Knowledge-based recommender system for stocks using clustering and nearest neighbors

Software Engineering
Master's Degree Programme in Information and Communication Technology
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis

Author:
Joonatan Vanhala

Supervisors:
Antero Järvi
Jukka Heikkonen

June 2023

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Software Engineering
Programme: Master's Degree Programme in Information and Communication Technology
Author: Joonatan Vanhala
Title: Knowledge-based recommender system for stocks using clustering and nearest neighbors
Number of pages: 79 pages
Date: June 2023

Recommendation systems and algorithms are part of many services we use today. Online marketplaces, social media sites, streaming services, and many others lean on the algorithms to provide content for a user that match one's likings. A practical example of such system is Netflix which may recommend movies to a user based on one's viewing history. "Since you watched X, you might also be interested in Y". Even though these algorithms are used in multiple services, there are still applications where the power of recommendation systems hasn't been fully utilized for a public consumer. One of these are publicly traded stocks. Investing into publicly listed stocks is a common way to generate wealth. There are thousands of companies listed in NYSE and NASDAQ stock markets in the USA only. For an investor this is a lot to choose from. Some may prefer growth stocks and others blue-chip stocks with high dividend yield. One can search higher risk-reward returns from stocks that are dropping heavily and other seek steady growth in their preferred stocks.
This thesis aims to implement a knowledge-based recommendation system that considers not only stock's financial data but also historical price development to give meaningful stock recommendations based on an input of a single stock in a case-based manner. The implementation considers two different approaches when combining these distinctly different data types. The experimental development relies on clustering techniques to categorize similar stocks into different recommendation lists and finally sorting the lists using nearest neighbors. The evaluation of the approaches is conducted using machine learning evaluation methods combined with evaluation metrics used in recommender systems. The final best performing implementation is built on top of K-means clustering technique and t-SNE dimensionality reduction method. Trendlines and financial data of the stocks are combined using separately computed distance matrices. Similarity between the trendlines is computed using customized cosine-distance function. Finally the thesis presents a Stock Recommender using Similarity-based Methods (StockRSM).

Keywords: Recommender systems, data mining, clustering, nearest neighbors, stocks

Table of Contents

Figures

Tables

1 Introduction

1.1 Background

Recommendation systems are part of many services we use today. Online marketplaces, social media sites, streaming services, and many others lean on the algorithms to give their users the content that is the most suitable for them. The aim and the technology behind these algorithms vary and there is no single algorithm that can handle any recommender task. For example we can recommend items based on given constraints by filtering which items fill the given rules. The user's role in this case is to provide information reactively which the recommender system is able to use. Then again in some cases the recommendations are made to a user proactively, which is common in social media sites. [1]

When it comes to finding suitable stocks, investor may look at the stock's profiles and their financial statements. Each publicly listed company have this data presented publicly to all investors so that they can freely search and study what kind of companies there is and how they are performing in the market. This data contains basic profile information as name of the company, sector, industry, country, and market value but also more detailed financial data. Financial data contains income statement, balance sheet and cash flows. These different data sets contain multiple values and ratios that describe the profitability, placement in the market and overall performance of a company. Common values include for example, market

value, revenue, and debt. [2]

Trendline in the stock market is a chart that describes price movements by connecting series of prices together. These trendlines can be drawn in many different intervals. We can take for example one for each day of the last year and draw a trendline based on those price values. Or we can take an interval of week or even just one minute. The more price points a chart has and the more frequent the points are, the more accurate the chart is. Trendline as the name suggests describes a trend in the market. Trendline can describe either a single stock or a group of stocks. [3]. Correlation in trendlines is a statistical measure that describes how different stocks or group of stocks may move in relation to one another. Often similar companies with similar outlooks and expectations in the market tend to move strongly correlated in relation to one another. [4]

As stated previously there are thousands of stocks for the investor to choose from in US only. A common tool to find new stocks to invest is to use stock screeners. While these applications provide easy way of filtering data with given rules, they may not be suitable for all use cases, and they contain limitations. Stock screeners allow user to select constraints for a set of features. For example user could set market value between 1 and 2 billion and, sector to finance and ebitda margin greater than 10 percent. Then user receives a list of companies that fill these rules. Depending on the data set, how strict and limiting the rules are, the length of the result set varies largely. These kinds of systems fall under the knowledge-based recommendation algorithms, more precisely constraint-based recommenders. [5]

1.2 Use cases

Investors follow their investing plans when making decisions to buy and invest in companies. Investing plan is each individual investor's rulebook which guides what kind of companies are suitable for one's portfolio. Portfolio is a set of multiple stocks

that an investor has chosen to match one's investing plan. [6] One common way to find new stocks to invest on is to use stock screeners mentioned in the previous chapter. This helps the investor to narrow down the number of potential stocks. However it requires lots of knowledge from the investor to know what kind of parameters are suitable for each case. Even a slight difference in the constraints can exclude a result that could have been a good recommendation for the investor. The issue with these screeners is that they do not show the actual closest match or matches as they are only designed to filter the fitting items. In addition to the issue of selecting exactly correct constraints, these screeners usually only consider the financials of the stocks and do not consider the trendlines of the stocks. Trendlines contain information for a stock that financial data cannot describe. This is problematic since price actions of stocks often tell an investor what kind of expectations the market has for a specific stock. Often these expectations match for many stocks in comparison to each other's, and we can therefore find correlated patterns from the trendlines. However, it is difficult or even impossible to go through thousands of stocks manually to find these correlated patterns from stock key figures and there is no generally known way for an investor to find similar stocks based on trendlines. We can consider the process of finding similar stocks in three different example use cases.

Use case 1. Diversification based on modern portfolio theory

The first use case is related to the modern portfolio theory. Modern portfolio theory relies on the idea that a portfolio should be diversified. Diversification is an investing strategy that aims to lower the risk of portfolio by allocating money in multiple investments that are not correlated positively. Positive correlation refers to an occurrent where two or more investments provide the same response to market movements. Negative on the other hand would mean that the investments are not correlated. [6] Suppose an investor is planning a portfolio that bases its investing

plan on modern portfolio theory. In order to minimize risk according to MPT, the investor would aim to add stocks to the portfolio that are not correlated nor closely similar. Simple stock screeners are not able to detect on how correlated specific stocks might be so investor will have to do the manual work in order to find these correlations in the market. If the investor would be able to get a recommendation of most similar stocks for a given stock, the investor would be able to avoid of choosing closely correlating stocks.

Use case 2. Highly focused stock portfolio

Let's consider a second use case where an investor X is starting a new portfolio. X decides to create an investment plan for the portfolio. Investment plan may for example contain rules for P/E ratio, market value or operating margins. X is confident that these factors are key points for the following years to come to outperform the market. In contrast to MPT the investor is betting on specific factors in order to beat the market. X is aware of a company which is a perfect candidate for the portfolio. Now to balance the risk X decides to find other companies that fill the same investment plan rules as this specific company but still correlate with the financial data and historical performance. In other words, the investor needs similar companies that are as well positioned. By investing in other similar companies the investor may be able to avoid problems that the single specific company could face. But still, with the combination of multiple stocks filling the same rules, X is able to balance the risk for unexpected events which may occur for that specific company. If a single stock fails due to factors not in investors hands, there still are many other candidates that can fulfill the original plan that was set to beat the market. Now the investor seeks to find good candidates along with the one stock that has already passed the rules. However, X does not remember the actual key figures by memory, and it would take lots of effort to find the correct key figures and use them in a

stock screener. Screeners also do not recommend the closest match for the given arguments since they work in constraint-based manner. Whichever item checks all the rules set by the user will be recommended, it is up to user decide which of these are actually closest to the original investing plan. Now investor will then go through all of the matches and try to limit the number of candidates. The other option for the investor would be to narrow down the set of rules but this could possibly in worst case scenario exclude candidates that were missed out on a single rule but were perfect fits based on the other rules. Also, if Y was to also find the companies that have been correlating based on historical price performance, it would be extremely difficult to find these stocks using manual work.

Use case 3. Comparison of similar companies

Third use case contains another investor Y who is interested to look the competition and matching stocks for one's portfolio. Y has recently noted that a specific company has great key figures and historical price chart is matching the investor's expectations. However the investor is not sure if the stock is the best fit for one's portfolio. There might be better suited companies that one is not aware of. How can Y quickly find the alternatives for this stock by finding matching companies with similar price chart and key figures. Again, the investor could use stock screeners but would face eventually the problem of finding exact correct rules to filter by and even then, one would go through each one of them in order to find the most similar ones since the screeners do not show the closest matches.

As seen, there are multiple problems in finding correlating and similar companies from the markets. There are no fast applications to process thousands of stocks to find similar stocks for an input without managing constraints to search by. Managing and changing constraints to filter out the results requires lots of back-and-forth

manual work and from the user's perspective it is not

1.3 Research questions

The hypothesis for this thesis is that there are correlations and similarities included in the financial data and trendlines of publicly traded stocks and therefore the stocks can be categorized. Categorizing stocks enables similar stocks to be recommended based on a single stock. This thesis aims to answer the following research questions.

RQ1. Is it possible to cluster a set of stocks based on their trendlines and financial data in order to form a recommendation system?

RQ2. How the distinctly different data types, financials and trendlines of the stocks, can be combined for the clustering process?

RQ3. If multiple approaches and techniques can cluster the set of stocks, which one of them provides the most promising results?

1.4 Scope and methods

Approach for finding a working solution relies on experimental software development within the chosen algorithms. Different approaches will be evaluated firstly based on their capability to categorize a set of stocks. Secondly, if an algorithm is able to achieve this it is evaluated based on following factors in order to measure the performance considering the RQ3.

1. Diversity of list of recommendations 2. Recommendation coverage 3. Distinguishability and clearness of the stock clusters

Implementation was designed for common stocks only and therefore funds, ETFs, ETCs, acquisition companies and warrants were excluded from the work. The stocks chosen for the thesis were from NYSE and NASDAQ and the data was collected from those markets only. Stocks outside of those markets are excluded from the work. As the aim is to suggest active stocks to a user, we cannot use stocks that have been delisted from the market. In addition we cannot find enough data for stocks that have been recently listed as they have not yet published annual report or possibly not even 10-K report. Therefore stocks listed in the last year cannot be considered. The timeframe for historical data of the stocks is narrowed down to last five years to speed up the development process. In terms of historical price development recommendation should find stocks that have similar price history. Price history should consider direction of trendlines and not magnitude as it is not reasonable to compare stock prices when the float of stocks count varies and market caps differ largely. However when considering the financial data algorithm should consider the magnitude of the values.

Updating the data with new values is outside of the scope of the thesis. Therefore, the data is collected from a point in time.

Given an input stock ticker, algorithm should produce an output list containing similar stocks based on their historical data and key figures. The aim is to produce a recommending system without suffering from cold start problem, so the algorithm won't consider how a user would rate the recommendations. Therefore the system will not be designed to learn from the user interaction with the system but instead act based on given input and give reasonable recommendations based on similarity and correlation. The implemented system does not contain any user interfaces. Instead the development is done using interactive computational environment with Python and Jupyter notebook in proof-of-concept manner. During development different approaches will be compared to each other by evaluating with machine

learning techniques and recommendation algorithm evaluation methods such as silhouette score for each clustering technique, elbow method for K-means and BIC and AIC evaluation for Gaussian mixture model. From a recommender algorithm's perspective on the other hand the evaluation is done based on diversity and coverage which are common metrics in recommender systems.

1.5 Overview

Chapter 2 lays a foundation for the theory behind the thesis. 2.1 describes the function, characteristics, main components and the types of different recommender systems. The second part, 2.2, of theory focuses on data mining and machine learning methods. What kind of techniques exist and what kind of use-cases they contain in the field of recommendation systems. Last three chapters of the theory focus on distinct areas related to this thesis. 2.3 explains the evaluation methods used in the implementation considering the specialty of the evaluation of recommender systems. 2.4 describes briefly different challenges related to knowledge-based recommenders. Chapter 2.5 gives an outline on the characteristics of clustering time-series data.
Chapter 3 contains the whole implementation for the thesis providing the description for evaluation methods used, data collection, preprocessing and the analysis conducted including the comparison of different approaches and techniques. The final implementation is put to test in chapter 4 by feeding different stock tickers into the algorithm and analyzing the recommendations with visual representation. In addition further improvements are described and explained. Lastly, chapter 5 concludes the thesis and describes the main points and the conclusions in terms of research questions.

2 Methodologies of transforming raw data into recommendations

2.1 Recommender systems

Recommendation systems are a subcategory of machine learning and data mining. Recommendation systems aim to provide content or suggestions based on existing knowledge. A simple example of such system is online marketplaces where the website may suggest user items that one might like based on user's existing items in the basket. From user's perspective this is useful as one does not need to know exactly what to search for when buying products. From the online marketplace's perspective on the other hand it is profitable if gets meaningful recommendations and therefore buys more items. Creating and managing recommender systems may lean into many different kinds of methods, knowledge, and skills from multiple areas. Artificial intelligence, human computer interaction, information technology, data mining, statistics, adaptive user interface, decision support systems, marketing, consumer behavior and many more. [7]

As in many machine learning processes, recommendation systems do not differ much when it comes to the importance of knowing the data and the specific domain. Recommendation algorithms differ largely based on the purpose and goal of the system. A recommender system can be as simple as using filters in a travel booking site in

order to filter the locations and hotels based on criteria user has set. Another example could be a streaming site where user can rate the movies after watching and the system is able to predict the ratings user could give to unseen movies. These predictions could then guide the system's recommendations.[5] The following chapters aim to give an outline of what kind of types and approaches are available for developing recommender systems.

2.1.1 User and Item

The key aspect of recommendation systems are items and users. Items may be any kind of content, videos, text, images, sounds etc. In short item refers to the suggested element. User on the other hand is the other side of recommendation systems receiving the recommendations. In some cases even user can be the item itself that is being recommended to users. For example social media sites where user may be recommended users to follow. However there are also other aspects for users in these systems.

Often items contain only limited amount of data and often this data is not meaningful enough in order to draw good recommendations. In this case users are a good way to power the recommendations to become more intuitive. For example if we know two users view the same items in a streaming service, we may assume they like same kind of movies and have similar taste in content. Now then again if a user A views a new item that the user B has not yet seen we could recommend the same movie to the user B.

In some systems users review the content they watch. This makes it even more accurate to recommend the movie to the user B if we know that the user A gave it five stars. [1] User-item model-based approach refers to a technique where user's past interaction with items is used to power the recommendations. For example user may prefer certain kind of style, size or colors and based on the items the user views

the website is able to provide similar items to the user that one has already viewed. As the system learns user's taste it becomes aware of what the user would like to view next. In this case system aims to provide strictly similar items. However this might not always be the case.

Let's suppose we have a music streaming service which adapts to user's taste based on the music one listens to. We may find similar items based on artist, music style etc. and provide content user would like to listen to next. The issue in this case is that if the system recommends continuously the same songs and bands the user might easily get annoyed by the recommendations. This brings us to variety of recommendations. As in some cases it can be possible that we do not only want similar items, but instead enough similar items that user may not have become familiar yet. In some cases this kind of user-item model-based approaches are not possible. For example when we consider an example where a company launches a new online marketplace and there is no user-item interaction history that could be used to develop such recommendation system. User-item interaction is a term that describes the historical data of how user has interacted with specific items. As for example which songs user has listened to, how they were rated and how long user spent listening to the songs. In this case we have to be able to develop a recommending system using existing data. For the online marketplace this could be the items that are available in the site. Item-item models represent a recommendation system where the aim is to find relationships between the items. Similar items are closer to each other, and it is probable that to user who views for example a pair of blue is interested in other kinds of blue pants also. [5]

2.1.2 Reactive and Proactive

Recommender systems can be either reactive or proactive. Reactive approach aims to view the recommendations as conversational process. Reactive systems expect

user to explicitly search and iterate with the process to get the recommendations one is searching for. This conversational process can be based on user's feedback into the system, or it can be based on rules or queries user inputs to the system. Reactive systems do not act before user is interacting with them. Proactive approaches on the other hand view the recommendations as a process that do not require user interaction with the system. User will get the recommendations whether they were being searched for or not. If an item does not match user's liking it can be simply disregarded and skipped. Proactive systems often suffer from cold start problems since they need large amount of historical interaction data in order to provide meaningful recommendations. [1]

2.1.3 Types

There are multiple kinds of purposes and technicalities to be considered when developing recommendation systems. Following chapters describe in more detail of how the recommendation systems can be categorized based on their approaches and methodologies. However it is important to notice that many systems can utilize multiple methodologies and some approaches may overlap with each other.

