

https://dspace.org/

Pairs Trading via Unsupervised Learning on the JSE

Laher, Muhammad

2024
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/44140
Laher, Muhammad. (2024). Pairs Trading via Unsupervised Learning on the JSE [Master's dissertation, University of the Witwatersrand, Johannesburg].WireDSpace.

Downloaded from a DSpace repository. https://dspace.org/

School of Economics and Finance
Faculty of Commerce, Law and Management

Pairs Trading via Unsupervised Learning on the JSE

Master of Commerce (50% Research) in Finance

Muhammad Laher **[1364040]**



Supervisor: Prof. Yudhvir Seetharam

School of Economics and Finance

February 2024

Abstract

Pairs trading, a strategy that capitalises on temporary price discrepancies between two correlated assets, has garnered attention for its potential to generate profits in financial markets. This research explores the viability of employing unsupervised learning techniques for pairs trading on the Johannesburg Stock Exchange (JSE). Using clustering algorithms to identify pairs and considering both price data and firm characteristics, the study examines the performance of pairs trading portfolios constructed via different clustering methods. Empirical results reveal that while agglomerative clustering shows promise with the highest monthly mean return for long-short portfolios, none of the strategies consistently outperform benchmark indices. Furthermore, considering only momentum features in the clustering process leads to deteriorated portfolio performance, emphasizing the importance of incorporating firm characteristics. Despite the potential benefits offered by unsupervised learning, challenges such as the limited number of listed stocks and algorithm selection hinder the strategy's effectiveness on the JSE. The findings suggest that further research is needed to refine methodologies and address practical implementation challenges for pairs trading strategies in emerging markets like the JSE.

Contents

Abstract **2**
1. Introduction **4**
2. Literature review **6**
3. Description of Data and Research Methodology **10**
3.1. Data 10
3.1.1 Momentum Features 11
3.1.2 Firm Characteristics 11
3.1.3 Data Preprocessing 12
3.2. Unsupervised Learning (Clustering Algorithms) 13
3.2.1 K-Means Clustering (MacQueen, 1967) 13
3.2.2 DBCSAN (Ester, Kriegel, Sander, & Xu, 1996) 14
3.2.3 Agglomerative Clustering (Johnson, 1967) 14
3.3. Portfolio Construction and Strategy 15
4. Empirical Results **16**
4.1. Clustering Characteristics 17
4.2. Strategy Results 17
4.3. Effects of Firm Characteristics 23
5. Conclusion **28**
6. References **30**
Appendices **32**
Appendix 1: Firm Feature Variables 32
Appendix 2: Python Code 32

1. Introduction

Pairs trading involves identifying two closely related financial instruments that historically exhibit a strong correlation in their price movements (Goetzmann, Gatev, & Rouwenhorst, 1999). The two instruments are referred to as a "pair" and are expected to move in a highly correlated manner due to their underlying characteristics. The main idea behind pairs trading is that while the long-term correlation between the two instruments remains stable, in the short-term, there may be periods when one security deviates from its historical relationship with the other. When such a deviation occurs and the pair's prices diverge, investors can take advantage of the expected reversion to their historical correlation (Goetzmann, Gatev, & Rouwenhorst, 1999).

To execute the pairs trading strategy, an investor typically takes a long position in the underperforming security and simultaneously takes a short position in the outperforming security, aiming to profit from the convergence of the two prices as they revert to their mean correlation. The success of the strategy depends on the ability to accurately identify pairs with stable correlations and exploit temporary price discrepancies effectively (Goetzmann, Gatev, & Rouwenhorst, 1999). Goetzmann, Gatev, and Rouwenhorst (1999) hypothesised that if U.S markets were efficient at all times, then risk-adjusted returns from pairs trading should not be positive. However, the study found average annualised excess returns of 12 percent for pairs portfolios.

Traditional pairs trading strategies typically identify pairs based on statistical measures such as distance metrics or cointegration. The distance method is a simple algorithm for choosing pairs, minimising the sum of squared deviations between two normalised price series on a constructed cumulative total return index. Cointegration is a statistical property of time series variables, two time series are said to cointegrated if there exists a linear combination of the two-time series that is stationary (Rad, Low, & Faff, 2016). Unsupervised learning, on the other hand, uses clustering algorithms to identify pairs based on similarities in both price movement and other firm characteristics (Han, He, & Toh, 2021).

Unsupervised learning is a machine learning technique used to identify patterns in data sets that are neither classified nor labelled. This can be applied to large financial datasets with high

dimensions. Traditional pairs trading strategies make use of cointegration or distance methods, derived from past price data to identify correlated pairs (Gatev, Goetzmann, & Rouwenhorst, 2006; Stander, Marais, & Botha, 2013; Rad, Low, & Faff, 2016).

Machine learning has become more popular and has been widely applied in various fields, including finance, in recent years. Barclays estimated that 56 percent of hedge funds utilised machine learning in 2018 while only 20 percent used machine learning in 2017 (Chahn, 2018). Han, He and Toh (2021) found that incorporating firm characteristics together with price data significantly improve the performance of a pairs trading strategy. Applied to the US stock market from January 1980 to December 2020, the portfolio constructed via agglomerative clustering (Johnson, 1967), an unsupervised learning algorithm, earned a statistically significant annualised mean return of 24.8 percent and a Sharpe ratio of 2.69. During the same period the S&P500 had an annualised return of 12.6 percent with a Sharpe ratio of 0.835.

An extension of this is to test this on the Johannesburg Stock Exchange (JSE) to explore if this strategy would be profitable historically. Unsupervised learning is also rarely used in finance literature due to limited perceived application. This is one of the earliest works to examine its application in a South African context. Profits from pairs trading strategies on the JSE are disappointing as they are mainly consumed by trading costs (Stander, Marais, & Botha, 2013). The use of unsupervised learning algorithms applied to price data as well as firm characteristics showed significant profits, with monthly excess returns of 1.9 percent in the US market (Han, He, & Toh, 2021). The study aims to test whether a similar result appears on the JSE historically.

Adapting the methodology of Han, He, and Toh (2021), three clustering algorithms are selected and applied to JSE data from January 2010 to December 2023. K-Means clustering (MacQueen, 1967), density-based spatial clustering of applications with noise (DBSCAN) (Ester, Kriegel, Sander, & Xu, 1996) and agglomerative clustering (Johnson, 1967) are chosen as representatives of different categories of clustering algorithms. Two sets of data are used, the first set containing momentum features made up of 5 different price momentums for each stock. The second set contains firm features made up of 20 variables. The strategy consists of two steps. Stocks are first clustered based on their past returns and characteristics. A trading rule is then set up based on the

divergence of the past one-month returns within each cluster. Portfolios are formed at the end of each month and positions are held for one month before rebalancing. The results are compared to the JSE All Share Index (ALSI) and the JSE Top 40 Index (TOP40). Over the sample period, none of the constructed portfolios outperform the benchmarks (ALSI or TOP40 Index) on a risk adjusted basis. The long only portfolio using agglomerative clustering earns an annualised return of 10.7 percent over the period with a Sharpe ratio of 0.525, over the same period the ALSI returned 4.9 percent with a Sharpe ratio of 0.473 while the TOP40 had an annualised return of 5.2 percent with a Sharpe ratio of 0.488.

The importance of including firm features is tested by clustering stocks based on the momentum features only and comparing the results. It is found that portfolio performance decreases for all portfolios and all clustering methods by removing the firm features. This indicates that firm features contain important information that have an important role in clustering and identifying stocks using unsupervised learning algorithms.

The rest of the paper is organised as follows. Section 2 reviews existing literature on pairs trading, machine learning and pairs trading on the Johannesburg Stock Exchange (JSE). Section 3 describes the data and variables used; it also provides details on unsupervised learning, the clustering methods used, the trading algorithm and portfolio formation. Section 4 presents the results of the strategies and their performance over time; it also shows the results of the strategies by excluding firm features. Section 5 concludes.

2. Literature review

This literature review provides a comprehensive overview of the evolution of pairs trading strategies, encompassing seminal works by Goetzmann, Gatev, and Rouwenhorst (1999) on the distance method, as well as subsequent studies on copula-based approaches by Liew and Wu (2013) and Rad, Low, and Faff (2016). These studies shed light on the diversification benefits, risk-adjusted returns, and performance under varying market conditions associated with pairs trading strategies. Furthermore, recent innovations, such as the application of unsupervised learning techniques introduced by Han, He, and Toh (2021), offer promising avenues for further exploration, particularly in emerging markets like the Johannesburg Stock Exchange (JSE).

Goetzmann, Gatev and Rouwenhorst (1999) in the seminal work on Pairs Trading, test the risk and return characteristics of pairs trading with daily data from 1962 to 1997 and extended to 1962 to 2002 (Gatev, Goetzmann, & Rouwenhorst, 2006). A simple algorithm for choosing pairs is used, minimising the sum of squared deviations between two normalised price series on a constructed cumulative total return index, known as the distance approach. The strategy is found to have average annualised returns of 12% for top-pairs portfolios. These profits were uncorrelated to the S&P 500 but did exhibit low sensitivity to the spreads between small and large stocks and between value and growth stocks in addition to the spread between high grade and intermediate grade corporate bonds and shifts in the yield curve. Their view is that abnormal returns to pairs strategies are compensation to arbitrageurs for enforcing the "Law of One Price". The two pieces of evidence given to support this view are: firstly, while raw returns have fallen risk-adjusted returns have persisted despite increased hedge fund activity. Secondly, the results suggest that the change in risk-adjusted returns of pairs trading is accompanied by the lower importance of a common factor that drives the returns to pairs strategies. The study also shows the diversification benefits of combining multiple pairs in a portfolio, as the number of pairs in a portfolio increases the standard deviation of the portfolio decreases. Results are shown for the long and short portfolios separately, where it is shown that the pairs risk-adjusted excess returns come from the short portfolio, which are made up of stocks that have increased in value relative to their respective paired stock. The asymmetry of the results of the long and short portfolios show that the returns are not due to simple one-month mean reversion.

This profitable strategy using the distance approach experienced a declining trend in the following years, where the sample was extended to 2009 (Do & Faff, 2010). The mean excess return between 2003 and 2009 was 0.24 percent, down from 0.37 percent from 1989 to 2002. The Sharpe ratios over this period also declined from 1.04 between 1962-1988 to 0.43 from 1989 to 2002 and then to 0.34 between 2003 and 2009. This was partly attributed to increased competition in the hedge fund industry while worsening arbitrage risks in various pairs portfolios accounted for 70% of the decline. Pairs trading performance was particularly strong during market downturns, specifically the global financial crisis. Two additional metrics were incorporated into the study, which were found to significantly enhance trading profits: industry homogeneity and the frequency of

historical reversal in the price spread. Industry homogeneity is a first step to incorporating a fundamental aspect in pairs trading. Do and Faff (2010) stated that feasible strategies should start by defining fundamentally homogeneous asset groups because doing so helps not only avoid unnecessary search costs but also reduce nonconvergence risk. Fundamentally similar assets are likely to converge, and if they do not, they are likely not to drift apart (Do & Faff, 2010).

Liew and Wu (2013) explore the use of copulas, a statistical tool used to establish the dependence structure between variables. The objective of the pairs trading technique using copula approach is to apply the optimal copula between two stock returns and identify the relative positions between stock pairs. A copula is applied in a two-step approach, where the best-fitting marginal distribution is first determined for each variable, and then a suitable copula is used to describe the dependence structure between the variables. The copula approach, which involves applying the optimal copula between two stock returns to identify the relative positions between stock pairs, showed consistent association between variables in both time periods. The empirical results demonstrate that the copula approach for pairs trading is superior to the conventional distance and cointegration methods. The copula approach resulted in higher returns, 26.7 percent, 11.76 percent, and 11.66 percent compared to 19.27 percent, 11.7 percent and 10.98 percent for the distance approach for the three constructed pairs. The copula approach also resulted in more trading transactions compared to the other approaches which failed to demonstrate consistency in the behaviour of spread between the formation period and trading period.

