agglomerative' process as it starts from singleton clusters and then successively merge all pairs of clusters until all clusters have been merged into a single cluster (Manning, Raghavan e Schütze 2008). Another fundamental difference is that partitional clustering requires a specific number of clusters as user input; instead, hierarchical clustering avoids this step as the only inputs required in the first place consist in the actual algorithm, the distance function, the linkage criterion (divisive or agglomerative, as defined above) and the distance threshold that leads to successive linkages. More about the number of clusters as a user input will be detailed later, as this step can clearly be crucial for the construction of an effective clustering mechanism. All in all, differences between the two classes are structural but rather broad; as such, a presentation on the main clustering algorithms can be useful to shed light on other features.

3.3 Main clustering models

Behind the broader distinction of classes outlined above, clustering algorithms can take several forms. This reason is that the notion of cluster itself is not precisely defined (Estivill-Castro 2002); nevertheless, the common ground is that clusters conceptually coincide with groups of data.

3.3.1 Connectivity models

Considering its functioning, connectivity models coincide with the class of hierarchical clustering per se. Connectivity models establish hierarchy of clusters with a process that is either agglomerative (i.e., bottom-up) or divisive (i.e., top-down), as anticipated. In connectivity models, the clustering is therefore based on distance connectivity: a criterion for linkage is established, and the results of the model are typically presented as dendrograms, which consists

in a tree-like diagram illustrating the series of steps taken by the method (either agglomerative or divisive) and the resulting arrangement (Everitt, Dictionary of Statistics 1998). A fundamental step of connectivity models consists in the choice of the linkage criterion, which is essential to define when clusters should be combined or split, after first defining a specific distance metric. A common linkage criterion used in connectivity model is the maximum or complete-linkage clustering. In this scenario, first, the two closest clusters are combined; then, at each step, the distance between clusters is computed as the distance between the elements in each cluster that are furthest away from each other, therefore responding to $D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$. As an opposite scenario, the minimum or single-linkage clustering is instead based on the distance between the two most similar (closest) elements, therefore responding to $D(X,Y) = \min_{x \in X, y \in Y} d(x,y)$. Single-linkage clustering tends to yield long clusters with elements being 'chained' one to each other, while complete-linkage clustering tends to produce more compact clusters (Adams 2018). Other than this, complete-linkage clustering tends to be strongly affected by the presence of outliers, while this issue is not commonly present with single-linkage clustering (Milligan 1980). This demonstrates that the choice of the linkage criterion, other than that of the appropriate distance metric, can highly influence the results of the clustering procedure and, particularly with connectivity models, the specific shape of the clusters. Moreover, it is crucial to devote the necessary attention to the choice of the criterion according to the data sets features as this behaves differently in the presence of outliers or similar features.

3.3.2 Centroids models

Unlike connectivity models, centroids models form non-hierarchical structures, meaning that only one level of data partitioning is produced. A particular algorithm belonging to centroids models is the K-Means clustering algorithm, whose functioning can be summarized in a few steps (MacKay 2003):

- Selection of *k* initial ‘seeds’ that serve as initial centroids (i.e., vectors representing the cluster’s center, where *k* is a specified number selected by the user as input).
- Assignment step, where k clusters are created by assigning each observation to the closest mean (the one with the least squared Euclidean distance). This step results into Voronoi cells; i.e., such that each cell is the region of space consisting of all points that are closer to a particular site than to any other (Arya, Malamatos e Mount 2002).
- Update step, where the centroids of each of the obtained *k* clusters become the new means. In other words, the means of each cluster are recalculated so as to mathematically coincide with the centroid of the given cluster.
- The last two steps (namely, those of assignment and of update) are repeated until convergence; i.e., until the following update yields no further changes.

Centroids initialization can consist in different strategies: the most common ones, used in the *sklearn* Python implementation, are either the random initialization or the K-Means++ initialization. The former consists in selecting random elements from the data set and considering them as initial centroids; the latter, instead, assigns the first initial centroid to a random element in the data set and the subsequent initial centroids from the remaining observations with a probability proportional to the squared distance of a point from the nearest centroid (developers 2007-2022). The distance is typically Euclidean, and it is squared to progressively put more weights on elements that have longer distance (Spencer 2013). Therefore, at the basis of any initialization strategy is a random process and the results may depend on the initial clusters. This will be shown in the empirical part of the project, where results have been set to be reproducible by determining specific initial centroids. The task to be solved by K-Means clustering is a complex optimization problem, and K-Means can be classified as a heuristic as it implies a trade-off that yields lower accuracy but higher computational speed. For this reason, the K-Means

process is not guaranteed to converge to a global optimum (but only to a local optimum) at the end of the convergence (Hartigan e Wong 1979). This is a two-fold aspect of K-Means: on one side, the fact that the method is but a heuristic often only converging to a local optimum makes it less accurate; on the other side, precisely this aspect allows for its speed and straight-forward nature. The advantage of K-Means being able to handle large data sets, alongside with its straightforward nature and its extended use, led to its choice for this research.

3.3.3 Distribution-based and density-based models

While clustering models analyzed so far were based on proximity, distribution models mostly leverage statistical properties of the data. The main idea is that elements with the same statistical distribution belong to the same cluster, if several kinds of statistical distributions exist in the original data set (Xu e Tian 2015). Being backed by statistics, distribution-based models lie on solid theoretical foundations. Disadvantages of such methods mainly relate to the fact that such theoretical premises might not be fully correct, and that many parameters might be necessary, even risking overfitting of data. However, precisely this statistical basis is a relevant advantage of these models, alongside with their high scalability of results even with different distributions and the numbers of clusters (Xu e Tian 2015). Density-based clustering consists in forming clusters that correspond to areas with higher density of elements than the remaining data set. Clusters are separated from each other by contiguous regions of low density of objects and data in these low-density regions are considered as noise or outliers (Kriegel, et al. 2011). A popular density-based clustering method is DBSCAN, which connects points that satisfy a certain density criterion. In other words, the density in the neighborhood (indicated as radius ε and defined by a distance function) of an element in a given cluster has to exceed some threshold ($minPts$), usually defined as a minimum number of objects (Ester, et al. 1996). A cluster includes all objects directly connected thorough this density criterion (core points), plus all objects within their range (reachable points). DBSCAN allows for clusters of different sizes and

shapes (unlike other algorithms); secondly, it effectively handles outliers; lastly, it has a mostly deterministic rather than random approach (Amiruzzaman, et al. 2021). On the other side, while only two inputs need to be chosen ($minPts$ and ε), the choice of distance metric might make it difficult to find an appropriate threshold ε; moreover, the clustering task might be difficult if data sets have largely different densities of elements (Kriegel, et al. 2011).

