FH Salzburg

MultiMediaTechnology

Identifying the Ideal Length of Time to Aggregate 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

SmartEater ist eine bevorstehende mHealth-Smartphone-App (Mobile Health) mit dem Ziel, dem Benutzer inhaltsabhängiges Feedback zu geben, um eine Episode mit Verlangen nach Essen abzuwenden. Mithilfe von Machine Learning und Smartphone Sensor- und Nutzungsdaten können Stress und bevorstehendes Verlangen vorhergesagt werden. Das Ziel dieser Arbeit ist es, Unsupervised Machine Learning zu verwenden, um das beste Zeit Delta zu finden, um aus diesen Daten qualitativ hochwertige Cluster zu erstellen. Die aufgezeichneten Smartphone Sensor- und Nutzungsdaten umfassen den Beschleunigungsmesser, die Lautstärke des Audios, den Prozentsatz der Einschaltdauer des Bildschirms, die Anzahl der Benachrichtigungen, die Lichtsensorwerte und die App-Nutzung in den Kategorien „Kommunikation", „Videoplayer" und „Sonstige". Die von Testpersonen aufgezeichneten Daten werden in zwei Datensätzen (1 Stunde und 3 Stunden) mit jeweils mehreren Zeitlängen aggregiert. Nach dem Bereinigen und Normalisieren dieser Datensätze, wird die Anzahl der Dimensionen, mithilfe von t-SNE, von acht auf zwei reduziert. Geeignete t-SNE-Parameter werden abgestimmt, um die bestmögliche t-SNE-Visualisierung zu erzielen. Die dichtebasierten Clustering-Methoden DBSCAN und OP-TICS werden verwendet, um Datenpunkte für jede der insgesamt zehn Zeitlängen in Cluster zu gruppieren. Die Ergebnisse werden unter Verwendung der mathematischen Bewertungsmaße Silhouette Score, Davies-Bouldin Index und Caliński-Harabasz Index verglichen. Die Ergebnisse zeigen, dass die Zeit Deltas 30 Minuten (aus dem 1-Stunden-Datensatz), 1 Stunde und 2 Stunden (beide aus dem 3-Stunden-Datensatz) die deutlichsten und klarsten Cluster bilden.

Abstract

SmartEater is an upcoming mHealth (mobile health) smartphone app, with the goal to provide the user with content-dependent feedback, to avert a food craving episode. Using machine learning and smartphone sensor and usage data, stress and upcoming cravings can be predicted. The goal of this thesis is to use unsupervised machine learning to find the time delta to construct high quality clusterings from this data. The smartphone sensor and usage data recorded, includes accelerometer, volume of the audio, percentage of screen-on-time, number of notifications, light sensor values, and app usages in the categories "communication", "video players", and "other". The data recorded from test subjects is aggregated into two datasets (1 hour and 3 hours), each with multiple time lengths. After cleaning and normalising these datasets, the number of dimensions is reduced from eight to two using t-SNE. Suitable t-SNE parameters are tuned to create the best possible t-SNE visualisation. The density-based clustering methods DBSCAN and OPTICS are used to group data points into clusters for each of the total ten time lengths. These results are compared using the mathematical evaluation scores Silhouette Score, Davies-Bouldin Index, and Caliński-Harabasz Index. The results indicate, that the time deltas 30 minutes (from the 1h dataset), 1 hour, and 2 hours (both from the 3h dataset) create the most distinct and well-defined clusters.

Contents

List of Figures

Abbreviations

PCA	Principal Components Analysis
SNE	Stochastic Neighbor Embedding
t-SNE	t-Distributed Stochastic Neighbor Embedding
DBSCAN	Density Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points To Identify the Clustering Structure

1   Introduction

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 and usage 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

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.

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)

---

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

- Grid-based methods (examples: STING, CLIQUE)

According to Han, Pei, and Kamber (2011)[414], the above-mentioned clustering methods work well with datasets that are not high-dimensional and have less than 10 attributes. Since the SmartEater dataset only has 8 unique dimensions (columns), it is not considered high-dimensional. This paper will therefore be able utilise and compare the results of these types of clustering methods.

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 dataset 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. 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 smartphone sensor and usage 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 was 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. investigated how people were currently tracking their meals by conducting a survey. Under half of the participants had previously used food tracking/journalling tools. 48.9%

---

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

of these stopped using them within the first month. Respondents also wished for the app to take action directly before a meal/snack ("About-to-Eat" moments), thus supporting the authors' previous assumptions.

The authors used a variety of different sensors to record the following data: physical movement (raw accelerometer, gyroscope, step count, speed), caloric expenditure, heart rate, skin temperature, electrodermal activity (good indicator for psychological arousal), chewing and swallowing sounds (for detection of current eating events) and GPS location. Furthermore, the test subjects recorded self-reports, such as when they started eating, emotional state when eating, intensity of desire/craving and hunger, and end of eating. Eight participants, aged 26-54, participated in this study for five days. The recorded data then underwent cleaning and preprocessing, feature extraction, feature selection and machine learning. In the feature extraction step, two parameters were used on the processed sensor time series to create windows. Firstly, the feature extraction window size parameter regulated 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, established 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.

In the feature selection step, the most relevant features were selected (e.g. the location features were not beneficial and merely represented noise). Signal processing and machine learning were used on the sensor streams to train an "About-to-Eat" moment classifier. The 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.

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. Collected data included: activity, app usage, network traffic, reboot activity (power on and off events), calls (min, max and mean dB-values collected by the microphone), environment brightness, timestamps of received messages, noise exposure, and screen activity. Fifteen 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 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
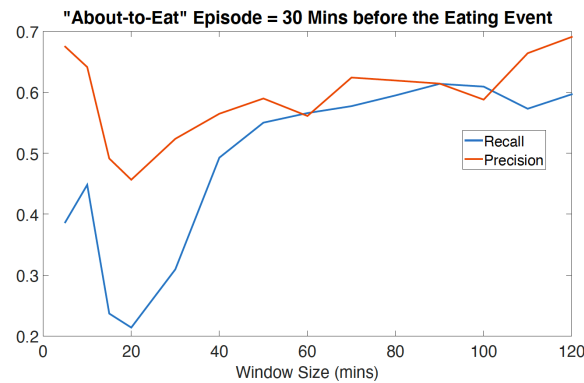
Figure 1: This graph depicts the F measure with the different feature extraction window sizes of the "About-to-Eat" classifier (Rahman et al. 2016)[148]. According to Han, Pei, and Kamber (2011)[306-307], precision and recall are measures used in classification. Precision describes the exactness (how many of the as positive selected items are positive), recall defines the completeness (how many positive items have been correctly selected). The F-score is a measure that combines precision and recall (harmonic mean).