Rad, Low and Faff (2016) perform an extensive study on the performance of three different pairs trading strategies: distance, cointegration and copula methods. The distance, cointegration, and copula methods show a mean monthly excess return of 91, 85, and 43 bps (38, 33, and 5 bps) before transaction costs (after transaction costs), respectively. The findings of Do and Faff (2010) are confirmed, from 2009 the frequency of trading opportunities via the distance method is reduced. A similar result is found for the cointegration method while the frequency remains stable for the copula method. The cointegration method performed superior to the other methods in turbulent market conditions although all strategies performed well during these periods. Although the copula method had a weaker performance in terms of returns compared to the distance and cointegration

methods, the frequency of its trades has not fallen and as such results in a more stable economic performance over time.

Stander, Marais, and Botha (2013) use a copula approach to identify trading opportunities on the Johannesburg Stock Exchange using the mathematical structure that captures the relationship between the two equities or the bivariate dependence structure of two equities. The relationships between the equity pairs are modelled with bivariate copulas and is applied to JSE data over the period from January 2002 to December 2009. The results of the pairs-trading strategy show that while it leads to profits in most cases, the profits are largely consumed by trading costs. For the FSR-RMH pair highlighted in the study 14 trades which were profitable before trading costs led to a loss once trading costs were accounted for. Before trading costs 25 of the 26 trades on the pair were profitable whereas only 11 of the 26 trades were profitable after trading costs.

Han, He and Toh (2021) introduce a pairs trading strategy using unsupervised learning, which incorporates firm characteristics and price information through momentum features to identify pairs. Three clustering algorithms are chosen to represent categories of clustering algorithms, K-Means (MacQueen, 1967), DBSCAN (Ester, Kriegel, Sander, & Xu, 1996) and agglomerative clustering (Johnson, 1967). Stocks either form part of a cluster or are identified as an outlier. Within each cluster a long position is taken in the stocks with the lowest previous one-month momentum and a short position is taken in the stocks with the highest previous one-month momentum. The strategy is tested on the US stock market from December 1979 to November 2020. The study shows that the long-short portfolio constructed through agglomerative clustering achieves a statistically significant annualised mean return of 24.8% and a Sharpe ratio of 2.69 while the K-Means and DBSCAN algorithms has Sharpe ratios of 2.34 and 2.04 respectively. The agglomerative strategy attained an annualised Sharpe ratio of 1.73 and a mean return of 15.9% after subtracting 20 basis point transaction costs. The study shows that clustering stocks based on their characteristics and past returns reduced volatility and downside risk, improving performance. When stocks are clustered using only price information, the performance diminishes; the Sharpe ratio decreases from 2.34 to 1.76 for K-Means clustering, 2.04 to 1.57 for DBSCAN, and 2.69 to 1.44 for agglomerative clustering. This result supports the authors hypothesis that firm characteristics play a non-trivial role in identifying pairs. As in previous studies, the strategies

performed well during financial crises, such as the 2007 financial crisis and the 2020 market crash. Unsupervised learning concepts can be applied to the SA market to determine if similar results are observable in the SA market.

In conclusion, the literature reviewed highlights the dynamic nature of pairs trading strategies and the ongoing search for innovative methodologies to increase performance and adapt to changing market conditions. While traditional approaches have provided valuable insights and demonstrated profitability (Gatev, Goetzmann, & Rouwenhorst, 2006) (Do & Faff, 2010), these returns have been declining over time. Recent studies have highlighted the potential of advanced statistical techniques and machine learning algorithms, such as copulas (Rad, Low, & Faff, 2016) and unsupervised learning (Han, He, & Toh, 2021), to further improve pairs trading strategies.

3. Description of Data and Research Methodology

The data and research methodology employed in this study aim to investigate the viability of pairs trading strategies on the Johannesburg Stock Exchange (JSE) using unsupervised learning techniques. Two sets of features, encompassing momentum factors derived from price data and firm characteristics sourced from accounting and asset pricing perspectives, are generated for each month from January 2010 to December 2023. The dataset, collected from Bloomberg, consists of all JSE-listed stocks and is filtered based on liquidity requirements. Momentum features provide insights into historical price movements, while firm characteristics offer additional fundamental information. Data preprocessing techniques, including normalization and principal component analysis (PCA), are applied to ensure robustness and reduce dimensionality. Subsequently, three unsupervised clustering algorithms; K-Means, DBSCAN, and agglomerative clustering are employed to group stocks into clusters based on similarities in features. The resulting clusters are then used to form pairs for a long-short portfolio. The methodology follows from Han, He and Toh (2021), with a focus on consistency, and relevance to the JSE context.

3.1. Data

The data used for the study consists of two sets of features as in Han, He and Toh (2021). The two feature sets generated are momentum features incorporating price data, and firm characteristics used in Green, Hand, and Zhang (2016) which provide fundamental and accounting information.

The feature set consists of 5 momentum factors and 20 firm characteristics which are generated for every month in the sample period from January 2010 to December 2023.

Data is collected on all Johannesburg Stock Exchange (JSE) listed stocks from January 2010 to December 2023 from Bloomberg. The Bloomberg price considers the effects of any dividends and corporate actions during the period. Monthly data is used as is done in Han, He and Toh (2021) as this allows for consistent monthly rebalancing of portfolios. After empirical analysis, a liquidity filter is added such that only stocks with a monthly ZAR trading volume of at least R100 million are considered. This is to ensure sufficient trading volume and to ensure returns are not driven by illiquidity.

3.1.1 Momentum Features

Momentum features give information on the historical movement of stock prices, stocks with similar momentum features are expected to move together in the future (Han, He, & Toh, 2021). Han, He, and Toh (2021) calculate 1-month to 48-month price momentum on their US market dataset, due to the limited number of listed stocks on the JSE only 1, 3, 6, 12 and 24-month price momentums are calculated. The *i-month* price momentum at the end of month $t-1$ is defined as the cumulative return from month $t-i$ to $t-2$ for $i > 1$ and as the previous one-month return for $i = 1$.

$$\begin{aligned} \text{mom}_i &= r_{t-1}, && i = 1, \\ \text{mom}_i &= \prod_{j=t-i}^{t-2} \left(r_j + 1\right) - 1 && i \in \{3, 6, 12, 24\} \end{aligned} \tag{1}$$

where r_j denotes the return in month j.

3.1.2 Firm Characteristics

Firm characteristics add more information from an accounting and asset pricing view as stocks that have moved together may diverge if they have certain characteristics and are considered forward-looking (Han, He, & Toh, 2021). Firm characteristics as used in Han, He and Toh (2021) are adapted from the methodology of Green, Hand, and Zhang (2016) which consists of 78 firm

features. For the purposes of this study twenty variables have been used, this is due to the number of stocks available on the JSE as well as the variables available on JSE listed stocks. The variables are sourced from Bloomberg for the same sample period from January 2010 to December 2023. The full list of variables used can be found in Appendix 1.

3.1.3 Data Preprocessing

Unsupervised clustering algorithms cluster data based on distance measures and as such variables are normalised using their cross sectional mean and standard deviation to ensure features with higher magnitude do not skew the weighting in the algorithm (Han, He, & Toh, 2021). Principal Component Analysis (PCA) (Pearson, 1901) is a procedure that converts a set of possibly correlated variables into a set of linearly uncorrelated variables. Certain features contain more information than others for the identification of similar stocks, but the distance measures used in clustering assign the same weight to all features (Han, He, & Toh, 2021). By inputting the features directly into a clustering algorithm, it can dilute the information contained in important features. PCA is applied to the features and the principal components are used as the inputs into the clustering algorithms. This allows the important features to be emphasised (Han, He, & Toh, 2021). Another benefit of PCA is to alleviate the curse of dimensionality (Pearson, 1901).

The number of components is chosen such that it explains at least 99% of the variation. In this study an average of 18 principal components are selected. Figure 1 below shows the variance as the number of components increase based on the feature set in this study:

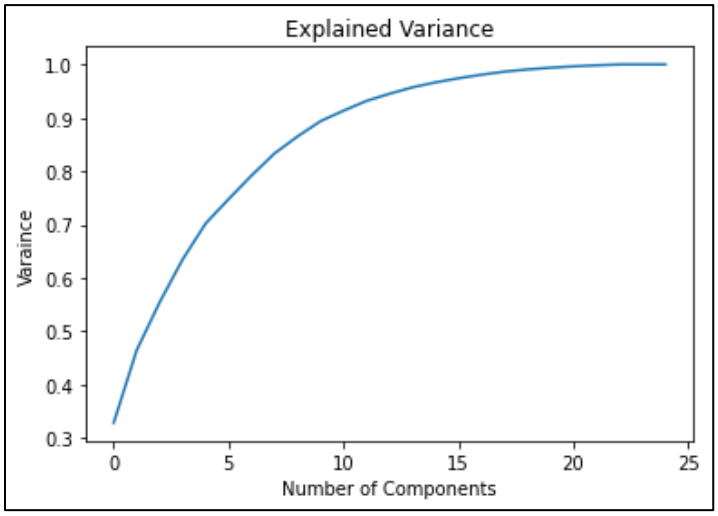


Figure 1: Number of PCA components. Source: From this study.

3.2. Unsupervised Learning (Clustering Algorithms)

Unsupervised learning uses unlabelled, unclassified, and categorised training data with the main goal of discovering hidden and interesting patterns in the unlabelled data (Wittek, 2014). Clustering is the most common unsupervised learning algorithm used to explore the data analysis to find hidden patterns or groupings in the data (Wittek, 2014). Three clustering algorithms are used: K-Means clustering which is a partition-based algorithm, DBSCAN which is density based and agglomerative clustering which is a hierarchical clustering algorithm.

3.2.1 K-Means Clustering (MacQueen, 1967)

K-means clustering (MacQueen, 1967) is a popular algorithm for grouping data points into clusters by minimising the within-cluster sum of squares (WCSS). It involves assigning data points to clusters and updating centroids iteratively until convergence. The objective function is given by:

$$W = WCSS = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2, \quad (2)$$

where x^i is the *i-th* data point, μ_k the centroid of cluster *k*, w_{ik} = 1 if x^i belongs to cluster *k* or *0* otherwise, and *N* is the total number of data points.

The K-Means algorithm does not identify outliers, which can be addressed using a modified method. The method of Hautamaki, Cherednichenko, Karkkainen, Kinnunen and Franti (2005) is used to identify outliers (Han, He, & Toh, 2021). For each data point, the distance to its centroid and the distance to its nearest neighbour is measured, the distances are sorted to the nearest neighbour in ascending order and the distance is then chosen at α percentile as the threshold, ε, any data point whose distance to its centroid is greater than ε is classified as an outlier and removed.

We set the number of clusters *K = 2, 5 and 10*. The, ε, distance parameter is set to the average distance between nearest neighbours (Han, He, & Toh, 2021). Any stock whose distance to its centroid is greater than ε is regarded as an outlier and removed. Han, He, and Toh, (2021) set K

ranging from 2 to 1500, however due to the limited number of stocks on the JSE we limit the number of clusters to 10.

3.2.2 DBCSAN (Ester, Kriegel, Sander, & Xu, 1996)

Density-based spatial clustering of applications with noise (DBSCAN) identifies areas of high density in a high dimensional data set separated by regions of low density. It requires two parameters, the minimum number of datapoints per cluster, $minpts$, and the maximum distance between datapoints, ε, to be considered in the same cluster.

DBSCAN creates a circle of ε radius around every data point and classifies them into either a core point, border point, or noise. A data point is a core point if the circle around it contains at least $minpts$ number of points. If the number of points is less than $minpts$, then it is classified as border point, and if there are no other data points around any data point within ε radius, then it is treated as noise.

As in Han, He and Toh (2021) we choose l_1 norm as the distance metric for DBSCAN. $minpts$ is set as the natural logarithm of the total number of datapoints. The minimum distance, ε, is set as the average of the distances to the nearest neighbours (Han, He, & Toh, 2021).

