LAPPEENRANTA LAHTI UNIVERSITY OF TECHNOLOGY
School of Engineering Science
Business Analytics

Ilari Tuomela

# EMPLOYING SOFTWARE USE DATA FOR CUSTOMER SEGMENTATION

Supervisors: Professor Pasi Luukka
M.Sc. Christoph Lohrmann

## ABSTRACT

**AUTHOR: Ilari Tuomela**

**TITLE: EMPLOYING SOFTWARE USE DATA FOR CUSTOMER SEGMENTATION**

**YEAR: 2019** **LOCATION: Helsinki**

Master's thesis. Lappeenranta Lahti University of Technology, School of Engineering Science, Business Analytics

82 pages, 16 figures, 16 tables.

Examiners: Professor Pasi Luukka, M.Sc. Christoph Lohrmann

**Keywords:** Software use data, software telemetry, logged use data, feature selection, clustering, customer segmentation, customer analysis, business analytics

The potential for collecting and logging software use data has increased substantially with the rise in popularity of various cloud computing and storage services. As the number of data sources and customers has increased, it has become ever more challenging to gain a consistent overview of how software products are used. The aim of this thesis was to gain insight on how the software product is used in practice. Another objective was to identify user actions or patterns that could be used for customer segmentation.

The thesis focused on finding the optimal combination of a feature selection method and an unsupervised machine learning algorithm. The goal was to be able to differentiate customers according to their software use patterns. Filter feature selection methods were deemed best for the task. Variance threshold, Spectral feature selection, Laplacian score and Multi-Cluster feature selection methods were experimented with and evaluated. Based on a literature review it was decided to use clustering to create the model. K-means, Fuzzy C-means, K-medoids and Hierarchical clustering were examined and assessed with different feature selection methods and a varying number of clusters and features. The best clusters were produced with Spectral feature selection in combination with K-means clustering. Optimal results were obtained by setting the number of clusters to two and including 75% best ranked features.

Based on the findings, a model was created to cluster customers into distinct segments. Using the model, users could be segmented into one of four segments according to how they've used the software. Cluster descriptive values were noted for each segment and used to characterize them.

**TIIVISTELMÄ**

**TEKIJÄ: Ilari Tuomela**

**OTSIKKO: OHJELMISTON KÄYTTÖDATAN HYÖDYNTÄMINEN ASIAKAS-SEGMENTOINTIIN**

**VUOSI: 2019** **PAIKKA: Helsinki**

Diplomityö. Lappeenrannan Lahden Teknillinen Yliopisto, School of Engineering Science, Business Analytics

82 sivua, 16 kaaviota, 16 taulukkoa.

Examiners: Professori Pasi Luukka, M.Sc. Christoph Lohrmann

**Hakusanat:** Ohjelmiston käyttödata, ohjelmistotelemetria, tiedonkeruu, piirteenvalinta, klusterointi, asiakassegmentointi, asiakasanalyysi, business analytiikka

Ohjelmistojen käyttödatan keräämisen potentiaali on kasvanut huomattavasti erilaisten pilvipalveluiden suosion kasvamisen myötä. Datalähteiden ja asiakkaiden määrän kasvaessa on yhä vaikeampi omaksua järjestelmällinen yleiskuva siitä, kuinka ohjelmistoa käytetään. Diplomityön tavoitteena oli saavuttaa ymmärrys siitä, kuinka tosiasiallisesti ohjelmistotuotetta käytetään. Työn toinen tavoite oli tunnistaa käyttäjätoimia tai toistuvuuksia, joiden avulla asiakkaat voidaan segmentoida.

Diplomityön painopisteenä oli löytää optimaalinen piirteenvalintamenetelmä sekä ohjaamaton koneoppimenetelmä. Tavoitteena oli segmentoida asiakkaat perustuen kerättyyn ohjelmiston käyttödataan. 'Filter'-piirteenvalintamenetelmät osoittautuivat parhaiksi tehtävää varten. Variance threshold, Spectral-piirteenvalinta, Laplacian score ja Multi-Cluster piirteenvalintamenetelmiä tutkittiin sekä arvioitiin. Kirjallisuuteen perustuen tehtiin päätös käyttää klusterointia koneoppimallin tekemiseen. K-meansia, Fuzzy C-meansia, K-medoidsia ja Hierarkista klusterointimenetelmiä tarkasteltiin ja arvosteltiin käyttäen vaihtuvia piirteenvalintamenetelmiä sekä klusterien määriä. Parhaat tulokset saavutettiin käyttämällä Spectral piirteenvalintamenetelmää sekä K-means -klusterointia. Tulokset optimoitiin käyttämällä kahta klusteria ja 75% kaikista piirteistä.

Löydösten perusteella luotiin klusterointimalli, jonka avulla asiakkaat pystyttiin segmentoida yhteen ryhmään neljästä perustuen asiakkaan käyttäytymiseen ohjelmistossa. Klusterien kuvailevia arvoja käytettiin asiakassegmenttien kuvaamiseen.

**ACKNOWLEDGEMENTS**

First and foremost I'd like to express my greatest gratitude to the commissioner of the thesis. It would not have been possible for me to complete this thesis if not for the amazing opportunity presented to me. The trust and responsibility I was placed with enabled me to exceed my initial expectations. In particular I'm grateful for the guidance and support of my research supervisor Robin Wikström. My special thanks are also extended to the rest of the staff of the company. It would not have been possible to complete this thesis without the aid and input of several colleagues.

I'd like to thank the staff Lappeenranta Lahti University of Technology for interesting lectures and studies. A special thanks should be given to my thesis supervisor Pasi Luukka for providing me with valuable input and advice during the process of completing my thesis. I wish to thank my classmates for the incredible time I had at university. Throughout my studies I had friends whom I could rely on. Finally I'd like to thank my family for their encouragement and support for all these years.

# 1. Introduction

The potential for collecting and logging software use data has increased substantially with the rise in popularity of various cloud computing and storage services. Logged software use data refers to data obtained by monitoring users and their actions within a software product. Currently, logged use data is predominantly being used to locate software bugs and conduct root cause analysis. However, the potential ways to utilize and benefit from logged software use data are almost limitless: logged use data presents an opportunity to utilize a huge amount of quantitative data and receive insight on *how* the product is used, and *which* features are utilized. In this thesis the term 'user' refers to a customer's company-level administrator account.

As the number of customers increases, it becomes ever more challenging to gain a consistent overview of how the product is used. Especially in the software industry, where the churn rate is high, the importance of creating a customer experience driven product is paramount. By analysing feature use and segmenting customers into distinct groups, it is possible to create a product that better fits the customers' specific needs. By adopting a more customer-driven mindset for product development, companies can expect an increase in the number of satisfied customers. This often leads to a significant improvement in customer loyalty, ultimately leading to a decrease in the likelihood of a customer to churn.

## 1.1. Purpose of study

The thesis was conducted as part of a large software technology company's ongoing project. The aim of this thesis was to gain insight on how the product is used in practice. This information is valuable to a wide audience, that includes decision makers in charge of the direction of the company's strategy and management, and experts from product management, Research & Development (R&D), User Interface (UI), User Experience (UX), marketing, product training and other life cycle related services. The goal was to create an easily understandable customer analysis, that could be used in various settings by decision makers with differing backgrounds.

Another objective was to identify user actions or patterns, that can be used to identify which segment the user belongs to. The aim was to create a model, that can segment a user into one of the identified groups according to the user's actions. In the long run, the model(s) and features derived from this project could be used to improve an on-going churn prediction model project and potentially create a more personalized user experience for each separate user segment.

The main goal of this project was to answer the following question:

'***What* kinds of users does the software have?**'

This goal was then split into two sub-questions:

***'How are the user groups different from each other?'***

***'What is the most efficient method to segment customers?'***

## 1.2. Significance of study

Customer journey mapping based on logged use data is becoming an increasingly relevant field of study and research. As companies develop more software with increased functionality and complexity, getting a realistic picture of which features are being used and how often becomes more difficult. The number of application users is on a rise, and, as a result, the group of users is becoming increasingly diverse. This has led companies to recognize the importance of identifying and segmenting similar users into smaller groups. Also, a 'customer experience'-driven approach to R&D and software development has become a popular theme among software and technology companies. The goal is to keep customers happy and build customer loyalty in an increasingly competitive field.

In practise, the only way to achieve this cost-efficiently is by inspecting, analysing and drawing conclusions from a large set of logged use data. Currently, only relatively little research has been conducted on the subject, especially when it comes to segmenting and clustering users according to their logged software use data. However, previous research has shown, that many companies have agreed on the potential in the field of analysing logged use data, and that there is a clear need for further investigation on the practical implementation of related analytics.

## 1.3. Delimitations, Limitations and Assumptions

The main risks in this project were related to the data itself. Since no previous analytics had been conducted on the data set used in this thesis, the amount of useful data was fairly unknown. There was also the possibility that critical features might be missing, meaning that the value and reliability of the analysis and predicted values was unknown. Another risk to consider was the possibility of messy data, missing values or outliers. If present, a significant effort might have to be made to preprocess the data into a clean analysable form. Also, new EU General Data Protection Regulation (GDPR) rulings had to be kept in mind, when handling potentially sensitive customer data.

# 2. Literature review

This section examines previous scientific research and studies on the subject and themes of this thesis. A number of studies were found on the subjects of feature selection and customer segmentation via clustering. Numerous studies described potential benefits, processes, methods, and algorithms. (Tripathi S., 2018, p.802. Bernard G. & Androitsos P., 2017a, p. 49-51. Väätäjä H., 2016, p.1-2). However, only a handful of studies contained an example of a practical implementation of the methods, and often only used a small subset of data (Qiuru C., Ye L., Haixu X., Yijun L. & Guanping Z., 2012, p. 1179-1182. Renaud K. & Gray P., 2004, p. 118). The studies often described only how the findings could be utilized instead of putting them into practise. In addition, no studies related directly to the practical implementation of logged software use data from multiple sources to cluster customers into segments. The literature sources used include books, online articles, software manufacturers' websites and documentation, the employer company's internal documentation and expert interviews. Various databases were used to look for scientific literature, namely Web of Science, Science Direct, Google Scholar, Semantic Scholar and IEEE Xplore.

## 2.1. Feature selection

Feature reduction is steadily becoming a more prominent factor in business intelligence and data analytics. Data sample size and the number of features is continuously growing over time, in other words, the data's dimensionality is growing. Since high dimensional data is often sparser, traditional machine learning algorithms, originally created for lower-dimensional data, often deteriorate in quality when applied to high-dimensional data. There is also a greater demand for computation and memory storage requirements for high-dimensional data. To combat these issues, feature selection as a dimension reduction technique has been proven to be an effective strategy. (Li J., 2017, p.9).

In general, irrelevant features are features, that do not help distinguish samples from different segments for supervised learning algorithms or clusters for unsupervised learning algorithms. In addition, data with a high number of features but low number of training data often leads to overfitting of the model. (Wang S., Tang J. & Liu H., 2016, p.1-3).

There are two main methodologies to reduce the number of features: features extraction and feature selection. They are both capable of improving algorithm performance, lowering computational complexity and therefore reducing requirements and costs, building better and more generalized models, and decreasing the amount of required storage. Feature extraction maps the original features into a new feature space with a lower dimensionality, whereas feature selection chooses a smaller subset of the original set of features. Feature extraction is an effective method to aid building a generalized model. However, it makes further analysis harder, since features obtained from the feature extraction process have no physical meaning. (Li J., Cheng K., Wang S., Morstatter F., Trevino R., Tang J., Liu H., 2016, p.2-4). Since one of the goals of this thesis is to cluster customers into distinct groups and gain meaningful insight on their use of a software technology company's product, the research conducted in this paper was focused primarily on feature selection methods.

There are numerous algorithms, strategies and presumptions that need to be considered, when selecting a feature selection method. Figure 1 depicts the main characteristics to take into consideration when selecting a feature selection procedure (Wang S. et al., 2016, p.1-9):

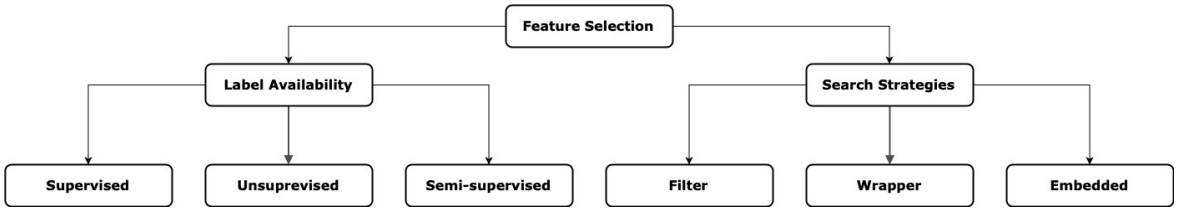


*Figure 1: Feature Selection Categories*

Before selecting a feature selection algorithm or method, the dataset itself has to be inspected. Depending on whether or not segment labels are available, the appropriate feature selection methods might vary. For labelled datasets, feature selection methods suitable for supervised machine learning algorithms have to be used, whereas for unlabelled datasets, feature selection methods for unsupervised machine learning algorithms should be applied. There are also feature selection methods that are suitable for partially labelled datasets, that utilize semi-supervised learning algorithms. (Wang S. et al., 2016, p.1-9).

Once the desired type of learning algorithm has been established, a suitable feature selection search strategy can be selected. There are three main types of search methods: filter methods, wrapper methods and embedded methods. Filter methods evaluate the score of each feature according to a selected criterion, and select the features with the best score (Alelyani S., Tang J., Liu H., 2013, p.6). Wrapper methods use a learning algorithm, and based on an evaluation criterion, such as accuracy, score features and evaluate feature relevance. Embedded methods embed feature selection into the learning model itself. (Liu H. & Motoda H., 2007, p.10). The methods and strategies mentioned previously are described in more detail in the following segments.

### 2.1.1. Supervised and unsupervised methods

As stated previously, feature selection algorithms can be divided into two main categories according to the availability of cluster labels: supervised and unsupervised. Supervised feature selection works usually as a preprocessing step for the creation of classification or regression models. It aims to choose features, that can separate the data samples from different clusters or regression targets well. A feature's relevance is usually assessed by its correlation with cluster labels. (Liu H. & Motoda H., 2007, p. 9). The training phase of a classification or regression model depends heavily on the input features. The goal is to train the models on a smaller subset of features, chosen by the feature selection method, and achieve as accurate of a model as possible. Supervised feature selection models can utilize wrapper, filter or embedded methods. (Li J. et al., 2016, p.2-4). Figure 2 demonstrates a typical model creation framework utilizing supervised feature selection as proposed by Wang et al (2016).

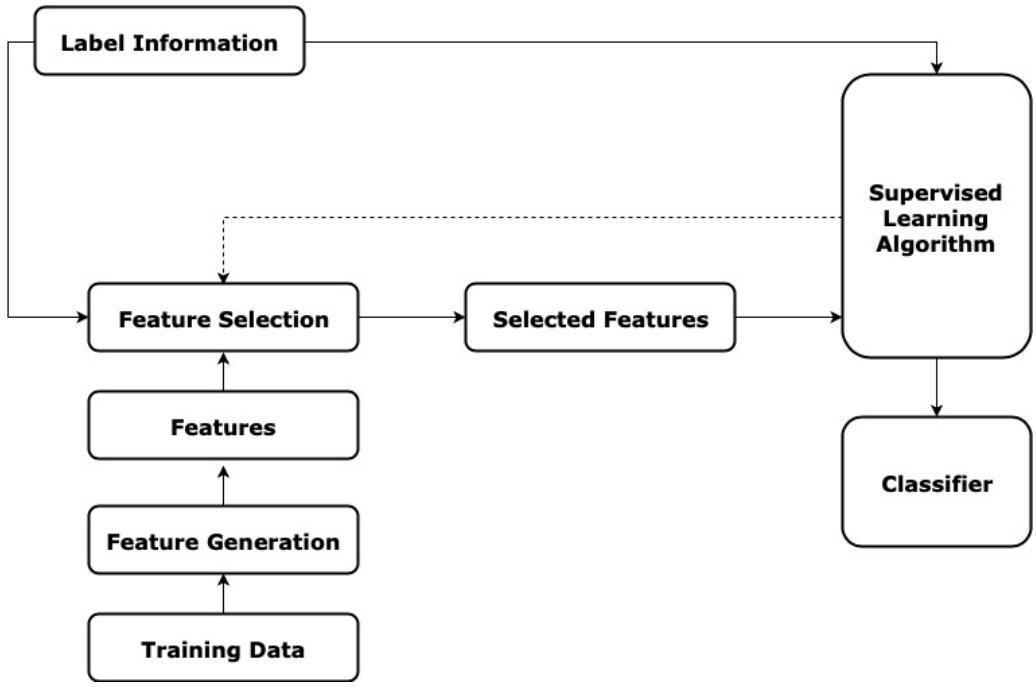


*Figure 2: Supervised Feature Selection Process*

Initially, the data is split into two segments: training data and test data. Training data usually contains at least half of the whole dataset, and is used for model creation (Wang S. et al., 2016, p.1-9). Depending on the characteristics and granularity of the data, new features may be generated or derived from the original training dataset and its features. Granularity refers to the level of detail in the data. These features are then evaluated by a feature selection algorithm chosen by the user. Since label data is available for the data samples, the feature selection algorithm can utilize them to rank features, as individuals or groups, according to how well they are able to differentiate features with different labels. Once the features have been evaluated, a select group of features, chosen by the algorithm or by the user, are used to create a classification model. The model is then evaluated, and according to the user's needs and requirements, either accepted or rejected. If the model is rejected, feature selection is applied on the training dataset again. The result of the feature selection process can be affected by adjusting the parameters used by the feature selection algorithm, selecting a different number or subset of features after the feature selection algorithm or by changing the feature selection algorithm altogether. This process is repeated, until the desired result from the learning algorithm is achieved.

