Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Benefitting from Bitcoin Beliefs: can Twitter Sentiment obtained using k-means clustering predict Bitcoin Returns?

Jeroen Frasa (621094)

Supervisor:	Riley Badenbroek
Second assessor:	Wilco van den Heuvel
Date final version:	1st July 2024

Abstract

Bitcoin is increasingly being considered a legitimate asset due to the launch of ETFs in the United States and its role as a diversifying asset. Moreover, the predictive power of Twitter sentiment in financial markets is increasingly recognized in academics. Predicting bitcoin returns is relevant to investors, who are trying to mitigate bitcoin's volatility to maximize returns. This paper studies the predictive power of bitcoin Twitter sentiment for bitcoin returns. Our approach involves using tweets about bitcoin, k-means clustering with feature weighting, a rule-based sentiment algorithm and Granger causality tests. We find that k-means clustering with feature weighting is able to cluster tweets effectively based on sentiment, but these sentiment scores do not have predictive power for bitcoin returns. However, we also show that sentiment scores obtained without using k-means clustering do not have predictive power for bitcoin anymore, showing that the bitcoin market has become more efficient. This paper also contains a replication of the results of Modha & Spangler (2003), who propose the convex k-means clustering algorithm with feature weighting. We find different results because not enough details are provided to exactly replicate the data preprocessing. However, we find the same patterns as Modha & Spangler (2003), making us state the same conclusion: optimal feature weighting leads to nearly the best clustering precision.

1 Introduction

Cryptocurrencies have sparked the interest of many investors due to their unprecedented returns and unique technical features. The first cryptocurrency was bitcoin (Nakamoto, 2008), introducing a new technology called *blockchain*. This is a digital, distributed ledger of transactions kept by all participants in the network. Transactions are peer to peer, meaning there is no bank or other middle man. After its launch in 2009, bitcoin remained relatively unknown until the end of 2017, when it hit a market capitalization of 249 million U.S. dollars. This was preceded by a price increase of 1312% in 2017 (CoinGecko, 2024). Although bitcoin's price dropped sharply afterward, it continued to develop and gain relevance: solutions were proposed to make bitcoin more scalable (Vujičić et al., 2018), and bitcoin *Exchange Traded Funds* (ETFs) became tradable in the United States. These ETFs attracted record-breaking inflows in their first weeks (Mazur & Polyzos, 2024). This demand for bitcoin can be explained, among other reasons, by the increased use of bitcoin in retirement savings strategies (Olabanji et al., 2024), and by its role as a diversifying asset. Introducing bitcoin in a diversified portfolio can decrease the portfolio's risk, improving its overall performance (Guesmi et al., 2019; Akhtaruzzaman et al., 2020). This institutional demand shows that bitcoin is gradually being considered as a legitimate asset, slowly leaving behind its old reputation of being merely a vehicle for speculation.

Still, bitcoin remains a volatile asset. It is therefore lucrative for investors to see if there is any predictability to bitcoin's price movements, as it could help to maximize returns. In this paper, we investigate whether sentiment helps to predict bitcoin returns. The research question is: does k-means clustering aid in providing sentiment scores with predictive power for bitcoin returns?

With the contemporary abundance of information on news sites and social media, investors are increasingly informing their decisions with *sentiment analysis*. It is defined as quantifying

the underlying emotions, opinions and subjectivity of a body of text. Sentiment analysis is a relatively new field of study: research on opinions and sentiment mainly started around the year 2000. It quickly became an active research area due to its many applications (Liu, 2022). Often, microblogging website *Twitter* (also known as X) is used as a data source for sentiment analysis. Twitter is a platform where people can share anything they want in short, public messages called *tweets*. Twitter sentiment quickly became a popular area of research due to the large number of available tweets and high sentiment analysis accuracy, as tweets are usually opinionated and straight to the point (Liu, 2022). Applications of Twitter sentiment analysis include predicting election results (Tumasjan et al., 2010), predicting movie ticket sales revenue (Asur & Huberman, 2010), and predicting the stock market. Bollen et al. (2011) showed that Twitter mood states significantly improve predictions for the Dow Jones Industrial Average index.

Similar research has been performed regarding bitcoin. Twitter is the primary place where cryptocurrency related news and opinions are shared (Kraaijeveld & De Smedt, 2020), making it again an interesting source of data for sentiment analysis. Kraaijeveld & De Smedt (2020) find that Twitter sentiment has predictive power for bitcoin returns. However, this study only focuses on a 2-month period in 2018, which may not be representative for the current state of the cryptocurrency market. Another relevant study by Anamika et al. (2023) uses a survey-based approach to gauge sentiment, also showing that bitcoin sentiment has significant predictive power for bitcoin prices. The downside of this approach is that survey data is not available in real time (only on a weekly basis) and often hard to access for individual investors.

Our contribution to this literature is threefold. The first is to use bitcoin tweets from a more recent time frame for sentiment analysis. In total, 117062 tweets are used ranging from January 1st, 2022, to December 31st, 2022. These tweets all contain either '#bitcoin' or '#btc'. The second contribution to the literature is the use of feature weighting in k-means clustering before calculating sentiment scores. Lastly, we apply the convex k-means clustering algorithm in a different area.

Ever since k-means clustering was introduced by MacQueen (1967), it has been a fundamental technique in statistics and machine learning. It partitions observations into k groups called *clusters*. These clusters are based on similarity, where observations within a cluster are more similar to each other than observations across clusters. Over the years, many improvements to the original algorithm have been proposed. One of these is *feature weighting*, first introduced by DeSarbo et al. (1984). Their idea was that different features may have different relevance, so being able to give different weights to features may improve the clustering performance. Since then, a lot of further work has been done on feature weighting, as summarized by De Amorim (2016). One of the more important works in this area came from Modha & Spangler (2003), who introduced the convex k-means clustering algorithm. Their ideas are to assign different distortion measures to each feature vector, to combine these measures in a convex way by assigning different weights to each, then determining the optimal feature weighting. They apply their algorithm to text data, showing consistently better results when non-uniform weightings are used versus the uniform weighting. We attempt replicating the results of Modha & Spangler (2003) concerning text data in this paper.

Another application of clustering involving text is sentiment analysis. Ahuja & Dubey (2017) show that clustering can efficiently and quickly distinguish tweets based on sentiment, finding what the weekly sentiment is. This result encourages us to research the usefulness of using clustering to obtain bitcoin sentiment. To our best knowledge, sentiment analysis on bitcoin tweets using k-means clustering has not been researched yet. The work of Barradas et al. (2022) comes close, but is focused on the volume of bitcoin tweets rather than the actual sentiment.

In our study, we use the convex k-means clustering algorithm of Modha & Spangler (2003) to cluster tweets after having preprocessed them to remove noise. The tweets are grouped by week, and we perform a clustering for every week. For each cluster, we obtain the sentiment score using the *Valence Aware Dictionary and Sentiment Reasoner* (VADER) algorithm, proposed by Hutto & Gilbert (2014). It is a rule-based algorithm for calculating sentiment scores, working especially well when applied to social media posts because it also extracts sentiment from slang and emoticons. With the sentiment scores obtained for each cluster, we test with *Smirnov tests* if the clusters have significantly different sentiment scores. Then, *Vector Autoregressive models* (VAR models) are constructed. VAR models are natural tools for finding causal relationships, as they explain the independent variable by its own lags, a set of regressors, and lags of the regressors (Lütkepohl, 2013). Bitcoin returns are used as the independent variable, the sentiment scores as regressors. Finally, Granger causality tests are applied to test predictive power of the sentiment scores for bitcoin returns.

The results of Modha & Spangler (2003) can not be perfectly replicated, because details are missing about the way they preprocessed their data. Still, we managed to come reasonably close to some numbers they presented on the data, like the number of documents containing at least 1 word, 2-word phrase and 3-word phrase. The discrepancy in data leads to different clustering performance numbers: we find lower micro-p values, and higher objective function values. However, we still find a clear negative correlation between micro-p values and objective function values. Most importantly, the optimal feature weighting nearly attained the highest micro-p value. We can thus state the same conclusion as Modha & Spangler (2003): optimal feature weighting achieves nearly the best precision.