3.2.3 Agglomerative Clustering (Johnson, 1967)

Agglomerative clustering (Johnson, 1967) is a hierarchical clustering method that groups data points based on their similarity. Agglomerative requires the specification of one of two hyperparameters: the number of clusters K or the maximum distance for clusters to be merged, ε, known as linkage distance. Figure 2 below from Han, He, and Toh (2021) demonstrates the process:

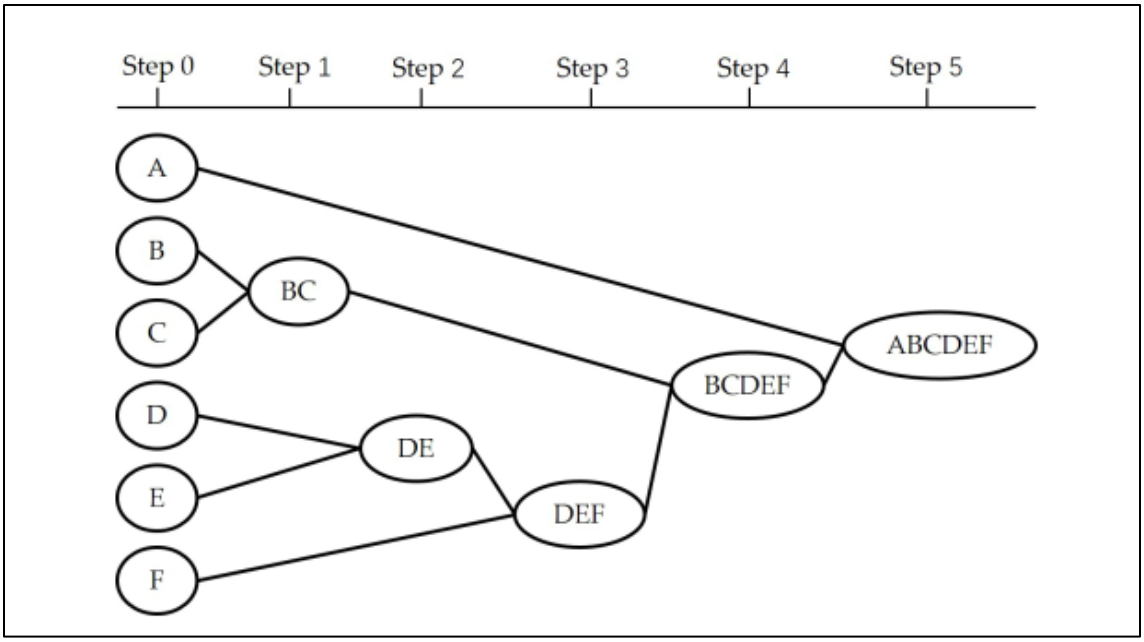


Figure 2: Agglomerative Clustering (Han, He, & Toh, 2021)

The six individual data points are treated as individual clusters. Their proximity to each other is calculated, and similar clusters are merged to form a new cluster (BC and DE), should their distance be less than ε. This merging process is repeated until there are no more clusters whose distance to another cluster is less than ε. If the distance between merged clusters is always below the specified ε, eventually, all clusters are merged to form cluster *ABCDEF* (Han, He, & Toh, 2021).

We set the maximum distance, ε, to be the average distance between a pair of nearest neighbour datapoints (Han, He, & Toh, 2021).

3.3. Portfolio Construction and Strategy

During each month in the sample period stocks must have a complete feature set to be considered for clustering, all five momentum features as well as the twenty firm characteristics must be populated. Stocks with an incomplete set are not considered for that specific month's portfolio.

Stocks are clustered using a clustering method (unsupervised learning) where they are either assigned to a cluster or identified as an outlier. Stocks within each cluster are sorted based on the 1-month price momentum, the highest momentum stock is paired with the lowest momentum stock and so on, within each cluster. An equally weighted long-short portfolio is formed using the pairs

whose 1-month momentum difference is greater than the cross-sectional standard deviation of all pair's momentum differences. Positions are held for one month and the portfolio is then rebalanced. The 2 benchmarks used for comparison are the FTSE/JSE All Share Index (ALSI) and the FTSE/JSE Top 40 Index (TOP40). A 25-basis point trading cost is applied to each trade, this follows from Han, He, and Toh (2021) who use a 10 basis point trading cost for the US market, while Stander, Marais, and Botha, (2013) use a securties transfer tax of 0.25 percent for their JSE related study. From these studies, A limting assuption of 25bps trading costs has been applied with a note that his should be further refined for future research.

The portfolios are seperated into long only and short only portoflios for further analysis. Gatev, Goetzmann and Rouwenhorst (2006) provide three arguments for separately examining the long and short portfolios that make up a pairs trading position. The first is to simply get further insights into the question of mean reversion, if pairs trading exploits mean reversion one would expect the abnormal returns to the long and short positions to be equal as the opening of a pair is equally likely to be triggered by either stock in the pair. The second reason for separating the portfolios is to examine if excess returns are driven by the short positions. The third reason is to look at the risk metrics of the portfolios separately and examine the risk return profile of the portfolios.

The data collection and research methodology outlined in this section provide a robust framework for investigating pairs trading strategies on the Johannesburg Stock Exchange (JSE). Two distinct sets of features; momentum factors and firm characteristics, derived from historical data spanning from January 2010 to December 2023 are used. By applying data preprocessing techniques, including normalisation and principal component analysis (PCA), the dimensionality of the dataset is reduced while preserving crucial information for clustering analysis. The application of three unsupervised clustering algorithms; K-Means, DBSCAN, and agglomerative clustering allows for the identification of stock clusters and pairs for portfolio formation.

4. Empirical Results

In this section the empirical results are presented. The clustering characteristics of each clustering algorithm are summarised. The financial returns and risks are then summarised for each clustering

algorithm for the long-short, long only and short only portfolios. The impact of firm features are assessed by comparing the performance of these portfolios with momentum features only.

4.1. Clustering Characteristics

We begin by analysing the clustering characteristics of the three clustering methods. Table 1 below shows the clustering results from K-Means, DBSCAN and agglomerative clustering. Stocks are clustered every month during the sample period, from January 2012 to December 2023. The reported values are time series averages.

Table 1; Clustering Characteristics (Time Series Average)

	K-Means (10)	DBSCAN	Agglomerative
Number of Stocks in Total	74	74	74
Number of Clusters	8	1	12
Number of Outliers	48	60	44
Number of Stocks Traded	26	14	30

An average of 74 stocks has a complete set of momentum and firm features and can be clustered every month. This is significantly less than the 3157 stocks which are available in the US market in Han, He and Toh (2021).

K-Means (10), where the number of clusters chosen is 10 is the best performing strategy from the other K-Means instances (K = 2, 5, 10). An average of 8 clusters are formed each month after removing outliers and an average of 26 out of the 74 stocks are clustered. DBSCAN forms only one large cluster each month with 14 out of the 74 stocks. Agglomerative clustering forms 12 clusters which compared to DBSCAN forms many small clusters which covers 30 out of the 74 stocks.

4.2. Strategy Results

Table 2 shows the monthly time series average return statistics of the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms, it also reports the ALSI and TOP40 metrics for comparison, over the period from January 2012 to December 2023.

Table 2: Monthly Return Summary Statistics (Time Series Average)

	K-Means (10)			DBSCAN			Agglomerative			Benchmarks	
	L	S	L-S	L	S	L-S	L	S	L-S	ALSI	TOP40
Mean	0.003	-0.002	0.001	0.008	-0.004	0.003	0.011	-0.007	0.005	0.007	0.007
Standard Deviation	0.024	0.020	0.020	0.062	0.058	0.066	0.060	0.049	0.054	0.040	0.043
Min	-0.068	-0.061	-0.048	-0.249	-0.234	-0.229	-0.220	-0.177	-0.105	-0.128	-0.112
25%	-0.011	-0.013	-0.011	-0.020	-0.030	-0.035	-0.017	-0.033	-0.029	-0.022	-0.024
50%	0.005	-0.002	-0.002	0.007	-0.003	-0.004	0.010	-0.006	0.004	0.008	0.006
75%	0.015	0.009	0.013	0.036	0.023	0.047	0.036	0.023	0.029	0.032	0.035
Max	0.099	0.063	0.055	0.330	0.232	0.256	0.426	0.142	0.300	0.131	0.142

From the results presented in Table 2 we observe that none of the long-short (L-S) portfolios which are constructed with pairs have a higher mean return than the ALSI or TOP40 benchmarks of 0.7 percent. This indicates that pairs trading using the selected clustering algorithms do not produce positive excess returns on the Johannesburg Stock Exchange (JSE). For the L-S portfolios, the agglomerative clustering algorithm gives the highest monthly mean return and has the highest individual monthly return of 30%. Overall, none of the L-S portfolios have a monthly mean return greater than the benchmarks while only the K-Means (10) L-S portfolio has a lower standard deviation than the benchmarks.

All the long only portfolios have a positive monthly mean return with the DBSCAN and agglomerative algorithms giving a higher monthly mean return (0.8 percent and 1.1 respectively) than the benchmarks. The standard deviations of the DBSCAN and agglomerative clustering algorithms (0.062 and 0.06) are higher than the ALSI (0.04) and TOP40 (0.043) benchmarks. This indicates that there is more risk taken to earn the higher returns from these portfolios.

All the short only portfolios have a negative monthly mean return with the agglomerative clustering algorithm having the lowest mean return of -0.7 percent. This shows that there does not appear to exist a mean reverting nature within clusters for any of the algorithms. For there to be a mean reverting process within the clusters we would expect the returns for the long and short portfolios to be equal.

Table 3 reports annualised risk-return metrics of the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms (K-Means, DBSCAN and agglomerative), as well as for the ALSI and TOP40 indices for the period from January 2012 to December 2023. The Sharpe ratio (Sharpe, 1966), one of the most widely used risk return metrics, is used to compare the risk-return profiles of the different portfolios for each clustering algorithm.

Table 3: Annualised Risk-Return Metrics

	K-Means (10)			DBSCAN			Agglomerative			Benchmarks	
	L	S	L-S	L	S	L-S	L	S	L-S	ALSI	TOP40
Mean (before tc)	0.010	-0.057	-0.020	0.055	-0.069	0.014	0.107	-0.105	0.031	0.049	0.052
Standard Deviation	0.088	0.075	0.069	0.191	0.219	0.245	0.204	0.142	0.214	0.103	0.106
Sharpe Ratio (before tc)	0.113	-0.764	-0.282	0.290	-0.313	0.055	0.525	-0.742	0.143	0.473	0.488
Min	-0.149	-0.197	-0.161	-0.304	-0.567	-0.424	-0.098	-0.392	-0.218	-0.140	-0.137
Max	0.190	0.053	0.113	0.284	0.299	0.371	0.491	0.099	0.620	0.205	0.197
Profitable Years	8	2	7	8	5	7	7	3	7	8	9
Unprofitable Years	4	10	5	4	7	5	5	9	5	4	3

To analyse the performance of the pairs trading strategy we look at the performance of the long-short (L-S) portfolios. The annualised returns for the long-short (L-S) portfolios for the three clustering algorithms (K-Means, DBSCAN and agglomerative clustering) of -2 percent, 1.4 percent and 3.1 percent (before trading costs) respectively are all lower than the ALSI and TOP40 benchmarks of 4.9 percent and 5.2 percent. None of the long-short portfolios have mean annualised returns greater than the benchmarks nor do they have better Sharpe ratios than the benchmarks. This indicates the portfolios unable to outperform the benchmark from a return nor from a risk-return perspective. All the long-short portfolios have the same amount of profitable (7) and unprofitable years (5).

The long only (L) portfolios have a positive return for all clustering methods, with the agglomerative clustering algorithm performing the best earning an annualised return of 10.7 percent after trading costs. The Sharpe ratio for this portfolio is 0.525 which is higher than the benchmarks, this shows that the agglomerative clustering long only portfolio does have desirable return and risk-return characteristics and may be a viable investment portfolio for investors. The

DBSCAN long only portfolio earned an annualised return of 5.5 percent slightly beating the annualised benchmark returns, while having a Sharpe ratio of 0.290 which is lower than the benchmarks. All the short only (S) portfolios have negative returns and Sharpe ratios, confirming the findings from the monthly returns in Table 2 that there does not appear to be any mean reverting process within clusters for any of the clustering algorithms.

