FH Salzburg
MultiMediaTechnology

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

Bachelor Thesis 2

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

Salzburg, Austria, 29.06.2020

Affidavit

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

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

Date *Signature*

First Name *Last Name*

Kurzfassung

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

Abstract

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

Contents

List of Figures

Listings

List of Tables

1 Introduction

(**Todo: adapt to experiment**)

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

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

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

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

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

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

1. Background volume

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

2. Relative movement of the smartphone (gyro and accel)
3. Time and duration of phone calls (without storing the numbers)
4. Time of messages (e.g. SMS, WhatsApp) (without collecting identifying information such as content, addresses, numbers)
5. Screen activity (so-called touch events)
6. Screen-on-time (illuminated display)
7. Ambient brightness
8. Data volume per unit of time (summary value of all smartphone activities on the internet)
9. Switch-on and switch-off times of the smartphone

This sensor data will be recorded for different lengths of time (**TODO: name lengths of time when experiment is complete**). It is necessary to establish which time period will be most fitting to make accurate predictions for the future. This thesis will use cluster analysis to determine which time period is most significant.

According to Han, Pei, and Kamber (2011)[414], the above-mentioned clustering methods work well with data sets that are not high-dimensional and have less than 10 attributes. Since the SmartEater data set only has 9 dimensions, it is not considered high-dimensional. This paper will therefore utilise these clustering methods. Since different clustering algorithms can yield different results, multiple methods will be used and compared.

To reduce the size and amount of data, dimensionality reduction will be used. Han, Pei, and Kamber (2011)[93] define dimensionality reduction as a type of data reduction, which removes random attributes and creates a smaller data set with close to equal integrity. This thesis will use principal component analysis (PCA) to reduce the dimensionality. Furthermore, T-Distributed Stochastic Neighbor Embedding (t-SNE) will be employed to depict the data set in this thesis. Maaten and Hinton (2008)[2579] first introduce t-SNE, which is used to visualise data with a higher dimensionality.

The clustering methods will be implemented using a Python machine learning platform or library (e.g. Anaconda[2], scikit-learn[3]). Next, these will be implemented on the other time lengths. The resulting clusters of each time length will be compared to one another and evaluated. Rousseeuw (1987) reveals how silhouettes can be used to measure the separation between clusters and therefore evaluate the quality of the resulting clusters are.

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

2. `https://www.anaconda.com/`
3. `https://scikit-learn.org/stable/`

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

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

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

In order to predict "About-to-Eat" moments, eight participants, aged 26-54, wore these four technologies for five days. The recorded data then underwent cleaning and preprocessing, feature extraction, feature selection and machine learning. During the preprocessing, it was made sure that the raw sensor data streams were not dirty and had high accuracy. Some of these steps included resampling, normalisation, recognising chewing and swallowing sounds, extracting longitude and longitude, etc. In the feature extraction step, two parameters are used on the

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

processed sensor time series to create windows. Firstly, the feature extraction window size parameter regulates the time duration in a specific window. While a short window duration could catch immediate characteristics in the sensor time series, a coarse one could be used for long term trends. The authors used a variety of window sizes, varying from 5 to 120 minutes. The prediction model results would decide the best window. The second parameter, the window shift size, establishes the time duration between two neighbouring windows, for example meaning window *n* is shifted one minute compared to window *(n - 1)*. A constant shift of one minute was used for all window sizes. Statistical functions (e.g. min, max, mean, standard deviation, etc.) were used to extract a total of 158 window-level features. Two features were added to the sensor streams: current time (minutes since the start of the day) and time since the last eating event (minutes) and number of eating events in that day. In the feature selection step, the most relevant features were selected (e.g. the location features were not beneficial and merely represented noise). In order to train an ”About-to-Eat” moment classifier, signal processing and machine learning were used on the sensor streams. The classifier was trained to recognise the two classes ”About-to-Eat” and ”Not-About-to-Eat”. The average recall of this model resulted in a recall, precision and F score of 0.77, 0.67 and 0.69 (respectively).

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

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

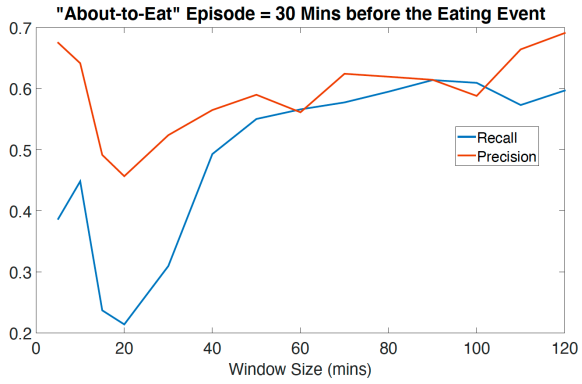

Figure 1: This graph depicts the F measure with the different feature extraction window sizes of the ”About-to-Eat” classifier.