Knowledge-based

Knowledge-based recommendation systems aim to provide recommendations based on domain knowledge of item's features. In this case user-item interaction is not needed as the recommendations rely solely on the characteristics of items. These kinds of recommendation systems may be divided into two more subcategories, constraint- and case-based.

Constraint based filtering is one of the most common recommendation algorithms used. Constraint based is a quick solution and it provides understandable and easily interpreted results. User inputs the constraints of which items one wishes to

search for, and the algorithm limits the result set into the items that fill the given constraints. In this case the system does not need to learn the user's needs. Constraint based systems are often used in online marketplaces where user is able to select items only based on selected constraints. For example a traveling website may have a filtering option for location, vacation length or pricing and user will only get recommendations which fulfill these rules. [5]

Case-based filtering aims to provide more intuitive recommendations. In these algorithms the result is matched to users' needs based on for example similarity. If user seeks to find movies similar to Batman the system may rank the suggested items based on similarity for the given input. Case-based systems are often also interpreted to be a subcategory of content-based recommender systems as they both recommend items based on features of items. However, usually in content-based systems user-item ratings are taken into consideration when making recommendations. Case-based recommenders on the other hand provide recommendations in reactive manner. Distinct difference between case-based and constraint-based systems is that case based systems aim to find similarities between items whereas constraints match user's requirements with item features. [8]

Demographic

Demographic recommender systems aim to recommend content based on for example location, age, gender, language, or country. In simple terms, demographic attributes. These kinds of systems are often used in websites where content is provided based on language selected in the browser so that the visitor in the page will be shown content in one's own language. Another example is Netflix where a profile that is selected to be in the use of Kids will not be shown any content that is not suitable for people of younger age. [5]

Content-based

Content based systems are close to case-based systems and they overlap in terms of their implementation and use-cases. These systems provide users' recommendation based on items that user has liked in the past. As in case-based systems, the similarity between the items is calculated based on each item's features. Content-based systems are used for example in movie recommendation systems where user inputs one's favorite movies when first entering the streaming site. Recommendation system is then able to depict on which movies user could want to watch next. [5]

Collaborative

Collaborative filtering is an approach that brings users tastes into the equation. In collaborative filtering we do not have to have such close knowledge of item's features but rather we rely on users to know their taste by grouping similar users together. In simplest form users can be grouped by their previous liking. Of course in this case we must have some kind of data from users' interaction with the items. Collaborative filtering often therefore tends to suffer from cold start problem. In more accurate systems collaborative systems may also have a rating data where users have not only viewed specific items but also rated them. This gives more detailed view on how similar users are to each other when we are able to depict on how much they liked a specific item. If two users gave the same rating for multiple items, it is more probable that they like the same items in the future also compared to users whose ratings are the opposite for each item. [5]

Hybrid

Hybrid recommender systems are a combination of multiple techniques. A common approach is to combine content based with collaborative filtering. Users may be grouped based on their similarities in order to form clusters that have the same

ranking in the items they prefer. When recommending items for a user, the system may first find the group user belongs to and then see what items the other users have liked. The system may then order the items based on their features using content- based approach. In some cases hybrid systems may provide a more accurate recommendations compared to collaborative filtering and at the same time it can also be more scalable with large data sets compared to content-based systems. [5]

2.2 Data mining and machine learning in recommendation systems

As the amount of data has grown by time, people and companies have started to figure out on how to learn and benefit from it. Similarly to nature we can find patterns and rules in the data. They often are not easily seen, but when looked more deeply we may see things that not only explain a phenomenon but also helps us to develop something new. Data mining turns unprocessed data into meaningful information. This information can be used in marketing, sales, user personalization, understanding customers, finding trends, developing processes and many more. Data mining uses similar methods as machine learning, and they are closely related in terms of their approaches as they both are a subcategory of data science. However there are some differences. While data mining is used to find patterns and learn from the data, machine learning aims to teach the system to understand the data.[9] [10] Data mining and machine learning are commonly used for recommender systems. Depending on the approach chosen for the recommendation system we may benefit from both of them. In general, data mining is a sum of three different steps. Data preprocessing, data analysis and result interpretation. Considering recommendation systems the importance of dimensionality reduction and distance measures is extremely significant. [5]

2.2.1 Preprocessing

Preprocessing is the act of preparing data for an analysis. As in other machine learning tasks, also recommender systems rely on preprocessing before building the recommendation model. Preprocessing often contains basic operations such as transforming categorical values into numerical labels or one hot encoded binary value, removing nulls or unifying data formats. In addition to these basic operations common methods used in preprocessing include scaling, which broadly can be divided into standardization and normalization. Most commonly used method for scaling is to transform each feature having a zero mean and unit variance. This approach is called standardization.

$$\mathrm{X}' = \frac{\mathrm{X} - \mu}{\sigma} \quad [5]$$

Another option is to simply scale the values in a way that the minimum value is for example zero and maximum is 1, this then again is called min-max scaling or normalization.

$$\mathrm{X}' = \frac{\mathrm{X} - \mathrm{X}_{\mathrm{min}}}{\mathrm{X}_{\mathrm{max}} - \mathrm{X}_{\mathrm{min}}} \quad [5]$$

Depending on the case the more suitable one is used. However commonly the first approach works more appropriately as it handles outliers more efficiently. [10]

2.2.2 Similarity evaluation

Similarity and distance evaluation are key components of recommendation systems. Most common way to determine similarity is to calculate the geometrical distance of individual points in space. One of most used metrics is Euclidean distance which is defined as

$$\mathrm{d}_{\mathrm{E}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x}_{\mathrm{i}} - \mathrm{y}_{\mathrm{i}})^2}$$

[11]

In recommender systems similarity aims to determine either correlation or distance between items or users. Similarity and distance are often misinterpreted as one

metric but there are actually distinct differences between them. For example let's consider three square implements of height, width, and depth with values of A(10, 11, 12) and B(20, 20, 20) and C(30, 31, 32). Now that there are only few items, we can quickly detect that B and C are actually closer to each other compared to A and C since their properties are greater in magnitude. However if we forget the magnitude of these items and consider only their correlation, we can actually see that A and C are correlating more than A and B, since A and C:s values increment in each attribute whereas B is the same in each side of the square. Distance usually reflects the actual distance of two items by magnitude. Distance metrics do not account correlations between items but instead they provide information on how close the items are on another in geometrical terms. Common similarity metric used is cosine similarity which is defined as following.

$$\text{cosinesimilarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}.$$

Further cosine distance is defined as,

$$\text{cosine distance } = D_C(A, B) := 1 - S_C(A, B).$$

Cosine similarity ranges from -1 to 1 whereas cosine distance ranges only in positive space from 0 to 1. The difference between these two is that the similarity is based on the angle between two vectors, and this can therefore indicate either opposite vectors -1, independent vectors 0 or similar vectors 1. Then again cosine distance reflects the dissimilarity of given vectors. Considering that if the vectors are similar the similarity is 1 and therefore cosine distance is 1 – 1 = 0 which would describe no dissimilarities. Then again if the angle is 0 the dissimilarity would be 1, indicating strong dissimilarity. Important factor to notice here is that cosine distance should not be interpreted same way as Euclidean distance which determines the geometri-

cal distance. Cosine distance, despite the name, reflects the dissimilarity of given vectors.[5]

2.2.3 Dimensionality reduction

Dimensionality reduction is an unsupervised machine learning process where high-dimensional data is converted into lower-dimensions. Often data sets may contain thousands of features and the difficulty can be first how to visualize such data and secondly how to process such large amount of data efficiently. Developer could simply exclude some of the features from the visualization or from the analysis, but this could also cause the results to be inaccurate. There are multiple options on how to reduce the dimensions in the data. Each of them uses different methodologies in order to determine which components are the deciding factors. [12]. In some cases dimensionality reduction can be used in recommendations systems as the main component. [13]

Principal component analysis (PCA)

Principal component analysis, also known as PCA is one of the most common dimensionality reduction methods. PCA aims to find the path in the data where the points vary the most by maximizing variance. In other words, which components are the most dominating ones. In cases where data sets contain such large quantity of features that the dimensionality of the data is too large to be processed or visualized, PCA may help to reduce it into even 2 dimensions. PCA works by computing new components from the existing features. The first step is to write the N datapoints as vectors in a matrix X with size of N x M, where N determines the number of datapoints and M the number of features respectively. Secondly the matrix M is centered by subtracting the mean from each column and covariance matrix C is computed from the M with a formula of $C = C = \frac{1}{4}M^tM$. Then eigenvalues and

eigenvectors are computed from the C.$D = V^{-1}CV$, where V is the eigenvectors of C and D is diagonal eigenvalue matrix with a size of M x M. Lastly the columns of the eigenvalue matrix are sorted into decreasingly and same order is applied to the columns of V. Lastly values that have eigenvalue smaller than n are excluded and the dimensionality is therefore reduced. [10]

Multidimensional scaling (MDS)

Multidimensional scaling is close to PCA. Similarly to PCA, MDS aims to find the linear approximation from the data that is used to reflect the data into lower dimension. MDS tries to preserve the actual distances of the data. The closer neighbors are not prioritized but instead the distances are kept as they are. MDS seeks to minimize the cost function that is used. Common cost functions are for example Kruskal-Shephard scaling and Sammon mapping. The algorithm uses following steps. First the pairwise similarities are computed and stored in a matrix D. Secondly, compute $J = I_N - 1/N$, where I_N is the N x N identity function and N is the number of datapoints. Next compute $B = -\frac{1}{2}JDJ^t$ and find the L largest eigenvalues of B with the corresponding eigenvectors. Then, place the eigenvalues into a diagonal matrix V and set the eigenvectors to be columns of matrix P. Finally it is computed that $X = PV^{1/2}$. [10]

T-distributed stochastic neighbor embedding (t-SNE)

T-distributed stochastic neighbor embedding is a nonlinear dimensionality reduction method that represents the similarity of two N-dimensional points x_i and x_i with a conditional probability p(i, j). T-SNE enables the possibility to convert high-dimensional data into two- or three-dimensional form. Unlike MDS t-SNE tries to preserve the closest neighbors such that they are prioritized. This is also called neighborhood is preservation. Given a D-dimensional data set $X \epsilon R^D$, t-SNE generates a

lower dimension $Y \epsilon R^d$ where if x_i and x_j are close to one another in the input space X, then their resulted lower embedding points y_i and y_j are also close. [14] When a data set is converted into lower dimension, t-SNE first calculates the distances between each datapoint i and its neighbors j. The metric to calculate the similarity can be for example cosine distance, Euclidean distance, or Minkowski. These distances can also be referred to as a distance matrix, which describes the distances between each pair. This distance matrix is then represented as normal distribution which is converted to probabilities. The higher the probability is, the closer an item i is to item j. The next phase is to generate gaussian distribution with mean at x_i and variance ϵ_i Variances start from the selected perplexity. $Perplexity\,(P) = 2^{H(P)}$

A low value of perplexity would indicate a low uncertainty. Closer neighbors yield to higher probabilities. T-SNE conditional probabilities are defined as following.

$$p\left(\bar{x}_j \mid \bar{x}_i\right) = \frac{e^{-\frac{\|\bar{x}_i - \bar{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|\bar{x}_i - \bar{x}_k\|^2}{2\sigma_i^2}}}$$

These probabilities are usually then symmetrized in order to obtain joint probabilities.

$p\left(\bar{x}_j \mid \bar{x}_i\right) = \frac{p(\bar{x}_i|\bar{x}_j) + p(\bar{x}_j|\bar{x}_i)}{2N}$ [15]

2.2.4 Nearest neighbors

Nearest neighbors can be used for unsupervised and supervised learning tasks. The idea behind nearest neighbors is to find a predefined number of candidates that are closest in terms of distance or similarity based on a given point. The metrics mentioned earlier ,cosine and Euclidean distance, are common methods to determine the neighborhood for a given point. Nearest neighbor is one of the simplest forms of machine learning methods and it has many use cases for example in classification

tasks and recommendation systems. [11]. In top-N recommenders nearest neighbors approach is a common way to provide sorted results. [16]. Nearest neighbors can also be utilized alongside with clustering in order to implement a recommender system by first categorizing the items and then sorting them afterwards in smaller subsets. [17]

2.2.5 Clustering and evaluation

K-means

K-means is a partitioning based clustering method which is one of the most common clustering algorithms. The principle in it is that it aims to partition a data set of N items into k number of subsets S_j, called clusters. These subsets contain N_j items that are as close to each other. Each cluster has a centroid from which other items have surrounded. Choosing this centroid for each cluster is an iterative process that aims to find a point where the distance to other items in the same cluster is as minimal as possible. After this process the clusters should contain only items that are close to the centroid of the cluster. The process can be defined as following, where the value of E is aimed to minimize, x_n is a vector λ_j centroid of the item in S_j and d is a distance measure used. Commonly d is Euclidean distance.

$$E = \sum_1^k \sum_{n \in S_j} d\left(x_n, \lambda_j\right)$$

K-means relies on the fact that the value of K must be predetermined before running the algorithm. K-means itself therefore does not conclude how many clusters there should be. Instead it uses the predetermined value of K and tries to find the centroids for each cluster. The iteration process continues until there is no longer items that switch their cluster. In some cases determining the number of K, may become an issue. Other problem K-means often phases is outliers. [5]

Elbow method is commonly used in K-means to find the optimal number of clusters. The theory behind it relies on withing-cluster sum of square (WCSS).WCSS is a term that refers to the sum of the squared distance between centroid of the cluster and each point in the cluster. The idea is to plot the WCSS values at each value of K. This should then form an elbow-shaped plot. From this plot it can be then deduced, at the point of the elbow curve, the optimal number of clusters.Elbow method is a great tool for determining and giving an outline of mathematical representation of the number of optimal K. However in some data sets elbow method might not provide the most optimal scores in cases where the clusters are relatively close to one another. Since in this case the method might try to optimize the shortness of distance between the clusters by placing the centroid between both of them. [18]

Hierarchical clustering

As K-means, also hierarchical clustering belongs to the category of distance-based clustering methods, which are suitable for many datatypes. Hierarchical clustering is an algorithm which represents the clusters as a dendrogram. These clusters can be created using either agglomerative or divisive approach. The difference between them is how the clusters are formed. Agglomerative clustering uses bottom-up approach which forms a tree-like structure by merging clusters. How the clusters are merged depends on the linkage type selected. Possible options are single-linkage, complete linkage, average linkage, and ward linkage. Single linkage works by using a pair of points that have the shortest distance from each other. Average linkage uses the average distance between all pairs, complete linkage uses the maximum distances between all points. Ward linkage then again aims to minimize the variance when merging clusters. On the contrary to agglomerative, divisive works by using top-down approach. Similarly to agglomerative, divisive also tries to form the same tree-like structure. Divisive partitioning. [19]

Gaussian mixture model

Gaussian mixture model is a probabilistic clustering technique that is generated through three main components. A mean μ, covariance Σ and a mixing probability π. Suppose X describes data points and D is the number of dimensions of each data point X. In order to find the optimal values for mean, covariance and mixing probability the following formula is used.

$$\mathcal{N}(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right) \quad [20]$$

The formula can be solved into any of three components and the solution follows Maximum Likelihood Estimates (MLE).

Bayesian information criteria (BIC) and Akaike information criteria (AIC) are evaluation techniques used in Gaussian mixture model to find the optimal number of clusters. Bayesian information criteria follows the principle of maximum likelihood and it is often performing well for simple models. For using BIC, the value of the formula BIC = -2 * LL + log(N) * k is recorded at each value of K. The minimum point from the curve is considered as the optimal number of clusters.[21]

Similarly to BIC, Akaike's information criteria finds the optimal value of clusters at the point where the value of AIC is the minimum. Compared to BIC, AIC is often more suitable for complex models. The formula for AIC is the following. AIC = -2/N * LL + 2 * k/N. Variables are namely same as in BIC [21]

Silhoutte coefficient

Silhouette coefficient is a metric that can be used to evaluate the "goodness" of formed clusters by determining how well apart and distinguishable the clusters are.

