FH Salzburg
MultiMediaTechnology

Identifying the Ideal Length of Time to Record Smartphone Data, in Order to Obtain Distinct Clusters to Predict Eating Crises

Bachelor Thesis 2

Author: Natasha Lauren Troth
Advisor: FH-Prof. DI Dr. Simon Ginzinger, MSc.

Salzburg, Austria, 29.06.2020

Affidavit

I herewith declare on oath that I wrote the present thesis without the help of third persons and without using any other sources and means listed herein; I further declare that I observed the guidelines for scientific work in the quotation of all unprinted sources, printed literature and phrases and concepts taken either word for word or according to meaning from the Internet and that I referenced all sources accordingly.

This thesis has not been submitted as an exam paper of identical or similar form, either in Austria or abroad and corresponds to the paper graded by the assessors.

Date *Signature*

First Name *Last Name*

Kurzfassung

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean venenatis nulla vestibulum dignissim molestie. Quisque tristique tortor vitae condimentum egestas. Donec vitae odio et quam porta iaculis ut non metus. Sed fermentum mauris non viverra pretium. Nullam id facilisis purus, et aliquet sapien. Pellentesque eros ex, faucibus non finibus a, pellentesque eu nibh. Aenean odio lacus, fermentum eu leo in, dapibus varius dolor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sit amet ornare velit. Donec sit amet odio eu leo viverra blandit. Ut feugiat justo eget sapien porttitor, sit amet venenatis lacus auctor. Curabitur interdum ligula nec metus sollicitudin vestibulum. Fusce placerat augue eu orci maximus, id interdum tortor efficitur.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean venenatis nulla vestibulum dignissim molestie. Quisque tristique tortor vitae condimentum egestas. Donec vitae odio et quam porta iaculis ut non metus. Sed fermentum mauris non viverra pretium. Nullam id facilisis purus, et aliquet sapien. Pellentesque eros ex, faucibus non finibus a, pellentesque eu nibh. Aenean odio lacus, fermentum eu leo in, dapibus varius dolor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sit amet ornare velit. Donec sit amet odio eu leo viverra blandit. Ut feugiat justo eget sapien porttitor, sit amet venenatis lacus auctor. Curabitur interdum ligula nec metus sollicitudin vestibulum. Fusce placerat augue eu orci maximus, id interdum tortor efficitur.

Contents

List of Figures

Listings

List of Tables

1 Introduction

Han, Pei, and Kamber (2011)[18, 32, 362, 363, 367] declare, that data mining is used to discover patterns and knowledge from data. Cluster Analysis is a type of machine learning algorithm known as unsupervised machine learning. It is used in data mining to divide data into groups (clusters). Each cluster contains data that is similar to each other, but dissimilar to the data allocated to other clusters. Cluster Analysis can be used to acquire knowledge on the distribution of the data, discover characteristics, detect outliers and reduce noise, or to pre-process data for other algorithms.

There are several different methods to create clustering. Han, Pei, and Kamber (2011)[364, 366-367, 374, 385, 392] explain, that objects are often arranged into clusters using distance measures (e.g. Euclidean or Manhatten distance measures). The authors divide the clustering algorithms into the following categories:

- Partitioning methods (examples: k-means, k-medoids)
- Hierarchical methods (examples: BIRCH, Chameleon)
- Density-based methods (examples: DBSCAN, OPTICS)
- Grid-based methods (examples: STING, CLIQUE)

Bermad and Kechadi (2016) introduce in their paper, how clustering can be used in digital forensics to provide information on all the events that led up to a certain crime. They used ascending hierarchical clustering to receive clusters of events (e.g. phone calls, SMS) ordered in time, thus creating a timeline of events leading up to the incident.

Dey and Chakraborty (2015)[1,2,6,7] give an example, where clustering was implemented to predict future weather. Air pollutant data was preprocessed and then arranged into clusters using (incremental) DBSCAN clustering. Finally, priority based protocol was used on them to predict weather conditions and a temperature range. The accuracy of the technique, based on hit and miss times, was calculated to approximately 74.5%.

SmartEater [1] is an upcoming mHealth (mobile health) app, with the goal to provide the user with content-dependent feedback, to avert a food craving episode. The app will predict future eating crises based on the user's past behaviour. In order to reduce intense user input, the app records and uses various smartphone sensor data. With the help of data mining, machine learning algorithms, and pattern recognition, this recorded situational context data will aid in predicting stress. The following data is recorded by the app:

1. Movement of the smartphone (accelerometer)
2. Volume of the audio

1. https://sites.google.com/site/eatingandanxietylab/resources/smarteater

3. Percentage of screen on time
4. Number of notifications
5. Light sensor values
6. App usages in the categories communication, video players, and other

This sensor data was recorded for different lengths of time on different test subjects and aggregated to 1 hour and 3 hour files. It is necessary to establish which time period will be most fitting to make accurate predictions for the future. This thesis will use cluster analysis to determine which time period is most significant.

According to Han, Pei, and Kamber (2011)[414], the above-mentioned clustering methods work well with data sets that are not high-dimensional and have less than 10 attributes. Since the SmartEater data set only has 8 unique dimensions (columns), it is not considered high-dimensional. This paper will therefore utilise these clustering methods. Since different clustering algorithms can yield different results, multiple methods will be used and compared.

To reduce the size and amount of data, dimensionality reduction will be used. Han, Pei, and Kamber (2011)[93] define dimensionality reduction as a type of data reduction, which removes random attributes and creates a smaller data set with close to equal integrity. This thesis will compare principal component analysis (PCA) and t-SNE to reduce the dimensionality.

The clustering methods will be implemented using the Python machine learning platform (e.g. Anaconda[2], with the library scikit-learn[3]. These will be implemented on all the time lengths. The resulting clusters of each time length will be compared to one another and evaluated. Rousseeuw (1987) reveals how silhouettes can be used to measure the separation between clusters and therefore evaluate the quality of the resulting clusters are. Other mathematical evaluation scores used are Davies-Bouldin Index (Davies and Bouldin (1979)) and Caliński-Harabasz Index (Caliński and Harabasz (1974)).

The thesis will be structured as follows: Section 2 will briefly present existing work relating to this subject. The following chapter, section 3, will concentrate on the theory of data mining and cluster analysis. After covering these topics, section 4 will describe the conducted experiment and its results. In the final sections, the findings of the experiment will be discussed and summarised.

2 Related work

As introduced in section 1, SmartEater [4] will be a mHealth (mobile health) app, that predicts future eating crises based on the user's past behaviour. The predictions are made on smart-

2. https://www.anaconda.com/
3. https://scikit-learn.org/stable/
4. https://sites.google.com/site/eatingandanxietylab/resources/smarteater

phone sensor data, therefore reducing strenuous user input. The app will give the user content-dependent feedback, to avert a food craving episode.

Rahman et al. (2016) introduced a similar idea in their paper. Their goal is to predict "About-to-Eat" and "Time until the Next Eating Event" stages by using wearable sensing devices, in order to reduce serious health issues (e.g. obesity). The authors state, that detecting when a person is eating is not helpful. It is more beneficial to predict moments shortly before the user is about to eat ("About-To-Eat"). Rahman et al. learnt more on how people were currently tracking their meals by conducting a survey. They asked 75 people with varied ages and body sizes. 34 of 75 participants revealed, that they had never used any type of eating tracking or food journals. The remaining 41 respondents were further asked how long they used the respective tool. 48.9% of these had used the tool for less than a month. 65 of the 75 participants revealed that they no longer use a food tracking/journalling a tool. Further questions resulted in the following revelations: 33 of the 75 respondents wished for the app to take action directly before a meal/snack ("About-to-Eat" moments), thus supporting the authors previous assumptions. Ideas for interventions included a calorie calculator, reminders to eat balanced or of calorie allowances, visualisations of previous food eaten, or a breakdown of the nutrients taken in. The authors used a variety of different sensors, which at the time, were not all available in one device. The list of utilised sensors and the recorded data are as follows:

- Microsoft Band: physical movement (raw accelerometer, gyroscope, step count, speed), caloric expenditure, heart rate, skin temperature, etc.
- Affectiva Q sensor: measures electrodermal activity (good indicator for psychological arousal)
- Wearable microphone: chewing and swallowing sounds (for detection of current eating events)
- Android smartphone application: GPS location, recording self-reports (right before eating: "Start of Eating Event" button, current emotional state when eating, intensity of desire/craving and hunger; when finished eating: "End of Eating Event" button)

In order to predict "About-to-Eat" moments, eight participants, aged 26-54, wore these four technologies for five days. The recorded data then underwent cleaning and preprocessing, feature extraction, feature selection and machine learning. During the preprocessing, it was made sure that the raw sensor data streams were not dirty and had high accuracy. Some of these steps included resampling, normalisation, recognising chewing and swallowing sounds, extracting longitude and longitude, etc. In the feature extraction step, two parameters are used on the processed sensor time series to create windows. Firstly, the feature extraction window size parameter regulates the time duration in a specific window. While a short window duration could catch immediate characteristics in the sensor time series, a coarse one could be used for long term trends. The authors used a variety of window sizes, varying from 5 to 120 minutes. The prediction model results would decide the best window. The second parameter, the window shift size, establishes the time duration between two neighbouring windows, for example meaning

window *n* is shifted one minute compared to window *(n - 1)*. A constant shift of one minute was used for all window sizes. Statistical functions (e.g. min, max, mean, standard deviation, etc.) were used to extract a total of 158 window-level features. Two features were added to the sensor streams: current time (minutes since the start of the day) and time since the last eating event (minutes) and number of eating events in that day. In the feature selection step, the most relevant features were selected (e.g. the location features were not beneficial and merely represented noise). In order to train an "About-to-Eat" moment classifier, signal processing and machine learning were used on the sensor streams. The classifier was trained to recognise the two classes "About-to-Eat" and "Not-About-to-Eat". The average recall of this model resulted in a recall, precision and F score of 0.77, 0.67 and 0.69 (respectively).

According to Han, Pei, and Kamber (2011)[306-307], precision and recall are measures used in classification. Precision describes the exactness (how many of the positive selected items are positive), recall defines the completeness (how many positive items have been correctly selected). The F-score (or F_1 score) is a measure that combines precision and recall (harmonic mean).

Rahman et al. (2016)'s classifier's performance was further inspected, in order to see how the different feature extraction window sizes affected it. The small window sizes were vulnerable to noise and the coarser ones were unable to detect recent events, which were crucial in creating the classifier. As can be seen in figure 1, the smaller window sizes have a higher gap between precision and recall. The performance with window sizes between 60 and 80 minutes is higher and more consistent.

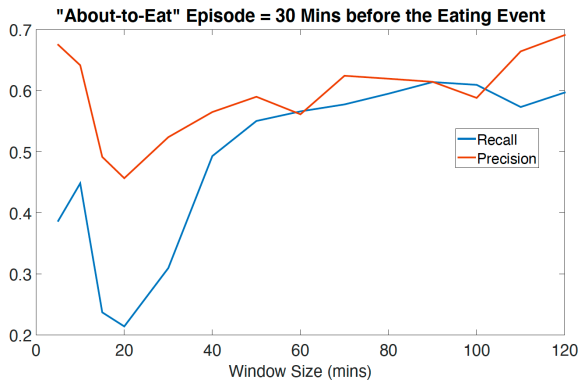


Figure 1: This graph depicts the F measure with the different feature extraction window sizes of the "About-to-Eat" classifier.

The performance for the "Time until Next Eating Event" model gained a correlation coefficient of 0.49. The most fitting feature extraction window size was assessed, the best performance was reached with a window size of 100 minutes. The authors further claim, that both models could be improved by incorporating person-dependent data from the target user to the models (e.g. person-specific eating pattern, lifestyle).

Stütz et al. (2015)[240, 242-249] research, whether data collected by a user's smartphone can be used to predict stress. In their study, they used an android smartphone app called "TheStress-

Collector" (TSC) to collect smartphone usage and sensor data in the background. The following data was collected:

- Activity: whether the device on the user is on foot, on a bike, in a vehicle, tilting, still or unknown.
- App Usage: apps for information, system, health, entertainment, social, or work
- Network Traffic: amount of network traffic (received and transmitted)
- Reboot Activity: power on and off events
- Calls: min, max and mean dB-values collected by the microphone
- Light: environment brightness
- Messages: timestamps of received messages
- Noise exposure: min, max and mean dB-values
- Screen Activity: duration of a user session (between screen power on and off)

15 participants installed the app on their smartphone and took part in the conducted study for two weeks. Seven times a day they would fill out questionnaires (at set times), which included the perceived stress score (PSS) and queried them on their current stress status. The stress levels were predicted by WEKA's machine learning algorithms. The mean absolute error (MAE) and Pearson correlation were used for the evaluation. The results of this study presented compelling correlations between PSS and the data collected by the smartphone app. The weekly PSS average, as opposed to the daily one, included the highest correlation coefficient. Thus, it is expected, that it easier to spot longer periods of high stress than shorter ones. This paper also a team publication featured in the research project from which SmartEater was developed.

In their experiment, Ameko et al. (2018) intend to predict user's negative affect states, in order to provide targeted just-in-time mental health interventions. 65 students participated in the study. Through a smartphone app, GPS location data, accelerometer (activity) data, phone calls, SMS, and ecological momentary assessments (EMA), data was collected and used to cluster participants according to their behavioural profiles. Grouping the participants this way seemed to improve the predictive model's performance.

Other related work: Pius Owoh, Mahinderjit Singh, and Zaaba (2018) use the k-means clustering algorithm to create automatic annotation of unlabelled smartphone sensor data, to observe sensitive location information of mobile crowd sensing users. Sornbootnark and Khoenkaw (2019) use smartphone sensor data (e.g. accelerometer) to predict excessive alcohol consumption, in order to replace breathalysers. Their algorithm produced 100% accuracy, however with 15 minute lags.

3 Literature Review

3.1 Data Mining

Larose and Larose (2015)[4] declare, that data mining is used to recognise patterns and trends in large amounts of data. Han, Pei, and Kamber (2011)[16-18] explain, that the term ”data mining” is a misnomer. A more suitable phrase would be ”knowledge mining from data”. The word ”mining” represents valuable nuggets found within large amounts of raw material. Other names used to describe the same process include: knowledge discovery from data (KDD), knowledge extraction, data/pattern analysis, data archaeology, and data dredging. The discovery of data is an iterative process represented in the following steps: Data cleaning, data integration (combine multiple data sources), data selection (relevant data is extracted), data transformation (into applicable forms for data mining), data mining (discover patterns), pattern evaluation (determine if patterns have a meaning), and knowledge presentation. The following data forms, are typically used for mining: database data, data warehouse data, and transactional data. Other forms include data streams, ordered/sequence data, graph or networked data, spatial data, text data, multimedia data, and the World Wide Web. As stated by Larose and Larose (2015)[9-13, 15-16], data mining requires continuous human supervision for quality monitoring and evaluation. Software alone will serve wrong results. Data mining is used for description of patterns and trends, estimation of numerical values, prediction of future results, classification of categorical variables, clustering of similar objects and association of attributes.

Larose and Larose (2015)[160] describe the two types of data mining methods: *supervised* and *unsupervised*. Han, Pei, and Kamber (2011)[363] interpret supervised learning as *learning by examples*, whereas unsupervised learning is *learning by observation*. Larose and Larose (2015)[160-163] continue, the majority of methods are supervised. In supervised methods, there is a predefined target variable. The method receives several examples, where the target variable value is defined, thus learning which values of the target variable correspond to which values of the predictor variable. The goal of the unsupervised approach is to find patterns and structure in the inserted variables. Therefore, no target variable is established. Clustering is the most prevalent unsupervised method. As reported by Han, Pei, and Kamber (2011)[32], through using unsupervised machine learning, it is possible to detect classes within data.

As stated in Larose and Larose (2015)[160-163], data dredging is a problem in data mining, that arises when false results develop in data mining due to random variations of data. Cross-validation is used to prevent data dredging, by guaranteeing that the results can be generalised to an independent data set.

Other problems in data mining include underfitting and overfitting. In their paper on attempting to balance underfitting and overfitting, Aalst et al. (2010)[87-89] clarify when these can occur. When fitting a model to a log, underfitting or overfitting the model are main problems. Underfitting the model means not fitting the model well enough to the log, therefore allowing too much behaviour which is not present in the log (”anything is possible”). Overfitting has the opposite effect. The model is fitted too well to the log, thus reducing it to another depiction of the log. It therefore only permits the same paths present in the log. These problems can take effect, for

example, in a hospital process. Each patient's path of events is most probably unique, two patients are unlikely to have the same process. Hence, the event log is always growing, therefore generalisation is required to avoid overfitting. Likewise, underfitting (allowing all possibilities) should also be averted. Larose and Larose (2015)[164-165] mention, that another way to describe the overfitting/underfitting problem is through the bias-variance trade-off. In figure 2 there are three scatter plots with data points in two different colours, which need to be separated by a line. The first scatter plot depicts a low-complexity separator (e.g. a straight line), which may have some classification errors (*high bias*), however it needn't change much to accommodate new data points. Therefore, it has *low variance*. The second scatter plot illustrates a high-complexity separator (e.g. curvy line that can separate more of the points correctly), which reduces the amount of errors (*low bias*), but has to change a lot when new data points are added. Thus, it has *high variance*. The higher the complexity of the model gets, the bias is reduced, the variance however increases. The third scatter plot shows both a low-complexity separator and a high one, with additional data. While the low-complexity separator can can still classify with little change, the high-complexity one needs to be altered more. The ideal model has neither high bias or variance.

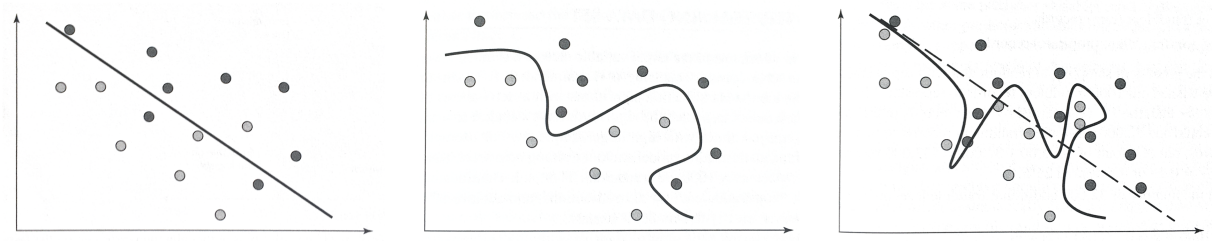


Figure 2: These three scatter plots from Larose and Larose (2015)[164-165] depict the bias-variance trade-off. The first plot portrays a low-complexity resulting in a high error rate. The second plot achieves a low error rate by using a high-complexity separator. The third graph compares the two separators.

3.2 Data Preprocessing

In his book, McCue (2014)[53] mentions that generally in data mining, the analysis itself on requires 20% of the time. The other 80% is the preparation of the data.

To make data useful in data mining, Larose and Larose (2015)[20] point out, that data sets first need to undergo a data preprocessing step, including data cleaning and data transformation. Raw data extracted directly from databases can be incomplete (values are missing), noisy (contains outliers), or may contain out-dated or redundant data. García, Luengo, and Herrera (2015)[45] list the following sources of dirty data: data entry errors, data update errors, data transmission errors and bugs. Dirty data can impact the produced model, making it less reliable. The significance of its effect depends on the implemented data mining method. Larose and Larose (2015)[20] define the goal as to decrease garbage in, garbage out (GIGO). To clarify,

decreasing *garbage in* means to reduce the irrelevant data that is fed into the model, thus reducing the amount of irrelevant data received out of the model (*garbage out*). Further information to different types of dirty data and their solutions are outlined in the next sections below.

3.2.1 Noisy Data

Pyle (1999)[71-72] interprets outliers as objects that have low recurrence and that are separated from the main collection of values. These values are often mistakes and can lead to distortion of the data set. Insurance companies provide a good example of outliers. The majority of insurance claims are only for a small sum, however every so often a customer may be in need of a large claim (outlier). Han, Pei, and Kamber (2011)[28-29] mention, that in some cases, these uncommon events are of more interest. One of these instances is detecting unusually large payments compared to the card holders normal payments, to uncover fraudulent usage of credit cards. As stated by Larose and Larose (2015)[26-27], there are some data mining algorithms that have trouble functioning correctly when fed outliers. Moreover, outliers may be data errors. Graphical methods used to identify outliers include, histograms or two-dimensional scatter plots. In order to smooth the data, Han, Pei, and Kamber (2011)[84] use binning, regression and outlier analysis (e.g. clustering).

As reported by Larose and Larose (2015)[45], ID fields should be removed from the data set used in data mining algorithms, since the value is different for each record and not helpful. It could however prove harmful, since a relationship might be presumed that isn't there.

3.2.2 Missing Values

According to Pyle (1999)[81-82, 260, 264, 267], it is good practice to differentiate "empty" values from "missing" ones. Empty values do not have a comparable real-world value. Missing values, however, do have underlying values, they simply weren't recorded. The author does not recommend ignoring the record with the missing value, since it would mean wasting the data stored in the other fields of that record. These fields may contain relevant information. Substituting the value, means that the record can be used. One of the problems with not having these values, is that this missing information content (e.g. predictive or inferential) could be carried by the pattern. Another consideration is how to substitute the missing value, without adding bias to the data set. An inadequately chosen replacement value could distort the data set, by adding data which doesn't exist in the real world. A crucial focus is reserving the relationship between variables. Substitution of values, if not suitable, may disrupt the between-variable variability, thus hiding or distorting patterns in the data.

Larose and Larose (2015)[23, 25] give an example, of how replacing missing values can lead to invalid results. The authors experimented with a database of cars. Substituting a missing brand with a random value (here "Japan") led to a car, that didn't even exist. Data imputation takes into account the other attributes stored in the record and from these, calculates what the missing value would most likely be. Larose and Larose suggest, that the value can be replaced, either with a constant determined by the data analyst, with a field mean (for numerical values) or mode

(for categorical values), with a random value, or with imputed values based on the different features of the record. Pyle (1999)[260, 267-269] points out, that regression can be used to find supplement values. When using regression (e.g. linear regression), one can calculate a value with the help of another given value. There are several different methods to replace missing values. Some of which promise to generate more information. Such methods however are computationally complex.

3.2.3 Normalisation

Han, Pei, and Kamber (2011)[105-106] describes normalisation as giving the attributes of a data set equal weight. For example, it can transform the data to fall in a smaller, common range (e.g. [-1, 1]). It therefore hinders variables with large ranges from outweighing ones with smaller ranges. Income would, for instance, have a larger range than binary attributes.

García, Luengo, and Herrera (2015)[46-48] explain, that raw data is often transformed to produce new attributes with more applicable properties in the process of normalisation. These new attributes are then known as *modeling variables* or *analytic variables*. *Min-Max Normalization*, *Z-score Normalization*, and *Decimal Scaling Normalization* are methods that convert the distribution of the existing attributes. For the following examples, A is a numerical attribute from a data set, a single value of this attribute is represented with v:

- Min-max normalization scales the original numerical values to a newly defined range, with a new minimum ($newMin_A$) and maximum ($newMax_A$) (e.g. 0.0 and 1.0). The original minimum and maximum values found in A are presented as min_A and max_A respectively:
$$v' = \frac{v - min_A}{max_A - min_A}(newMax_A - newMin_A) + newMin_A$$
The intervals [0, 1] and [-1, 1] are common intervals for normalisation.

- Z-score (or zero-mean) normalization normalises the values using the mean ($\bar{A}$) and standard deviation σ_A of the values A.
$$v' = \frac{v - \overline{A}}{\sigma_A}$$
After this transformation, the mean equals zero and the standard deviation is one. The advantages of this normalisation method take effect, when the min and max values of A are not known, or when there are outliers that could bias the min-max method.

- The decimal scaling method moves the decimal point enough spaces, so that the maximum absolute attribute value of A is below one. The smallest required number of digits to move the decimal point, so that the largest absolute number in A is below zero, is represented by j:
$$v' = \frac{v}{10^j}$$

3.3 Dimensionality Reduction

Bellman (1957)[20-22] first introduces the *curse of dimensionality*. The curse effects a mathematical model, when there are a large number of variables. The real world is complex and by trying to incorporate as many real world features into a mathematical model as possible, it becomes complicated. A too simple model, however, will not be suitable for prediction. Bellman (1961)[94] further details the results of *the curse of dimensionality*. Functions with one variable can be visualised as curve in a 2D space and a function with two variables in a 3D space. Depicting functions with more variables however, is more problematic (both for visualisation and tabulation). According to Larose and Larose (2015)[93], high quality visualisation methods usually cannot depict more than five dimensions. Bellman (1961)[94, 198] further gives the following example: Imagine if the variables of a function take on the values between 1 and 100. While a function with one variable would need to tabulate 100 values, a function with 2 variables would need to tabulate 100 x 100 = 10^4 values and a function with 3 variables 10^6. Each additional variable adds more complexity.