3.4 Number of clusters

In most partitional clustering methods (except for density-based models), the number of clusters to be formed is a user input to be specified beforehand. The optimal number of clusters often cannot be intuitively grasped by data set properties, so several methods can be used. Among them, the Elbow method plots a function between the number of clusters and the percentage of explained variation: the method stops adding clusters when the marginal percentage of explained variation does not exceed a threshold (graphically, it drops sharply); here, the measure of explained variation is usually the ratio of the between-group variance to total variance (Prado 2020). The main drawback is that the threshold evaluating the marginal explained variation is chosen arbitrarily (Goutte, et al. 1999). Another strategy is the silhouette method: this shows which objects lie well within the remaining data in their own cluster and which ones are between clusters, in a scale between +1 and -1 (where higher values indicating good fitting) and a distance metric. The silhouette method provides another graphical solution by combining the silhouettes into a single plot with the average silhouette width (Rousseuw 1987). In this research, the Elbow method will be used for its common use and to explain its working as a heuristic method.

4. Experimental results in Python: K-Means clustering on stock prices

4.1 Construction

The following part of this research aims at showing a concrete example of portfolio construction through K-Means clustering and subsequent backtesting. This section, which incorporates the

experimental goal of the research and its core focus, will proceed in a twofold manner: firstly, with the strategy construction; secondly, with the strategy backtesting. The chosen programming language is Python due to its large availability of libraries for data sets and, particularly, for applications of machine learning. The Python implementation of some parts of the code built for the strategy's construction and backtested will be shown here to yield a complete picture on how the strategy has been formed and evaluated. The Python libraries used for this research are *Pandas* (useful for data structures and operations with time series); *NumPy* (useful for operations on matrices and arrays); *Sklearn* (machine learning library); *Matplotlib* (plotting library) and *Statsmodels* (useful for statistical tests performed in the backtesting part). The choice of performing K-Means specifically is due to several reasons: on one side, its ability to handle large data sets (as outlined below, as the starting data sets comprises daily prices of stocks over several decades, this is a fundamental feature) and its flexibility with different data sets (that allowed to test several solutions in the process of construction by preserving consistency). Other reasons include K-Means being one of the most popular solutions among clustering algorithms and its need to specify the number of clusters as user input. This latter point, despite being usually seen as a drawback of such method, is used here as an opportunity to illustrate how such input can be determined within the functioning of the broader mechanism. Stock prices have been chosen as the clustering focus: on one side, this choice is due to the fact that stock can be said to allow for intuitive clustering criteria; on the other, stock prices can potentially present high volatility, so that differences in this regard can present interesting clustering opportunities. After construction, the obtained portfolio will be backtested by comparing in-sample and out-of-sample performances. This is a necessary step that allows to validate or question the results and to verify whether the results are merely the outcome of overfitting or whether its validity can be maintained even outside of the training sample. Backtesting by splitting the data into two different (in-sample and out-of-sample) periods has been chosen as backtesting strategy due to

its faster computational requirements and the opportunity to directly verify the promise of higher out-of-sample predictive power of machine learning. The entire sample period has been chosen as starting from the first day of January 1990 until 19 October 2022 (present date as of construction) so as to have a large enough data set. The in-sample period spans from 1 January 1990 to 19 October 2015, while the out-of-sample period from 19 October 2015 to 19 October 2022: this is due to the fact that the typical size of the test set is usually around 20% of the total sample (Hyndman e Athanasopoulos 2018). As such, after importing the Python libraries, daily stock prices of the 100 stocks with larger market capitalization that are currently part of the S&P500 index have been uploaded. Working on the 100 stocks with largest market cap has been deemed as necessary because, if considering all 500 stocks, some elements would have had missing values in terms of returns and variance over the entire in-sample period (as some of the smaller-cap stocks have been listed later than 19 October 2015). In general, clustering methods (and K-Means specifically) cannot analyze items that have missing data values (Wagstaff 2004), unless the missing values are treated accordingly (often by simply ignoring them, which would reduce the data set similarly to what has been done by considering the largest stocks only). From daily prices of such stocks, variances and returns are annualized and computed as in *Figure 2*. Stocks are then plotted on a x-y space according to their prices' return and variance using Euclidean distances among them. As anticipated, before doing that it is necessary to determine the optimal number of clusters. This can be done through the Elbow method (*Figure 3*), which is chosen here for its common use and for its emphasis on explained variance. The Elbow curve solution can be graphically rendered as a x-y space between the number of clusters and the inertia (as sum of explained variance) (*Figure 4*):

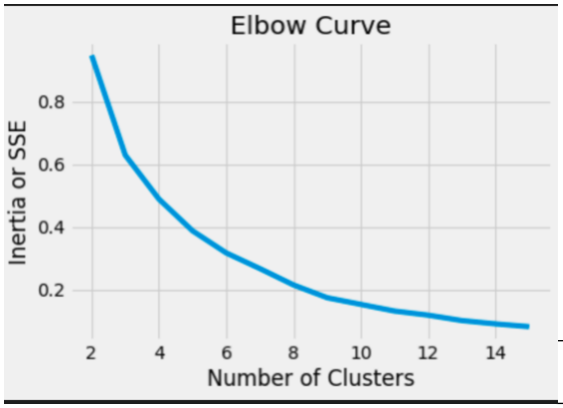

Figure 4: Elbow curve graph

However, here the graphical solution of the Elbow method does not clearly convey the optimal number of clusters as there is no defined turning point in the curve: this might happen as the Elbow curve method is but a heuristic strategy; namely, it produces an approximate solution that allows for its higher computational speed. As such, it is necessary to explicitly obtain the optimal number of cluster (which here results to be 6) by exploiting additional Python libraries (*Figure 5*). Once the number of clusters has been determined, the proper K-Means clustering process can take place by using the input just defined to assign a specific cluster (labeled from 0 to 5) to each stock in the data set (*Figure 6*). In the algorithm, the parameter *random_state* has to role of determining a random number generation necessary for centroid initialization (developers 2007-2022); in this case, an integer has been used to make the randomness deterministic in order to have reproducible results. If this was not done, the clusters assigned to each specific stock would vary each time the code is run. Making the code reproducible has been chosen for purposes of potential comparison and to allow for performance evaluation of clusters. Once the algorithm has formed the clusters, it is possible to graphically visualize them in a x-y space that relates variance and returns and forms clusters according to them (*Figure 7*):