The value of the score is calculated by considering both the intra- and inter-cluster distances. Silhouette score can be used for example in probabilistic, centroid and hierachical clustering techniques. The formula starts by calculating the average from all of the distances inside the same cluster for a point x_i. The average is called a_i. After this, for each cluster that does not contain the point x_i the average distance of x_i to other data points in other clusters are calculated. Considering both these values, the average silhouette coefficient of all points can be calculated.

$$S = \frac{\sum_{i=1}^{N} \frac{b_i - a_i}{max(a_i, b_i)}}{N} \quad [19]$$

2.3 Evaluation of knowledge-based recommender systems

Evaluating knowledge-based recommender systems differs and is strongly depended on the architecture of the system. Compared to a collaborative system where we have user-item interaction data that can be used to the development, in knowledge-based systems that possibility is not availabl as referred to in the last chapter. Therefore the development focuses on the item features itself. Experimental evaluation of recommender systems can follow the basic guidelines which are the following. First, form the hypothesis of what is expected from the evaluation. Secondly, keep the variables fixed in the evaluation. Algorithms and outcomes cannot be compared if the test sets differ, or parameters vary. Thirdly it is important to notice how much and in which ways we can actually generalize our conclusions from the evaluation to actual production set algorithm. If data and properties grow or it becomes more mixed with various data types, can we still trust that the results from the evaluation still apply. [5] Recommender systems can be evaluated from many different

perspectives and often there is no clear answer on which metrics should be used for every system. However there are multiple options to evaluate the recommending algorithm even in systems where only item data is available.

Coverage

First one of them being the coverage of the recommendations. Coverage is the range of items that can be recommended to a user with a given system. For example if with a given input to a recommender algorithm 10 items can be recommended the coverage is then counted simply as 10. Another option is to use a percentage value as the coverage. In this case the amount of items that can be recommended is compared to the number of total items. [5]

Diversity

As stated, coverage is a the length of a list that can be recommended based on the input. Depending on the system the amount of diversity in the recommendations can be evaluated and adjusted. Diversity describes how big of a variety a set of recommended items have. Suppose if a music streaming platform only suggested similar items that a user is already familiar with, one would get fed up with the recommendations easily. However in some cases, as in the aim of this thesis, it is not meaningful to provide diverse set of items when the target is to find only the most similar ones. When diversity is increased it is done in the cost of accuracy. Diversity is a similar factor to silhouette score but instead of computing intra- and inter cluster distances the focus is on determining the distances inside the groups. The diversity of a system can be determined as the average difference of items in the coverage list. Suppose d(i,j) is dissimilarity of items defined as one minus similarity of i and j. I is set of all elements and N is the length of the item list that can be recommended. U is a set of all users. The same method can be also used when items

are being recommended based on another item as input. Since distance matrices are used, the greater the result from the formula is, the greater the diversity is.
The actual similarity value naturally depends on the metric used. For example if the diversity for a given data set approaches closer to a number 1 when cosine distance is used, it can be concluded that the set is diverse. On the contrary if the diversity is zero it would indicate that there is only one item in the list or the items in the list are identical. Then again if Euclidean distance is used the diversity can be larger since the value represents the actual distance rather than an angle. However the same principle applies when approaching zero.

$D(L\,(u)) = \frac{1}{N(N-1)} \sum_{i \in R} \sum_{i \in R,\ i\ \neq j}^{n} d(i,j)$

For calculating diversity of a list compared to all the items, it can be done with z-scores calculation defined as following. Referred as z-diversity where D(I) is the diversity of all elements and SD(I) is the standard deviation of all item differences in I.

$Z(D(L\,(u)) = \frac{D(L(u)-D(I)}{SD(I)}$

[7]

Accuracy

Accuracy describes how good the recommended items are compared to a for example given input or user's preferences. Commonly in content-based recommender systems where user rates the items, the accuracy is determined by comparing the algorithm's prediction to the actual results of how user liked the item. If user rated the recommendation with good scores, naturally the accuracy is good. In knowledge-based systems this is a bit more problematic since the user-item interaction is limited and the algorithm assumes that user can just disregard the recommendations if they were not suitable. So there is not rating system to be used for evaluation. However we still can measure the accuracy of knowledge-based systems with a bit different

approach. It can be aimed for example to calculate the accuracy based on the similarity or dissimilarity of the recommended items compared to the input of the algorithm. In other words, how well the recommendations match the item that is used to search similar items or constraints that are set to filter the items. Common methods determining similarity are cosine and Euclidean distance. Naturally, accuracy can be also tested visually with for example plots and graphs to determine if the results match the expectations. [5]

Transparency

Transparency is a factor that describes how well the system is able to tell why a specific recommendations were made. In the case of constraint-based filtering, this can be viewed as showing the user the parameters which were used to filter the recommended items. Depending on the system and its design it can be difficult to interpret why a specific item is recommended when the feature set is large and the itemset contains mixed datatypes. Also often a resulted model from a machine learning algorithm can be difficult to analyze. Transparency is strongly depended on the combination of system's technical approaches and the ability to describe or visualize the recommendations. A case-based type of recommender could explain its reasons for the recommendations for example with a sentence as, "The item was recommended because you said you own item X". Transparency is often evaluated with surveys. Users can be asked if they understand the reasons behind the recommendations. [5]

Scalability and adaptivity

Taking system's technical capabilities into consideration is also a key factor for an efficient recommender. Some approach might work excellently with only few thousand rows but when the amount of data is multiplied with ten or even more the

system is not able to handle the load. Can the system handle the data and still provide meaningful recommendations or will it become slow and suggest items not matching to user's preferences. When these factors are considered the terms scalability and adaptivity are often evaluated. Scalability aims to answer the question of how the system would be able to expand if new rows were added to the list of items that can be recommended. A scalable system is able to quickly adapt to a new data without costly calculations. In many cases the amount of data processed through the recommender system can be massive and therefore the system has to be able to handle new items without training the model again from the start or finding the similarities between each item. Adaptivity is strongly linked to scalability, but it focuses more on how good the recommendations are even if new items were to be added. Does the newly added item confuse the recommendations or does it seamlessly adapt and find its closest pairs in the item set. [5]

2.4 Challenges of knowledge-based recommender systems

Recommender systems suffer from many same challenges as other software development processes. This chapter aims to provide an overview on some of the common issues and how they result in factors that can be evaluated when deciding the final implementation of the recommender system. Probably most common issue mentioned in relation to this topic is cold start problem. Often when starting a new system which requires a recommender algorithm to engage with the user before making any recommendations, developers will find themselves troubling with the lack of data. Depending on the type of the recommender, data fitting for that purpose is needed. Collaborative systems commonly need the user-item interaction data often including ratings and user feedback and content- and knowledge-based systems on

the other hand need broad knowledge of item features. Collection and storing item data can be costly and updating data requires specific pipelines to handle the work. Also data manipulation and validation require precise processing in order to keep the recommendations accurate and useful. On the other hand compared to lack of data also information overload can be faced. What information is actually meaningful for a specific recommendation algorithm to be developed. [7]

Another issue which is related to privacy and trust is managing the data itself when it has been collected. Especially collaborative and hybrid systems require user data to be clustered in order to recommend content based on similar users. Even though the user data strictly is not shared to other users, it is being used to train the models. In knowledge based or content-based recommenders this issue is not usually present since the recommendations are made strictly based on the item features. However in some cases the user may act as the item if the purpose is to recommend users to a user. In this case again the issue is the sharing of user data when training the model. Also when considering stock data which may contain specific details and names like company directors, we again face the issue of privacy and trust. [5]

Evaluating unlabeled data is a common problem faced in especially clustering based systems. This same issue applies also to recommender systems. In supervised models, the data on hand can be used to evaluated the performance of a developed model. However in unsupervised techniques, the lack of labeled data makes it difficult to evaluate what is actually correct. Given that if for example item-item based knowledge- or content based recommender system was to be developed with clustering techniques, the concept of an optimal model can be difficult to determine if there is no actual data to compare the model with. [22]

2.5 Similarity and clustering of time-series data

Time series data can be encountered in multiple places, such as stock market data, sensor data and machine state monitoring. Time series data is a continuous series of data.In financial markets time-series data can for example represent a series of stock prices. In time-series data it is important to understand the type of the underlying data. It is a measurement at a given time and it can be visualized in many different forms. Clustering time-series data is a common approach in finding similarities or correlations in a high dimensional space. The complexity of time-series data is commonly large in size as there can be for example sensor measurements taken each second. The dimensionality is therefore also often high and the continuous flow of data creates an environment where a clustering model generated from time-series becomes quickly outdated. [23]. Another aspect of time-series data is the fact that the objects may or may not be the same length. How to determine similarity in a data where some of the time-series can be half the length some other data point is. This can be faced especially in stock market data where new time-series data emerges when companies are listed to the market, new indexes are invented, or new funds are started. Each of these have different length of time-series data. Naturally one could only use the time-series of the shortest data point but in this case, the data would not describe the same events it did previously. Common approach to this issue is to unify the lengths of the time-series. [24]

Attributes

Broadly time-series data can be described with two attributes. First one being contextual attribute, which describes a timestamp or time value when a specific behavioral attribute is measured. The second one being the behavioral attribute which represent for example a sensor measurement or value of a stock price at a given time. [19]

Clustering

The clustering approaches for time-series data vary depending on the data type. The main formulations for these can be defined as correlation based online clustering and shape-based off-line clustering. Correlation based online clustering is a type of process where time-series data are clustered based on the data's correlations. Again this is typical in financial markets. These types of clustering formulations can be used to find similar trends. Shape-based off-line clustering on the other hand relies on clustering time-series data based on similar shapes that exist in the data. In this case it is necessary to create a similarity function which determines the similarity of a shape. [19] Clustering of time-series has been experimented in many different applications and with many techniques. For example K-means, fuzzy c-means, agglomerative hierachical clustering and self organizing maps. [25]

Concept of similarity

The challenge in clustering or finding similarities in time-series data is a task of defining similarity across a continuous time. The shapes the time-series data can form may effect on what kind of metrics or algorithms should be used. The main task is therefore to define the concept of similarity. Types of time-series data varies from application and use case. A measurement or a value in time-series data is not always a single value or a data point and determining the similarity between two sets of time-series data depends on their formulation. One approach in determining the concept of similarity in time-series is to define three segments, similarity in time, similarity in shape and similarity in change.

Similarity in time measures the resemble of two data points at a given time in terms of their distance or similarity. Such data is common for example in temperature measurements or stock price movements.

Similarity in shape on the other and measures the similarity of given points in shape

form regardless of the specific values in magnitude. For example if at given time there there is three recordings, two circles and one square. Despite of their magnitude, naturally the circles are more similar with each other than the square and a circle.

Similarity in change measures the similarity of time-series data in auto correlation structure. [26] Auto correlation structure is a term that is often used in stock trading algorithms. It is often also called lagged correlation or serial correlation. Unlike similarity in time, auto correlation measures the similarity of a time-series data to its previous past values. For example when finding trades an algorithm could find a pattern that is repeated in the past and which is occurring again. [27]

The key component in similarity and clustering of time-series data is the similarity metric. As the data in time-series comes in comes in many forms such as raw values, equal or unequal in length, vectors, feature-values pairs or transition matrices the similarity metric used effects greatly to the outcome. Common similarity metrics that are used in time-series data include Pearson's correlation coefficient, Euclidean distance, Minkowski distance and Cosine distance. [25]

3 Implementation of knowledge-based recommender system for stocks

This chapter describes the implementation of the knowledge-based stock recommender algorithm. 3.1 explains what kind of measures were used to evaluate different approaches and what kind of factors affected choosing of the best one of them. 3.2 explains the steps for data collection and 3.3 describes the data sets and what kind of features they contain. Chapter 3.4 goes through the preprocessing steps that were conducted on the collected data.

Chapter 3.5 refers to 2.5 chapter and provides an implementation for calculating the similarity of stock trendlines.

Before experimenting with different clustering techniques, both data sets were analyzed and evaluated in 3.6 as a whole in order to describe the characteristics of the data and create a baseline for clustering. Clustering techniques were examined in 3.7 and 3.8 with two different approaches. First approach relied on creating distance matrices for financial data set and trendline data set separately and then combining them to achieve a distance matrix that contained both feature sets. This distance matrix was then able to be clustered with the three selected techniques. Second approach used 3 steps to cluster the data. First financial data and trendline data were clustered separately with techniques that provided most promising results for each. Then the resulted labels from each clustering process was combined into a

new dataframe. This new dataframe was then clustered in order to form a final set of labels that represent both data sets equally.
Finally 3.9 presents the conclusions derived from the experimental implementation. As a last step, 3.10 provides an implementation of a sort function to order the recommendation lists based on similarity in descending order when recommendations are queried based on a single stock.

3.1 Evaluation factors

For comparing the clustering techniques between each other diversity, coverage and quality of clusters were used as evaluation factors. These evaluation factors were plotted at each value of K clusters. Diversity and coverage describe the function of the algorithm in from recommender system's perspective. Quality of clusters on the other hand refers to more common clustering evaluation methods.

3.1.1 Diversity

Often in recommendation systems the list of items to be recommended is aimed to contain some diversity. However in this case the aim is to strictly minimize the diversity of the recommendations and use only closest matches for recommendations. The minimal diversity is always reached when a cluster reaches size of one item. Therefore it requires balancing between coverage and diversity to achieve tight clusters which contain only similar stocks but still that still contain enough coverage. Diversity is closely also related to silhouette coefficient as both evaluate the closeness of points inside a cluster. However diversity considers only the items inside the same recommendation list, or cluster. Diversity as an evaluation metric is based on the theory set in chapter 2.3. In order to determine how the diversity of trendlines

and financial data changes when number of clusters K is increased, average diversity per cluster was measured at each value of K. This value is called diversity in the chapters. In addition the range of similarity on average was measured by taking the average of furthest points inside a cluster. This value is referred to as max diversity. The data for the algorithm is divided into two sets, trendlines and financial data. The types of both of these are completely different, but still both of them should be included in the clustering process and their effect should be visible. Therefore both sets are evaluated separately when determining the diversity of the clusters.

3.1.2 Coverage

As stated previously the minimal diversity is achieved when there is only one item in the cluster, or alternatively if the cluster contains only exact same item in duplicated form. Considering the data set in this thesis the stocks are unique items and therefore the minimum would be therefore achieved when each cluster contains only one item. However given the aim of this thesis that the similar stocks should be categorized into the same cluster and stocks can be recommended only from the same cluster which a specific stock belongs to, it becomes an issue if the outliers in the data form clusters containing for example only one item. Considering only diversity, the implementation would emphasize the approaches and techniques in the implementation that are prone to outliers. To recognize these approaches which results in outliers forming clusters the sizes of clusters is recorded. ´ Now if only diversity was optimized in the implementation, each cluster would contain only one item. Therefore the coverage is paid attention to in order to find a breaking point where diversity in each cluster is low, but not as low that each cluster would only contain one item. To achieve this the standard deviation of the clusters sizes was also measured. For the purpose of this implementation, it is important that the

clusters form recommendation lists that have a number of stocks that is reasonable for recommendations. The standard deviation between cluster sizes is recorded including the number of clusters where number of stocks is either below 10 or above 150. This is meant to give an outline of how the clusters are formed and would there be a possibility to use them for recommendations. As the goal is to able to provide recommendations for any stock.

3.1.3 Quality of clusters

For each technique used, silhouette score was used to evaluate the clusters at each number of K clusters. In addition to the silhouette score, other methods were used for specific techniques. When applying K-means elbow method was used. For Gaussian mixture model Bayesian information criteria and Akaike's information criteria were used.

3.1.4 Other considerations

In the testing phase the results are visualized and ordered in order to make the recommended results transparent. The visualizations should reflect the similarity of the given recommendations. Visual representation should describe and explain why the recommended stocks are similar to a given input. For example the trendlines of the stocks should visually correlate and financial data contain similar values in terms of magnitude.