We show by means of Smirnov tests that using feature weighting in k-means clustering effectively clusters tweets based on sentiment. However, the sentiment scores obtained by the clustering do not have predictive power for bitcoin returns, as judged by the VAR models and Granger causality tests. We also show that sentiment scores obtained without using clustering do not have predictive power for bitcoin returns anymore. This suggests that the bitcoin market has become more efficient, and we advise investors to turn to other methods in an attempt to predict bitcoin returns.

This paper is organized as follows. First, the general clustering methodology is in Section 2. Then we present a discussion on the replication of the study of Modha & Spangler (2003): the data used for that and its preprocessing is in Section 3. Section 4 contains the methodology concerning the replication, and Section 5 the corresponding results. We then move on to the study on Twitter sentiment regarding bitcoin. A discussion of the data and preprocessing is in Section 6. The methodology of the extension can be found in Section 7, with Section 8 containing the extension results. The conclusion is given in Section 9.

2 Clustering algorithm

The k-means clustering algorithm described in this section is consistent with that of Modha & Spangler (2003). Their idea is to combine distortion measures on different feature spaces in a convex way and assign a weight to each distortion measure, leading to what they call the k-means clustering algorithm.

2.1 Data model and distortion measure

Each of the n data objects is represented as a tuple of m component feature vectors, written as

$$\mathbf{x} = (\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_m), \tag{1}$$

where feature vector $\mathbf{F}_l, 1 \leq l \leq m$, is a column vector lying in some feature space $\mathcal{F}_l$. We can then say that a data object $\mathbf{x}$ lies in the m-fold product feature space $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_m$.

The goal is to partition the data set $\{\mathbf{x}_i\}_{i=1}^n$ into k clusters $\{\pi_u\}_{u=1}^k$ that are disjoint. To do this, we first need a way to measure distortion between two feature vectors. Let $D_l(\mathbf{F}_l, \tilde{\mathbf{F}}_l)$, be such a measure for $1 \leq l \leq m$, between component feature vectors $\mathbf{F}_l$ and $\tilde{\mathbf{F}}_l$ corresponding to respectively $\mathbf{x}$ and $\tilde{\mathbf{x}}$. Distortion measures of different feature vectors are combined using a weighted distortion measure, which is defined as

$$D^\alpha(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{l=1}^m \alpha_l D_l(\mathbf{F}_l, \tilde{\mathbf{F}}_l), \tag{2}$$

where $\{\alpha_l\}_{l=1}^m$ are the feature weights. They are non-negative and sum to 1. We refer to the vector of feature weights $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ as a feature weighting.

For our distance measure, we stay consistent with the approach of Modha & Spangler (2003) and use the cosine distance

$$D_l(\mathbf{F}_l, \tilde{\mathbf{F}}_l) = 2(1 - \mathbf{F}_l^T \tilde{\mathbf{F}}_l). \tag{3}$$

2.2 Generalized centroids

For a given partitioning $\{\pi_u\}_{u=1}^k$, the corresponding generalized centroid for each partition π_u is defined as

$$\mathbf{c}_u = \arg\min_{\tilde{\mathbf{x}} \in \mathcal{F}} \left(\sum_{\mathbf{x} \in \pi_u} D^\alpha(\mathbf{x}, \tilde{\mathbf{x}}) \right). \tag{4}$$

This centroid $\mathbf{c}_u$ can be split across the feature spaces, written as

$$\mathbf{c}_u = (\mathbf{c}_{(u,1)}, \mathbf{c}_{(u,2)}, \ldots, \mathbf{c}_{(u,m)}). \tag{5}$$

Note that the l-th component $\mathbf{c}_{(u,l)}$ is in $\mathcal{F}_l$. The generalized centroid can be thought of as being the nearest in D^α to all data objects in cluster π_u. Modha & Spangler (2003) show that D^α is component-wise-convex, meaning (4) can be solved by separately solving for each of its components $\mathbf{c}_{(u,l)}$, $1 \leq l \leq m$. For cosine distance, they show that the solution can be written

in closed form as

$$\mathbf{c}_{(u,l)} = \frac{\sum_{\mathbf{x}\in\pi_u} \mathbf{F}_l}{\| \sum_{\mathbf{x}\in\pi_u} \mathbf{F}_l \|}. \tag{6}$$

2.3 The convex k-means algorithm

The within-cluster distortion of each cluster $\pi_u, 1 \leq u \leq k$ is measured as

$$\sum_{\mathbf{x}\in\pi_u} D^{\alpha}(\mathbf{x}, \mathbf{c}_u). \tag{7}$$

The combined distortion of all k clusters in the partitioning is then

$$\sum_{u=1}^{k} \sum_{\mathbf{x}\in\pi_u} D^{\alpha}(\mathbf{x}, \mathbf{c}_u). \tag{8}$$

We want to find k disjoint clusters $\pi_1^{\dagger}, \pi_2^{\dagger}, \ldots, \pi_k^{\dagger}$ such that we minimize

$$\{\pi_u^{\dagger}\}_{u=1}^{k} = \underset{\{\pi_u\}_{u=1}^{k}}{\arg\min} \left(\sum_{u=1}^{k} \sum_{\mathbf{x}\in\pi_u} D^{\alpha}(\mathbf{x}, \mathbf{c}_u) \right), \tag{9}$$

with a fixed feature weighting. The convex k-means clustering algorithm attempts to find the clustering that minimizes (9) through the following steps:

Step 1: Set index $t = 0$ and start with random partitioning $\{\pi_u^{(0)}\}_{u=1}^{k}$, with corresponding generalized centroids $\{\mathbf{c}_u^{(0)}\}_{u=1}^{k}$.

Step 2: For each $\mathbf{x}_i$, $1 \leq i \leq n$, find the generalized centroid closest to $\mathbf{x}_i$. If multiple centroids are equally close, we pick one randomly. Then, compute the new partitioning $\{\pi_u^{(t+1)}\}_{u=1}^{k}$, for $1 \leq u \leq k$, which is created by the old centroids:

$$\pi_u^{(t+1)} = \{\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^{n} : D^{\alpha}(\mathbf{x}, \mathbf{c}_u^{(t)}) \leq D^{\alpha}(\mathbf{x}, \mathbf{c}_v^{(t)}), 1 \leq v \leq k\}. \tag{10}$$

Step 3: The new generalized centroids $\{\mathbf{c}_u^{(t+1)}\}_{u=1}^{k}$ are now computed, using the partitioning in (10) and the generalized centroid in (6), where $\pi_u^{(t+1)}$ is used instead of π_u.

Step 4: If $\pi_u^{(t+1)} = \pi_u^{(t)}$, all data objects $\mathbf{x}_i$, $1 \leq i \leq n$, stay in the same cluster and the algorithm is finished, meaning $\pi_u^{(t+1)} = \pi_u^{\dagger}$ and $\mathbf{c}_u^{(t+1)} = \mathbf{c}_u^{\dagger}$. Otherwise, increment t by 1 and go to Step 2 again.