According to Larose and Larose (2015)[92, 93], a high amount of predictor variables in data mining can lead to overfitting and overlooking crucial relationships between predictors. Dimensionality reduction techniques have the ability to reduce the number of predictor items, aid in ensuring that these predictor items are independent, and present a framework for interpretability of the results. As stated by Han, Pei, and Kamber (2011)[93], dimensionality reduction is a data reduction method. Data reduction is utilised to attain a smaller, more concentrated data set, whilst mostly keeping the integrity of the initial data set. Principal components analysis is a dimensionality reduction technique.

3.3.1 Principal Components Analysis (PCA)

Principal Components Analysis was first proposed by Pearson (1901) and Hotelling (1933). Pearson's approach is to identify a line or plane that best fits the collected variables plotted to a plane. In order to determine the best fitting line or plane, means, standard-deviations, and correlations are used (Pearson 1901)[559-560]. Hotelling (1933)[5] introduces his method as *the method of principal components*. In his paper, Jolliffe (2002)[7] clarifies, that while these two papers used different methods, the standard algebraic derivation was announced by Hotelling (1933).

Han, Pei, and Kamber (2011)[95-96] lists the first step of PCA is to standardise the input data, therefore making the data-range identical. Larose and Larose (2015)[94] declares, that after standardising the data, the mean is zero and the standard deviation is one. Han, Pei, and Kamber (2011)[95-96] describes the next step, which entails calculation k orthonormal vectors, the so called *principal components*. These unit vectors present a basis for the input data, which are a linear combination of the principal components. Larose and Larose (2015)[94] explain, that the principal components can be discovered, by rotating the initial coordinate system to the direction of maximum variability. These then create a new coordinate system.

In the following step, as stated by Han, Pei, and Kamber (2011)[95-96], the principal components are selected.

Hotelling (1933)[4, 5, 15, 18] When choosing the calculated components, they are chosen with the decreasing amount of variance. Therefore, the one with the highest variance (γ_1) is chosen first. The next highest (γ_2) is chosen orthogonal to γ_1 and so on, until the number n dimensions are reached (γ_n). The components left with small variance are disregarded, since they are trivial.

Han, Pei, and Kamber (2011)[93, 95-96] stated, in data mining the vectors with the lowest variance that are removed, reduce the amount of data and number of dimensions. Despite the loss of data, the components with higher variance can approximate the original data. The authors suggested wavelet transforms (e.g. discrete wavelet transform (DWT)) as another method of dimensionality reduction.

3.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

In their paper, Maaten and Hinton (2008)[2579-2580] introduce t-SNE. It is a method of visualising high dimensional data on a two or three-dimensional plot. This technique is also used to reduce the number of dimensions. The authors mention, that while linear methods like PCA concentrate on making sure low-dimensional representations of data points apart from each other, they are typically unable to bring similar low-dimensional representations near. The authors describe, that t-SNE is a variation of Stochastic Neighbor Embedding (SNE), introduced by Hinton and Roweis (2003).

Maaten and Hinton (2008)[2581] explain that the first step in SNE is to transform the Euclidean distances between points in high-dimensional space into the conditional similarity probabilities. According to Hinton and Roweis (2003)[2], this probability $p_{i,j}$ would arise from the similarity between the two datapoints x_i and x_j. This is the probability of x_i picking the point x_j as its neighbour. Maaten and Hinton (2008)[2581] clarify, that this is the probability of x_i picking the point x_j as its neighbour under the circumstance that the neighbours were chosen according to their probability density under a Gaussian, x_i being its centre. This results in the conditional probability being high for close data points and imperceptibly small for those further apart. In a similar way, the conditional probability between the low-dimensional data points y_i and y_j (counterparts to high-dimensional x_i and x_j), represented by $q_{i,j}$ is calculated. Maaten and Hinton (2008)[2581] state, that if the similarity of x_i and x_j has been correctly mapped by y_i and y_j, then $p_{i,j}$ will be equal to $q_{i,j}$.

Hinton and Roweis (2003)[2] further explain, that the goal is to reduce the difference between $p_{i,j}$ and $q_{i,j}$, thus finding a suitable low-dimensional representation of the high-dimensional data. This is accomplished by minimising a cost function, comprised of the sum of the Kullback-Leibler divergences between $\mathrm{p_{i,j}}$ and $q_{i,j}$. Maaten and Hinton (2008)[2583] then list the advantages of t-SNE over SNE are the improvement of its cost function (symmetrised version of SNE) and the use of Student-t distribution as opposed to Gaussian, for the similarity calculation in the low-dimensional area.

To apply t-SNE to a data set, Maaten and Hinton (2008)[2582] describe the σ_i that has to be chosen for variance of the Gaussian centred over x_i. Depending on the density, different values of σ_i provide better results. For example, in a denser region, a smaller σ_i is recommended, but not in less denser regions. This makes it unlikely to be able to find a overall optimal σ_i for the

data set. Hinton and Roweis (2003)[2] declare that σ_i can be chosen by hand or is found using a binary search. This would equalise the distribution entropy to log$k$, k being the effective number of local neighbors or the so called perplexity. The perplexity is a parameter chosen by the user for the SNE by hand. Maaten and Hinton (2008)[2582] imply, that this parameter should be chosen between 5 and 50 and that SNE is quite robust to the chosen value.

3.4 Cluster Analysis

Hartigan (1975)[1] describes clustering as a means to group similar objects together. For example, two planets are considered similar, if (given measurement error) it is probable they could be perceived as the same planet. **romesburg2004cluster**[2] gives the gathering of a variety of pebbles and sorting them into piles of similar attributes (e.g. shape, size, colour) as an example of cluster analysis. Hartigan (1975)[1-3, 6] further explains, that it can be expected from similar objects, for them to act and be treated the same. Clustering is also used to name, display, summarise, predict, and require explanation of the objects in the cluster. If some of the objects assigned to a cluster exhibit certain properties, it is expected that the other objects in this cluster will exhibit them as well. Clustering is almost equivalent to classification. Real-world examples of clustering include classifications of animals, plants and diseases.

Han, Pei, and Kamber (2011)[361-363] state, that cluster analysis is another term for clustering. It divides a data set of objects into subsets (clusters). The objects placed into one cluster are dissimilar to the objects assigned to other clusters. Therefore, such a cluster can also be defined as an implicit class. For this reason, clustering is occasionally referred to as automatic classification. The fact that cluster analysis can find groups by itself, gives it its unique advantage. As mentioned in section 3.1, clustering is a type of unsupervised machine learning. It is unsupervised, since the class label for each group is unknown and needs to be discovered. In data mining, it is utilised to understand the distribution of the data and inspect the distinctions between clusters. Moreover, it can be used as a preprocessing tool for other data mining methods, such as characterisation, attribute subset selection, and classification. Cluster analysis is used in various fields, including: biology, security, business intelligence, image pattern recognition, and web search. It can be used to place customers into groups, organise projects into categories in project management, and to sort web search results into concise groups. Furthermore, it can be used to detect outliers, since these are located outside of clusters (as pointed out in section 3.2.1).

According to Hartigan (1975)[9-10], there are usually five different types of variables used in practice in clustering:

- Counts: no arbitrary scale (e.g. number of legs on a spider)
- Ratio scale: only defined in proportion to a standard volume (e.g. volume of a liquid in a glass)
- Interval scale: chosen from a standard position in a standard unit (e.g. height of a building)

- Ordinal scale: ordered classification, can be changed by a monotonic transformation (e.g. socio-economic status)
- Category scale: classification that can be adjusted by a one-to-one transformation (e.g. religion)

A data set can be comprised of various variable types (*mixed*), of the same type but with different ranges (*heterogeneous*), or of variables with the same range (*homogenous*). There are also methods for conducting type or scale conversions.

Larose and Larose (2015)[524-525] explain, that data should be normalised before being put into a clustering algorithm, thus optimising the performance. Min-max normalization or Z-score standardization can be used to do so (see section 3.2.3).

3.5 Overview of clustering algorithms

Han, Pei, and Kamber (2011)[363-365] list the following requirements clustering methods must meet:

- Scalability: clustering algorithms need to work on large databases, which may contain millions or billions of entries.
- Ability to work with different attribute types: The algorithm must be able to handle various data types, for example: binary, nominal (categorical), and ordinal data. More complex data types include graphs, sequences, images, and documents.
- Recognising clusters with arbitrary shapes: Methods that use distance measures (e.g. Euclidean or Manhattan) to compute clusters usually find clusters of spherical shape. The size and density also tend to be similar. Clusters could however be of any shape, therefore the algorithms need to be capable of detecting any shape.
- Requirements for domain knowledge: For some clustering algorithms, parameters (e.g. desired number of clusters) need to be determined. These can affect the cluster results. Parameters are hard to define, if the data is not understood.
- Ability to handle noise (see section 3.2.1)
- Incremental clustering: The method should be able to integrate incremental data updates into existing structures, without recomputing the clustering.
- Insensitivity to the order of the input: The clustering results should be the same, regardless of the order the objects are inserted into the algorithm.
- Ability to cluster high-dimensional data (see section 3.3)
- Capability to cluster under certain constraints

- Interpretability and usability of the results

todo: add images to compare how different methods cluster Han, Pei, and Kamber (2011)[366-396] present different clustering algorithms. They state, that it is not easy to divide these into distinct categories, since some algorithms share features from other categories. The general categories are partitioning methods, hierarchical methods, density-based methods and grid-based methods.

(**Todo: Explain used methods in more detail after Experiment and add figures of graphs**)

3.5.1 Partitioning Methods

Partitioning methods are the easiest and most significant types of clustering methods. The data is divided into k (generally pre-defined) number of groups (clusters). The data consists of n objects, thus $k \geq n$. Each group must contain at least one object. A data object can only be classified into one group (*exclusive cluster separation*). Fuzzy partitioning methods relax this condition. Many of the partitioning methods use distance measures to calculate their clusters. If the number of clusters (k) is pre-defined, then the clustering algorithm will create an initial segregation into k clusters. Objects are then relocated to improve the partitioning. The partitioning is considered good, when objects assigned to the same cluster are ”similar” and ”dissimilar” from the objects in the other clusters. Traditional partitioning methods can also be applied onto subspaces (for many attributes and sparse data).

Examples: k-means, k-medoids

3.5.2 Hierarchical Methods

The data is grouped into a hierarchy (”tree”) of clusters. Depending on how the hierarchical decomposition is constructed, there are two different approaches: *agglomerative* or *divisive*. In the *agglomerative* or *bottom-up* approach, each object creates its own cluster. Step by step it is then merged into its closest neighbours until all objects belong to one cluster, or a termination condition comes true. In the *divisive* or *top-down* approach, all objects initially form one cluster together. Step by step, each cluster is divided, until each object is contained in its own cluster, or a condition is met to terminate the process. Once a merge or split step has been performed, it cannot be reversed. Once merged/split, the objects also cannot swap cluster. Each merge or split decision influences the quality of the resulting clusters and must therefore be well chosen. Hierarchical methods can be used in subspaces and can use distance measures, or can be density- and continuity-based.

Examples: BIRCH, Chameleon

3.5.3 Density-Based Methods

The majority of clustering methods (e.g. partitioning and hierarchical methods) use distance-based approaches, which results in only finding clusters with spherical shapes. Density-based

methods have the ability to find clusters with random shapes. In these methods, objects are continuously added to the cluster, so long as the number of objects/data points (density) close by is larger than a given threshold. The clusters are comprised of high-density areas of objects. These are separated by spaces with low-density. Accordingly, this method is also useful for removing noise and outliers. These methods can also be used to cluster sub spaces.

Examples: DBSCAN, OPTICS, DENCLUE

3.5.3.1 DBSCAN

Ester et al. (1996)[226-229] introduce a new density-based clustering algorithm, the Density Based Spatial Clustering of Applications with Noise. This method is able to find clusters no with different shapes and work efficiently on large spatial datasets. The algorithm searches in a give radius (Eps = epsilon parameter) around a data point. If within this radius a minimum number of points (MinPts parameter) exist, then this point is added to the cluster (core point). A data point (p) is considered a border point, if inside of the Eps neighbourhood there is a core point (q). A data point is labelled as noise, if it does not belong to any clusters (has no core points within the given radius). According to Han, Pei, and Kamber (2011)[388], a weakness of DBSCAN is the fact that the results rely on the chosen parameters. If these are selected differently, the clustering results can differ. These parameters can often be challenging to select.

Ester et al. (1996)[230] supports the selection of the Eps and MinPts parameters. The idea is to select the appropriate parameters for the ”thinnest” cluster. The first step is to construct a sorted k-dist graph. For a specific k, calculate the distance d of every point p to its kth nearest neighbour. Sort these distances in descending order and depict them on a graph. Such a graph can be seen in figure 3. The goal is to locate the *threshold* of the highest distance to the kth nearest neighbour (the ”thinnest” cluster). The first point in the ”valley”, as can be seen at the tip of the arrow in figure 3, is this threshold point. The points to the left with higher distances are likely to be noise and the points after are part of clusters. The authors suggest selecting 4 for k. Experiments have shown that the results for higher values for k do not vary greatly. They therefore recommend defining MinPts as 4 and using a 4-dist graph to estimate the Eps parameter.

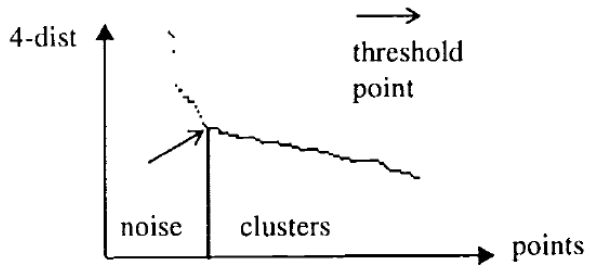


Figure 3: Sorted 4-dist graph (distance for each point to its fourth nearest neighbour) (Ester et al. 1996)[230].

3.5.3.2 OPTICS

Han, Pei, and Kamber (2011)[388] mentions, that OPTICS was create d to improve the selection of global parameters problem in DBSCAN. Ankerst et al. (1999)[49, 51-] present the density base clustering algorithm OPTICS (Ordering Points To Identify the Clustering Structure). This method in itself does not specifically create clusters. Instead, it orders the data set according to its density-based clustering structure. For each object, the values *core-distance* and *reachability-distance* are calculated. The *core-distance* of an arbitrary data point (object) *o* is the distance to the nearest point within Eps that completes the MinPts rule and therefore labels point *o* as a core point. If there is not the number of MinPts in the Eps neighbourhood, then the *core-distance* of that point is undefined. The *reachability-distance* of an object *p* to core object *o* is defined as the max value of the core-distance and the distance from object *o* to object *p*. Likewise, if *o* is not a core object, then the reachability-distance of *p* is undefined. The *core-distance* and *reachability-distance* are visualised in figure 4.

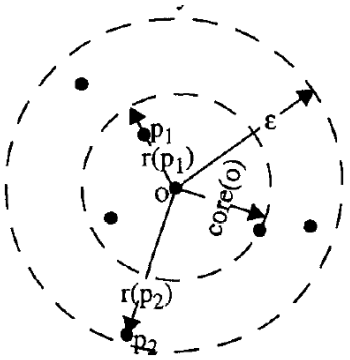


Figure 4: Core-distance of object *o* (MinPts = 4) and reachability-distances for objects *p1* and *p2* (Ankerst et al. 1999)[52].

The data points are ordered by the OPTICS algorithm (using their reachability-distance) to create a reachability plot. This plot is relatively stable towards the input parameters. In figure 5 a reachability plot calculated from a 2D data set is depicted.

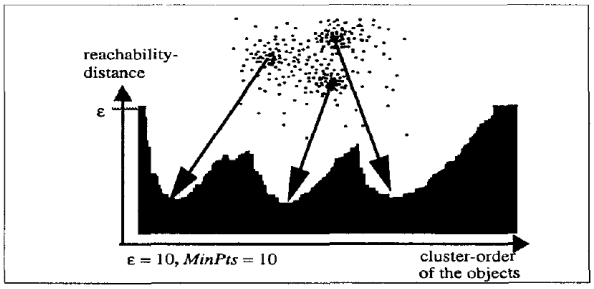


Figure 5: OPTICS reachability plot for a 2D data set. Three ”Gaussian bumps” can be seen (Ankerst et al. 1999)[54].

The clusters can then be automatically constructed from the reachability plot by pinpointing the start-of-cluster and end-of-cluster regions and combining regions that match into clusters

(or nested clusters). Since the reachability-distance of a point is the distance from the set of its predecessors and through OPTICS' specific ordering, the clusters are the dips in the reachability plots (as can also be seen in figure 5). In figure 6, a cluster could hence be extracted from this plot starting at the 3rd data point and ending with the 16th.

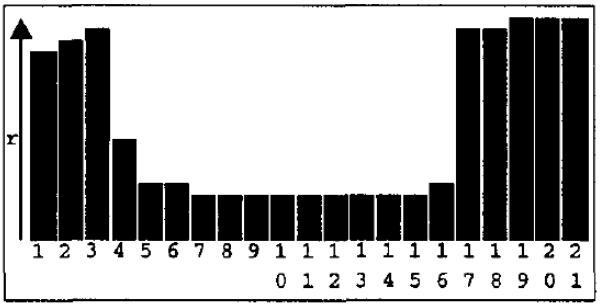


Figure 6: Since clusters are defined as the dips/dents in the reachability plots create by the OPTICS algorithm, a cluster can be extracted from this plot starting at the 3rd data point and ending with the 16th. The x-axis of the reachability plot depicts the order of the data points (objects) and the y-axis the reachability-distance for each datapoint (Ankerst et al. 1999)[57].

3.5.4 Grid-Based Methods

The previously mentioned clustering methods are data-driven (they accommodate the distribution of the data objects). Grid-based methods are space-driven (they do not rely on the distribution of the data objects). The data objects are quantised into grid cells on a multiresolution grid. The actions required for clustering are performed on the grid structure. The processing time depends on the grid size (number of cells) in each dimension and not on the number of objects and is more accelerated than other clustering methods.

Examples: STING, CLIQUE

Han, Pei, and Kamber (2011)[414, 416] clarify, that the clustering methods mentioned above have a good functionality when used on a data set with fewer than 10 attributes. Another way to cluster high-dimensional data is *subspace clustering*, in which subspaces (subset of attributes) are investigated to find clusters. The CLIQUE method is used for subspace clustering.

3.6 Evaluating clustering results

The resulting clusters received from the previously mentioned clustering algorithms are assessed in the *cluster evaluation* step. Han, Pei, and Kamber (2011)[396] describe this stage as assessing the quality of the results. There are different steps to be taken in evaluating clusters.

3.6.1 Assessment of the cluster tendency

As explained by Han, Pei, and Kamber (2011)[396-397], the tendency must be assessed, meaning it is tested, whether structures exist that aren't random. Running a clustering algorithm on any data set will return clusters. However, only non random structures are significant and not misleading. For example, if a data set consists of data points that are uniformly distributed, if a clustering algorithm delivers clusters, these will be random and have no purpose. Spatial randomness tests (e.g. Hopkins Statistic) can be used to measure how likely the data was created by uniform data distribution.

3.6.2 Evaluation of the cluster quality

Han, Pei, and Kamber (2011)[399, 401] further mentions, that the cluster quality needs to be evaluated. Generally, there are two ways to measure the quality of clustering: extrinsic methods and intrinsic methods. In extrinsic methods, there is a ground truth available, therefore these are also referred to as supervised methods. This ground truth is usually produced by experts (humans). Intrinsic methods are used, when there is no ground truth available. In intrinsic methods, the clusters are evaluated by how well they are separated from one another and how compact they are (e.g. *silhouette coefficient*). The experiment described in this paper uses the intrinsic method silhouette coefficient, since there is no ground truth for comparison.

3.6.2.1 Silhouette Coefficient

In his paper, Rousseeuw (1987)[53-57, 59] proposes a new graphical display using silhouettes, to help determine how well objects belong to their assigned clusters. It can be used to interpret and validate the results of clustering. It is also utilised to compare the resulting clusters with those output using alternative algorithms (the input data being the same). In an example, where countries are assigned a value of how dissimilar they are to another country, the results are listed in a table. A structure contained in the results (consisting of 66 numbers), is hard to identify. Therefore, the countries are categorised into clusters using k-median. It is however uncertain, whether the clusters follow a specific structure, or if the groups are simply artificial. With the use of silhouettes, the author's goal is to answer the following questions: Is the quality of the clusters high, therefore the dissimilarities of the objects within a cluster are small, and large compared to the objects in other clusters? Are the objects well-classified, misclassified and which ones were not classified (between clusters)? Is it possible to perceive which "natural" clusters are available in the data set?
According to Rousseeuw, the silhouettes are ideal when the distance between objects are on a ratio scale (e.g. Euclidean distances) and when the goal is to receive clear and compact clusters. For each object i (in cluster A) the value $s(i)$ is calculated. $a(i)$ contains the average dissimilarity ob the object i to each other object in the same cluster. If there are no other objects in the cluster, $s(i)$ is set to zero (most neutral value). $b(i)$ is determined, by firstly calculating the average dissimilarity, for each neighbouring cluster that isn't A. The shortest of these values, therefore the next closest cluster to A, is then assigned to $b(i)$. This cluster can so to say be seen

as the next best choice for *i*. *b(i)* can only be calculated, if there are other clusters beside *A*. The formula is as follows:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

The resulting value *s(i)* is a number in the range of *-1* $\leq$ *s(i)* $\leq$ *1*:

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

A *s(i)* value close to 1 reveals, that the dissimilarity within a cluster is smaller than the dissimilarity to the neighbouring cluster. Therefore it suggests, that the assignment of that object is good, since it is the most likely the most suitable cluster for *i* (well-classified). A *s(i)* value close to 0 means that *a(i)* and *b(i)* are almost equal and it is unclear whether *A* or the neighbouring cluster is a more suitable fit. If *s(i)* is close to -1, then the dissimilarity within a cluster is larger than the dissimilarity to the neighbouring cluster. Thus, it would have been more natural to assign *i* to the neighbouring cluster, since it is closer to it (misclassified). The function can also be adapted to work with similarities. The *average silhouette width* for each cluster is received by calculating the average of all objects that belong said cluster. The higher the average silhouette width, the more pronounced the cluster is. An average score can also be calculated from each object *i* for the entire plot (data set), the so called *overall average silhouette width*.

Aranganayagi and Thangavel (2007)[15] use the silhouette coefficient in their proposed clustering algorithm, which clusters categorical data. The coefficient was used assess the quality of the clusters and to relocate objects to more fitting clusters. The cluster efficiency in their algorithm was therefore enhanced.

3.6.2.2 Davies-Bouldin Index

A second cluster evaluation method is the Davies-Bouldin Index, which was announced by Davies and Bouldin (1979)[224-227].

The succeeding formula describes the average similarity of a cluster with the cluster that is most similar to it (R_{ij}). *i* and *j* represent the determined clusters,S_i and S_j stand for the dispersions of the clusters, and M_{ij} is the distance between the two cluster centroids.

$$R_{ij} = \frac{S_i + S_j}{M_{ij}}$$

The Davies-Bouldin Index equals:

$$\overline{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

This metric can be used, to compare clustering results. The lower of the two $\bar{R}$ values indicates the better partitioning.

3.6.2.3 Caliński-Harabasz Index

As a third evaluation score, Caliński and Harabasz (1974)[3, 7, 11, 23, 24, 26] is used to evaluate and compare the resulting clusters in the experiment. The Caliński-Harabasz Index or Variance Ratio Criterion (VRC) is calculated as

$$VRC = \frac{BGSS}{k-1} / \frac{WGSS}{n-k}$$

If k is not known, it is set to 2, then 3 and so on. The density of the clusters can be calculated with the sums of the squared distances from the cluster centroids, to the points. A higher VRC result indicates the "best number" of clusters (k).

3.6.3 Establishing the number of clusters

Han, Pei, and Kamber (2011)[398] states, that the number of clusters (k) found in the data set needs to be established. For some clustering methods (e.g. k-means), this number is defined before the clustering process. This number can be challenging to determine and depends on the shape and scale of the input data. A good number of clusters creates a balance between *compressibility* and *accuracy*. Having only one cluster would have maximum compression, but no value. Contrarily, if each data object formed its own cluster, the clusters would be most accurate, but not allow for summarisation of the data. Rousseeuw (1987)[59] describes, how silhouettes can be used to determine the ideal amount of clusters. Picture a data set with dense clusters which each have large distances to the other clusters. When k is chosen too small, naturally occurring clusters must be artificially joined, to satisfy the value k. Implementing the silhouette calculation will result in high within-cluster dissimilarities (*a(i)*) leading to a narrow silhouette (small *s(i)*). Likewise, if k is chosen too large, natural clusters will have to be artificially split, in order to gain k clusters. The objects in a split natural cluster will however still be very close to the other half of their cluster, therefore resulting in low dissimilarities between clusters (*b(i)*) and a small *s(i)*. This logic denotes, that silhouettes should be capable of finding the most 'natural' number of clusters in a data set.