3.2 Data collection

The implementation process was started with data collection. The aim was to collect US stocks from NYSE and Nasdaq marketplaces. In order to get reasonable results and enough data for the development, it required at least a thousand of unique

stocks including their financial data and historical price development. Financial and trendline data for stocks is very different from one on each other's. Whereas financial data is more concerned on ratios and single values, trendlines are series of continuous data points. There is no single access point to collect such data at once. Therefore the data sets were collected in three separate steps:

1. Collecting stock symbols, names, and descriptive data
2. Collecting financial data for each stock
3. Collecting trendline data for each stock

The first step was executed using eodhistoricaldata.com that provided an API for collecting stock symbols for a specific exchange in csv format. This data also included country, name, currency, and other descriptive data. At this point a total of 8349 tickers were found from Nasdaq and NYSE markets which were the targeted markets for the development. Steps 2 and 3 required a bit of programming as the data had to be collected for each stock separately. The source of the data was switched to yfinance python library for finding financial data and historical price development for each stock. In step 2 financial data was fetched for each stock by iterating the csv file. As a result a new csv file was created containing both descriptive information and financial data. Third step was to fetch the trendlines for stocks which were collected into a new file outside of the financial data. As stated in the scope of the thesis, the historical data was limited to a five-year time period in order to limit the amount of data. Or more precisely 5 years and one month. Without the limit there could have been a possibility to end up with files containing over 20 years of data points for thousands of companies. For development purposes this is not reasonable. In addition, the data was fetched using month as an interval to limit the number of rows furthermore. This means that for example a stock that has been listed for a 5-year period the total number of data points was 60. Naturally the more there would be data points the more accurate the trendlines would be.

Trendline data was collected in batches without grouping the trendlines by a stock symbol. This way all of the historical data was able to be put into a single file. Each row in the file contained date, symbol, open, high, low, and close. Using the month interval, for total of 8349 stocks, the count of historical price data points ended up to 306059. So, on average this would count as 36.66 data points, or approximately 3 years per stock. The reason for why all of the stocks did not have the same amount of data points is that they have been listed to the market at different times. Some may have been in the market for decades whereas some only few years.
In the end of the data collection phase the whole data set consisted of two separate csv files. One containing stock tickers, names and financials and the other containing the trendline data.

3.3 Overview of the data sets

3.3.1 Financial data set

operatingCashFlow	grossProfits	totalCash	totalDebt	totalRevenue
Float	Float	Float	Float	Float

Table 3.1: Financial data set, financial values

Ticker	quoteType	shortName
String	Float	Float

Table 3.2: Financial data set, descriptive data

Any strict expectations on which kind of financial data would be collected were not made beforehand as there was not any certainty as to what would be available in suitable format. Instead the main goal was to collect reliable and consistent data. There would have been also a possibility to collect also financial ratios such as ebitda

or price to earnings but in this case, it was decided to focus onto core values of the stocks which would enable the development process to focus more on the algorithm itself rather than optimizing the feature set. However the included features were aimed to describe different aspects from the company's performance. Therefore the following six features were included into the financial data set.

Operating cash flow (OCF)

Operating cash flow describes the inflow and outflow of cash considering the main operations of a company. It is one of the main indicators for determining how profitable company's core business is. OCF is the sum of net income and depreciation, depletion, and amortization. [28]

Gross profit

Gross profit is the value that is left after reducing the cost of goods. [2]

Total cash

The total amount of capital company has in cash. The more cash a company has the more capable it is to pay for example supplier costs, dividends, debt or labor.

Total debt

Total debt describes the sum of short- and long-term liabilities.

Total revenue

The amount of income that the company is able to generate through its services or sale activities. Total revenue however does not tell how much profit a company is able to produce.[29]

Descriptive data

In table 3.2 the descriptive data included in the financial dataset are listed. Quote type is a a categorical column that describes the type of the investing item. In this data set it can be either fund, equity or mutual fund. Short name is the name of the company whereas Ticker is a stock's identification in the market.

3.3.2 Trendline data set

Date	Ticker	Close	High	Low	Open	Volume
String	Float	Float	Float	Float	Float	Int

Table 3.3: Trendline data set

Trendline -data set contains seven columns from which only three are used in this thesis. The main features are the following. The first column describes the date that the numerical values are corresponding to and it acts as a contextual attribute. Numerical values on the other hand are behavioral attributes. Lastly ticker is a descriptive feature which describes to which stock a specific data point belongs to. The prices of the stocks are automatically split adjusted in the data set so there is no need to consider the splits of a company when determining historical price development.

3.4 Preprocessing and preparing the data

Load the csv files into dataframes

Using panda library both data sets were imported from CSV-files into financial and trendlines dataframes named respectively.

Exclude null values and other than common stocks

First other than common stock type investing instruments were removed from the data set to match the original scope. Profile data of stocks contained quoteType column that was used to find only the rows where type was 'EQUITY'. Stocks are often referred as equities. At this point it was however noticed that the data still consisted of warrants and acquisition companies. These could not be included in the data for following reasons. Warrants are options and do not act as normal shares as they are not actually normal stocks but rights to buy the stock at a certain price. Acquisition companies, also known with a name "spac", on the other hand do not contain basic financial data as they are blank check companies that are still finding company to merge with. So, based on acquisition companies or warrants companies it could be an issue to find similar companies based on financial data. To remove these types, it was noted that if the column "shortName", consisted of the word acquisition company or warrant it was needed to remove from the data. Filtering other than common stocks resulted stock count to reduce from 8349 to 5850. After this the actual columns in the data set were looked more closely. When it comes to stock data it is difficult to gather reliably data that consists all of the values wanted. For example if a company has been publicly trading under a year, they have not yet published annual report or possibly not even 10-K report. So, it can be difficult to find correlations and recommendations based on these companies. When looking at different columns and rows it was noticeable that some of the data contained null values in the fields that were needed later on. Stocks that had null values in financial data or that had historical data under a year were therefore also removed from the data set. This resulted the stocks to reduce from 5850 to 3272.

Group trendline data by symbol ticker

Data in the trendlines csv file was originally non-ordered. Since the historical data was aimed to be compared to every other stock in the data set, it was easier to handle it when all of the datapoints for a single stock were grouped. This was done using panda's builtin group by functionality. This enabled fetch any stock's historical prices from a single dataframe row by index. Data is then more easily accessible, and it can be transformed and processed more easily. Grouping values also ordered the stocks alphabetically based on ticker, so also the financial dataframe was ordered in the same order with sort functionality. Now data for any company was accessible with same index in both data sets.

Standardize and scale

The columns in financial dataframe differ distinctively by their scales. For example revenue can be billions and total cash on the other hand is only a small percentage of this. If these were to be clustered as they are, revenue would end up dominating the clustering process. Solution for this is to scale the data using standard scaling.

3.5 Utility function for calculating time-series similarity

As stated in the chapter 2.5. Finding similarities and clustering time-series data is depended on the similarity function. Sometimes a basic metric such as Euclidean distance is appropriate but in this case the correlation of the time-series data is the focus rather than the distance from one array to another.

Uniform the lengths of trendlines

Array sizes for each stock's historical price depends on the day the stock has been listed into the market. When calculating cosine distance the aim is to compare even those stocks that have been in the market different time lengths. The maximum length for the array is 61 data points or 5-year and 1 month time period that was the original limit set to the collection of trendline datapoints. This means that if there exists an array that has a length of below 61, the missing values are filled with with zeros. Those values however are not added to the end of the array but in the beginning in order to match the timeframe for all stocks that have been listed later. In the end of this step all the trendline arrays were the same length of 61. In addition the timeframes of each were uniformed so that the price of any stock at a given data was able to be fetched using a dataframe index. In short, those stocks that were listed in the market later than five years ago, were filled with zeros until the listing date. 1 shows pseudocode of filling zeros for shorter trendlines in a set T.

Listing 1 Unify the length of stock trendlines T

```
for i in T do
    addzeroscount = 61 - length(i)
    zerosarray = [ ]
    for x = 0 To addzeroscount do
        zerosarray ← 0
    end for
    i = zerosarray + i
end for
```

Computing the similarity of trendlines

When comparing historical price charts between stocks, it is important to remove the effect of magnitude as only the correlation of the trendlines during a period

of time should be considered. Using Euclidean distance would not be suitable for achieving this as it focuses distance rather than correlation. Cosine distance on the other hand fits into this purpose. However default cosine function from for example scipy cdist would not work since the trendlines would end up be weighed differently because of the fact that they are listed at different times. This is the issue of calculating the similarity in symmetric manner such as (i.e., sim(i, j) = sim(j, i)) and not (i.e., sim(i, j) != sim(j, i)).

Suppose there is items i and j that have different length of vectors and their similarity is to be calculated. In case their similarity was to be measured symmetrically that would mean that the similarity of vector i to j is same as from j to i. However if their difference in length is considered the results will be much different. Suppose length of i was much longer than j. From the perspective of item i the similarity compared to j would not be that high as only a part of the item i can match with the item j. On the opposite side if item j is compared to item i, their similarity would be higher because of the fact that large portion of j can match with item i. [16].

In the case of stock trendlines this can be thought similarly. For example considering a stock i with full five years data points of 61 and stock j with 13 data points. If the similarity of the trendlines is calculated from stock i to stock j, from i's perspective the trendlines are completely different for the first four years. After that the similarity is higher as the stock j has been listed to the market and the prices for stock j are available. The first four years for stock j are interpreted as zero values as implemented in 3.5. Now then again, if the trendlines are compared from stock j to stock i only last year is considered in order to emphasize that from stock j's perspective it is possible for only a part of the trendlines to correlate. To achieve the correct weighing for the trendlines a custom function for cosine distace

was created.

Pseudocode 2 shows how the distance matrix for the trendlines is generated using the zero filled array and then emphasizing the correlation from i to j so that (i.e., sim(i, j) != sim(j, i)). This is achieved by matching the timeframes of the trendlines from i to j. At row 2 the zeros are removed from the trendline i but kept in j. If the lenght of i is then shorter than the trendline j, j's trendline is matched accordingly to the timeframe of i. For each trendline i the correlation to a trendline j is computed by calculating the dot product of i and j and dividing it by the multiplication of the vector norms i and j. Finally this value is subtracted from 1 to transform cosine similarity into cosine distance. Variable D describes the distance matrix in which the values are appended to.

Listing 2 Computing distance matrix D for stock trendlines T

```
1: for i in T do
2:     i = i[i! = 0]
3:     for j in T do
4:         if length(i) < length(j) then
5:             j = j[length(j) − length(i) :]
6:         end if
7:         D ← 1 − dotproduct(i, j)/(vectornorm(i) ∗ vectornorm(j))
8:     end for
9: end for
```

3.6 Pre-analysis of data sets

Before moving onto implementation phase, the diversity of the whole data set was evaluated from both perspectives, trendlines and financial data. Later these values can be compared to the diversity values of the recommendation lists. Naturally the lists should have lower amount of diversity as they contain lower amount of

stocks and as targeted, only similar stocks. For trendlines the similarity matrix is calculated using the cosine distance function created in the previous chapter. This function is used in cdist function from scipy as a custom metric. Cdist is also used for calculating the distance matrix for financial data set using Euclidean distance.

Dataframe	Distance measure	Average Distance	Minimum Distance	Maximum Distance	Average minimum distance	Average maximum distance
Trendlines	Cosine	0.15	1.553e-05	0.99	0.0122	0.7756
Financials	Euclidean	2.23	0.0	79.69	0.0549	79.27

Table 3.4: Pre-analysis of dataframes

In the table 3.4 the distances among the whole data set are listed. In theory the cosine distance ranges between 0 to 1. In this case, it varies from 1.553e-05 to 0.99. Euclidean on the other hand describes the actual geometrical difference of two items which in this case varies from 0 to 79.66 when the values have been scaled using standard scaling.

The trendline minimum diversity pair value of 1.553e-05 is very close to zero. But when looked more closely this is no coincidence or a mistake since the pair of stocks is in this case actually Berkshire Hathaway class A and Berkshire Hathaway Class B stocks. Sometimes stocks can be listed twice with different properties such as voting rights. These kind of 2 share classes represent the same company and therefore their trendlines correlate. Also the minimum distance in stock financials seems odd at first glance when the distance states that the financials are actually exactly the same. However also in this case the closely related stocks from financial values perspective are simply America Movil SAB class A and B stocks which have the same values in their data.

In addition to actual minimum and maximum differences found in the data sets, the

average values for both are listed. Average minimum and maximum measures are calculated by taking the closest or furthest pair of each stock and calculating the average from all of their differences. Average distance on the other hand represents the average difference of all stocks in the data set. Next it was tested, what happens when the stocks are separated into smaller groups with number of K, randomly without paying attention to their similarity. This is merely to visualize on how the effect of splitting the stocks into groups effects the diversity. As whether the stocks are similar in the groups or not, the diversity decreases as the number of stocks in a group decreases. Therefore it is important that the algorithm can reduce the diversity much more sharply compared to the assignment of groups randomly.

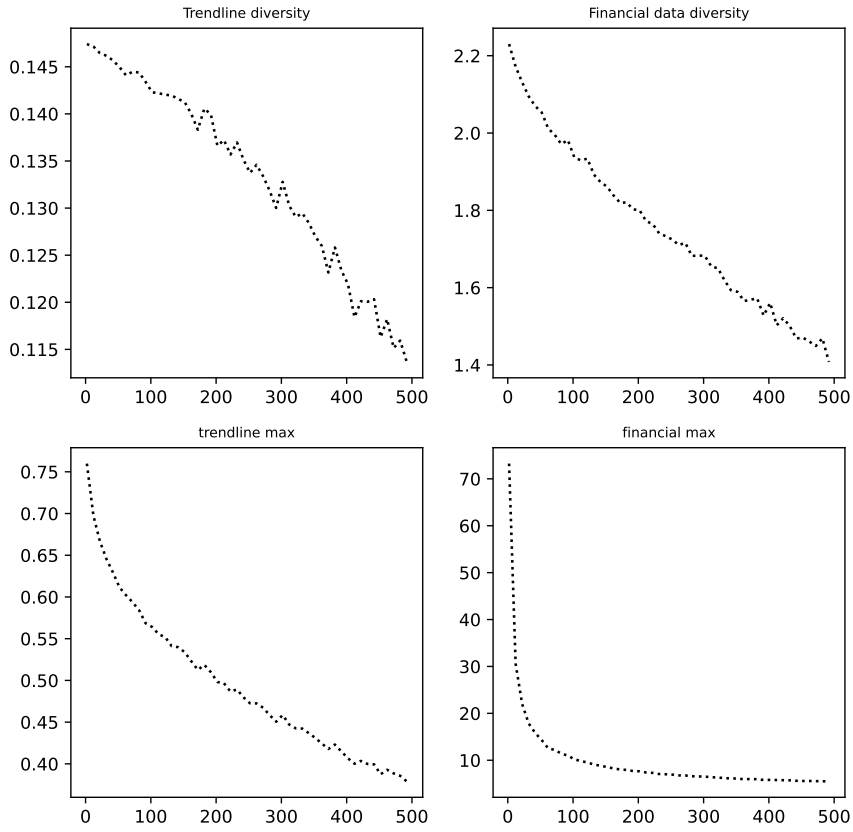

Figure 3.1: Pre-analysis of the data sets, assigning stocks to groups randomly

In the figure 3.1 it can be detected that the diversity drops approximately linearly when number of K increases. This is the cause of there being simply less items in each group by each iteration. The higher number there is stocks in single group the more diversity there is. The items in the groups can still be dissimilar but just the effect of having less stocks in the same list causes the diversity to drop. In the following chapters when the stocks are actually clustered based on their similarity the diversity should drop significantly faster, since the labels are assigned based on the actual similarity between the stocks. For a comparison point it can be still noted from here that by 500 groups the diversity in average is 0.116 for trendlines and 1.42 for financial data. The range of diversity in the groups on the other hand drops to 0.4 compared to 0.99 in the beginning for trendlines and for financial data it goes down until around 5 at 500 groups. At K value of 500, the number of stocks per group would be on average 6.5 which is quite low. In the implementation phase it can be compared that at which value of K, the same value of diversity is achieved.

3.7 Approach 1. Clustering combined distance matrices

The first approach that was experimented with was based on a hypothesis that calculating the distance matrices for both data sets independently and combining them to a single matrix would enable to cluster the stocks based on both data sets. As a whole, the combined distance matrix would produce a total sum of both data sets. The distance matrices created in the previous chapter are used in this approach. The steps for the whole approach are the following.