The performance for the ”Time until Next Eating Event” model gained a correlation coefficient of 0.49. The most fitting feature extraction window size was assessed, the best performance

was reached with a window size of 100 minutes. The authors further claim, that both models could be improved by incorporating person-dependent data from the target user to the models (e.g. person-specific eating pattern, lifestyle).

Stütz et al. (2015)[240, 242-249] research, whether data collected by a user's smartphone can be used to predict stress. In their study, they used an android smartphone app called "TheStress-Collector" (TSC) to collect smartphone usage and sensor data in the background. The following data was collected:

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

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

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

Other related work: Pius Owoh, Mahinderjit Singh, and Zaaba (2018) use the k-means clustering algorithm to create automatic annotation of unlabelled smartphone sensor data, to observe sensitive location information of mobile crowd sensing users. Sornbootnark and Khoenkaw

(2019) use smartphone sensor data (e.g. accelerometer) to predict excessive alcohol consumption, in order to replace breathalysers. Their algorithm produced 100% accuracy, however with 15 minute lags.

3 Theory/Literature Review

3.1 Data Mining

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

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

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

Other problems in data mining include underfitting and overfitting. In their paper on attempting to balance underfitting and overfitting, Aalst et al. (2010)[87-89] clarify when these can occur.

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

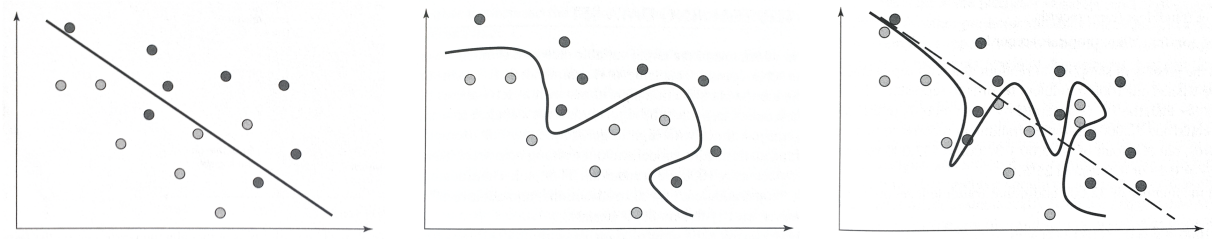

Figure 2: These three scatter plots from Larose and Larose (2015)[164-165] depict the bias-variance trade-off. The first plot portrays a low-complexity resulting in a high error rate. The second plot achieves a low error rate by using a high-complexity separator. The third graph compares the two separators.

3.2 Data Preprocessing

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

To make data useful in data mining, Larose and Larose (2015)[20] point out, that data sets first need to undergo a data preprocessing step, including data cleaning and data transformation. Raw data extracted directly from databases can be incomplete (values are missing), noisy (contains outliers), or may contain out-dated or redundant data. García, Luengo, and Herrera

(2015)[45] list the following sources of dirty data: data entry errors, data update errors, data transmission errors and bugs. Dirty data can impact the produced model, making it less reliable. The significance of its effect depends on the implemented data mining method. Larose and Larose (2015)[20] define the goal as to decrease garbage in, garbage out (GIGO). To clarify, decreasing *garbage in* means to reduce the irrelevant data that is fed into the model, thus reducing the amount of irrelevant data received out of the model (*garbage out*). Further information to different types of dirty data and their solutions are outlined in the next sections below.

3.2.1 Noisy Data

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

3.2.2 Missing Values

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

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

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

3.2.3 Normalisation

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

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

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

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

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

3.2.4 Data Transformation

Todo: complete (if needed)

As reported by Larose and Larose (2015)[39-41, 45], flag variables can be used to transform categorical variables into numerical. A flag variable can take on one of two values: 0 and 1 (e.g. female = 0, male = 1). When k $\geq$ 3 (k being the amount of categorical predictors), the variables can be transformed into k-1 flag variables. Assigning categorical variables numerical values is not advised, since this orders the categorical variables. For example, if North = 1, East = 2, South = 3 and West = 4, West would be closer to South than to North, etc.

ID fields should be removed from the data set, since the value is different for each record and not helpful.

3.3 Dimensionality Reduction

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

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

3.3.1 Principal Components Analysis (PCA)

Principal Components Analysis was first proposed by Pearson (1901) and Hotelling (1933). Pearson's approach is to identify a line or plane that best fits the collected variables plotted to a plane. In order to determine the best fitting line or plane, means, standard-deviations, and correlations are used (Pearson 1901)[559-560]. Hotelling (1933)[5] introduces his method

as *the method of principal components*. In his paper, Jolliffe (2002)[7] clarifies, that while these two papers used different methods, the standard algebraic derivation was announced by Hotelling (1933).