4 Method

The goal of this paper is to identify, which time delta for aggregation is ideal to construct distinct clusters from smartphone sensor and usage data. The data for this experiment was collected for the upcoming SmartEater [5] mobile health app. The goal of this app is to present the user with content-dependent feedback, with the hope to prevent food craving episodes. By evaluating the user's behaviour (through smartphone sensor and usage data), the app predicts eating crises (through stress), therefore eliminating the need of intense user input.

5. https://sites.google.com/site/eatingandanxietylab/resources/smarteater

Various sensor and usage data was recorded for the SmartEater project, by the 46 testers' smartphones (for different periods of time). The columns of the data were organised as follows (N is the number of times the data was recorded in the time period):

TODO: Explain what accelerometer, ... is

- TIME: timestamp, when the data was aggregated (format: YYYY-DD-MM hh:mm:ss)
- ACC (1-N): values received from the accelerometer (average jerk)
- AUDIO (1-N): volume of the audio
- SCRN (1-N): percentage of screen on time
- NOTIF (1-N): number of notifications
- LIGHT (1-N): light sensor values
- APP_COM (1-N): app usage in the category *communication* in percent of lag-interval minutes (if communication apps were used for 5 minutes in a 15 minute interval, the value would be 0.66666)
- APP_VID (1-N): app usage in the category *video players*
- APP_OTHER (1-N): app usage of all other categories (excluding *video players* and *communication*)

The recorded smartphone sensor and usage data was aggregated into multiple .csv (Comma-Separated values[6]) files. Furthermore, these files were distinguished into folders, according to their time delta. Two different time lengths were used:

- 1h: The data was aggregated in 2.5 hour intervals, whereby each row contained data from an aggregation of 1 hour, in four 15 minute lags.
- 3h: The data was aggregated in 1.5 hour intervals, whereby each row contained data from an aggregation of 3 hours, in six 30 minute lags.

Each row contains the data value of a specific test user for one of the time periods (e.g. 1h or 3h).

Python was used to conduct the experiment, more specifically using the Anaconda[7] Python distribution platform for data science. The scikit-learn[8] (short sklearn) Python package provides simple tools for predictive data analysis and was used for data preparation, dimensionality reduction, and clustering.

6. `https://tools.ietf.org/html/rfc4180`
7. `https://www.anaconda.com/`
8. `https://scikit-learn.org/stable/index.html`

4.1 Preparation of the Data Set

Initially, the raw data was distributed into multiple .csv files (one file per user, per time interval). The files were read and processed by the Python Pandas[9] tool. The library offers fast and flexible functionalities for data analysis. It utilises DataFrames which are fast and efficient 2D data structures, used to store tabular data. Using the Pandas *read_csv*[10] and *concat*[11] methods, the files with identical time periods were transformed and concatenated to one collective DataFrame.

4.1.1 Missing Values

The raw data contained several rows with empty cells. Section 3.2.2 details approaches on how to substitute missing data, thus preserving the row, which could otherwise contain important information. In doing so however, the existing patterns could be disrupted. In order to maintain the data's patterns, rows with missing values were removed. Using the Pandas' *dropna*[12] function (deletes rows with missing values), all rows containing empty cells were dropped. Of the original 8283 rows from the 1 hour time period files, only 3279 (39.59%) rows were complete and remained. In the 3 hour files, only 6218 from 14091 (44.13%) persisted.

4.1.2 Normalisation

The data recorded values from the different smartphone sensors were spread over different number ranges. For example, whilst the values that describe the screen on time ranged between -20 and 2.0, the light sensor values could reach from 0 to over 60,000. As explained in 3.2.3, values with higher ranges can inadvertently outweigh smaller values. To be sure that the values are weighted the same, initially the sklearn *MinMaxScaler*[13] (Min-Max Normalization) was used to map all the values into the common range, e.g. [0,1]. Most of the values in the data set ranged between 0 and just over 100. The light sensor values were the exception, with its values reaching above 60,000. As mentioned in section 3.2.3, Z-score normalization is more robust to outliers, that could otherwise bias Min-Max Normalization. Therefore, the sklearn *StandardScaler*[14] was used instead of the MinMaxScaler, which implements Z-score Normalization.

9. https://pandas.pydata.org
10. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html
11. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.concat.html
12. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html
13. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
14. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

4.1.3 Selection of columns (attributes)

In order to only use meaningful data to receive significant results, it was important to remove columns that do not contain any predictive content. The TIME column was removed for this reason. Since the TIME column only showed the time and data when the data was recorded periodically (in fixed intervals), it was different for each row (per user) like an index. It could have however bias the results if left in.

To reduce the number of dimensions (number of columns), columns with that had recorded the same feature (e.g. ACC1-N), but at different time lags were compressed to one column. Each unique feature only requires one column, and therefore reduced the number of columns to only 8 (instead of 32 in the 1h data set or 48 in the 3h data set).

4.1.4 Chain shaped data

Initial visualisations of the data set (with t-SNE dimensionality reduction) showed a chain formalisation of the data. This chain can be seen in figure 7 (light blue data points, swirl like a snake). A similar accumulation of data points, though this time shaped as a line, can further also be seen in the PCA dimensionality reduced data set (see figure 8).

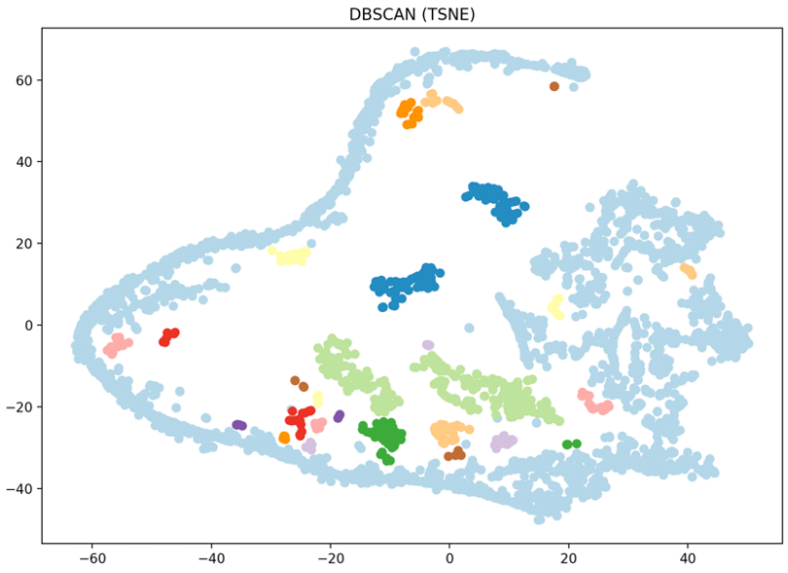


Figure 7: 1h data set, a chain of data points (light blue points) can be seen in the data set.

An initial idea, was that this chain was the result of the utilised t-SNE parameters. However, even with changing the t-SNE perplexity, learning rate and number of iterations parameters, the chain persisted in some form. Another thought, as to where these data points originated from or why they resulted in a chain, was that there were dependencies within the data or columns. For example, the three APP columns indicate the percentage of time in a lag-interval a certain

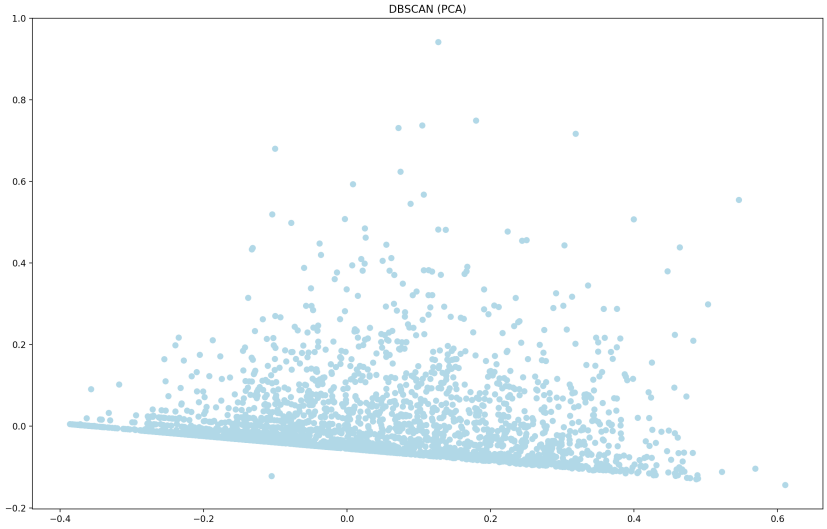
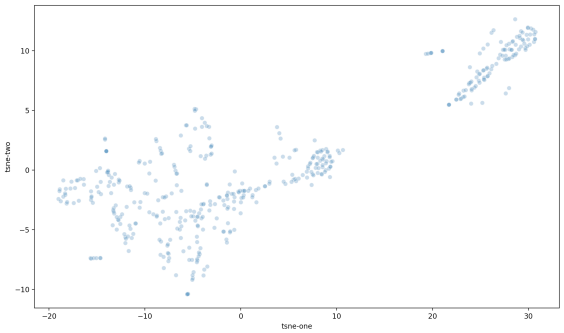


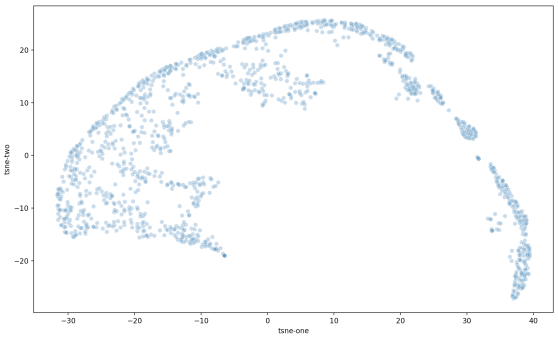
Figure 8: There is a collection of points along a line. This same collection can also be visualised as a chain in the t-SNE data set.

type of app was used. Therefore, if say a communication app was used 100% of this time, the APP_COM cell for this time would be 1. Using this app for 100% of the time would mean, that the values in the other APP columns for this time would have to be 0 (0%), since they would not have been able to be used. To test this theory, the APP columns were removed from the data set and the 2D scatter plot was recalculated. The chain was still visible. Step by step each column was removed, to see if it was the cause of the chain. The chain did not disappear though, which indicated, that the problem was not due to the columns, but rather the rows.

The chain was less visible when reducing the number of inserted .csv files (lower number of records) to 10 (see figure 9a). When the number of features were reduced to only AUDIO and ACC (two columns), the chain was once again visible (see figure 9b). The presumption was that when there were less data points and more features (e.g. 10 files but using all the features), there were enough different data points and the chain was less dense, making it less noticeable. When there were less data points but with less features (e.g. 10 files but using only 2 features), or when there were many data points even with multiple features (e.g. all data files and all the features), there were more rows with very similar or equal values and the chain appears more dense and prevalent.



(a) t-SNE calculated with only 10 data files and all features. The chain is not as visible.



(b) t-SNE calculated with 10 data files and only 2 features. The chain is visible.

Figure 9

The next theory, was that points from a same test subject were similar and might be causing the chain. To see which data points were contained in the chain, the row index number was added as a label to each data point in the scatter plot. Furthermore, each test user data file was assigned a different random colour, so that in the plot it was visible, which data points belonged to the same user. The resulting scatter plot is visualised in figure 10. From this plot it can be seen, that several data points from the same test subject cluster together in the chain, indicating similarity. There are however points from each user in different locations of the chart, not only in the chain. This implied, that the chain was not being caused by the subjects.

The ensuing step, was to use the data point indices to look for similarities in the values. Several rows found in the chain were extracted from the cleaned data set, directly before being fed into the t-SNE algorithm. After comparing these, it was evident that they shared a lot of the same feature values. Rows not in the chain however had different values. Looking at the same rows in the unclean data set (after concatenation of the .csv data files and removal of rows with missing values), rows found in the chain had several columns all with the value 0. Further inspection showed, that when the SCRN value was 0, LIGHT, NOTIF and the APP columns where also 0. When a smartphone screen is off, the apps are not being used, which explains why the APP columns were mostly 0. The NOTIF values were often 0, presumably because the user was not constantly receiving notifications. The environment light is also often 0 when a phone is not being used, maybe because it is in a pocket or bag. To test whether these rows were causing the chain to occur, rows with less than 50% of values that weren't 0 were removed. As can be seen in figure 11, the chain much less prevalent. The colours belonging to the test users are also more distributed. Naturally, this reduced the total number of rows left. As an example, the number of rows left for the 1h data files only using the first feature columns (15 min) was 1681 from initial 8283 (20.29%). In the 3h data set (first column, 30 min) only 3949 from 14091 rows remained (28.02%).

todo: name number of rows left

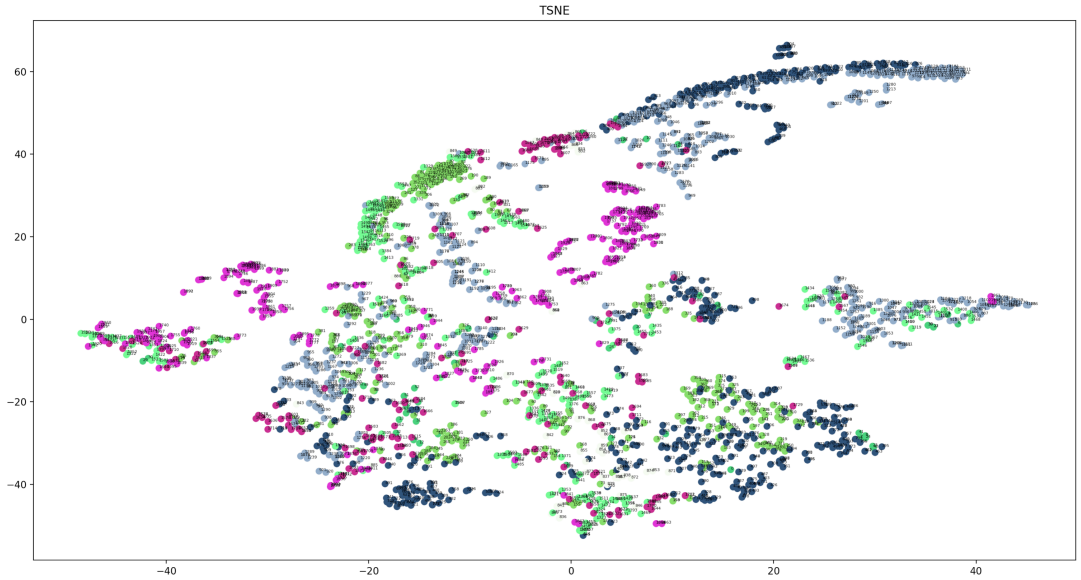


Figure 10: t-SNE result in a scatter plot, each colour indicates one test subject. The test subjects have clustered together slightly in the chain.

4.2 Dimensionality Reduction

Dimensionality reduction was implemented to reduce the number of dimensions (number of attributes, so in this case number of columns). Principal Components Analysis (PCA) and t-SNE, as described in section 4.2, were used to reduce the dimensionality of the data set.

4.2.1 PCA

PCA was the initial approach used in the experiment. The sklearn *PCA*[15] function was used to reduce the number of dimensions to 2, which simplified visualisation in 2D scatterplots. The PCA allowed 65%-95% **todo - recalc after chain removal** (depending on data preparation type) of the data's important structures to be accounted for in only the first two or three principle components. As can be seen in figure 12a, the resulting data from these components, didn't show any significant clusters, in comparison to the t-SNE results. The principle components were further depicted in a 3D scatter plot, in the hopes that the extra dimension reveal more structure. However, as can be seen in 12b, the results were practically the same. **todo: explain that probably because pca is linear and data is not**

15. `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

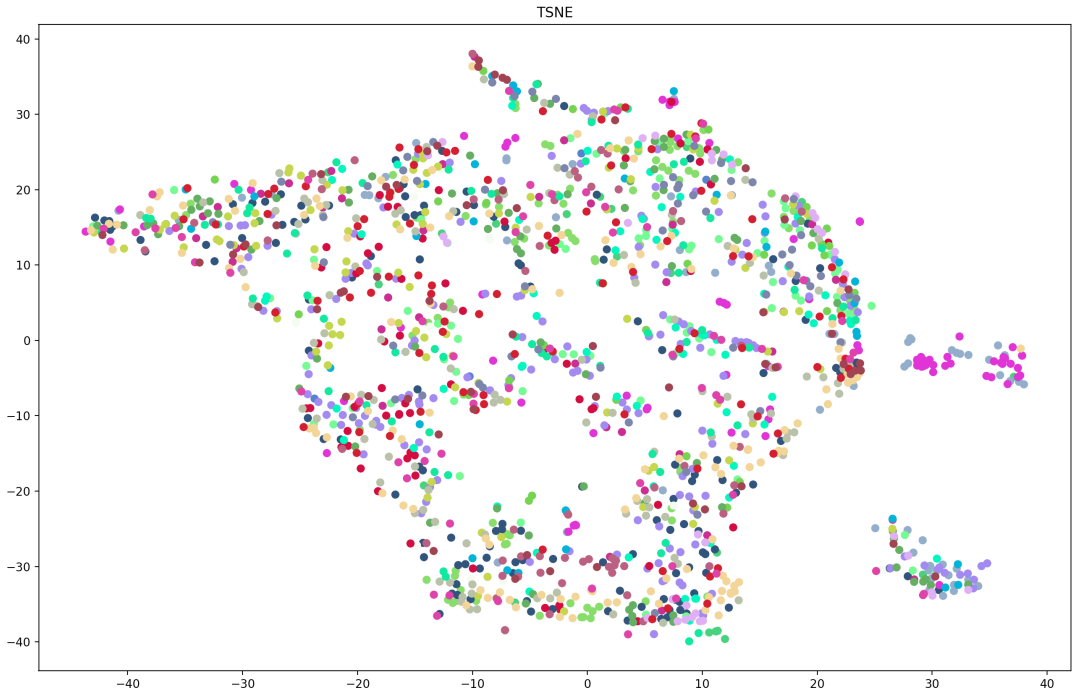


Figure 11: t-SNE calculated after the removal of rows with less than 50% of values that weren't 0. The chain is less visible and dominant.

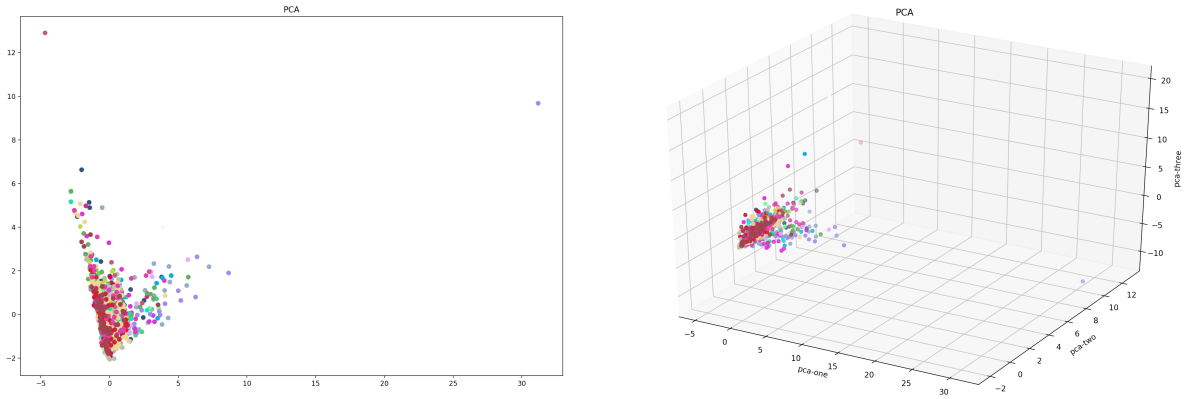


Figure 12: 2D (a) and 3D (b) scatter plots of the PCA results. The data points only appear to form one cluster with little visible structure.

4.2.2 t-SNE

The t-SNE algorithm was implemented using the sklearn *t-SNE*[16] implementation. To be able effectively apply the t-SNE algorithm, it was important to identify the parameters (perplexity, learning **todo: explain what the learning rate is**, and number of iterations) most suitable to the SmartEater data set, as described in section 3.3.2. The number of iterations was originally set to the default value 1000, but was later raised to 5000, to avoid receiving significantly different results each time t-SNE was run. Raising the number of iterations provided more consistent results. The overall structure of these were the same, though usually rotated differently. 5000 iterations was also used in the experiments by Wattenberg, Viégas, and Johnson (2016), who claimed that their t-SNE plot had reached a stable point by then.

To find suitable perplexity and learning rate t-SNE parameters to create the most distinct clusters possible, multiple results created using various parameter values were compared. For this, 2D t-SNE scatter plots were created using different perplexity and learning rate values. First to the perplexity. Wattenberg, Viégas, and Johnson (2016) proved to be a helpful source in tuning this parameter. Multiple values between the recommended 5 and 50 were tested. The resulting scatter plots are listed in the appendix A.1. The graphs on the left depict the t-SNE results, while the scatter plots on the right illustrate DBSCAN clustering of these results. The DBSCAN cluster colourings were placed here to support in perceiving the differences between the t-SNE structured data and to see when the most distinct and well defined clusters are formed. When visually comparing the different perplexities, the data points in the lower perplexity graphs (e.g. 5 and 10) appear to be randomly scattered and have no stucture. The higher the perplexity gets, the more the data points take on structure and more distinct and well defined clusters appear. From a perplexity of 20, the one hour data files appear to start forming such clusters. When scanning the higher perplexities, it can be seen, that clusters become more separated. The scatter plots from perplexity 30 to 50 appear to have formed well defined clusters.

To aid in selecting the perplexity that creates the best clusters, the three mathematical evaluation scores mentioned in section 3.6 were compared. Figure 13 displays the Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index were calculated for the DBSCAN and OPTICS clusterings of different perplexities. The light green highlighted values illustrate the best values (best defined and distinct clusters) for the one hour or three hour time lengths, whilst the darker green features the overall best value (1h and 3h files). With 4 ”wins”, a perplexity of 20 has the highest number of best achieving score values. The perplexities of 10, 40, and 45 tied in second place with 2 ”wins”. While according to the scores, perplexity 20 creates the best clusters. Visually the clusters appear better defined with a perplexity between 30 and 50. For this reason, the focus of the comparison of the scores was moved to the scores that came second and whose perplexity was between 30 and 50. Perplexity 40 is a clear winner in figure 14 and was chosen as the final parameter. These tests were conducted a second time (see appendix A.1.8), using the average of two different t-SNE results for 1h and 3h files. While the scores are slightly different, a perplexity of 40 was again the overall winning result.

16. `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

1h Files							
	Perplexity: 5	Perplexity: 10	Perplexity: 20	Perplexity: 30	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)							
Silhouette	0.012301321	0.17785606	0.23409332	0.13068332	0.020629907	-0.001459693	-0.13791381
Davies Bouldin	1.332029385	1.502340712	1.236333265	1.396269294	1.765516415	2.312260633	2.06274638
Calinski Harabasz	11.79207398	24.68251028	100.4317614	337.5606908	527.1113507	555.0973937	381.4331244
OPTICS							
Silhouette	-0.56731904	-0.2652787	0.0904175	0.12797767	0.028060276	-0.053797465	-0.13770019
Davies Bouldin	1.173244861	1.624939639	1.271594621	1.423696525	1.627679428	2.927482693	2.039984915
Calinski Harabasz	5.865016692	11.64971005	34.93509218	165.8283109	427.7636283	420.6156948	343.8819753
3h Files							
	Perplexity: 5	Perplexity: 10	Perplexity: 20	Perplexity: 30	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)							
Silhouette	0.19653113	0.30535686	0.28253406	0.162854	0.005408226	-0.056066718	-0.11255615
Davies Bouldin	1.357905465	1.454077361	1.295397242	1.345344464	1.595251196	1.633439218	1.568908348
Calinski Harabasz	18.52401582	43.48411771	149.8457962	454.3884285	821.6969448	418.2231883	427.8332514
OPTICS							
Silhouette	-0.33854422	0.003906393	0.16133659	0.14873675	0.06374386	0.052369475	-0.045151174
Davies Bouldin	1.302466526	1.276339339	1.338689898	1.522294895	1.600873186	1.72163966	1.598151632
Calinski Harabasz	7.909086677	18.77438054	56.32308984	214.1240169	538.5429769	547.9833581	480.4260331
	best values in files (1h or 3h)						
	best values total (1h and 3h)						

Figure 13: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **perplexities**.