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 is easier to spot longer periods of high stress than shorter ones. This paper is 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. Sixty-five 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 Methods

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. 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, in supervised methods, there is a predefined target variable and the model learns 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 used in this thesis and further detailed in section 3.4.

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 datasets first need to undergo a data preprocessing step. 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] states, that 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, 45] 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*). The authors also report, that ID fields should be removed from the dataset 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. Further information preparation steps are outlined in the next sections below.

3.2.1 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 dataset. An inadequately chosen replacement value could distort the dataset, 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.

3.2.2 Normalisation

Han, Pei, and Kamber (2011)[105-106] describe normalisation as giving the attributes of a dataset 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. People's income would, for instance, have a larger range than binary attributes.

García, Luengo, and Herrera (2015)[46-48] review the following normalisation methods: *Min-Max Normalization*, *Z-score Normalization*, and *Decimal Scaling Normalization*. For the following examples, A is a numerical attribute from a dataset, 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.

$$v' = \frac{v - \bar{A}}{\sigma_A}$$

  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). 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 dataset, whilst mostly keeping the integrity of the initial dataset. Principal components analysis and t-SNE are dimensionality reduction techniques used to reduce the SmartEater dataset.

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. Jolliffe (2002)[7] clarifies, that while these two papers used different methods, the standard algebraic derivation was announced by Hotelling (1933). According to Han, Pei, and Kamber (2011)[95-96], the general idea is to calculate 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. Hotelling (1933)[4, 5, 15, 18] explains, that the components are chosen with decreasing amount of variance. Therefore, the one with the highest variance is chosen first. The next highest is chosen orthogonal to the one before, and so on. Han, Pei, and Kamber (2011)[93, 95-96] mentions, in data mining, the vectors with the lowest variance are removed, thus reducing the number of dimensions. Despite the loss of data, the components with higher variance can approximate the original data.

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

In their paper, Maaten and Hinton (2008)[2579-2580] introduce t-SNE, a method of dimensionality reduction and visualising high dimensional data on a two or three-dimensional plot. The authors mention, that while linear methods like PCA concentrate on making sure low-dimensional representations of data points are 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], such a probability $p_{i,j}$ would arise from the similarity between the two data points 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 $p_{i,j}$ 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 dataset, Maaten and Hinton (2008)[2582] describe, the σ_i 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 an overall optimal σ_i for the dataset. 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 neighbours 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. Another parameter to be set is the learning rate. Han, Pei, and Kamber (2011)[332] clarify, that the learning rate parameter is used to support the finding of global minimums, instead of being caught in a local minimum. A learning rate set too low will result in slow learning. In contrast, a learning rate set too high could lead to poor results.

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. Romesburg (2004)[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. Real-world examples of clustering include classifications of animals, plants and diseases.

Han, Pei, and Kamber (2011)[361-363] state, that the objects placed into one cluster are dissimilar to the objects assigned to other clusters. The fact that cluster analysis can find groups by itself, gives it its unique advantage. Clustering is a type of unsupervised machine learning, 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 to preprocess data 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.

3.5 Overview of clustering algorithms

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

- Scalability

- Ability to work with different attribute types

- Recognising clusters with arbitrary shapes

- Requirements for domain knowledge (for parameter selection)

- Ability to handle noise

- Incremental clustering (integrate updates without recomputation)

- Insensitivity to the order of the input

- Ability to cluster high-dimensional data

- Capability to cluster under certain constraints

- Interpretability and usability of the results

Han, Pei, and Kamber (2011)[366-368, 373, 374, 385] 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.

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 $k \geq n$. A data object can only be classified into one group (*exclusive cluster separation*), and every group has to have a minimum of one object. 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 in 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 (Han, Pei, and Kamber 2011)[366, 368].

3.5.2   Hierarchical Methods

The data is grouped into a hierarchy ("tree") of clusters. 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 condition for termination 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 (Han, Pei, and Kamber 2011)[366, 367, 373, 374].

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. The following two density-based clustering algorithms were used in the experiment: DBSCAN and OPTICS.

3.5.3.1   DBSCAN

Ester et al. (1996)[226-229] introduce a density-based clustering algorithm, the Density Based Spatial Clustering of Applications with Noise. This method is able to find clusters with different shapes and work efficiently on large spatial datasets. The algorithm searches in a given radius (Eps = epsilon parameter) around a data point. If within this radius a minimum number of points (MinPts parameter) exists, then this point is added to the cluster (core point). A data point (p) is considered a border point, if inside of its 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 ideal parameters for the "thinnest" cluster. The first step is to construct a sorted k-dist graph. For a specific k, the distance d of every point p to its kth nearest neighbour is calculated. These distances are then sorted in descending order and depicted on a graph. Such a graph can be seen in figure 2. 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 2, is this threshold point. The points to the left of it with higher distances are likely to be noise and the points to the right 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.
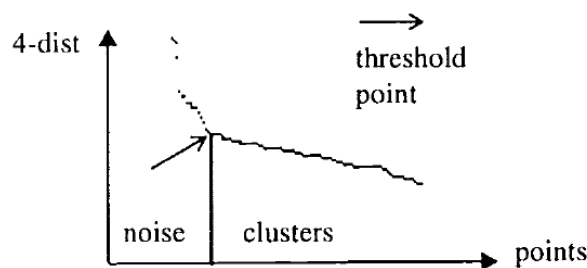
Figure 2: Sorted 4-dist graph (distance for each point to its fourth nearest neighbour) to calculate DBSCAN Eps parameter. The first point in the "valley" is supposedly the ideal value. The points to the left with higher distances are likely to be noise and the points to the right are part of clusters (Ester et al. 1996)[230].

3.5.3.2   OPTICS

Han, Pei, and Kamber (2011)[388] mention, that OPTICS was created to improve the selection of global parameters problem in DBSCAN. Ankerst et al. (1999)[49, 51-54, 57, 60] 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 dataset 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 3.
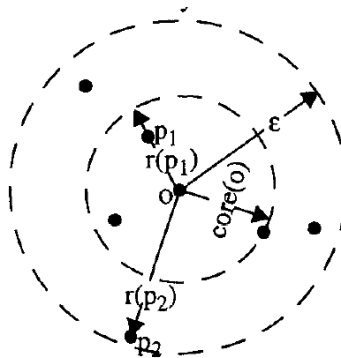
Figure 3: Core-distance of object o (MinPts = 4) and reachability-distances for objects *p1* and *p2* (Ankerst et al. 1999)[52].