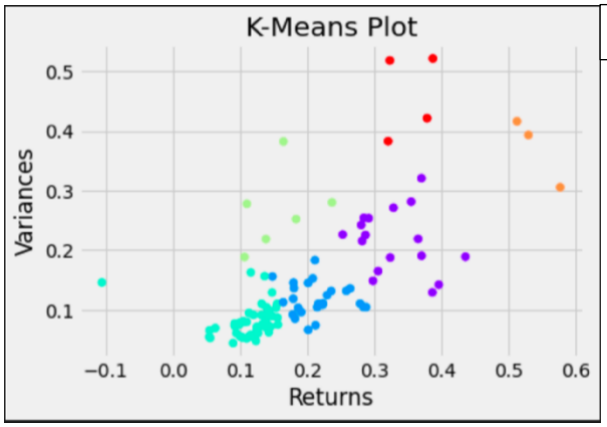

Figure 7: K-Means plot

Each of the six clusters includes several stocks, so to perform further analysis on the cluster performance, it is necessary to consider each clusters' return and variance as the average return and average variance of the included stocks (*Figure 8*); this is done on purpose in order to eliminate idiosyncratic noise. The return and variance of each cluster correspond to (*Figure 9*):

	Variance	Return
Cluster 0	21.55%	33.02%
Cluster 1	11.79%	21.36%
Cluster 2	8.10%	11.30%
Cluster 3	26.67%	15.66%
Cluster 4	37.18%	53.98%
Cluster 5	46.11%	35.24%

Figure 9: Clusters' variance and returns

4.2 Discussion

To correctly assess the validity of a strategy, backtesting is a necessary step. Backtesting allows to effectively determine whether in-sample returns are merely the results of overfitting or other incorrect features or whether, contrarily, they adapt well even to a sample outside of that where relevant signals have been formed. To do so, returns of each of the six clusters have been uploaded on a monthly (rather than daily) basis because monthly data better fit the features of some of the measures presented below, such as that of rolling correlation. It is therefore possible to plot the returns of the overall clustering strategy as the mean of the returns of each of the six clusters (*Figure 10*). Hence, the strategy returns are computed as the equally-weighted returns of the six clusters:

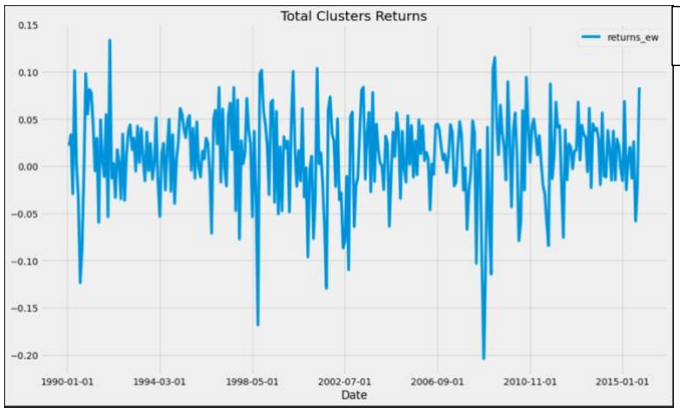

Figure 10: Total clusters' returns

Most of the monthly returns of the strategy expectedly lie in the range between -10% and +10%, with the exceptions of the period around the end of 1998 and, most notably, the financial crisis of 2008. It can be worthwhile annualizing returns and comparing the rolling version of annualized returns with the monthly returns. As such, after defining the function to annualize returns, their rolling version can be compared with the graph above as (*Figure 11*):

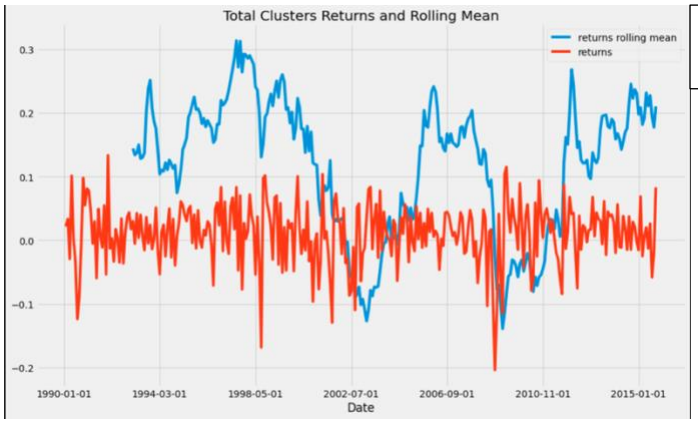

Figure 11: Total clusters' returns and rolling mean

Returns can also be cumulated with the *cumprod* method, so as to plot the cumulated wealth that can be obtained over the in-sample period from investing $100 dollar at inception (*Figure 12*), alongside with the version with the rolling mean on a 36-months window (*Figure 13*):

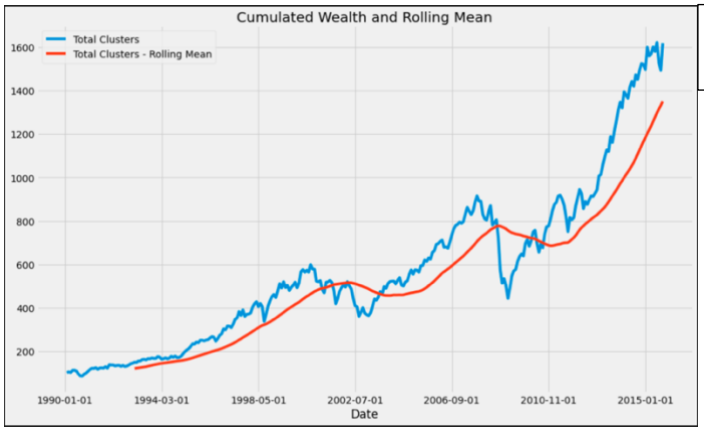

Figure 13: Cumulated wealth and rolling mean

An initial investment of $100 in such strategy in 1990 would have amounted around $1,600 cumulated in 2015, with significant plunges around 2002 (corresponding to the dot-com bubble) and 2008 (the financial crisis). Other than returns, measures of risk are also worth being considered. As anticipated earlier, higher diversification tends to correspond to lower correlation; this measure, however, varies over time: for this reason, the average rolling correlation among the six clusters (on a 36-months basis) has been computed here for the entire in-sample period through the *rolling* and *corr* Python methods. After grouping by date, it is possible to obtain the 6x6 matrix of the rolling correlation among clusters on a daily basis (*Figure 14*):