Table 4: Annualised Return and Sharpe Ratios after Trading Costs (25bps)

	K-Means (10)			DBSCAN			Agglomerative			Benchmarks	
	L	S	L-S	L	S	L-S	L	S	L-S	ALSI	TOP40
Mean (after tc)	0.006	-0.060	-0.022	0.038	-0.097	-0.017	0.091	-0.116	0.013	0.044	0.047
Sharpe Ratio (after tc)	0.141	-0.122	0.112	0.141	-0.122	0.050	0.191	-0.133	0.093	0.171	0.169

From Table 4 above we can see that the performance of each of the portfolios is diminished once accounting for trading costs. Only the agglomerative clustering long-only portfolio has a better annualised mean return and Sharpe ratio than the benchmarks after trading costs.

Figures 3,4 and 5 below show the growth of a R1 investment over time for the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms (K-Means, DBSCAN and agglomerative clustering) accounting for a trading cost of 25bps.

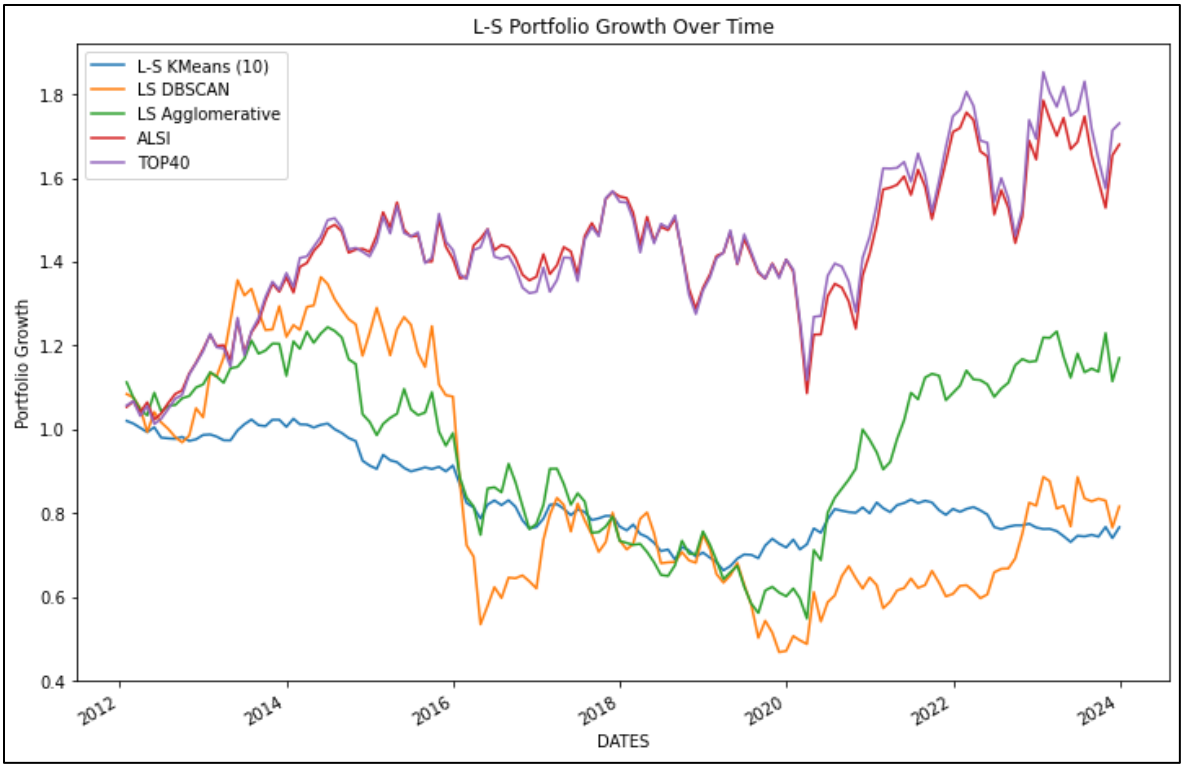


Figure 3: Long Short Portfolio Growth Over Time after Trading Costs (25bps)

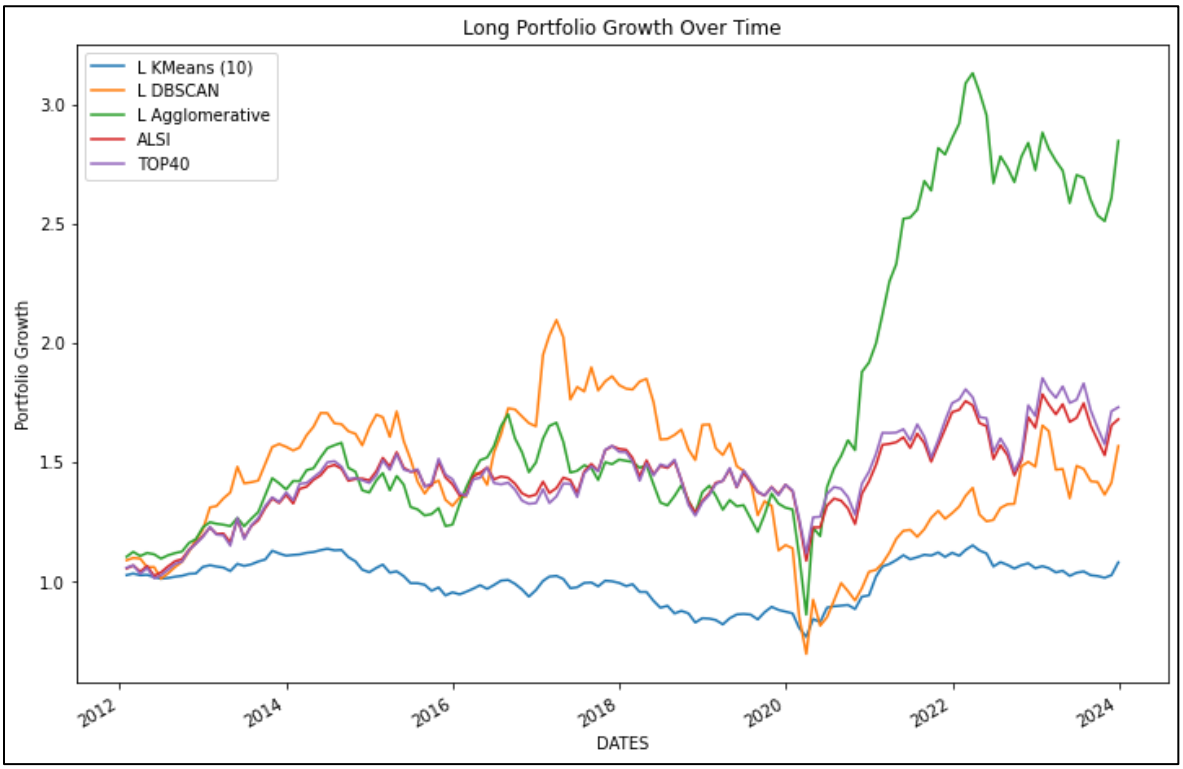


Figure 4: Long Only Portfolio Growth Over Time after Trading Costs (25bps)

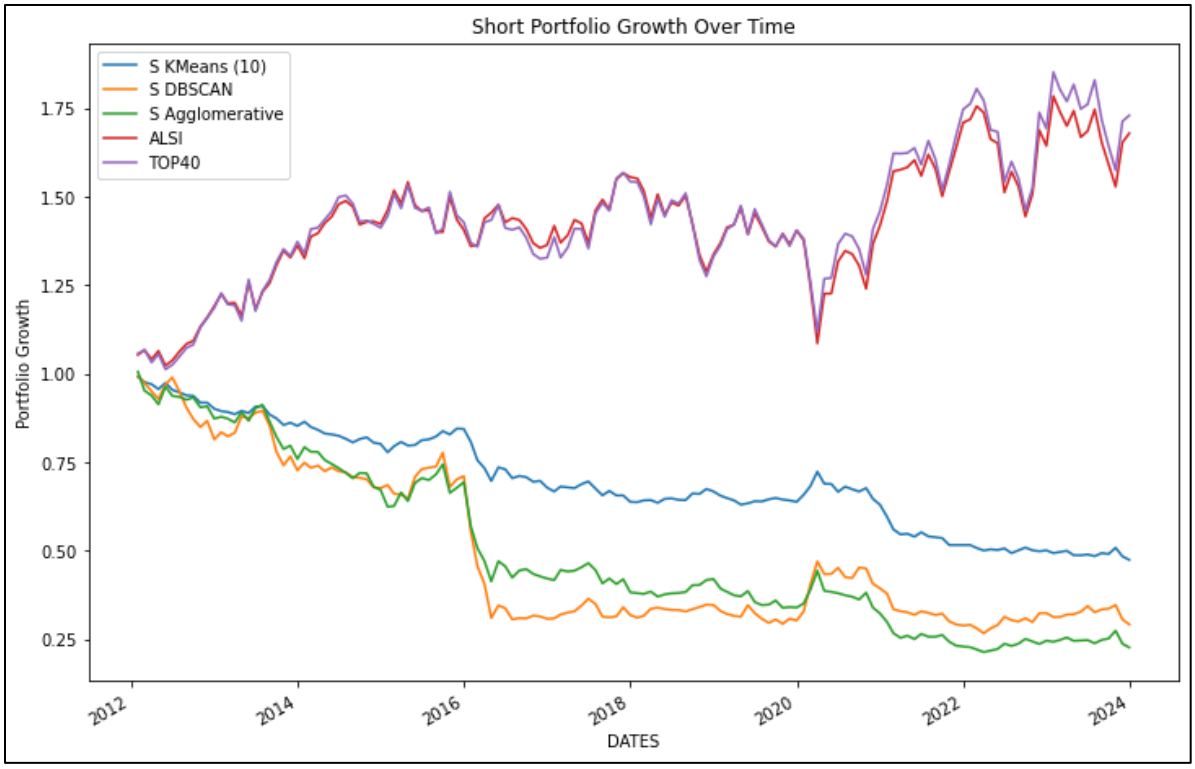


Figure 5: Short Only Portfolio Growth Over Time after Trading Costs (25bps)

From Figure 3 we see that none of the long-short (L-S) portfolios manage to outperform the benchmarks over time while only the L-S agglomerative clustering algorithm ends with a portfolio value greater than R1. This indicates that a pairs trading strategy based on clustering algorithms to form pairs does not perform well on the Johannesburg Stock Exchange (JSE). The long-short portfolios do not appear to perform better than the benchmarks in periods of financial distress, particularly during the COVID19 crash, December 2019 to February 2020, previous studies have shown that pairs trading strategies tend to perform well during times of marker turbulence (Do & Faff, 2010) (Rad, Low, & Faff, 2016). While Han, He, and Toh (2021) show that the pairs trading strategies using unsupervised learning also perform well during market crises. Our results show that this is not the case on the Johannesburg Stock Exchange (JSE).

For the long only portfolios from Figure 4 we observe that the DBSCAN algorithm manages to outperform the benchmarks between 2012 and 2019, however it then performs badly during the COVID19 crash and drops below the benchmarks and cannot recover. The agglomerative long only portfolio performs extremely well after the COVID19 crash and ends with the highest ending

portfolio value, significantly outperforming the benchmarks post the crash but does not outperform in the years prior to this. All the short only portfolios have negative performance, and the portfolios do not exceed the starting value at any point. This indicates that the strategy for selecting stocks on which to go short based on the previous months 1-month momentum is not a good signal of when to enter the short position. This has an effect on the pairs trading strategy as the long-short portfolio performance is diminished by the returns on the short positions.