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

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

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

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

todo: more detail (if use PCA in experiment).

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

In their paper, Maaten and Hinton (2008)[2579-2580] introduce t-SNE. It is a method of visualising high dimensional data on a two or three-dimensional plot. This technique is also used to reduce the number of dimensions. The authors mention, that while linear methods like PCA concentrate on making sure low-dimensional representations of data points apart from each other, they are typically unable to bring similar low-dimensional representations near. The authors describe, that t-SNE is a variation of Stochastic Neighbor Embedding (SNE), introduced by Hinton and Roweis (2003). The basic algorithm, as explained by Maaten and Hinton (2008)[2581-2582] and introduced by Hinton and Roweis (2003)[2], is as follows: The first step in SNE is to transform the Euclidean distances between points in high-dimensional space into the conditional similarity probabilities. For example, the conditional probability $p_{i,j}$ would arise from the similarity between the two datapoints x_i and x_j. This is the probability of x_i picking the point x_j as its neighbour, under the circumstance that the neihbours were chosen according to their probablity density under a Gaussian, x_i being its centre. This results in the conditional probability being high for close data points and imperceptible 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. 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 $q_{i,j}$.

The goal is to reduce the difference between $p_{i,j}$ and $q_{i,j}$, thus finding a suitable low-dimensional representation of the high-dimensional data. This is accomplished by minimising a cost function, comprised of the sum of the Kullback-Leibler divergences between $\mathrm{p_{i,j}}$ and $q_{i,j}$.

Maaten and Hinton (2008)[2583] further explains 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.

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. Clustering is almost equivalent to classification. Real-world examples of clustering include classifications of animals, plants and diseases.

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

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

- Counts: no arbitrary scale (e.g. number of legs on a spider)
- Ratio scale: only defined in proportion to a standard volume (e.g. volume of a liquid in a glass)

- Interval scale: chosen from a standard position in a standard unit (e.g. height of a building)
- Ordinal scale: ordered classification, can be changed by a monotonic transformation (e.g. socio-economic status)
- Category scale: classification that can be adjusted by a one-to-one transformation (e.g. religion)

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

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

3.5 Overview of Clustering Algorithms

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

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

- Capability to cluster under certain constraints
- Interpretability and usability of the results

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

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

3.5.1 Partitioning Methods

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

Examples: k-means, k-medoids

3.5.2 Hierarchical Methods

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

Examples: BIRCH, Chameleon

3.5.3 Density-Based Methods

The majority of clustering methods (e.g. partitioning and hierarchical methods) use distance-based approaches, which results in only finding clusters with spherical shapes. Density-based methods have the ability to find clusters with random shapes. In these methods, objects are continuously added to the cluster, so long as the number of objects/data points (density) close by is larger than a given threshold. The clusters are comprised of high-density areas of objects. These are separated by spaces with low-density. Accordingly, this method is also useful for removing noise and outliers. These methods can also be used to cluster sub spaces.

Examples: DBSCAN, OPTICS, DENCLUE

3.5.3.1 DBSCAN

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

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

3.5.3.2 OPTICS

Han, Pei, and Kamber (2011)[388] mentions, that OPTICS was create d to improve the selection of global parameters problem in DBSCAN. Ankerst et al. (1999)[49, 51-] present the density base clustering algorithm OPTICS (Ordering Points To Identify the Clustering Structure). This method in itself does not specifically create clusters. Instead, it orders the data set

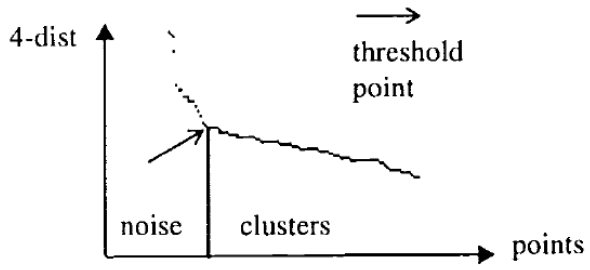

Figure 3: Sorted 4-dist graph (distance for each point to its fourth nearest neighbour) (Ester et al. 1996)[230].

according to its density-based clustering structure. For each object, the values *core-distance* and *reachability-distance* are calculated. The *core-distance* of an arbitrary data point (object) *o* is the distance to the nearest point within Eps that completes the MinPts rule and therefore labels point *o* as a core point. If there is not the number of MinPts in the Eps neighbourhood, then the *core-distance* of that point is undefined. The *reachability-distance* of an object *p* to core object *o* is defined as the max value of the core-distance and the distance from object *o* to object *p*. Likewise, if *o* is not a core object, then the reachability-distance of *p* is undefined. The *core-distance* and *reachability-distance* are visualised in figure 4.