Unsupervised feature selection methods are generally applied for clustering tasks. Often, the model defines the features that are potentially useful for unsupervised learning tasks. However, often some of the selected features are not relevant or relevant features might be left out. In such cases there is a need for an automated feature subset selection algorithm for unlabelled data. (Dy J. & Brodley C., 2004, p.845).

Since cluster labels are not available, the feature selection algorithm evaluates feature importance by utilizing a criterion such as data similarity or local discriminative information. (Liu H. & Motoda H., 2007, p. 10). Unsupervised feature selection models can use filter, wrapper and embedded methods (Li J., et al., 2016, p.3). Figure 3 demonstrates a general model creation framework utilizing supervised feature selection as suggested by Wang et al. (2016).

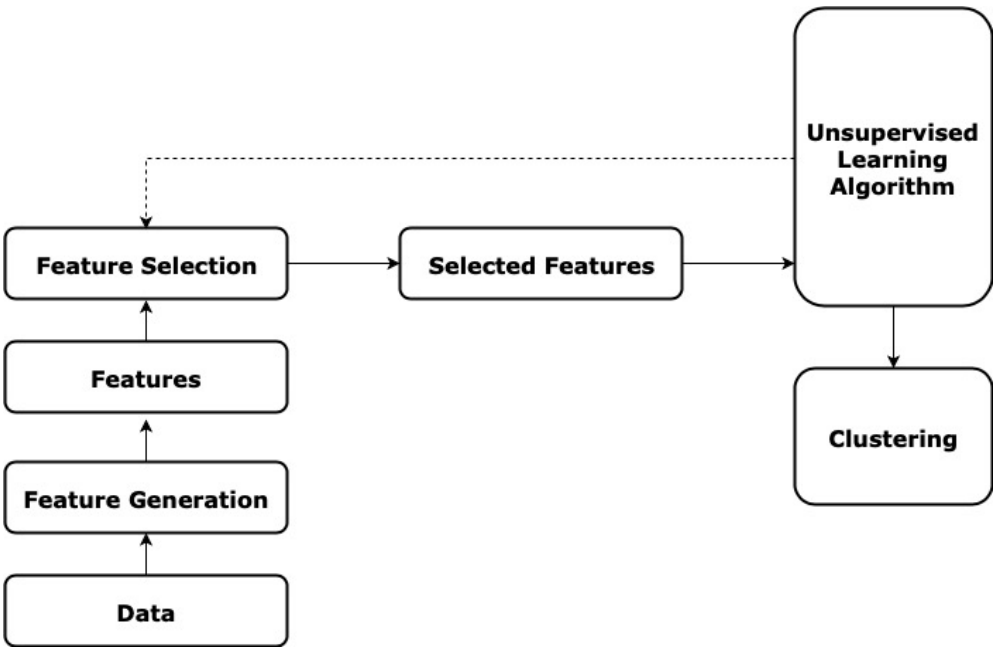


*Figure 3: Unsupervised Feature Selection Process*

For unsupervised feature selection, the process is similar to supervised feature selection. First, features are derived or generated from the dataset according to the complexity and granularity of the data for all of the samples in the dataset. These features are then evaluated by a feature selection algorithm. Since the dataset has no labels by which to evaluate the goodness of a feature, the goodness is evaluated by measuring how well a feature or set of features fulfils an objective function. These selected features, chosen either by the user or

the feature selection algorithm, are then used to create a model. Depending on whether the model fulfils the criterion or criteria set by the user, the model is either accepted or rejected. If the model is rejected, feature selection is applied again on the dataset. Likewise, as with supervised feature selection, the resulting subset of features can be affected by altering the parameters used by the feature selection algorithm, by changing feature selection algorithm itself or by selecting a different subset of features.

It should be noted, that there are potentially three different ways to apply feature selection for unlabelled data: before clustering, during clustering or after clustering (Dy J. & Brodley C., 2004, p.875). A common method is to first seek cluster memberships through clustering algorithms. Thus the unsupervised feature selection process can be transformed into a supervised process based on the cluster labels. Initially, cluster memberships have to be found, after which feature selection is performed to remove or select certain features. The clustering algorithm is then performed again, followed by the feature selection algorithm, until the desired amount of features are left. An advantage is, that the clustering performance can be evaluated at the same time. (Wang S. et al., 2016, p.5).

A third case is also possible, where a small portion of the data is labelled. In this case, semi-supervised feature selection is usually used. In this case it is common, that neither supervised nor unsupervised feature selection methods give the best results. Due to labelled data being insufficient, supervised feature selection might not be able to select relevant features to give a true representation of the distribution of features. Unsupervised feature selection won't use label information, which might give important information on how to separate groups. The general framework for semi-supervised feature selection is the same as for supervised feature selection, but it relies on the construction of a similarity matrix. Features are selected based on which features best fit the similarity matrix. (Wang S. et al., 2016, p.5).

### 2.1.2. Wrapper methods

Wrapper feature selection methods use a classifier to evaluate feature subsets by their predictive accuracy on test data. This is achieved by statistically resampling or cross-validating different subsets of features. In other words, wrapper feature selection algorithms repeatedly choose a different subset of features and then evaluates the learning performance using the

selected features. This process is repeated, until the highest learning performance is obtained. (Liu H. & Motoda H., 2007, p.10). Figure 4 below represents the process for wrapper feature selection as suggested by Gutierrez-Osuna R. (2019, p. 4-8).

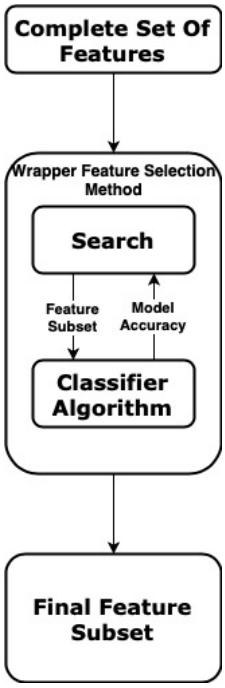


*Figure 4: Wrapper Feature Selection Process*

In general, wrapper feature selection algorithms achieve better recognition rates than filter feature selection algorithms. This is mainly due to wrapper feature selection algorithms being tuned to the specific interactions between the chosen learning algorithm and dataset. Wrapper algorithms also have the benefit of being able to generalize and avoid overfitting. This is typically achieved by cross-validation. (Gutierrez-Osuna R., 2019, p. 4-8).

However, there are some drawbacks to wrapper feature selection algorithms. Wrapper algorithms are often slow to execute. Since the wrapper algorithm has to train a learning algorithm for each feature subset, or multiple learning algorithms if using cross-validation, the method can become very heavy computationally, or even unfeasible in some cases. It should also be noted, that the wrapper feature selection method partially lacks generality. This is due to the wrapper algorithm being tied to the bias of the classifier used in the evaluation function. The optimal solution presented by the wrapper algorithm will be specific to the classifier used by the algorithm. (Gutierrez-Osuna R., 2019, p. 4-8).

### 2.1.3. Filter methods

Filter feature selection methods are feature selection methods that are independent of learning algorithms and rely on the characteristics of the data to assess the importance of a feature. In other words, filter feature selection methods evaluate feature subsets by their information content, a measure of how well a feature can describe the dataset. A feature that describes the dataset well can, for example, have a strong link to cluster labels and be effective at separating samples from different clusters from each other. A filter feature selection algorithm typically consists of two separate steps. First, features are ranked according to a criterion. In univariate cases, each feature is ranked individually one-by-one, whereas in multivariate cases combinations of multiple features are ranked. Second, the features are ranked according to how well the feature evaluation criteria ranked them, and the lowest ranked features are filtered out. Multiple different approaches to information criteria, also called evaluation criteria, have been proposed. Proposed approaches include, for example, feature correlation, mutual information and feature discriminative ability. (Li J. et al., 2016, p.2-4). Figure 5 represents the filter feature selection process (Gutierrez-Osuna R., 2019, p. 4-8).

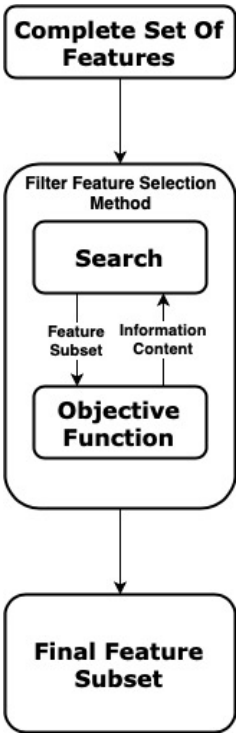


*Figure 5: Filter Feature Selection Process*

The main benefit of filter feature selection methods is, that they are fast to execute. Filter algorithms generally use non-iterative computation on the dataset, which are usually faster to execute than classifier algorithms. Additionally, filter feature methods are very general, and can be used in almost any situation. Since they rely on the inherent properties of the data rather than the interactions and accuracy with a particular classifier, the results are often more general. In practise, this means that the solution has the potential to be good for a larger family of classifiers. (Gutierrez-Osuna R., 2019, p. 4-8).

The main disadvantage to filter feature selection methods is, that they tend to select large subsets of features. Due to the filter method's objective functions being generally monotonic, the algorithm often selects a large portion or all of the features as the optimal solution. This often forces the user to select an arbitrary cut-off point to limit the number of features to be selected. (Gutierrez-Osuna R., 2019, p. 4-8).

#### 2.1.3.1. *Removing features with low variance*

Removing features with low variance is one of the simplest baseline approaches to feature selection. The algorithm removes all features, whose variance doesn't meet a certain threshold set by the user. The algorithm is one of the most basic methods of feature evaluation and feature selection (Xiafei H., Deng C., Partha N., 2006, p.1). By default, the algorithm removes all zero-variance features, or in other words all features that have the same value in all samples. The higher the threshold set by the user, the more variance is required by the features to be accepted by the algorithm. (Pedregosa F. et al., 2011, p. 2825 – 2830). Removing features with zero variance is a simple feature selection method, since it removes all features that are ineffective at segmenting samples into different clusters. This leaves the user with a subset of features, that help explain the dataset. Removing features with low variance can also be an effective approach especially when selecting which features to use to visualize dataset (Xiafei H. et al., 2006, p.1-9).

#### 2.1.3.2. *Laplacian score*

The Laplacian score is a filter feature selection method first introduced by Xiafei H. et al. (2006, p.1-9). The method is applicable for both supervised and unsupervised feature selection tasks. The Laplacian score method is based on the knowledge, that in many real-life classification problems, the data within the same cluster is often close to one another. By utilizing this knowledge, the importance of a feature is evaluated by its ability of locality preservation.

The main argument for the effectiveness of the Laplacian Score method is that more traditional feature selection criteria, such as data variance, often find features that are useful for representing data. However, the same features are not necessarily useful for separating data into different clusters. The Laplacian Score is based on the observation, that two points of data are most likely related to the same topic if they are near one another. The method's basic idea is to calculate a value for each feature that effectively reflects its locality preserving power, and can be used to compare features with each other. The algorithm is fundamentally based on Laplacian Eigenmaps and Locality Preservation Projection. The algorithm can be described as follows (Xiafei H. et al., 2006, p.1-9):

1) Define the following:
   $L_r$ denotes the Laplacian Score for the $r$-th feature
   $f_{ri}$ denotes the $i$-th sample of the $r$-th feature, where $i = 1, \dots, n$.
2) Construct nearest neighbour graph $G$ with $n$ nodes. Here the $i$-th node corresponds to $\boldsymbol{x}_i$. If $\boldsymbol{x}_i$ is among the $k$ nearest neighbours of $\boldsymbol{x}_j$ or the other way around, the two nodes are determined to be 'close' and edge is put between nodes $i$ and $j$. If label information is known, an edge can be put between two nodes sharing the same label.
3) If nodes $i$ and $j$ are connected, set $S_{ij} = e^{\frac{\|x_i - x_j\|^2}{t}}$ where $t$ is a suitable value. Otherwise set $S_{ij} = 0$. The weight matrix $S$ of the graph models represents the local structure of the data space.
4) For the $r$-th feature, the following is defined:
   $\boldsymbol{f}_r = [f_{r1}, f_{r2}, \dots, f_{rn}]$,
   $D = diag(S\mathbf{1}), where\ 1 = [1, \dots, 1]^T$,

$L = D - S$, where the matrix $L$ is often called the graph Laplacian.

Let:

$$\widetilde{\boldsymbol{f}_r} = \boldsymbol{f}_r - \frac{\boldsymbol{f}_r^T D\mathbf{1}}{\mathbf{1}^T D\mathbf{1}}\mathbf{1} \quad (1)$$

5) The Laplacian score for the $r$-th feature can be calculated as follows:

$$L_r = \frac{\widetilde{\boldsymbol{f}_r^T} L \widetilde{\boldsymbol{f}_r}}{\widetilde{\boldsymbol{f}_r^T} D \widetilde{\boldsymbol{f}_r}} \quad (2)$$

As presented above, a weighted graph G is constructed with edges connecting nearby points to each other. $S_{ij}$ is used to evaluate the similarity between the $i$-th and $j$-th nodes. Therefore the significance of a good feature can be described as being the degree to which it follows the graph's structure. The number of neighbours ($k$) has an effect on the feature ranking: a larger $k$ will capture more of the global structure of the data. In experiments conducted by Xiaofai H. et al. (2006, p. 1-9), the Laplacian Score feature selection method performed notably better than using variance only as a feature selection method.

#### 2.1.3.3. *Spectral feature selection*

The Spectral Feature selection framework, introduced by introduced by Zhao Z. and Liu H. (2007, p. 1-7), is based on the spectral graph theory. It is suitable for both supervised and unsupervised feature selection. The importance of each feature is measured by its consistency with the structure of the graph is derived from its similarity matrix. A graph ($G$) is produced to represent and reflect a pairwise sample similarity set ($S$). Using graph theory, the spectrum of a graph can be used to gain structure information from it. Spectral feature selection investigates which features to select according to the graph structure induced from the pairwise sample similarity set. A feature that is consistent with the graph structure will assign values that are similar to the samples that are close to each other on the graph. If the feature $F$ regularly appoints values to samples that are consistent with the graph structure, the feature can be assumed to be relevant to the target concept. This means that the feature separates data effectively. The Spectral feature selection framework allows users to select various similarity matrix measures, $\gamma(\cdot)$, and ranking functions. Thus, it can be used to produce various Spectral Feature selection algorithms, applicable for both supervised and unsupervised machine learning tasks. An example of the Spectral feature selection algorithm

with a similarity matrix measure, $\gamma(\cdot)$, and ranking function ($\varphi(\cdot)$), selected for unsupervised learning can be presented in the following way (Zhao Z. & Liu H., 2007, p. 1-7):

1) Define the following:
   $X = (x_1, x_2, \dots, x_n)$ denote a dataset of $n$ samples,
   $F_1, F_2, \dots, F_r$ denote $r$ features,
   $\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_r$ denote the $r$ corresponding feature vectors,
   $W$ denotes the adjacency matrix,
   $D$ denotes the degree matrix,
   $L$ denotes the Laplacian matrix,
   $\mathcal{L}$ denotes the normalized Laplacian matrix.
2) Construct similarity set $S_{ij} = e^{\frac{\|x_i - x_j\|^2}{(2\delta^2)}}$ from $X$
3) Construct graph $G$ from similarity matrix $S$
4) Build $W$, $D$ and $\mathcal{L}$ from $G$, where:
   $W(i,j) = w_{ij}$
   $\begin{cases} D(i,j) = d_i, if\ i = j \\ 0 \end{cases}$, where $d_i = \sum_{k=1}^{n} w_{ik}$
   $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, where $L = D - W$
5) For each feature vector $\boldsymbol{f}_r$ do:

$$\boldsymbol{f_r} \leftarrow \frac{D^{\frac{1}{2}}\boldsymbol{f}_r}{\left\| D^{\frac{1}{2}}\boldsymbol{f}_r \right\|} \tag{3}$$

   $SF_{SPEC}(r) \leftarrow \varphi(F_r)$, where $\varphi(F_r) = \sum_{j=0}^{n-1} \alpha_j^2 \gamma(\lambda_j)$, where:
   $(\lambda_j, \varepsilon_j), 0 \le j \le n-1$ denotes the eigensystem of $\mathcal{L}$, and
   $\alpha_j = \cos\theta_j$, where $\theta_j$ is the angle between $\boldsymbol{f}_j$ and $\varepsilon_j$, and
   $\gamma(\mathcal{L}) = \sum_{j=0}^{n-1} \gamma(\lambda_j)\, \varepsilon_j \varepsilon_j^T$ denotes the ranking algorithm.
6) Rank $SF_{SPEC}$ in ascending order.

The algorithm returns the indices of the features in accordance to how well they performed. The first features returned are considered the best ones. In a study conducted by Zhao Z. & Liu H. (2007) it was noticed, that a high accuracy could be achieved by using certain com-

binations of similarity matrices and ranking functions when compared to other feature selection algorithms. It should be noted, that the Laplacian Score feature selection algorithm presented in this thesis can be thought of as being a special case of the Spectral feature selection algorithm. (Zhao Z. & Liu H., 2007, p. 1-7).