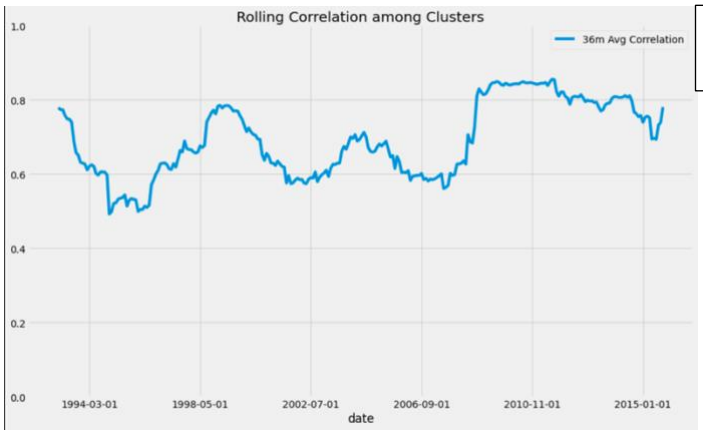

Figure 14: Rolling correlation among clusters

The graph shows higher correlation peaks in periods with lower returns, indicating that correlations move over time and tend to be higher during bear markets: this translates the fact that returns are also affected by the lower diversification benefits over recession periods. In other words, minimizing inter-clustering similarities (one of the purposes of the clustering strategy) is much more difficult during bear markets. Other than this, the general trend is that of increasing correlation over time, with higher values mostly after 2008 and over the entire 2010s decade: this seems to confirm a more global tendency in the equity space; namely, that correlations of international stock returns have significantly trended upwards over time (Christoffersen, et al. 2014). It is possible therefore to suppose that the upward trend in clusters' average rolling correlation simply reflects a global trend, rather than suggesting persistently lower diversification benefits (even though upward variations can certainly be indicative of lower diversification during general bear markets, as mentioned above). Other than the most traditional measures of risk, drawdown and skewness can also be computed: the former is the percentage decline from a peak to a through; the latter, instead, measures the asymmetry in the returns' statistical distribution. As standard deviation (the usual metric of risk) assumes normal distribution of returns, analyzing skewness can be a meaningful way to assess risk whenever such assumption does not hold true. After computing drawdown, maximum drawdown to date (defined as the highest drawdown from the start of the sample until each date) can be plotted as (*Figure 15*):

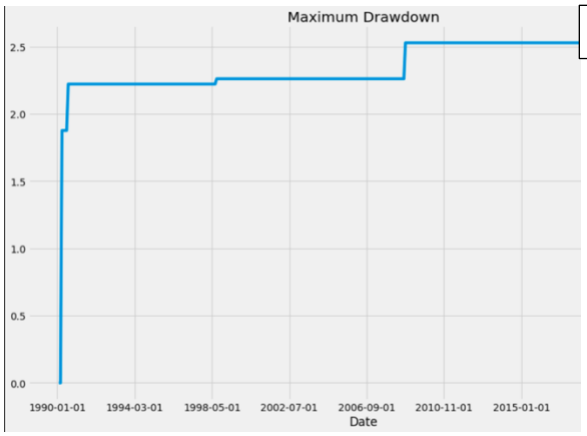

Figure 15: Maximum Drawdown

The maximum drawdown values reach a peak of 2.5% approximately over the 2008 financial crisis, with precedent values usually slightly above 2% and with only gradual increases. Skewness is computed both for each of the six clusters and for the overall strategy (*Figure 16*):

	Skewness
Cluster 0	-0.546507
Cluster 1	-0.58812
Cluster 2	-0.612356
Cluster 3	-1.020965
Cluster 4	-0.37375
Cluster 5	-0.585336
Overall strategy	-0.755514604

Figure 16: Clusters' and overall strategy's skewness

It can be noted that skewness is negative in all cases: this indicates that the returns distribution is left-skewed (i.e., the median is greater than the mean); in other words, investors might expect numerous small gains and few large losses from the strategy (which shows an overall skewness of -0.75). This is a risk indication that might be taken into account when considering the risk aversion and preferences of investors and seems to be rather strong across all clusters. As measures of skewness demonstrated that returns tend not to be normally distributed, it is even more important to include such consideration in the assessment of the strategy. Other than all these preliminary measures, a full backtesting strategy involves observing the performance of the in-sample returns of the strategy so as to compare those to the out-of-sample period (which here is the window from 19 October 2015 to 19 October 2022). The clustering strategy's returns are compared to the returns of the S&P500 index, which is a common benchmark for

performance analysis (*Figure 17* and *Figure 18*), for both the in-sample and out-of-sample periods:

Figure 17: In-sample performance

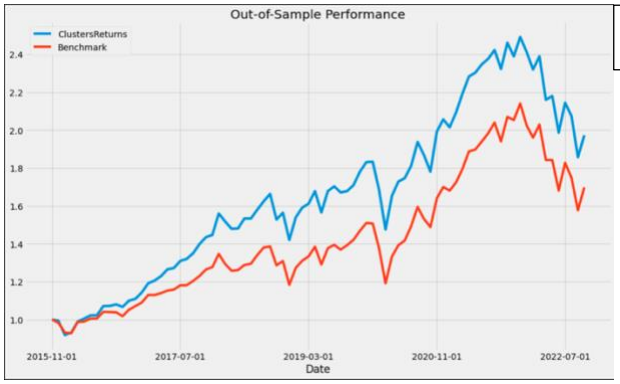

Figure 18: Out-of-sample performance

The training (in-sample) period defines the signals of the strategy: in other words, clusters are formed with data belonging to this window only; on the other side, the test period is useful to verify whether performance results are comparable across samples. As it appears from the graphs, out-of-sample performance mimics the in-sample performance in that the clustering strategy's returns tend to outperform that of the benchmark. Nevertheless, to fully assess the strategy's validity, factor analysis might be necessary (namely, assessing which investment factors carry more weight). In doing so, instead of benchmarking the strategy against the market only, a more appropriate risk-adjusted benchmark can be $r = \beta r_{mkt} + (1 - \beta) r_f$, which takes into account risk-free assets (such as T-Bills) as well. After running the regression on the clustering strategy's returns and constructing a replicating portfolio (i.e., a portfolio reproducing the strategy's returns through risk and risk-free assets allocation) for both sample periods, regression results (*Figure 19* and *Figure 20*) yield the following measures (*Figure 21*):