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 4 a reachability plot calculated from a 2D dataset is depicted.
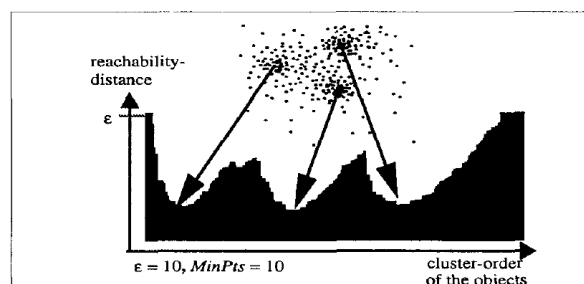
Figure 4: OPTICS reachability plot for a 2D dataset. Three "Gaussian bumps" can be seen (Ankerst et al. 1999)[54].

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
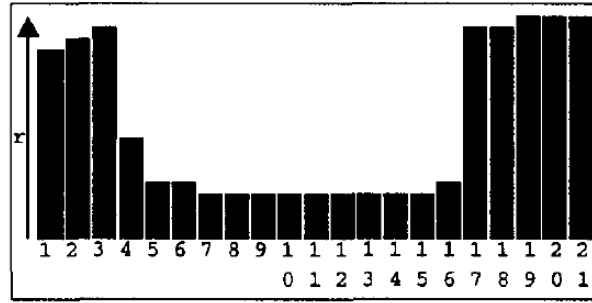
Figure 5: Since clusters are defined as the dips/dents in the reachability plots created by the OPTICS algorithm, a cluster can be extracted from this plot starting at the 3rd data point and ending with the 16th. The x-axis of the reachability plot depicts the order of the data points (objects) and the y-axis the reachability-distance for each datapoint (Ankerst et al. 1999)[57].

(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 4). In figure 5, a cluster could hence be extracted from this plot starting at the 3rd data point and ending with the 16th.

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 executed on the grid structure. The grid size (number of cells) determines the processing time. It is faster than other clustering methods. Examples: STING, CLIQUE (Han, Pei, and Kamber 2011)[367, 392].

Han, Pei, and Kamber (2011)[414, 416] clarify, that the clustering methods mentioned above have a good functionality when used on a dataset 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, 398] describe this stage as assessing the quality of the results. There are different steps to be taken in evaluating clusters, e.g. evaluation of the cluster quality, assessment of the cluster tendency (whether non random structures exist), and establishing the number of clusters (e.g. for k-means clustering). The experiment described in this thesis used mathematical evaluation methods to compare cluster quality among the different time lengths.

Han, Pei, and Kamber (2011)[399, 401] point out, 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 intrinsic methods, since there is no ground truth for comparison.

3.6.0.1   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 clusters created by different algorithms, the input data being the same.

The formula is as follows: With the use of silhouettes, the author's goal is to find out, if the quality of the clusters is high. Hence, the dissimilarities of the objects within a cluster are small, and the dissimilarities are large compared to the objects in other clusters. For each object *i* (in cluster *A*) the value *s(i)* is calculated. The variable *a(i)* contains the average dissimilarity of 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). The variable *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 may be seen as the next best choice for *i*. *b(i)* can only be calculated, if there are other clusters beside *A*. The formula for *s(i)* 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 ≤ s(i) ≤ 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 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 uncertain whether cluster *A* or its neighbour is a more suitable fit. If *s(i)* is near -1, then the dissimilarity within a cluster is larger than the dissimilarity to the next closest cluster. Thus, it would have been more natural to assign *i* to the neighbouring cluster, since it is closer to it (misclassified).

The *average silhouette width* can be calculated for single clusters. It is established by calculating the average of all objects that belong to said cluster. An average score can also be calculated from each object *i* for the whole chart (dataset), the so called *overall average silhouette width*.

3.6.0.2   Davies-Bouldin Index

A second cluster evaluation method is the Davies-Bouldin Index, which was proposed 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.

3.6.0.3   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 follows, where *n* is the number of points, *k* is the number of clusters, *WGSS* is the within-group (cluster) sum of squares, and *BGSS* is the between-group (cluster) sum of squares.

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

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. The more natural the clusterings are, the higher VRC will be, since the variation within the cluster is lower.

4   Results

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

---

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

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.

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, app usage in percent of lag-interval minutes):

- TIME: timestamp, when the data was aggregated (format: YYYY-DD-MM hh:mm:ss)

- ACC (1-N): values received from the accelerometer (average jerk). According to the Android developers documentation[6], the accelerometer (acceleration sensor) records the acceleration (including the force of gravity) enforced onto the smartphone.

- AUDIO (1-N): volume of the audio

- SCRN (1-N): percentage of screen-on-time

- NOTIF (1-N): number of notifications

- LIGHT (1-N): (environment) light sensor values

- APP_COM (1-N): app usage in the category *communication*

- 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[7]) 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).

---

6. https://developer.android.com/guide/topics/sensors/sensors_motion
7. https://tools.ietf.org/html/rfc4180

Python was used to conduct the experiment, more specifically using the Anaconda[8] Python distribution platform for data science. The scikit-learn[9] (short sklearn) Python package provides simple tools for predictive data analysis and was used for data preparation, dimensionality reduction, and clustering.

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[10] tool. This programming 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*[11] and *concat*[12] 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 3.2.1 mentions substitution of missing data, thus preserving rows, 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*[13] 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 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 3.2.2, values with higher ranges can inadvertently outweigh smaller values. To be sure that the values are weighted the same, the sklearn *StandardScaler*[14](Z-score Normalization) was used to normalize the data. As mentioned in section 3.2.2, Z-score normalization is more robust to outliers, that could

---

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

otherwise bias Min-Max Normalization. Therefore, Z-score Normalization was used instead of Min-Max Normalization.

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 did not contain any predictive content. The TIME column was removed for this reason. Since the TIME column only showed the time and date 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 that had recorded the same feature but at different time lags (e.g. ACC1-N), were compressed to one column. Each unique feature only required one column, and therefore reduced the number of columns to only 8 (instead of 32 in the 1h dataset or 48 in the 3h dataset).