#### 2.1.3.4. *Multi-Cluster Feature Selection*

The Multi-Cluster Feature Selection (MCFS) algorithm, introduced by Cai D., Zhang C. & He X. (2010, p.333-341), is based on the idea that the features that preserve the multi-cluster structure of the dataset are best suited for unsupervised machine learning tasks. Unlike traditional unsupervised feature selection methods that select the best ranked features based only on specific scores calculated independently for each feature, the MCFS algorithm considers the possible correlations between different features. By using spectral analysis techniques, MCFS enables a reliable way to measure the correlations between various features without using label information. This enables MCFS to handle data with a multiple cluster structure. The aim of the algorithm is to select features, that best preserve the cluster structure of the data, and can 'cover' all of the possible clusters in the data. The MCFS algorithm can be described as follows (Cai D., Zhang C. & He X., 2010, p.333-341):

1) Define the following:
   $\boldsymbol{W}$ denotes the weight matrix,
   $\boldsymbol{D}$ denotes the diagonal matrix,
   $\boldsymbol{L}$ denotes the graph Laplacian ($\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$),
   $X = (x_1, x_2, \dots, x_n)$ denote a dataset of $n$ samples,
   $\boldsymbol{Y} = [\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_k}]$, where $\boldsymbol{y_k}$'s are the eigenvectors of the generalized eigen-problem,
   $\boldsymbol{a} = [a_1, a_2, \dots, a_k]$, where each $a_k$ represents sparse coefficient vector of a feature,
   $K$ denotes the intrinsic dimensionality of the data
2) Solve the generalized eigen-problem:
   $\boldsymbol{Ly} = \lambda \boldsymbol{Dy}$, where $\boldsymbol{Y}$ contains the top $K$ eigenvectors with respect to the smallest eigenvalues.
3) Solve $K$ L1-regularized regression problems:
   $\min_{\boldsymbol{a}_k} \|\boldsymbol{y}_k - \boldsymbol{X}^T a_k\|^2$, $|\boldsymbol{a}_k| = \sum_{j=1}^{M} |a_{k,j}|$ denotes the L1-norm of $\boldsymbol{a}_k$.

4) For every feature $r$, define the MCFS score for each feature as:

$$MCFS(j) = \max_k |a_{k,r}| \quad (3)$$

where $a_{k,r}$ is the $r$-th element of vector $a_k$.

5) Return the top $d$ features according to their MCFS scores.

It should be noted, that the MCFS method utilizes the *p*-nearest neighbour method to construct the graph, where the value of neighbours *p* can be varied. In experiments conducted by Cai D., Zhang C. & He X. (2010), it was noticed that MCFS outperformed other algorithms when looking at clustering accuracy. As the number of selected features increased, the clustering performance for all methods also increases, thus making the differences in performance between methods smaller. The MCFS method worked especially well compared to other features selection methods, when the number of selected features was small. (Cai D. et al., 2010, p.333-341).

### 2.1.4. Embedded methods

Embedded feature selection methods occur by utilizing the internal mechanisms of the classification algorithm (Pohjolainen J., Räsänen O. & Kadioglu S., 2013, p.4). They provide a trade-off solution between wrapper and filter methods by embedding feature selection into the model learning process, therefore inheriting benefits from both wrapper and filter feature selection methods. By embedding the feature selection process into the learning model, they inherently include interactions with the learning algorithm, while also being more effective than wrapper methods. This is due to the fact, that embedded algorithms don't have to evaluate feature sets iteratively. (Liu H. & Matada H., 2007, p. 10).

### 2.1.5. Feature selection for clustering

The increase of high-throughput technologies has led to considerable growth in harvested data in terms of sample size and feature dimensionality. Companies seek to find commercial value inside the pool of information available to them, with the aim of being able to leverage themselves over competitors, or give support for strategic decisions. (Macedo F., 2017, p.1). However, real world data contains a lot of irrelevant, redundant and noisy features (Li J., et al, 2016, p.2). This has led to an increased need for data storage and made further processing

of the data significantly harder. Dimension reduction techniques have become increasingly popular methods to remove uninformative or noisy data that cannot be used effectively. (Alelyani S. et al., 2013, p.1).

The dimensionality of data has increased drastically from the 1980s, with a notable increase in the 2000s. Datasets with very high dimensionality are relatively common nowadays. For example, eBay's data warehouse had 10 petabytes of data in 2010. (Zhao Z. & Liu H., 2011, p.109). Due to the huge amount of features being collected, learning models have a tendency to overfit and degrade learning performance. To combat this phenomenon, dimensionality reduction techniques, such as feature selection, are required. From a clustering point of view, removing irrelevant features will not have a negative effect on the clustering performance. However, the required amount of storage can decrease notably, and the computational time required to execute the task can drop drastically. (Alelyani S. et al., 2013, p.4-5). In addition, with a large number of features, it will be very difficult to understand the model itself, and extract useful knowledge from it, meaning that the interpretability of the model decreases. (Zhao Z. & Liu H., 2011, p.30).

For an unsupervised learning task, such as clustering, the user can opt to either conduct feature selection before or after clustering. Both options have advantages: if feature selection is applied before clustering, it can improve clustering quality significantly, whereas if feature selection is applied after clustering, the resulting clusters can be used as indicators of relevant features. As most feature selection methods are intended for supervised machine learning methods, better results might be obtained by conducting feature selection after clustering. Nevertheless, both options have their own drawbacks: if feature selection is applied before clustering, the optimal number of features might be difficult to select, having a notable effect on the clustering result. However, if feature selection is applied after clustering, it may be hard to select the optimal number of clusters to gain the best information on the most relevant features. (Alelyani S. et al., 2013, p.26).

## 2.2. Customer Analysis

Using data mining applications in data-intensive industries can provide good guidance for marketing strategies. Data mining is the process of extracting unknown and potentially valuable patterns from large amount of collected data. However, as the amount of data increases, traditional data analysis tools become unable to effectively provide decision makers with relevant knowledge and insights, needed for decision support. This often leads to a so-called 'wealth of data, poor knowledge' –phenomenon, where enormous amounts of data are available for decision makers to use, but no meaningful insight is gathered from the data due to the lack of tools to manage large datasets. (Qiuru C., et al., 2012, p. 1179-1182).

Treating customers as a principal asset has become an increasingly prominent trend for many organizations. Companies are investing large amounts into developing strategies for better customer acquisition, development and maintenance. Business intelligence derived from customers is gradually starting to play a more crucial role by enabling organizations to use technical expertise to achieve better customer insight for various outreach programmes. (Tripathi S., 2018, p.802). The goal of using data mining technologies is to obtain and utilize knowledge to be able to provide customers with better services and improve current commercial opportunities (Qiuru C. et al., 2012, p. 1179). This section discusses methods, previous research and literature related to customer segmentation and use of logged software use data to gain further insight on customer behaviour.

### 2.2.1. Customer Journey Mapping

Currently companies are using a wide assortment of methods to understand users and their experiences (Baesens B., 2017, p.39). Methods utilized include, for example, usability testing, interviewing and web analytics. The goal is often to improve the overall customer experience from software rollout to subscription renewal. Customer journey mapping is the act of mapping out and analysing a customer's life cycle experience when using a service. The journey is mapped all the way from the initial contact through the various steps of engagement until potentially landing into a long-term relationship. The goal is to understand the customer's motivations and expectations, which can help considerably when trying to figure out how to increase sales, uptake and product quality. (Baesens B., 2017, p.39). However,

without a proper journey mapping journey, the end result is often sub-optimal, resulting in inconsistent experiences, messages and approaches to service and support. Customer journey mapping provides an opportunity to look at the customer experience as a whole in a holistic way. (Fichter D., 2015, p.74).

One of the biggest benefits of using customer journey mapping is that it helps change the mentality of both developers and decision makers from an organization-centric mind-set to a more user-centric mind-set. The aim is to bring together both cross-functional and cross-departmental teams to identify and look at issues found, and try to find solutions to them. By looking at the issues from many different perspectives, the goal is to be able to prioritize development work and investments according to where it is most needed, thus improving the customer experience more efficiently and providing a service, that customers are happy to use. Once the map has been completed, it should be analysed for 'low-hanging fruits'. These are sections in the process, where relatively small and easy changes can eliminate an issue or simplify a process. (Fichter D., 2015, p.74-75).

Various ways and formats to present a customer journey exist, and no unified standard for the mapping process exists (Bernard G. & Andritsos P. 2017a, p. 49-51). However, there are some expressions and concepts that reoccur often, the four most essential ones being: touchpoint, channel, experience and barrier. A touchpoint describes the interaction between a customer and a company's product or services. A channel is a method chosen by a customer to interact with a touchpoint, such as website. An experience portrays the customer's feedback and emotions during the different phases and touchpoints of the journey. A barrier describes friction or discontent felt by the customer that could occur at some point in the process. (Bernard G. & Andritsos P. 2017a, p. 49-51).

A good customer journey map is data-driven and visual. To ensure that the customer journey map is evidence-driven, the mapping process usually starts by mining software use data by collecting data from, for example, web analytics, interviews and feedback forms. (Fichter D., 2015, p.75-76). In order to get a realistic idea of the current state of the journey and notice possible misconceptions, an expected journey map should be mapped out before map-

ping out the true journey. In this way, the data-based journey can be compared to the expected or intended journey, therefore making it easier to notice similarities, differences and possible 'low-hanging fruits'. (Bernard G. & Andritsos P., 2017b, p. 1-5).

As a matter of interest, web analytics have taken a more prominent role in many companies. For web analytics to be efficient, it must be ensured that all events are properly tracked across the various customer touchpoints. All customers should be uniquely identifiable across all touchpoints so that the corresponding information from each individual touchpoint can be correctly matched. This is done to secure a reliable flow of information. In general, it is recommended to capture all events as detailed as possible. During the subsequent analysis the complexity can be reduced by using appropriate aggregation operations. (Baesens B., 2017, p. 39-40).

An example of mapping the customer journey based on web analytics is clickstream analysis. Clickstream analysis is the act of recording and analysing actions and clicks of customers to understand how customers navigate through their application or website. This information can be used to investigate how long each customer spends on each page and when they drop off. This analysis can be further enrichened by performing segmentation on customers and considering how different customer segments may have different journeys within the same webpage or application. (Baesens B., 2017, p. 39-40). As a concrete example, 'The North Face', an outdoor gear company, was trying to figure out why their online shop customers were going to their checkout page but not completing their purchase. It was noticed via web analytics, that a promotional banner for a rewards program had been placed above their checkout button. This was causing customers to get distracted, and therefore not checking out and purchasing their products. The company ran an A/B test, where one had the rewards program banner, and the other had it removed. As a result, the checkout percentage for group that had the banner removed increased from 45% to 63%. (Lamont J., 2016, p. 9).

### 2.2.2. Logged Software Use Data

The opportunity to use logged software use data throughout a products life-cycle is still a largely unexplored area. The aim of collecting and analysing logged software use data is to identify potential problems and issues affecting user experience. (Väätäjä H., 2016, p.1-2).

This is often measured by user experience measurements that can include the use of a device function, access of features by different user groups or the identification of changes in the software use patterns (Ketola P. & Roto, V., 2009, p. 78-79). A common approach to acquiring software use data is to configure software applications to log user interaction events. In practise, there are two ways to implement the collection of logged data: either by adding software use data logging into the source code, often a laborious approach adding complexity into the system, or by applying semi-automatic logging of relevant user interactions. The level of abstraction and granularity of logged data should be taken into consideration, in order to get enough detail to be able to analyse desired actions, but still keep the data as relevant as possible. (Väätäjä H., 2016, p.2).

Once the log data has been collected, the following guidelines, as presented by Väätäjä H. (2016, p. 2), should be followed to transition from raw data to obtaining insights:

1) Preprocessing of data to transform it into a suitable format for data analysis
2) Verification, modelling and analysis of data
3) Reporting of analysis procedures and illustration of results.

Traditional descriptive statistics can provide a basic understanding of the data. However, typically some kind of data preprocessing methods need to be applied prior to conducting detailed analysis, especially when dealing with low-level events, such as mouse clicks on a website. Event logs without preprocessing are often noisy and incomplete (Rudnitckaia J., 2015, p. 7). The low-level events can then be mined and interpreted, with the goal of characterizing system use and detecting or comparing product use patterns (Väätäjä H., 2016, p.2). Timestamps together with action identifiers provide a natural ordering for events (Renaud K. & Gray P., 2004, p. 116). It is crucial for companies to highlight the value and benefits customers receive from sharing data with the company conducting the analytics, as permission is needed before utilizing the data. It is important to build a mutual trust between the customer and supplier company for data sharing by telling how and what the data is used for, and who can access the data for what purpose. (Väätäjä H., 2016, p.2-4).

The inspection and analysis of logged software use data has considerable benefits for both the company collecting and analysing the data, and the customer company. First off, the

supplier company, the main benefiter, can utilize the analysis results to better their research and development (R&D) and service business development activities. Information beneficial to R&D, such as analysis and visualization of feature use logs, could benefit system developers, and improve user experience (UX) and user interface (UI). By tracking user feature use, it would be possible to measure how old and new features are used and develop them to better serve the true needs of the customer. For example, it might be noticed that an important function is behind a 'hidden' button, that users are having a hard time to find. (Väätäjä H., 2016, p.5). However, establishing the context of actions has to be thought out with care. For example, if a user has two windows open, it might be difficult to figure out which one the user did actions on (Renaud K. & Gray P., 2004, p. 118).

The service business, such as system update services, can be developed to better fit the real needs of users. In addition, new ways to develop and offer product training can be introduced where, for example, certain activities that have been noticed to be troublesome for users can be concentrated on. Marketing investments can also be improved by targeting marketing campaigns on chosen groups, and advertising the product based on how users truly use it. The second benefiter is the customer itself buying the product.

Various visualization and other business intelligence related advances will become available, enabling customers to receive a better understanding of how the company uses the product itself. Companies often want to support their customer's efficiency and effectiveness when using their product. This can be accomplished by both improving and developing features, and by also showing and informing what is being done in order to better the product for the advantage of the customer. In other words, companies need to make sure that their customers understand what they are doing to improve the product and maintain a certain level of transparency to their action. (Väätäjä H., 2016, p.5).

### 2.2.3. Clustering

Clustering is an application-oriented form of explorative data mining technique used in many areas, such as machine learning, pattern recognition and classification. Clustering is typically an unsupervised machine learning method meaning, that cluster labels are unknown or not used (Canny J., 2018, p. 3). The main objective of clustering is to create groups from the

dataset, where group members have similar characteristics and properties within the same cluster, but are as different as possible from the members in other clusters. In other words, the goal is to create groups, where the members within a group are homogenous and the members of different groups are heterogeneous. (Kashwan K., 2013, p. 856-857). Clustering customers can be thought of as being one of the most useful techniques in business analytics for the analysis of customer behaviour. By clustering customers into homogenous groups, it is easier to identify unique segments of customers who think and operate differently and follow various approaches. Finding a suitable number of clusters, which are suitable as well as useful for analysis purposes is one of the main goals of clustering. The clustering task should be iterative and repeatable, where a significant amount of raw data is scanned for similarities and patterns. Patterns or knowledge are searched from the unorganized data, enabling the data points to be assigned. To achieve optimal results, various clustering algorithms along with differing parameters have to be experimented with for each individual market domain. (Tripathi S., 2018, p.802). Time intervals used for features in the context of business applications are usually months, seasons or years, depending on the type of business. This should be taken into account when selecting and creating features to use. (Ivett E., Nápoles G., García L. & Vanhoof K., 2018, p.1-2). A typical clustering process, as described by Qiuru C. et al (2012, p. 1179-1182), is presented below in figure 6:

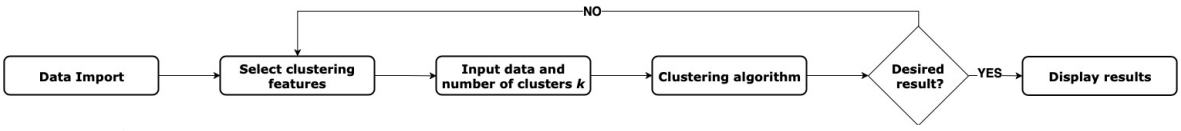


*Figure 6: Clustering process*

Clustering typically starts with the import of the given dataset. Usually, the structure of the dataset is inspected and potential outliers are scouted out. Other data preprocessing procedures may also have to be conducted. For example, if the scale of values in different features varies considerably, the data has to be normalized in order to obtain useful results. Once the data has been processed, the features for clustering are selected. The features can either be selected by the user based on intuition or prior knowledge, or by a feature selection algorithm. The selected features and the corresponding datapoints are then input into the clustering algorithm along with the desired number of clusters. The output results are then inspected, usually by looking at whether the clusters provide the desired knowledge from the

data, and by looking at some metric that measures the goodness of fit for the clusters. If the desired result isn't achieved, the input to the clustering algorithm is changed, either by choosing a different set of features or by varying the number of clusters. If the results are adequate, they are often further analysed and compared to previous and current knowledge and results. Ultimately, the goal is to display the results to a wide audience and be utilized to support and drive decision making. This thesis inspects and utilizes four different clustering algorithms: K-means, K-medoids, Fuzzy C-means and Hierarchical clustering. These methods are described in more detail later on in this thesis.