1h Files			
	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)			
Silhouette	0.020629907	-0.001459693	-0.13791381
Davies Bouldin	1.765516415	2.312260633	2.06274638
Calinski Harabasz	527.1113507	555.0973937	381.4331244
OPTICS			
Silhouette	0.028060276	-0.053797465	-0.13770019
Davies Bouldin	1.627679428	2.927482693	2.039984915
Calinski Harabasz	427.7636283	420.6156948	343.8819753
3h Files			
	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)			
Silhouette	0.005408226	-0.056066718	-0.11255615
Davies Bouldin	1.595251196	1.633439218	1.568908348
Calinski Harabasz	821.6969448	418.2231883	427.8332514
OPTICS			
Silhouette	0.06374386	0.052369475	-0.045151174
Davies Bouldin	1.600873186	1.72163966	1.598151632
Calinski Harabasz	538.5429769	547.9833581	480.4260331
	best values in files (1h or 3h)		
	best values total (1h and 3h)		

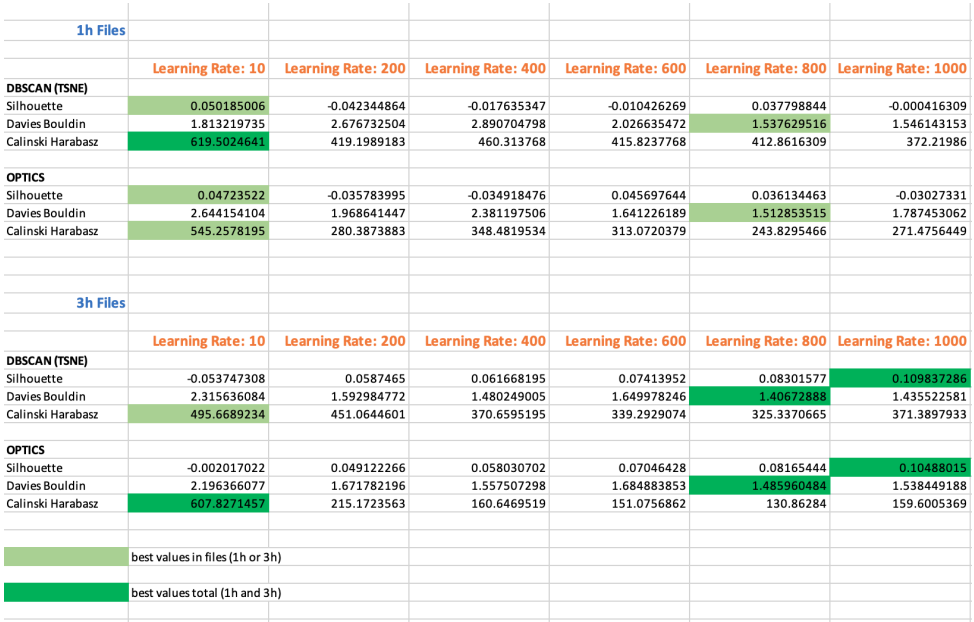
Figure 14: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **perplexities**.

The same principle was used to determine the ideal value for the learning rate parameter. The sklearn t-SNE documentation states, that the learning rate is normally set between 10 and 1000, hence the learning rate in the scatter plots was chosen in this interval. The various resulting scatter plots can be seen in the appendix A.2. The differences in the arrangement of the data points over the plots is not as substantial as it was when testing for the perplexity. The results, listed in figure 15 reveal, that the learning rate 10 scores highest with a learning rate of 800 coming in second. Since roughly 200 steps were taken when testing the learning rate value, smaller steps were taken (i.e. 50) around the winning value to see if an even better performance could be accomplished (figure 58, in appendix A.2.7). While 10 still achieved best in the one hour time length files, 800 came top in the three hour files, indicating that 10 might not be the best rate for these. Consequently, the values 20 and 30 were also evaluated (figure 59, in appendix A.2.7). While a learning rate of 10 still came top in the one hour files, 30 came top in the three hour files. With the hope of finding a value in between 10 and 30 that satisfied both time lengths, the learning rates 10, 15, 20, 25, and 30 were ultimately compared exclusively. The results in figure 16 reveal, that this final comparison lead to the learning rate 20 being the most suitable. As with the perplexity, the learning rate tests were also reproduced, averaging two different t-SNE outcomes. The results are illustrated in appendix A.2.8. In general, the results are similiar. 800 appears to more dominant than the lower parameter values (e.g. 10 and the favourite 20). To further compare 20 and 800, appendix A.2.9 contains a table (figure 64) comparing 4 different runs of scores from average t-SNE results. With 2 wins more than 20, 800 has the overall better score results. Figures 65 and 66 visually compare the scatter plots created with a learning rate of 20 and 800. The plots and clusters appear very similar. The clusters from perplexity 20 however seem slightly more compact (less smaller clusters). A learning rate of 20 was thus selected for the final learning rate parameter. However, 800 was also used in the final results.

The final t-SNE parameters, from which the results were then used by the DBSCAN and OPTICS clustering methods in the next section, were as follows: perplexity = 40, learning_rate = 20, and n_iter = 5000. A run of the resulting graphs are depicted in figures 18 and 17.

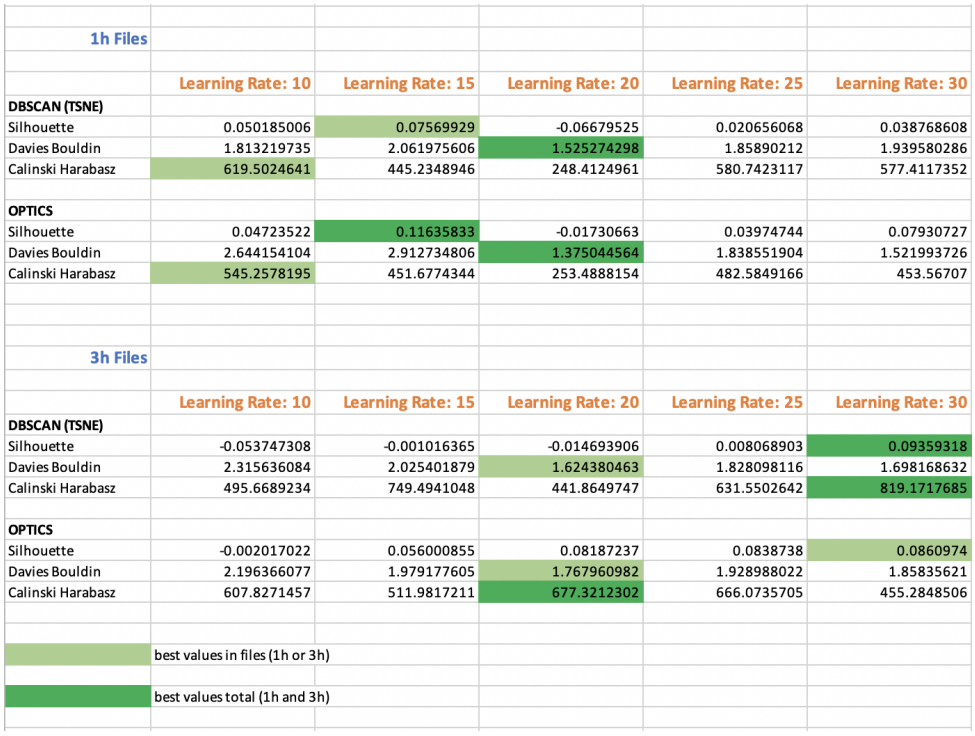
4.3 Clustering

The clustering algorithms used for the SmartEater data set, were DBSCAN and OPTICS. One of the advantages of these density-based clustering methods (as mentioned in section 3.5.3) are that there are less parameters to configure. Another reason for choosing these methods, is that there is no need to define a fixed number of k clusters to find, since the cluster boundaries are regulated by density. This technique also allows arbitrary-shaped clusters to be correctly identified. The t-SNE dimensionality reduction approach provided more significant results than PCA. Thus, the cleaned data and t-SNE modified data (with two components) was fed into the clustering algorithms DBSCAN and OPTICS.



1h Files						
	Learning Rate: 10	Learning Rate: 200	Learning Rate: 400	Learning Rate: 600	Learning Rate: 800	Learning Rate: 1000
DBSCAN (TSNE)						
Silhouette	0.050185006	-0.042344864	-0.017635347	-0.010426269	0.037798844	-0.000416309
Davies Bouldin	1.813219735	2.676732504	2.890704798	2.026635472	1.537629516	1.546143153
Calinski Harabasz	619.5024641	419.1989183	460.313768	415.8237768	412.8616309	372.21986
OPTICS						
Silhouette	0.04723522	-0.035783995	-0.034918476	0.045697644	0.036134463	-0.03027331
Davies Bouldin	2.644154104	1.968641447	2.381197506	1.641226189	1.512853515	1.787453062
Calinski Harabasz	545.2578195	280.3873883	348.4819534	313.0720379	243.8295466	271.4756449
3h Files						
	Learning Rate: 10	Learning Rate: 200	Learning Rate: 400	Learning Rate: 600	Learning Rate: 800	Learning Rate: 1000
DBSCAN (TSNE)						
Silhouette	-0.053747308	0.0587465	0.061668195	0.07413952	0.08301577	0.109837286
Davies Bouldin	2.315636084	1.592984772	1.480249005	1.649978246	1.40672888	1.435522581
Calinski Harabasz	495.6689234	451.0644601	370.6595195	339.2929074	325.3370665	371.3897933
OPTICS						
Silhouette	-0.002017022	0.049122266	0.058030702	0.07046428	0.08165444	0.10488015
Davies Bouldin	2.196366077	1.671782196	1.557507298	1.684883853	1.485960484	1.538449188
Calinski Harabasz	607.8271457	215.1723563	160.6469519	151.0756862	130.86284	159.6005369
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 15: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.



1h Files					
	Learning Rate: 10	Learning Rate: 15	Learning Rate: 20	Learning Rate: 25	Learning Rate: 30
DBSCAN (TSNE)					
Silhouette	0.050185006	0.07569929	-0.06679525	0.020656068	0.038768608
Davies Bouldin	1.813219735	2.061975606	1.525274298	1.85890212	1.939580286
Calinski Harabasz	619.5024641	445.2348946	248.4124961	580.7423117	577.4117352
OPTICS					
Silhouette	0.04723522	0.11635833	-0.01730663	0.03974744	0.07930727
Davies Bouldin	2.644154104	2.912734806	1.375044564	1.838551904	1.521993726
Calinski Harabasz	545.2578195	451.6774344	253.4888154	482.5849166	453.56707
3h Files					
	Learning Rate: 10	Learning Rate: 15	Learning Rate: 20	Learning Rate: 25	Learning Rate: 30
DBSCAN (TSNE)					
Silhouette	-0.053747308	-0.001016365	-0.014693906	0.008068903	0.09359318
Davies Bouldin	2.315636084	2.025401879	1.624380463	1.828098116	1.698168632
Calinski Harabasz	495.6689234	749.4941048	441.8649747	631.5502642	819.1717685
OPTICS					
Silhouette	-0.002017022	0.056000855	0.08187237	0.0838738	0.0860974
Davies Bouldin	2.196366077	1.979177605	1.767960982	1.928988022	1.85835621
Calinski Harabasz	607.8271457	511.9817211	677.3212302	666.0735705	455.2848506
	best values in files (1h or 3h)				
	best values total (1h and 3h)				

Figure 16: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

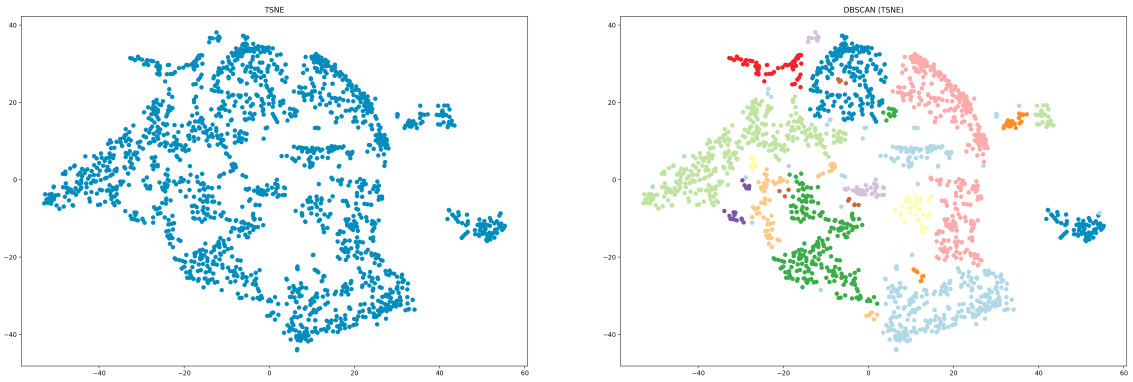


Figure 17: Final t-SNE parameters (1h data files): perplexity=40, n_iter=5000, learning_rate=20

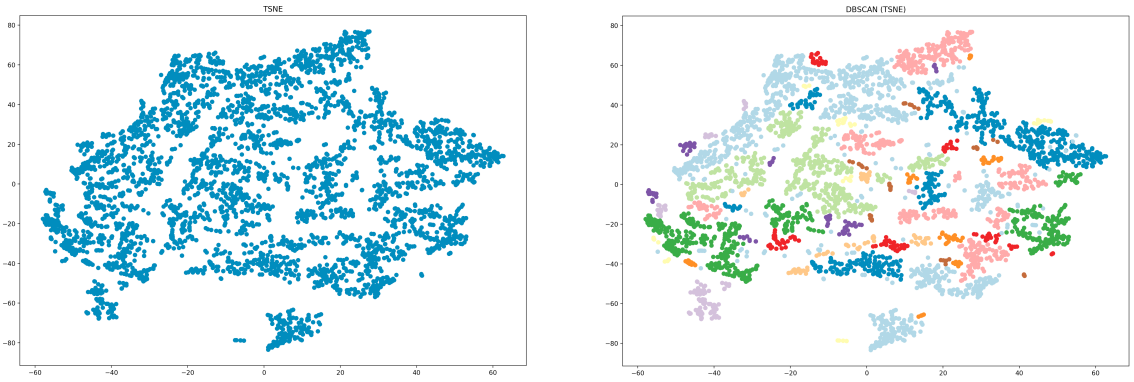
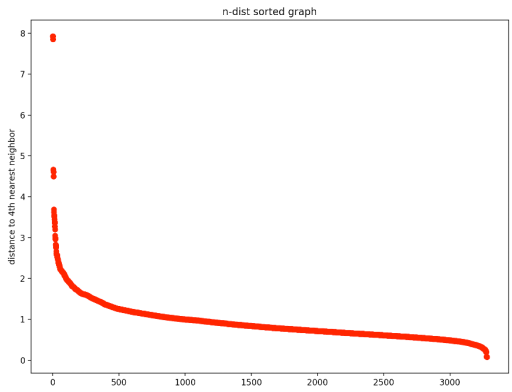


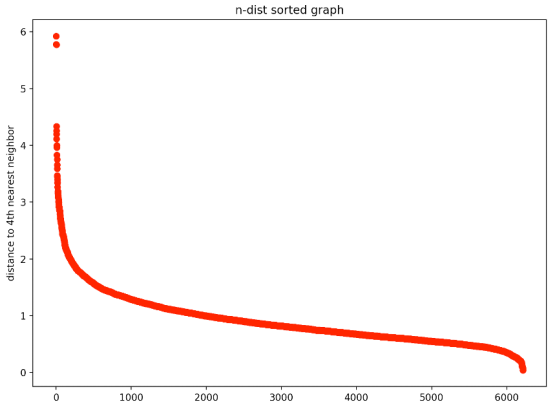
Figure 18: Final t-SNE parameters (3h data files): perplexity=40, n_iter=5000, learning_rate=20

4.3.1 DBSCAN

The DBSCAN algorithm was applied on the SmartEater data set using the sklearn *DBSCAN*[17] function. Section 3.5.3.1 describes the functionality of the DBSCAN clustering method. As mentioned, one of the disadvantages of DBSCAN, is the need to specify parameters, which can change the outcome of the results. In order to establish suitable parameters, k-dist graphs were generated for the 1h and 3h data sets. The graphs contained the distances to the k nearest neighbors. As recommended by Ester et al. (1996)[230], *MinPts* and k were set to 4 and the graphs were used to determine *Eps*. The idea is to select *Eps* suitable for the "thinnest" cluster, however being careful to avoid noise. As can be seen in figures 19a and 19b, the valley starts at roughly a 4th nearest neighbor distance of 2.



(a) 1h data set

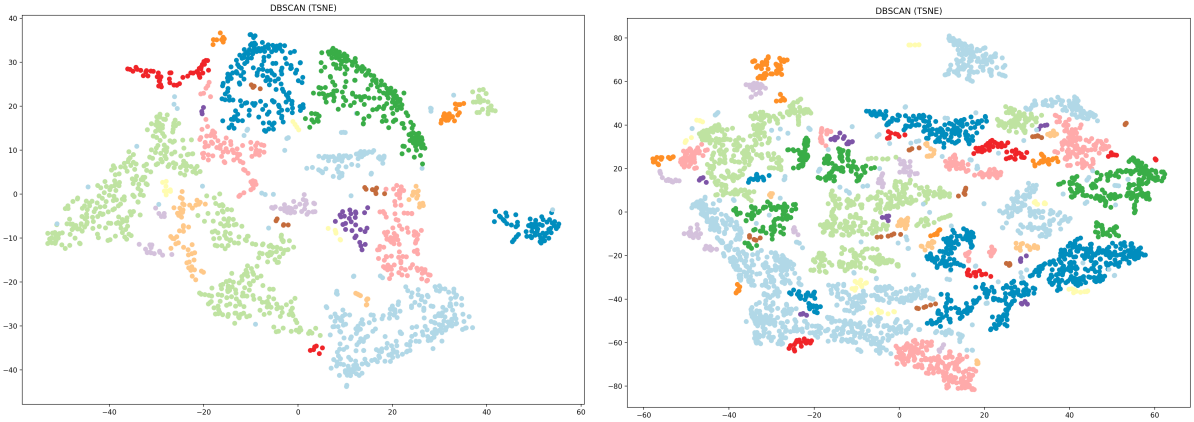


(b) 3h data set

Figure 19: Sorted 4-dist graphs (distance for each point to its fourth nearest neighbor), used to determine a suitable Eps parameter for the DBSCAN algorithm. The valley starts at roughly the 4th nearest neighbor distance of 2, therefore Eps should be 2.

The DBSCAN method was applied, with the parameters eps = 2 and min_samples (MinPts) = 4. The results of this clustering method can be seen in figure 20.

17. `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`



(a) 1h data set DBSCAN clustering (first column - 15 min).

(b) 3h data set DBSCAN clustering (first column - 30 min).

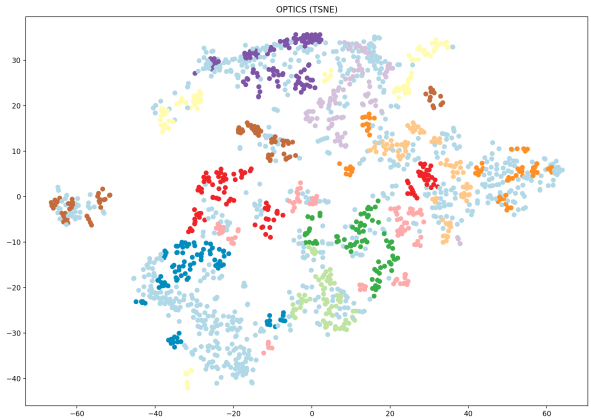
Figure 20

4.3.2 OPTICS

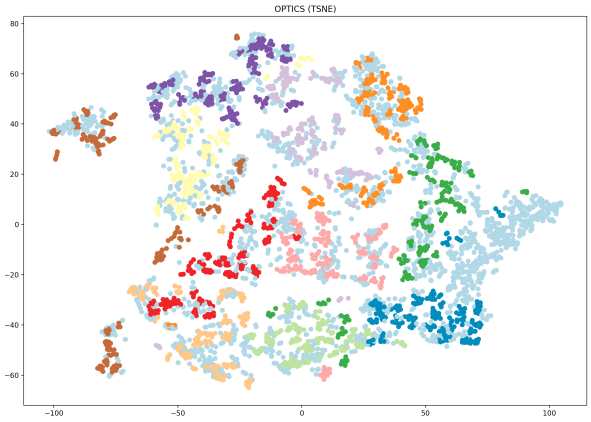
The OPTICS algorithm was also implemented with its sklearn implementation[18]. As stated in section 3.5.3.2, the OPTICS algorithm creates a reachability plot for the data points. Initially, the clusters were extracted using the clustering method ”xi”. This is the automatic cluster extraction method, as explained in section 3.5.3.2 and introduced in Ankerst et al. (1999)[57]. The visual results of these clusterings appeared to contain many points that were not assigned to clusters, as can be seen in figure 21. Moreover, the corresponding reachability plots showed many points that weren't assigned to a cluster (no colour). This resulted in the DBSCAN method being used to cluster the OPTICS results. The Eps parameter was set to 2, as determined before for the DBSCAN clustering algorithm.

todo: see reachability plots full size in appendix todo: convert reachabili ty plot to bar chart

18. `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html`

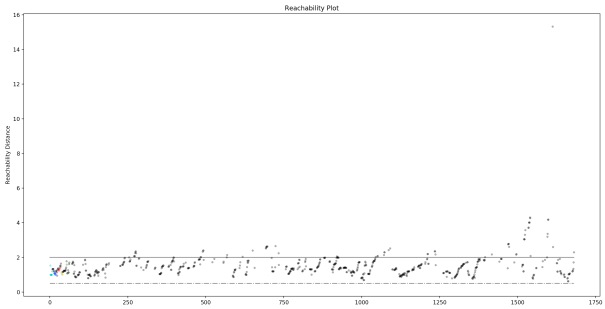


(a) 1h data set OPTICS clustering (first column - 30 min), using OPTICS automatic cluster extraction (xi).

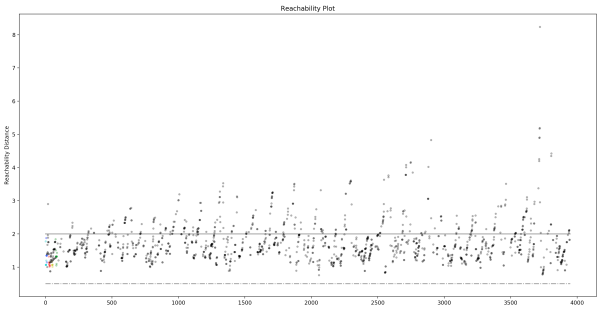


(b) 3h data set OPTICS clustering (first column - 30 min), using OPTICS automatic cluster extraction (xi).

Figure 21

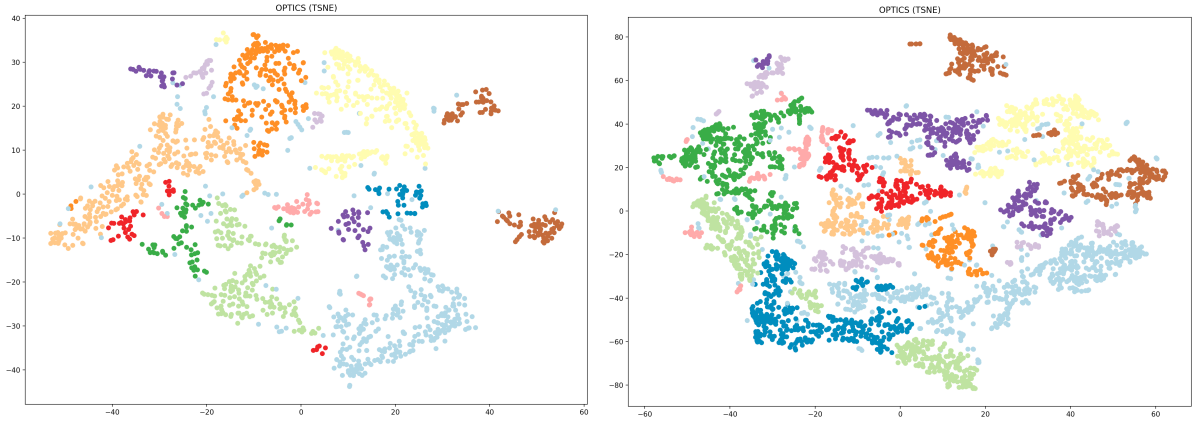


(a) 1h data set OPTICS clustering (first column - 15 min), using OPTICS automatic cluster extraction (xi).



(b) 3h data set OPTICS clustering (first column - 30 min), using OPTICS automatic cluster extraction (xi).

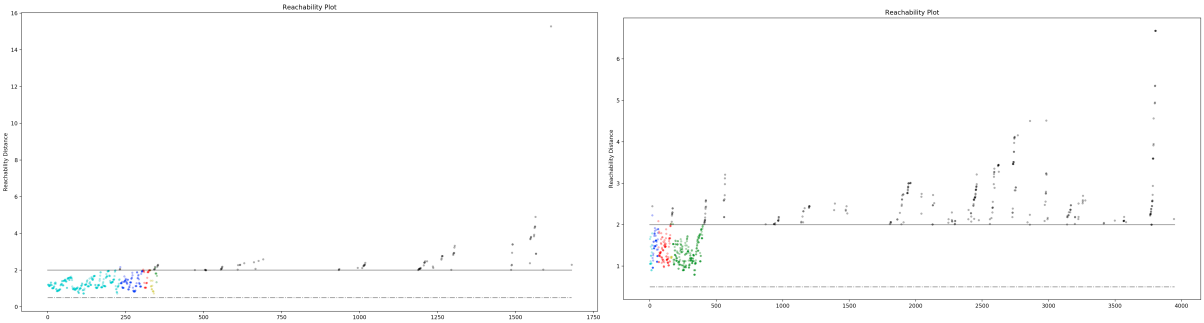
Figure 22



(a) 1h data set OPTICS clustering (first column - 15 min), using DBSCAN clustering.