	In-Sample (R2=0.889)	Out-of-Sample (R2=0.951)
Market loading	1.0475	0.9606
Alpha	0.11%	-0.02%
Tracking error	1.60%	1.04%
Tracking error (annual)	5.53%	3.59%
Information Ratio	0.0677	-0.0234
Information Ratio (annual)	0.2359	-0.0809

Figure 21: Performance analysis

The market loading of the strategy, both in-sample and out-of-sample, is particularly high: this means that the replicating portfolio would have to put a lot of weight on the risk-bearing assets to mimic the strategy's returns and that the market is the main risk factor. Relatively to this, the strategy is adding a 0.11% of excess returns (measured by alpha): this measure, however, drops in the out-of-sample period to around 0%. Nevertheless, the out-of-sample period ranges over 7 years that most notably include that 2020 pandemic crisis and the following market shock due to inflationary pressures: as such, these elements might have largely hindered any strategy's chances to realize significant excess returns. This is also reflected by the information ratio, which is defined as the ratio of the excess returns of strategy with respect to the benchmark over its volatility. This measure, slightly positive in the training period, drops slightly below 0 in the test period, therefore confirming the observation made for the alpha values. The difficulty of attaining high alpha values, however, is not surprising, and might also be simply interpreted as the proof of the challenging task of active portfolio management. Most importantly, however, the statistical significance (R^2) of the regression displays high values for both sample periods. As it appears, R^2 for the in-sample is equal to 0.889, indicating that almost 90% of the variation in the observed data is explained by the regression model: this value attains a similarly high value for the out-of-sample period. This confirms to some extents the good out-of-sample predictive power of machine learning, one of the main questions of this research. Another relevant measure of performance analysis is the tracking error, instead, which coincides with the standard deviation of the replicating portfolio and measures how closely the strategy follows the benchmark: in this case, it is relatively low in both samples, confirming that the strategy does not diverge a lot from the risk-adjusted benchmark. This is expected as the strategy consists in

an asset allocation mechanism that does not operate any particularly active stock selection, therefore ending up to closely resemble its benchmark. For both tracking error and information ratio, the respective annualized version of such measures is also displayed above for purposes of comparison: these confirm the trend observed for their monthly version in each of the respective cases. All in all, measures of performance analysis displayed above can be much more explanatory than simple observations of returns or cumulated wealth, as they analyze to a higher degree of detail the performance of the strategy. This is also useful to fully assess the effectiveness of the strategy as a machine learning method, which might significantly differ from more traditional asset allocation strategies. Therefore, in this section of the research, a full backtesting and performance analysis have been implemented, and this step has proved fundamental to draw significant and rather conclusive interpretation on the strategy effectiveness. This latter section, therefore, embodies the core of this research work as it empirically demonstrates that effective implementations of machine learning techniques can be feasible with respect to the asset management spectrum as well, as long as the suitability of the data is ascertained beforehand and, in the case of specific portfolio strategies, their potential effectiveness is fully backtested and assessed, especially against one or more benchmarks for full comparison.

5. Conclusion

As the need to handle large financial data sets strengthens, machine learning can provide and efficient answer. However, methods in this regard vary widely, and choosing the appropriate algorithm is a pivotal aspect that depends on several variables. Among them, understanding the very characteristics of the starting data set is the first aspect, alongside with determining the distance metric and other inputs, when required. In the case analyzed above, the K-Means method displays moderately good results; nevertheless, the starting sample might be even larger, although this might require some data processing that might affect the original data. This is a

distinguishable point of machine learning, that often requires data cleaning to properly work: for instance, as saw before, K-Means does not work with missing values, so that a careful data handling might be needed in this case. The performance statistics outlined above highlight the difficulty to yield good excess returns, even more so when those are measured against a risk-adjusted benchmark as in this case. Nevertheless, the statistical significance of the regression analysis is high for both samples, suggesting a good out-of-sample predictive power, one of the main promises of machine learning. All in all, these methods can provide good strategies for asset allocation; nevertheless, it is worth reminding that each strategy should be followed by an appropriate backtesting to potentially questions some results or, at least, raise useful doubts. Other than this, framing any of these methods within broader theoretical frameworks is a necessary step, as anticipated. Any implementation of machine learning that does not disregard any of these aspects can appropriately yield interesting opportunities for the financial and asset management field, both in industry and academia.

6. References

Mitchell, Tom. 1997. *Machine Learning.* New York City: McGraw Hill.

Di, Yan, Wu Tao, and Ying Liu. 2017. "An efficient sparse-dense matrix multiplication on a multicore system." *2017 IEEE 17th International Conference on Communication Technology (ICCT).* 1880-1883.

Prado, Marcos M. López de. 2020. *Machine Learning for Asset Managers.* Cambridge: Cambridge University Press.

Ang, Andrew. 2014. *Asset Management: A Systematic Approach to Factor Investing.* New York City: Oxford University Press.

Booth, David G., and Eugene F. Fama. 1992. "Diversification Returns and Asset Contributions." *Financial Analysts Journal, Vol. 48, No. 3* 26-32.

Ang, Andrew, and Geert Bekaert. 2002. "International Asset Allocation With Regime Shifts."

The Review of Financial Studies, Volume 15, Issue 4 1137–1187.

Markowitz, Harry. 1952. "Portfolio Selection." *The Journal of Finance Vol. 7, No. 1* 77-91.

López de Prado, M. 2016. "Building Diversified Portfolios that Outperform Out-of-Sample." *Journal of Portfolio Management, Vol. 42, No. 4* 59-69.