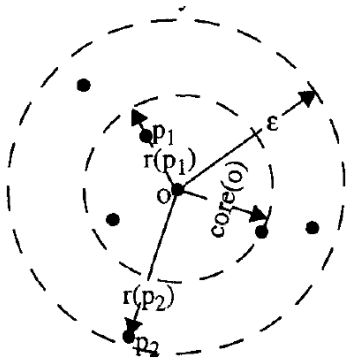

Figure 4: Core-distance of object *o* (MinPts = 4) and reachability-distances for objects *p1* and *p2* (Ankerst et al. 1999)[52].

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

The clusters can then be automatically constructed from the reachability plot by pinpointing the start-of-cluster and end-of-cluster regions and combining regions that match into clusters (or nested clusters). Since the reachability-distance of a point is the distance from the set of its predecessors and through OPTICS' specific ordering, the clusters are the dips in the reachability plots (as can also be seen in figure 5). In figure 6

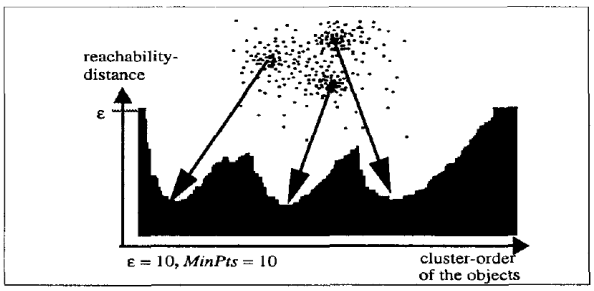

Figure 5: OPTICS reachability plot for a 2D data set. Three "Gaussian bumps" can be seen (Ankerst et al. 1999)[54].

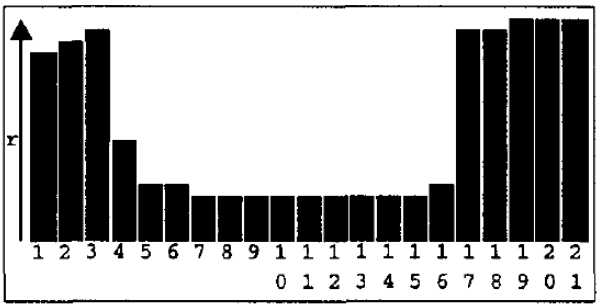

Figure 6: Since clusters are defined as the dips/dents in the reachability plots create by the OPTICS algorithm, a cluster can be extracted from this plot starting at the 3rd data point and ending with the 16th. The x-axis of the reachability plot depicts the order of the data points (objects) and the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

3.5.4 Grid-Based Methods

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

Examples: STING, CLIQUE

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

3.6 Evaluating Clustering Results

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

3.6.1 Assessment of the cluster tendency

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

3.6.2 Evaluation of the cluster quality

(**Todo: adjust to the methods used in the experiment**)

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

3.6.2.1 Silhouette Coefficient

In his paper, Rousseeuw (1987)[53-57, 59] proposes a new graphical display using silhouettes, to help determine how well objects belong to their assigned clusters. It can be used to interpret and validate the results of clustering. It is also utilised to compare the resulting clusters with those output using alternative algorithms (the input data being the same). In an example, where countries are assigned a value of how dissimilar they are to another country, the results are listed in a table. A structure contained in the results (consisting of 66 numbers), is hard to identify. Therefore, the countries are categorised into clusters using k-median. It is however uncertain, whether the clusters follow a specific structure, or if the groups are simply artificial. With the use of silhouettes, the author's goal is to answer the following questions: Is the quality of the clusters high, therefore the dissimilarities of the objects within a cluster are small, and large compared to the objects in other clusters? Are the objects well-classified, misclassified and which ones were not classified (between clusters)? Is it possible to perceive which "natural"

clusters are available in the data set?
According to Rousseeuw, the silhouettes are ideal when the distance between objects are on a ratio scale (e.g. Euclidean distances) and when the goal is to receive clear and compact clusters. For each object *i* (in cluster *A*) the value *s(i)* is calculated. *a(i)* contains the average dissimilarity ob the object *i* to each other object in the same cluster. If there are no other objects in the cluster, *s(i)* is set to zero (most neutral value). *b(i)* is determined, by firstly calculating the average dissimilarity, for each neighbouring cluster that isn't *A*. The shortest of these values, therefore the next closest cluster to *A*, is then assigned to *b(i)*. This cluster can so to say be seen as the next best choice for *i*. *b(i)* can only be calculated, if there are other clusters beside *A*. The formula is as follows:

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

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

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

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

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

3.6.2.2 Davies-Bouldin Index

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

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

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

The Davies-Bouldin Index equals:

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

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

3.6.2.3 Calinski-Harabasz Index

Caliński and JA (1974) introduce their

3.6.3 Establishing the number of clusters

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

4 Method

The goal of this paper is to identify, which time delta for aggregation is ideal to construct distinct clusters from smartphone sensor and usage data. The data for this experiment was collected for the upcoming SmartEater [5] mobile health app. The goal of this app is to present the user with content-dependent feedback, with the hope to prevent food craving episodes. By evaluating

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

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):

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

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

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

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

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

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

4.1 Preparation of the Data Set

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

4.1.1 Missing Values

The raw data contained several rows with empty cells. Section 3.2.2 lists approaches on how to substitute missing data, thus preserving the row, which could otherwise contain important information. In doing so however, the existing patterns could be disrupted. In order maintain the data's patterns, rows with missing values were removed. Using the Pandas' dropna[10] function (deletes rows with missing values), all rows containing empty cells were dropped. Of the original 8283 rows from the 1 hour time period files, only 3279 (39.59%) rows were complete and remained. In the 3 hour files, only 6218 from 14091 (44.13%) persisted.

4.1.2 Normalisation

The data recorded from the different smartphone sensors returned values with different ranges. For example, whilst the values that describe the screen on time ranged between -20 and 2.0, the light sensor values could reach from 0 to over 60,000. As explained in 3.2.3, values with higher ranges can inadvertently outweigh smaller values. To be sure that the values are weighted the same, *Min-Max Normalization* was used to map all the values into the common range, e.g. [0,1]. The sklearn MinMaxScaler was initially used to calculate the new, normalised values in the data set. Most of the values in the data set ranged between 0 and just over 100. The light sensor values were the exception, with its values reaching above 60,000. As mentioned in section 3.2.3, Z-score normalization is more robust to outliers, that could otherwise bias Min-Max Normalization. Therefore, the sklearn StandardScaler was used instead of the MinMaxScaler, which implements the Z-score 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 do not contain any predictive content. The TIME column was removed for this reason. Since the TIME column only showed the time and data when the data was recorded

9. `https://pandas.pydata.org`
10. `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html`

periodically (in fixed intervals), it was **serial ???????** data that had no influence on the data or any predictive value (like an index). It could have however bias the results if left in.