(b) 3h data set OPTICS clustering (first column - 30 min), using DBSCAN clustering.

Figure 23



(a) 1h data set OPTICS clustering (first column - 15 min), using DBSCAN clustering.

(b) 3h data set OPTICS clustering (first column - 30 min), using DBSCAN clustering.

Figure 24

The final DBSCAN and OPTICS clusterings scatter plots for each time delta are compared in appendix A.3.

4.4 Comparison and evaluation of cluster results from different time lengths

The three mathematical evaluation scores mentioned in section 3.6, i.e. Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index were used to compare the resulting clusters. They were each implemented using the sklearn library[19], the same way as in section 4.2.2.

19. `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`, `https://scikit-learn.org/stable/modules/generated/`

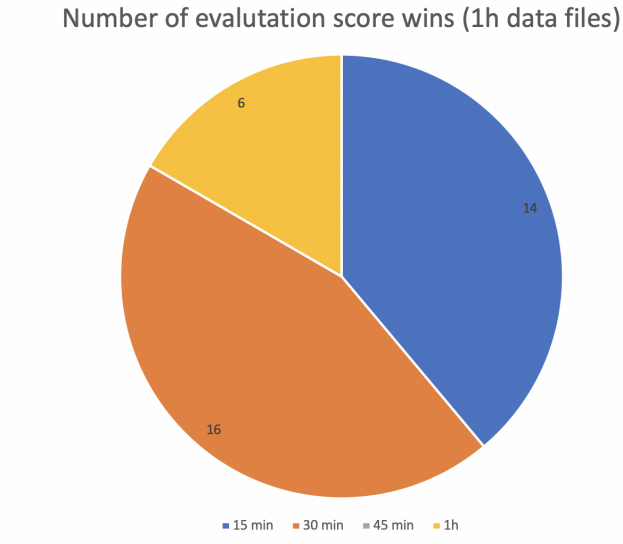
As also explained in the respective sklearn documentations, the Silhouette Score indicates better, denser clustering, when it is closer to 1, and incorrect clustering results if the value is close to -1. The Davies-Bouldin Index indicates well chosen clusters, when the value is closer to 0 (lowest possible score). A higher Caliński-Harabasz Index suggests well separated and dense clusters. These three scores were calculated and stored for each data frame, after each clustering algorithm was applied (DBSCAN and OPTICS) for each time length. The resulting values where then compared and are depicted in figures below. The experiment was run on each time length (1h files: 15 min, 30 min, 45 min, 1h; 3h files: 30 min, 1h, 1 h 30 min, 2h, 2h 30 min, 3h). The numbers were different for each time the results were calculated, since t-SNE produces slightly different results. Therefore, the t-SNE and clustering was run multiple times and the results were compared. In appendix A.4 figures 77 and 78 two individual t-SNE and clusterings were run, figure 79 depicts the average scores of these two. Figure 80 depicts an average of a different two runs. These mentioned runs all held the learning rate parameter 20. Since 800 also proved to be a viable choice for this parameter, the t-SNE and clustering was also run twice and averaged (figure 81) with a learning rate of 800. The mean was taken from all these revealed results, thus creating figure 25 (the figure can be seen in this section). From these, there is no clear winner, although there are some stronger candidates, i.e. 15 min (1h), 30 min (1h), 1h (3h), and 2h (3h).

Like before, when comparing the t-SNE results, the light green field indicate the best and most distinct clusters from the 1h or 3h data files, while the dark green fields highlight the best value for that score overall for all time lengths. These green fields, or number of ”wins” were summed to see which time length had the best number of scores the most. The light green wins (1h or 3h) for the 1h data files are pictured in figure 26a, in figure 26b for the 3 hour data files, and in figure 27 for the comparison of the dark green wins (1h and 3h). The two top winners for the 1h data files and the 3h data files, were: 30 min (1h), 15 min (1h), 2h (3h), and 1h (3h). The top four results when comparing all time lengths, despite which files they were aggregated to, were: 30 min (1h), 1h (3h), 2h (3h), and 30 min (3h). The number of ”wins” is to some extent reliable on the resulting scores from other time lengths it happened to be compared with. It also does not fully factor in the full scale by how far this time length was better. For this reason, the average scores from figure 25 for thee five ”winning” time lengths were compared solely to each other. The results are detailed in figure 28. The 2h time delta achieved the majority of best scores in this final comparison. When removing the 2h column, 30 min (1h) and 1h (3h) come in second place. 1h (3h) however out performs when compared directly to 30 min (1h).

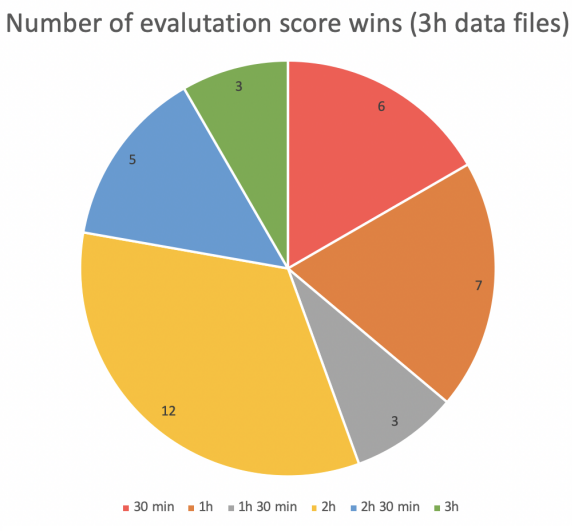
sklearn.metrics.davies_bouldin_score.html, and https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.053476365	0.013788914	-0.038610232	-0.052956536		
Davies Bouldin	2.04244793	1.597171222	1.755245051	1.577807591		
Calinski Harabasz	494.5009903	349.3147449	344.8215987	364.2258626		
OPTICS						
Silhouette	0.078437062	0.12075056	-0.017926106	0.05120006		
Davies Bouldin	1.844121062	1.596800429	1.828162369	1.461980828		
Calinski Harabasz	398.3315599	347.2439435	268.4922909	303.9834533		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	0.011948289	0.045027432	0.029207257	0.084418905	0.058235747	0.066334556
Davies Bouldin	1.702623809	1.581150429	1.713516524	1.523790147	1.590690427	1.608006213
Calinski Harabasz	513.226822	639.5570987	496.3219016	525.4316832	525.3219704	523.1398047
OPTICS						
Silhouette	0.052804309	0.077179238	0.105263528	0.08859444	0.077369563	0.083093493
Davies Bouldin	1.774005053	1.649207334	1.621448729	1.4901967	1.45541583	1.653579405
Calinski Harabasz	442.9552584	408.6641751	325.6016964	320.8891861	325.764943	298.096556
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 25: Evaluation scores comparison averaged from figures 77, 78, 79, 80, and 81.



(a) Number of light green evaluation score ”wins” (1h or 3h) for the 1h data files.



(b) Number of light green evaluation score ”wins” (1h or 3h) for the 3h data files.

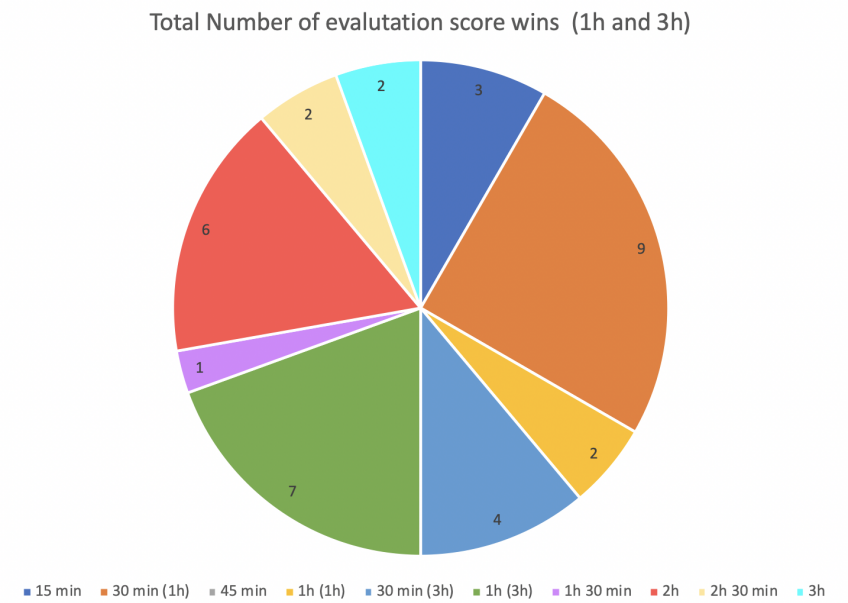


Figure 27: Number of dark green evaluation score ”wins” (1h and 3h).

	15 min (1h)	30 min (1h)	30 min (3h)	1h (3h)	2h (3h)
DBSCAN (TSNE)					
Silhouette	0.053476365	0.013788914	0.011948289	0.045027432	0.084418905
Davies Bouldin	2.04244793	1.597171222	1.702623809	1.581150429	1.523790147
Calinski Harabasz	494.5009903	349.3147449	513.226822	639.5570987	525.4316832
OPTICS					
Silhouette	0.078437062	0.12075056	0.052804309	0.077179238	0.08859444
Davies Bouldin	1.844121062	1.596800429	1.774005053	1.649207334	1.4901967
Calinski Harabasz	398.3315599	347.2439435	442.9552584	408.6641751	320.8891861
	best values in files (1h or 3h)				

Figure 28: Top evaluation scores performers comparison from figures 26a, 26b, and 27.

1h vs 3h Files		
	1h	3h
DBSCAN (TSNE)		
Silhouette	-0.052956536	0.066334556
Davies Bouldin	1.577807591	1.608006213
Calinski Harabasz	364.2258626	523.1398047
OPTICS		
Silhouette	0.05120006	0.083093493
Davies Bouldin	1.461980828	1.653579405
Calinski Harabasz	303.9834533	298.096556
	best values in files (1h or 3h)	

Figure 29: Evaluation scores comparison from averaged 1h and 3h run of t-SNE and clustering.

5 Discussion

Section 4.4 reveals, that the 2h time length produced the most distinct and well defined clusters, according to the average Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index results. The 1h time length (from the 3h data set) came second, and the 30 min time delta (from the 1h data set) came third. When considering the highest number of evaluation score ”wins” over different t-SNE and clustering results, the 30 min time delta (from the 1h data set) however came top, followed by the 1h time length (from the 3h data set) and then the 2h time length (see figure 27). It can therefore be concluded, that these 3 time deltas are the best performers.

To determine whether the 1 hour or 3 hour aggregation files led to better clusters, the average results of both time files are compared in figure 29. Both the 1h and 3h data sets have the same amount of ”wins”. When scanning other comparisons of these files (also the ones concluded for finding the t-SNE parameters), it is noticeable, that the 1h and 3h data sets either have the same number of ”wins”, or the 3h data set has more. This indicates that overall, the three hour aggregation files produce more superior clusters.

The 45 minute time delta in the 1 hour data files performed the lowest, with 0 ”wins”. The fact that 45 never out performed the other time lengths shows consistency, despite the different results. 1h 30 min(3h), 1h(1h), 2h 30 min (3h) and 3h (3h) all had no more than 2 wins.

The results from Rahman et al. (2016) (as mentioned in section 2) showed, that higher time lengths (e.g. 100 minutes) performed better than shorter ones. They deducted, that smaller window sizes were susceptible to noise and had a higher gap between precision (exactness) and recall (completeness). This discovery could also apply to the results of this experiment, considering that the 2h and 1h time lengths in the three hour data files were under the top 3 results.

Of the 36 number of ”wins” across all time deltas (the 1h and 3h aggregation files combined), 14 of these were achieved by the 1h data file time lengths (38.9%), the other 22 by the 3 hour data set (61.1%). A possible explanation, as to why the 3h data set might have created better clusters, was that the data set had more rows. After the data preparation step, the 1h aggregation set was left with an average (depending on number of time length columns considered) of 1654 rows, whilst the 3h set was left with over double the amount of rows with an average of 3976.5 rows. The use of more data could have resulted in more similar rows and more improved clusters. Another aspect to consider, is that shorter sensor data recorded for shorter time lengths might not be long enough to detect underlying stress patterns.

Moreover, the higher the number of rows, the more robust the clusters could be towards potential outliers. As explained in section 3.6.2.1, the Silhouette Score compares the within cluster distances to the distance of a neighbouring cluster. If many points are well placed within a cluster, an outlier will have little impact against the many short distances. However, if the cluster is not very dense and only contains a few ”good” assigned data points, an outlier could have a larger impact on the result.

In order to provide ”just in time” aid, the SmartEater app would need to predict upcoming stress. Stress can be longer and build slowly (e.g. increasing worry as an exam gets closer), or sudden (e.g. find out you have failed an exam). **Kann ich das so sagen (ohne Zitat)? - weil es ist glaube ich eher etwas allgemein bekanntes.** The data points would need to account for and recognise these different lengths of stress. For example, if stress is short, then a longer time delta will likely make it seem less relevant, compared to all the unstressed data. However, if it is long, a shorter time delta might miss it or only perceive a small part of it. This could lead to false cluster assignments or noise. However, it could also create its own clusters. This could be the reason why middle time lengths, such as 2h, 1h (3h data files) created more distinct clusters, since it may have been more likely to recognise these differences. The fact that there was not one specific time length that always performed higher than the others shows, that a single time length may not be enough to be able to create clear clusters. Different time lengths might be needed to detect different patterns.

When visually comparing the clusters in appendix A.3, the 3h data sets appear to be denser, whilst the 1h data sets have more whitespace. The 1h data set top performer, 30 minutes, seems to be scattered to a slightly different shape to the other scatter plots of the 1h data set. It appears to be spread out wider along the x axis, more like the 3h data files. The 30 minute time length achieved the highest number of wins in the 1h data set. It could be argued, that the shape of the scattering of the points with similarity to the 3h data scatter plots, that also performed well, could be an indicator for more enhanced clustering results.

For future work, it might be advisable to get more data from users for a longer period of time. However, it would also be necessary confirm that the clusters were created for stress/not stress and for each user (since users behaviour could exhibit similarities). It might also be beneficial to support the mathematical evaluations with hand drawn clusters found by test users in user studies. Especially since the scores are different each time t-SNE is computed, sometimes leading to different results.

todo: finish

6 Conclusion

This thesis compares different time deltas for aggregation, to determine which one is ideal to construct high quality clusterings from smartphone sensor and usage data. This data was recorded from different test subjects for the SmartEater mobile health app. The in 1h and 3h aggregated data sets were preprocessed, in which missing values, unnecessary columns and rows with more than 50% zeros were removed. The resulting rows were normalised using z-score normalization. Using t-SNE, the 8 existing dimensions (attributes) were reduced to 2 and visualised in scatter plots. Such plots were created for each time length, a total of 10, within the data set. DBSCAN and OPTICS clustering algorithms were used to group the data points together into clusters. The Silhouette Score, Davies-Bouldin Index and Caliński-Harabasz Index are used to mathematically evaluate the resulting clusters for each time length. The comparison of these scores leads to believe, that the following three time lengths produce the most distinct and well defined clusters: 30 minutes (from the 1h data set), 1 hour, and 2 hours (both from the 3h data set).

todo: finish

References

Aalst, W M P van der, V Rubin, H M W Verbeek, B F van Dongen, E Kindler, and C W Günther. 2010. "Process mining: a two-step approach to balance between underfitting and overfitting." *Software & Systems Modeling* 9 (1): 87–111. ISSN: 1619-1374. doi:10.1007/s10270-008-0106-z.

Ameko, M. K., L. Cai, M. Boukhechba, A. Daros, P. I. Chow, B. A. Teachman, M. S. Gerber, and L. E. Barnes. 2018. "Cluster-based approach to improve affect recognition from passively sensed data." In *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, 434–437.

Ankerst, Mihael, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. "OPTICS: Ordering Points to Identify the Clustering Structure." *SIGMOD Rec.* (Philadelphia, Pennsylvania, USA), SIGMOD '99, 28, no. 2 (June): 49–60. ISSN: 0163-5808. doi:10.1145/304181.304187. https://doi.org/10.1145/304181.304187.

Aranganayagi, S., and K. Thangavel. 2007. "Clustering Categorical Data Using Silhouette Coefficient as a Relocating Measure." In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, 2:13–17. Sivakasi, Tamil Nadu, India, December. doi:10.1109/ICCIMA.2007.328.

Bellman, R. E. 1957. *Dynamic Programming*. Rand Corporation research study. Princeton University Press. ISBN: 9780691079516.

———. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton Legacy Library. Princeton University Press. ISBN: 9781400874668.

Bermad, N., and M. T. Kechadi. 2016. "Evidence analysis to basis of clustering: Approach based on mobile forensic investigation." In *2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 300–307. Hammamet, Tunisia, December. doi:10.1109/SETIT.2016.7939884.

Caliński, Tadeusz, and JA Harabasz. 1974. "A Dendrite Method for Cluster Analysis." *Communications in Statistics - Theory and Methods* 3 (January): 1–27. doi:10.1080/03610927408827101.

Davies, D. L., and D. W. Bouldin. 1979. "A Cluster Separation Measure." *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1 (2): 224–227.

Dey, R., and S. Chakraborty. 2015. "Convex-hull DBSCAN clustering to predict future weather." In *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, 1–8. Vancouver, BC, Canada, October. doi:10.1109/IEMCON.2015.7344438.

Ester, M., H. Kriegel, J. Sander, and X. Xu. 1996. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, 96:226–231. 34. Portland, Oregon, August.

García, Salvador, Julián Luengo, and Francisco Herrera. 2015. *Data preprocessing in data mining.* Springer.

Han, Jiawei, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques.* Burlington, Massachusetts: Elsevier.

Hartigan, John A. 1975. *Clustering algorithms.* John Wiley & Sons, Inc.

Hinton, Geoffrey E, and Sam T Roweis. 2003. "Stochastic neighbor embedding." In *Advances in Neural Information Processing Systems,* 15:833–840. Cambridge, MA, USA.

Hotelling, Harold. 1933. "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology* 24 (6): 417–441.

Jolliffe, I.T. 2002. *Principal Component Analysis: Second Edition.* Springer Series in Statistics. Springer-Verlag New York. ISBN: 0-387-95442-2. doi:10.1007/b98835.

Larose, Daniel T, and Chantal D Larose. 2015. *Data mining and predictive analytics.* 2. ed.. Wiley series on methods and applications in data mining. Hoboken, New Jersey: John Wiley & Sons. ISBN: 9781118116197.

Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing data using t-SNE." *Journal of Machine Learning research* 9 (Nov): 2579–2605.

McCue, C. 2014. *Data Mining and Predictive Analysis: Intelligence Gathering and Crime Analysis.* Butterworth-Heinemann (Elsevier). ISBN: 9780128004081. https://books.google.at/books?id=re1MBAAAQBAJ.

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

Pius Owoh, Nsikak, Manmeet Mahinderjit Singh, and Zarul Fitri Zaaba. 2018. "Automatic Annotation of Unlabeled Data from Smartphone-Based Motion and Location Sensors." *Sensors* 18 (7): 2134.

Pyle, Dorian. 1999. *Data preparation for data mining.* morgan kaufmann.

Rahman, Tauhidur, Mary Czerwinski, Ran Gilad-Bachrach, and Paul Johns. 2016. "Predicting 'About-to-Eat' Moments for Just-in-Time Eating Intervention." In *Proceedings of the 6th International Conference on Digital Health Conference,* 1–10. New York, NY, USA: Association for Computing Machinery. ISBN: 9781450342247. doi:10.1145/2896338.2896359.

Rousseeuw, Peter J. 1987. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." *Journal of computational and applied mathematics* 20:53–65.

Sornbootnark, P., and P. Khoenkaw. 2019. "Excessive Alcohol Craving Prediction Algorithm Using Smartphone Accelerometer Sensor." In *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, 335–338.

Stütz, Thomas, Thomas Kowar, Michael Kager, Martin Tiefengrabner, Markus Stuppner, Jens Blechert, Frank H. Wilhelm, and Simon Ginzinger. 2015. "Smartphone Based Stress Prediction." In *User Modeling, Adaptation and Personalization*, edited by Francesco Ricci, Kalina Bontcheva, Owen Conlan, and Séamus Lawless, 240–251. Cham: Springer International Publishing. ISBN: 978-3-319-20267-9.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson. 2016. "How to Use t-SNE Effectively." *Distill*. doi:10.23915/distill.00002. http://distill.pub/2016/misread-tsne.

Appendices

Anhänge löschen, die nicht verwendet werden.

A t-SNE parameters comparison figures

A.1 Perplexity

In the following figures, the t-SNE results of different perplexitites are compared, for the different time length files (1h and 3h), using the first columns of each feature (1h: first 15 minutes, 3h: fist 30 minutes). The left scatter plots depict t-SNE results, the right scatter plots visualise DBSCAN clusterings of t-SNE results).

A.1.1 Perplexity = 5

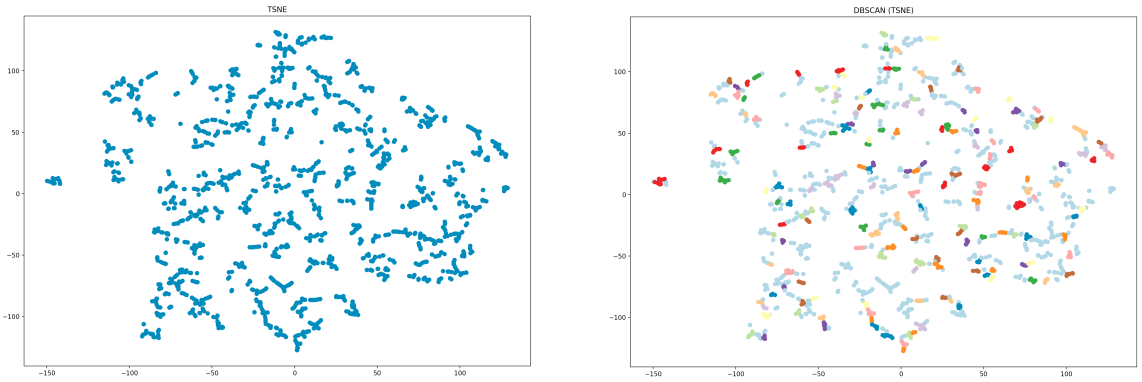


Figure 30: **1h** data files, t-SNE calculated with the following parameters: **perplexity=5**, n_iter=5000, learning_rate=50

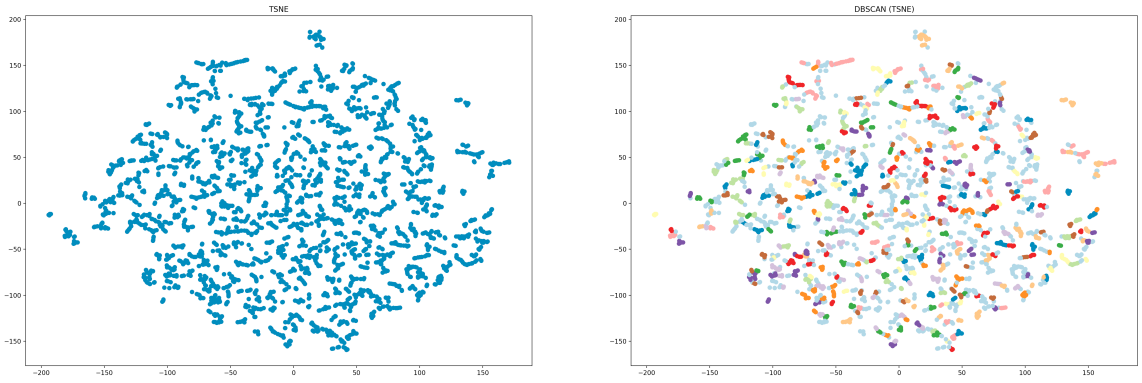


Figure 31: **3h** data files, t-SNE calculated with the following parameters: **perplexity=5**, n_iter=5000, learning_rate=50