Cvitanić, Jakša, Vassilis Polimenis, and Fernando Zapatero. 2008. "Optimal portfolio allocation with higher moments." *Annals of Finance. 4 (1)* 1-28.

Hinton, Geoffrey, and Terrence Sejnowski. 1999. *Unsupervised Learning: Foundations of Neural Computation.* Cambridge, Massachusetts: MIT Press.

Shalev-Shwartz, Shai, and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms.* New York City, New York: Cambridge University Press.

Cohen, David. 2004. *Precalculus: A Problems-Oriented Approach.* Cengage Learning.

Kraskov, A., H. Stoegbauer, and P. Grassberger. 2008. "Estimating Mutual Information." *Working paper.*

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

Giglio, Stefano, Bryan Kelly, and Dacheng Xiu. 2021. "Factor Models, Machine Learning, and Asset Pricing." *Annu. Rev. Financ. Econ. 2021* 1-35.

Simon, H. 1962. "The Architecture of Complexity." *Proceedings of the American Philosophical Society, Vol. 106, No. 6* 467-482.

Baba Yara, Fahiz. 2021. "Machine Learning and Return Predictability Across Firms, Time and Portfolios."

Estivill-Castro, Vladimir. 2002. "Why so many clustering algorithms – A Position Paper." *ACM SIGKDD Explorations Newsletter. 4* 65-75.

Everitt, Brian. 1998. *Dictionary of Statistics.* Cambridge: Cambridge University Press.

Milligan, G. W. 1980. "An examination of the effect of six types of error perturbation on

fifteen clustering algorithms." *Psychometrika* 45: 325–342.

Spencer, Neil H. 2013. "5.4.5 Squared Euclidean Distances." In *Essentials of Multivariate Data Analysis*, by Neil H. Spencer, 95. CRC Press.

Hamerly, Greg, and Charles Elkan. 2002. "Alternatives to the k-means algorithm that find better clusterings." *Proceedings of the eleventh international conference on Information and knowledge management (CIKM).*

MacKay, David. 2003. "Chapter 20. An Example Inference Task: Clustering." In *Information Theory, Inference and Learning Algorithms*, by David MacKay, 284–292. Cambridge University Press.

Arya, S., T. Malamatos, and D. M. Mount. 2002. "Space-Efficient Approximate Voronoi diagrams." *Proc. 34th ACM Symp. on Theory of Computing.* 721-730.

developers, scikit-learn. 2007-2022. *sklearn.cluster.KMeans.* Accessed 11 15, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

Hartigan, J. A., and M. A. Wong. 1979. "Algorithm AS 136: A k-Means Clustering Algorithm." *Journal of the Royal Statistical Society, Series C. 28 (1)* 100–108.

Whelan, Christopher, Greg Harrell, and Jin Wang. 2015. "Understanding the K-Medians Problem." *Int'l Conf. Scientific Computing | CSC'15.* 219-222.

Xu, Dongkuan, and Yingjie Tian. 2015. "A Comprehensive Survey of Clustering Algorithms." *Ann. Data. Sci. (2015) 2(2)*, 165–193.

Kriegel, Hans-Peter, Peer Kröger, Jörg Sander, and Arthur Zimek. 2011. "Density-based Clustering." *WIREs Data Mining and Knowledge Discovery 1 (3)*, 231–240.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. "A density-based algorithm for discovering clusters in large spatial databases with noise." *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96).* AAAI Press. 226–231.

Amiruzzaman, Md, Rashik Rahman, Md. Rajibul Islam, and Rizal Mohd Nor. 2021. "Evaluation of DBSCAN algorithm on differentprogramming languages: An exploratory study." *5th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT 2021).*

Goutte, C., P. Toft, E. Rostrup, F. Nielsen, and L. Hansen. 1999. "On Clustering fMRI Time Series." *NeuroImage,,* 298–310.

Rousseuw, Peter J. 1987. "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Computational and Applied Mathematics. 20* 53–65.

Wagstaff, Kiri. 2004. "Clustering with Missing Values: No Imputation Required." *Studies in Classification, Data Analysis, and Knowledge Organisation.*

Hyndman, Rob J., and George Athanasopoulos. 2018. "Evaluating forecast accuracy." In *Forecasting: Principles and Practice*, by Rob J. Hyndman and George Athanasopoulos. Melbourne, Australia: OTexts.com/fpp2.

Christoffersen, Peter, Vihang Errunza, Kris Jacobs, and Xisong Jin. 2014. "Correlation dynamics and international diversification benefits." *International Journal of Forecasting 30(3).*

Fuertes, T. 2016. *https://quantdare.com/hierarchical-clustering/.* Accessed 11 26, 2022. https://quantdare.com/hierarchical-clustering/.

Chu, Ba, and Shafiullah Qureshi. 2022. "Comparing Out-of-Sample Performance of Machine Learning Methods to Forecast U.S. GDP Growth." *Comput Econ.*

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B, Vol. 58, No. 1* 267-288.

Burago, Dmitri, Yuri Burago, and Sergei Ivanov. 2001. *A course in metric geometry.* Providence: American Mathematical Society.

Adams, Ryan P. 2018. "Hierarchical Clustering COS 324 – Elements of Machine Learning."

Princeton University. Accessed 11 13, 2022. https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/hierarchical-clustering.pdf.

7. Appendix

Figure 1: Hierarchical vs partitional clustering (Fuertes 2016)

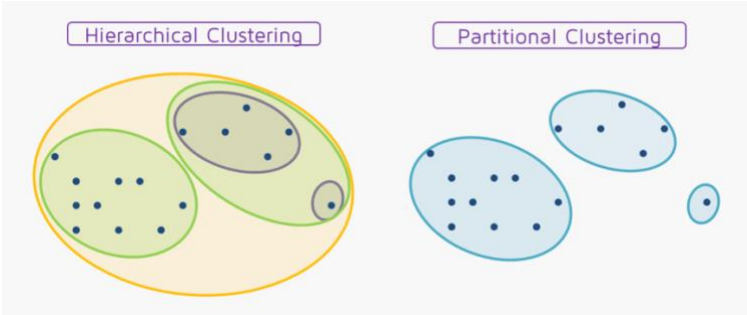

Figure 2: Stocks' annualized variances and returns (Python implementation)

```
daily_returns = df.pct_change()
annual_mean_returns = daily_returns.mean() * 252
annual_return_variance = daily_returns.var() * 252
```

```
df2 = pd.DataFrame(df.columns, columns = ['Stock_Symbols'])
df2['Variances'] = annual_return_variance.values
df2['Returns'] = annual_mean_returns.values
df2
```