2.4 Optimal feature weighting

In order to determine which feature weighting achieves the best clustering performance, Modha & Spangler (2003) note that you cannot compare the total distortion of different clusterings, as the different feature weightings lead to different distortion measures D^{α}. Therefore, they introduce the concept of average within-cluster distortion and average between-cluster distortion along the l-th component vector. They are respectively defined as

$$\Gamma_l(\boldsymbol{\alpha}) = \sum_{u=1}^{k} \sum_{\mathbf{x} \in \pi_u^\dagger(\boldsymbol{\alpha})} D_l(\mathbf{F}_l, \mathbf{c}_{(u,l)}^\dagger(\boldsymbol{\alpha})), \tag{11}$$

$$\Lambda_l(\boldsymbol{\alpha}) = \sum_{i=1}^{n} D_l(\mathbf{F}_{(i,l)}, \bar{\mathbf{c}}_l) - \Gamma_l(\boldsymbol{\alpha}), \tag{12}$$

where $\bar{\mathbf{c}}_l$ denotes the generalized centroid for the entire dataset. For $1 \leq l \leq m$, it is calculated as

$$\bar{\mathbf{c}}_l = \arg\min_{\tilde{\mathbf{c}} \in \mathcal{F}_l} \left(\sum_{i=1}^{n} D_l(\mathbf{F}_{(i,l)}, \tilde{\mathbf{c}}) \right). \tag{13}$$

The clustering is best when $\Gamma_l(\boldsymbol{\alpha})$ is minimized, and when $\Lambda_l(\boldsymbol{\alpha})$ is maximized, meaning that the clusters are coherent and separated from each other. To achieve this, we minimize

$$\mathcal{Q}_l(\boldsymbol{\alpha}) = \left(\frac{\Gamma_l(\boldsymbol{\alpha})}{\Lambda_l(\boldsymbol{\alpha})} \right)^{n_l/n}, \tag{14}$$

where n_l is the number of data objects with a corresponding non-zero l-th feature vector. To be able to compare across clusterings, we take a product:

$$\mathcal{Q}(\boldsymbol{\alpha}) = \prod_{l=1}^{m} \mathcal{Q}_l(\boldsymbol{\alpha}). \tag{15}$$

Now we can state the minimization to obtain the optimal feature weighting as

$$\boldsymbol{\alpha}^\dagger = \arg\min_{\boldsymbol{\alpha} \in \Delta} \big(\mathcal{Q}(\boldsymbol{\alpha}) \big), \tag{16}$$

where Δ is the set of all possible feature weightings, defined as

$$\Delta = \left\{ \boldsymbol{\alpha} : \sum_{l=1}^{m} \alpha_l = 1, \alpha_l \geq 0, 1 \leq l \leq m \right\}. \tag{17}$$

By selecting the optimal feature weighting in this way, Modha & Spangler (2003) show that nearly the best precision and recall is achieved.

3 Data replication

The dataset used for the replication of the results of Modha & Spangler (2003) is called Twenty Newsgroups. It is obtainable for free from the UCI Machine Learning Repository (Mitchell, 1999). The dataset contains 1000 documents per newsgroup, except for soc.religion.christian, which contains 997 documents. 10 newsgroups are used, as listed in Appendix A. This gives us a total number of 9997 documents. These documents consist of a *header*, containing information on the writer, date, length, etcetera, and a *body*, containing the actual content. After removing empty documents, there were 9977 documents left. This already differs slightly from the numbers of Modha & Spangler (2003): they start with 10000 documents and are left with 9961 documents

after removing empty ones. We have tried several definitions of "empty" to remove documents: documents with 0 letters in the body, documents containing "lines: 0" in the header, and documents containing no letters at all. None of these definitions gave us the exact number of 9961 documents, so we stuck to the definition that was most logical to us: documents with 0 letters in their body.

There are no details mentioned by Modha & Spangler (2003) on how they preprocess the documents. However, they do mention various statistics regarding the number of words, number of unique 2-word phrases, and number of unique 3-word phrases. Table 1 shows us that the number of unique words, 2-word phrases and 3-word phrases found by Modha & Spangler (2003) is much larger than the numbers we obtained without preprocessing. This clearly shows that some preprocessing was performed, and leaves us to guess what has actually been done.

Before pruning	Number of words	Number of 2-word phrases	Number of 3-word phrases
Modha & Spangler (2003)	72586	429604	461132
No data preprocessing	135332	869412	1833712
With data preprocessing	70577	808282	1041411

Table 1: The size of respectively the word dictionary, 2-word phrase dictionary and 3-word phrase dictionary as obtained by Modha & Spangler (2003), by us without preprocessing steps, and with preprocessing steps

We have implemented several techniques in an attempt to reduce the dictionary sizes to come closer to the values of Modha & Spangler (2003). These steps were chosen because we believed them to be advantageous for the clustering, whilst not having too aggressive effects. The common goal is to remove most of the noise surrounding informative words. Here are the data preprocessing steps:

1. Set all capital letters to lowercase
2. Expand contractions (for more details, see Appendix B)
3. Remove all punctuation
4. Remove digits
5. Remove the following common words: article, com, edu, would
6. Remove stop words according to the NLTK English stop word list

The words in step 5 are specific to this dataset, and are removed because they appear across all newsgroups, significantly more than other words. They are thus only noise when trying to distinguish the newsgroups. The final row of Table 1 shows the sizes of the dictionaries after performing these steps. The number of words is now in line with Modha & Spangler (2003), while the number of 2-word phrases and 3-word phrases is still higher.

After collecting all unique words, 2-word and 3-word phrases, Modha & Spangler (2003) prune many them according to the following rules: only keep words (respectively 2-word and

3-word phrases) that appear in more than 64 (32, 16) documents. The number of words that are left can be found in Table 2.

After pruning	Number of words	Number of 2-word phrases	Number of 3-word phrases
Modha & Spangler (2003)	2583	2144	2268
No data preprocessing	3186	8837	10474
With data preprocessing	2957	857	1111

Table 2: The size of respectively the word dictionary, 2-word phrase dictionary and 3-word phrase dictionary after pruning as obtained by Modha & Spangler (2003), by us without preprocessing steps, and with preprocessing steps

Without data preprocessing, the dictionary sizes after pruning are larger than those of Modha & Spangler (2003). With preprocessing applied, the number of words is larger, while the number of 2-word and 3-word phrases became roughly twice as small as these of Modha & Spangler (2003). This is unexpected, because the dictionary sizes before pruning were roughly twice as large for two-word and three-word phrases. This suggests that most of the phrases only appear in a few documents. Again, the numbers obtained with data preprocessing are closer to those of Modha & Spangler (2003) than the numbers when data preprocessing is not applied.

	Average number of words per document	Average number of 2-word phrases per document	Average number of 3-word phrases per document
Modha & Spangler (2003)	50	7.19	8.34
No data preprocessing	81.6	1.63	0.05
With data preprocessing	100.8	6.37	3.54

Table 3: The average number of words, 2-word and 3-word phrases per document after pruning as obtained by Modha & Spangler (2003), by us without preprocessing steps, and with preprocessing steps

We then look at the average number of words, two-word and three-word phrases per document, as shown by Table 3. Here we can clearly see that the numbers of Modha & Spangler (2003) are far off our numbers when data preprocessing is not applied, especially for 2- and 3-word phrases. With data preprocessing, the numbers come closer, but are still off by a factor of two for words and 3-word phrases.

	n_1	n_2	n_3
Modha & Spangler (2003)	9961	8639	4664
No data preprocessing	9961	5944	343
With data preprocessing	9969	8761	4601

Table 4: The number of documents containing at least one word, one 2-word phrase and one 3-word phrase, as obtained by Modha & Spangler (2003), by us without preprocessing steps, and with preprocessing steps

Finally, the number of documents containing at least one word, 2-word phrase and 3-word

phrase can be found in Table 4. We call these quantities respectively n_1, n_2 and n_3 for later use. Again, we see poor performance when the data is not preprocessed. Surprisingly, the numbers with preprocessing are very close to those of Modha & Spangler (2003) again, even though the average number of words and three-word phrases were not in line. This similarity is convenient, as these numbers are used in the evaluation part of the clustering algorithm.

In general, the numbers resulting from our data preprocessing are relatively close to the numbers of Modha & Spangler (2003), but sometimes off by a factor of 2 to 3. The data preprocessing has notably improved the numbers, compared to no preprocessing at all. However, differences in input always lead to differences in output, meaning that our results are not exactly the same as those of Modha & Spangler (2003).