1. Calculate the distance matrix based on trendlines
2. Calculate the distance matrix based on financial data
3. Combine distance matrices by scaling and weighing

4. Cluster the combined distance matrix

3.7.1 Preprocessing

First distance matrix was formed for trendlines of the stocks using cosine distance function created in chapter 3.5 and cdist function from scipy library. This produced a distance matrix of size 3272x3272. Next the same was done for financial data set but with Euclidean distance. Then both distance matrices were scaled with MinMaxScaler into a range of zero to one. At this point it is though important to note that as the average diversity differs in both data sets it is necessary to weigh them equally. As calculated in chapter 3.6. the average diversity in trendlines data set was 0.15 and average maximum difference 0.7756. This would mean that the average diversity is 19.3% from the maximum. For the financial data set this would be only 2.81% since the average diversity of financial data set is 2.23 and average from the maximum differences is 79.27. If these data sets would be combined without weighing, the trendlines would be emphasized more. This can be fixed however by multiplying the trendlines distance matrix by 0.15 since 2.81 / 19.3 * 100 = 15%. Finally both matrices were combined in order to achieve a distance matrix that reflects the distances between each stock to one another based on both financial and trendline data.

3.7.2 Visualization and dimensionality reduction

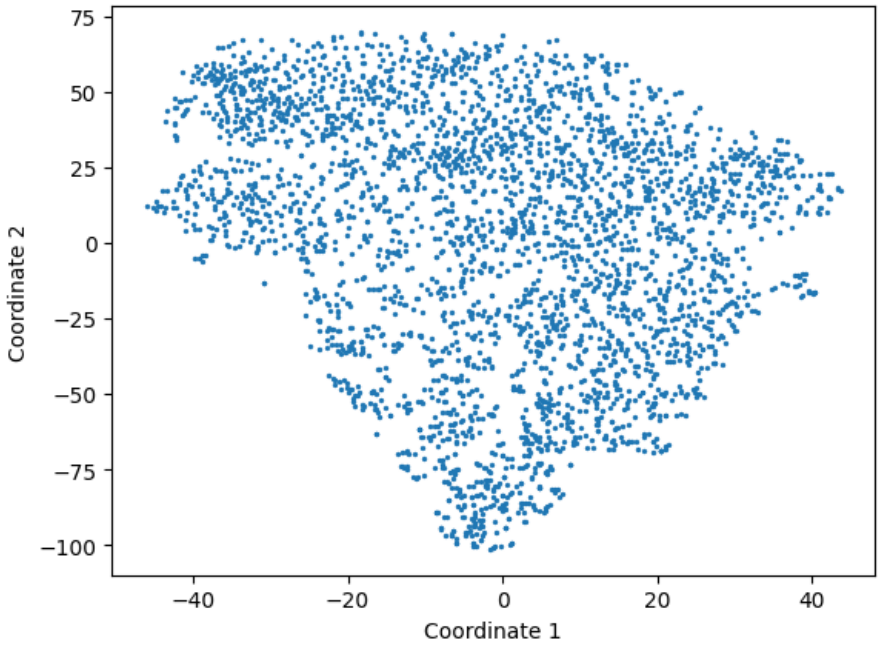

Figure 3.2: Approach 1. Visualization of a combined distance matrix with t-SNE

Before experimenting with the clustering techniques, the distance matrix was aimed to be visualized.The visualizations were tried with MDS and t-SNE as both methods from sklearn enable inputting a precomputed distance matrix. However after plotting the visualizations it was quickly detected that t-SNE provided more reasonable results. The results can be seen in figure 3.2. MDS aims to preserve all of the distances in the distance matrix in their actual form, in this case it is not however meaningful to preserve all of those values as the focus is only in the closest neighborhood for each stock. This way the closest pairs can be over emphasized. Therefore t-SNE was used. Each scatter in the plot represents a stock. For perplexity a value of 30 was used.

3.7.3 Clustering

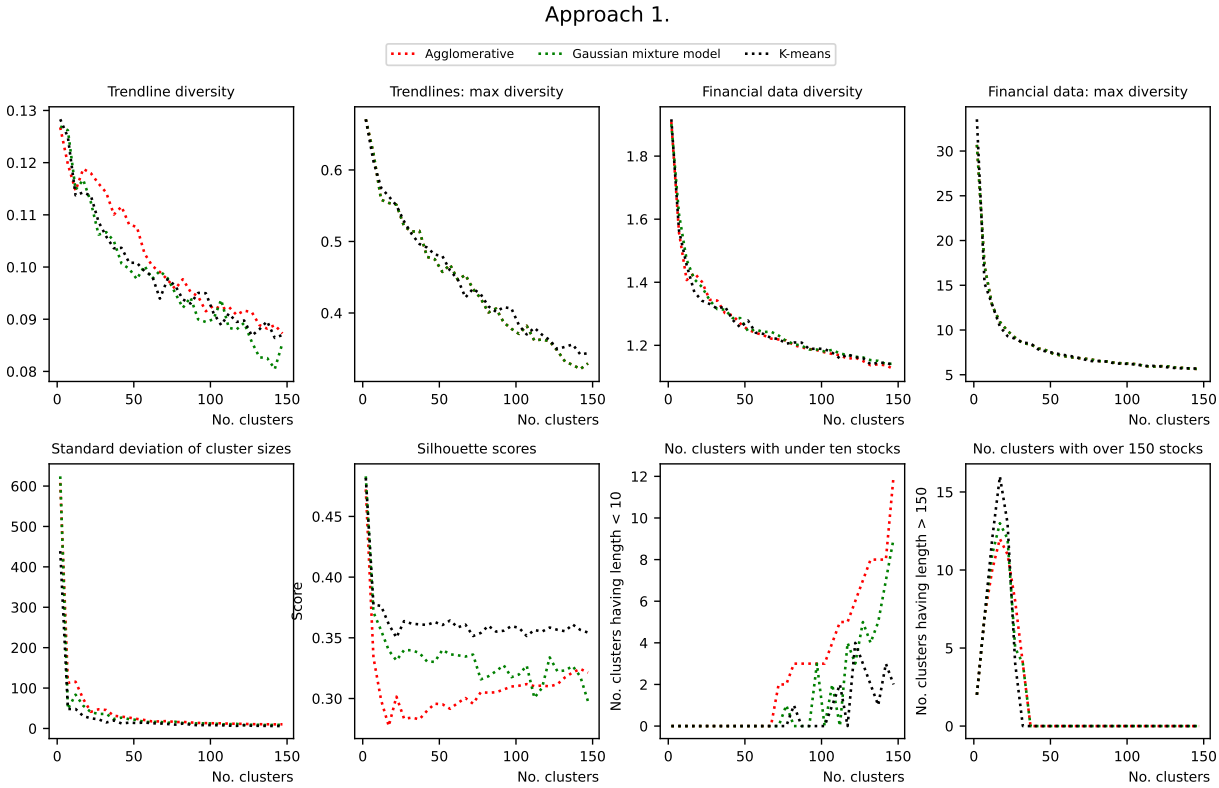

Figure 3.3: Approach 1. Comparison of clustering techniques for a combined distance matrix

The optimal number of clusters was evaluated between 2 and 150. First clustering method analyzed was K-means. With K-means elbow method was used to determine the number of clusters in the data set. As a results elbow method was able to detect a number of 10 clusters. Considering the aim, this amount wasn't sufficient in order to determine moderately sized clusters containing fewer stocks than three hundred. In comparison, Gaussian method using full covariance with BIC analysis provided a value of 12 clusters. Interestingly, when analyzing the optimal number of clusters using AIC, it showed an optimal number of 50 using full covariance. This could be actually a cause of that the AIC penalizes complex models less than BIC

as stated in 2.2.5.

Next, for each clustering method, more analysis was done to determine the diversity and coverage in each cluster at each value of K. In addition silhouette value was measured. The figure 3.3 shows what kind of results were achieved with each clustering technique. For each plot the X-axis represents the number of clusters K formed and Y-axis the diversity, standard deviation, silhouette score or number of clusters where count of stocks is either below 10 or above 150. Let's first analyze the diversity of the clusters which is represented in the top row. As calculated in the previous chapter the average diversity of trendlines across all of the stocks is 0.15. An average diversity of 0.09, for the clusters, is achieved at K value at 100. This represents a drop of 40 percentages from the whole data set's average. As noted, before in chapter 3.6 the score of 0.11 for trendlines was achieved at K number of 500 when assigning each stock randomly. So the angle of the diversity for trendlines drops much more sharply as expected now that the similarity of stocks is actually considered compared to assigning labels randomly, as expected. The same can be noticed from the diversity based on financial data. The diversity drops sharply to 1.4 and then more steadily towards 1.2 which is achieved at K value of 150. At K value of 100 a measure of 1.22 is taken of average financial data diversity for clusters. This would describe a drop of 45% which is also quite promising result.

The max diversity from both aspects, trendlines and financial data describes how far the furthest pair of stocks are on average in each cluster. In this case there is a significant drop also. As stated previously the clusters in the visualization were not able to be detected only by visualization. In addition the silhouette scores do not show particularly high results. The highest silhouette score is achieved with K-means at K value of 2. However the diversity at this point is still decreasing sharply so it would not be reasonable to cluster the stocks using this value. In addition the coverage would be too large in order to give actually similar recommendations based

on the cluster label of a stock. Silhouette score however stays at the same level after K value of 10.

Considering the coverage it would be aimed that there are no clusters with only one or two stocks. On the other hand too large clusters create large diversity and large recommendation lists. Therefore the number of clusters where the number of stocks is either under 10 or above 150 is also considered. With K-means this goal is achieved when K value is between 40 and 80 or 90 to 100.

In summary the key seems to be to find the balance with diversity and coverage. With this approach K-means seems to provide the most effective results in terms of diversity, coverage and silhouette score. However Gaussian mixture model provides the most optimal number of clusters using AIC since it recognizes the highest number of clusters.

3.8 Approach 2. Multi-step clustering

The second approach experimented with relied onto clustering the different data sets in multiple steps. The trendlines of each stock were first clustered with a technique that provided the best results in terms of diversity, accuracy, and coverage. This feature was able to be then transformed to categorical value and finally one hot coded into binary columns. The same process was then done to the financial data set. After this the new dataframe containing both labels from each step was clustered again using the same techniques as in the first approach. The steps for this approach were the following.

1. Cluster trendlines
2. Cluster financial data
3. Add resulted labels from each clustering processes to a dataframe
4. Cluster the new dataframe containing both features

3.8.1 Step 1. Clustering trendlines

Preprocessing

In terms of preprocessing there wasn't other steps needed in addition to the ones done in chapter 3.4

Visualization

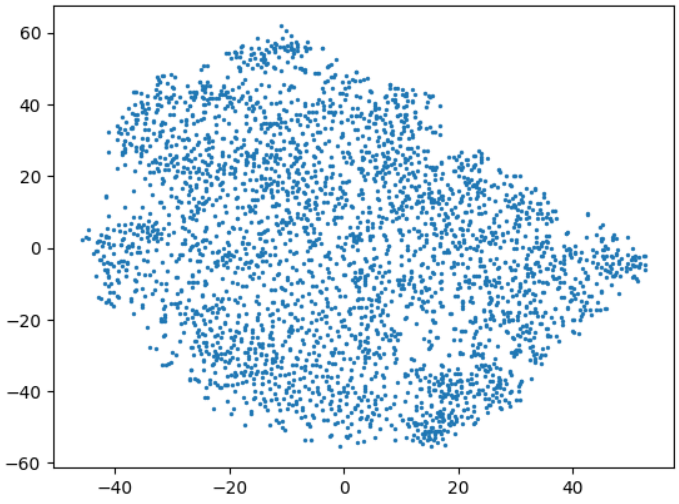

Figure 3.4: Approach 2. Similarity matrix of stock trendlines visualised with t-SNE

In this case also, the same notes were done as in approach 1 visualization. Figure 3.4 visualizes the trendlines of the stocks using t-SNE. t-SNE provided more reasonable visualization as it does not preserve the furthest datapoints from a given point but instead only the closest ones are kept. For perplexity a value of 30 was used.

Clustering

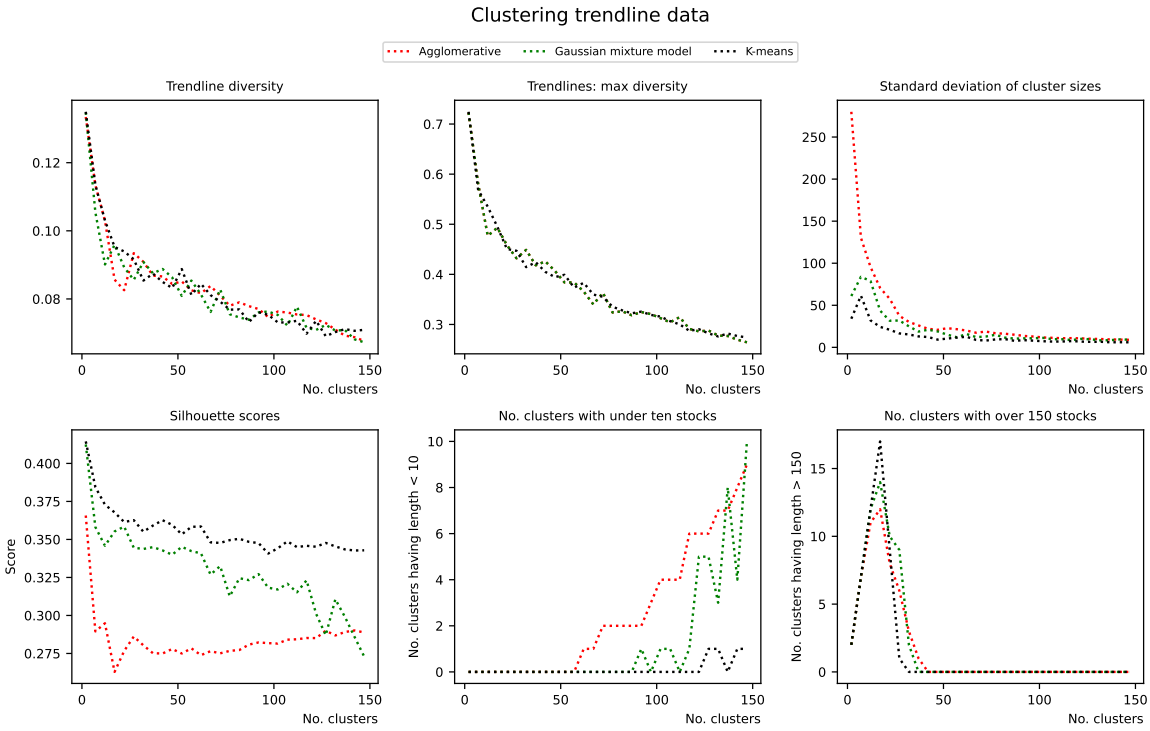

Figure 3.5: Approach 2. Comparison of clustering techniques for trendline data

Clustering trendline data set only was also started with analyzing elbow method and BIC score. Elbow method gave an optimal cluster count to be at 7 and on the other hand BIC gave values of 9 using full covariance, 19 using spherical covariance, 18 tied covariance and 12 diag covariance. AIC however again recommended the highest value of K as 30. Figure 3.5 shows the results of clustering the trendlines data. Compared to approach 1 where the financial data was included in the matrix, now the trendlines diversity expectedly drops sharper as the focus is only on one data set.

To reference this the diversity can be compared when 0.08 is achieved. In approach 1 results the diversity of trendlines dropped below 0.08 at K value of 132 whereas in this case it is achieved at K value of 82. Again in terms of silhouette score K-means achieves the highest results. With GMM and K-means the coverage of recommendation lists stays between 10 and 150 when K value is between 40 and 100. Based

on these results there isn't huge differences between K-means and Gaussian mixture model but based on silhouette score K-means provides better results. However if considering coverage the number of K should be in the range of 50 and 110 as at this point the sizes of clusters are between 10 and 150. Agglomerative clustering on the contrary does not provide as high results as K-means and GMM.

3.8.2 Step 2. Clustering financial data set

Preprocessing

In addition to chapter 5, other preprocessing steps was not necessary to be done. As stated in that chapter, the financial data was scaled using StandardScaler from sklearn library.