	Stock_Symbols	Variances	Returns
0	AAPL	0.215589	0.281950
1	MSFT	0.108485	0.222673
2	AMZN	0.416301	0.512633
3	GOOGL	0.104584	0.287946
4	TSLA	0.305621	0.577086
...	...	...	...
95	SYK	0.110461	0.216787
96	TGT	0.103611	0.153476
97	NOW	0.189232	0.435785
98	PGR	0.092180	0.178478
99	BKNG	0.518311	0.323060

100 rows × 3 columns

Figure 3: Elbow method (Python implementation)

```
X = df2[['Returns', 'Variances']].values
inertia_list = []
for k in range(2,16):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    inertia_list.append(kmeans.inertia_)
```

Figure 4: Elbow method graph

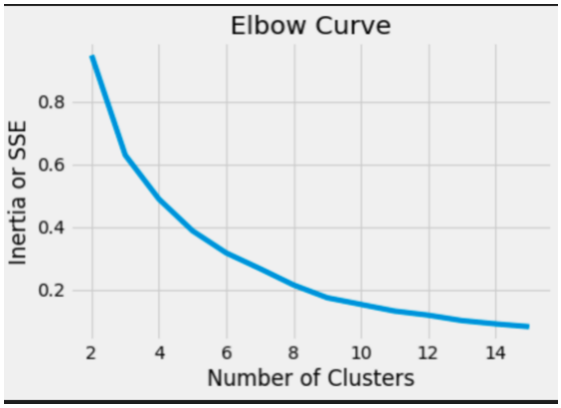

Figure 5: Determination of the number of clusters (Python implementation)

```
from kneed import DataGenerator, KneeLocator

x = range(2,16)
y = inertia_list
kn = KneeLocator(
    x,
    y,
    curve = 'convex',
    direction = 'decreasing',
    interp_method = 'polynomial'
)

print(kn.knee)
```

```
6
```

Figure 6: K-Means clustering (Python implementation)

```
kmeans = KMeans(n_clusters=6, random_state=1).fit(X)
labels = kmeans.labels_
labels
```

Figure 7: K-Means plot

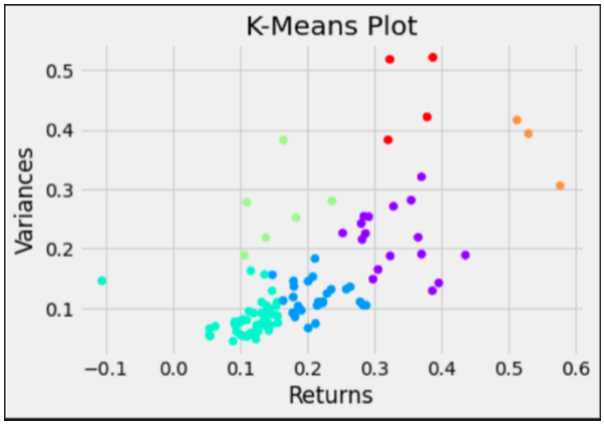

Figure 8: Computation of clusters' variance and return (Python implementation)

```
def diversed_port():
    for i in range(0, 6):
        print(df2[df2['Cluster_Labels'] == i].mean())
```

Figure 9: Clusters' variance and returns

	Variance	Return
Cluster 0	21.55%	33.02%
Cluster 1	11.79%	21.36%
Cluster 2	8.10%	11.30%
Cluster 3	26.67%	15.66%
Cluster 4	37.18%	53.98%
Cluster 5	46.11%	35.24%