4 Methodology replication

4.1 Feature vectors

After the data has been preprocessed, we transform the data into features. To do this, the approach of Modha & Spangler (2003) is followed. Let $\mathbf{x}_i$ represent a data object for $1 \leq i \leq n$, so there are n documents in total. Each data object is represented by three feature vectors of term frequencies. We use terms of 1-, 2- and 3-word phrases for these feature vectors, where a phrase is any combination of words found in the documents. This choice was made because Mladenic (1998) shows that adding longer phrases does not improve classifier performance anymore. Creation of the first feature vector is done by making a list of all unique words. These words are then sorted based on the number of documents in which they appear, with words appearing in less than 0.64% of the documents being removed. If there are f_1 unique words remaining after the elimination, feature vector $\mathbf{F}_1$ is a f_1-dimensional vector where column entry j is the number of times the j-th word occurs in document $\mathbf{x}$, for $1 \leq j \leq f_1$. The second and third feature vector, $\mathbf{F}_2$ and $\mathbf{F}_3$ are constructed similarly using 2- and 3-word phrases. However, we keep words that appear in more than 0.32% and 0.16% of the document respectively, as the number of phrases becomes increasingly large. With these feature vectors, we perform clustering using the convex k-means clustering algorithm, as described in Section 2.

4.2 Evaluation

After having clustered the documents, we evaluate the performance of the algorithm. As the Newsgroups Dataset is pre-classfied, we can compare the clustering performance with the given ground truth. Modha & Spangler (2003) do this by looking at precision, which measures the overlap between a given clustering and the ground truth classification. Specifically, let there be c classes $\{\omega_t\}_{t=1}^{c}$ in the ground truth classification. We define a_t as the number of data objects that are assigned to class ω_t correctly. Micro-precision (micro-p) is then defined as

$$micro - p = \frac{1}{n} \sum_{t=1}^{c} a_t, \tag{18}$$

measuring the fraction of data objects that have been assigned to the correct class. It is important to note that micro-p values are only used postfactum to quantify the performance of the clustering, and are not used to find the optimal feature weighting.

5 Results replication

We applied the convex k-means clustering algorithm to the newsgroup dataset with $k = 10, 15, 20$ and with 31 different feature weightings for each k, which can be found in Appendix C. The ground truth classification is given by the 10 different newsgroups. Table 5 shows us, for every k, the optimal feature weighting and its corresponding objective function value and micro-p. Furthermore, the best and worst micro-p value of any feature weighting are shown. Lastly, the uniform weighting and its objective function value and micro-p are displayed in Table 6.

k	$\alpha^\dagger$	$Q(\alpha^\dagger)$	*micro-p*	*Best*	*Worst*
10	$(.75, .25, 0)$	131.61	0.598	.650	.186
15	$(.5833, .33, .0833)$	71.22	0.553	.647	.200
20	$(.5, .25, .25)$	46.41	0.590	.646	.213

Table 5: Results for the Newsgroups data set for $k = 10, 15, 20$ clusters with optimal feature weighting.

k	$\tilde{\alpha}$	$Q(\tilde{\alpha})$	*micro-p*
10	$(.33, .33, .33)$	179.47	.390
15	$(.33, .33, .33)$	87.99	.529
20	$(.33, .33, .33)$	49.87	.547

Table 6: Results for the Newsgroups data set for $k = 10, 15, 20$ clusters with uniform feature weighting.

Comparing this to Modha & Spangler (2003), we find for every k a different optimal feature weighting, a notably higher value for the objective function and a micro-p value of roughly 0.10 lower. For the uniform feature weightings, the objective values are again higher, with lower micro-p values. The best micro-p values are again slightly lower for us than for Modha & Spangler (2003). This indicates that the overall performance of our clustering is slightly worse. We attribute this to the differences in data preprocessing. The steps that Modha & Spangler (2003) took while extracting words out of the documents have seemingly been more effective in filtering out noise, which has helped their clustering to attain slightly higher micro-p values. Interestingly, our worst micro-p values are notably lower than those of Modha & Spangler (2003). All three worst micro-p values are attained at feature weighting (0, 0, 1), meaning that all the weight is on the feature vector containing 3-word phrases. This shows that the effect of using 3-word phrases on clustering performance is notably worse for us than for Modha & Spangler (2003), giving a possible explanation for the overall worse performance.

In Figure 1, the objective function values are plotted against the corresponding micro-p values for all feature weightings for $k = 10$. The points (2150, 0.19) and (800, 0.21) are not

shown for better visibility, but the general pattern remains similar when they are included. The optimal feature weighting is shown by putting a square around it. This plot is very similar to the plot of Modha & Spangler (2003), showing a clear negative correlation between the objective function and micro-p values. This is what we hope to see, as it means that a lower objective value corresponds to better clustering performance. Also, similar to Modha & Spangler (2003) is that the optimal feature weighting almost has the highest micro-p value, getting beaten only by one other feature weighting. We thus state the same conclusion as Modha & Spangler (2003): optimal feature weighting leads to nearly the best precision.

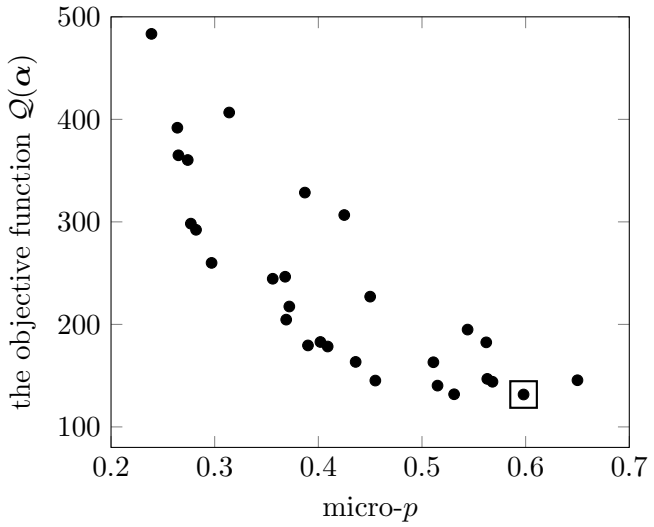

Figure 1: Scatter plot of the objective function $\mathcal{Q}(\boldsymbol{\alpha})$ versus micro-p values for $k = 10$. The micro-p value as selected by the optimal feature weighting is marked by putting a square around it.

6 Data extension

6.1 Tweets

The dataset that is used is freely obtainable via Kaggle (Suresh, 2023). It contains tweets mentioning either #btc or #bitcoin. In total, there are 4.693.144 tweets ranging from February 5th, 2021 to January 9th, 2023 in this dataset. However, we only use the tweets from January 1st, 2022, to December 31st, 2022. To keep computational times reasonable, we keep 1 in 20 tweets in this time period. There were 55 incorrectly formatted tweets and 17923 empty tweets, leaving us with 117062 tweets to use. The dataset does not contain tweets for every week in 2022: only for 32 weeks. Summary statistics of the number of tweets per week are shown in Figure 7. The longest sequence of consecutive weeks containing tweets is 9 weeks long, ranging from the end of May to July.

Each tweet comes with 13 features (like time, username, etc.), of which we use only the tweet text to evaluate sentiment. We preprocess tweet texts, trying to maintain most of the signal while filtering out noise. The following steps are taken, which are mostly consistent with Kraaijeveld & De Smedt (2020):

Statistic	Value
Number of weeks	32
Total number of tweets	117062
Mean number of tweets per week	3658
Maximum number of tweets per week	10775
Minimum number of tweets per week	386

Table 7: Summary statistics of weekly number of bitcoin tweets used in 2022

1. Remove URLs and mentions
2. Replace all capital letters by lowercase
3. Reduce character sequences longer than 3 to 3
4. Remove hashtags if they are not in the NLTK Reuters English dictionary
5. Remove ticker symbols (e.g. $BTC, $eth)
6. Expand contractions (for more details, see Appendix B)
7. Remove all non-letters (numbers, punctuation, emoticons, etc.)
8. Apply lemmatization using the NLTK WordNet Lemmatizer (reducing words like "builds" to the lemma "build")
9. Remove stop words according to the NLTK English stop word list

The goal of these steps is to remove words that bear no meaning when clustering and analyzing sentiment (e.g. done by removing stop words), and to guarantee consistent word usage across tweets when the meaning of the word is the same (e.g. done by lemmatization and expansion of contractions).

6.2 Bitcoin returns

Daily bitcoin price data for 2022 is obtained from CoinGecko (CoinGecko, 2024). We then transform this into weekly returns. Figure 2 shows the distribution of weekly bitcoin returns in 2022. We can clearly see a negative mean and a tail to the left, meaning that most weekly returns were negative. Indeed, it was a bad year for bitcoin, with a total return of -64.1% in 2022.