Visualization

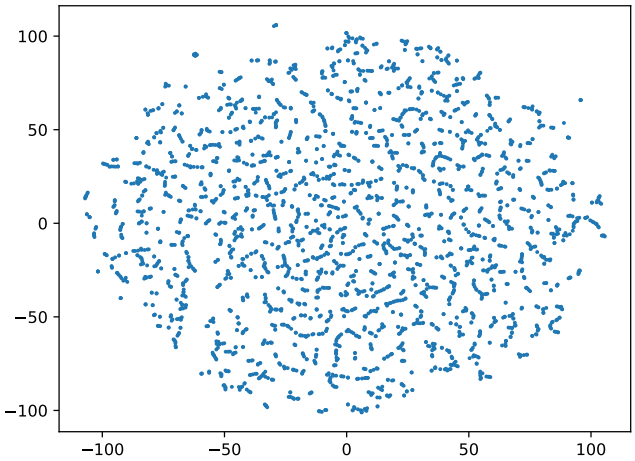

Figure 3.6: Approach 2. Distance matrix of stocks' financial data visualized with t-SNE

For visualizing the financial data of the stocks, PCA, t-SNE and MDS was experimented with. PCA was able to reduce the dimensionality successfully but at

the same time it lost the actual distances for a point's neighbor. PCA ended up scattering all of the data in the same spot except for few outliers. MDS worked more reasonably but compared to t-SNE it was not able to visualize any distinguishable clusters. Also at this point it was conducted that t-SNE is more suitable given the aim of the implementation. As only the closest neighbors are considered important in the clustering process preserving all of the distances is not meaningful. Therefore also financial data was visualized with t-SNE. Figure 3.6 shows that visualizing the data represented some visible clusters where scatters seemed to form small groups of points. Compared to trendline visualization the clusters seem to be more distinct.

Clustering

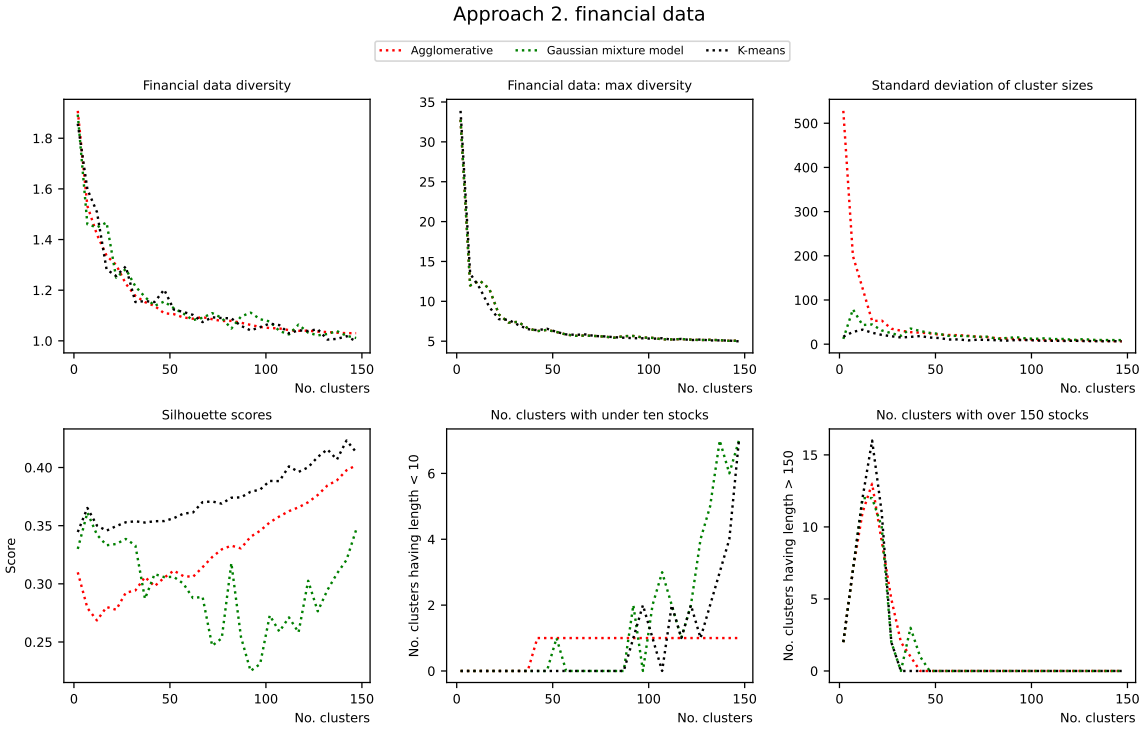

Figure 3.7: Approach 2. Comparison of clustering techniques for financial data

Analyzing the number of clusters with Gaussian mixture model and BIC gave and value of 7-8 when using either full, tied, spherical or diag covariance types. Elbow method did not differ from this conclusion as it also provided a value of 7 clusters.

In this case AIC method maximised the number of clusters as from range of 2 to 152 it showed that the value of AIC kept dropping as the number of K increased. Figure 3.7 shows the results of clustering the financial data of the stocks. For financial data the diversity drops most sharply between K value of 2 and 25. After 50 the diversity begins steadily approach 1. Again the silhouette score is highest when K-means is used. If comparing the growth of silhouette score and the visualization, it can be detected that the highest silhouette score is probably achieved at higher number of K. Considering coverage the optimal number of K would be between 50 and 100 as at that range the sizes of clusters would not differ largely. Based on financial data the best option would be to use K-means to cluster the stocks at number of 100. At that point the silhouette score is highest, diversity at lowest point but there still isn't clusters with only under 10 stocks.

3.8.3 Step 3. Cluster the new dataframe containing both cluster labels

Preprocessing

From previous steps it was conducted that K-means and gaussian mixture model provide the best form of clusters for this use case. In step 3. K-means was used to cluster both data sets separately with number of 100 K. This provided evenly sized clusters where diversity is still low. Even though the elbow method detected the optimal value to be 7, the diversity inside the clusters would not have dropped enough at that point yet. When both data sets were clustered using K-means, their labels were added to a new dataframe that now contained categories for trendlines and financials. In order to cluster this dataframe again, the values were one hot encoded into binary values.

Visualization

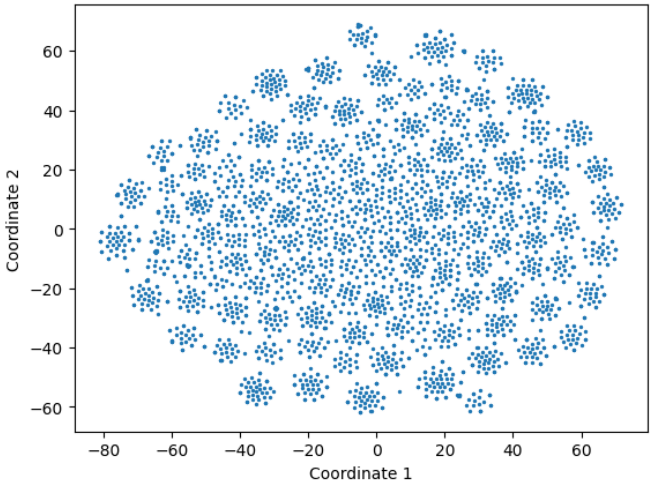

Figure 3.8: Approach 2. Combined dataframe visualized with t-SNE

Now that the dataframe contained only categorical values in binary form it was expected that t-SNE would produce a scatter plot with quite visible clusters. From the figure 3.8 it can be deduced that the stocks that belong to the same category considering both trendlines and financials are in the same scatter group. Those stocks that only fall either one of them are then again somewhere in between.

Clustering

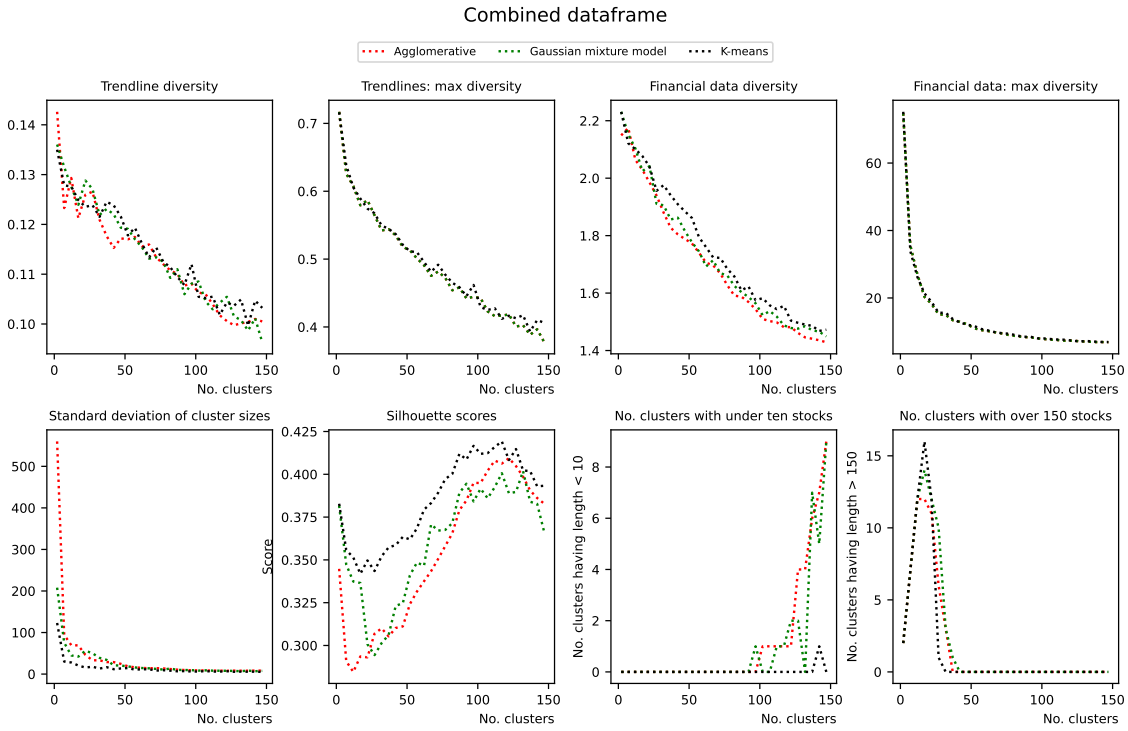

Figure 3.9: Approach 2. Comparison of clustering techniques for a dataframe containing trendline and financial labels

Dataframe containing labels from both clusters was lastly clustered in order to form groups containing both data sets. The first experiment made was to try to see what kind of results elbow method and bic score give to the t-SNE matrix created from the dataframe. Suprisingly elbow method nor BIC score were not able to detect the clusters from the matrix as both suggested only from 7-12 clusters. Next same methods were tested without running t-SNE first. In this case the methods were able to detect about 100 clusters. Naturally this is only a results of the fact that both data sets were clustered using 100 as K value. However in this case there were number of clusters that contained either only few stocks or over hundreds. On the other hand both methods provided similar diversity values in clusters. When analyzing the t-SNE matrix again with AIC, the optimal value of clusters was found at 74 which again seems the most reasonable when comparing it to the visualized

scatter plot in the figure 3.8.

So in conclusion t-SNE was still used in clustering in order to form more evenly balanced clusters. From the figure 3.9 it can be detected that silhouette score achieves the highest point at 100 clusters.In terms of recommendation coverage, the clusters are evenly formed as there aren't any huge clusters nor there are small clusters either. Diversity for financial data reaches 1.42 at K value of 150 and trendline diversity drops below 0.10 at K value of 150. In this case it is a bit more difficult to detect which technique performs most optimally as each technique is able to produce quite similar results. As the data is categorical values only the methods seem to perform evenly based on evaluated factors.

3.9 Conclusions of the clustering approaches

Both approaches were successful in categorizing stock data based on financial data and trendlines of the stocks. In addition both approaches were able to reduce diversity in the recommendation lists significantly and optimizing coverage by dividing the stocks into approximately even sized clusters. In terms of techniques K-means and Gaussian mixture model provided most accurate results in both approaches. Between them there was not big differences in terms of the evaluated factors, diversity, coverage and quality of clusters. Common methods such as BIC and Elbow method successfully recognized clusters in the data sets but the amount of clusters was rather small. However AIC technique found more clusters in both approaches. If the number of K was used based on BIC or elbow method, the diversity would still be rather high in the clusters. Therefore the amount of diversity was paid attention to in each phase. If the diversity was still dropping sharply for example at K value of 20 it was deduced that the K value should be larger in order to create tighter clusters. When visualizing the data sets it was noted that the trendlines where quite evenly spread and they effected the clustering process more than the financial

data where it was easier to find more separated clusters visually. The key in the clustering process was deduced to find the actual closest and tightest clusters with less diversity compared to finding clearly separated clusters. As clearly separated clusters did not directly result in less diverse clusters.

In terms of simplicity, approach 1. was more simple to create and it was also significantly faster as it only require one step of clustering whereas approach 2 required three. In approach 2, there also is multiple steps where there could be done some significant parameter tuning in order to optimize the results. On the other hand also approach 1 has this possibility. When experimenting and testing different dimensionality reduction techniques it was noticed that common method such as PCA was not the most optimal for this use case. As in both data sets the data is quite evenly spread and there isn't strictly noticable clusters. Instead there are close neighbors for each stock and the data overlaps significantly in each feature. This makes it difficult to form clear clusters. However looking at diversity it was easier to focus on finding the similar stocks for each cluster without paying too much attention on the clearness of the clusters. As the purpose was to in fact find the similar stock groups for each stock, t-SNE was concluded to be optimal method for this. Going further, the clustering techniques provided an interface to cluster the dimension reduced matrix. In the figure 3.10 both approaches using K-means are yet compared with each other. From the figure it can be seen that approach 1 provides more tighter clusters whereas approach is able to produce higher silhouette score. Of course it has to be remembered at this point that both data sets were clustered and categorized beforehand so it is only expected that the clustering process finds some noticeable clusters from the data. In terms of coverage, approach 2 on the other hand is able to produce more evenly sized clusters. With approach 1, it must be paid attention to that the number of K is optimal in terms of coverage as after 100 clusters it starts to produce clusters with only under ten stocks. In summary, it was deduced that in

terms of the evaluation factors and the given the aim to find similar stocks for each stock approach 1 provided more accurate results with K-means by small margin over Gaussian mixture model. Lastly the clusters were visualized using approach 1 with t-SNE and K-means. A K value of 50 was used to reach a point where diversity of financials in the clusters is starting to flatten, the sharpest drop in the diversity of trendlines is passed and silhouette score makes a slight rise after the drop from K value of 2 to 20.

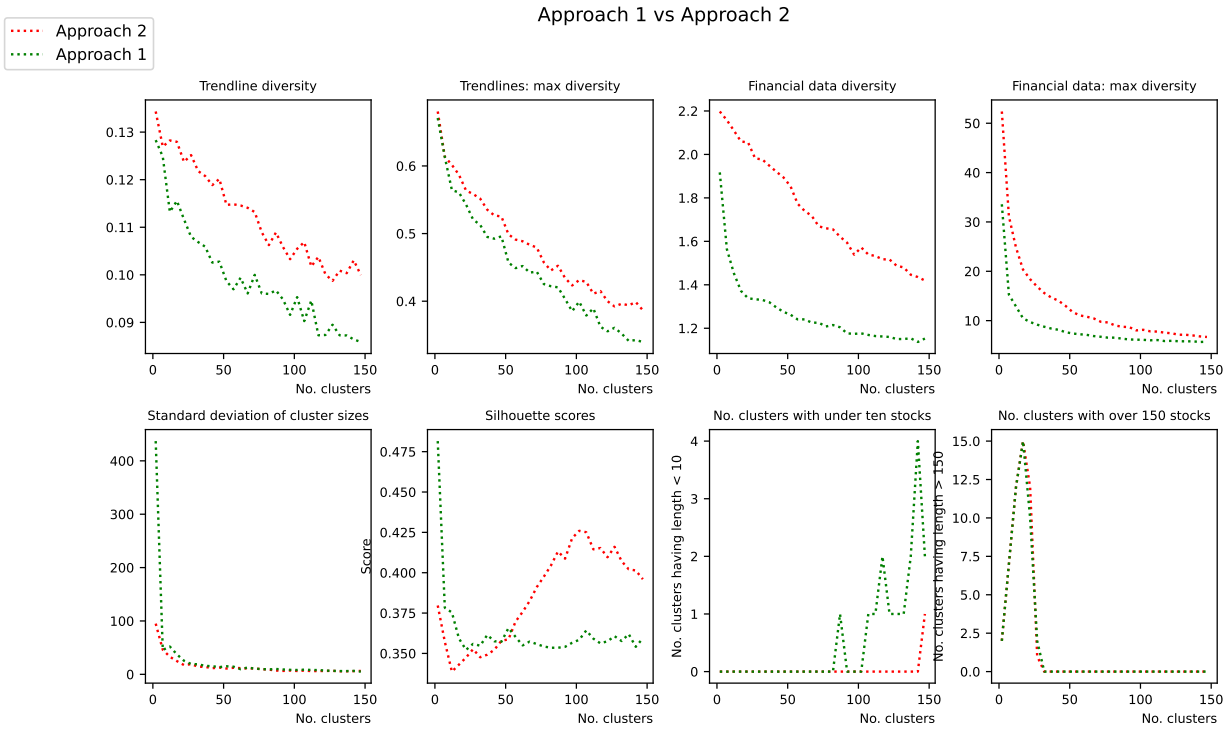

Figure 3.10: Comparison of best performing clustering techniques from approach 1. and approach 2.