Figure 10: Total clusters' returns

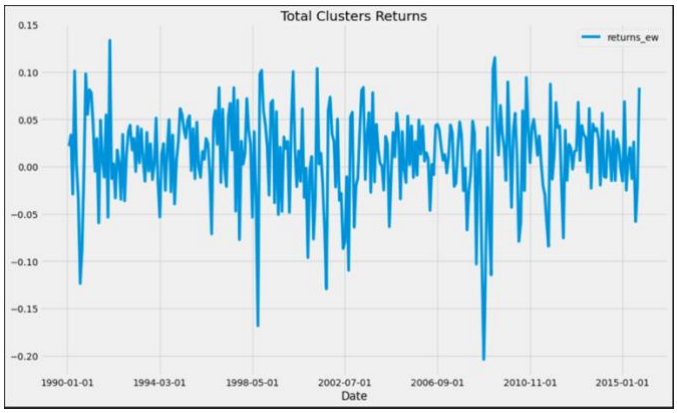

Figure 11: Total clusters' returns and rolling mean

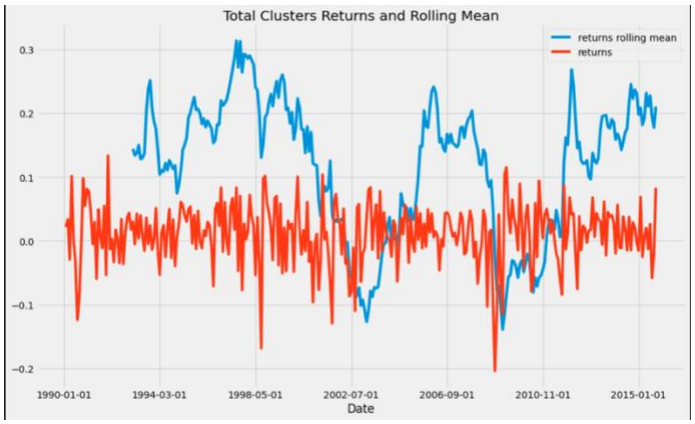

Figure 12: Cumulated wealth

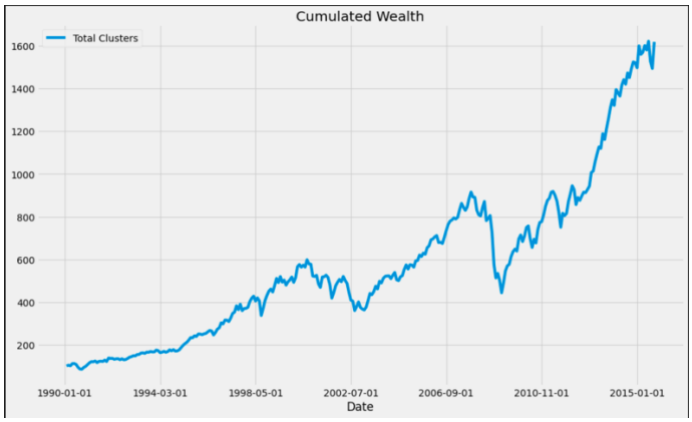

Figure 13: Cumulated wealth and rolling mean

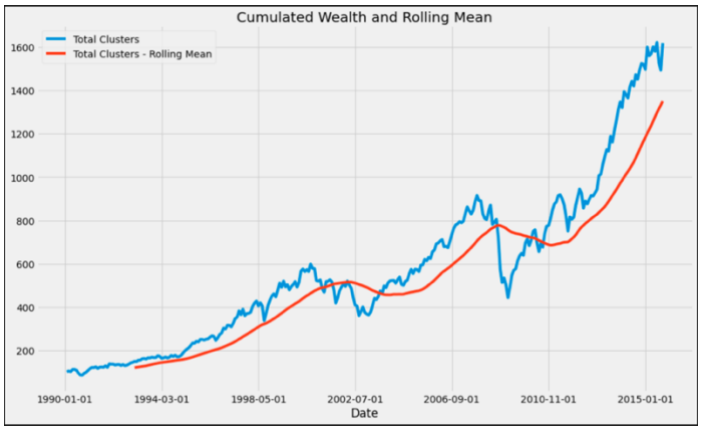

Figure 14: Rolling correlation among clusters

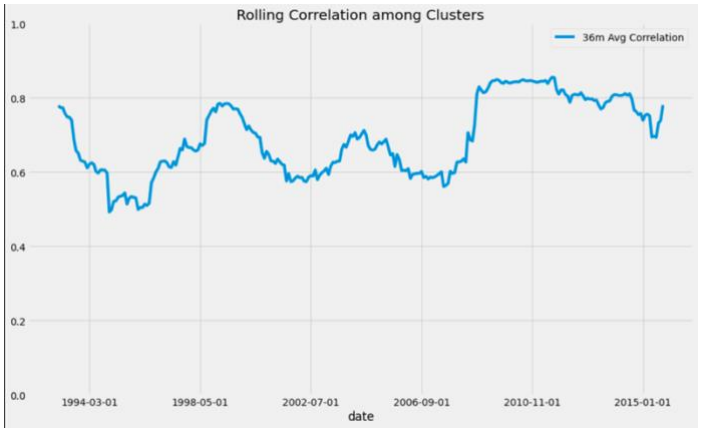

Figure 15: Maximum Drawdown

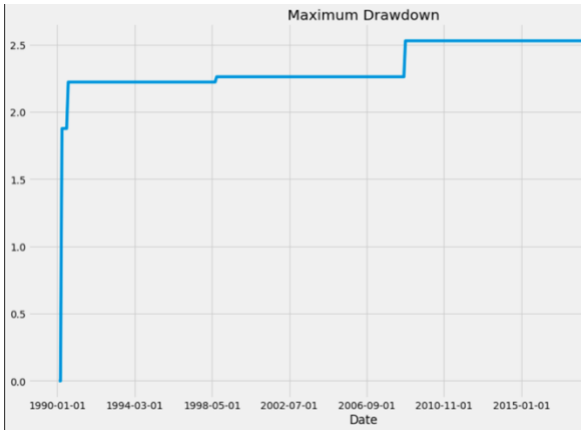

Figure 16: Clusters' and overall strategy's skewness

	Skewness
Cluster 0	-0.546507
Cluster 1	-0.58812
Cluster 2	-0.612356
Cluster 3	-1.020965
Cluster 4	-0.37375
Cluster 5	-0.585336
Overall strategy	-0.755514604

Figure 17: In-sample performance

Figure 18: Out-of-sample performance

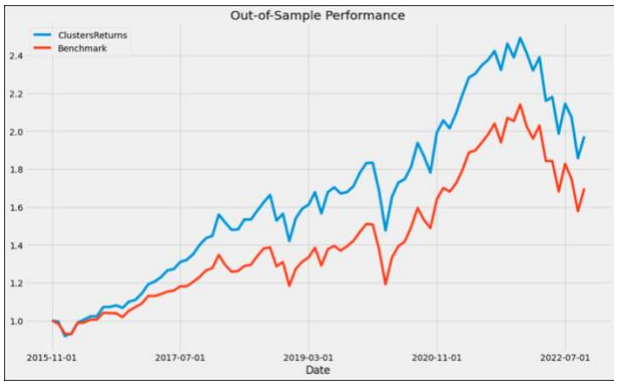

Figure 19: In-sample performance analysis computation (Python implementation)

```
import statsmodels.api as sm
clusters_excess = mktrf.loc["1990-02-01":"2015-10-01",['Clusters']] - mktrf.loc["1990-02-01":"2015-10-01", ['RF']].values
mkt_excess = mktrf.loc["1990-02-01":"2015-10-01",['Mkt-RF']]
exp_var = mkt_excess.copy()
exp_var["Constant"] = 1
lm = sm.OLS(clusters_excess, exp_var).fit()
```

Figure 20: Out-of-sample performance analysis computation (Python implementation)

```
import statsmodels.api as sm
clusters_excess = mktrf.loc["1990-02-01":"2015-10-01",['Clusters']] - mktrf.loc["1990-02-01":"2015-10-01", ['RF']].values
mkt_excess = mktrf.loc["1990-02-01":"2015-10-01",['Mkt-RF']]
exp_var = mkt_excess.copy()
exp_var["Constant"] = 1
lm = sm.OLS(clusters_excess, exp_var).fit()
```

Figure 21: Performance analysis

	In-Sample (R2=0.889)	Out-of-Sample (R2=0.951)
Market loading	1.0475	0.9606
Alpha	0.11%	-0.02%
Tracking error	1.60%	1.04%
Tracking error (annual)	5.53%	3.59%
Information Ratio	0.0677	-0.0234
Information Ratio (annual)	0.2359	-0.0809