Table 8 tells us the same story, showing a negative mean and a minimum with larger magnitude compared to the maximum. Another thing to note is that the standard deviation is quite high, meaning that bitcoin price was volatile in 2022.

7 Methodology extension

7.1 Feature creation

After the data has been preprocessed, we transform the data into features. This done similarly as in Section 4.1, but now each data object $\mathbf{x}$ represents a tweet, with n tweets in total. Feature

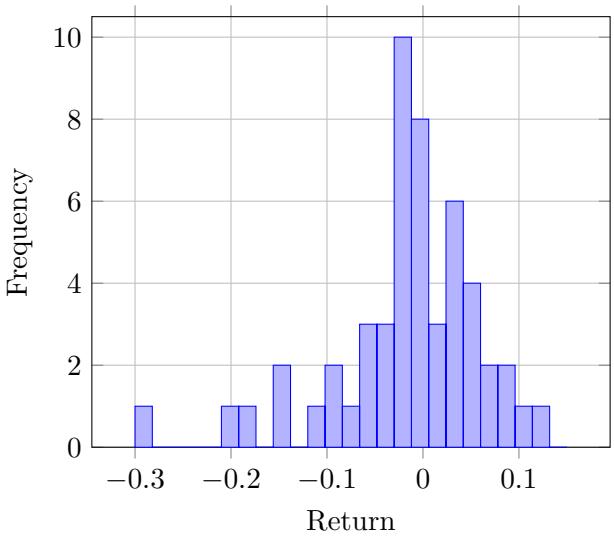

Figure 2: Histogram of weekly bitcoin returns in 2022

Statistic	Value
Number of observations	52
Mean	-0.016
Standard Deviation	0.077
Maximum	0.126
Minimum	-0.296

Table 8: Summary statistics of weekly bitcoin returns in 2022

vectors $\mathbf{F}_1$, $\mathbf{F}_2$ and $\mathbf{F}_3$ contain the number of occurrences of each word, 2-word phrase and 3-word phrase again. However, pruning of the lowest-used words is done differently. Pruning based on the number of tweets a word appears in is not suitable, as tweets only contain a few words, whereas documents contained up to thousands. Therefore, we sort phrases based on usage and keep only the top 10%, 5% and 2.5% of phrases respectively. As we perform weekly clusterings, the word lists and corresponding feature vectors are also constructed on a weekly basis. Feature vectors in different weeks thus have different lengths, as the number of unique words is not the same in every week.

7.2 Sentiment scores

Clusters are constructed using $k = 3$ and $k = 5$. We perform a clustering for each week, as described in Section 2. To reduce computational times, we use only 19 different feature weightings, which are listed in Appendix C. Let t denote the week number, for $1 \leq t \leq T$. We can then denote the clusterings as $\{\pi^{\dagger}_{u,t}\}_{u=1}^{k}, 1 \leq t \leq T$. For each week, the partitioning obtained by the optimal feature weighting is used to calculate sentiment scores. For this, we use the VADER algorithm. VADER is a lexicon and rule-based algorithm which gives a body of text either a negative, neutral or positive rating. This is done by calculating the *Compound score*, also referred to as *Sentiment score*, which ranges between -1 and 1. The sentiment of a text is said to be negative when the score is smaller than -0.05, positive when it is larger than

0.05, and neutral for a score between -0.05 and 0.05.

We calculate the sentiment score of a cluster by obtaining the compound score of all individual tweets using the VADER algorithm, then taking the mean. This gives us the $k \times 1$ compound score vectors $\mathbf{s}_t$, where element u corresponds to the compound score of cluster $\pi^{\dagger}_{u,t}$, for $1 \leq u \leq k$, $1 \leq t \leq T$. We then sort the sentiment scores in $\mathbf{s}_t$ in ascending order, giving us $\mathbf{s}^*_t$. The lowest weekly compound score is always $s^*_{t,1}$, and the highest compound score is $s^*_{t,k}$. For later use, let $\mathbf{s}_u = (\mathbf{s}^*_{1,u}, \mathbf{s}^*_{2,u}, \ldots, \mathbf{s}^*_{T,u})$ for $1 \leq u \leq k$ be the compound scores of the same clusters in different weeks.

7.3 Evaluation

The first step in the evaluation is to test whether the convex k-means clustering algorithm creates clusters with statistically different sentiment scores from each other. We use Smirnov tests for this, which tests how likely it is that two samples are drawn from the same unknown distribution, with a null hypothesis of identical distributions. We refer to Berger & Zhou (2014) for more details on this test. The test is performed for each combination of $\mathbf{s}_u$ and $\mathbf{s}_v$, with $u \neq v$, for $1 \leq u, v \leq k$.

Then, we use the sentiment score vectors to create VAR models. Let r_t be the return of bitcoin in week t. The sentiment scores in $\mathbf{s}^*_t$ are highly correlated with each other: they are all higher in periods with overall good sentiment, and lower in periods with overall bad sentiment. Therefore, we include only one element of them at a time in a VAR model, to avoid multicollinearity issues. We define the 2×1 vector of variables $\mathbf{y}_{t,u} = (r_t,\ s^*_{t,u})'$. We can then write the VAR model as follows, for $1 \leq t \leq T$:

$$\mathbf{y}_{t,u} = \mathbf{c} + \sum_{q=1}^{p} A_q \mathbf{y}_{t-q,u} + \boldsymbol{\epsilon}_t, \tag{19}$$

where $\mathbf{c}$ is a 2×1 constant, A_q are the 2×2 coefficient matrices, and $\boldsymbol{\epsilon}_t$ is the 2×1 vector of error terms. We estimate these models for $1 \leq u \leq k$ using Ordinary Least Squares. The lag order p is fixed at $p = 2$, to still include some lags while not losing too many observations. As the VAR model requires stationary data, we apply the *Augmented Dickey-Fuller test* (ADF test) on $\mathbf{y}_{t,u}$ beforehand, and transform the variables to be stationary if necessary. We refer to Mushtaq (2011) for more details about the ADF test.

Data with missing values can not be used to construct a VAR model, so we use the longest sequence of consecutive weeks containing tweets, which is a 9 week period in May-July. This period is referred to as *the longest sequence*. However, there is only 1 week with no tweets before that, preceded by 5 consecutive weeks with tweets. We refer to this entire 15-week period as *the imputed longest sequence*, as we impute compound scores for the week with missing data by taking the average of the scores in the week before and after it. While having more observations leads to lower standard deviations, imputation does come at the cost of decreased model accuracy. Therefore, we estimate VAR models for both sets of observations.

After estimating these models, we apply Granger causality tests to see whether the score vectors have predictive power for bitcoin returns. The null hypothesis is that $s^*_{t-1,u}$ and $s^*_{t-2,u}$ do not Granger-cause r_t. This test is equivalent to testing whether the coefficients of $s^*_{t-1,u}$ and

$s^*_{t-2,u}$ are jointly equal to zero, for which we use an F-test.

Finally, we want to compare the performance of sentiment scores to sentiment scores obtained without using k-means clustering. This is done by calculating the compound score of each individual tweet, then taking the average per week. The difference is that there is now one sentiment score per week, instead of k scores. The new compound scores are evaluated in the same way as before: we apply the ADF test, construct a VAR model if the data is stationary and apply a Granger causality test, for both the longest sequence and the imputed longest sequence. Imputation is done again in the same way.

8 Results extension

The results shown in this section are all for $k = 3$; using $k = 5$ gave similar results.

8.1 Smirnov tests

Comparison	Test Statistic	P-value	Conclusion
$\mathbf{s}_1$ and $\mathbf{s}_2$	0.46875	0.001	Reject H_0 of identical distributions
$\mathbf{s}_2$ and $\mathbf{s}_3$	0.5625	<0.001	Reject H_0 of identical distributions
$\mathbf{s}_1$ and $\mathbf{s}_3$	0.84375	<0.001	Reject H_0 of identical distributions

Table 9: Results of Smirnov test comparing the weekly highest, middle and lowest sentiment scores with each other.