4.3. Effects of Firm Characteristics

We assess the impact of firm features in the clustering algorithms by looking at the same results as in Section 4.2 but using only the momentum features in the feature set. This allows us to compare the results and performance of the portfolios including the firm features and the results of the portfolios excluding firm features. Han, He, and Toh (2021) show that by excluding firm features portfolio performance diminshes for all clustering methods inidcating that firm features play an important role in the clustering algorithms as well as strategy performance. Table 5 below shows the clustering characteristics for the three clustering algorithms after firm features are removed.

Table 5: Clustering Characteristics (Momentum Features Only)

	K-Means (10)	DBSCAN	Agglomerative
Number of Stocks in Total	101	101	101
Number of Clusters	8	2	17
Number of Outliers	63	89	87
Number of Stocks Traded	38	12	14

From table 5 we observe that he average number of stocks available to be clustered has increased, as there are less variables which may have had missing values as more stocks have a complete feature set each month. K-Means produces the same number of clusters on average as the complete feature set, 8. DBSCAN now produces 2 clusters on average compared to 1 cluster when including firm features. The agglomerative clustering algorithm now produces 17 clusters compared to 12 clusters when firm features were included. Agglomerative clustering also produces more outliers (87) than with the complete feature set (44) and as a result less stocks are traded each month.

Table 6 shows the monthly time series average return statistics of the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms using momentum features only, it also reports the ALSI and TOP40 metrics for comparison over the period from January 2012 to December 2023.

Table 6: Monthly Return Summary Statistics for Momentum Features Only (Time Series Average)

	K-Means (10)			DBSCAN			Agglomerative			Benchmarks	
	L	S	L-S	L	S	L-S	L	S	L-S	ALSI	TOP40
Mean	0.003	-0.003	0.000	-0.002	-0.001	-0.003	-0.001	-0.003	-0.003	0.007	0.007
Standard Deviation	0.024	0.021	0.020	0.072	0.054	0.062	0.037	0.028	0.029	0.040	0.043
Min	-0.128	-0.063	-0.087	-0.504	-0.139	-0.208	-0.223	-0.103	-0.131	-0.128	-0.112
25%	-0.008	-0.014	-0.010	-0.030	-0.030	-0.036	-0.010	-0.014	-0.017	-0.022	-0.024
50%	0.003	-0.005	0.001	0.000	-0.005	-0.007	0.000	-0.001	0.000	0.008	0.006
75%	0.014	0.009	0.011	0.029	0.023	0.036	0.009	0.004	0.007	0.032	0.035
Max	0.134	0.086	0.095	0.208	0.296	0.180	0.156	0.190	0.106	0.131	0.142

From Table 6 we can see that none of the long-short portfolios have a positive monthly mean return, the K-Means long-short portfolio had a monthly mean return of 0 percent when excluding firm features which is slightly less than the 0.1 percent monthly mean return when firm features were included while the monthly standard deviation of 0.2 percent is the same as when firm features were included. The DBSCAN and agglomerative clustering long-short portfolios both had a monthly mean return of -0.3 percent which is down from 0.3 percent and 0.5 percent respectively when firm features were included while both the standard deviations are higher than when firm features were included.

Table 7 reports annualised risk-return metrics of the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms (K-Means, DBSCAN and agglomerative) with momentum features only, as well as for the ALSI and TOP40 indices for the period from January 2012 to December 2023. The Sharpe ratio (Sharpe, 1966) is used to compare the risk-return profiles of the different portfolios for each clustering algorithm.

Table 7: Annualised Risk-Return Metrics (Momentum Features Only)

	K-Means (10)			DBSCAN			Agglomerative			Benchmarks	
	L	S	L-S	L	S	L-S	L	S	L-S	ALSI	TOP40
Mean	0.008	-0.069	-0.032	-0.061	-0.047	-0.074	-0.041	-0.063	-0.071	0.049	0.052
Standard Deviation	0.068	0.070	0.058	0.215	0.168	0.164	0.095	0.064	0.079	0.103	0.106
Sharpe Ratio	0.115	-0.983	-0.553	-0.284	-0.279	-0.450	-0.436	-0.989	-0.891	0.473	0.488
Min	-0.083	-0.208	-0.127	-0.445	-0.320	-0.308	-0.203	-0.185	-0.202	-0.140	-0.137
Max	0.188	0.034	0.071	0.217	0.320	0.196	0.087	0.006	0.114	0.205	0.197
Profitable Years	6	2	2	5	4	4	5	1	1	8	9
Unprofitable Years	6	10	10	7	8	8	7	11	11	4	3

From Table 7 above we observe that all the long-short portfolios have a negative annualised mean return and negative Sharpe ratios. The performance of all the long-short portfolios is diminished when firm features are excluded confirming the results of Han, He, and Toh (2021) that firm features play an important role in clustering stocks in a pairs trading strategy. Analysing the long only, we observe that the performance of these portfolios are also diminshed by exclduing firm features for all three clustering methods. We can thus extend the finding that firm features do not only improve clustering and portfolio performance for pairs trading but also when constructing other portfolios with long only strategies. Firm features contain important information that prove the clustering process and hence portfolio perfomance.

Figure 6,7 and 8 below show the growth of a R1 investment over time for the equally weighted long-short (L-S), long only (L) and short only (S) portfolios constructed via the three clustering algorithms (K-Means, DBSCAN and agglomerative clustering) using only momentum features, accounting for a trading cost of 25bps. These figures confirm our results from Tables 6 and 7 that portfolio performance is diminished for all clustering algorithms when excluding firm features.

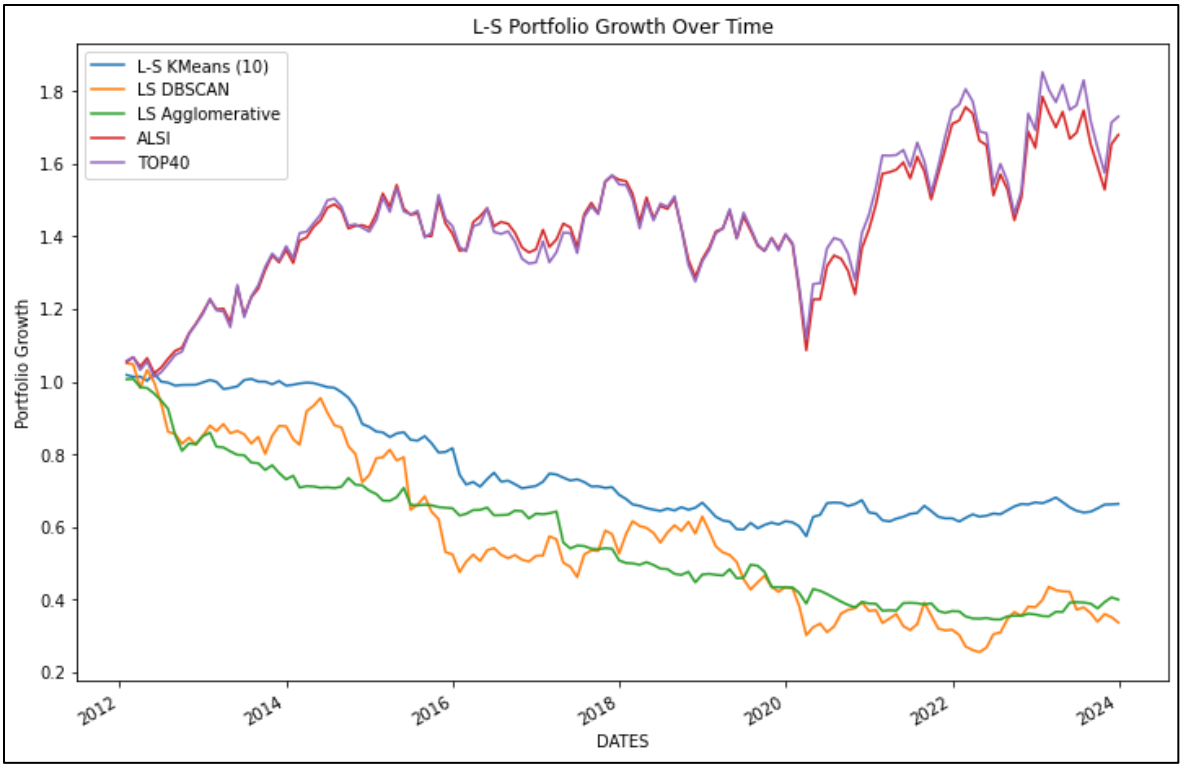


Figure 6: Long Short Portfolio Growth Over Time (Momentum Features Only)

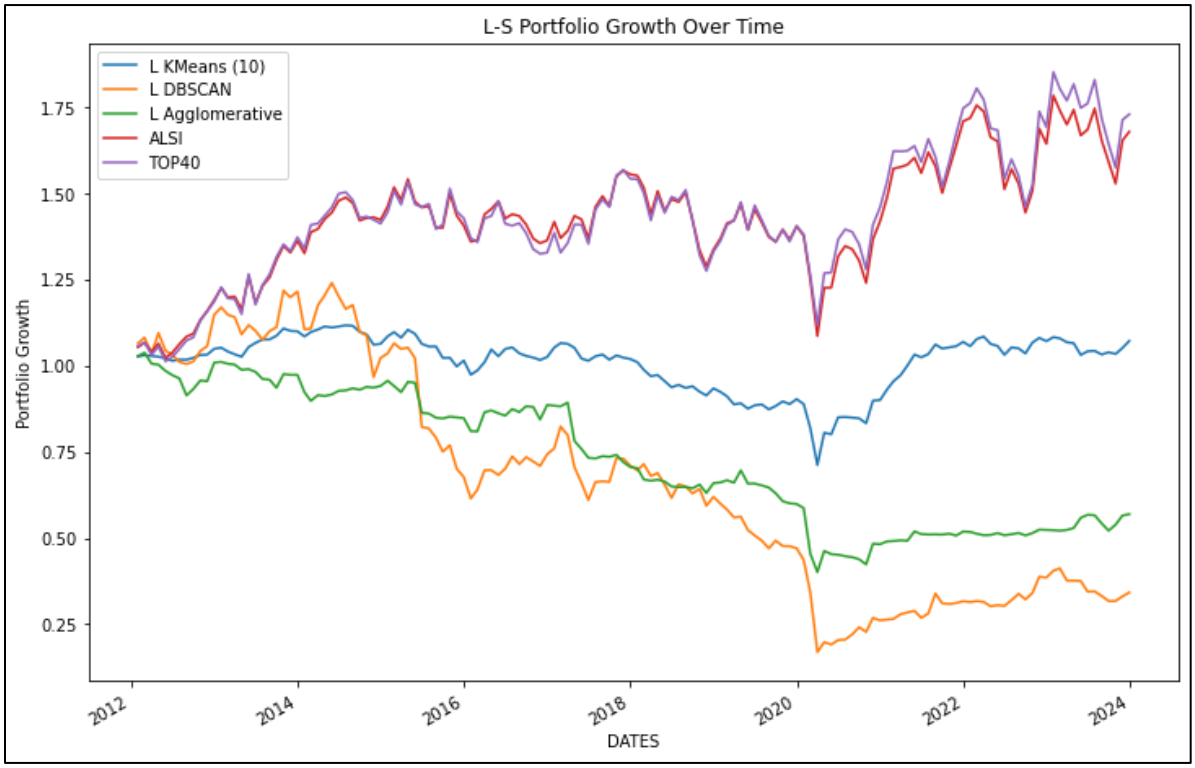


Figure 7: Long Only Portfolio Growth Over Time (Momentum Features Only)