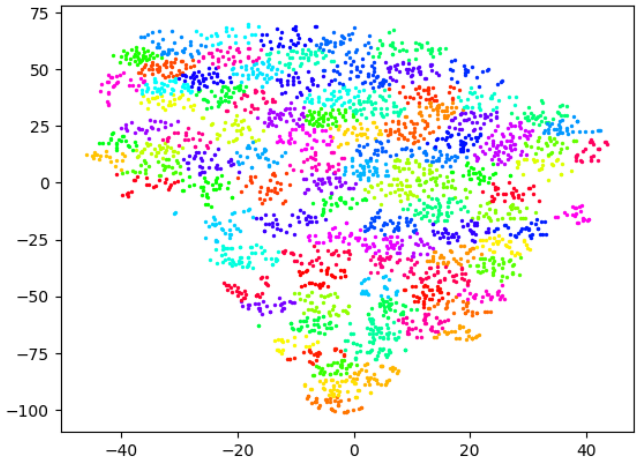

Figure 3.11: Approach 1. Combined distance matrix processed using t-SNE and K-means and visualized with each color representing a cluster

3.10 Sorting function for recommendations using nearest neighbors

Both approaches were successful in categorizing stock data based on financial data and trendlines of the stocks. In addition both approaches were able to reduce diversity in the recommendation lists significantly and optimizing coverage by dividing the stocks into approximately even sized clusters. However, when asking recommendations for a stock based on clusters, the results are not ordered in any way. Therefore a final step to the algorithm was to add a utility function that sorts the stocks of the recommended list based on similarity. For finding the top N results getTopN-function was created. The input into the function is a set of stocks that are based on the clusters formed with the previous chapter's technique. For example if a stock i is inputted to the algorithm the stocks belonging to the same cluster as stock i are feed into to the getTopN function. After this the same steps as in approach 1 are processed in order to create the dissimilarity matrix. This matrix is then used in Nearest Neighbors function from sklearn. Now when a recommendations for stocks

is requested the algorithm first finds the stocks belonging to the same cluster and then applies getTopN function to sort the results in descending order starting from the most similar.

4 Presenting and testing StockRSM

As a result Stock Recommender using Similarity-based Methods (StockRSM) was created. Techniques in the implementation included preprocessing methods, t-SNE and K-means. StockRSM uses combined distance matrices that are computed separately in order to use different similarity metrics for financial data and historical performance of stocks. The combined distance matrix is scaled in order to level the effect of both data sets. In addition, a customized cosine-function was created to calculate the similarity and correlation in the trendlines of the stocks. Finally a sorting function using nearest neighbors was implemented to sort the results from the recommendation lists.

4.1 Testing

In this chapter StockRSM is tested by inputting stock tickers into the system and visualizing the recommendations it produces. For each input the top 3 recommendations are visualized using a bar plot for financial data and line graph for trendlines. The idea is to test different kinds of companies in order to see how the algorithm categorized stocks with distinctly different features.

Recommended stocks based on Advanced micro devices (AMD)

Top 10. Recommended stocks:

1. LRCX Lam Research Corp

2. EL Estee Lauder Companies Inc
3. NOW ServiceNow Inc
4. STM STMicroelectronics NV ADR
5. INTU Intuit Inc
6. INFY Infosys Ltd ADR
7. NEM Newmont Goldcorp Corp
8. KLAC KLA-Tencor Corporation
9. ZTS Zoetis Inc
10. SPGI S&P Global Inc

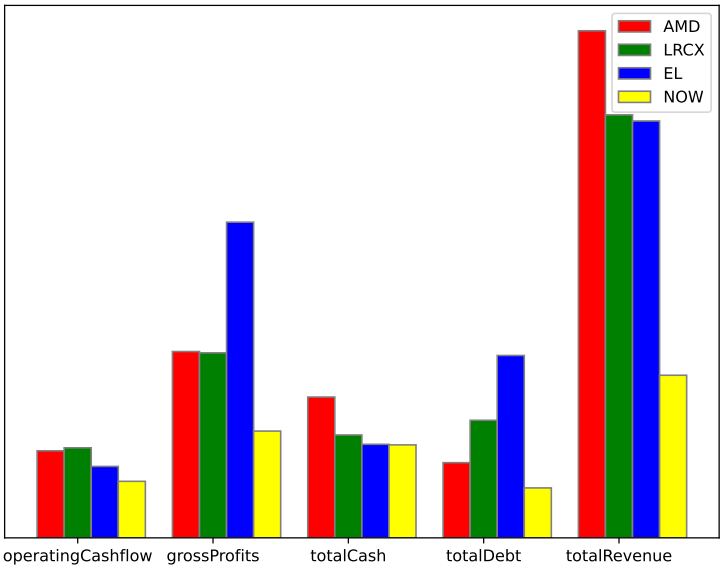

Figure 4.1: Testing of StockRSM: Financials of three first recommendations based on Advanced Micro Devices

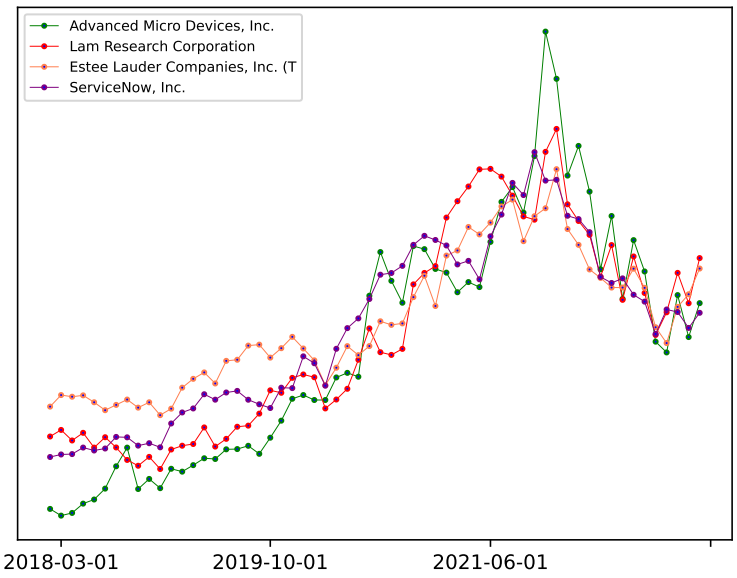

Figure 4.2: Testing of StockRSM: Trendlines of three first recommendations based on Advanced Micro Devices

Recommended stocks based on Coca-Cola Bottling Co Consolidated (COKE)

Top 10. Recommended stocks:

1. AMN AMN Healthcare Services Inc
2. IMKTA Ingles Markets Incorporated
3. AYI Acuity Brands Inc
4. STN Stantec Inc
5. THG The Hanover Insurance Group Inc
6. POOL Pool Corporation
7. HUBB Hubbell Inc
8. GMS GMS Inc
9. TTC Toro Co
10. CIGI Colliers International Group Inc Bats

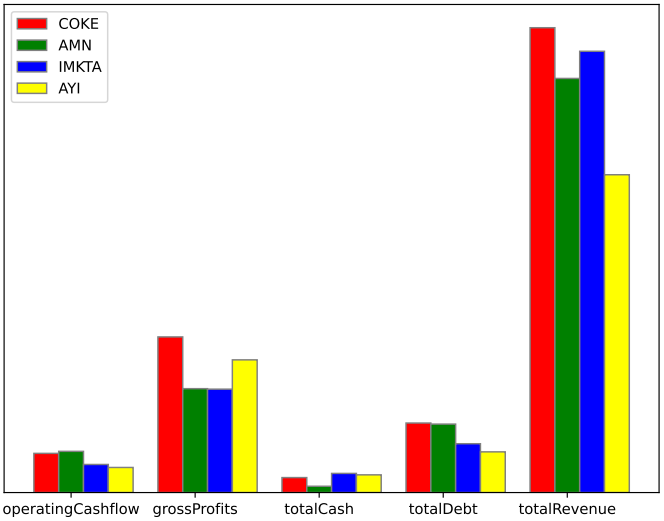

Figure 4.3: Testing of StockRSM: Financials of three first recommendations based on Coca-Cola Bottling Co Consolidated

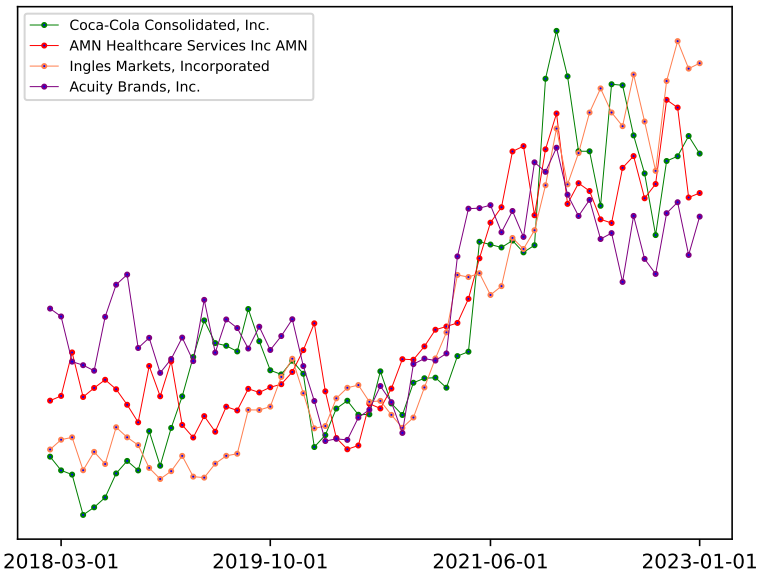

Figure 4.4: Testing of StockRSM: Trendlines of three first recommendations based on Coca-Cola Bottling Co Consolidated

Recommended stocks based on Ford Motor Company (F)

Top 10. Recommended stocks:

1. GM General Motors Company
2. CVS CVS Health Corp
3. PRU Prudential Financial Inc
4. TV Grupo Televisa SAB ADR
5. HUM Humana Inc
6. UPS United Parcel Service Inc
7. MFC Manulife Financial Corp
8. GGB Gerdau SA ADR
9. E Eni SpA ADR
10. BAK Braskem SA Class A

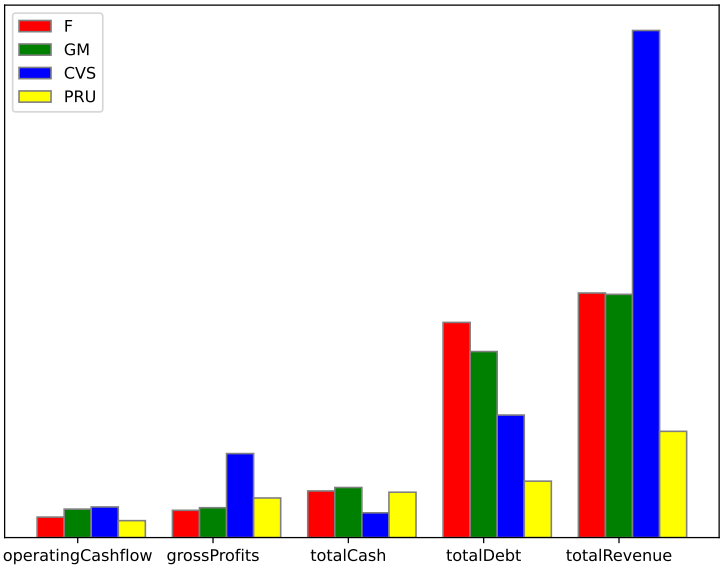

Figure 4.5: Testing of StockRSM: Financials of three first recommendations based on Ford Motor Company

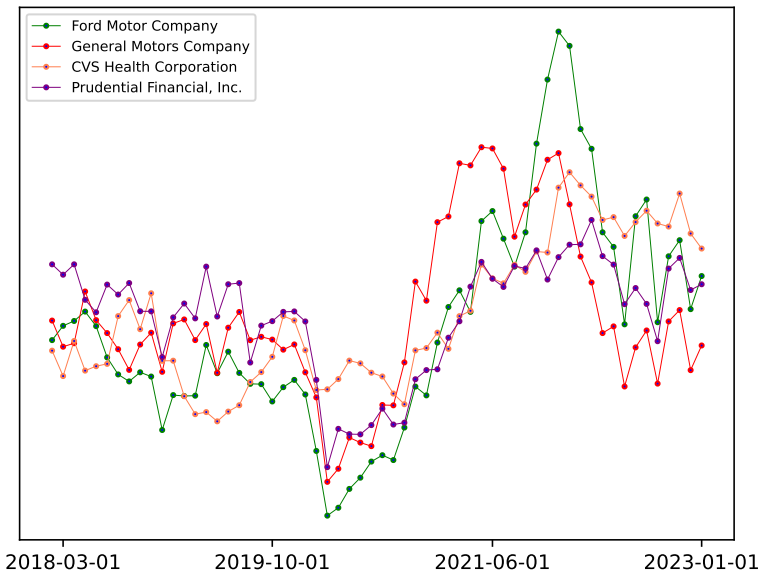

Figure 4.6: Testing of StockRSM: Trendlines of three first recommendations based on Ford Motor Company

Recommended stocks based on American Airlines Inc (AAL)

Top 10. Recommended stocks:

1. BA The Boeing Company
2. UAL United Airlines Holdings Inc
3. DAL Delta Air Lines Inc
4. ERJ Embraer SA ADR
5. PARA Paramount Global Class B
6. LUV Southwest Airlines Company
7. EOG EOG Resources Inc
8. RNR Renaissancere Holdings Ltd
9. DVN Devon Energy Corporation
10. AFL Aflac Inc

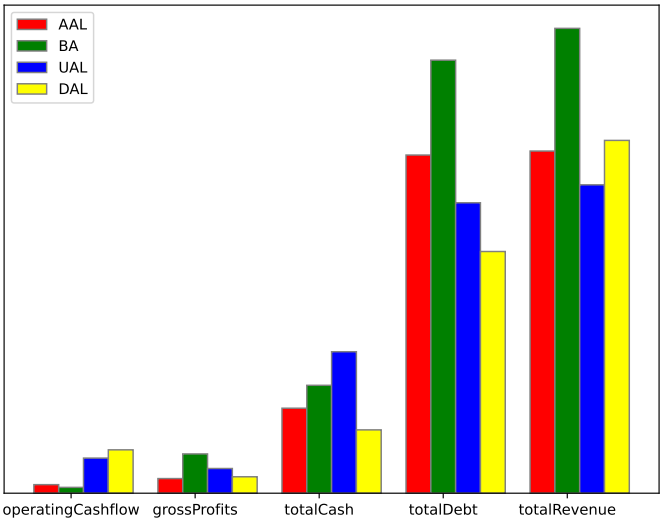

Figure 4.7: Testing of StockRSM: Financials of three first recommendations based on American Airlines Inc