4.1.4   Chain shaped data

Initial visualisations of the dataset (with t-SNE dimensionality reduction) showed a chain formalisation of the data across various t-SNE parameters. This chain can be seen in figure 6 (light blue data points, almost swirl shaped). A similar accumulation of data points, though this time shaped as a line, can further also be seen in the PCA dimensionality reduced dataset (see figure 7).
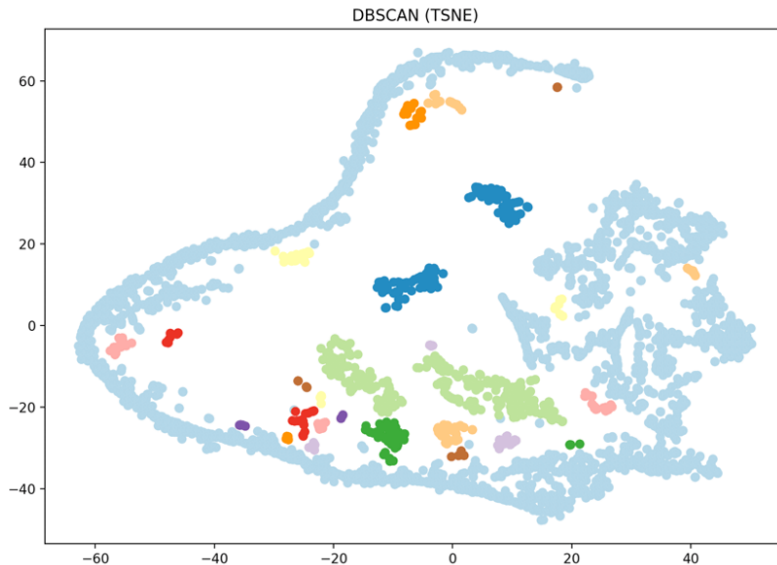
Figure 6: 1h dataset, a chain of data points (light blue points) can be seen in the dataset.
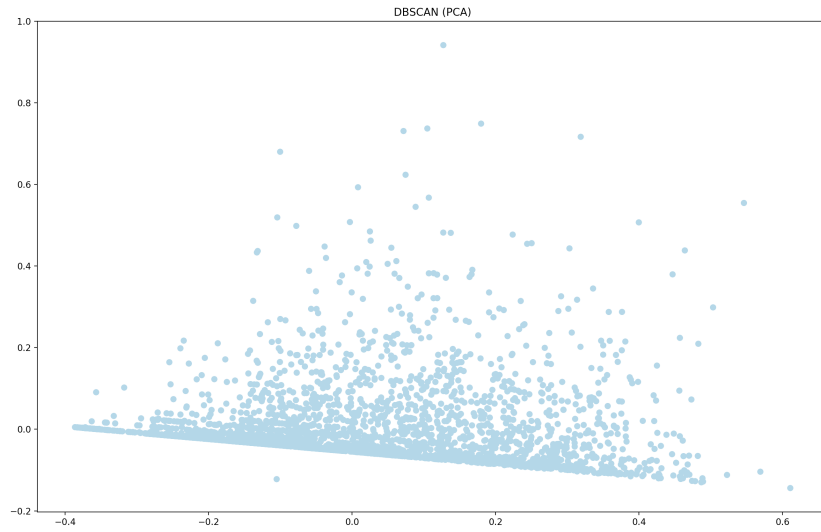
Figure 7: There is a collection of points along a line. This same collection can also be visualised as a chain in the t-SNE dataset in figure 6.

An initial thought, was that the chain was a consequence of dependencies within the data or columns. For example, the three APP columns indicate the percentage of time in a lag-interval a certain 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 dataset and the 2D scatter plot was recalculated. The chain however 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 also became less dense and less visible when reducing the number of inserted .csv files (lower number of records). This supported the notion, that the rows could be the cause.

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 8. 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. It was evident that multiple rows 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 dataset (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 was also often 0 when SCRN was 0, maybe because it was 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 9, the chain is 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 average number of rows left for the 1h data files, depending on the time length, was 1654 from initial 8283 (19.97%). In the 3h dataset, only an average of 3976.5 from 14091 rows remained (28.22%).
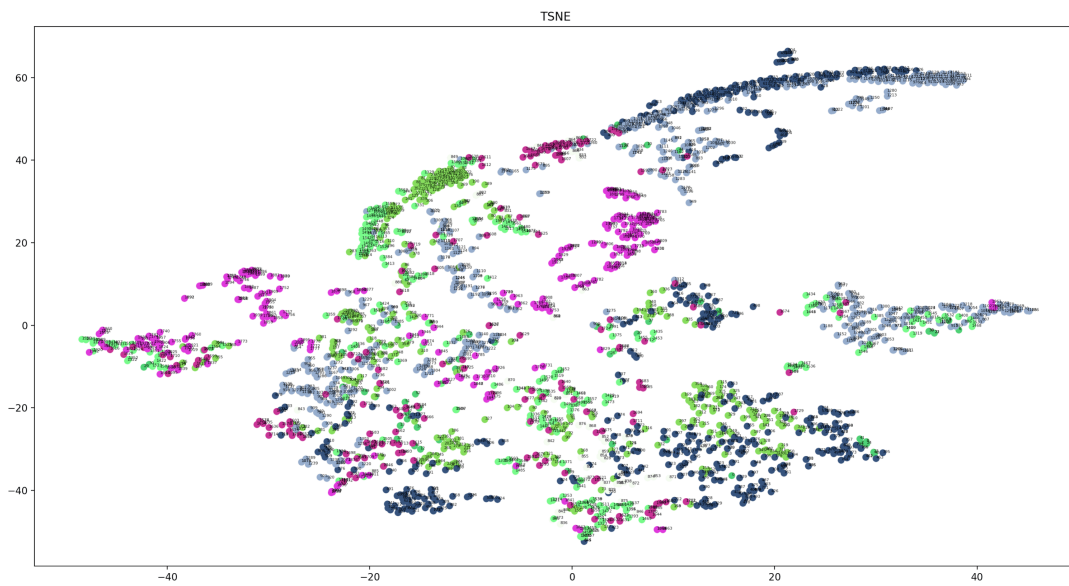
Figure 8: t-SNE result in a scatter plot, each colour indicates one test subject. The test subjects have clustered together slightly in the chain.

4.2 Dimensionality Reduction

The dimensionality reduction methods PCA and t-SNE were used to reduce the number of dimensions (number of attributes, so in this case number of columns).

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.