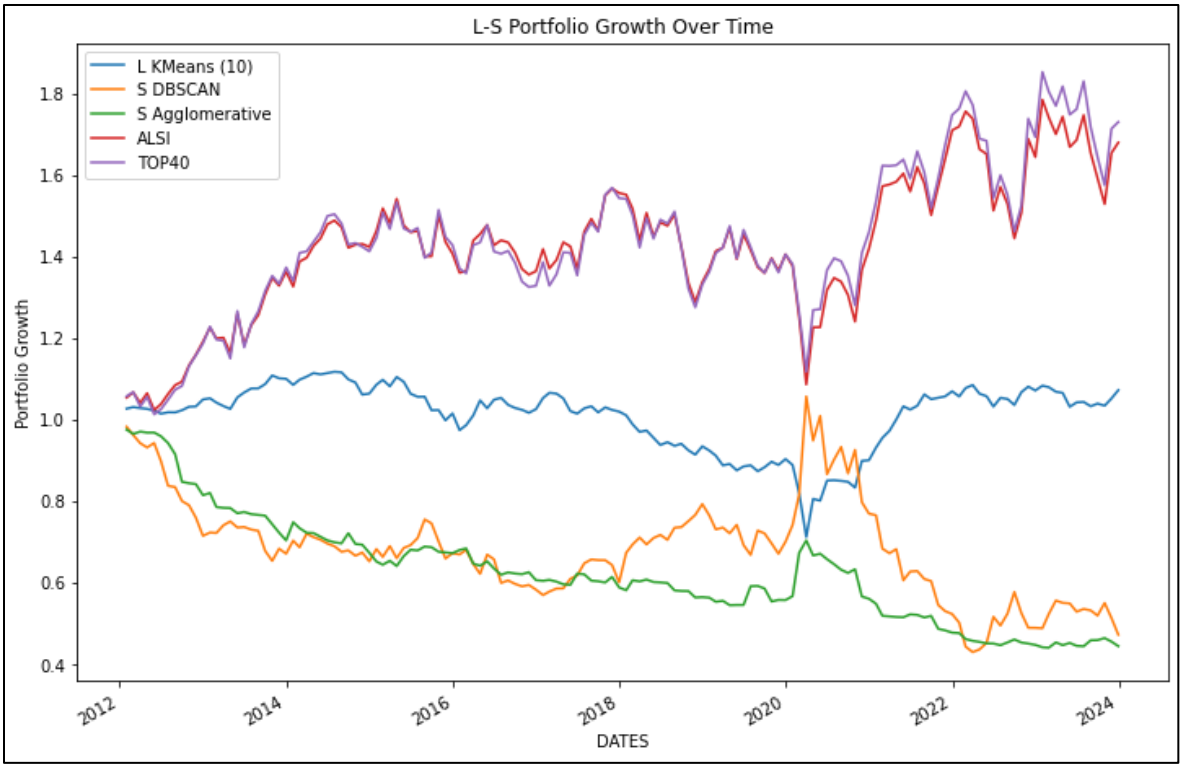


Figure 8: Short Only Portfolio Growth Over Time (Momentum Features Only)

In this section, a comprehensive analysis of the empirical results is obtained from applying various clustering algorithms to construct pairs trading strategies on the Johannesburg Stock Exchange (JSE). The clustering characteristics of each algorithm were carefully examined, revealing insights into the formation of clusters and the identification of outliers. Notably, the K-Means and agglomerative clustering algorithm managed to form between 8-12 distinct while the DBSCAN algorithm formed just one giant cluster on average.

Analysing the strategy performance of each algorithm, the analysis of monthly return statistics highlighted important findings. While none of the long-short portfolios outperformed the ALSI or TOP40 benchmarks, the agglomerative clustering algorithm exhibited the highest monthly mean return among them. It is worth noting that none of the long-short portfolios achieved a monthly mean return greater than the benchmarks, highlighting the challenge of generating positive excess returns with pairs trading strategies on the JSE. The long-only portfolios, specifically those constructed using DBSCAN and agglomerative clustering, showed promising results with positive monthly mean returns, with higher standard deviations, indicating increased risk.

Further analysis of annualised risk-return metrics and Sharpe ratios confirmed the underwhelming performance of the long-short portfolios, with none surpassing the benchmarks. However, the agglomerative clustering long-only portfolio demonstrated compelling performance, outperforming the benchmarks in terms of mean return, and exhibiting a higher Sharpe ratio, even after accounting for trading costs.

The assessment of firm characteristics revealed their significant impact on clustering algorithms and portfolio performance. Excluding firm features led to diminished portfolio performance across all clustering methods, highlighting the importance of incorporating firm-specific information in both clustering and portfolio construction processes.

5. Conclusion

The research conducted in this paper aimed to explore the profitability of pairs trading strategies on the Johannesburg Stock Exchange (JSE) using unsupervised learning algorithms. By adapting methodologies from previous studies (Han, He, & Toh, 2021) and incorporating firm characteristics alongside price data, the study sought to evaluate the historical performance of pairs trading on the JSE.

The empirical results indicate that while traditional pairs trading strategies have shown profitability in other markets, particularly when utilising advanced statistical techniques, the application of unsupervised learning algorithms on the JSE did not yield consistent outperformance compared to benchmark indices. Three clustering algorithms (K-Means, DBSCAN and agglomerative clustering) were used, incorporating momentum features and firm characteristics. None of the constructed long-short portfolios were able to outperform the ALSI or TOP40 index on a risk-adjusted basis over the sample period indicating that the use of unsupervised learning to construct pairs trading portfolios do not provide profitable results on the JSE.

The clustering methods (K-Means, DBSCAN, and agglomerative clustering) yield varying results, with agglomerative clustering forming the most clusters but with a higher number of outliers. This

suggests that the choice of clustering algorithm significantly impacts the composition of the pairs trading strategy. While agglomerative clustering shows the highest monthly mean return for the long-short portfolios, none of the portfolios demonstrate superior performance in terms of returns or risk-adjusted returns compared to the benchmarks. The growth of a R1 investment over time shows that none of the long-short portfolios manage to consistently outperform the benchmarks. While the agglomerative long-only portfolio performs well post-COVID19 crash, it fails to outperform in the years prior to this, suggesting limited consistency in performance.

The study found that the inclusion of firm features significantly improved the performance of the clustering algorithms, indicating the importance of incorporating fundamental information in pairs trading strategies. This confirms the finding of Han, He, and Toh (2021) that firm characteristics play a crucial role in identifying viable pairs and improving the effectiveness of unsupervised learning algorithms even on smaller markets with less stocks available such as the JSE.

The results indicate that unsupervised learning techniques applied to pairs trading, has limited effectiveness on the JSE. Challenges such as limited number of listed stocks with sufficient trading volume, algorithm selection, and the importance of incorporating firm characteristics pose significant hurdles to achieving consistent profitability.

In summary, the research contributes to the literature by examining the application of unsupervised learning techniques in pairs trading on the JSE, the findings underscore the challenges of implementing pairs trading strategies in emerging markets. Future research could explore alternative methodologies or incorporate additional data sources to enhance the performance of pairs trading strategies in the South African market. Moreover, the study reinforces the importance of considering fundamental factors alongside price data in developing effective trading strategies using unsupervised learning algorithms.

6. References

Chahn. (2018). Majority of hedge fund pros use AI/machine learning in investment strategies. *Barclayhedge survey*.

Do, B., & Faff, R. (2010). Does Simple Pairs Trading Still Work? *Financial Analysts Journal, 66*(4), 83-95.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226-331.

Gatev, E., Goetzmann, W., & Rouwenhorst, G. (2006). Pairs Trading: Performance of a Relative-Value Arbitrage Rule. *The Review of Financial Studies, 19*(3), 787-827.

Goetzmann, W. N., Gatev, E., & Rouwenhorst, G. K. (1999). Pairs Trading: Performance of a Relative Value Arbitrage Rule. *National Bureau of Economic Research*.

Green, J., Hand, J. R., & Zhang, F. (2016). The Characteristics that Provide Independent Information about Average U.S. Monthly Stock Returns. *Available at: https://ssrn.com/abstract=2262374*. doi:10.2139/ssrn.2262374

Han, C., He, Z., & Toh, A. (2021). Pairs Trading via Unsupervised Learning. *Available at: https://ssrn.com/abstract=3835692*. doi:10.2139/ssrn.3835692

Hautamaki, V., Cherednichenko, S., Karkkainen, I., Kinnunen, T., & Franti, P. (2005). Improving k-means by outlier removal. *Scandinavian Conference on Image Analysis*, 978-987.

Johnson, S. (1967). Hierarchical clustering schemes. *Psychometrika, 32*, 241-254.

Liew, R. Q., & Wu, Y. (2013). Pairs trading: A copula approach. *Journal of Derivatives and Hedge Funds 19*, 12-30.

MacQueen, J. e. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 281-297.

Pearson, F. K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2*(11), 559–572. doi:10.1080/14786440109462720

Rad, H., Low, R., & Faff, R. W. (2016). The Profitability of Pairs Trading Strategies: Distance, Cointegration, and Copula Methods. *Quantitative Finance 16.10*, 1541-1558.

Sharpe, W. F. (1966). Mutual Fund Performance. *The Journal of Business, 39*(2), 119-138.

Stander, Y., Marais, D., & Botha, I. (2013). Trading Strategies with Copulas. *Journal of Economic and Financial Services April 2013 (6)1*, 83-108.

Wittek, P. (2014). *Quantum Machine Learning.*

Xie, W., Liew, R. Q., Wu, Y., & Zou, X. (2016). Pairs Trading with Copulas. *The Journal of Trading, 11*(3), 41-52.

Appendices

Appendix 1: Firm Feature Variables

Variable
Net Debt to EBIT (Earnings Before Interest and Taxes
Cash from operations to sales
Book Value per Weighted Diluted Share
Price Earnings Ratio
Basic Earnings Per Share
Quick Ratio
Book Value Per Share
Current Ratio
Sum of Rand Monthly Trading Volume
Price to Book ratio
Current market capitalisation
Working capital accruals
Asset growth
Cash holdings
Cash flow to price ratio
Change in inventory
Growth in long-term debt
Return on assets
Sales to cash
Sales growth

Appendix 2: Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import math

from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn import metrics
from datetime import datetime, timedelta
from sklearn.neighbors import NearestNeighbors


%matplotlib inline
```

Data

```
#tradingcost 25bps

tc = 25/10000

# import fundamental data
var=pd.read_excel('Fields Per Variable.xlsx',index_col=0, sheet_name=None)
for v in list(var):
    var[v].index = var[v].index - timedelta(days=1)

#import data

#import price data
pri=pd.read_excel('JSE Stocks 2010 - 2023 Monthly Prices.xlsx',index_col=0)
pri.index = pri.index - timedelta(days= 1)

#only keep stocks with average monthly volume > R10bn
var['RVol'] = var['Vol']*pri
vol = var['RVol'].drop(var['RVol'].columns[var['RVol'].mean()<=100e8], axis =
 'columns')

pri = pri[list(vol.columns)]

#calculate log returns
ret=(pri/pri.shift()-1).dropna(how='all')

#drop na columns
#ret1=ret.dropna(axis=1)0
#ret1

#caluclate momentum factors
mom={}
norm_mom={}
mom[1]=ret

for i in range(2,25):
    mom[i]=(mom[i-1]+ret.shift(i-1)).dropna(how='all')
    mom[i]=mom[i].loc['2011-12-31':]
    norm_mom[i] = (mom[i]-mom[i].mean())/mom[i].std()

mom[1]=mom[1].loc['2011-12-31':]
norm_mom[1] = (mom[1]-mom[1].mean())/mom[1].std()

#import benchmark data

bnc = pd.read_excel('Benchmark Data 2010-2023.xlsx', index_col=0)
bnc = bnc['2011-12-30':]
bnc.index = mom[1].index
bnc_ret = (bnc/bnc.shift()-1).dropna(how='all')

#create new variables
```

```
var['acc']=(var['Sales']-var['Op CFs'])/var['AVG ASSETS']
var['agr']=np.log(var['TOT_ASSETS']/var['TOT_ASSETS'].shift(periods=12))
var['cash']=var['BS_CASH_NEAR_CASH_ITEM']/var['TOT_ASSETS']
var['cfp']=var['Op CFs']/var['CUR_MKT_CAP']
var['chinv']=(var['BS_INVENTORIES']-var['BS_INVENTORIES'].shift(periods=12))/
var['AVG ASSETS']
var['lgr']=np.log(var['BS_TOT_LIAB2']/var['BS_TOT_LIAB2'].shift(periods=12))
var['roa']=var['Sales']/var['AVG ASSETS']
var['salescash']=var['Sales']/var['BS_CASH_NEAR_CASH_ITEM']
var['sgr']=np.log(var['Sales']/var['Sales'].shift(periods=12))