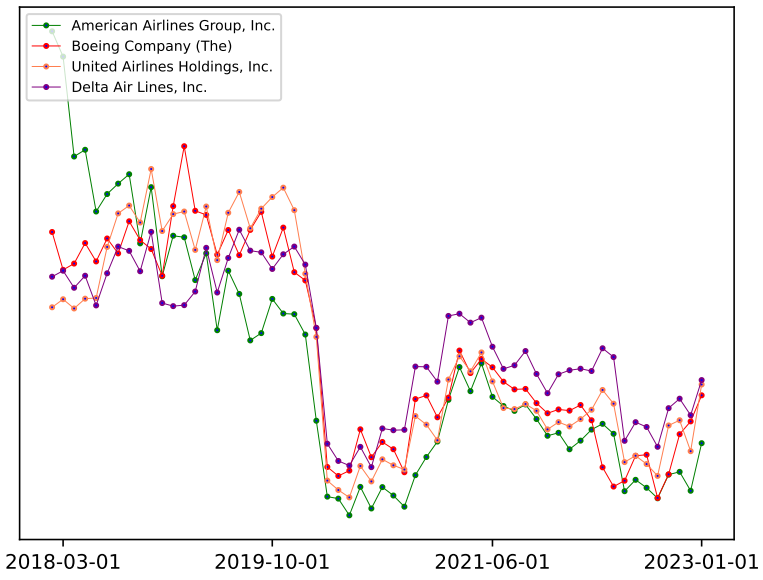

Figure 4.8: Testing of StockRSM: Trendlines of three first recommendations based on American Airlines Inc

4.2 Conclusions

From the tests it was clearly seen that StockRSM produced similar stocks to the input. When visualising the top three recommended stocks, the financials and trendlines matched quite closely, in case of the financial data it is important to also note that the values are in the same range in terms of magnitude as euclidean distance is used. Therefore it is more probable that companies with revenues in the same value range end up in the same cluster. For example company with a revenue and profit below billions is not going to be measured similar with a company that has revenue in tens of billions. This was also seen in the testing results. Interesting findings from the algorithm was that similar companies in terms of sector or industry can be found even without specifying the sector of the company explicitly in the data set. As mentioned already in the chapter 3.2, data sets contained only trendlines and financial key values. No descriptive data. From the result's standpoint this interesting as for example when finding similar stocks for American Airlines 4.1, the top 4 recommendations were either airlines or aircraft manufacturers. One of the main reasons for this can be the trendlines as they correlate strongly. Just adding the trendlines into the feature set adds the aspect of same market environment into the algorithm. In this aspect the data set was versatile enough in order to notice such patterns. In a sense, from smaller amount of features the correlations and reasons for the recommendations are easier to deduce and test. As compared to if there were hundreds of features the concept of similarity would possibly be more difficult to define.

Balancing between two data sets makes defining the similarity more complex. In this case the testing showed that there are quite similar companies based on the feature sets used in this thesis. However it is naturally probable that only looking either one of the data sets would probably produce somewhat different results. From testing it can be seen that there is a trade off between very close match based on

financial data and not so close match based on trendlines or vice versa. Between those two aspects is the similarity defined in this thesis.

Considering the original use cases for the algorithm stated in 1.2 the recommendations tested would be able to fill those needs differently compared to stock screener tools available.

Considering the first scenario of diversification of the portfolio StockRSM is able to detect the correlations of stocks across sector levels as seen in the example with Advanced Micro Devices. In this case the investor could simply avoid choosing strongly correlated stocks in one's portfolio when the closest matches are visually presented. The second scenario states an example of an investor that seeks to create a highly focused portfolio on similar stocks. Using StockRSM the investor could input a stock that is found to contain the factors to outperform the market. The recommendations could then be used to avoid the risk of choosing only one stock for the portfolio. Instead the portfolio would then contain strongly similar stocks while still focusing on avoiding a company specific risks.

For the third scenario the idea behind recommendations is a bit more complex. As the recommendations may also contain few recommendations that are not strictly from the same sector or industry as the input. This can be seen as a positive or negative factor depending on the investor's needs. If the investor only wants similar stocks based on sector the recommendations might not be the most suitable as there are also other evaluation factors, financials and the trendlines which cause the output to contain also other stocks. On the other hand if the investor only wants to compare stocks that correlate based on trendlines and financials the recommendations can be used to find the best one among those.

5 Summary

The aim of this thesis was to build a knowledge-based recommender system for stocks by comparing multiple approaches and algorithms. The idea was that based on a single stock the algorithm could recommend similar stocks based on trendlines and financial data of the stock. Chapter 2 explained the knowledge required in the implementation phase. It consisted of the theory of recommender systems, data mining, challenges and evaluation of knowledge based recommender systems and clustering of time-series data.

Methods and techniques that were used in the implementation included dimensionality reduction algorithms MDS, PCA and t-SNE, clustering algorithms K-means, agglomerative clustering, Gaussian mixture model and other basic operations such as standard scaling and min-max scaling. At first it was required to find a solution to find the similarities between the time-series data, in other words the trendlines of the stocks. For this case a customized cosine distance function was built to be used as a custom metric when computing a distance matrix. In the clustering section it was concluded that from dimensionality reduction perspective t-SNE provided most reasonable results as the other methods aimed to preserve the distances to all of the data points. From this thesis' perspective that however was not meaningful as only the similarity and neighborhood of the closest points is relevant in terms of recommending similar stocks. Clustering phase was conducted using two approaches. The first one was based on the hypothesis that the data sets of financial data and

trendlines could be combined using distance matrices. The second approach on the other hand relied on clustering both data sets separately and the finally combining the labels from each clustering process to a new dataframe. This dataframe was then again clustered again in order to form a final set of clusters containing both feature set's impact on the recommendation lists. In each approach K-means, agglomerative clustering and Gaussian mixture model was compared in order to find the best result. The best results were evaluated based on diversity in each cluster, the goodness of each cluster using silhouette score and finally coverage of each recommendation list. From these approaches, the first approach using K-means provided better results after the comparisons done using the evaluation factors. For a final addition, sorting utility function was added to the algorithm. This sorting algorithm functions using nearest neighbor method. Finally the algorithm, named StockRSM, was tested with multiple different stocks, the results were visualized and evaluated in terms of their visual representations.

5.1 Answers to research questions

RQ1. Is it possible to cluster a set of stocks based on their trendlines and financial data in order to form a recommendation system?

It was deduced that it is possible to cluster a set of stocks in order to form a recommendation system that bases its idea behind similarity. The concept of similarity is a sum of the chosen features and the metrics that is used to categorize them. Trendlines itself is a strong correlative feature which is able to detect stocks with similar macro environment. Financial data on the other hand describes in more detail the financial performance of a company. Determining the features from financial aspect is an important factor and it was important to cover the main attributes of the companies. In this case it, total revenue, total debt, operating cash flow, gross

profits and total cash was chosen.
The algorithm to find the similar stocks as a whole was deduced to be a combination of preprocessing, similarity metrics, clustering, dimensionality reduction and sorting capabilities with nearest neighbors.
It was also deduced that the trendlines of the stocks required a customized similarity function in order categorize the stocks based on historical performance. For financial data Euclidean distance was found to be the suitable option.

RQ2. How the distinctly different data types, financials and trendlines of the stocks, can be combined for the clustering process?

During the process of this thesis it was conducted that the datatypes were able to be combined using two approaches. The first one relied on computing the distance matrices for both datasets separately and finally clustering the combined distance matrix by reducing dimensionality with t-SNE.
The second approach clustered both data sets separately with the most suitable technique. The labels from each clustering process resulted a new dataframe. This dataframe was then able to be clustered again in order to form the final categories containing the both data sets.

RQ3. If multiple approaches and techniques can cluster the set of stocks, which one of them provides the most promising results?

For clustering techniques the methods used contained K-means, Gaussian mixture model and Agglomerative clustering. For dimensionality reduction PCA, MDS and t-SNE was experimented with. From these techniques K-means and Gaussian mixture model achieved the most promising results with the use of t-SNE. PCA and

MDS were not found to be suitable for the data sets used in this thesis as they aimed to preserve all the distances between the stocks whereas only the closest neighbors were found to be meaningful considering recommendations.
Both approaches mentioned in RQ2 were successful in the clustering process including the three different clustering techniques used in each. However it was conducted that the first approach using t-SNE and K-means provided most promising results in terms of diversity, coverage and quality of clusters.
Finding clusters in a such complex data environment is highly technical. Even though successful results were achieved with multiple approaches, optimizing the algorithm and finding the best techniques requires a lot of tests and comparisons.

5.2 Future work

As this thesis and the implementation was conducted there were phases where some ideas for improvements were noticed. These improvements were outside of the scope of this thesis, but in this chapter those ideas are discussed. In this thesis the focus wasn't on optimizing the feature set and finding the most optimal features for financial data of the stocks. Instead the financial data set was limited in the core values of the financials in order to speed up the development process by keeping the focus more on the algorithm itself. For further improvement of the algorithm more financial data including historical values could be added to the algorithm to make it aware of the financial performance of the stocks similarly to the trendline data set. For example revenue growth quarterly or annually. Or growth of profit overtime.
From trendline data sets perspective, the number of features was also limited. A possible improvement could be to add also the close, high, low and volume values to the equation similarly as in candlestick charts. But most importantly the interval of the trendline datapoints should be more frequent as in this thesis it was set to month to speed up the implementation process. However in order to get more pre-

cise recommendations even based on shorter dataframe from recent weeks or months the interval could be set to even days. This naturally would also require more computing power as the number of datapoints would be much higher. In some cases it was noted that only a year of datapoints with month as an interval might not be sufficient.

A reasonable addition to the algorithm would be to add a pipeline handling the data by keeping it up to date. This would also require the model to be built again from time to time. New stocks however could be added to the clusters by predicting even without computing the model again from ground up. Pipelines should be able to provide data for both financial data sets and trendlines. This requires lots of memory and computing power.

In this thesis also parameter tuning was not paid much detail. Even though the effect of the number of clusters were compared in each approach, some parts were left for a smaller focus. Comparison for different preprocessing methods and more detailed view on dimensionality reduction could be a probable improvement. In addition the parameters in the techniques used such as perplexity in t-SNE could be tuned in more to find more clear clusters. From an investors perspective an interesting addition could be also a possibility to add a selection of which features should be weighed more in the recommendations. The weights could then match the individual investors likings more precisely. This would further improve the recommendation accuracy for an individual. Also the weighing of features could possibly add a layer of finding more interesting similarities between the stocks.

References

[1] S. S. Anand and B. Mobasher, "Intelligent techniques for web personalization", 2003.

[2] G.-D. Bordeianu and F. Radu, "Basic types of financial ratios used to measure a company's performance", *Economy Transdisciplinarity Cognition*, vol. 23, no. 2, pp. 53–58, 2020, Num Pages: 53-58 Place: Bacau, Romania Publisher: George Bacovia University, ISSN: 14545675. (visited on 04/21/2023).

[3] J. Chen. "Trendline: What it is, how to use it in investing, with examples", Investopedia. (Jul. 10, 2022), [Online]. Available: `https://www.investopedia.com/terms/t/trendline.asp` (visited on 03/30/2023).

[4] J. Edwards. "Why market correlation matters", Investopedia. (Oct. 31, 2022), [Online]. Available: `https://www.investopedia.com/articles/financial-advisors/022516/4-reasons-why-market-correlation-matters.asp` (visited on 03/30/2023).

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, ISBN: 978-0-387-85819-7 978-0-387-85820-3. (visited on 11/04/2022).

[6] D. Kim, J. C. Francis, and D. Kim, *Modern Portfolio Theory: Foundations, Analysis, and New Developments*. New York, UNITED STATES: John Wiley & Sons, Incorporated, 2013, ISBN: 978-1-118-41720-1. (visited on 03/30/2023).

[7] I. Beregovskaya and M. Koroteev, *Review of clustering-based recommender systems*, Sep. 27, 2021. arXiv: 2109.12839[cs]. (visited on 10/01/2022).

[8] B. Smyth, P. Brusilovsky, A. Kobsa, and W. Nejdl, "The adaptive web, methods and strategies of web personalization", 2007.

[9] A. Twin. "What is data mining? how it works, benefits, techniques, and examples", Investopedia. (Aug. 2, 2022), [Online]. Available: https://www.investopedia.com/terms/d/datamining.asp (visited on 03/30/2023).

[10] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*. Bosa Roca, UNITED STATES: CRC Press LLC, 2014, ISBN: 978-1-4665-8333-7. (visited on 03/10/2023).

[11] E. Alpaydin and F. Bach, *Introduction to Machine Learning*. Cambridge, UNITED STATES: MIT Press, 2014, ISBN: 978-0-262-32574-5. (visited on 03/10/2023).

[12] P. D. Waggoner, *Modern Dimension Reduction*, 1st ed. Cambridge University Press, Jul. 31, 2021, ISBN: 978-1-108-98176-7 978-1-108-98689-2. (visited on 04/21/2023).

[13] M. G. Vozalis and K. G. Margaritis, "A recommender system using principal component analysis", Jul. 1, 2008.

[14] "How t-SNE works — openTSNE 0.3.13 documentation". (Jan. 20, 2021), [Online]. Available: https://opentsne.readthedocs.io/en/latest/tsne_algorithm.html (visited on 04/21/2023).

[15] G. Bonaccorso, *Mastering Machine Learning Algorithms: Expert Techniques to Implement Popular Machine Learning Algorithms and Fine-Tune Your Models*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2018, ISBN: 978-1-78862-590-6. (visited on 04/21/2023).

[16] M. Deshpande and G. Karypis, "Item-based top- N recommendation algorithms", *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143–177, Jan. 2004, ISSN: 1046-8188, 1558-2868. (visited on 09/23/2022).

[17] R. Ahuja, A. Solanki, and A. Nayyar, "Movie recommender system using k-means clustering AND k-nearest neighbor", Jan. 1, 2019, pp. 263–268.

[18] B. Saji. "Elbow method for finding the optimal number of clusters in k-means", Analytics Vidhya. (Jan. 20, 2021), (visited on 04/26/2023).

[19] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Philadelphia, PA, UNITED STATES: CRC Press LLC, 2013, ISBN: 978-1-4665-5822-9. (visited on 04/22/2023).

[20] O. C. Carrasco. "Gaussian mixture models explained", Medium. (Feb. 21, 2020), [Online]. Available: `https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95` (visited on 05/05/2023).

[21] J. Brownlee. "Probabilistic model selection with AIC, BIC, and MDL", MachineLearningMastery.com. (Oct. 29, 2019), [Online]. Available: `https://machinelearningmastery.com/probabilistic-model-selection-measures/` (visited on 04/26/2023).

[22] A. Geiger. "Evaluating recommender systems in absence of labeled data | SAP blogs". (Mar. 29, 2021), [Online]. Available: `https://blogs.sap.com/2021/03/29/evaluating-recommender-systems-in-absence-of-labeled-data/` (visited on 04/27/2023).

[23] X. Huang, Y. Ye, L. Xiong, R. Y. K. Lau, N. Jiang, and S. Wang, "Time series k-means: A new k-means type smooth subspace clustering for time series data", *Information Sciences*, vol. 367-368, pp. 1–13, Nov. 1, 2016, ISSN: 0020-0255. (visited on 05/06/2023).

[24] T. Izzet. "Introduction to time series clustering". (2021), [Online]. Available: https://kaggle.com/code/izzettunc/introduction-to-time-series-clustering (visited on 05/06/2023).

[25] T. Warren Liao, "Clustering of time series data—a survey", *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, Nov. 1, 2005, ISSN: 0031-3203. (visited on 05/06/2023).

[26] E. E. Özkoç, "Clustering of time-series data", in *Data Mining - Methods, Applications and Systems*, IntechOpen, Feb. 3, 2020, ISBN: 978-1-83968-319-0. (visited on 05/06/2023).

[27] T. Smith. "Autocorrelation: What it is, how it works, tests", Investopedia. (Mar. 19, 2023), [Online]. Available: https://www.investopedia.com/terms/a/autocorrelation.asp (visited on 05/06/2023).

[28] F. Paolone, *Accounting, Cash Flow and Value Relevance* (SpringerBriefs in Accounting). Cham: Springer International Publishing, 2020, ISBN: 978-3-030-50687-2 978-3-030-50688-9. (visited on 04/21/2023).

[29] "Revenue vs. income: What's the difference?", Investopedia. (), [Online]. Available: https://www.investopedia.com/ask/answers/122214/what-difference-between-revenue-and-income.asp (visited on 04/21/2023).