#### 2.2.3.1. *Customer Segmentation*

Customer segmentation can be defined as the classification of customers into different groups according to one or several features. By segmenting customers decision makers can perform different kinds of sectional analysis on current and future customers. The aim of customer segment analysis is to gain knowledge and a deeper understanding on, for example, the customers' overall composition, group characteristics of various valuable customers and loss customers, and the consumption characteristics of customers. If these goals are achieved, customers within target segments can be analysed further, enabling analysts to break down and better understand valuable customers. This, in turn, enables the implementation of a better personalized service and marketing for target customers, improving the success of other activities such as cross-selling other products. (Qiuru C. et al., 2012, p. 1179-1182).

Due to an increasing amount of data being collected and analysed, customer segment based marketing is becoming an increasingly important subject for consideration. By analysing large volumes of data collected from customers, businesses can improve their marketing decisions based on their customers' preferences. Clustering makes it possible to allocate customers to various groups, such as valuable and less valuable. By allocating available resources to promote the most useful and loyal customer segment, the profitability relative to used resources can be increased considerably. Tripathi S. (2018) presented a process for targeted marketing, that can be divided into three main steps:

1) Customers in the chosen market are segmented into distinct groups based on their characteristics
2) The segments are analysed for their properties and various ways in which different marketing strategies could be applied to each specific group
3) Studies and comparisons to competing brands and potentially their customers' behaviour can be conducted.

The process presented above should be used as a baseline framework before attempting segment based marketing. The results and potential success of various marketing strategies should be recorded and analysed continuously in order to achieve the desired results.

#### 2.2.3.2. *K-means*

MacQueen J. (1967, p. 281-297) introduced the K-means clustering algorithm. The algorithm partitions the data into a preselected number of sets, often annotated with the letter *k*. The solution is then a set of *k* centres, each of which is located at the centroids of the data clusters. For each data point, a membership value is assigned for each cluster. For K-means clustering, the following hard partition constraints exist:

1) $i = 1, 2, \dots, n; j = 1, 2, \dots, k; m(C_j|x_i) \epsilon \{0,1\}$
2) $\sum_{j=1}^{k} m(C_j|x_i) = 1$
3) $0 < \sum_{i=1}^{n} m(C_j|x_i) < n$, where

$i$ denotes the index of sample $x_i$ and $n$ the number of samples, and
$j$ denotes the index of cluster $C_j$ and $k$ the number of clusters, and
$m$ denotes the cluster membership value.

The hard partition constraints can be characterized in the following way:

1) The index $i$ of sample $x_i$ has to be between 1 and the number of samples $n$, the index $j$ of cluster $C_j$ has to be between 1 and the number of clusters $k$, and each sample $x_i$ has to have a cluster membership value of either 0 or 1.
2) The sum of cluster membership values for sample $x_i$ over all clusters $C_j$ has to be equal to 1. This means that sample $x_i$ has to be a member of a single cluster $C_j$.

3) The sum of membership values for cluster $C_j$ over all samples $x_i$ has to be greater than 0 but smaller than the number of samples $n$. This means that all clusters have to have at least one sample as a member, and that a single cluster cannot have all samples as members.

The objective function that the K-means algorithm optimizes is :

$$\min_{j\epsilon\{1..k\}} \sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i - c_j\right\|^2 \quad (4)$$

Where $x_i$ represents an individual datapoint belonging to cluster $C_j$ and $c_j$ the centre of cluster $C_j$. The K-means algorithm can be illustrated as follows:

1) Initialize the algorithm with random or chosen centres $c$.
2) For each data point $x_i$, compute its membership $m(C_j|x_i)$ for each centre $c_j$, where

$$m(C_j|x_i) = \begin{cases} 1;\ if\ \arg min_j \left\|x_i - c_j\right\|^2 \\ 0; otherwise \end{cases} \quad (5)$$

3) For each centre $c_j$, recompute its location from all data points $x_i$ that are assigned to that cluster:

$$c_j = \frac{\sum_{i=1}^{n} m(C_j|x_i)x_i}{\sum_{i=1}^{n} m(C_j|x_i)} \quad (6)$$

4) Repeat steps 2 and 3 until convergence is achieved or the condition for the maximum number of iterations is met.

K-means has a hard membership function and a constant weight function that gives all data points equal importance. The K-means algorithm is easy to implement and understand, therefore making it a popular algorithm for various clustering tasks. (Hamerly G. & Elkan C., 2002, p. 602).

#### 2.2.3.3. *Fuzzy C-means*

The Fuzzy C-means algorithm, introduced by Bezdek J. (1981, p 65-80), is an adaptation of the more traditional K-means algorithm. Unlike the K-means algorithm that uses a hard membership function, the Fuzzy C-means algorithm uses a soft membership function. This means that instead of each data point being assigned to its closest centre, the Fuzzy C-means

algorithm allows a datapoint to partly belong to multiple or all centres. For Fuzzy C-means clustering, the following fuzzy partition constraints exist:

1) $i = 1, 2, \dots, n; j = 1, 2, \dots, k; m(C_j|x_i) \epsilon [0,1]$
2) $\sum_{j=1}^{k} m(C_j|x_i) = 1$
3) $0 < \sum_{i=1}^{n} m(C_j|x_i) < n$, where
   $i$ denotes the index of sample $x_i$ and $n$ the number of samples, and
   $j$ denotes the index of cluster $C_j$ and $k$ the number of clusters, and
   $m$ denotes the cluster membership value.

The fuzzy partition constraints can be characterized in the following way:

1) The index $i$ of sample $x_i$ has to be between 1 and the number of samples $n$, the index $j$ of cluster $C_j$ has to be between 1 and the number of clusters $k$, and each sample $x_i$ has to have a cluster membership value between 0 and 1.
2) The sum of cluster membership values for sample $x_i$ over all clusters $C_j$ has to be equal to 1. This means that sample $x_i$ can be a member of several clusters $C_j$, as long as the total sum of cluster membership values over all clusters is equal to 1.
3) The sum of cluster membership values for cluster $C_j$ over all samples $x_i$ has to be greater than 0 but smaller than the number of samples $n$. This means that all clusters have to have at least one sample that have membership in it, and that there can't be one cluster that all samples are exclusively members of.

The objective function that the Fuzzy C-means algorithm optimizes can be described as:

$$\sum_{i=1}^{n} \sum_{j=1}^{k} m(C_j|x_i)^r \left\| x_i - c_j \right\|^2, \ 1 \leq r < \infty \qquad (7)$$

Where $x_i$ represents an individual datapoint, $c_j$ represents a cluster centre, and $r$ the fuzziness parameter. A value of $r = 2$ is often used for the fuzziness parameter. The Fuzzy C-means algorithm can be illustrated as follows:

1) Initialize the algorithm with random or selected centres $c$.
2) For each datapoint $x_i$, compute its membership $m(C_j|x_i)$ for each centre $c_j$, where

$$m(C_j|x_i) = \frac{\|x_i - c_j\|^{-2/(r-1)}}{\sum_{j=1}^{k}\|x_i - c_j\|^{-2/(r-1)}} \quad (8)$$

3) For each centre $c_j$, recompute its location from all data points $x_i$:

$$c_j = \frac{\sum_{i=1}^{n} m(C_j|x_i)x_i}{\sum_{i=1}^{n} m(C_j|x_i)} \quad (9)$$

4) Repeat steps 2 and 3 until convergence is achieved or the condition for the maximum number of iterations is met.

As stated before, Fuzzy C-means has a soft membership function and a constant weight function. As $r$ approaches 1 from above, the algorithm behaves more like standard K-means. Therefore, a larger value for $r$ makes the method more 'fuzzy' meaning, that the centres share more data points. (Saghafi L., 2017, p. 1).

#### 2.2.3.4. *K-medoids*

K-medoids clustering is also an adaption of the K-means algorithm introduced by Kaufman L. & Rousseeuw P. (1987, p. 405-416). Instead of using the cluster means as centroids, individual datapoints from the dataset called medoids are used. Since the method is based on the object located most centrally in a cluster, it responds less to outliers in comparison to K-means clustering. The objective function the algorithm optimizes can be characterized as:

$$\sum_{j=1}^{k}\sum_{i=1}^{n} \left\|x_i - c_j\right\|^2 \quad (10)$$

Where $x_i$ represents the individual datapoints belonging to the cluster $C_j$and $c_j$ the medoid of cluster $C_j$. The algorithm can be described as presented below:

1) Initialize the algorithm with random medoids $c$ from the given set of datapoints $X$.
2) For each data point $x_i$ that is not a medoid, calculate the distances to each medoid $c$ using any common distance metric, and associate the datapoint with its closest medoid.
3) For each medoid $c_j$ and each datapoint $x_i$ that is not a medoid:
    a. Switch $c_j$ and $x_i$ and re-associate each datapoint to the closest medoid. Recalculate the distances for all datapoints.

b. If sum of total distances is more than it was, undo step *a.*

4) Repeat steps 2 and 3 until no further improvement is achieved or the condition for the maximum number of iterations is met.

K-medoids has a hard membership function and a constant weight function that gives all data points equal importance. It should be noted that the initial medoid selection has a considerable influence on the speed and the outcome of the algorithm, and can often be optimized using a separate algorithm to initialize the medoids. (Park S., Jun C., 2009, p. 3336-3337).

#### 2.2.3.5. *Hierarchical clustering*

Hierarchical clustering is a clustering method introduced by Ward H. (1963, p. 236-244). It aims to build a hierarchy of clusters. Hierarchical clustering can be divided into two main approaches: agglomerative and divisive. The agglomerative method, also known as the bottom-up approach, starts with each observation being an individual cluster. These individual clusters are then merged into larger clusters. The divisive method, also known as the top-down approach, starts with all observations being in one cluster. The single cluster is then divided iteratively into smaller clusters. The agglomerative approach is more common for market research purposes. (Tripathi S., 2018, p. 805). An example of an agglomerative hierarchical clustering algorithm can be presented as follows (Ward H., 1963, p. 236-244):

1) Compute a proximity matrix using datapoints $x_i$ with a chosen measure of similarity
2) Set each datapoint $x_i$ to be a single cluster
3) Merge the two closest clusters and update the proximity matrix
4) Repeat step 3 until a single cluster remains.

There are several different methods for defining the similarity between methods, for example maximum dissimilarity, also known as complete linkage, or minimum dissimilarity, also known as single linkage (Greenacre M., 2008, p.11). It should be noted, that unlike the previous clustering methods introduced, the hierarchical clustering method doesn't have a cluster centre point, therefore making it challenging to directly compare the methods to each other. The cluster formation is usually visualized with a dendrogram, which portrays the

hierarchical organization of samples. Choosing the number of clusters can be done by selecting the maximum measure of distance, for example Euclidean distance, between individual clusters, and setting a cut-off line at that point. Hierarchical clustering is popular for market research purposes, because it can provide a visual representation of the end result of the different clusters. (Tripathi S., 2018, p. 805)

#### 2.2.3.6. *Clustering evaluation methods*

The basic principle of data clustering is to generate clusters, where objects within the same cluster are as similar to each other as possible, while objects from different clusters are as dissimilar from each other as possible. As there are many different kinds of algorithms for clustering, it can be difficult to compare and select the best method for the given dataset and objective. For this reason, clustering validity analysis is often conducted after the initial clustering algorithm has been run. This enables the user to compare both clustering methods to each other and select the optimal amount of clusters for the dataset. (Thinsungnoen T. et al., 2015, p. 45). Four methods for evaluating the goodness of clusters are used in this thesis: Within-Cluster Sum of Squares, the Silhouette Score method, the Calinski-Harabasz Index and the Davies Bouldin Index.

The *Sum of Squared Error* is one of the most known and common measures for evaluating the goodness of a clustering task. The error for each individual datapoint is the distance from the point to the nearest cluster centre. To get the Sum of Squared Error metric for each datapoint in relation to the cluster assigned to it, the *Within-Cluster Sum of Squares* was invented. The Within-Cluster Sum of Squares metric can be calculated by squaring the errors and summing them together for all clusters, as portrayed below (Whittle P., 1983, p. 125-127):

$$WCSS = \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| x_i - c_j \right\|^2 \qquad (11)$$

where $c_j$ represents the cluster centre for cluster $C_j$, and

$x_i$ a datapoint assigned to cluster $C_j$.

The smaller the Within-Cluster Sum of Squares, the better the result of the clustering for a given number of clusters. The result helps the user to select the most efficient clustering

method and optimal number of clusters. However, one disadvantage to using the Within-Cluster Sum of Squares is, that the value of the measure often decreases as the number of clusters is increased, making it difficult to select the optimal number of clusters. Therefore, it should be remembered that a good clustering method with a smaller number of clusters can have a lower Within-Cluster Sum of Squares metric than a poor clustering with a higher number of clusters. (Zhang J., 2004, p. 25).

The *Silhouette Score* was introduced by Rousseeuw P. (1987, p. 53-65). It is a method to display the goodness of a clustering algorithm's performance. Each cluster can be represented by a 'silhouette': a visual presentation that is based on the comparison of its separation and tightness. Each silhouette shows which objects lie well within their designated cluster, and which ones are only somewhere in between different clusters. The clustering as a whole is often presented by combining the silhouettes into a single plot, enabling the evaluation of the comparative quality of clusters and a general view of the formation of the data. The mean silhouette width measurement can be used as a basis for clustering validity evaluation, and can be used to select the optimal number of clusters. The silhouette score for datapoint $x_i$ can be calculated as follows (Rousseeuw P., 1987, p. 53-65):

$$S_i = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}} \tag{12}$$

where $S_i \epsilon [-1,1]$ ,

$a(x_i)$ denotes the average dissimilarity of object $x_i$ to all other objects within its designated cluster A, and

$b(x_i)$ denotes the smallest average dissimilarity of object $x_i$ to objects in other clusters.

Dissimilarity between different objects is usually calculated by using a measure of distance. A high silhouette value indicates that the datapoint $x_i$ is well-matched within its own cluster and poorly matched to other clusters. (Rousseeuw P., 1987, p. 53-65). A high Silhouette score is sought after, because it promotes qualities desirable in clustering, therefore giving support for deciding the optimal number of clusters and evaluating different clustering methods. The final goodness of clustering metric used to compare different clustering results to each other is usually either the sum or average of silhouette values over all datapoints.

First introduced by Calinski T. & Harabasz J. (1974, p. 1-27), the *Calinski-Harabasz Index* evaluates the cluster validity based on the average within-cluster sum of squares and between-group sum of squares. The index measures the compactness of clusters based on the sum of distances between objects and their cluster centres and cluster separation based on the maximum distance between cluster centres. The index can be characterized as follows:

$$C = \frac{n-k}{k-1} \cdot \frac{BGSS}{WCSS} \tag{13}$$

where BGSS depicts the between-group sum of squares:

$$BGSS = \sum_{j=1}^{k} n_j \cdot \left\| c_j - c_{ALL} \right\|^2 \tag{14}$$

WCSS depicts the within-cluster sum of squares:

$$WCSS = \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| x_i - c_j \right\|^2 \tag{11}$$

$c_{ALL}$ depicts the centroid for all datapoints,

$k$ depicts the number of groups or clusters,

$n$ depicts the number of observations, and

$n_j$ depicts the number of observations for cluster $C_j$.

The higher the Calinski-Harabasz Index, the better the separability between the clusters. This is also logical by looking at the equation presented above. We want the between-group sum of squares to be as large as possible, and the within-group sum of squares to be as small as possible. The method also punishes the use of excess clusters. (Liu Y., Zhongmou L., Hui X., Xuedong G. & Junjie W., 2010, p. 912).

The *Davies Bouldin Index* is based on measuring the intra-cluster and between-cluster distances. The objective is to identify clusters, that are far away from each other and compact in composition. David Davis and Donald Bouldin (1979, p. 224-227) presented the index for the first time in their paper as follows:

$$DB = \frac{1}{k} \sum_{j=1}^{k} \max_{i=1,\dots,k, i \neq j} \left( \frac{\delta_i + \delta_j}{\Delta ij} \right) \tag{15}$$

$k$ denotes the number of clusters,

$\delta_j$ denotes the diameter of a cluster as follows:

$$\delta_j = \sqrt{\frac{1}{n_j}\sum_{i=1}^{n_j}\left\|x_i - c_j\right\|^2} \quad (16)$$

where $c_j$ denotes the centre for cluster $C_j$, and

$x_i$ denotes a datapoint assigned to cluster $C_j$,

and $n_j$ denotes the number of datapoints assigned to cluster $C_j$,

$\Delta ij$ denotes the distance between the cluster centres $c_i$ and $c_j$ as follows:

$$\Delta ij = \left\|c_i - c_j\right\|^2 \quad (17)$$

The smaller the Davies Bouldin Index, the better the result of the clustering. This is due to the main objective of clustering: to obtain clusters, that are compact and have the smallest intra-cluster distances as possible, while being as far away, or different, from each other as possible. (Saitta, S., Raphael, B. & Smith I, 2008, p. 4).

# 3. Research setting

The following section describes a case study conducted for a large software technology company carried out by utilizing the methods and themes introduced previously in this thesis. First, the approach is described on a general level to give the reader an overview of the methods used, after which a broad summary of the various tools and commercial applications used are presented. The approach is then dissected into three main more detailed parts: data preprocessing, customer segmentation and reporting. Each part describes the methods used, results and findings obtained from the research.