---

15. `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`
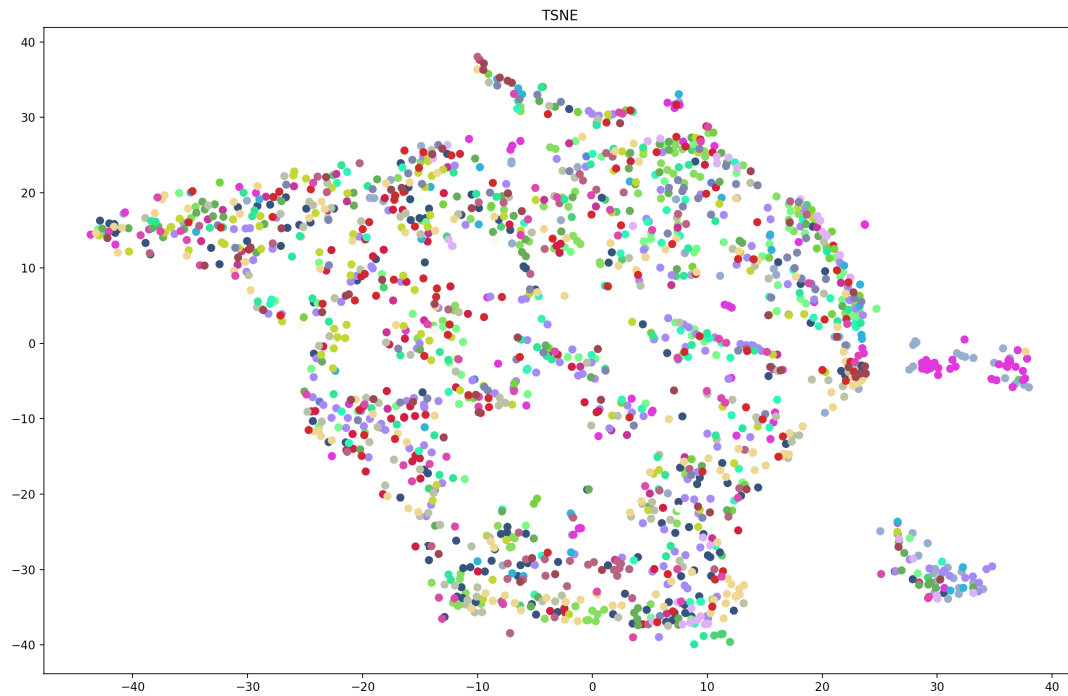
Figure 9: t-SNE calculated after the removal of rows with less than 50% of values that weren't 0. The chain is less visible and less dominant.

Before the removal of the chain (section 4.1.4), the PCA allowed 65%-95% (depending on data preparation type) of the data's important structures to be accounted for in only the first two or three principle components. After the removal of the chain, the first three components only contained almost 50% of the important structures. Of the 8 components, 7 would have been needed to account for over 90% of these formations. As figure 10a shows, the resulting data from these components did not show any significant clusters, in comparison to the t-SNE results. The principle components were further depicted in a 3D scatter plot, to evaluate whether the extra dimension revealed more structure. However, as can be seen in 10b, the results were almost identical. It is considered, that as PCA is a linear dimensionality reduction method, it might not be suitable to transform the SmartEater dataset.
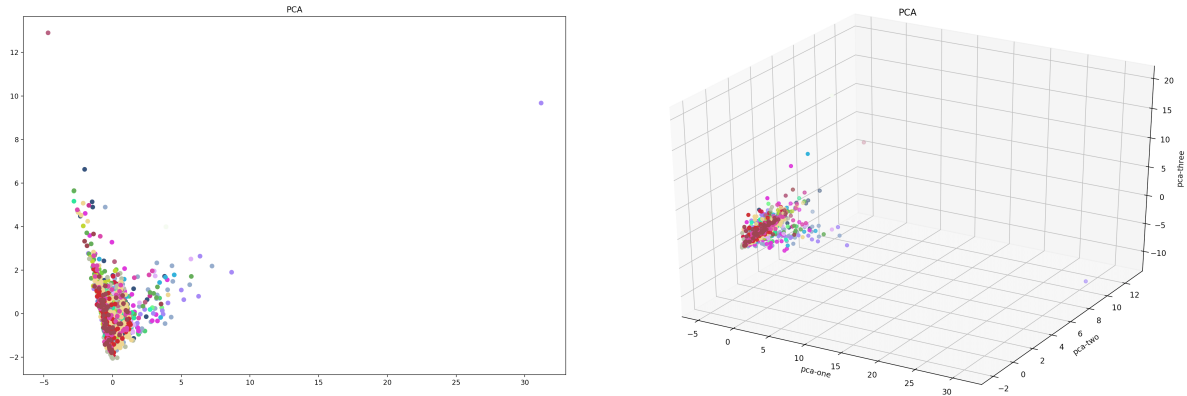
Figure 10: 2D (a) and 3D (b) scatter plots of the PCA results. The data points only appear to form one cluster with little visible structure.

4.2.2 t-SNE

The t-SNE results proved to be more significant than the PCA ones. The t-SNE algorithm was implemented using the sklearn *t-SNE*[16] implementation. To effectively apply the t-SNE algorithm, it was important to identify the parameters (perplexity, learning, and number of iterations) most suitable to the SmartEater dataset, 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. Wattenberg, Viégas, and Johnson (2016) also used 5000 iterations in their experiments and stated 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. The first parameter to determine was 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, in steps of 10 (except for the first step which was 5) were tested. A perplexity of 45 was further added, since it had shown good results in tests performed before the chain described in section 4.1.4 was removed. The resulting scatter plots are listed in the appendix A.1. In the appendix, the plots on the left side 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 were 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

---

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 11: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **perplexities** in steps of 5 and 10. The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).

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 assist in selecting the perplexity that creates the best clusters, the three mathematical evaluation scores, mentioned in section 3.6, were compared. Figure 11 displays the Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index, 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). The number of these best scoring fields will henceforth be referred to as number of "wins". 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 were highest, whose perplexity was between 30 and 50. Perplexity 40 had the most wins in figure 12 and was chosen as the final parameter. These tests were conducted a second time (see appendix **??**), 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 best achieving value.

The same principle was used to determine the ideal value for the learning rate parameter. The

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 12: Comparison of the evaluation scores of the top three perplexity candidates 40, 45, and 50. The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).

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 13 revealed, that the learning rate 10 scores highest with a learning rate of 800 coming in second. Since roughly 200 steps were taken between the learning rate values, smaller steps were taken (i.e. 50) around the winning value to see if an even better performance could be accomplished (figure **??**, in appendix **??**). 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 **??**, in appendix **??**). 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 14 exposed, 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 **??**. In general, the results are similar. 800 appears to be more dominant than the lower parameter values (e.g. 10 and the favourite 20). To further compare 20 and 800, appendix **??** contains a table (figure **??**) 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 **??** and **??** 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

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 13: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values, in (except the first step of 190). The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).