We first apply Smirnov tests to the sentiment scores. Table 9 shows us that the null hypothesis is rejected for all three combinations, meaning that the distributions are significantly different from each other. In other words: the convex k-means clustering algorithm is able to cluster tweets such that the average sentiment scores in the clusters are significantly different from each other.

8.2 Correlation of sentiment and returns

Next, we turn our attention to the correlation between weekly returns and sentiment scores. Figure 3 shows scatter plots of weekly returns against respectively the lowest cluster compound scores and the highest cluster compound scores of the previous week. We do not see a clear correlation in either scatter plot, and there are some outliers to the downside which do not have a low compound score. Figure 4 shows a scatter plot of weekly returns against the average compound score of the three clusters of the previous week. Again, we do not see a clear correlation between returns and sentiment score, and we even see high compound scores corresponding to some of the lowest returns.

Looking at the Pearson correlation coefficients in Table 10, we surprisingly see that there is a negative correlation between all compound scores and returns in the next period. This may be caused by the outliers with low returns but high compound scores. Another potential cause is the autocorrelation of returns being negative (-0.086), combined with a positive correlation

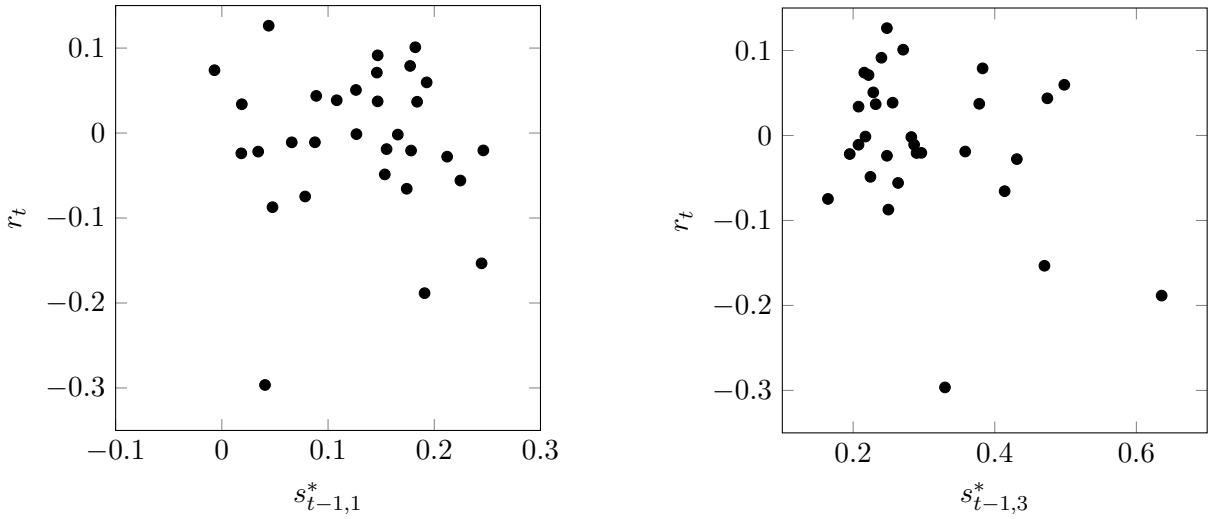

Figure 3: Scatter plots of weekly returns against the minimum compound scores and maximum compound scores of the previous week.

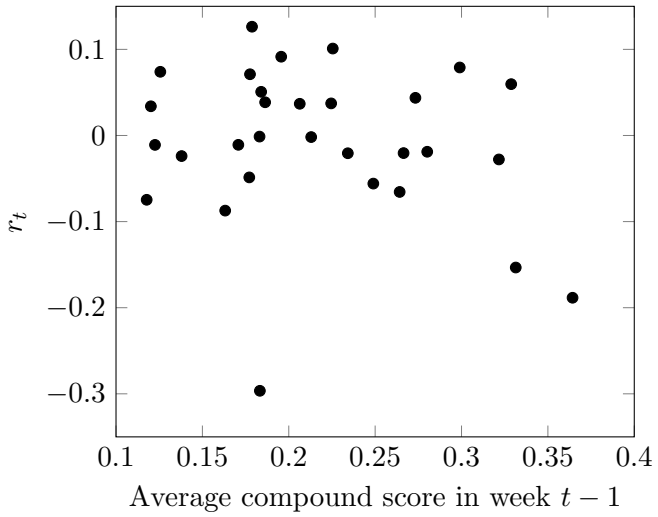

Figure 4: Scatter plot of returns versus the average compound score of the previous week.

between returns and compound scores in the same period. Nevertheless, there are no indications of a positive relationship between returns and sentiment scores in the previous week.

8.3 VAR models

We now turn our attention to evaluation with statistical models, first looking at the VAR models for the longest sequence. The variables used are all stationary as judged by the ADF test, of which the results are shown in Appendix D. The VAR models are shown in Table 11. The rows correspond to the 3 different models, using respectively the compound score of the lowest, middle and highest clusters of each week. Looking at the p-values of the coefficients, they are insignificant at a 5% level for all variables except for the sentiment score of the highest cluster in the third model. This seems promising, but the Granger causality test results in Table 12 show us that the highest cluster sentiment score does not have predictive power for bitcoin returns,

	r_{t-1}	r_t
$s^*_{t-1,1}$	0.116	-0.058
$s^*_{t-1,3}$	0.118	-0.321
Average	0.148	-0.174

Table 10: Correlation coefficients between different compound scores and returns.

and neither do the other sentiment scores in their models.

Model	Variable	Coefficient	p-value
VAR with the lowest compound scores	constant	0.030	0.901
	r_{t-1}	-0.170	0.792
	$s^*_{t-1,1}$	0.746	0.573
	r_{t-2}	0.265	0.679
	$s^*_{t-2,1}$	-1.293	0.345
VAR with the middle compound scores	constant	0.103	0.929
	r_{t-1}	-0.134	0.832
	$s^*_{t-1,2}$	0.839	0.809
	r_{t-2}	0.063	0.919
	$s^*_{t-2,2}$	-1.567	0.600
VAR with the highest compound scores	constant	0.760	0.022
	r_{t-1}	-0.186	0.603
	$s^*_{t-1,3}$	-3.845	0.010
	r_{t-2}	0.413	0.247
	$s^*_{t-2,3}$	0.807	0.418

Table 11: Results of VAR models for the longest sequence. Each row corresponds to a different model, containing respectively the lowest, middle and highest weekly compound score.

Variable	F-test	p-value	Conclusion
$\mathbf{s}_1$	0.7231	0.5804	Do not reject H_0
$\mathbf{s}_2$	0.4383	0.6953	Do not reject H_0
$\mathbf{s}_3$	3.5974	0.2175	Do not reject H_0

Table 12: Granger causality test results of all three different models for the longest sequence. We test whether the variable Granger-causes returns for all three variables separately.

In an attempt to increase the sample size, we look at VAR models for the imputed longest sequence. With the extra observations added, the variables are still stationary as judged by ADF tests (more details are in Appendix D). The VAR models are shown in Table 13, with rows corresponding again to the 3 different models, using respectively the compound score of the lowest, middle and highest clusters of each week. There are now no significant coefficients at a 5% level: all p-values are larger than 0.05. The results of the Granger causality tests, shown in Table 14, tell the same story: no significant test statistics, meaning no predictive power.

Model	Variable	Coefficient	p-value
VAR with the lowest compound scores	constant	-0.007	0.943
	r_{t-1}	-0.214	0.514
	$s^*_{t-1,1}$	0.756	0.263
	r_{t-2}	0.143	0.662
	$s^*_{t-2,1}$	-0.938	0.115
VAR with the middle compound scores	constant	0.115	0.583
	r_{t-1}	-0.242	0.493
	$s^*_{t-1,2}$	-0.019	0.980
	r_{t-2}	0.100	0.780
	$s^*_{t-2,2}$	-0.722	0.318
VAR with the highest compound scores	constant	0.138	0.278
	r_{t-1}	-0.277	0.448
	$s^*_{t-1,3}$	-0.448	0.143
	r_{t-2}	0.158	0.631
	$s^*_{t-2,3}$	-0.090	0.797

Table 13: Results of VAR models for the imputed longest sequence. Each row corresponds to a different model, containing respectively the lowest, middle and highest weekly compound score.