## 3.1. Approach

This segment describes the general approach taken for the study. Tools and general practises used within the case company were implemented when possible. One of main objectives of the case study was to figure out whether the data being collected could be utilized for analytics. A detailed study would then be conducted from the data deemed suitable for analytics, with the objective of finding out how the software was being used. The goal of the project was to be able to group users according to their software use patterns. To achieve the desired outcome, the main research and analysis can be divided into three main topics:

1) Data preprocessing
2) Customer segmentation
3) Visualization and reporting

*Data preprocessing* is always relevant when conducting analysis on a new set of data. Although the data in this case was not 'new', little to no analytics had been conducted on it. This meant that it had to be treated as a new dataset. The structure and features of the data were unknown, and therefore had to be explored. Exploration was conducted by experimenting with various methods and trying to model the dataset with distinct structures. Different visual and statistical experiments were also conducted during the exploration phase. Due to the fact that most of the data was historical or event-based and came from many different sources, new features had to be derived and created. Since features in the data came from

various sources and contained different types of data, different methods to evaluate the goodness and usefulness of different features across different datasets had to be used. Feature selection was an essential component in evaluating the goodness of different features.

*Customer segmentation* is a widely used and well-known data analytics task. However, software use data usually contains very basic features such as age and gender. The data used in this thesis contained both historically logged software use data and event data from various sources. The nature of the data made it more complex, therefore making it challenging to gain insights from the data using traditional algorithms. In addition, the sizeable amount of data caused some algorithms not to run at all. Both methods found from previous literature and practical experimentation were utilized to gain a better understanding of the customer segments.

*Behavioural data analytics* refers to data analytics that aims to find new insights from the large amount of raw event data, that is generated from various applications (Yamaguchi K., 2013, p.1). Event-based data refers to data that is related to user actions within the portal, such as procedures completed, navigation, use of features, visit length, and visit frequency. Behavioural data analytics is a branch of business analytics, a field of study focused on the use of various methods and technologies to investigate and employ diverse data analytics methods to understand business performance and drive business planning based on data (Pratt M & White S., 2019, p. 1). The aim of behavioural data analytics is to find patterns between features from within the data, that might at first seem unrelated or completely random. Patterns found can be used to map out what a typical visit for a user in the portal might look like for different users, and group or differentiate users from each other accordingly. Methods and approaches typically used for behavioural data analytics were utilized to support and enhance customer segmentation.

*Data visualization and reporting* is one of the most important steps for every successful data analysis task. Despite not containing analysis tasks in itself, a good and clear visualization and report or reporting platform is often required in order to influence the intended audience. Different methods utilized and evaluated as useful for visualization tasks were explored and noted. A tool used to create dashboards for different audiences was also investigated and experimented with.

## 3.2. Software and methodology used

This section discusses the various software and applications used to explore, preprocess, analyse and visualize data. The aim of this section is to give the reader an understanding of the various methods used and why they were used.

### 3.2.1. Python

Python is an interpreted open-source general-purpose programming language. Python was first released in 1991, and can be regarded as a successor to the ABC programming language. Python has a simple approach to object-oriented programming and offers efficient high-level data structures. Python's design philosophy emphasizes code readability with its distinguished use of whitespaces. Due to the previously mentioned features, Python is often credited as being easy to learn, making it an ideal language for scripting and fast application development for many different use cases and platforms. The Python interpreter can also be extended with new functions and data types implemented in both C, C++ and other languages callable from C. Python can also be used as an extension language for customizable applications. (Python Software Foundation, 2019, p.1).

#### 3.2.1.1. *Python for data analytics*

As time has moved on, Python has become one of the most popular dynamic programming languages. The adoption of Python for scientific computing tasks in both academic research and industry applications has increased notably since the early 2000s. In regards to data analysis and visualization, Python will often be compared to other domain-specific programming languages such as R, MATLAB and SAS. However, due to Python's strong and continuously improving library support, Python has become a solid alternative for various data manipulation and analysis tasks. In many organizations, it is common to research, prototype and test new idea first using a more domain-specific computing language, such as R or MATLAB, and later port the ideas to be part of a larger production system written in another programming language. Due to Python's nature as a programming language, it is often noted to be effective not only for research and prototyping, but also for building production systems. By combining Python's data manipulation and analysis libraries with its potential as a

general purpose programming language, it has become a prominent choice as a single language for creating many data-centric applications. (McKinney W., 2013, p. 2-3).

#### 3.2.1.2. *Software packages*

Python has a number of software packages with useful functionalities for data manipulation and analytics. The software packages used to carry out the project presented in this paper are shortly introduced in this section.

*SciPy* is a Python-based ecosystem of open-source software. It contains open-source software packages for various use cases, such as mathematics, science and engineering. SciPy can be considered as a core package for many data manipulation and analytics activities. (SciPy Developers, 2019a, p.1). *NumPy* is the fundamental package for scientific computing with Python. It contains functionalities such as a powerful N-dimensional array object, linear algebra, Fourier transform and random number generation capabilities. (SciPy Developers, 2019b, p.1). *Pandas* is a Python library providing easy-to-use data structures and data analysis tools. It enables users to potentially carry out the entire data analysis workflow in Python without having to switch to a more domain specific language. However, pandas doesn't implement significant modelling functionality outside of linear and panel regression. (SciPy Developers, 2019c, p.1). *Matplotlib* is a 2D plotting library for Python, that produces figures in a variety of hardcopy formats and interactive environments for multiple platforms. Matplotlib is based around the idea of keeping things simple; plots, histograms, scatterplots and many other visualizations can be produced with only a few lines of code. The library can be used for simple plotting, which is similar to the MATLAB style, but also has the capability to give the user full control of the style and layout of the plot. (SciPy Developers, 2019d, p.1).

*Scikit-learn* is a Python library with simple and efficient tools for data mining and data analysis. The library is built on NumPy, SciPy and matplotlib. Scikit-learn provides the means to perform various data analytics tasks, such as dimensionality reduction, data preprocessing, model selection, classification, regression analysis and clustering. (Pedregosa et al., 2011b., p.1). *Scikit-feature* is a feature selection repository for Python. It is built on the widely-used data analytics and machine learning packages scikit-learn, NumPy and SciPy. Scikit-feature

contains over 40 popular feature selection algorithms, and therefore is often used as a platform for facilitating feature selection application, research and comparative study. (Li, J et al., 2019, p.1).

*Apache Spark* is written in the Scala programming language. In order to support the collaboration of Apache Spark and Python, *PySpark* was created. In practise, the PySpark library is a Python API for Spark. The library takes advantage of the *Py4J* library to allow Python to dynamically interface with JVM objects. PySpark contains a number of libraries to help write efficient programs. In terms of data analytics and machine learning, two libraries stand out above others: PySparkSQL and MLlib. PySparkSQL allows users to apply SQL-like analysis on a huge quantity of structure or semi-structured data. MLlib is a wrapper over PySpark and operates as Spark's machine learning library. It uses the data parallelism technique to efficiently store and work with large amount of data. (Databricks, 2019, p.1).

#### 3.2.1.3. *JupyterLab*

*JupyterLab* is an interactive development environment for Jupyter notebooks, code and data born out of the IPython project. JupyterLab is web-based, extendable, modular and it allows users to configure and arrange the user interface to complement a wide variety of workflows for data science, scientific computing and machine learning. *Jupyter Notebook* is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations and explanatory text. It is especially useful in the case of data cleaning and transformation, numerical simulation, statistical modelling, data visualization and machine learning. Jupyter supports over 40 programming languages, such as Python. It also has the capability to be integrated into big data systems and leverage big data tools, such as Apache Spark. (Project Jupyter, 2019, p. 1)

### 3.2.2. Amazon Web Services

Amazon Web Services (AWS) is a cloud platform offering over 165 services from data centres globally. It consists of over 22 servers within different geographic locations. (Amazon, 2019a, p.1). *Amazon Elastic Compute Cloud* (Amazon EC2) is a web service, that provides resizable computer capacity in the cloud. It provides users control of computing resources,

making it possible to scale capacity both up and down, as computing requirements change. (Amazon, 2019b, p.1). Amazon Simple Storage Service (Amazon S3) is an object storage service. It offers query-in-places services, making it easier and faster to run database queries, and services to make the collection and analysis of data simpler. These features ease the use of big data analytics. (Amazon, 2019c, p.1). *Amazon Redshift* is a cloud data warehouse service. Redshift can be fully integrated and query data from data lakes without having to move or transform it, therefore making it a useful companion with services such as Amazon S3. Queries can be performed on the data lake alone or in combination with the data warehouse. Redshift provides an efficient way to build modern data analytics tools. (Amazon, 2019d, p.1).

### 3.2.3. Tools for parallel computing

The Apache Hadoop software library is a framework that enables distributed processing of large data sets across multiple clusters of computers with the help of programming models. With the goal of delivering high-availability, the library is designed to detect and handle failures at the application layer instead of relying on hardware, thus delivering a service on top of a cluster of computers. (Apache Software Foundation, 2019a, p.1). *Apache Spark* is a unified engine for large-scale data processing. Spark aims to achieve high performance for both batch and streaming data by using a unique DAG scheduler, query optimizer and a physical execution engine. Spark can be run on multiple different platforms, such as Apache Hadoop and Amazon EC2. (Apache Software Foundation, 2019b, p.1). *Azkaban* is an open-source batch workflow job scheduler created to run Hadoop jobs. It resolves the issue of ordering jobs by building job dependencies and provides an a web user interface to maintain and track workflows. (Azkaban, 2019, p. 1).

### 3.2.4. Redash

Redash is an open-source tool built for querying and visualizing data. Redash can be connected to most data sources. The goal of Redash is to make it easy to query data from various data sources, and create dashboards that can be shared within the organization. Redash was originally created as an alternative to more traditional business intelligence suits, with the ideology of creating a simple and technically easy solution. Redash's features a query editor,

tools to visualize the query, an editor to combine multiple visualizations into one dashboard, and the ability to send a notification via a selected channel when a query's results requires the attention of a single or group of users. (redash.io, 2019, p.1)

### 3.2.5. Orange

Orange is an open-source data mining and machine learning suite for data analysis. Orange utilises visual programming and Python scripting to achieve interactive data analysis and various components for data mining and analysis procedures. The Orange library is a toolbox containing hierarchically organized components for data mining. The aim is to make it easier for developers to add functionalities to any level of the hierarchy and combine it with the current existing code. The main branches of the hierarchy are: data management and preprocessing, regression, classification, ensembles, association, clustering, evaluation and visualizations. The Orange library's objective is to simplify the construction of data analysis workflows and creation of data mining approaches by offering a range of components that can be used alone or in combination with each other. (Demšar J. et al., 2013, p. 2349-2351). In other words, Orange provides a quick and easy interactive approach to data mining, machine learning and data visualization to explore and experiment with data.

## 3.3. Data preprocessing

The section below discusses the various steps taken to preprocess the dataset. First, different exploratory data analysis approaches were used to get a thorough understanding of what the data from different sources consists of. Once a satisfactory level of understanding was achieved, multiple procedures for data preprocessing, aggregating and combining were experimented with and evaluated.

### 3.3.1. Datasets

The data used in this study came from six different sources. The datasets were named according to their source: Activity, CRM, Operations, Portal, Subscription and Updater. The individual samples in the dataset were all related to individual companies. Table 1 below describes the different datasets.

| Dataset name | Data type | Data types | Number of features |
|---|---|---|---|
| Activity | Event | Numerical, categorical | 27 |
| CRM | Historical | Numerical, categorical, string, date | 49 |
| Operations | Historical | Numerical, categorical, string, date | 12 |
| Portal | Historical | Numerical, categorical, string, date | 101 |
| Subscription | Event | Numerical, categorical, string, date | 12 |
| Updater | Event | Numerical, categorical, string, date | 12 |

*Table 1: Description of datasets*

The historical data type means, in this context, that the dataset describes the state of the features for each sample at the exact moment of time when the data is queried and sent to the database. The usual period for querying was once a day, meaning that daily historical data was available for each company. On the other hand, the event data type logs actions and events executed by users. The data is queried and sent to the database on a daily basis as well. However, if a company hadn't executed an action or event during the time period between the database queries, there was no record of them for that time period.

The activity dataset contained data related to the overall activity of a company's devices, such as number of active devices. The CRM dataset comprised of data from customer relationship manager, such as price of sale. The Operations dataset held data related to actions or operations executed on companies by users, such as sending an update. The Portal dataset contained data related to user visits in the software portal and individual actions conducted during that visit, such as clicking on an icon. The Subscription dataset consisted of data related to the subscription and subscription status for companies, such as number of licenses sold. The Updater dataset comprised of data related to individual updates sent, such as name and version of update.

### 3.3.2. Data processing

There are many components, that have an effect on the success of a machine learning task. Sample representation and data quality are often considered as the two most prominent components. The presence of irrelevant and redundant data, as well as noisy or unreliable data can cause the insight gained to be poor in quality or even misleading. Data preprocessing consists of many different phases, that include among others data transformation, cleaning, normalization and feature selection. The end product of data preprocessing is the final dataset to be used for analysis and machine learning model creation. (Kotsiantis S., Kanellopoulos D. & Pintelas P., 2006, p. 111-112).

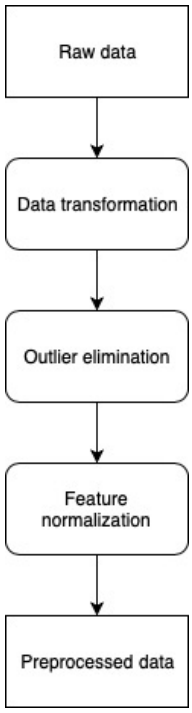


*Figure 7: Process for data preprocessing*

Figure 7 above represents the process that was selected to preprocess the datasets described previously. The process can be described as follows: initially the data is transformed into a more suitable form, after which outliers are detected and removed, followed by normalizing the numerical features, ending up with the final dataset. The rest of this section describes the process in more detail.

#### 3.3.2.1. *Inspection of raw data*

Initially, the data preprocessing task began with the *inspection of raw data*. Initial data analysis was conducted on each dataset to gain a basic idea of the different distributions and correlations within each dataset. This task was especially important for the datasets containing event data; historical data could be used rather effectively without major changes, whereas event data had to be grouped and aggregated in order to be useful. Different possibilities for data transformation were experimented with and evaluated. Unnecessary features, such as duplicate features or features that didn't contain any useful information content such as database-specific transaction identifiers, were identified and taken note of. Various identifiers and features connecting samples from one dataset to another were recognized. Orange was then used to visually explore the data by looking at, for example, the different distributions and correlations between features found within each individual dataset. As an example, table 2 below depicts the Pearson correlation between two different features in Orange.

| **Variable 1** | **Variable 2** | **Correlation** |
|---|---|---|
| company_free_seats | company_seats | +0.91 |
| company_active_days_per_admin | company_active_days | +0.80 |
| company_active_clients | company_seats | +0.61 |

*Table 2: Pearson correlation between features*

#### 3.3.2.2. *Data transformation*

Once an adequate level of understanding for each dataset was achieved, the data was *transformed*. The datasets needed to be combined into one large comprehensible form, that enabled easy manipulation and analysis of the dataset as a whole. By using the knowledge gathered from previous data exploration, a time period of one month was selected to be used as a basis for the transformed dataset. In practise, this meant that the monthly averages were taken for the historic datasets, whereas for event based datasets, all events within the given month were taken into account. A simplified example (figure 8) of event entry aggregation and transformation can be seen below.

| Initial event entries | | | | | | | |
|---|---|---|---|---|---|---|---|
| admin_id | 43a45d | device_id | 8a9361ba | command_type | SEND_UPDATE | created_date | 10/06/2019 |
| admin_id | 43a45d | device_id | a0642b1a | command_type | SEND_UPDATE | created_date | 10/06/2019 |
| admin_id | 43a45d | device_id | 238467d4 | command_type | SEND_UPDATE | created_date | 10/06/2019 |
| admin_id | 43a45d | device_id | c6be5bc2 | command_type | SEND_UPDATE | created_date | 10/06/2019 |
| admin_id | 43a45d | device_id | c6be5bc2 | command_type | SEND_UPDATE | created_date | 10/06/2019 |

**Transformed to:**

| company | admin_id | device_count | action_count | command_type | date |
|---|---|---|---|---|---|
| 02gk0sp | 43a45d | 4 | 5 | SEND_UPDATE | 6 |

*Figure 8: Data transformation*

The data transformation was conducted primarily in Amazon Web Services by utilizing PySpark in EC2 instances. However, not all data samples were available in Amazon Web Services S3 buckets. Therefore, some of the data transformation had to be conducted on a local computer. Local transformation was conducted utilizing the Python libraries pandas and NumPy. The data was then combined and downloaded onto a local computer to be uploaded to the cloud as a whole. This made the combined dataset available for both local and cloud computing, depending on the computing requirements of the task.

The transformed dataset essentially contained summaries for the amounts of each action conducted in every database during the set time period. If the same user sent updates to more individual devices or during numerous separate times during the time period, these entries would be added to the transformed dataset. Duplicate features and features with no information content identified in previous examinations were removed. The date feature was made to represent the number of the month at the time the action was completed.

The goal of the transformation process was to keep the data as granular as possible, while still making features useful for customer segmenting and clustering. A challenge was to figure out how to join data with differing levels of granularity. The main decline in granularity was the loss of ordering for the events. The end product of the data transformation process was a large data frame containing features from all databases. A value of zero was set for event-based features for the companies that hadn't conducted any actions during the time period.

#### 3.3.2.3. *Outlier elimination*

Two different methods were experimented with to *eliminate outliers*: Z-Score and removing a predefined percentile. Z-Score is a measure used in statistics to demonstrate a value's relationship to the mean of a group of values. If the Z-Score is 0, the data point's Z-Score is equal to the mean score, whereas a Z-Score of 1 indicates a value that is one standard deviation above the mean value. Since the aim is to eliminate outliers, an upper and lower limit for the Z-Score was set. For each sample the Z-Score was calculated. All samples, that had a feature with a Z-Score over the set upper limit or below the set lower limit were removed. This ensured, that samples with features differing notably from the mean were eliminated. Removing a predetermined percentile of samples functions similarly to the Z-Score method. The method removes all samples containing a feature with a value within the chosen upper percentile or within the chosen the lower percentile.