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 OP-TICS 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 15 and 16.

4.3   Clustering

The clustering algorithms used for the SmartEater dataset, 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: 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 14: Comparison of Silhouette Coefficient, Davies-Bouldin Index, and Caliński-Harabasz Index for different t-SNE **learning rate** values. The goal is to find a perplexity value between 10 and 30, that satisfies both the 1h and 3h datasets. The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).
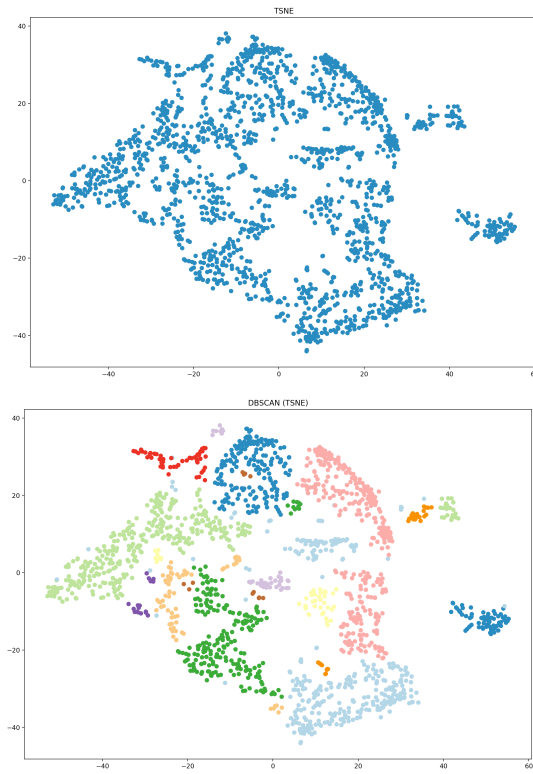
Figure 15: Final t-SNE parameters (1h data files): perplexity=40, n_iter=5000, learning_rate=20
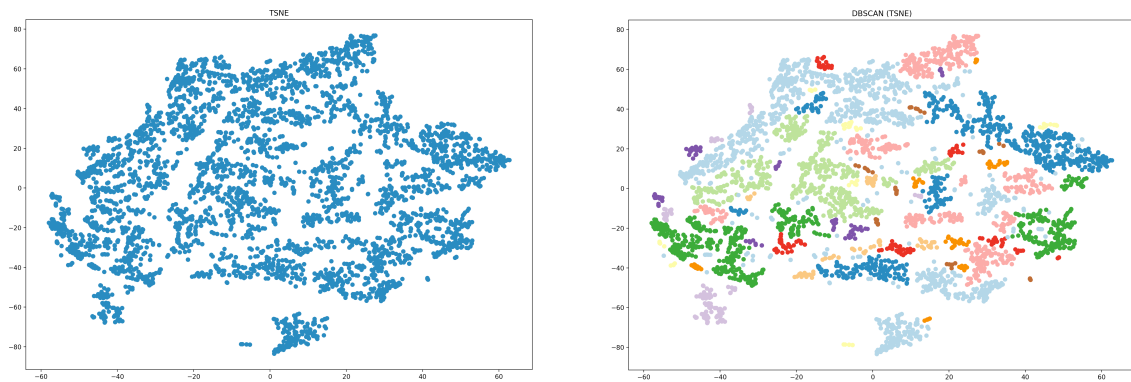
Figure 16: Final t-SNE parameters (3h data files): perplexity=40, n_iter=5000, learning_rate=20

4.3.1 DBSCAN

The DBSCAN algorithm was applied on the SmartEater dataset 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 datasets. The graphs contained the distances kth 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 was to select *Eps* suitable for the "thinnest" cluster, however being careful to avoid noise. As can be seen in figures 17a and 17b, the valley starts at roughly a 4th nearest neighbor distance of 2.

(a) 1h dataset

(b) 3h dataset
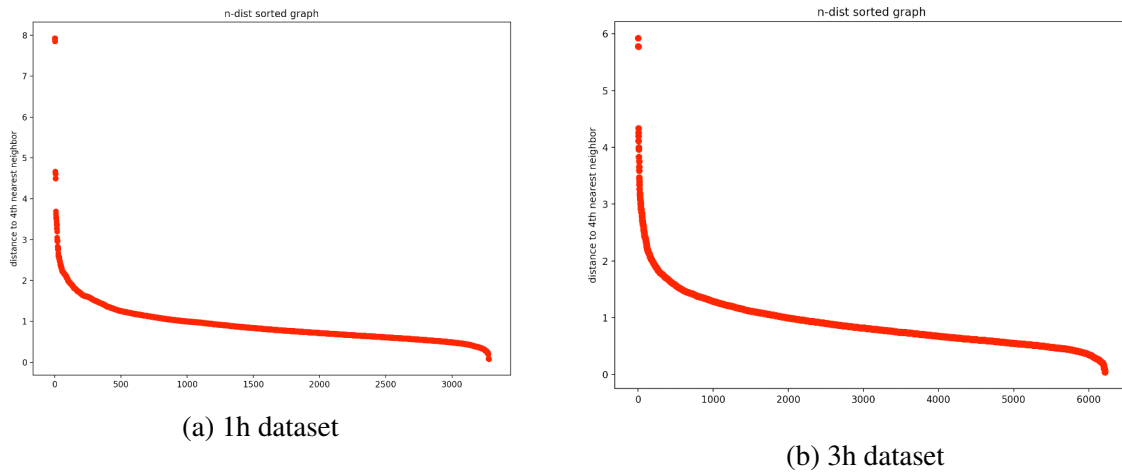
Figure 17: Sorted 4-dist graphs (distance for each point to its fourth nearest neighbor), used to determine a suitable Eps parameter for the DBSCAN algorithm. The valley starts at roughly the 4th nearest neighbor distance of 2, therefore Eps should be 2.

The DBSCAN algorithm was applied, with the parameters eps = 2 and min_samples (MinPts) = 4. The results of this clustering method can be seen in figure 18.