A.1.2 Perplexity = 10

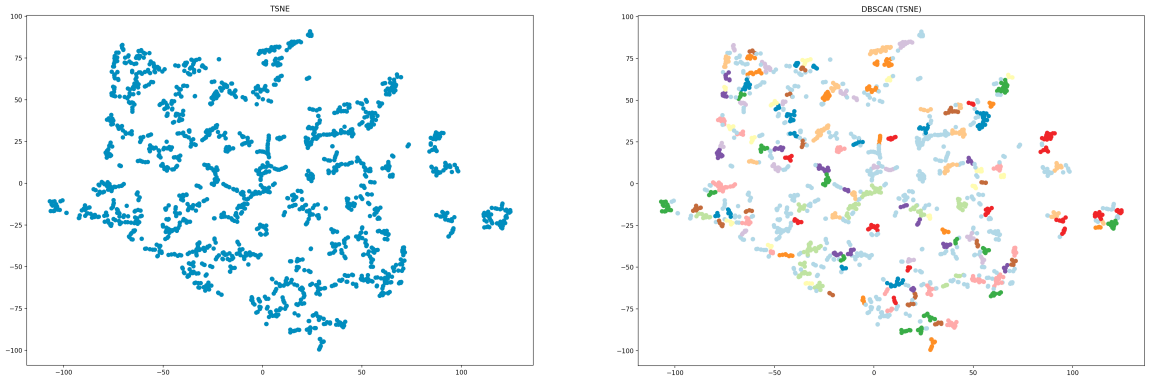


Figure 32: **1h** data files, t-SNE calculated with the following parameters: **perplexity=10**, n_iter=5000, learning_rate=50

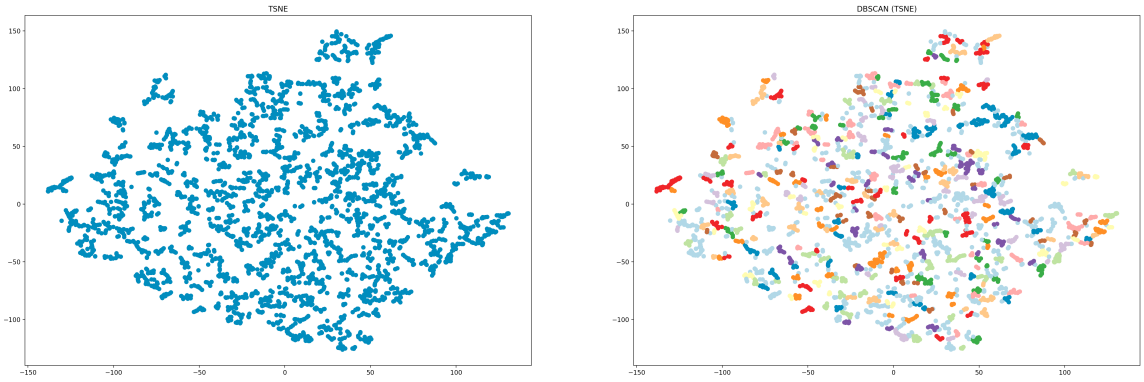


Figure 33: **3h** data files, t-SNE calculated with the following parameters: **perplexity=10**, n_iter=5000, learning_rate=50

A.1.3 Perplexity = 20

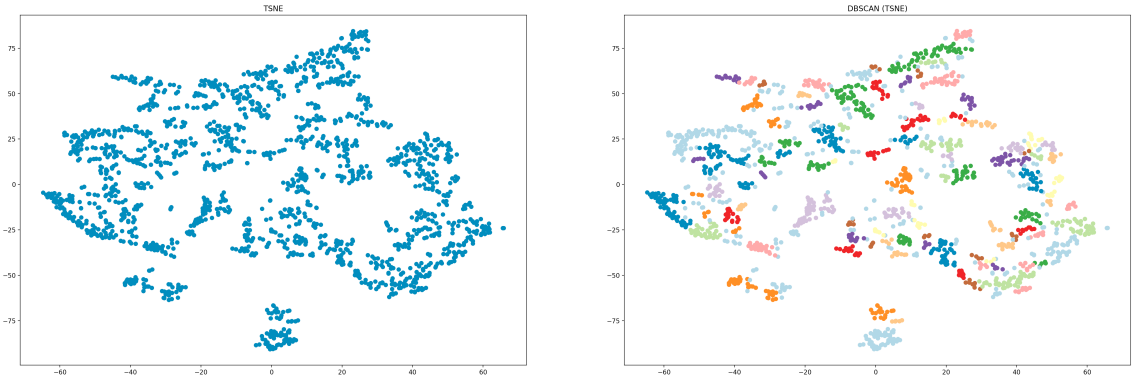


Figure 34: **1h** data files, t-SNE calculated with the following parameters: **perplexity=20**, n_iter=5000, learning_rate=50

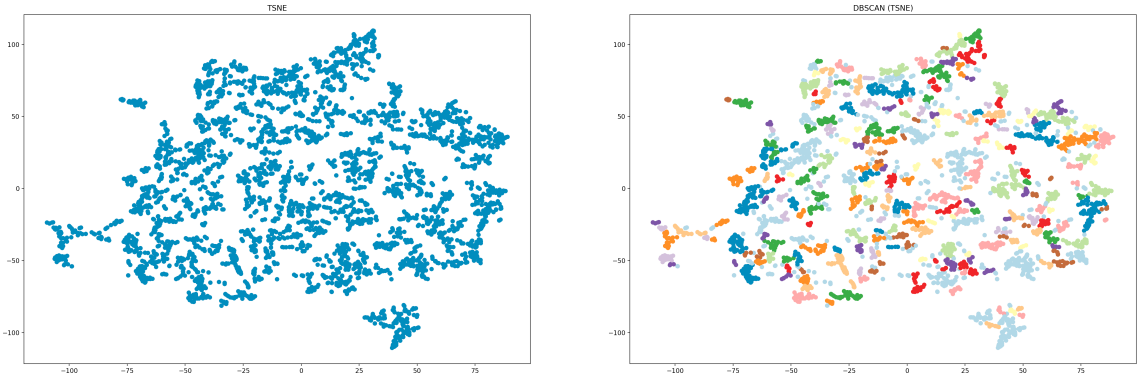


Figure 35: **3h** data files, t-SNE calculated with the following parameters: **perplexity=20**, n_iter=5000, learning_rate=50

A.1.4 Perplexity = 30

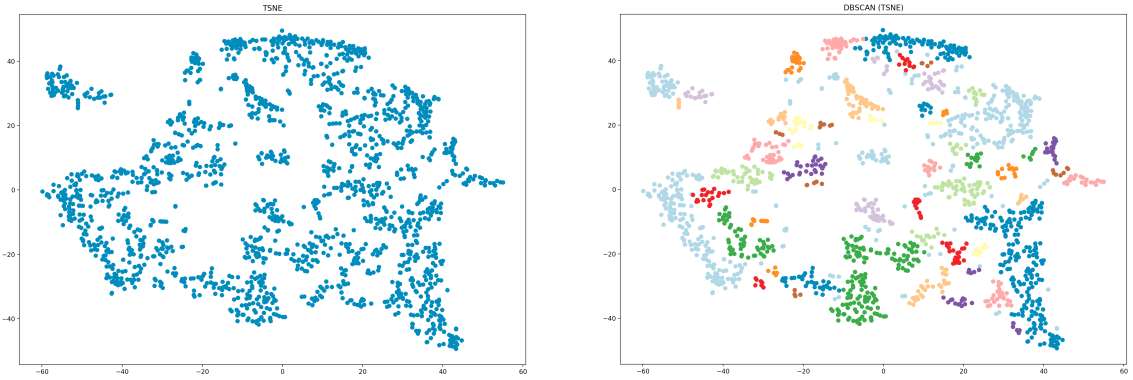


Figure 36: **1h** data files, t-SNE calculated with the following parameters: **perplexity=30**, n_iter=5000, learning_rate=50

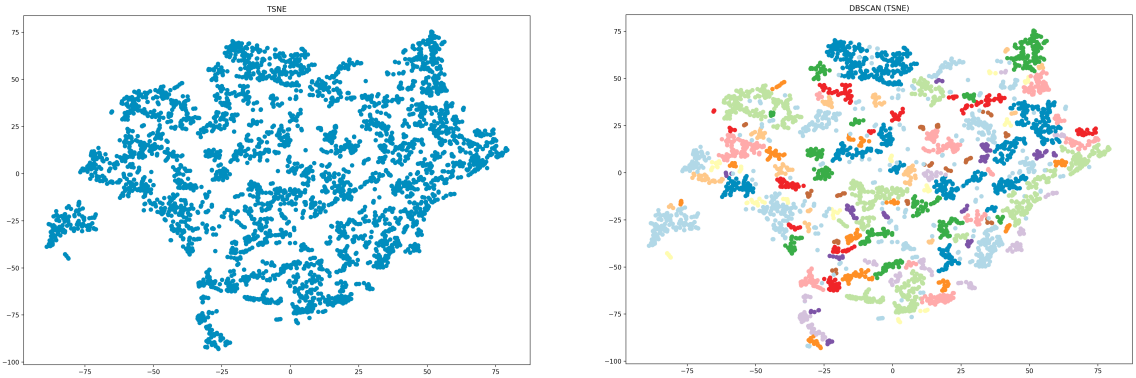


Figure 37: **3h** data files, t-SNE calculated with the following parameters: **perplexity=30**, n_iter=5000, learning_rate=50

A.1.5 Perplexity = 40

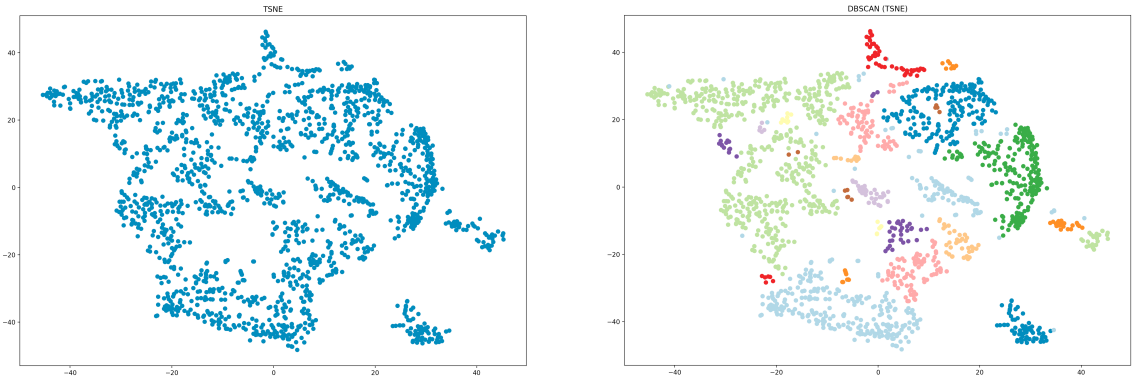


Figure 38: **1h** data files, t-SNE calculated with the following parameters: **perplexity=40**, n_iter=5000, learning_rate=50

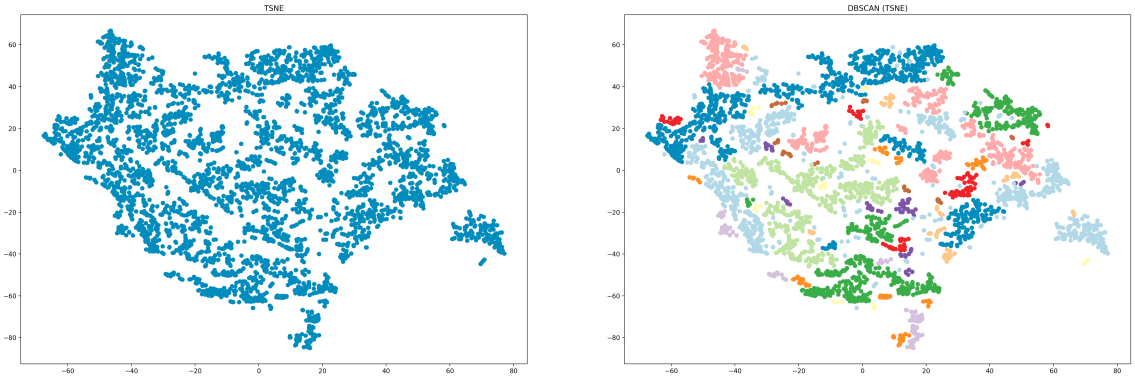


Figure 39: **3h** data files, t-SNE calculated with the following parameters: **perplexity=40**, n_iter=5000, learning_rate=50

A.1.6 Perplexity = 45

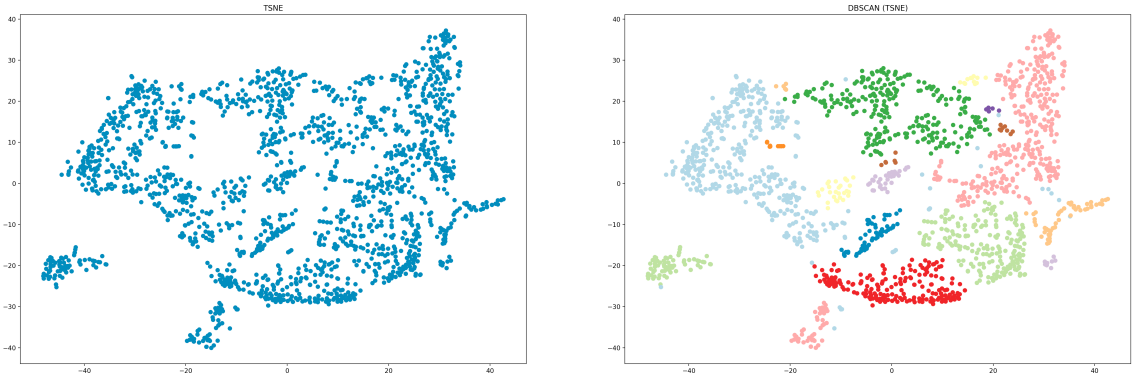


Figure 40: **1h** data files, t-SNE calculated with the following parameters: **perplexity=45**, n_iter=5000, learning_rate=50

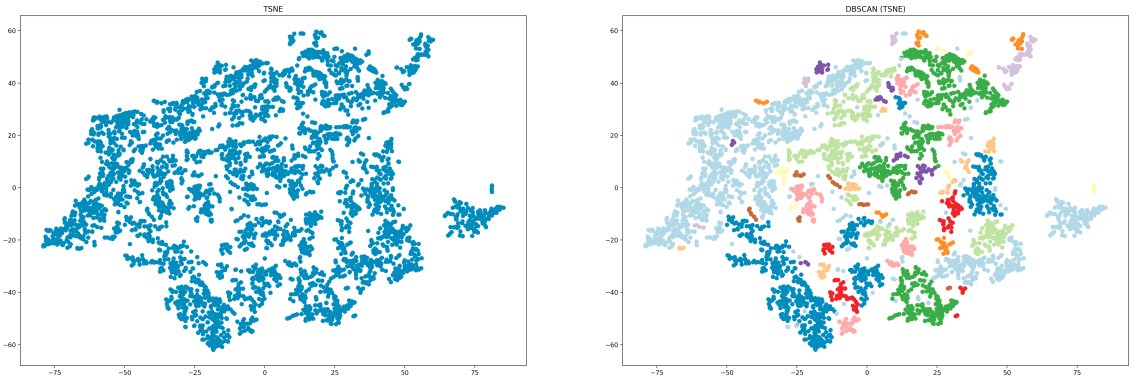


Figure 41: **3h** data files, t-SNE calculated with the following parameters: **perplexity=45**, n_iter=5000, learning_rate=50

A.1.7 Perplexity = 50

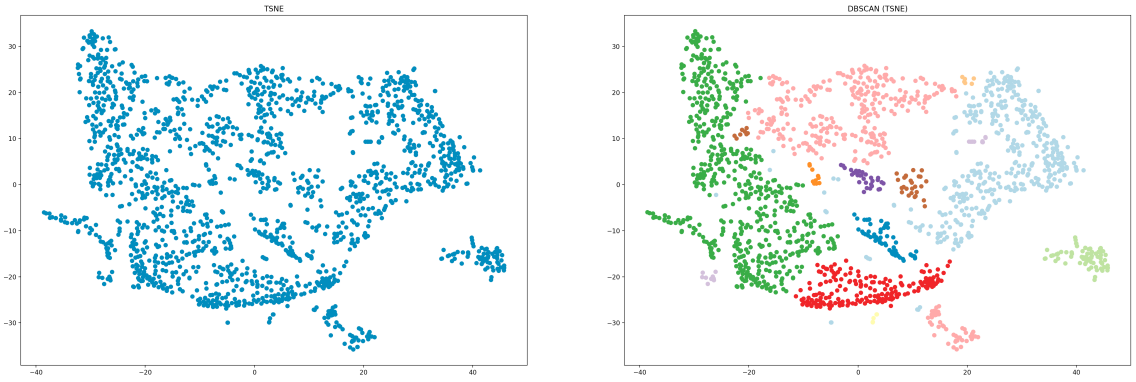


Figure 42: **1h** data files, t-SNE calculated with the following parameters: **perplexity=50**, n_iter=5000, learning_rate=50

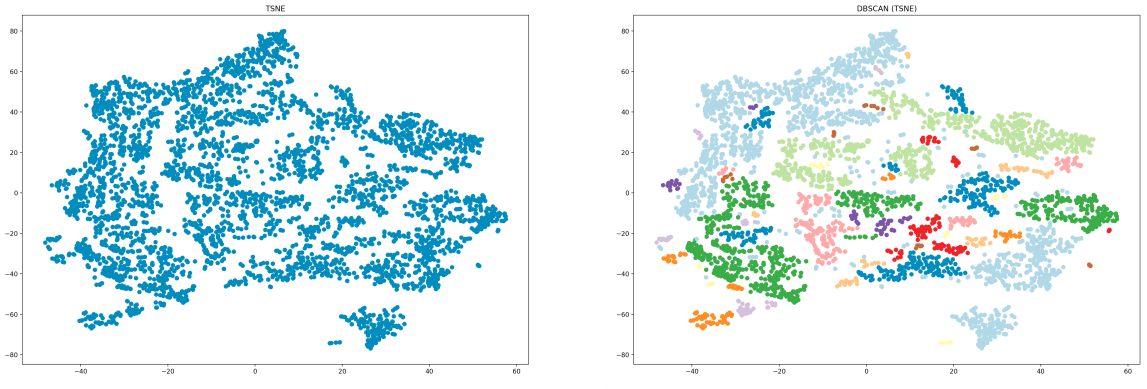


Figure 43: **3h** data files, t-SNE calculated with the following parameters: **perplexity=50**, n_iter=5000, learning_rate=50

A.1.8 Perplexity Comparison Results (Average of two different t-SNE runs)

1h Files							
	Perplexity: 5	Perplexity: 10	Perplexity: 20	Perplexity: 30	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)							
Silhouette	0.017019369	0.19126232	0.257092565	0.17063424	0.054603353	-0.074523993	-0.171010017
Davies Bouldin	1.403668307	1.446398049	1.239208781	1.767317317	1.84461589	2.479778037	1.825281178
Calinski Harabasz	12.15170592	24.79178797	100.0146068	241.9789113	563.5035835	437.3758651	329.6007263
OPTICS							
Silhouette	-0.561546922	-0.246936843	0.072907798	0.110969819	0.034138191	-0.048403323	-0.168361336
Davies Bouldin	1.206983736	1.477320921	1.379751733	2.01127775	1.841140623	2.805988833	2.087488648
Calinski Harabasz	6.242372565	11.29904904	34.8081723	121.1196102	390.1107083	443.3149674	288.6146754
3h Files							
	Perplexity: 5	Perplexity: 10	Perplexity: 20	Perplexity: 30	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)							
Silhouette	0.187188387	0.292015851	0.285667479	0.193456471	0.031343713	-0.053399093	-0.181974486
Davies Bouldin	1.378079722	1.360209014	1.297094447	1.442256841	1.735051282	1.795796982	1.483932052
Calinski Harabasz	18.05272734	41.40515662	125.5679156	324.4924046	603.9489233	431.4155601	294.2764864
OPTICS							
Silhouette	-0.345607281	-0.027698424	0.14453043	0.16002439	0.060340196	-0.001508603	-0.110961363
Davies Bouldin	1.322159387	1.378912049	1.286376249	1.525305635	1.862524881	1.703454207	2.190983827
Calinski Harabasz	8.067148543	16.79004273	49.04119384	134.4265238	431.4900781	510.0124077	454.230048
	best values in files (1h or 3h)						
	best values total (1h and 3h)						

Figure 44: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **perplexities**.

1h Files			
	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)			
Silhouette	0.054603353	-0.074523993	-0.171010017
Davies Bouldin	1.84461589	2.479778037	1.825281178
Calinski Harabasz	563.5035835	437.3758651	329.6007263
OPTICS			
Silhouette	0.034138191	-0.048403323	-0.168361336
Davies Bouldin	1.841140623	2.805988833	2.087488648
Calinski Harabasz	390.1107083	443.3149674	288.6146754
3h Files			
	Perplexity: 40	Perplexity: 45	Perplexity: 50
DBSCAN (TSNE)			
Silhouette	0.031343713	-0.053399093	-0.181974486
Davies Bouldin	1.735051282	1.795796982	1.483932052
Calinski Harabasz	603.9489233	431.4155601	294.2764864
OPTICS			
Silhouette	0.060340196	-0.001508603	-0.110961363
Davies Bouldin	1.862524881	1.703454207	2.190983827
Calinski Harabasz	431.4900781	510.0124077	454.230048
	best values in files (1h or 3h)		
	best values total (1h and 3h)		

Figure 45: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **perplexities**.