Outlier elimination was conducted using the Python libraries NumPy and SciPy. After experimenting with both methods, removing samples containing features with values within the top 0.5% percentile of all numerical values was chosen. The reasoning for this was that due to the data transformation process a relatively large amount of all samples had features with a value of zero. This meant that removing values within the bottom percentile wasn't reasonable, and standard deviation was an untruthful measure for some features, making the Z-Score an ineffective method. In addition, it was noticed during previous data exploration, that most outliers in the data had values in the upper percentiles.

#### 3.3.2.4. *Feature normalization*

Two methods were investigated and evaluated for *feature normalization*: Min-Max scaling and Z-Score scaling. The idea of Min-Max normalization is to scale all values to be in between 0 and 1. For each feature, the maximum value was scaled to be 1 and the minimum value was scaled to be 0. The scaled Z-Score metric is calculated by subtracting the mean from the sample's feature value and dividing the result by the standard deviation. The mean and standard deviation values were calculated for each feature.

The Python library scikit-learn was used to conduct the feature selection process. After inspecting the resulting datasets from both feature normalization methods, Min-Max scaling was chosen as the preferred method. The reason was similar as for the choice of the outlier elimination method: a large portion of samples had features with a value of zero, making the values for both the mean and standard deviation too small. Due to the Z-Score method relying on both mean and standard deviation, it was deemed ineffective at standardizing the dataset.

### 3.3.3. Data exploration

Data exploration is the process of efficiently extracting knowledge from data, even when the exact goal or knowledge wanted isn't known beforehand. Exploratory data analytics can be thought of as active knowledge seeking, with a strong emphasis on the discovery of unexpected knowledge. The aim is to isolate patterns and features from the data and reveal them to the analyst. Exploratory data analytics does not necessarily need to take probability, significance or confidence into account. It is more of an ideology for the desire to learn more about the data. (Tukey J., 1993., p. 530-531).

Data exploration was conducted on the preprocessed data combined from all six sources. Initial exploration was performed using Orange, after which the Python libraries pandas, NumPy and matplotlib were used to further analyse and visualize the results. The aim of the data exploration conducted in this thesis was to get a baseline idea of what kinds of customers are using the product and what kinds of features are being used. The features were also investigated in terms of how well they could describe different samples and whether meaningful customer segments and journeys could be constructed from them.

### 3.3.4. Feature selection

Feature selection is a crucial component often needed to successfully implement machine learning methods and create a machine learning model. Especially in samples that contain a large number of features, the necessity of feature selection is emphasized. Due to the project having over 200 potential features, it was decided to invest a considerable amount of work-

ing hours into feature selection. The feature selection process used for this project was divided into five main components, as portrayed by figure 9 below. Initially a literature review was conducted while investigating for relevant Python libraries in parallel, after which the identified algorithms were tested on the datasets. The results were then noted and prepared for further use.

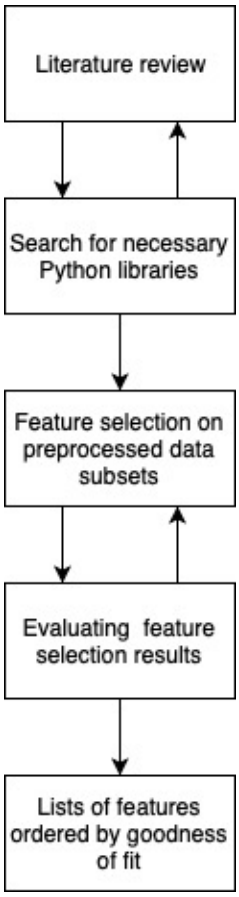


*Figure 9: Feature selection process*

The feature selection process was started by conducting a thorough literature review. Various feature selection approaches, such as wrapper, filter and embedded methods were investigated and evaluated. For example, methods that were only applicable to labelled datasets were excluded. Once potential feature selection algorithms had been identified, possible Python libraries containing the desired algorithms were searched for. If no library containing the required algorithm was found, the required effort to write and implement the algorithm were estimated and compared to the potential gained benefit. This process was repeated, until a satisfactory collection of feature selection algorithms and methods was collected. In

the end, no algorithms were implemented from scratch due to vast Python feature selection libraries available.

The next step of the feature selection process was to apply the feature selection algorithms on the dataset. All feature selection algorithms were applied on both the whole dataset and smaller subsets of data varying in both number of features and number of samples. For all feature selection methods that provided results, the algorithm was applied multiple times on the same dataset and the algorithm's results were noted for each iteration. The similarity for each iteration of the algorithm was noted. Each algorithm was then assessed according to how consistently they ranked the goodness of each feature. A local computer and EC2 samples were used to run the algorithms.

Finally, once the results from multiple iterations with various datasets and different algorithms had been evaluated, the most consistent algorithms were chosen. For each algorithm, the features were ordered from best to worst according to their performance given by the algorithm.

## 3.4. Customer segmentation

One of the main objectives of the project was to identify different customer segments based on use data collected from customers. Due to the large number of features and complexity of data, traditional segmentation based on, for example, company size was deemed not effective. This led to the decision that unsupervised machine learning methods were to be investigated. After researching various unsupervised machine learning frameworks, it was decided that clustering would most likely fulfil the intended purpose and produce the best results. A process was created to study and compare various clustering algorithms. In addition to selecting the best clustering algorithm for the task, different feature selection methods were also taken into account. The process has been visualized below in figure 10. The goal of the process was to identify the best combination of a feature selection method and a clustering method from the previously researched approaches.

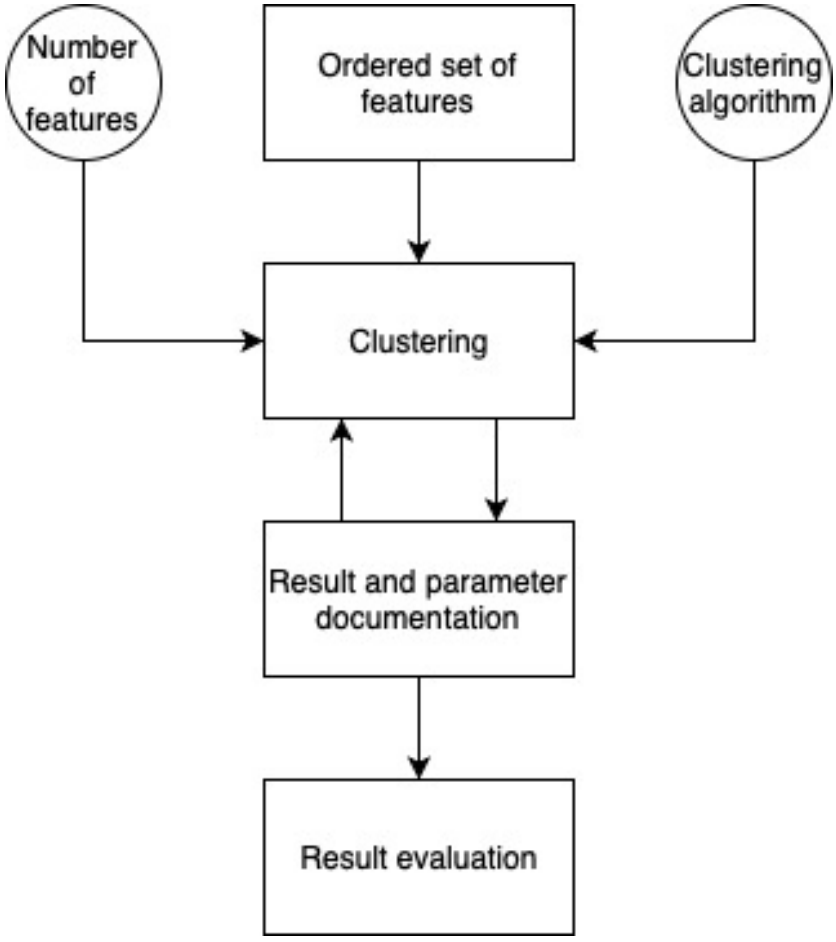


*Figure 10: Clustering evaluation process*

The evaluation process started with selecting a feature selection algorithm. The features were then ordered according to how the algorithm ranked them. Four clustering algorithms were chosen to be examined in the experiment: K-means, K-medoids, Fuzzy C-means and Hierarchical clustering. The experiment was then conducted in the following way: a fixed number of best performing features were input into the four previously stated clustering algorithms. Based on observations made using the 'elbow' -method on the whole dataset, the optimal number of clusters for each clustering algorithm was deemed to be between 2 and 6. Therefore each clustering algorithm was run with 2 to 6 clusters in order to get information on whether the clustering result improved or deteriorated significantly as the number of clusters was increased. Multiple iterations were run with the same number of features and parameters, in order to get data on the stability and robustness of the results. The results and the parameters for each iteration were documented. Once the predetermined number of iterations had been run, the number of features was increased and the algorithms run again.

After the previously described evaluation process had been run, the results were documented and summarized. The documentation contained information on the number of clusters, location of cluster centroids and metrics produced by four different clustering evaluation methods: Within-Cluster Sum of Squares, Silhouette score, Calinski-Harabasz index and Davies Bouldin index. Once the documentation was at a satisfactory level to draw conclusions from it, the evaluation process was run again. However, the feature selection algorithm was changed, and therefore the ranking of each feature was updated accordingly. The results of the clustering evaluation process with new features were documented for all of the previously selected feature selection algorithms.

Once the evaluation process had been run for all of the feature selection algorithms, the results were analysed. The results were evaluated quantitatively in the following ways: the cluster stability was tested by comparing cluster centroids for different iterations to each other, and the results for the clustering evaluation methods were compared. A qualitative evaluation for cluster separability was also administered by evaluated how well the project group could create easily understandable groups based on the cluster descriptive data.

## 3.5. Visualization and reporting

Visualization was especially important for the successful execution of the project. Due to no previous research having been conducted on the datasets before, it was critical to report findings in a visual and easy-to-understand form. The findings were distributed and presented to a wide audience from different departments and varying backgrounds. In practise, this meant that the results of the study had to be understandable to audiences with more commercial backgrounds, but still complex enough to bring value and new insights to audiences with a technical background. The aim was to both showcase the possibilities a deep analytical investigation into logged software use data presented and provide insight on product use and different user segments.

Initially, a report containing visualizations made using the Python library matplotlib with short descriptions of each figure and a broad overview of high-level trends found was introduced. The report was first distributed and discussed with a limited audience with the goal of receiving comments and input on which figures were informative and how to improve

them. The report was also discussed in terms of how the task of clustering should be approached and different customer segments and journeys could be visualized.

After the initial report, different avenues for creating a dashboard were explored. The first benefiters and users of the dashboard were assumed to be the software development teams. Due to Redash being simple to implement to the current data structures and already being in use in many of the development teams, it was decided to use it as an initial platform for data visualization. Other data visualization tools, such as Tableau, were decided to be investigated later once a more comprehensive understanding of the varying use cases and preferred tools had been obtained. A model for combining and presenting the customer segments and their respective journeys was also planned out.

# 4. Results

This segment describes the results obtained from the methods introduced earlier in this thesis. It is divided into four main parts: data exploration, feature selection, clustering and customer segmentation. First, findings and conclusions drawn from data exploration that had the most influence on decision making are introduced. Next, a description of how consistent the feature selection methods were and how they ranked the features in order of goodness. Then, the results of the previously described clustering process are presented and analysed. Finally, the process leading to the final version of the customer segmentation and journey is described. A simplified version of the derived customer segments and journeys are also displayed. It should be noted, that in order to keep the required level of anonymity in this thesis, the precise values and features are not always presented or might be anonymised depending on the sensitivity of the data.

## 4.1. Data exploration

Data exploration was one of the most time-consuming tasks in this thesis. Data exploration and data preprocessing were both essential components in order to successfully combine data from many sources. One of the main challenges was data granularity, since the data from different sources had varying levels of granularity. Another challenge was time: some datasets had daily data on the companies, whereas some had data only if a company administrator had executed an action. Many different iterations with varying timeframes and aggregations were produced in order to find the best way to combine the various datasets. During this process, numerous figures and plots were produced to be able to compare the effectiveness of different data transformations. The most influential and important findings have been summarized below. In this segment, a license refers to a permit admitting access to the software for a single device, whereas the portal signifies the software's user interface.

Figure 11 depicts the distribution of companies according to the number of licenses sold. As we can see, most companies have between 1-10 licenses. However, when taking into account the red line, representing the cumulative sum of licenses in each segment, companies with 51-100 licenses make up the largest individual segment. The cumulative sum of licenses is especially important since the potential revenue a company can generate can be calculated

by multiplying the price per license by the number of licenses. By looking at the statistics presented in figure 11, the most lucrative segment appears to be companies with 51-100 licenses. It should also be noted, that despite larger companies only accounting for a small number of all of the companies, they contain a notable portion of all of the licenses.

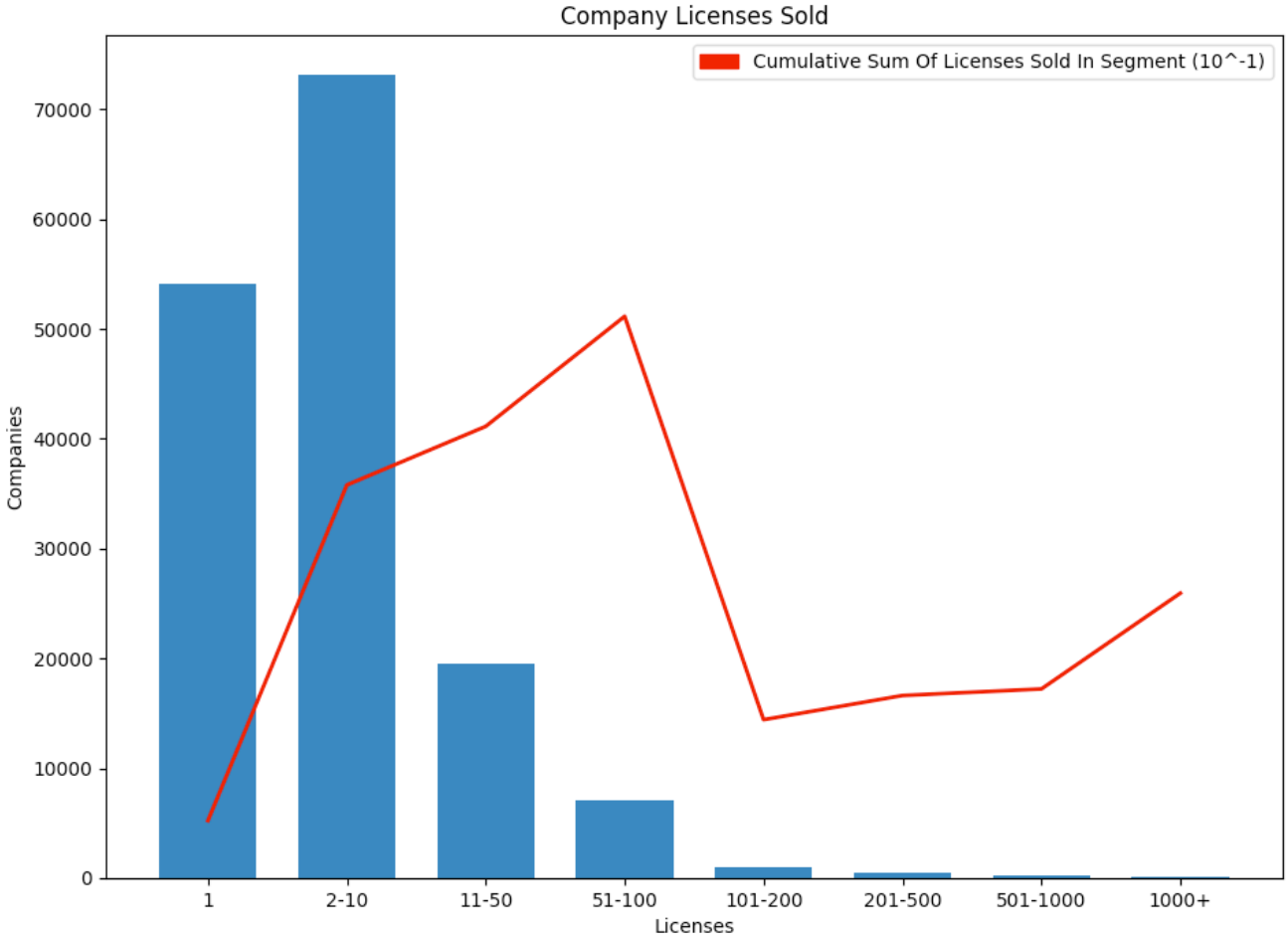


*Figure 11: Licenses sold per company*

As demonstrated in figure 12 below, only around 3-4% of all companies have user activity on a monthly basis. However, the active companies account for 23% of all of the software licenses. This gives a strong indication of larger companies being the most active users of the software portal. This hypothesis is also supported by figure 13. It should be noted that a relatively small number of companies with several thousands of licenses inflate the numbers in figure 12. Nevertheless, the 'companies with active users'–segment is a noteworthy group to be considered and explored in further analysis.