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

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

(b) 3h dataset DBSCAN clustering (first column - 30 min).
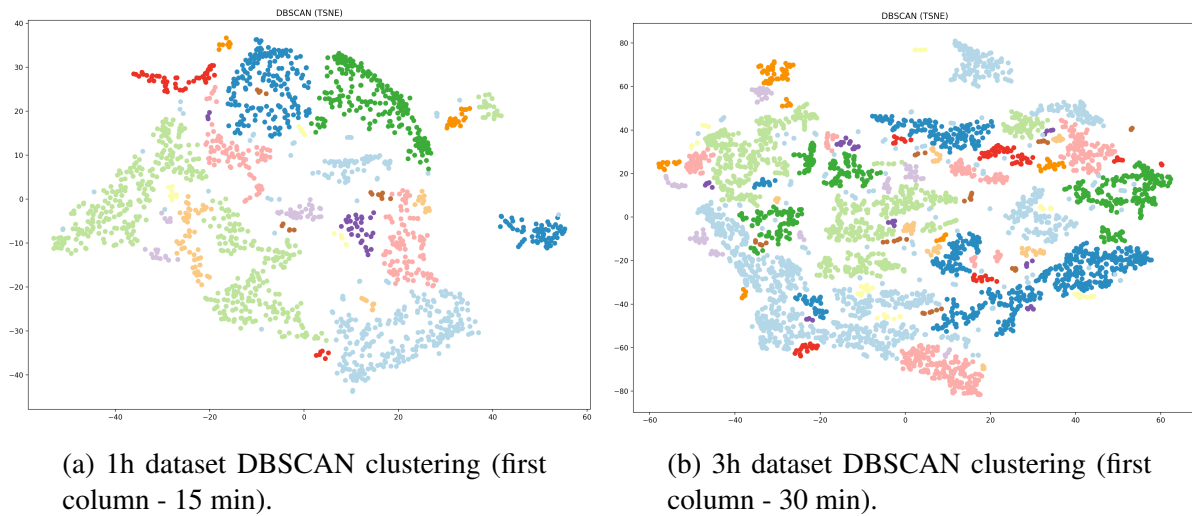
Figure 18

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 19. 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. The resulting clusters are depicted in figure 21. The corresponding reachability plots are illustrated in figure 22.

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

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

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

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

Figure 19

(a) 1h dataset OPTICS reachability plot (first column - 15 min), using OPTICS automatic cluster extraction (xi).

(b) 3h dataset OPTICS reachability plot (first column - 30 min), using OPTICS automatic cluster extraction (xi).

Figure 20

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

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

Figure 21

(a) 1h dataset OPTICS reachability plot (first column - 15 min), using DBSCAN clustering.

(b) 3h dataset OPTICS reachability plot (first column - 30 min), using DBSCAN clustering.

Figure 22

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, the same way as in section 4.2.2. As explained in section section 3.6, the Silhouette Score indicates better, denser clustering, when it is closer to 1, and incorrect clustering if the value is close to -1. The lower of two Davies-Bouldin Index and the higher of the Caliński-Harabasz Index scores suggests better clustering.

These three scores were calculated and stored for each DataFrame, after each clustering algorithm was applied (DBSCAN and OPTICS) for each time length. The resulting values where then compared and detailed in this section. 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 **??** and **??**, two individual t-SNE and clusterings were run, figure **??** depicts the average scores of these two. Figure **??** depicts an average of a different two runs. These mentioned runs all held the learning rate parameter 20. Since a learning rate of 800 also proved to be a viable choice for this parameter, the t-SNE and clustering was also run twice and averaged with a learning rate of 800 (figure **??**). The mean was taken from all the above results, thus creating figure 23 (this 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 in section 4.2.2, when comparing the t-SNE results, the light green fields 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 wins per dataset (1h or 3h) for the 1h data files are pictured in figure 24a, in figure 24b for the 3 hour data files, and in figure 25 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 extent of how far this time length was better. For this reason, the average scores from figure 23 for the five winning time lengths were compared solely to each other. The results are detailed in figure 26. 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).

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 23: Evaluation scores comparison averaged from figures **??**, **??**, **??**, **??**, and **??**. The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).
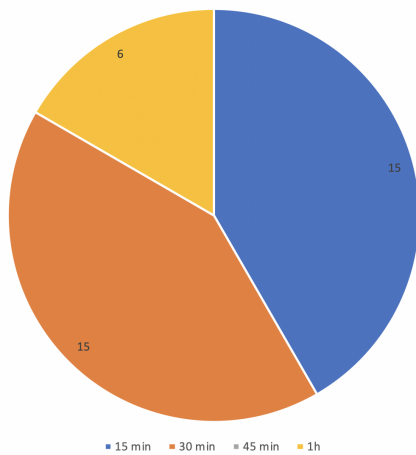
(a) Number of evaluation score wins (1h or 3h dataset) for the 1h data files.
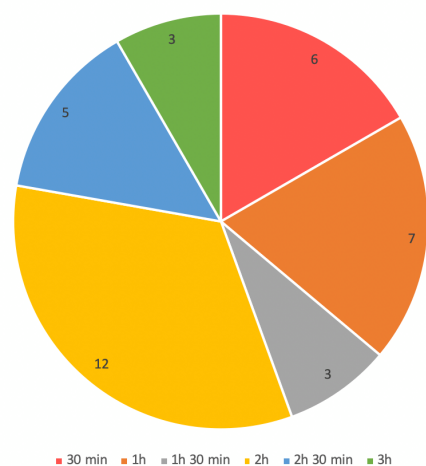
(b) Number of evaluation score wins (1h or 3h dataset) for the 3h data files.

Figure 24: The lighter green highlighted values indicate the best values of that file aggregation (1h or 3h files). The dark green highlighted values illustrate the overall best values over all files (1h and 3h files).

Total Number of evalutation score wins (1h and 3h)
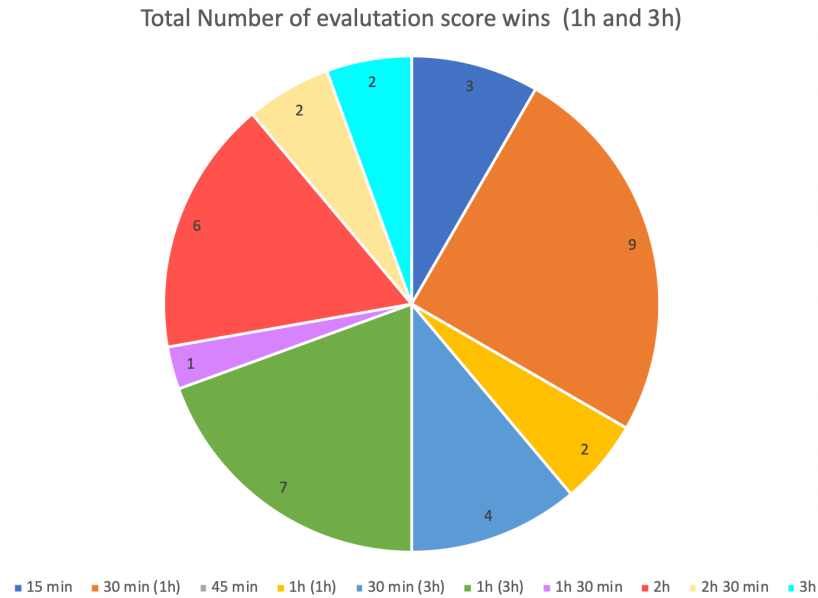
Figure 25: Number of dark green evaluation score wins (1h and 3h dataset).