Variable	F-test	p-value	Conclusion
s_1	1.2999	0.3245	Do not reject H_0
s_2	0.5108	0.6183	Do not reject H_0
s_3	1.3997	0.3011	Do not reject H_0

Table 14: Granger causality test results of all three different models for the imputed longest sequence. We test whether the variable Granger-causes returns for all three variables separately.

8.4 Without clustering

To investigate whether the lack of predictive power is a result of using k-means clustering, we investigate predictive power of sentiment scores obtained without clustering. This is done by calculating the compound score of each individual tweet, then taking the average per week. Figure 5 shows a scatter plot between these sentiment scores and returns in the next week. The Pearson correlation coefficient between returns and previous week average compound score is -0.288, likely influenced by the observations with low returns but relatively high compound scores.

Interestingly, the compound scores calculated without clustering are not stationary in the longest sequence. Even when applying first differences, and taking first differences of the log, there is still no stationarity. The p-values for these tests are in Appendix D. Therefore, we do not construct VAR models for the longest sequence. The scores are stationary again for the imputed longest sequence (see Appendix D), so we can construct a VAR model for that time period. This model is shown in Table 15. Interestingly, there are again no significant coefficients, besides the constant.

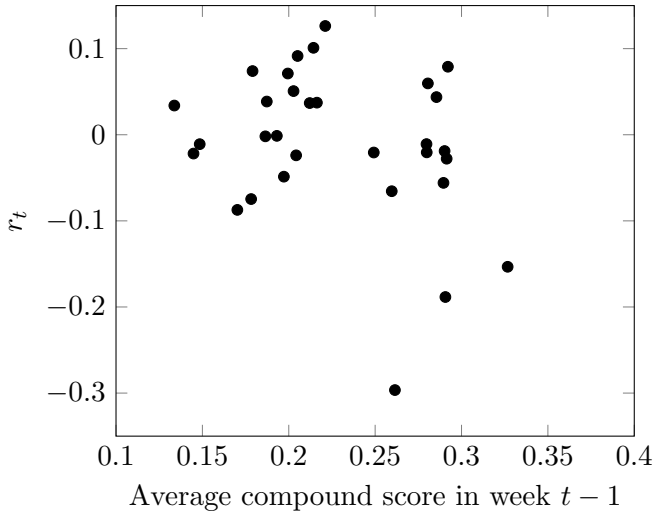

Figure 5: Scatter plot of returns versus the average compound score of the previous week, without using clustering.

Compound score used	Variable	Coefficient	p-value
Average without clustering	constant	0.426	0.047
	r_{t-1}	-0.252	0.500
	$s_{t-1,1}$	-1.107	0.287
	r_{t-2}	0.353	0.303
	$s_{t-2,1}$	-0.896	0.341

Table 15: VAR Model Results for longest sequence with imputation

When applying a Granger causality test in this model, we find a test statistic value of $F = 2.4692$, with corresponding p-value $p = 0.1462$. Again, the test concludes that there is no predictive power of the sentiment score for returns. This makes us conclude that twitter sentiment can not predict bitcoin returns anymore in 2022. Using k-means clustering to obtain sentiment score did not notably improve or worsen the performance, even though the convex k-means clustering algorithm is able to cluster tweets such that the sentiment scores in the clusters are significantly different from each other.

9 Conclusion

This study first tries to replicate the application of feature weighting in k-means clustering on text data by Modha & Spangler (2003). We then investigate whether Twitter sentiment has predictive power for bitcoin returns, using feature weighting in k-means clustering to obtain sentiment scores. Demand for bitcoin is increasing due to the launch of ETFs and its role as a diversifying asset. Bitcoin is a volatile asset, making it lucrative for investors to predict these movements to maximize returns. We propose a new methodology to extract sentiment scores from tweets. After removing most noise from the tweets, we use k-means clustering with feature weighting to cluster them on a weekly basis. We then use the VADER sentiment algorithm to calculate the mean sentiment score of clusters for each week. Afterward, we test if clusters have

significantly different sentiment scores, and if sentiment has predictive power for bitcoin returns.

The results of Modha & Spangler (2003) can not be replicated perfectly, because they mention no details on how they preprocess the documents. We attempted to apply steps such that our numbers come close. While we find slightly different objective function and precision values, there is still a negative correlation between these two. Besides that, the optimal feature weighting almost attains the highest micro-p value. This makes us state the same conclusion as Modha & Spangler (2003): optimal feature weighting nearly achieves the best clustering precision.

The results of our extension indicate that k-means clustering with feature weighting creates clusters with significantly different sentiment scores, as tested by Smirnov tests. Even though tweets are efficiently clustered by sentiment, the cluster sentiment scores do not have predictive power for bitcoin returns, as tested by Granger causality tests. However, we also show that sentiment scores obtained without using clustering do not have predictive power for bitcoin returns. We can therefore not say that the convex k-means clustering algorithm has improved or worsened the sentiment analysis. We thus advise investors to turn to other ways of gauging sentiment, or use completely different variables to predict bitcoin returns.

For future research, we suggest exploring whether sentiment scores can predict bitcoin returns jointly with other variables, such as bitcoin buy/sell volume, stock index returns, and total tweet volume. Additionally, shorter time frames could be explored, such as daily or even hourly correlation between bitcoin sentiment and returns. Finally, the analysis can be extended to smaller cryptocurrencies, as their markets may be still less efficient compared to bitcoin's.

References

Ahuja, S. & Dubey, G. (2017). Clustering and sentiment analysis on twitter data. In *2017 2nd international conference on telecommunication and networks (tel-net)* (pp. 1–5).

Akhtaruzzaman, M., Sensoy, A. & Corbet, S. (2020). The influence of bitcoin on portfolio diversification and design. *Finance Research Letters*, *37*, 101344.

Anamika, Chakraborty, M. & Subramaniam, S. (2023). Does sentiment impact cryptocurrency? *Journal of Behavioral Finance*, *24*(2), 202–218.

Asur, S. & Huberman, B. A. (2010). Predicting the future with social media. In *2010 ieee/wic/acm international conference on web intelligence and intelligent agent technology* (Vol. 1, pp. 492–499).

Barradas, A., Tejeda-Gil, A. & Cantón-Croda, R.-M. (2022). Real-time big data architecture for processing cryptocurrency and social media data: A clustering approach based on k-means. *Algorithms*, *15*(5), 140.

Berger, V. W. & Zhou, Y. (2014). Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*.

Bollen, J., Mao, H. & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, *2*(1), 1–8.

CoinGecko. (2024). *Bitcoin price dataset.* Author. (https://www.coingecko.com/en/coins/bitcoin/historical_data)

De Amorim, R. C. (2016). A survey on feature weighting based k-means algorithms. *Journal of Classification*, *33*, 210–242.

DeSarbo, W. S., Carroll, J. D., Clark, L. A. & Green, P. E. (1984). Synthesized clustering: A method for amalgamating alternative clustering bases with differential weighting of variables. *Psychometrika*, *49*, 57–78.

Guesmi, K., Saadi, S., Abid, I. & Ftiti, Z. (2019). Portfolio diversification with virtual currency: Evidence from bitcoin. *International Review of Financial Analysis*, *63*, 431–437.

Hutto, C. & Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international aaai conference on web and social media* (Vol. 8, pp. 216–225).

Kraaijeveld, O. & De Smedt, J. (2020). The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, *65*, 101188.

Liu, B. (2022). *Sentiment analysis and opinion mining.* Springer Nature.

Lütkepohl, H. (2013). Vector autoregressive models. In *Handbook of research methods and applications in empirical macroeconomics* (pp. 139–164). Edward Elgar Publishing.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Mazur, M. & Polyzos, E. (2024). Spot bitcoin etf. *Available at SSRN 4810965*.

Mitchell, T. (1999). *Twenty newsgroups dataset.* UCI Machine Learning Repository. (https://archive.ics.uci.edu/dataset/113/twenty+newsgroups)

Mladenic, D. (1998). Word sequences as features in text-learning. *Proc. the 7th ERK'98*, 145–148.