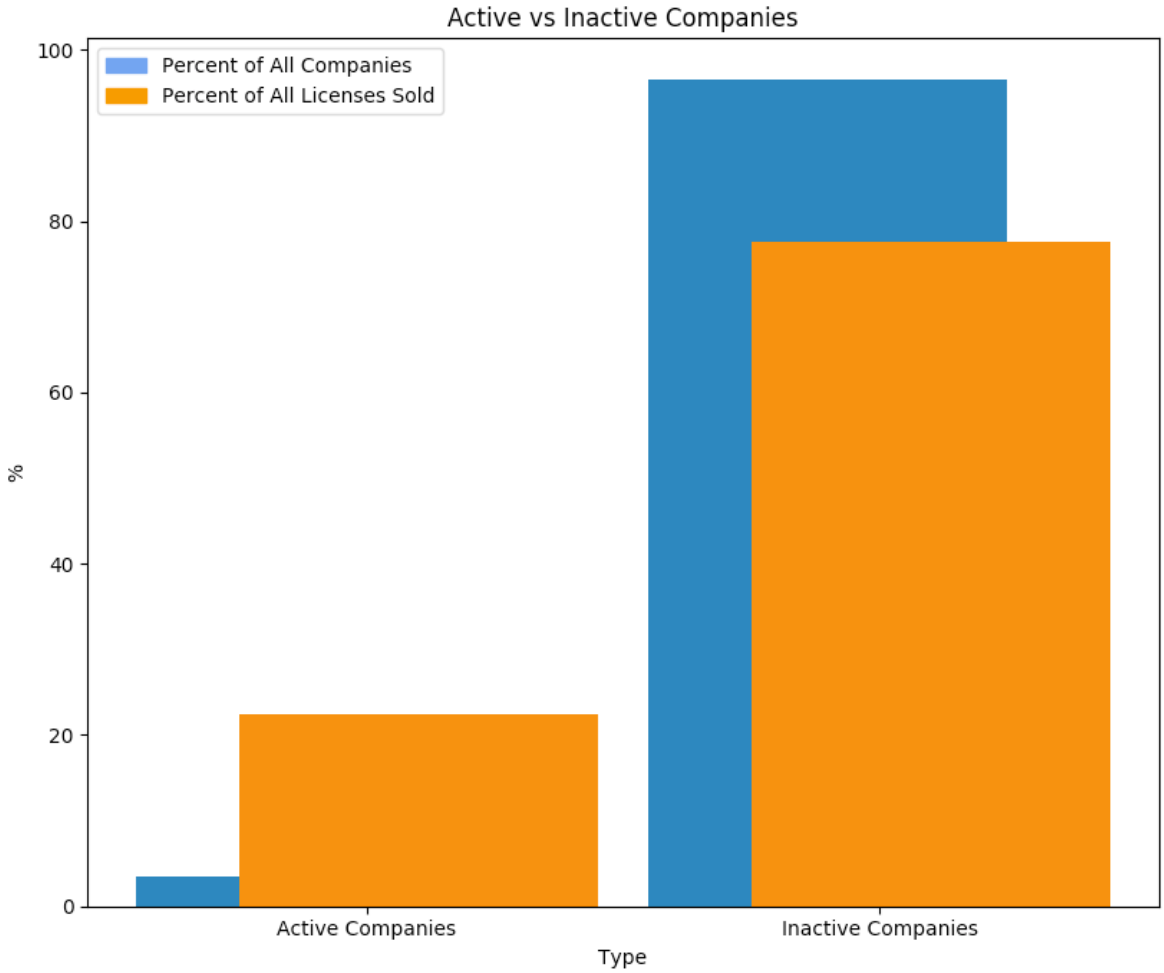


*Figure 12: Comparison of active and inactive companies*

Figure 13 below illustrates the distribution for the sizes of companies with a portal login in May. Compared to the distribution of the sizes for all companies (figure 11), the distribution in the figure below is considerably skewed towards companies with more than ten licenses. The distribution indicates that the bigger the company, the more likely they are to use the software portal. One can also derive from this, that bigger companies tend to be more self-managed, since self-management requires an user to login to the portal.

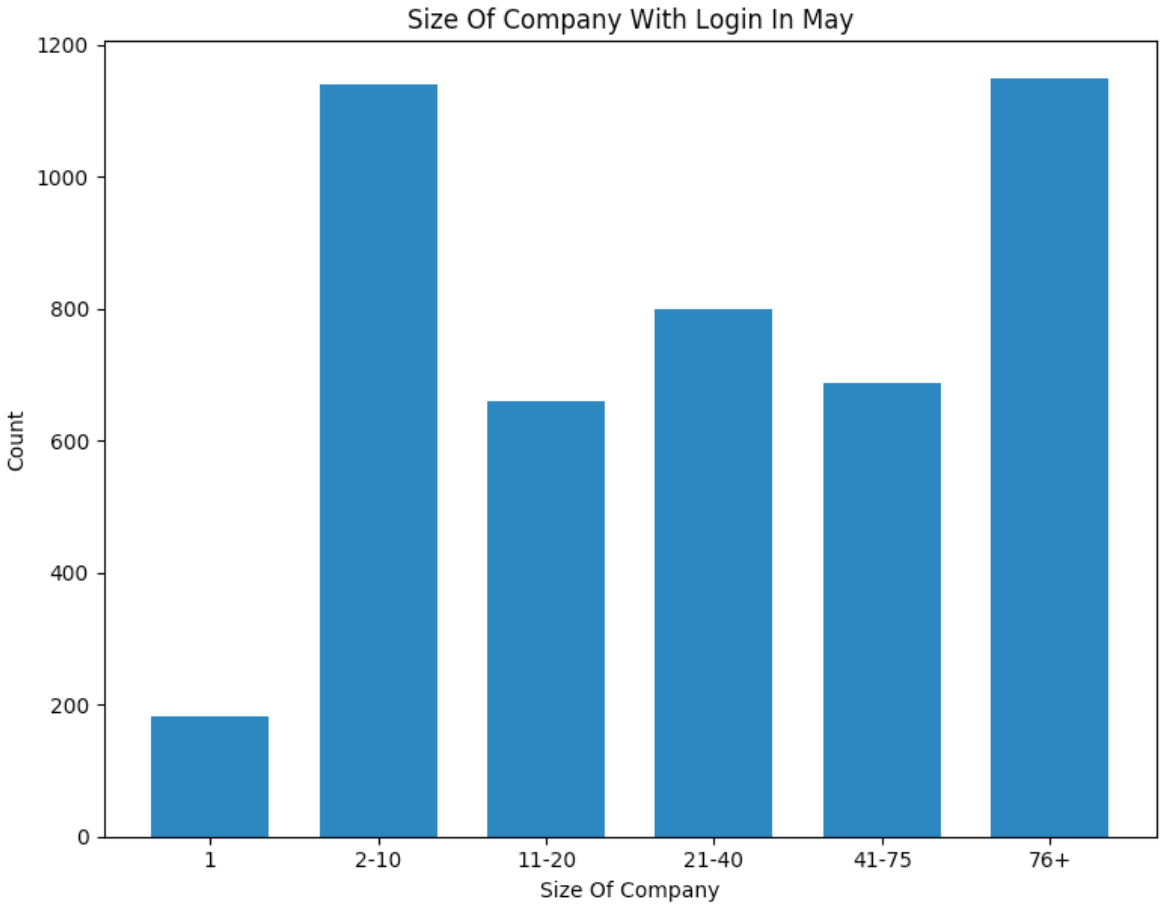


*Figure 13: Size of company with portal login*

Figure 14 represents the number of updates by update type. Automatically installed updates are represented by the 'No' bar, whereas manually installed updates are represented by the 'Yes' bar. The distribution in the figure shows, that there are clearly more automatically installed updates than updates installed by a user. Since automatic updates don't require an user to act, this might explain why such a small percentage of users log in on a monthly basis.

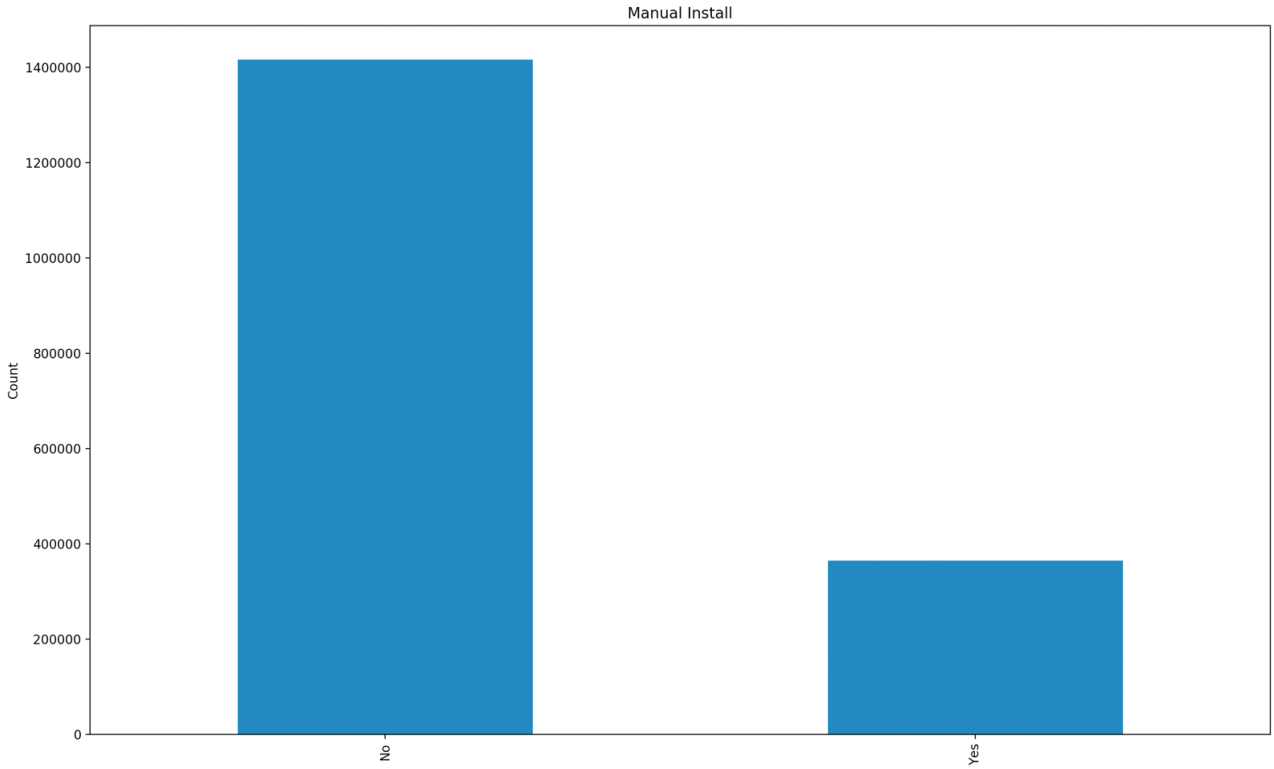


*Figure 14: Update installation type*

Initial clustering experiments with datasets containing most of the features were also conducted. Two critical findings were made during these experiments. First, it was noticed that the locations of the cluster centres stayed the same or very similar throughout many iterations with the same dataset and parameters. This meant that the clusters were stable and that clustering was a viable unsupervised machine learning method to apply onto the datasets. The cluster stability indicated that the dataset had underlying patterns that could be utilized by the clustering algorithms to group samples consistently into the same clusters.

The clustering algorithms produced stable clusters, since the cluster centroids ended up in the same locations upon multiple iterations. However, when inspecting the cluster descriptive values, another vital finding was made. The descriptive values, such as location of cluster centroid and standard deviation, were all relatively homogenous between different clusters. Since the aim of the thesis project was to create distinct customer segments and to be

able to utilize the segment descriptive values to characterize them, the fact that the cluster descriptive values for all segments were relatively homogenous posed a problem.
Considering the findings made during data exploration, the result wasn't surprising. Since the aim of the project was to find out how the customer uses the software, most of the datasets explored contained data on user actions from differing sources. The figures above all indicated a clear behavioural pattern for most users: they didn't use the software on a monthly basis. It was noticed that most companies rely completely on the software to automatically take care of all actions that might require user interaction. Since the majority of companies weren't actively using the software, most companies had the same entry of zero for a large portion of the dataset. As most samples shared a large set of homogenous data, it naturally caused the clusters and their descriptive values to very similar to each other.

As the goal was to create well-defined, consistent and informative customer segments, a decision was made based on the information gained from the initial clustering experiments. A preliminary split segmenting active and inactive companies was made: inactive companies were to be segmented according to more traditional segmenting methods, such as company size, whereas active companies were to be clustered. Especially when considering the data presented in figure 12 indicating that 3-4% of the users account for over 23% of all software licenses sold, it was decided that further investigations would be focused on active users. These active users are the so called 'power users' of the software, and were deemed to be the customers with the highest potential in terms of ability to influence, revenue and further sales opportunities. Most of the findings presented in the remaining part of the results section in this thesis are figures for the active users of the software.

## 4.2. Feature selection

To begin the investigation, both wrapper and filter methods were applied to the whole dataset. From the initial investigation, it was noticed that all other methods besides the variance threshold, the simplest method, were too heavy computationally to be applied to the entire dataset. Even when reserving huge amounts of computational capacity and memory from Amazon Web Services, the algorithms wouldn't run. However, this was largely due to the fact that the libraries containing the feature selection algorithms didn't support parallel computing, meaning that the full potential of cloud computing couldn't be used. This led to the

experimentation of feature selection methods on smaller sampled subsets of the original dataset.

Two main findings were made: without exception wrapper methods were too heavy computationally to be applied to the dataset, and consistent feature rankings were obtainable with a 10 percent sample of the whole dataset. No results were obtained from the wrapper methods, and therefore these were deemed unviable feature selection methods for the task in this thesis. On the other hand, some filter methods were successfully applied to a data subset. Varying sample sizes were experimented with, coming to the conclusion that a randomly sampled set containing 10 percent of the whole dataset could be used to obtain consistent feature rankings results. When filter methods were applied to the data subset, the computational capacity available was enough to run the algorithms, and the results of the methods were consistent.

As a result of the initial feature investigation, four feature selection methods were deemed to have the most potential for the task: variance threshold, Laplacian score, Spectral feature selection and Multi-Cluster feature selection. The selected algorithms were computationally viable and provided consistent results for the feature rankings. The dataset used to make the final decision contained all data on active companies only. Table 3 depicts feature rankings for each method for the 15 best-ranked features for active companies. The number in the table represents the feature's index in the dataset. The features are in descending order: the feature ranked the best by the algorithm is at the top of the list.

| | Variance Threshold | Laplacian Score | Spetral Feature Selection | Multi-Cluster Feature Selection |
|---|---|---|---|---|
| Feature ranking | 5 | 17 | 16 | 3 |
| | 17 | 5 | 2 | 4 |
| | 14 | 14 | 9 | 15 |
| | 15 | 15 | 1 | 11 |
| | 4 | 22 | 8 | 14 |
| | 21 | 13 | 7 | 19 |
| | 11 | 20 | 10 | 5 |
| | 19 | 19 | 0 | 17 |
| | 3 | 21 | 18 | 18 |
| | 12 | 11 | 23 | 8 |
| | 18 | 4 | 3 | 1 |
| | 7 | 12 | 4 | 2 |
| | 13 | 18 | 12 | 6 |
| | 20 | 3 | 1 | 7 |
| | 22 | 7 | 22 | 23 |

*Table 3: Feature rankings*

As the table above demonstrates, the methods seemed to give very different results for all methods, with some apparent correlation between the variance threshold and Laplacian score methods. Spectral Feature selection produced differing results from the other methods. The method was more likely to select features that resemble categorical features. However, when considering the large number of features, the methods shared a fair number of features as best-ranked features. Table 4 below exhibits how often a feature was ranked in the top 15.

| Feature number | Count |
|---|---|
| **0** | 1 |
| **1** | 3 |
| **2** | 2 |
| **3** | 4 |
| **4** | 4 |
| **5** | 3 |
| **6** | 1 |
| **7** | 4 |
| **8** | 2 |
| **9** | 1 |
| **10** | 1 |
| **11** | 3 |
| **12** | 3 |
| **13** | 2 |
| **14** | 3 |
| **15** | 3 |
| **16** | 1 |
| **17** | 3 |
| **18** | 4 |
| **19** | 3 |
| **20** | 2 |
| **21** | 2 |
| **22** | 3 |
| **23** | 2 |

*Table 4: Feature frequency*

Features with the index 2, 3, 7 and 18 were selected by all methods, whereas features 1, 5, 11, 12, 14, 15, 17, 19 and 22 were selected by almost all methods. This indicated, that even though the ordering of features was different for many methods, they still mostly ranked the same features highly. Still, purely based on the feature ranking results provided by the algorithm, it was almost impossible to make any conclusions on which method was best.

## 4.3. Clustering

Once results from feature selection had been obtained and documented, the performance for each algorithm was evaluated using the active users dataset. For each feature selection method the following clustering algorithms were applied and assessed: K-means, K-medoids, Fuzzy C-means and hierarchical clustering. Ten random initialization were used for K-means and C-means clustering. On the grounds presented by Bezdek (1981, p 65-80), a fuzziness value of 2 was used for the Fuzzy C-means clustering algorithm. Based on observations made using the 'elbow'-method on the whole dataset with different clustering methods, the optimal number of clusters was determined to be between 2 and 6 for all clustering methods. This resulted in the number of clusters being varied from 2 to 6 for each iteration. For each ordered set of features, the number of features was also varied to include the best ranked 25%, 50% or 75% of all features.