#See all Vars

#print(var.keys())

#keep only vars we want

var_sheet_names = ['NET_DEBT_TO_EBIT', 'CFO_TO_SALES', 'BV_PER_WEIGHTED_DILUT
ED_SHARE', 'PE_RATIO', 'IS_EPS',
                   'QUICK_RATIO', 'BOOK_VAL_PER_SH', 'CUR_RATIO',  'PX_TO_BOO
K_RATIO', 'CUR_MKT_CAP',
                   'RVol', 'acc', 'agr', 'cash', 'cfp', 'chinv', 'lgr', 'roa
', 'salescash', 'sgr']

#keep only stocks in returns data

for x in var_sheet_names:
    var[x]=var[x][list(ret.columns)]
    var[x]=var[x].loc['2011-12-31':]
    var[x]=var[x].fillna(method='ffill')
    var[x]=var[x].fillna(var[x].mean())


#normalise variables

norm_var = {}

for x in var_sheet_names:
    norm_var[x]=(var[x]-var[x].mean())/var[x].std()

#create df of all momentums per stock

#list of momentums we want to keep
moms = ['mom1', 'mom3','mom6','mom12','mom24']

momset_date = {}
norm_momset_date = {}

for i in range(0,np.shape(mom[1])[0]):
    momset = {}
```

```
    norm_momset = {}

    for x in range(1,len(mom)+1):

        momset[x] = mom[x].iloc[[i]].transpose()
        momset[x].columns = [f"mom{x}"]

        norm_momset[x] = norm_mom[x].iloc[[i]].transpose()
        norm_momset[x].columns = [f"mom{x}"]

    momset_date[i] = momset[1]
    norm_momset_date[i] = norm_momset[1]

    for x in range(2,len(mom)+1):
        momset_date[i] = momset_date[i].join(momset[x])
        norm_momset_date[i] = norm_momset_date[i].join(norm_momset[x])

    date = mom[1].axes[0]
    date = date[i].date()

    momset_date[i] = momset_date[i].rename_axis(date,axis=0)
    momset_date[i] = momset_date[i].rename_axis('Ticker',axis=1)
    momset_date[i] = momset_date[i][moms].dropna(how='all')

    norm_momset_date[i] = norm_momset_date[i].rename_axis(date,axis=0)
    norm_momset_date[i] = norm_momset_date[i].rename_axis('Ticker',axis=1)
    norm_momset_date[i] = norm_momset_date[i][moms].dropna(how='all')

#create df of all firm characteristics per stock

firmset_date = {}

for i in range(0,np.shape(mom[1])[0]):
    firmset = {}

    for x in var_sheet_names:

        firmset[x] = norm_var[x].iloc[[i]].transpose()
        firmset[x].columns = [f"{x}"]

    firmset_date[i] = momset[1]

    for x in var_sheet_names:
        firmset_date[i] = firmset_date[i].join(firmset[x])

    date = norm_var['NET_DEBT_TO_EBIT'].axes[0]  #first var name
    date = date[i].date()

    firmset_date[i] = firmset_date[i].rename_axis(date,axis=0)
```

```
    firmset_date[i] = firmset_date[i].rename_axis('Ticker',axis=1)
    firmset_date[i] = firmset_date[i].drop(columns=['mom1'])

#create datasets with momentume and firm characteristics

allset = {}

for i in range(0,np.shape(mom[1])[0]):
    allset[i]=norm_momset_date[i].join(firmset_date[i]).dropna()
```

PCA

```
#pca on returns

num_clusters = [2, 5, 10]

pca = {}
comp={}
n_comp={}
pca_t={}

for p in range(0,np.shape(mom[1])[0]):

    #fit pca
    pca[p] = PCA().fit(allset[p])


    #determine no. of components that explain at least 99% of variation

    comp[p] = np.cumsum(pca[p].explained_variance_ratio_)
    comp[p] = pd.DataFrame(comp[p], columns = ['n_comp'])
    n_comp[p] = max(comp[p].index[comp[p]['n_comp']<=0.99].tolist())+1

    #run pca with n_comp and transform
    pca_t[p] = PCA(n_components = n_comp[p]).fit(allset[p]).transform(allset
[p])
    pca_t[p] = pd.DataFrame(pca_t[p])

plt.figure()
plt.plot(np.cumsum(pca[100].explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Varaince %')
plt.title('Explained Variance')
```

Text(0.5, 1.0, 'Explained Variance')

png

KMeans Clustering

```
#apply kmeans clustering

num_kmclusters = [2, 5, 10]

results = {}

for p in range(0,np.shape(mom[1])[0]):

    results[p] = {}

    for k in num_clusters:
        results[p][k] = {}

        kmeans = KMeans(n_clusters = k, random_state = 0).fit(pca_t[p])

        sil = metrics.silhouette_score(pca_t[p], kmeans.labels_, metric='eucl
idean')

        results[p][k]['model'] = kmeans

#Create Pairs

km = {}
kmclus={}

for p in range(0,np.shape(mom[1])[0]):

    km[p] = {}
    kmclus[p]={}

    for k in num_kmclusters:
        km[p][k] = pd.DataFrame(results[p][k]['model'].labels_)
        km[p][k].index = allset[p].index
        km[p][k].columns = ['kmcluster']
        km[p][k] = km[p][k].join(momset_date[p])
        km[p][k]['Code']=allset[p].index
        km[p][k] = km[p][k][['kmcluster','mom1','Code']].sort_values(by=['mom
1'])
        km[p][k].index = pca_t[p].index

        kmclus[p][k]={}

        for n in range(1, k+1):
            kmclus[p][k][n]=km[p][k].loc[km[p][k]['kmcluster']==n].sort_value
s(by=['mom1'])


            if len(kmclus[p][k][n]) <=1:
                pass
```

```
            elif len(kmclus[p][k][n]) % 2 != 0:
                kmclus[p][k][n] = kmclus[p][k][n].drop(kmclus[p][k][n].index
[(len(kmclus[p][k][n])+1)//2])

                kmclus[p][k][n]=np.array_split(kmclus[p][k][n],2)
                kmclus[p][k][n] = pd.DataFrame(np.concatenate((kmclus[p][k]
[n][0],kmclus[p][k][n][1].sort_values(by=['mom1'], ascending = False)),axis=
1))
                kmclus[p][k][n]['momdiff'] = kmclus[p][k][n][1]-kmclus[p][k]
[n][4]

            else:
                kmclus[p][k][n]=np.array_split(kmclus[p][k][n],2)
                kmclus[p][k][n] = pd.DataFrame(np.concatenate((kmclus[p][k]
[n][0],kmclus[p][k][n][1].sort_values(by=['mom1'], ascending = False)),axis=
1))
                kmclus[p][k][n]['momdiff'] = kmclus[p][k][n][1]-kmclus[p][k]
[n][4]

for p in range(0,np.shape(mom[1])[0]):

    for k in num_kmclusters:

        for n in range(1, k+1):

            std = pd.concat(kmclus[p][k]).dropna(subset=[0])['momdiff'].std()

            if len(kmclus[p][k][n]) <=1:
                pass

            else:
                kmclus[p][k][n]['Trade'] = np.where(kmclus[p][k][n]['momdiff
']<-std, True, False)

#create equal weighted portfolio weights

w={}
wt={}
out={}

for p in range(0,np.shape(mom[1])[0]):
    w[p]=[]
    wt[p]=[]
    out[p] = {}

    for k in num_kmclusters:

        out[p][k] = pd.concat(kmclus[p][k])

        w[p].append(out[p][k]['Trade'].value_counts()[True])
```

```
        wt[p] = 1/(sum(w[p]))

#create portfolios

short = {}
long = {}
clus_ret = {}
port_ret = {}
short_ret = {}
long_ret = {}
short_port = {}
long_port = {}

for p in range(0,np.shape(mom[1])[0]-1):
    short[p] = {}
    long[p] = {}
    clus_ret[p] = {}
    port_ret[p] = {}
    short_ret[p] = {}
    long_ret[p] = {}
    short_port[p] = {}
    long_port[p] = {}

    for k in num_clusters:
        short[p][k] = {}
        long[p][k] = {}
        clus_ret[p][k]=[]
        short_ret[p][k]=[]
        long_ret[p][k]=[]

        for n in range(1, k+1):
            if len(kmclus[p][k][n]) <=1:
                pass
            else:
                short[p][k][n] = kmclus[p][k][n].loc[(kmclus[p][k][n]['Trade
']==True),[5]]
                short[p][k][n] = short[p][k][n].join(momset_date[p+1], on=5)
[[5,'mom1']]
                short[p][k][n]['wret'] = short[p][k][n]['mom1']*-wt[p]

                long[p][k][n] = kmclus[p][k][n].loc[(kmclus[p][k][n]['Trade']
==True),[2]]
                long[p][k][n] = long[p][k][n].join(momset_date[p+1], on=2)
[[2,'mom1']]
                long[p][k][n]['wret'] = long[p][k][n]['mom1']*wt[p]

                clus_ret[p][k].append(sum(short[p][k][n]['wret']) + sum(long
[p][k][n]['wret']))
                short_ret[p][k].append(sum(short[p][k][n]['wret']))
```

```
                long_ret[p][k].append(sum(long[p][k][n]['wret']))

        port_ret[p][k] = sum(clus_ret[p][k])
        short_port[p][k] = sum(short_ret[p][k])
        long_port[p][k] = sum(long_ret[p][k])

#create df with all returns

ls_ret_df = pd.DataFrame(port_ret).transpose()
ls_ret_df.index = mom[1].iloc[1:].index
ls_ret_df.columns = ['L-S KMeans (2)', 'L-S KMeans (5)', 'L-S KMeans (10)']

long_ret_df = pd.DataFrame(long_port).transpose()
long_ret_df.index = mom[1].iloc[1:].index
long_ret_df.columns = ['L KMeans (2)', 'L KMeans (5)', 'L KMeans (10)']

short_ret_df = pd.DataFrame(short_port).transpose()
short_ret_df.index = mom[1].iloc[1:].index
short_ret_df.columns = ['S KMeans (2)', 'S KMeans (5)', 'S KMeans (10)']

kmean_ret_df = pd.concat([ls_ret_df, long_ret_df, short_ret_df], axis=1)
```

DBSCAN

```
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

#apply kmeans clustering

eps_type = 'mean' #set as 'mean' for average of minpts distance or 'opt' for
optimal eps value

alpha = 1 #alpha percentile for mean eps type

results = {}
minpts = {}
eps = {}

for p in range(0,np.shape(mom[1])[0]):

    results[p] = {}
    minpts[p] = np.floor(np.log(pca_t[p].shape[0]*pca_t[p].shape[1])) #minpts
 = log of number of datapoints

    #calculate distance metric, distance to nearest neighbour
    neigh = NearestNeighbors(n_neighbors=2, metric = 'l1') #calculate distanc
e metric
    nbrs = neigh.fit(pca_t[p])
```