	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 26: Comparison of top evaluation score performers (of clusterings) from figures 24a, 24b, and 25. The light green highlighted cells indicate the best value across the different time lengths for that score.

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 27: Evaluation scores comparison from averaged 1h and 3h dataset runs of t-SNE and clustering.

5   Discussion

Section 4.4 reveals, that the 2h time length (from the 3h dataset) 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 dataset) came second, and the 30 min time delta (from the 1h dataset) 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 dataset) however came top, followed by the 1h time length (from the 3h dataset) and then the 2h time length (see figure 25). It can therefore be concluded, that these 3 time deltas are the best performers.

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.

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 27. Both the 1h and 3h datasets have the same amount of wins. When scanning other comparisons of these files (also the ones utilised for finding the t-SNE parameters), it is noticeable, that the 1h and 3h datasets either have the same number of wins, or the 3h dataset has more. This indicates that overall, the three hour aggregation files produce more superior clusters.

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 dataset (61.1%). A possible explanation, as to why the 3h dataset might have created better clusters, was that the dataset 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 3.6.0.1, 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 Intervention, the SmartEater app would need to predict upcoming stress. There are different types of stress. The Canadian Centre for Studies on Human Stress (CSHS) [19] differentiate acute and chronic stress. Acute stress is caused by a particular event or setting, when something is new and feels out of control. For example like a presentation in front of people. Chronic stress is longer term and caused by repeated situations that cause stress. The data points would need to account for and recognise these different types 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 also be a 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 datasets appear to be denser, whilst the 1h datasets have more wider non-populated regions between clusters. One of the 1h dataset top performers, 30 minutes, seems to be scattered in a slightly different shape to the other scatter plots of the 1h dataset. It appears to be spread out wider along the x axis, similar to the scatter plots from the 3h data files. The 30 minute time length achieved the highest number of wins in the 1h dataset, along with the 15 minutes time length. 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 beneficial to get more data from users for a longer period of time. However, it would also be necessary to confirm that the clusters were created for stress levels and not for each user (since a user's behaviour could exhibit similarities). As implemented

---

19. https://humanstress.ca/stress/understand-your-stress/acute-vs-chronic-stress/

in this thesis, this could be achieved by highlighting each user's results with a unique colour. 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. As mentioned in section 3.1 by Larose and Larose (2015)[9], data mining requires continuous human supervision for quality monitoring and evaluation.

6 Conclusion

This thesis compared different time deltas for aggregation, to determine which one is ideal to construct high quality clusterings from smartphone sensor and usage data. The data was recorded from different test subjects for the SmartEater mobile health app. The datasets aggregated into 1h and 3h files 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 of the total 10 time lengths (1h and 3h files combined). 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 suggests, that the following three time lengths produce the most distinct and well defined clusters: 30 minutes (from the 1h dataset), 1 hour, and 2 hours (both from the 3h dataset). The results of this study suggest, that various time lengths might be necessary to receive the clearest clusters. User studies to evaluate hand drawn clusters would be a further step to identify appropriate time lengths to generate more distinct clusters and therefore be used to predict eating crises.

References

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. Las Vegas, NV, USA, March. doi:`10.1109/BHI.2018.8333461`.

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. doi:`10.1145/304181.304187`.

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

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

Caliński, T., and J. 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, no. 2 (April): 224–227. doi:`10.1109/TPAMI.1979.4766909`.

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: AAAI Press, August.

García, Salvador, Julián Luengo, and Francisco Herrera. 2015. *Data Preprocessing in Data Mining.* Switzerland: Springer International Publishing. ISBN: 978-3-319-10246-7. doi:`10.1007/978-3-319-10247-4`.

Han, J., J. Pei, and M. Kamber. 2011. *Data Mining: Concepts and Techniques.* 3rd Edition. Burlington, Massachusetts: Elsevier. ISBN: 978-0-12-381479-1.

Hartigan, John A. 1975. *Clustering Algorithms.* New York, NY, USA: John Wiley & Sons, Inc. ISBN: 978-0-471-35645-5.

Hinton, Geoffrey E, and Sam T. Roweis. 2003. "Stochastic Neighbor Embedding." In *Advances in Neural Information Processing Systems,* 15:857–864. `http://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf`.

Hotelling, Harold. 1933. "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology* 24 (6): 417–441. doi:`https://doi.org/10.1037/h0071325`.

Jolliffe, I.T. 2002. *Principal Component Analysis: Second Edition.* Springer Series in Statistics. New York, USA: Springer-Verlag New York. ISBN: 0-387-95442-2. doi:`10.1007/b 98835`.

Larose, Daniel T, and Chantal D Larose. 2015. *Data Mining and Predictive Analytics.* 2nd Edition. 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. Oxford: Butterworth-Heinemann (Elsevier). ISBN: 9780128004081.

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* (Basel) 18, no. 7 (July): 2134. doi:`10.3390/s18072134`.

Pyle, D. 1999. *Data Preparation for Data Mining.* Burlington, Massachusetts: Morgan Kaufmann. ISBN: 9781558605299.

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,* 141–150. Montréal, Québec, Canada: Association for Computing Machinery, April. ISBN: 9781450342247. doi:`10.1 145/2896338.2896359`. `https://doi.org/10.1145/2896338.2896359`.

Romesburg, H. Charles. 2004. *Cluster Analysis for Researchers.* Morrisville, North Carolina: Lulu Press. ISBN: 9781411606173.

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. doi:`10. 1016/0377-0427(87)90125-7`.

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. Nan, Thailand, January. doi:`10.1109/ECTI-NCON.2019.8692267`.

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* (October). 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

A.2   Learning Rate

A.3   Clustering results

A.4   Clustering evaluation results

B   git-Repository

Link to the GitLab Repository on `gitlab.mediacube.at`:

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

Git repository contents:

- **Experiment:** Source code of the experiment

- **Thesis:** LaTeX code of the thesis

- **Literature:** Reference papers available as PDF

C   Archived Websites

`https://web.archive.org/web/20200623170602/https://distill.pub/`
`2016/misread-tsne/`, snapshot 23.06.2020, 17:06:02