A.2 Learning Rate

A.2.1 Learning Rate = 10

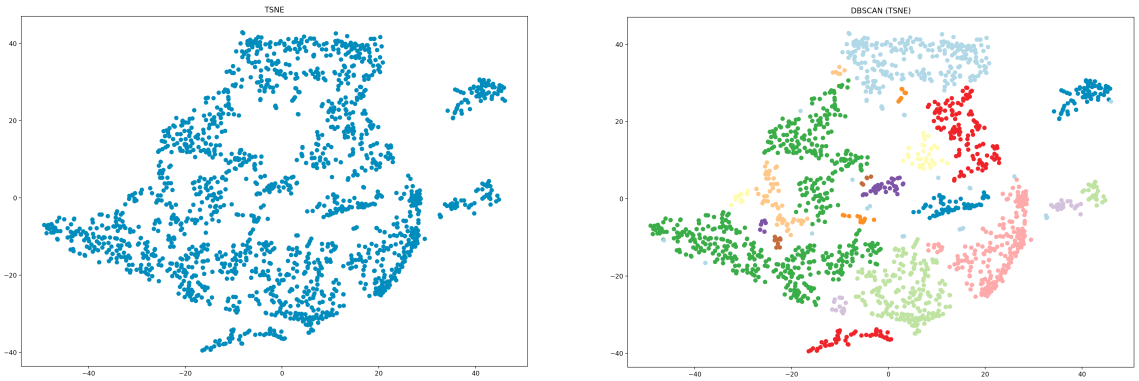


Figure 46: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=10**

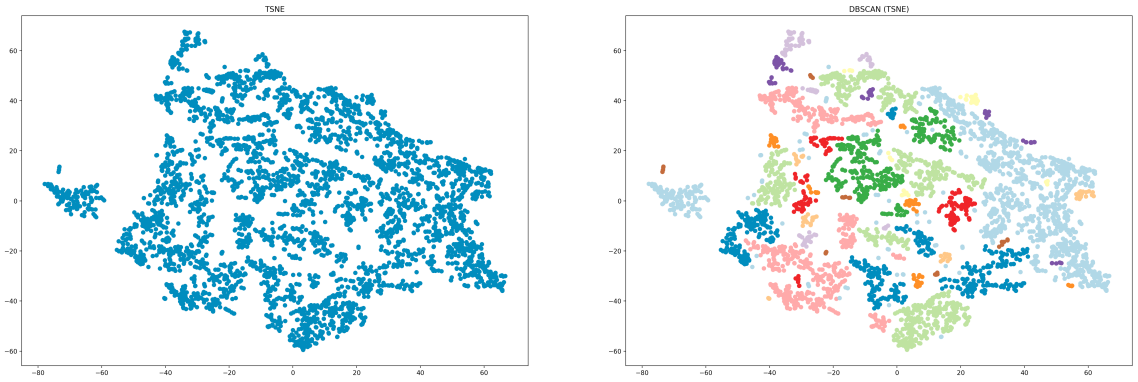


Figure 47: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=10**

A.2.2 Learning Rate = 200

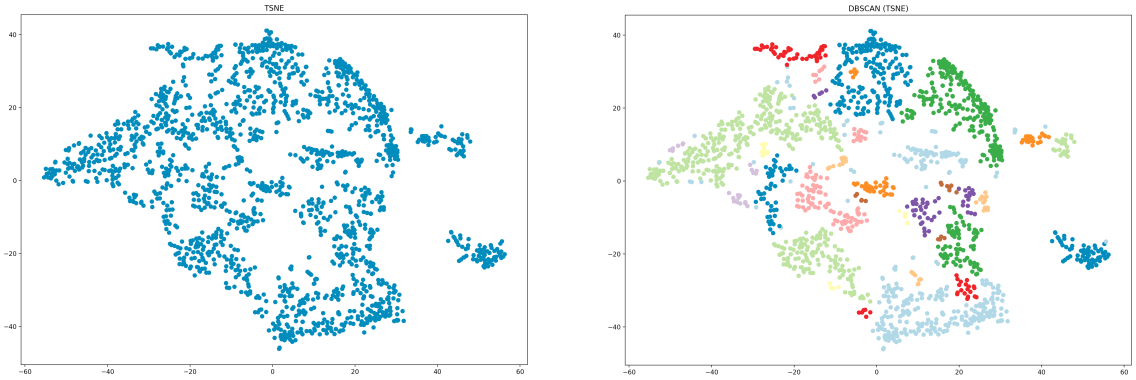


Figure 48: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=200**

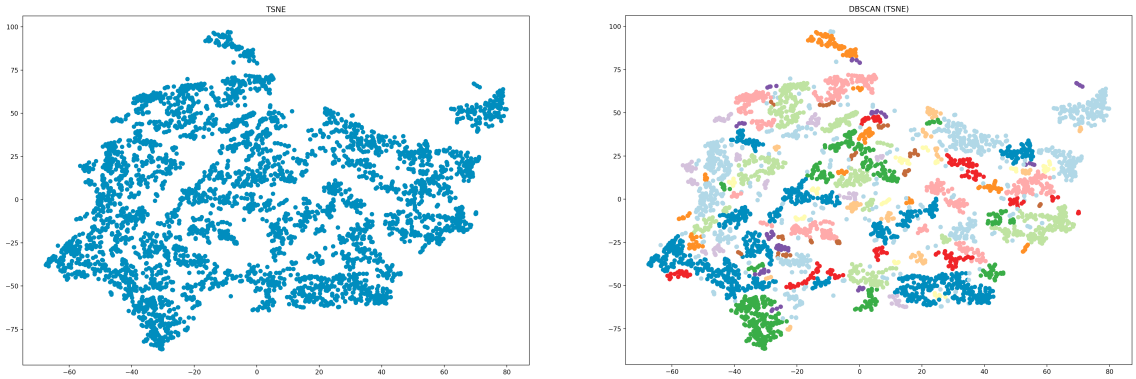


Figure 49: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=200**

A.2.3 Learning Rate = 400

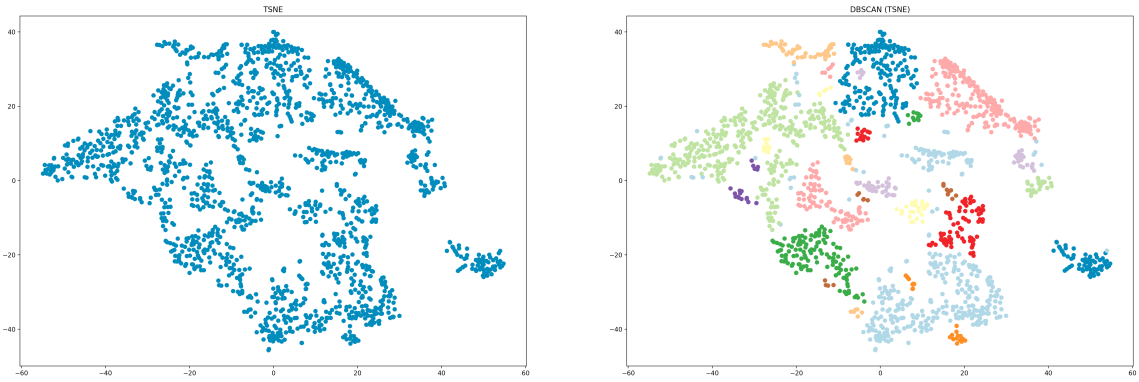


Figure 50: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=400**

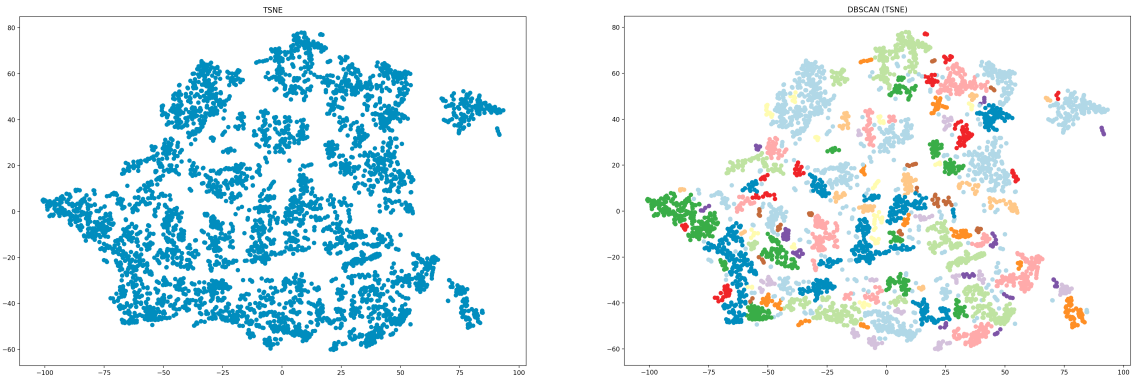


Figure 51: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=400**

A.2.4 Learning Rate = 600

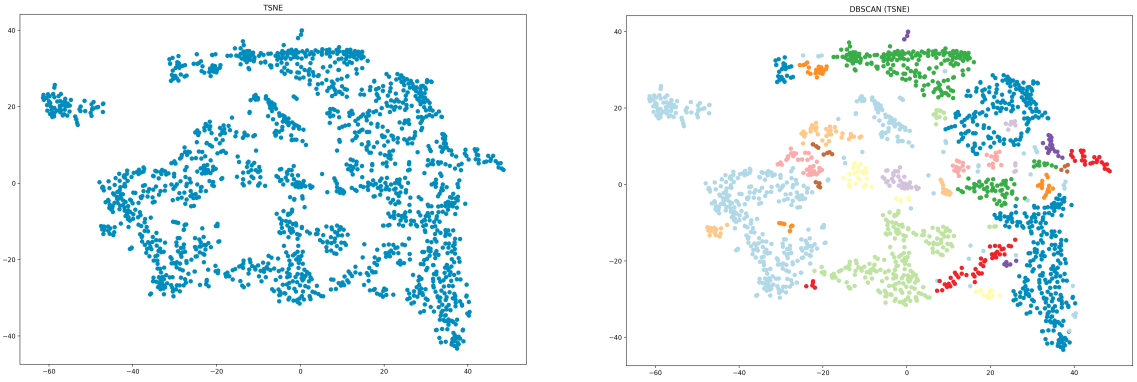


Figure 52: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=600**

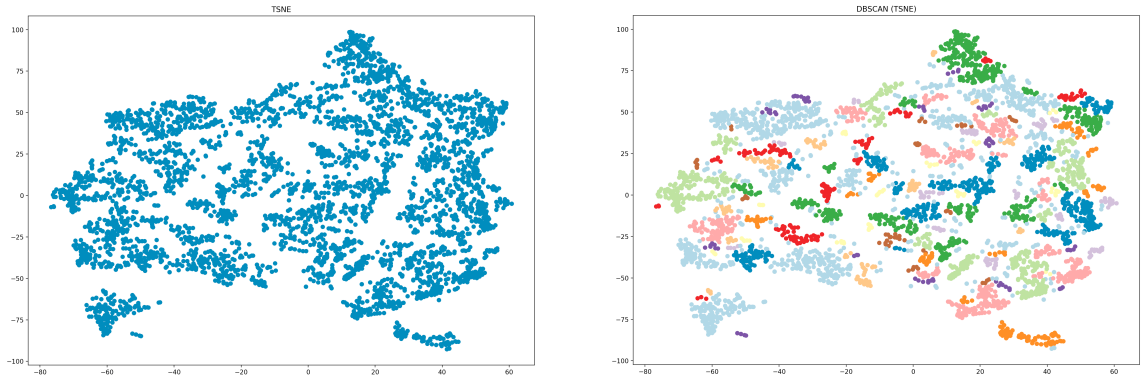


Figure 53: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=600**

A.2.5 Learning Rate = 800

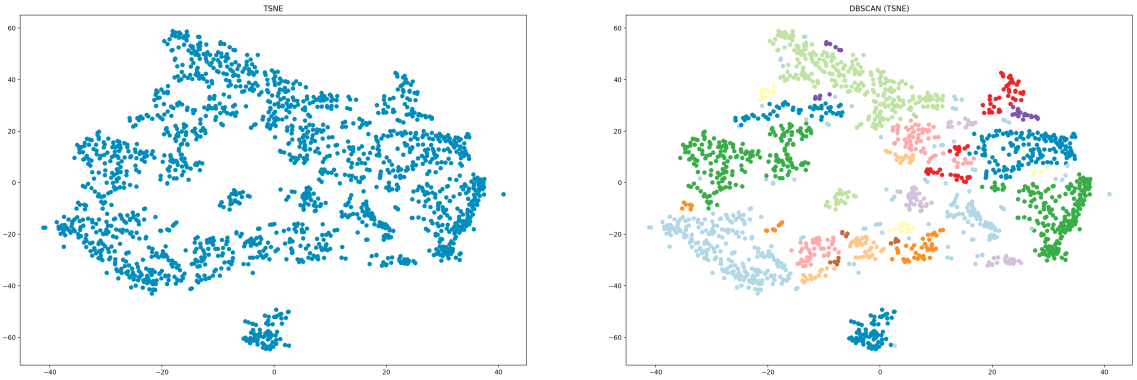


Figure 54: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=800**

Figure 55: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=800**

A.2.6 Learning Rate = 1000

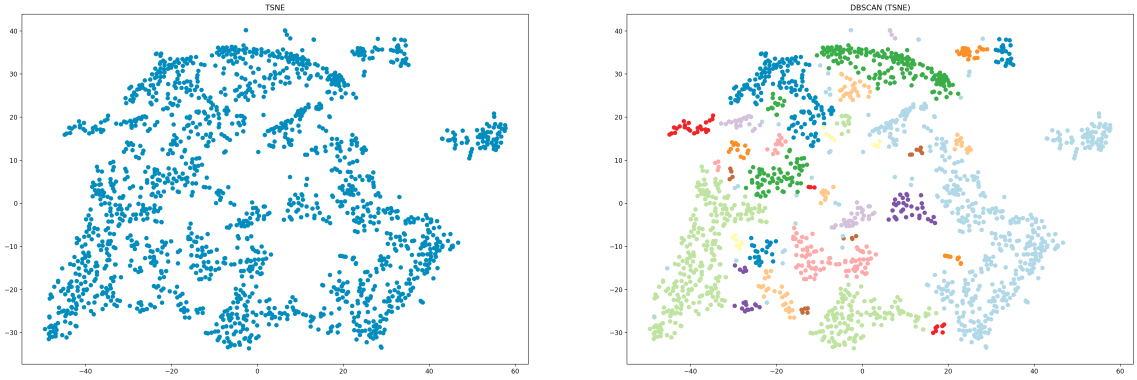


Figure 56: **1h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=1000**

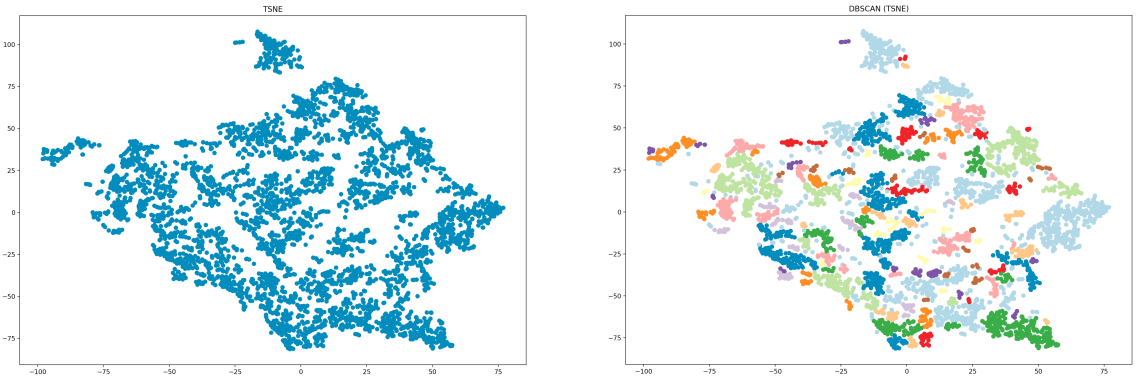


Figure 57: **3h** data files, t-SNE calculated with the following parameters: perplexity=40, n_iter=5000, **learning_rate=1000**

1h Files

	Learning Rate: 10	Learning Rate: 50	Learning Rate: 100	Learning Rate: 150	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	0.050185006	0.020629907	0.043174922	0.00881206	0.037798844
Davies Bouldin	1.813219735	1.765516415	1.792311036	1.84207799	1.537629516
Calinski Harabasz	619.5024641	527.1113507	508.9986796	477.6877339	412.8616309
OPTICS					
Silhouette	0.04723522	0.028060276	0.014973388	-0.010167263	0.036134463
Davies Bouldin	2.644154104	1.627679428	1.943985252	2.038386818	1.512853515
Calinski Harabasz	545.2578195	427.7636283	384.5284878	321.2113347	243.8295466

3h Files

	Learning Rate: 10	Learning Rate: 50	Learning Rate: 100	Learning Rate: 150	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.053747308	0.005408226	0.08698505	0.056991242	0.08301577
Davies Bouldin	2.315636084	1.595251196	1.65346247	1.51375092	1.40672888
Calinski Harabasz	495.6689234	821.6969448	632.2392584	519.0302785	325.3370665
OPTICS					
Silhouette	-0.002017022	0.06374386	0.08698636	0.091826566	0.08165444
Davies Bouldin	2.196366077	1.600873186	1.609330153	1.509671627	1.485960484
Calinski Harabasz	607.8271457	538.5429769	275.9978949	245.4457569	130.86284

best values in files (1h or 3h)

best values total (1h and 3h)

Figure 58: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

1h Files

	Learning Rate: 10	Learning Rate: 20	Learning Rate: 30	Learning Rate: 40	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	0.050185006	-0.06679525	0.038768608	0.01700433	0.037798844
Davies Bouldin	1.813219735	1.525274298	1.939580286	1.772676736	1.537629516
Calinski Harabasz	619.5024641	248.4124961	577.4117352	551.6767392	412.8616309
OPTICS					
Silhouette	0.04723522	-0.01730663	0.07930727	0.047520984	0.036134463
Davies Bouldin	2.644154104	1.375044564	1.521993726	1.630649614	1.512853515
Calinski Harabasz	545.2578195	253.4888154	453.56707	436.5401766	243.8295466

3h Files

	Learning Rate: 10	Learning Rate: 20	Learning Rate: 30	Learning Rate: 40	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.053747308	-0.014693906	0.09359318	0.028022699	0.08301577
Davies Bouldin	2.315636084	1.624380463	1.698168632	1.619440691	1.40672888
Calinski Harabasz	495.6689234	441.8649747	819.1717685	741.2875739	325.3370665
OPTICS					
Silhouette	-0.002017022	0.08187237	0.0860974	0.06130342	0.08165444
Davies Bouldin	2.196366077	1.767960982	1.85835621	1.767468795	1.485960484
Calinski Harabasz	607.8271457	677.3212302	455.2848506	429.6428288	130.86284

best values in files (1h or 3h)

best values total (1h and 3h)

Figure 59: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

A.2.7 Learning Rate Detailed Comparison Results

A.2.8 Learning Rate Comparison Results (Average of two different t-SNE runs)

1h Files						
	Learning Rate: 10	Learning Rate: 200	Learning Rate: 400	Learning Rate: 600	Learning Rate: 800	Learning Rate: 1000
DBSCAN (TSNE)						
Silhouette	-0.003757648	0.010789386	-0.018090254	-0.053359557	0.056309521	0.036684237
Davies Bouldin	2.196037766	1.802399748	2.290698136	1.896887987	1.77401152	1.703924631
Calinski Harabasz	543.0662147	461.4313593	427.1877972	384.7281333	432.2476651	485.0086521
OPTICS						
Silhouette	0.031175781	-0.000287511	0.017186441	0.036526404	0.075913355	0.088797659
Davies Bouldin	1.632716286	1.595774028	1.72943669	1.788465571	1.626137502	1.559785165
Calinski Harabasz	499.5216878	297.430407	297.1071965	347.6386925	298.9557343	325.8533935
3h Files						
	Learning Rate: 10	Learning Rate: 200	Learning Rate: 400	Learning Rate: 600	Learning Rate: 800	Learning Rate: 1000
DBSCAN (TSNE)						
Silhouette	-0.047304191	0.059058443	0.052361935	0.055812217	0.067903891	0.092090182
Davies Bouldin	1.958708445	1.509463728	1.486604812	1.485684064	1.476445909	1.401655093
Calinski Harabasz	478.6291387	445.8470406	454.0323481	368.3546676	369.9180331	339.4951441
OPTICS						
Silhouette	0.063918628	0.070420966	0.067922488	0.084858529	0.078014418	0.081889041
Davies Bouldin	1.908380671	1.563361642	1.739927567	1.535153239	1.521979543	1.456861854
Calinski Harabasz	649.3951529	203.4380081	208.4273457	157.9961534	150.9460601	137.0631837
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 60: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

1h Files					
	Learning Rate: 10	Learning Rate: 50	Learning Rate: 100	Learning Rate: 150	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.003757648	0.034104612	0.023020575	0.021991953	0.056309521
Davies Bouldin	2.196037766	1.839113174	1.918253028	2.077872768	1.77401152
Calinski Harabasz	543.0662147	487.2372447	408.9806962	495.7494691	432.2476651
OPTICS					
Silhouette	0.031175781	0.016049685	0.030655831	0.006348168	0.075913355
Davies Bouldin	1.632716286	1.703277523	1.917903749	1.702313578	1.626137502
Calinski Harabasz	499.5216878	360.6840936	308.6386538	307.6976429	298.9557343

3h Files					
	Learning Rate: 10	Learning Rate: 50	Learning Rate: 100	Learning Rate: 150	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.047304191	0.046594031	0.047196072	0.046747934	0.067903891
Davies Bouldin	1.958708445	1.47995198	1.561185204	1.463983483	1.476445909
Calinski Harabasz	478.6291387	679.3870954	557.5937578	611.2247497	369.9180331
OPTICS					
Silhouette	0.063918628	0.075235315	0.067135818	0.066724166	0.078014418
Davies Bouldin	1.908380671	1.625187367	1.58668401	1.589883483	1.521979543
Calinski Harabasz	649.3951529	379.5759629	290.1587463	324.4484223	150.9460601

	best values in files (1h or 3h)
	best values total (1h and 3h)

Figure 61: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

1h Files

	Learning Rate: 10	Learning Rate: 20	Learning Rate: 30	Learning Rate: 40	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.003757648	0.051418006	0.037215631	0.024115672	0.056309521
Davies Bouldin	2.196037766	1.830794508	2.060517734	2.123406309	1.77401152
Calinski Harabasz	543.0662147	609.8260254	564.2559305	569.1097081	432.2476651
OPTICS					
Silhouette	0.031175781	0.085824259	0.050168812	0.042932201	0.075913355
Davies Bouldin	1.632716286	1.562171815	1.75508086	1.576809691	1.626137502
Calinski Harabasz	499.5216878	518.8980998	440.7310544	459.9814168	298.9557343

3h Files

	Learning Rate: 10	Learning Rate: 20	Learning Rate: 30	Learning Rate: 40	Learning Rate: 800
DBSCAN (TSNE)					
Silhouette	-0.047304191	-0.006364155	0.044669248	0.046343155	0.067903891
Davies Bouldin	1.958708445	1.881712505	2.12244492	1.749100916	1.476445909
Calinski Harabasz	478.6291387	540.0932863	774.4791625	658.1757762	369.9180331
OPTICS					
Silhouette	0.063918628	0.050251506	0.076243207	0.049795788	0.078014418
Davies Bouldin	1.908380671	1.931390827	2.018979173	1.789714946	1.521979543
Calinski Harabasz	649.3951529	463.7734489	481.6214212	389.1751855	150.9460601

best values in files (1h or 3h)

best values total (1h and 3h)

Figure 62: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

1h Files					
	Learning Rate: 10	Learning Rate: 15	Learning Rate: 20	Learning Rate: 25	Learning Rate: 30
DBSCAN (TSNE)					
Silhouette	-0.003757648	0.040198717	0.051418006	0.032236848	0.037215631
Davies Bouldin	2.196037766	1.814792808	1.830794508	1.936491197	2.060517734
Calinski Harabasz	543.0662147	591.6592988	609.8260254	511.6861804	564.2559305
OPTICS					
Silhouette	0.031175781	0.067535594	0.085824259	0.038215108	0.050168812
Davies Bouldin	1.632716286	1.659834614	1.562171815	1.758027983	1.75508086
Calinski Harabasz	499.5216878	504.9991816	518.8980998	389.5913488	440.7310544
3h Files					
	Learning Rate: 10	Learning Rate: 15	Learning Rate: 20	Learning Rate: 25	Learning Rate: 30
DBSCAN (TSNE)					
Silhouette	-0.047304191	0.010419452	0.008094903	-0.012381725	0.044669248
Davies Bouldin	1.958708445	1.875585513	1.897687221	1.75000284	2.12244492
Calinski Harabasz	478.6291387	656.9170813	704.0852625	639.2657184	774.4791625
OPTICS					
Silhouette	0.063918628	0.056145657	0.06045679	0.061573535	0.076243207
Davies Bouldin	1.908380671	1.880787198	1.796566584	1.860657595	2.018979173
Calinski Harabasz	649.3951529	648.5003401	590.1095848	661.0214236	481.6214212
	best values in files (1h or 3h)				
	best values total (1h and 3h)				

Figure 63: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values.

A.2.9 Learning Rate Comparison of 20 and 800

1h Files								
	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800
DBSCAN (TSNE)								
Silhouette	0.051418006	0.056309521	0.047943175	-0.033673551	0.010282498	0.044229511	0.024733488	0.13869074
Davies Bouldin	1.830794508	1.77401152	1.874245766	3.045774889	2.077463481	1.680621133	1.844101414	1.659377639
Calinski Harabasz	609.8260254	432.2476651	628.4310062	284.5810113	555.3438459	457.8654554	543.4246332	561.299456
OPTICS								
Silhouette	0.085824259	0.075913355	0.076782033	0.023241794	0.06412942	0.06140846	0.057013668	0.145829111
Davies Bouldin	1.562171815	1.626137502	1.661925846	3.121565966	1.616248147	1.618311235	1.652082936	1.913545308
Calinski Harabasz	518.8980998	298.9557343	509.1943472	298.6138012	486.7785298	329.8427659	464.8406325	402.876503
3h Files								
	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800	Learning Rate: 20	Learning Rate: 800
DBSCAN (TSNE)								
Silhouette	-0.006364155	0.067903891	0.016517494	0.08418332	0.003429756	0.054691602	0.003097915	0.083377987
Davies Bouldin	1.881712505	1.476445909	2.090600357	1.534382164	1.923601101	1.437163727	2.006669308	1.516726218
Calinski Harabasz	540.0932863	369.9180331	696.4164007	349.0248311	650.4633185	359.6352328	763.8873521	304.9316986
OPTICS								
Silhouette	0.050251506	0.078014418	0.058208089	0.098020062	0.07693895	0.076097637	0.030158926	0.069448814
Davies Bouldin	1.931390827	1.521979543	1.918330176	1.493803848	1.693677609	1.505156616	1.956116792	1.578533436
Calinski Harabasz	463.7734489	150.9460601	453.7829979	138.1019652	568.1096084	149.4634161	488.459881	128.4181964
	best values in files (1h or 3h)							
	best values total (1h and 3h)							

Figure 64: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for the t-SNE **learning rate** values 20 and 80.