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

4.1.4 Chain shaped data

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

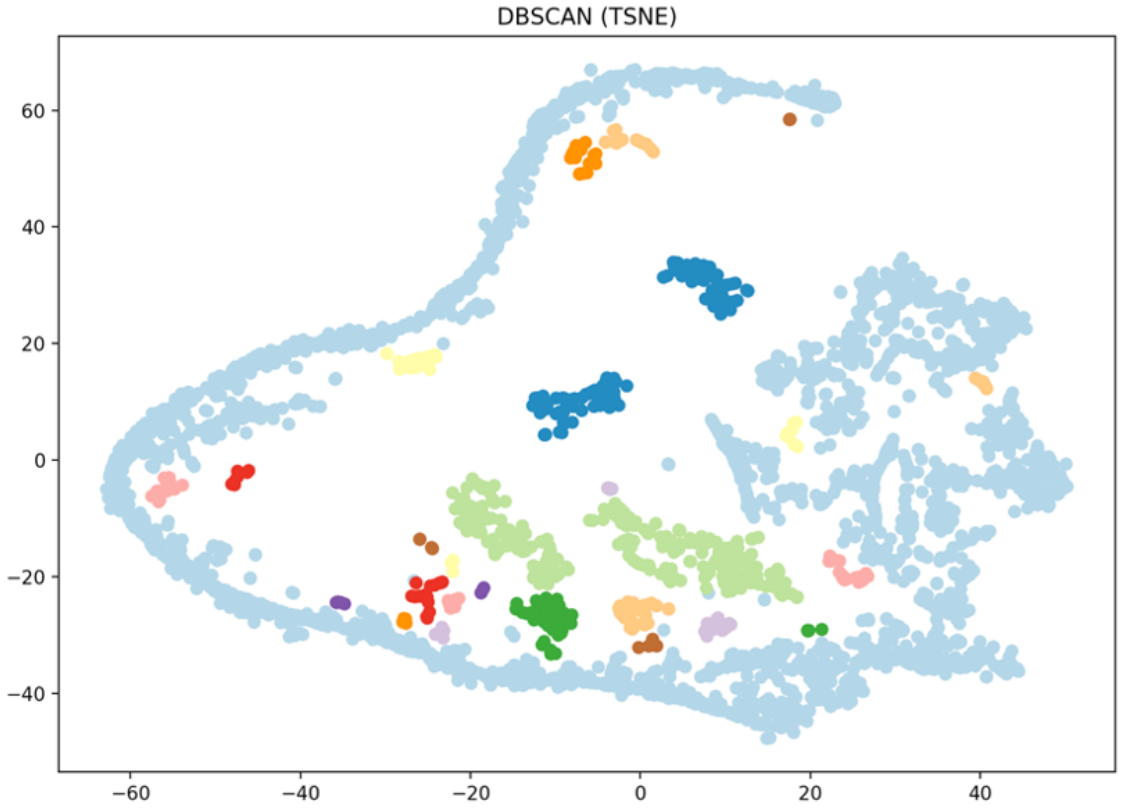

Figure 7: 1h data set, a chain of data points (light blue points) can be seen in the data set.

An initial thought, was that this chain was the result of the utilised t-SNE parameters. However, even with changing the t-SNE perplexity, learning rate and number of iterations parameters, the chain persisted in some form. Another thought, as to where these data points originated from

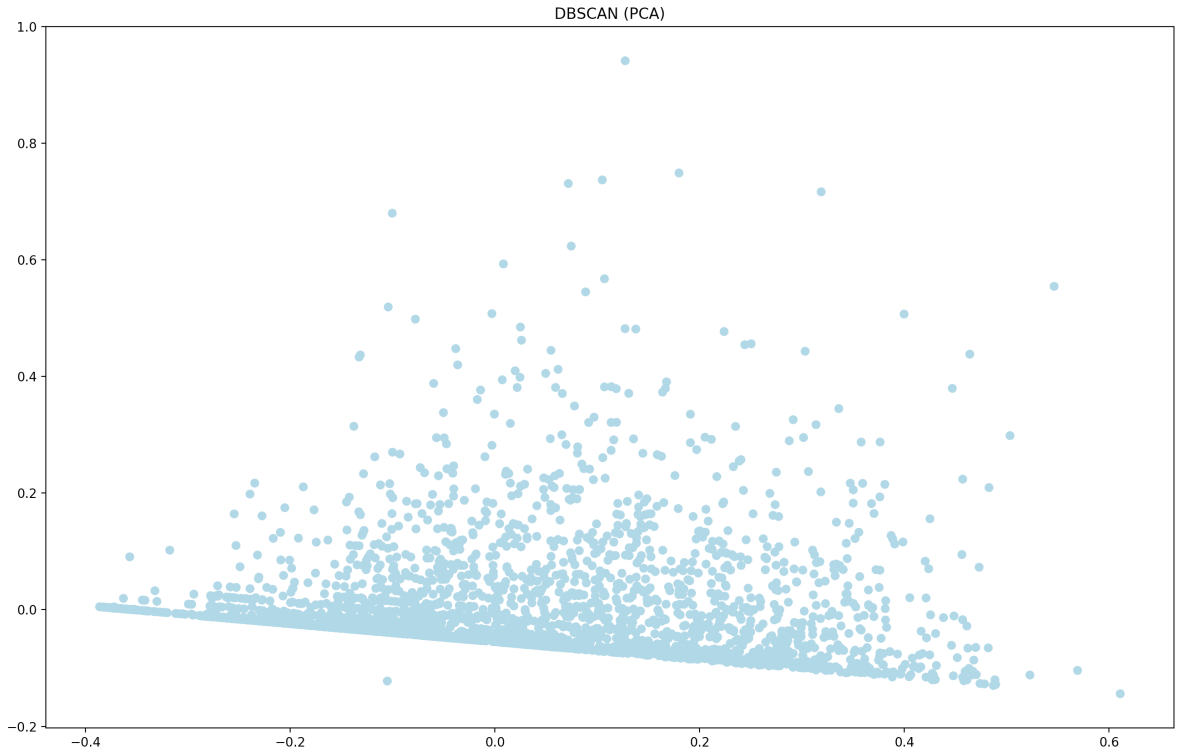

Figure 8: There is a collection of points along a line. This same collection can also be visualised as a chain in the t-SNE data set.

or why they resulted in a chain, was that there were dependencies within the data or columns. For example the three APP columns, these indicate the percentage of time in a lag-interval that a 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. By using this app for 100% of the time, would mean that the values in the other APP columns for this time would have to be 0 (0%), since they would not have been able to be used. To test this theory, the APP columns were removed from the data set and the 2D scatter plot was recalculated. The chain was still visible. Step by step each column was removed, to see if it was the cause of the chain. The chain did not disappear though, which indicated, that the problem was not due to the columns, but rather the rows. The chain was less visible when reducing the number of inserted .csv files (lower number of records) to 10 (see figure 9). When the number of features were reduced to only AUDIO and ACC (two columns), the chain was once again visible (see figure 10). The presumption is that when there are less data points and more features (e.g. 10 files but using all the features), there are enough different data points and the chain less dense, making it less noticeable. When there are less data points but with less features (e.g. 10 files but using only 2 features), or when there are many data points even with multiple features (e.g. all data files and all the features), there are more rows with very similar of equal values and the chain appears more dense and prevalent.