```
    distances, indices = nbrs.kneighbors(pca_t[p])
    distances = np.sort(distances, axis=0)
    distances = distances[:,1]
    distances_df = pd.DataFrame(distances)
    distances_df['diff'] = distances_df - distances_df.shift(1)

    if eps_type == 'opt':
        eps[p] = distances_df[0].iloc[distances_df.idxmax()['diff']] #eps set
 as the point where max change in distance metric

    else:
        eps[p] = distances_df[0].mean()*alpha

    results[p] = DBSCAN(eps = eps[p], min_samples = minpts[p], metric='l1').f
it(pca_t[p])

    pca_t[p]['DBSCAN Label']=results[p].labels_

# create pairs

temp = {}
clus={}

w=[]

for p in range(0,np.shape(mom[1])[0]):

    temp[p] = pd.DataFrame(pca_t[p]['DBSCAN Label']) #create temp df with DBS
CAN Labels
    temp[p].index = allset[p].index #set index to equity codes
    temp[p] = temp[p].join(momset_date[p]) #join on equity code to get moment
ums
    temp[p]['Code'] = allset[p].index #create equity code column
    temp[p] = temp[p][['DBSCAN Label','mom1','Code']].sort_values(by=['mom1
']) #keep only these cols and sort by mom1
    temp[p] = temp[p].loc[temp[p]['DBSCAN Label']!=-1] #drop outliers, DBSCAN
 Label = -1
    temp[p] = temp[p].reset_index(drop=True) #reset index of df

    clus[p]= {}
    clusters = len(temp[p]['DBSCAN Label'].unique()) #number of clusters form
ed


    for n in range(0,clusters):
        clus[p][n] = temp[p].loc[temp[p]['DBSCAN Label'] == n].sort_values(by
=['mom1']) #keep only equities in cluster n

        if len(clus[p][n])<=1:
            pass #if 1 or fewer items in cluster then keep blank
```

```
        elif len(clus[p][n]) % 2 != 0: #if odd number in cluster drop middle
value
            clus[p][n] = clus[p][n] = clus[p][n].drop(clus[p][n].index[(len(c
lus[p][n])+1)//2])

            clus[p][n]=np.array_split(clus[p][n],2)
            clus[p][n] = pd.DataFrame(np.concatenate((clus[p][n][0],clus[p]
[n][1].sort_values(by=['mom1'], ascending = False)),axis=1))
            clus[p][n]['momdiff'] = clus[p][n][1]-clus[p][n][4]


        else:
            clus[p][n]=np.array_split(clus[p][n],2)
            clus[p][n] = pd.DataFrame(np.concatenate((clus[p][n][0],clus[p]
[n][1].sort_values(by=['mom1'], ascending = False)),axis=1))
            clus[p][n]['momdiff'] = clus[p][n][1]-clus[p][n][4]


        std = pd.concat(clus[p])['momdiff'].std()

        clus[p][n]['Trade'] = np.where(clus[p][n]['momdiff']<-std,
                                  True, False)

    if len(clus[p]) == 0:
        pass

    elif len(clus[p]) == 1:

        out[p] = clus[p][0]
        w.append(1/out[p]['Trade'].value_counts()[True])
    else:
        out[p] = pd.concat(clus[p])

        w.append(1/out[p]['Trade'].value_counts()[True])

short = {}
long = {}
clus_ret = {}
port_ret = {}
short_port = {}
long_port = {}
short_ret = {}
long_ret = {}

for p in range(0,np.shape(mom[1])[0]-1):
    short[p] = {}
    long[p] = {}
    clus_ret[p] = []
    short_ret[p] = []
```

```
    long_ret[p] = []

    clusters = len(temp[p]['DBSCAN Label'].unique()) #number of clusters form
ed

    for n in range(0,clusters):
        if len(clus[p][n])<=1:
            pass
        else:
            short[p][n] = clus[p][n].loc[(clus[p][n]['Trade']==True),[5]]
            short[p][n] = short[p][n].join(momset_date[p+1], on=5)[[5,'mom1
']]
            short[p][n]['wret'] = short[p][n]['mom1']*-w[p]

            long[p][n] = clus[p][n].loc[(clus[p][n]['Trade']==True),[2]]
            long[p][n] = long[p][n].join(momset_date[p+1], on=2)[[2,'mom1']]
            long[p][n]['wret'] = long[p][n]['mom1']*w[p]

            clus_ret[p].append(sum(short[p][n]['wret']) + sum(long[p][n]['wre
t']))
            short_ret[p].append(sum(short[p][n]['wret']))
            long_ret[p].append(sum(long[p][n]['wret']))
    port_ret[p] = sum(clus_ret[p])
    short_port[p] = sum(short_ret[p])
    long_port[p] = sum(long_ret[p])

dbls_ret_df = pd.DataFrame(port_ret, index=[0]).transpose()
dbls_ret_df.index = mom[1].iloc[1:].index
dbls_ret_df.columns = ['LS DBSCAN']

dblong_ret_df = pd.DataFrame(long_port, index=[0]).transpose()
dblong_ret_df.index = mom[1].iloc[1:].index
dblong_ret_df.columns = ['L DBSCAN']

dbshort_ret_df = pd.DataFrame(short_port, index=[0]).transpose()
dbshort_ret_df.index = mom[1].iloc[1:].index
dbshort_ret_df.columns = ['S DBSCAN']

dbscan_ret_df = pd.concat([dbls_ret_df, dblong_ret_df, dbshort_ret_df], axis=
1)
```

Agglomerative Clustering

```
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

#apply agglomerative clustering
```

```
eps_type = 'mean' #set as 'mean' for average of minpts distance or 'opt' for
optimal eps value

alpha = 0.5 #alpha percentile for mean eps type

results = {}
eps = {}

for p in range(0,np.shape(mom[1])[0]):

    #pca_t[p] = pca_t[p].drop(columns=['DBSCAN Label'])

    results[p] = {}

    #calculate distance metric, distance to nearest neighbour
    neigh = NearestNeighbors(n_neighbors=2, metric = 'l1') #calculate distanc
e metric
    nbrs = neigh.fit(pca_t[p])
    distances, indices = nbrs.kneighbors(pca_t[p])
    distances = np.sort(distances, axis=0)
    distances = distances[:,1]
    distances_df = pd.DataFrame(distances)
    distances_df['diff'] = distances_df - distances_df.shift(1)

    if eps_type == 'opt':
        eps[p] = distances_df[0].iloc[distances_df.idxmax()['diff']] #eps set
 as the point where max change in distance metric

    else:
        eps[p] = distances_df[0].mean()*alpha

    results[p] = AgglomerativeClustering(n_clusters = None, linkage = 'averag
e', distance_threshold = eps[p]).fit(pca_t[p])

    pca_t[p]['Agg Label']=results[p].labels_

# create pairs

temp = {}
clus={}
out = {}

w=[]

for p in range(0,np.shape(mom[1])[0]):

    temp[p] = pd.DataFrame(pca_t[p]['Agg Label']) #create temp df with DBSCAN
 Labels
    temp[p].index = allset[p].index #set index to equity codes
    temp[p] = temp[p].join(momset_date[p]) #join on equity code to get moment
```

```
ums
    temp[p]['Code'] = allset[p].index #create equity code column
    temp[p] = temp[p][['Agg Label','mom1','Code']].sort_values(by=['mom1']) #
keep only these cols and sort by mom1
    temp[p] = temp[p].loc[temp[p]['Agg Label']!=-1] #drop outliers, DBSCAN La
bel = -1
    temp[p] = temp[p].groupby('Agg Label').filter(lambda x:len(x)>1) #keep cl
usters with 2 or more stocks
    temp[p] = temp[p].reset_index(drop=True) #reset index of df

    clus[p]= {}
    clusters = temp[p]['Agg Label'].unique().tolist() #number of clusters for
med

    for n in clusters:
        clus[p][n] = temp[p].loc[temp[p]['Agg Label'] == n].sort_values(by=['
mom1']) #keep only equities in cluster n

        if len(clus[p][n])<=1:
            pass

        elif len(clus[p][n]) % 2 != 0: #if odd number in cluster drop middle
value
            clus[p][n] = clus[p][n] = clus[p][n].drop(clus[p][n].index[(len(c
lus[p][n])+1)//2])

            clus[p][n]=np.array_split(clus[p][n],2)
            clus[p][n] = pd.DataFrame(np.concatenate((clus[p][n][0],clus[p]
[n][1].sort_values(by=['mom1'], ascending = False)),axis=1))
            clus[p][n]['momdiff'] = clus[p][n][1]-clus[p][n][4]

        else:
            clus[p][n]=np.array_split(clus[p][n],2)
            clus[p][n] = pd.DataFrame(np.concatenate((clus[p][n][0],clus[p]
[n][1].sort_values(by=['mom1'], ascending = False)),axis=1))
            clus[p][n]['momdiff'] = clus[p][n][1]-clus[p][n][4]

        std = pd.concat(clus[p])['momdiff'].std()

        clus[p][n]['Trade'] = np.where(clus[p][n]['momdiff']<-std,
                                       True, False)

    out[p] = pd.concat(clus[p])

    w.append(1/out[p]['Trade'].value_counts()[True])

short = {}
long = {}
clus_ret = {}
port_ret = {}
```

```
short_port = {}
long_port = {}
short_ret = {}
long_ret = {}

for p in range(0,np.shape(mom[1])[0]-1):
    short[p] = {}
    long[p] = {}
    clus_ret[p] = []
    short_ret[p] = []
    long_ret[p] = []

    clusters = temp[p]['Agg Label'].unique().tolist() #number of clusters for
med

    for n in clusters:
        if len(clus[p][n])<=1:
            pass
        else:
            short[p][n] = clus[p][n].loc[(clus[p][n]['Trade']==True),[5]]
            short[p][n] = short[p][n].join(momset_date[p+1], on=5)[[5,'mom1
']]
            short[p][n]['wret'] = short[p][n]['mom1']*-w[p]

            long[p][n] = clus[p][n].loc[(clus[p][n]['Trade']==True),[2]]
            long[p][n] = long[p][n].join(momset_date[p+1], on=2)[[2,'mom1']]
            long[p][n]['wret'] = long[p][n]['mom1']*w[p]

            clus_ret[p].append(sum(short[p][n]['wret']) + sum(long[p][n]['wre
t']))
            short_ret[p].append(sum(short[p][n]['wret']))
            long_ret[p].append(sum(long[p][n]['wret']))
    port_ret[p] = sum(clus_ret[p])
    short_port[p] = sum(short_ret[p])
    long_port[p] = sum(long_ret[p])

aggls_ret_df = pd.DataFrame(port_ret, index=[0]).transpose()
aggls_ret_df.index = mom[1].iloc[1:].index
aggls_ret_df.columns = ['LS Agglomerative']

agglong_ret_df = pd.DataFrame(long_port, index=[0]).transpose()
agglong_ret_df.index = mom[1].iloc[1:].index
agglong_ret_df.columns = ['L Agglomerative']

aggshort_ret_df = pd.DataFrame(short_port, index=[0]).transpose()
aggshort_ret_df.index = mom[1].iloc[1:].index
aggshort_ret_df.columns = ['S Agglomerative']

agg_ret_df = pd.concat([aggls_ret_df, agglong_ret_df, aggshort_ret_df], axis=
1)
```

Results

```
port_ret_df = pd.concat([kmean_ret_df, dbscan_ret_df, agg_ret_df, bnc_ret], a
xis=1)

ls = ['L-S KMeans (2)', 'L-S KMeans (5)', 'L-S KMeans (10)','LS DBSCAN', 'LS
Agglomerative',
      'ALSI', 'TOP40']

long = ['L KMeans (2)', 'L KMeans (5)', 'L KMeans (10)', 'L DBSCAN', 'L Agglo
merative',
        'ALSI', 'TOP40']

short = ['S KMeans (2)', 'S KMeans (5)', 'S KMeans (10)', 'S DBSCAN', 'S Aggl
omerative',
        'ALSI', 'TOP40']

#Overall Annual Return

ann_ret_all = ((port_ret_df).cumprod().iloc[[len(port_ret_df)-1]])**(1/(len(p
ort_ret_df)/12))-1
sharpe_all = pd.DataFrame(port_ret_df.mean()/port_ret_df.std()).transpose()

ann_summ_all = pd.concat([sharpe_all, ann_ret_all])
ann_summ_all.index = ['Sharpe', 'Annualised Return']

#Annual Return, Sharpe Per Year

years = port_ret_df.index.year.unique()

ann_ret={}
temp={}

for y in range(0,len(years)):
    temp[y] = port_ret_df[(port_ret_df.index.year==years[y])]
    ann_ret[y] = ((temp[y]).cumprod().iloc[[len(temp[y])-1]])**(1/(len(temp
[y])/12))-1

ann_ret_df = pd.concat(ann_ret).droplevel(0)
```