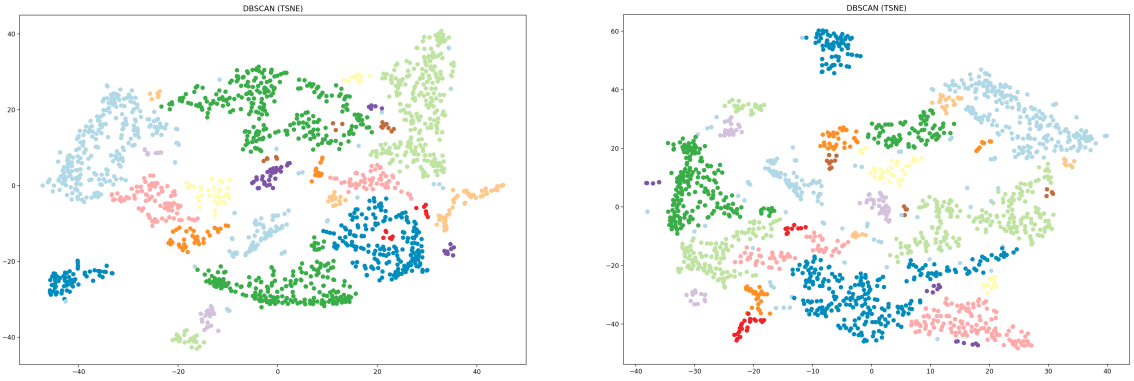


Figure 65: **1h** data files comparison of learning rate: a) 20, b) 800

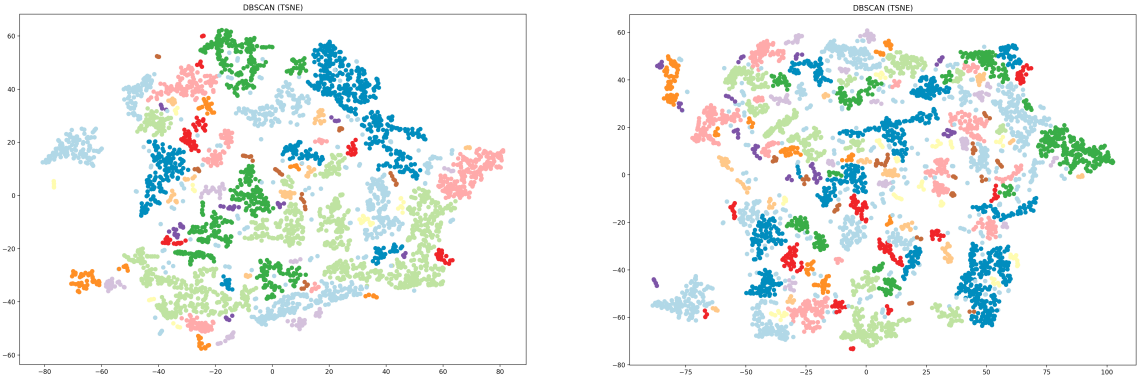


Figure 66: **3h** data files comparison of learning rate: a) 20, b) 800

A.3 Clustering results

A.3.1 1h aggregated data files

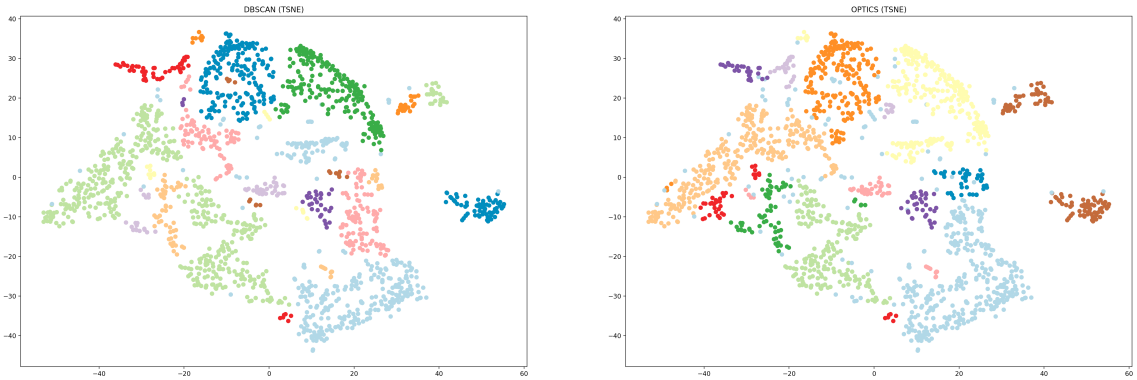


Figure 67: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the 1st column, so the first **15 minutes** (1h data files: first 15 minutes).

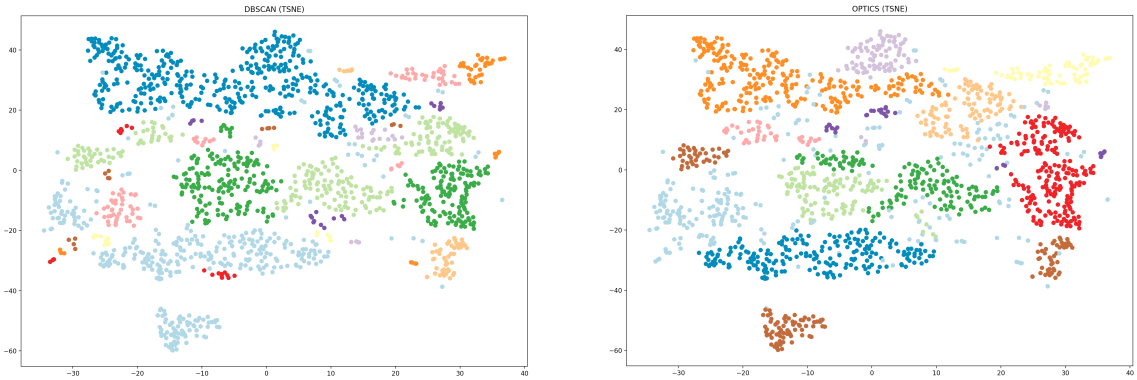


Figure 68: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column and 2nd column, so the first **30 minutes** (1h data files: 15 minutes & 30 minutes).

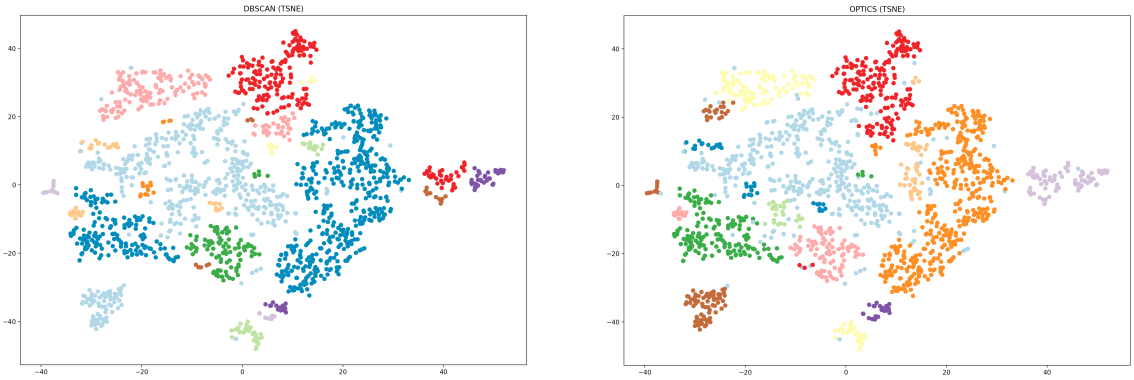


Figure 69: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 3rd column, so the first **45 minutes** (1h data files: 15 minutes, 30 minutes & 45 minutes).

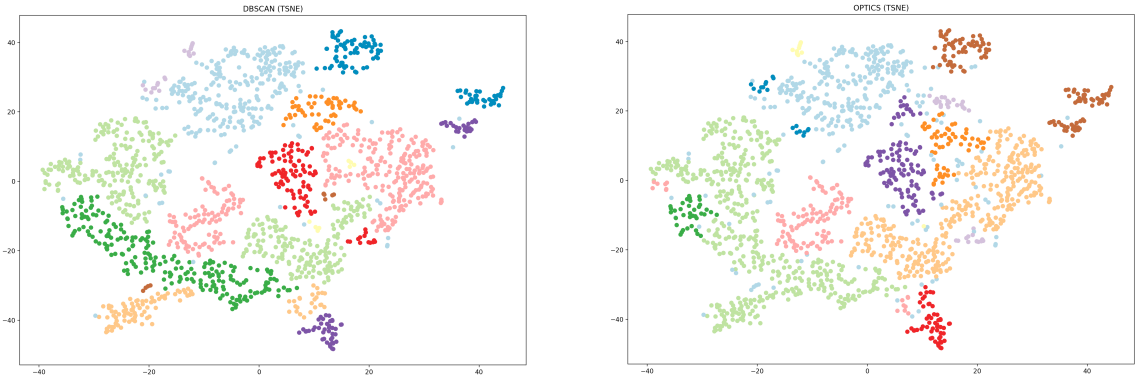


Figure 70: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 4th column, so the whole **1 hour** (1h data files: 15 minutes, 30 minutes, 45 minutes & 1 hour).

A.3.2 3h aggregated data files

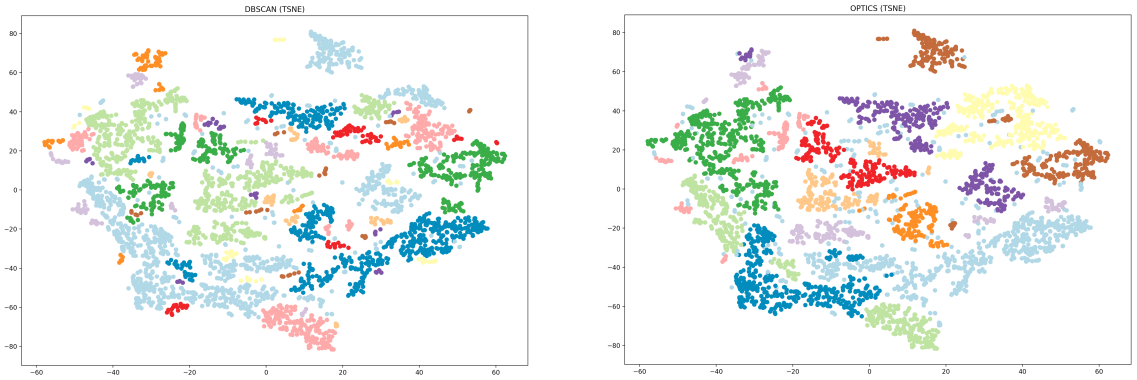


Figure 71: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the 1st column, so the first **30 minutes** (3h data files: first 30 minutes).

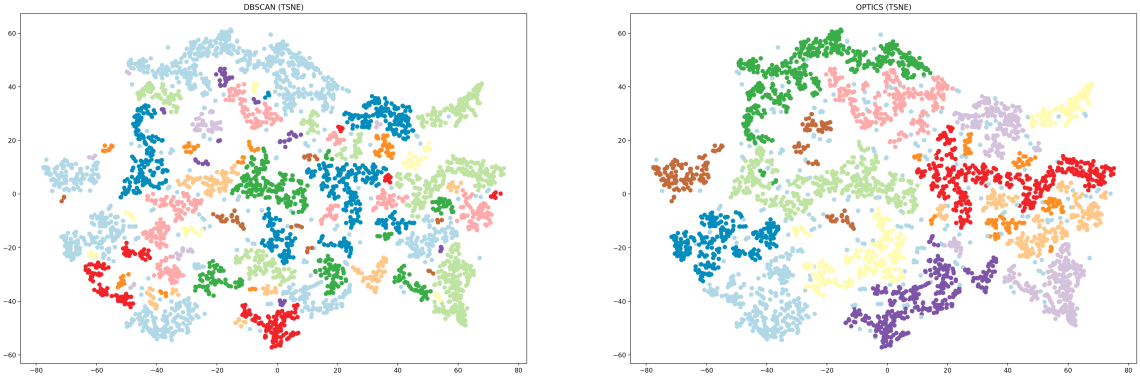


Figure 72: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column and 2nd column, so the first **1 hour** (3h data files: 30 minutes & 1 hour).

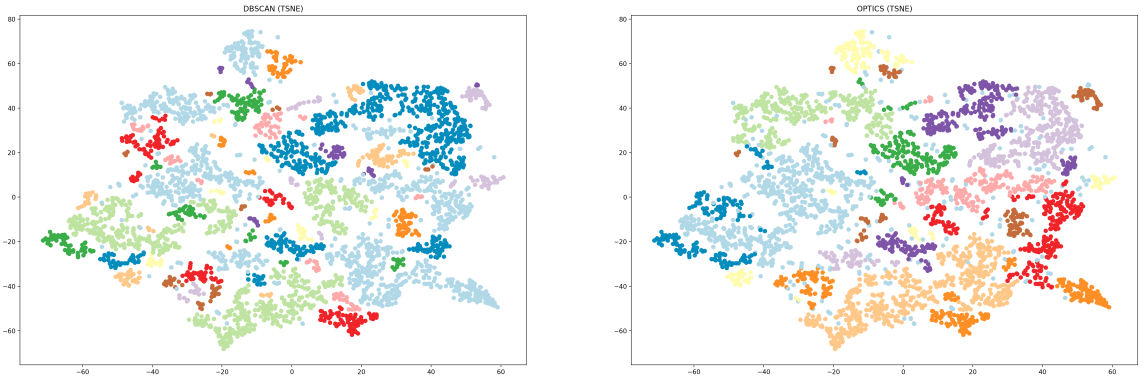


Figure 73: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 3rd column, so the first **1.5 hours** (3h data files: 30 minutes, 1 hour & 1 hour 30 minutes).

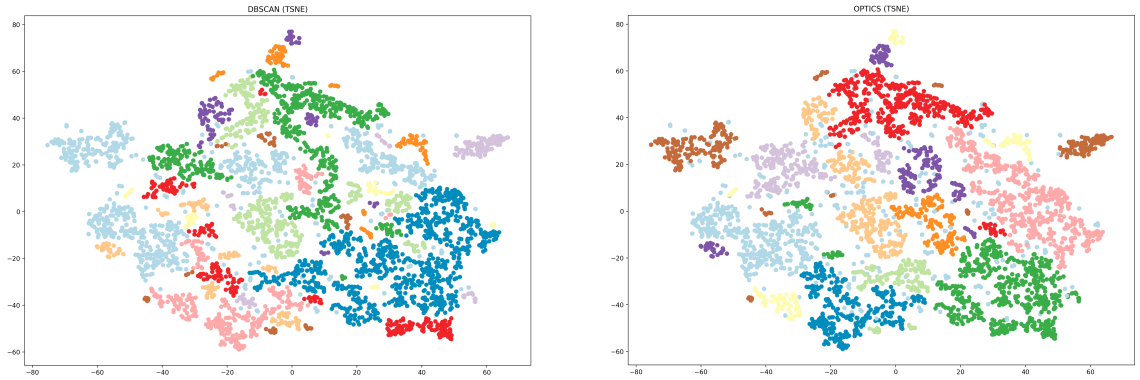


Figure 74: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 4th column, so the first **2 hours** (3h data files: 30 minutes, 1 hour, 1 hour 30 minutes & 2 hours).

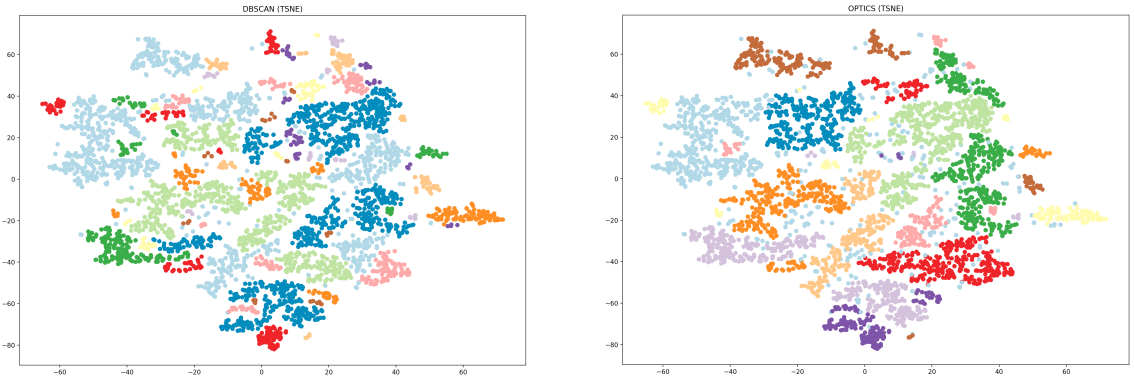


Figure 75: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 5th column, so the first **2.5 hours** (3h data files: 30 minutes, 1 hour, 1 hour 30 minutes, 2 hours & 2 hours 30 minutes).

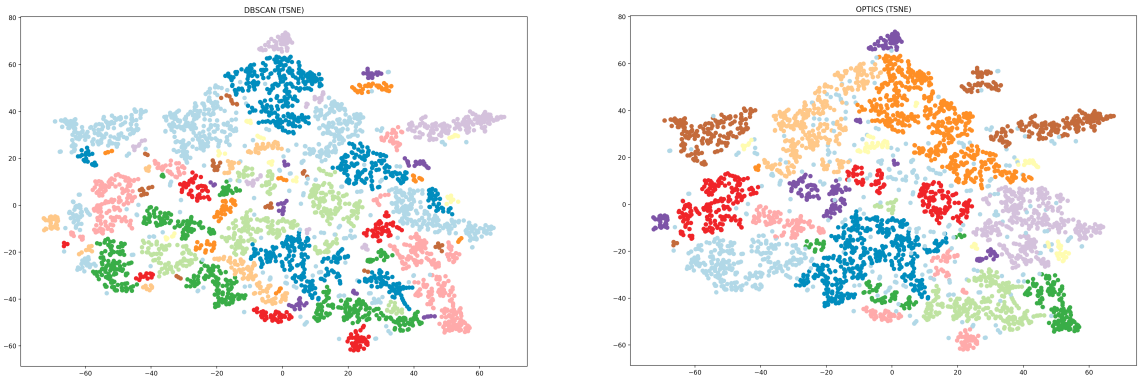


Figure 76: Comparison of the scatter plots from the DBSCAN (a) and OPTICS (b) clusterings of the average of the 1st column to the 6th column, so all **3 hours** (3h data files: 30 minutes, 1 hour, 1 hour 30 minutes, 2 hours, 2 hours 30 minutes & 3 hours).

A.4 Clustering evaluation results

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.04409917	-0.09849845	-0.0932329	-0.23330191		
Davies Bouldin	1.965553013	1.408535372	1.549759164	2.03076771		
Calinski Harabasz	379.250218	237.2311173	313.5420497	151.2110635		
OPTICS						
Silhouette	0.0790162	0.15246919	-0.05314931	0.042187728		
Davies Bouldin	1.909221804	1.432838844	1.604143722	1.487652613		
Calinski Harabasz	341.9348451	439.4869733	248.8970787	295.5351996		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	0.03163854	0.025676515	-0.02877813	0.063989125	-0.006347847	0.039090224
Davies Bouldin	1.595225606	1.386905334	1.831308621	1.565283337	1.685115208	1.765928396
Calinski Harabasz	660.300021	912.1243489	665.2710749	644.7382201	781.7245551	633.2488309
OPTICS						
Silhouette	0.05099258	0.06202586	0.07631619	0.11866711	0.08885	0.09892428
Davies Bouldin	1.640303918	1.719579308	1.708319079	1.722106779	1.419781716	1.47188292
Calinski Harabasz	700.3707789	523.0438423	504.3994953	418.4628186	506.8636832	340.4418048
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 77: Evaluation scores comparison from 1h run of t-SNE and clustering with a learning rate of 20.

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.061859984	0.049978126	0.007422133	-0.01603307		
Davies Bouldin	1.721318764	1.477412627	2.293371608	1.556510771		
Calinski Harabasz	587.3448151	314.7345941	363.751391	434.4817134		
OPTICS						
Silhouette	0.10139497	0.14339069	-0.000247999	0.091312036		
Davies Bouldin	1.851297487	1.21728504	2.107907761	1.483656401		
Calinski Harabasz	488.1085954	356.6377613	291.1266469	371.4243781		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	-0.002485462	0.02854126	-0.016798861	0.027074102	0.03494797	0.046173148
Davies Bouldin	1.938083017	1.452023108	1.878183603	1.401770691	1.643617425	1.593027178
Calinski Harabasz	467.7627938	881.5570846	507.1314968	579.5123589	627.0284348	724.8436508
OPTICS						
Silhouette	0.07204221	0.094111204	0.077061266	0.028278682	0.11853256	0.07739853
Davies Bouldin	2.196047237	1.783584125	1.876588633	1.401717031	1.486444695	2.063883814
Calinski Harabasz	707.6414399	586.3869193	353.3416269	336.8332351	406.5664744	433.4674822
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 78: Evaluation scores comparison from 1h run of t-SNE and clustering with a learning rate of 20.

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.052979577	-0.024260162	-0.042905383	-0.12466749		
Davies Bouldin	1.843435889	1.442973999	1.921565386	1.79363924		
Calinski Harabasz	483.2975165	275.9828557	338.6467203	292.8463885		
OPTICS						
Silhouette	0.090205585	0.14792994	-0.026698655	0.066749882		
Davies Bouldin	1.880259645	1.325061942	1.856025741	1.485654507		
Calinski Harabasz	415.0217203	398.0623673	270.0118628	333.4797888		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	0.014576539	0.027108888	-0.022788496	0.045531614	0.014300062	0.042631686
Davies Bouldin	1.766654312	1.419464221	1.854746112	1.483527014	1.664366317	1.679477787
Calinski Harabasz	564.0314074	896.8407168	586.2012859	612.1252895	704.3764949	679.0462408
OPTICS						
Silhouette	0.061517395	0.078068532	0.076688728	0.073472896	0.10369128	0.088161405
Davies Bouldin	1.918175578	1.751581717	1.792453856	1.561911905	1.453113206	1.767883367
Calinski Harabasz	704.0061094	554.7153808	428.8705611	377.6480269	456.7150788	386.9546435
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 79: Evaluation scores comparison averaged from figures 77 and 78

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.045056932	-0.043946639	-0.068129525	0.034058403		
Davies Bouldin	1.974228023	1.623043713	1.561681712	1.330793149		
Calinski Harabasz	559.9454484	231.1185974	314.9127009	518.1632965		
OPTICS						
Silhouette	0.087323241	0.078877471	-0.044939384	0.050577533		
Davies Bouldin	1.647662855	1.714452117	1.548555372	1.354076657		
Calinski Harabasz	455.8046908	306.9855407	268.0225278	378.5922696		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	-0.03370953	-0.028693724	-0.014684342	0.035463296	0.018177651	0.024473445
Davies Bouldin	1.843484805	1.914727542	1.881877562	1.578954586	1.668429671	1.776349426
Calinski Harabasz	627.1549248	725.4648494	627.0653245	756.8045021	680.7680095	692.6218533
OPTICS						
Silhouette	0.011725605	0.057709865	0.112025782	0.111947685	0.081173912	0.076792531
Davies Bouldin	1.871782347	1.728019021	1.670963152	1.39067254	1.537877073	1.928194677
Calinski Harabasz	480.0815334	567.1386341	445.5336128	501.6112069	454.7315535	437.6176714
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 80: Evaluation scores comparison averaged from 2 runs of t-SNE and clustering with a learning rate of 20.

1h Files						
	15 min	30 min	45 min	1h		
DBSCAN (TSNE)						
Silhouette	0.062392585	0.109573543	-0.004795788	-0.068260521		
Davies Bouldin	2.309679877	1.725495953	1.782488054	1.608990382		
Calinski Harabasz	440.260006	540.8427817	380.9053749	281.6679029		
OPTICS						
Silhouette	0.057782359	0.135444269	0.017859722	0.036272764		
Davies Bouldin	2.004440686	1.750887229	2.079905994	1.546211321		
Calinski Harabasz	324.1682685	336.6839227	267.442482	199.8783014		
3h Files						
	30 min	1h	1h 30 min	2h	2h 30 min	3h
DBSCAN (TSNE)						
Silhouette	0.054977857	0.136667132	0.125094607	0.172261804	0.142229527	0.131898537
Davies Bouldin	1.497732311	1.409259524	1.403925898	1.50888884	1.439275292	1.368191426
Calinski Harabasz	348.4941339	296.3657299	275.6990944	207.365258	190.8214068	197.7513201
OPTICS						
Silhouette	0.085169926	0.095759317	0.127076074	0.080362737	0.047243498	0.084326543
Davies Bouldin	1.532057234	1.468021265	1.400929179	1.518005655	1.37525721	1.264660171
Calinski Harabasz	144.7781324	104.1385105	102.4009152	83.40832451	65.84819665	69.71735313
	best values in files (1h or 3h)					
	best values total (1h and 3h)					

Figure 81: Evaluation scores comparison averaged from 2 runs of t-SNE and clustering with a learning rate of 800.

B git-Repository

todo: list contents of git repo

Das Repository dient zur Dokumentation und Nachvollziehbarkeit der Arbeitsschritte. Stellen Sie sicher, dass der/die BetreuerIn Zugriff auf das Repository hat. Stellen im Sinne des Datenschutzes sicher, dass das Repository nicht für andere zugänglich ist.

Verpflichtende Daten für Bachelorarbeit 1 und 2:

- LaTeX-Code der finalen Version der Arbeit
- alle Publikationen, die als pdf verfügbar sind.
- alle Webseiten als pdf

Verpflichtende Daten für Bachelorarbeit 2:

- Quellcode für praktischen Teil
- Vorlagen für Studienmaterial (Fragebögen, Einverständniserklärung, ...)
- eingescanntes, ausgefülltes Studienmaterial (Fragebögen, Einverständniserklärung, ...)
- Rohdaten und aufbereitete Daten der Evaluierungen (Log-Daten, Tabellen, Graphen, Scripts, ...)

Link zum Repository auf dem MMT-git-Server `gitlab.mediacube.at`:

`https://gitlab.mediacube.at/fhs41216/BacThesis`

C Archivierte Webseiten

`http://web.archive.org/web/20160526143921/http://www.gamedev.net/page/resources/_/technical/game-programming/understanding-component-entity-systems-r3013`, letzter Zugriff 1.1.2016

`http://web.archive.org/web/20160526144551/http://scottbilas.com/files/2002/gdc_san_jose/game_objects_slides_with_notes.pdf`, letzter Zugriff 1.1.2016