One theory was that points from a same test subject were similar and might be causing the

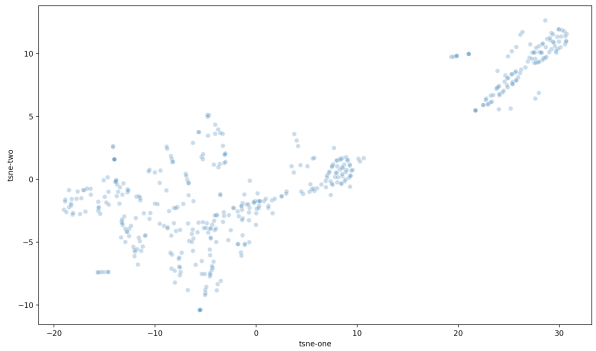

Figure 9

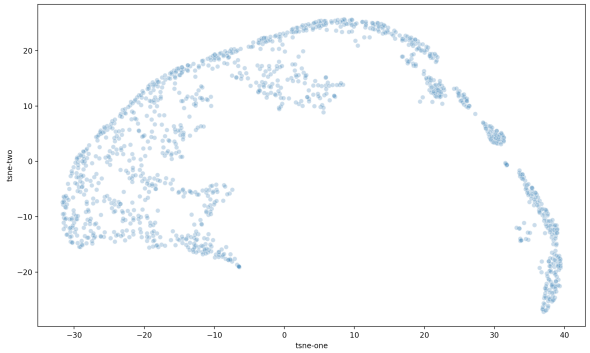

Figure 10

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 11. 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 next step was to use the data point indices and to look for similarities in the values. Several rows found in the chain were extracted from the cleaned data set, directly before being fed into the t-SNE algorithm. After comparing these, it was evident that they shared a lot of the same feature values. Rows not in the chain however had different values. Looking at the same rows in the uncleaned data set (after concatenation of the .csv data files and removal of rows with missing values), rows found in the chain had several columns all with the value 0. Further inspection showed, that when the SCRN value was 0, LIGHT, NOTIF and the APP columns where also 0. When a smartphone screen is off, the apps are not being used, which explains why the APP columns were mostly 0. The NOTIF values were often 0, presumably because the user was not constantly receiving notifications. The environment light is also often 0 when a phone is not being used, maybe because it is in a pocket or bag. To test whether these rows were causing the chain to occur, rows with less than 50% of values that weren't 0 were removed. As can be seen in figure **TODO: ADD FIGURE FROM LATEST**, the chain is only slightly visible.