Modha, D. S. & Spangler, W. S. (2003). Feature weighting in k-means clustering. *Machine learning*, *52*, 217–237.

Mushtaq, R. (2011). Augmented dickey fuller test. *SSRN*.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org*.

Olabanji, S. O., Oladoyinbo, T. O., Asonze, C. U., Adigwe, C. S., Okunleye, O. J. & Olaniyi, O. O. (2024). Leveraging fintech compliance to mitigate cryptocurrency volatility for secure us employee retirement benefits: Bitcoin etf case study. *Available at SSRN 4739190*.

Suresh, K. (2023). *Bitcoin tweets dataset.* Kaggle. (https://www.kaggle.com/datasets/kaushiksuresh147/bitcoin-tweets/data)

Tumasjan, A., Sprenger, T., Sandner, P. & Welpe, I. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the international aaai conference on web and social media* (Vol. 4, pp. 178–185).

Vujičić, D., Jagodić, D. & Ranđić, S. (2018). Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)* (pp. 1–6).

A Newsgroups used

The newsgroups in Table 16 were used in the replication part. They are the same as Modha & Spangler (2003) use.

sci.crypt	comp.windows.x	comp.sys.mac.hardware
rec.autos	rec.sport.baseball	soc.religion.christian
sci.space	talk.politics.guns	talk.politics.mideast
misc.forsale		

Table 16: The newsgroups used in the replication part.

B Contractions

Contractions are expanded by using the following table, consisting of three specific contractions for irregular words, and eight regular contractions

Contraction	Expanded
won't	will not
can't	can not
ain't	is not
n't	not
're	are
's	is
'd	would
'll	will
't	not
've	have
'm	am

Table 17: The contractions and how they are expanded

C Feature weightings

Table 18 shows the 31 different feature weightings that are used in the replication. They are the same as used by Modha & Spangler (2003).

1, 0, 0	0.75, 0.25, 0	0.5, 0.5, 0	0.25, 0.75, 0
0, 1, 0	0.8333, 0.0833, 0.0833	0.5833, 0.3333, 0.0833	0.3333, 0.5833, 0.0833
0.0833, 0.8333, 0.0833	0.6666, 0.1666, 0.1666	0.4166, 0.4166, 0.1666	0.1666, 0.6666, 0.1666
0.75, 0, 0.25	0.5, 0.25, 0.25	0.25, 0.5, 0.25	0, 0.75, 0.25
0.5833, 0.0833, 0.3333	0.3333, 0.3333, 0.3333	0.0833, 0.5833, 0.3333	0.4166, 0.1666, 0.4166
0.1666, 0.4166, 0.4166	0.5, 0, 0.5	0.25, 0.25, 0.5	0, 0.5, 0.5
0.3333, 0.0833, 0.5833	0.0833, 0.3333, 0.5833	0.1666, 0.1666, 0.6666	0.25, 0, 0.75
0, 0.25, 0.75	0.0833, 0.0833, 0.8333	0, 0, 1	

Table 18: The 31 different feature weightings that are used in the replication

Table 19 shows the 19 different feature weightings that are used in the extension. They are a subset of the 31 feature weightings in 18. We reduced the number of feature weightings to reduce the computational time.

1, 0, 0	0.75, 0.25, 0	0.5, 0.5, 0
0, 1, 0	0.6666, 0.1666, 0.1666	0.1666, 0.6666, 0.1666
0.75, 0, 0.25	0.5, 0.25, 0.25	0.25, 0.5, 0.25
0, 0.75, 0.25	0.3333, 0.3333, 0.3333	0.5, 0, 0.5
0.25, 0.25, 0.5	0, 0.5, 0.5	0.1666, 0.1666, 0.6666
0.25, 0, 0.75	0, 0.25, 0.75	0, 0, 1

Table 19: The 19 different feature weightings that are used in the extension

D Augmented Dicky-Fuller test results

Table 20 shows the results of the ADF tests when the longest sequence is used. We use a 10% significance level, as the number of observations is small. We conclude that all the variables are stationary.

Variable	**Test Statistic**	**p-value**	**Conclusion**
Returns	-3.698	0.071	Reject H0
Compound score cluster 1	-3.555	0.099	Reject H0
Compound score cluster 2	-4.999	0.001	Reject H0
Compound score cluster 3	-7.341	<0.001	Reject H0

Table 20: ADF Test Results longest sequence

Table 21 shows the results of the ADF tests when the imputed longest sequence is used. At a 10% significance level, we again conclude that all the variables are stationary.

Table 22 shows the results of the ADF tests when clustering is not used to obtain the sentiment scores. At a 10% significance level, none of the variables is stationary for the longest sequence. However, for the imputed longest sequence, the compound scores are stationary.

Variable	Test Statistic	p-value	Conclusion
Returns	-4.702	0.003	Reject H0
Compound score cluster 1	-3.322	0.017	Reject H0
Compound score cluster 2	-5.185	<0.001	Reject H0
Compound score cluster 3	-6.080	<0.001	Reject H0

Table 21: ADF Test Results longest sequence with imputation

Variable	Test Statistic	p-value	Conclusion
Compound scores	-2.727	0.439	Do not reject H_0
First difference compound scores	-2.083	0.783	Do not reject H_0
First difference of log of compound scores	-2.114	0.770	Do not reject H_0
Compound scores imputed sequence	-4.611	0.004	Reject H_0

Table 22: Augmented Dickey-Fuller (ADF) Test Results for no clustering

E Programming code

In this section, we describe the files and programming code which are contained in the ZIP folder. These files are sufficient to replicate all the results in this paper.

The ZIP folder contains two subfolders, corresponding to the replication and the extension. Both of these folders contain Python code files, data files, and pickle files. The data files are freely obtainable from the internet, as described in this paper. The Python code files execute the steps described in this paper to get the results. They can just be run and will print all the results; only the directory of the data file may need to be changed in some files, which is told in a comment if applicable. A description of each file is given in Table 23 and Table 24. The Python files should be run in the order they are put in the tables. Some of these files save intermediate results, which are used again in the next files. These intermediate results are in both folders in the sub-folder 'pickle files'. In this way, all files can be run without having to wait hours for the preceding code to finish. When using these pickle files, be sure to change the directory to the correct path when the pickle files are opened in the code.

Note that some files may need up to 14 hours to completely run the code; these files with longer running times have a comment at the top of the code indicating how long it approximately takes.

File name	Description
20_newsgroups	The 20 newsgroups dataset as obtained from UCI, with the 10 unused newsgroups deleted
replication preprocessing	The code for preprocessing the newsgroups dataset, prints the results in Section 3 .
replication nopreprocessing	The code for getting the results in Section 3 without preprocessing; the results are printed.
replication clustering randominit	The code for clustering the feature vectors obtained from 'replication preprocessing', with random initialization; the results from this file are used in Section 5 .
replication clustering conceptinit	The code for clustering feature vectors according to the concept initialization; this did not improve results, so we have not used the results obtained from this file.
replication evaluation	The code for making the table and figure in Section 5 .
feature weightings	The file we used to find what the feature weightings of Modha & Spangler (2003) were.

Table 23: The programming code used for the replication and a short description of each file

File name	Description
Bitcoin_returns	Daily bitcoin price data for 2022, as obtained from CoinGecko
Bitcoin_tweets	The database of bitcoin tweets obtained from Kaggle. Note that this file may be too big to open in Excel; either Notepad++ or PowerPivot in Excel are good options to examine the file
extension returns	The code for transforming daily bitcoin price into weekly returns and the summary statistics in Section 6.
extension preprocessing	The code that performs the preprocessing steps described in Section 6 on the tweets.
extension fvs	The code that transforms the preprocessed tweets into feature vectors
extension clustering randominit	The code that clusters the feature vectors obtained 'extension fvs'.
extension evaluation	The code for analyzing and evaluating the clusterings. All results in Section 8 and the tweets summary statistics in Section 6 are output by this code

Table 24: The programming code used for the extension and a short description of each file