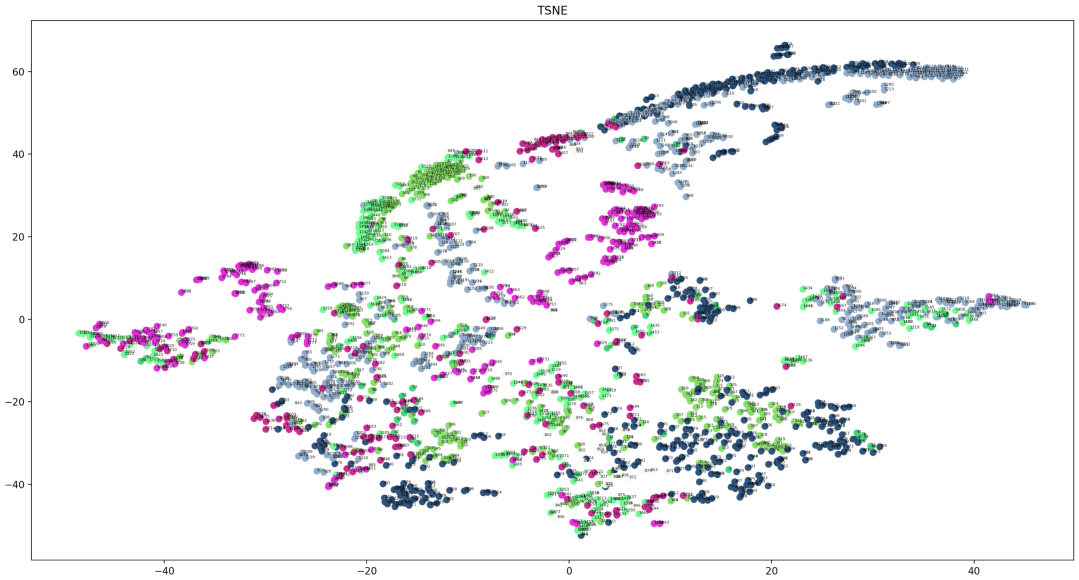

Figure 11

also thought from same person

4.2 Dimensionality Reduction

Dimensionality reduction was implemented to reduce the number of dimensions (number of attributes, so in this case number of columns). Principal Components Analysis (PCA) and t-SNE, as described in section 4.2, were used to reduce the dimensionality of the data set. PCA was the initial approach used in the experiment. The sklearn PCA[11] function was used to reduce the number of dimensions to 2, which simplified visualisation in 2D scatterplots. 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. As can be seen in figure **TODO: figures**, the resulting data from these components, didn't show any significant clusters, in comparison to the t-SNE results.

The t-SNE approach proved to be more significant.

Since the results

4.3 Clustering

The clustering algorithms used for the SmartEater data set, were DBSCAN and OPTICS. One of the advantages of these density-based clustering methods (as mentioned in section 3.5.3) are that there are less parameters to configure. Another reason for choosing these methods, is that

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

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 to arbitrary-shaped clusters to be correctly identified.

4.3.1 DBSCAN

Section 3.5.3.1... 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 generated for the 1h and 3h data sets. The graphs contained the distances to the k nearest neihbors. As recommended by Ester et al. (1996)[230], MinPts and k were set to 4 and the graphs were used to determine Eps. The idea is to select Eps suitable for the ”thinnest” cluster, however being careful to avoid noise. As can be seen in figures 12 and 13, the valley starts at roughly a 4th nearest neighbor distance of 2.

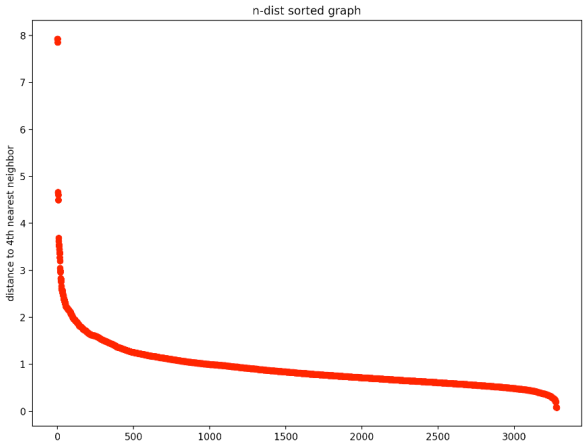

Figure 12: Sorted 4-dist graph of the 3h data set (distance for each point to its fourth nearest neighbor). The valley starts at roughly the 4th nearest neighbor distance of 2.

4.3.2 OPTICS

Since density based algorithms do not require specifying a number of clusters in advance

& clusters can be of any shape

optics: two options: xi (automatic technique from Ankerst et al. (1999), or dbscan)

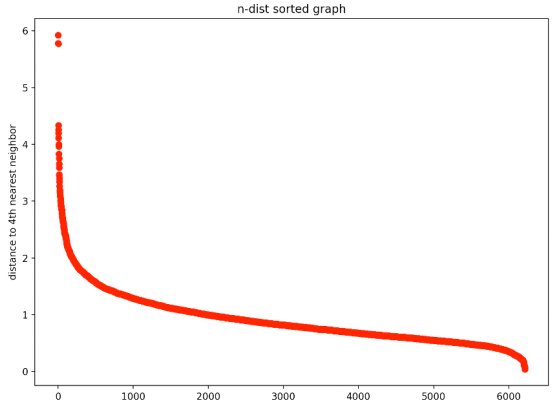

Figure 13: Sorted 4-dist graph of the 1h data set (distance for each point to its fourth nearest neighbor). The valley starts at roughly the 4th nearest neighbor distance of 2.

4.4 Comparison and Evaluation of clusters of different time lengths

4.4.1 Mathmatical Evaluation

4.4.2 Comparison of different time lengths

5 Discussion

CHAIN ARGUMENTATION

6 Conclusion

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Romesburg, H. Charles. 2004. *Cluster Analysis for Researchers.* 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.

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

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

Appendices

Anhänge löschen, die nicht verwendet werden.

A git-Repository

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

Verpflichtende Daten für Bachelorarbeit 1 und 2:

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

Verpflichtende Daten für Bachelorarbeit 2:

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

Link zum Repository auf dem MMT-git-Server `gitlab.mediacube.at`:
`https://gitlab.mediacube.at/fhs123456/Abschlussarbeiten-Max-Muster`

B Vorlagen für Studienmaterial

Vorlagen für Studienmaterial müssen in den Anhang.

C Archivierte Webseiten

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

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