One of the major finding was that clustering results were unobtainable for some methods. Firstly, the hierarchical clustering method doesn't utilize a cluster centre, meaning the Sum of Squared Error metric couldn't be calculated without separately determining the mean or medoid for each cluster. Since hierarchical clustering doesn't rely on a centroid, it was determined that the Within Sum of Squared metric wouldn't be calculated for hierarchical clustering. Secondly, the K-medoids clustering algorithm used was subpar. The algorithm initialized the cluster medoids poorly, causing the algorithm to sometimes cluster all samples into the same cluster. This caused the result of the K-medoids algorithm to be rejected. If the algorithm didn't produce acceptable results, it was reinitialized and run again several times. However, in some cases no viable results were able to be obtained despite multiple reruns and initializations. The results for each method are presented later on in this section.

### 4.3.1. Variance threshold

Variance threshold was the first feature selection method to be tested. Since the method relies on removing features with no or low variance, the features were ranked according to where the algorithm set the threshold. In other words, the best-ranked features were the ones that were left as the threshold for the algorithm was increased.

The goodness and optimal number of clusters were assessed by using the four clustering evaluation measures presented earlier in this thesis: Within-Cluster Sum of Squares, Silhouette score, Calinski-Harabasz score and Davies Bouldin score. The smaller the Within-Cluster Sum of Squares, the closer the cluster centroids are to the observations in the corresponding cluster. The method can be used to determine the optimal number of clusters by using the 'elbow'-method: the optimal number of clusters is at the point, where the Within-Cluster Sum of Squares metric improves only marginally compared to the previous improvements. It should be noted that the Within-Cluster Sum of Squares metric decreases monotonically as the number of clusters is increased. As for the Silhouette Score and Calinski-Harabasz, the higher the score, the better the result, whereas for the Davies Bouldin score, a lower score indicates better quality clusters.

| **25% Features** | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,68 | 0,50 | 3829,29 | 0,98 |
| 3 | 0,51 | 0,57 | 4487,10 | 0,86 |
| 4 | 0,35 | 0,68 | 5092,52 | 0,65 |
| 5 | 0,26 | 0,74 | 5993,24 | 0,59 |
| 6 | 0,17 | 0,83 | 7777,79 | 0,43 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | X | X | X | X |
| 3 | X | X | X | X |
| 4 | X | X | X | X |
| 5 | X | X | X | X |
| 6 | X | X | X | X |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,69 | 0,50 | 3822,09 | 0,98 |
| 3 | 0,47 | 0,57 | 4487,10 | 0,86 |
| 4 | 0,33 | 0,68 | 5092,52 | 0,65 |
| 5 | 0,22 | 0,74 | 5990,51 | 0,60 |
| 6 | 0,13 | 0,84 | 7772,79 | 0,44 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,50 | 3812,61 | 0,99 |
| 3 | None | 0,52 | 3632,02 | 1,00 |
| 4 | None | 0,66 | 4324,99 | 0,64 |
| 5 | None | 0,71 | 4606,50 | 0,63 |
| 6 | None | 0,80 | 5296,12 | 0,49 |

*Table 5: Clustering performance with 25% of variance threshold features*

Table 5 depicts the results from clustering for a dataset containing 25% of all features. The K-medoids algorithm was deemed ineffective: no results were obtained, due to the algorithm clustering all of the data samples into the same cluster. The K-means and Fuzzy C-means algorithms performed almost identically, and both performed marginally better than Hierarchical clustering. On a general level, all of the algorithms performed better as the number of clusters was increased indicating that six clusters worked best.

| **50% Features** | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,74 | 0,49 | 3924,80 | 0,97 |
| 3 | 0,57 | 0,49 | 4037,77 | 0,93 |
| 4 | 0,44 | 0,57 | 4146,21 | 0,73 |
| 5 | 0,37 | 0,60 | 4398,19 | 0,83 |
| 6 | 0,30 | 0,67 | 4813,56 | 0,69 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,76 | 0,33 | 1591,91 | 1,65 |
| 3 | 0,68 | 0,34 | 1290,93 | 1,48 |
| 4 | 0,45 | 0,50 | 3453,63 | 0,94 |
| 5 | 0,41 | 0,55 | 3111,21 | 0,87 |
| 6 | 0,26 | 0,67 | 4808,24 | 0,69 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,74 | 0,49 | 3924,80 | 0,97 |
| 3 | 0,55 | 0,49 | 4037,77 | 0,93 |
| 4 | 0,43 | 0,57 | 4146,21 | 0,73 |
| 5 | 0,35 | 0,60 | 4398,19 | 0,83 |
| 6 | 0,27 | 0,67 | 4813,56 | 0,69 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,48 | 3829,97 | 1,03 |
| 3 | None | 0,44 | 3313,47 | 1,06 |
| 4 | None | 0,55 | 3524,62 | 0,74 |
| 5 | None | 0,57 | 3627,84 | 1,01 |
| 6 | None | 0,64 | 3753,58 | 0,90 |

*Table 6: Clustering performance with 50% of variance threshold features*

The numbers in table 6 depict the results when clustering with half of all features. Due to the increase in the number of features, the K-medoids algorithm was also able to successfully cluster the samples. Once again, the K-means and Fuzzy C-means algorithms provided almost identical results. The K-medoids algorithm provided relatively poor results when using a small number of clusters, but as the number of clusters was increased, the results improved to almost the same level as K-means and Fuzzy C-means. Hierarchical clustering produced

similar results to K-means and Fuzzy C-means when using a small number of clusters. However, the cluster quality didn't improve as notably as the other algorithms when the number of clusters was increased. For most cases, six clusters provided the best result.

| 75% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,80 | 0,46 | 3491,14 | 1,01 |
| 3 | 0,64 | 0,42 | 3368,08 | 1,07 |
| 4 | 0,53 | 0,47 | 3270,97 | 0,87 |
| 5 | 0,49 | 0,49 | 3217,36 | 1,03 |
| 6 | 0,41 | 0,55 | 3279,23 | 0,87 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,87 | 0,28 | 1404,76 | 1,77 |
| 3 | 0,86 | 0,17 | 799,69 | 2,47 |
| 4 | 0,54 | 0,45 | 2999,76 | 0,92 |
| 5 | 0,53 | 0,36 | 2233,34 | 1,14 |
| 6 | 0,51 | 0,47 | 2126,94 | 1,10 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,80 | 0,46 | 3491,14 | 1,01 |
| 3 | 0,63 | 0,42 | 3368,08 | 1,07 |
| 4 | 0,53 | 0,47 | 3265,94 | 0,87 |
| 5 | 0,45 | 0,50 | 3238,34 | 0,97 |
| 6 | 0,39 | 0,55 | 3241,73 | 0,89 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,44 | 3392,31 | 1,08 |
| 3 | None | 0,38 | 2793,62 | 1,17 |
| 4 | None | 0,45 | 2809,99 | 0,91 |
| 5 | None | 0,47 | 2798,65 | 1,08 |
| 6 | None | 0,50 | 2712,24 | 1,20 |

*Table 7: Clustering performance with 75% of variance threshold features*

For the final assessment, three-fourths of all features were included into the dataset. The results are illustrated in table 7. An interesting pattern was noticed: the goodness of clustering became only slightly better as the number of clusters was increased for all methods except for K-medoids. The K-medoids algorithm produced varying results depending on the measure of goodness. In most cases, six clusters produced the best results. K-means and Fuzzy C-means outperformed the other clustering algorithms most of the time for the variance threshold method.

### 4.3.2. Laplacian Score

The second feature selection method to be tested was the Laplacian Score. Table 8 represents the results obtained from the clustering methods using one quarter of the features as ranked best by the Laplacian Score algorithm. A clear pattern could be seen: as the number of clusters was increased, the result of the clustering improved, with the exception of K-medoids. K-means and Fuzzy C-means algorithms produced almost identical and oftentimes the best results. K-medoids clustering provided varying results, where an odd number of clusters seemed to always produce a better clustering result. Generally speaking, six clusters produced the best results.

| 25% of Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,66 | 0,52 | 4194,77 | 0,94 |
| 3 | 0,48 | 0,59 | 5137,65 | 0,81 |
| 4 | 0,31 | 0,71 | 6191,87 | 0,61 |
| 5 | 0,22 | 0,76 | 7972,47 | 0,53 |
| 6 | 0,13 | 0,86 | 12332,95 | 0,35 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,63 | 0,40 | 2112,22 | 1,44 |
| 3 | 0,41 | 0,59 | 5137,65 | 0,81 |
| 4 | 0,41 | 0,41 | 3538,79 | 2,09 |
| 5 | 0,29 | 0,48 | 4125,20 | 1,02 |
| 6 | 0,29 | 0,48 | 3299,60 | 1,74 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,66 | 0,52 | 4194,05 | 0,94 |
| 3 | 0,45 | 0,59 | 5137,65 | 0,81 |
| 4 | 0,30 | 0,71 | 6191,87 | 0,61 |
| 5 | 0,20 | 0,76 | 7972,47 | 0,53 |
| 6 | 0,11 | 0,86 | 12332,95 | 0,35 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,51 | 4172,37 | 0,95 |
| 3 | None | 0,54 | 4252,87 | 0,96 |
| 4 | None | 0,71 | 5837,96 | 0,60 |
| 5 | None | 0,76 | 7261,76 | 0,58 |
| 6 | None | 0,85 | 10124,07 | 0,46 |

*Table 8: Clustering performance with 25% of Laplacian score features*

Next, the number of features was doubled to contain 50% of the highest ranked features. The outcomes for each clustering algorithm have been displayed below in table 9. All algorithms, except for K-medoids, largely produced monotonically improving results as the number of clusters was increased. However, for all other methods except for K-medoids, the results dropped when three clusters were applied. This suggested, that a two-cluster model could be optimal. K-means and Fuzzy C-means clustering commonly provided the best results, with K-means clustering performing slightly better. The K-medoids algorithm produced varying results with four and six clusters performing the best. On a general note, six clusters produced the best outcomes.

| **50% of Features** | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,73 | 0,54 | 4796,94 | 0,91 |
| 3 | 0,55 | 0,55 | 4754,03 | 0,84 |
| 4 | 0,47 | 0,60 | 5329,27 | 0,85 |
| 5 | 0,31 | 0,71 | 7043,54 | 0,69 |
| 6 | 0,22 | 0,80 | 8540,44 | 0,58 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,74 | 0,35 | 1460,57 | 1,69 |
| 3 | 0,64 | 0,32 | 3077,67 | 1,57 |
| 4 | 0,38 | 0,56 | 4191,03 | 0,89 |
| 5 | 0,35 | 0,49 | 3514,23 | 1,37 |
| 6 | 0,25 | 0,60 | 4302,41 | 1,02 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,73 | 0,54 | 4796,94 | 0,91 |
| 3 | 0,52 | 0,53 | 4595,16 | 0,93 |
| 4 | 0,38 | 0,63 | 4815,53 | 0,78 |
| 5 | 0,29 | 0,72 | 6952,59 | 0,72 |
| 6 | 0,20 | 0,81 | 8429,32 | 0,62 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,53 | 4504,80 | 0,98 |
| 3 | None | 0,50 | 3855,08 | 0,98 |
| 4 | None | 0,56 | 4576,46 | 0,96 |
| 5 | None | 0,72 | 6829,03 | 0,74 |
| 6 | None | 0,81 | 7969,60 | 0,65 |

*Table 9: Clustering performance with 50% of Laplacian score features*

Lastly, the number of features in the dataset was increased to contain 75% of all features. The results of the clustering are presented in table 10. Similar patterns were recognized with an increase in the number of features as with the results presented earlier in table 9. Broadly,

K-means provided the best results, followed by Fuzzy C-means, hierarchical and K-medoids respectively. K-medoids produced largely varying results depending on the number of clusters, without any clear patterns or insight to be gained.

| 75% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,80 | 0,55 | 4518,86 | 0,64 |
| 3 | 0,65 | 0,45 | 4233,84 | 1,06 |
| 4 | 0,56 | 0,50 | 4413,57 | 0,96 |
| 5 | 0,44 | 0,57 | 4970,66 | 0,82 |
| 6 | 0,38 | 0,63 | 5102,62 | 0,69 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,91 | 0,28 | 1125,87 | 1,93 |
| 3 | 0,72 | 0,43 | 3748,24 | 1,16 |
| 4 | 0,72 | 0,25 | 2127,32 | 2,31 |
| 5 | 0,64 | 0,31 | 1798,48 | 1,54 |
| 6 | 0,50 | 0,32 | 2314,99 | 2,65 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,82 | 0,52 | 4246,97 | 0,85 |
| 3 | 0,62 | 0,45 | 4233,84 | 1,06 |
| 4 | 0,51 | 0,51 | 4044,85 | 0,95 |
| 5 | 0,44 | 0,53 | 3517,40 | 0,79 |
| 6 | 0,36 | 0,63 | 5100,94 | 0,70 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,55 | 4518,86 | 0,64 |
| 3 | None | 0,47 | 3643,46 | 1,06 |
| 4 | None | 0,46 | 3811,22 | 1,07 |
| 5 | None | 0,55 | 4431,53 | 0,88 |
| 6 | None | 0,60 | 4306,50 | 0,77 |

*Table 10: Clustering performance with 75% of Laplacian score features*

The results were relatively stable throughout all of the clusters for all methods except K-medoids. It was also noticed that compared to the results presented in tables 8 and 9, the clustering performance increased only very moderately as the number of clusters was increased. In general, two or six clusters produced the best results.

### 4.3.3. Spectral feature selection

The next method to be tested was Spectral feature selection. Table 11 presents clustering results using a dataset containing the best 25% of all features ranked by the Spectral feature

selection algorithm. For the K-means, Fuzzy C-means and Hierarchical clustering algorithms a broad pattern could be drawn from the results: as the number of clusters increased, the clustering results degraded. The K-medoids algorithm provided the exact same result as K-means and Fuzzy C-means when using two clusters. However, atypical to the method, the within-cluster sum of squares for K-medoids increased as the number of clusters was changed to three. This was most likely caused by the initialization problems the K-medoids algorithm had. K-means clustering provided slightly better results than Fuzzy C-means and hierarchical clustering. Fuzzy C-means clustering provided better clustering results than hierarchical clustering when using two clusters. However, hierarchical clustering seemed to provide better results thereafter as the number of clusters was increased. On a general note, two clusters appeared to provide the best results.

| 25% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,03 | 0,96 | 213099,52 | 0,06 |
| 3 | 0,03 | 0,93 | 142542,28 | 0,66 |
| 4 | 0,02 | 0,82 | 115846,91 | 0,71 |
| 5 | 0,02 | 0,84 | 105481,60 | 0,53 |
| 6 | 0,02 | 0,75 | 98285,90 | 0,66 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,03 | 0,96 | 213099,52 | 0,06 |
| 3 | 0,18 | 0,06 | 33,97 | 12,42 |
| 4 | 0,18 | -0,03 | 17,07 | 14,19 |
| 5 | 0,02 | 0,53 | 72141,70 | 1,12 |
| 6 | 0,02 | 0,49 | 58569,07 | 1,07 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,03 | 0,96 | 213099,52 | 0,06 |
| 3 | 0,02 | 0,78 | 129742,28 | 0,84 |
| 4 | 0,02 | 0,73 | 91159,99 | 1,05 |
| 5 | 0,02 | 0,67 | 90125,29 | 0,91 |
| 6 | 0,02 | 0,56 | 79171,46 | 0,96 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,96 | 201015,37 | 0,06 |
| 3 | None | 0,91 | 131450,82 | 0,88 |
| 4 | None | 0,90 | 110564,56 | 0,95 |
| 5 | None | 0,90 | 97456,36 | 0,88 |
| 6 | None | 0,90 | 93456,48 | 0,66 |

*Table 11: Clustering performance with 25% of Spectral feature selection features*

The next experiment was conducted with a dataset containing half of the features. The results can be seen in table 12. The results were very similar to the ones presented in table 11, which contained 25% of all features. K-means produced the best outcomes, followed closely by Fuzzy C-means and hierarchical clustering. This time, however, the clustering result of Fuzzy C-means degraded a lot less as the number of clusters was increased. Fuzzy C-means and hierarchical clustering produced essentially the same results if only looking at clustering goodness metrics. K-medoids clustering produced, once again, a comparable result with two clusters as K-means and Fuzzy C-means, after which the quality of the clusters produced deteriorated. Generally, two clusters produced the best clustering results.

| **50% Features** | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,09 | 0,88 | 44473,25 | 0,17 |
| 3 | 0,07 | 0,74 | 38220,21 | 0,42 |
| 4 | 0,06 | 0,65 | 31704,60 | 0,52 |
| 5 | 0,05 | 0,58 | 28441,95 | 0,68 |
| 6 | 0,05 | 0,60 | 26126,98 | 0,63 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,08 | 0,88 | 44473,25 | 0,17 |
| 3 | 0,21 | 0,15 | 148,86 | 5,26 |
| 4 | 0,06 | 0,43 | 23941,80 | 0,94 |
| 5 | 0,05 | 0,54 | 23611,50 | 0,91 |
| 6 | 0,20 | 0,07 | 80,11 | 6,04 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,09 | 0,88 | 44473,25 | 0,17 |
| 3 | 0,06 | 0,72 | 39062,67 | 0,48 |
| 4 | 0,06 | 0,67 | 29998,06 | 0,67 |
| 5 | 0,05 | 0,54 | 28088,19 | 0,74 |
| 6 | 0,04 | 0,52 | 25416,98 | 0,76 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,88 | 43874,26 | 0,17 |
| 3 | None | 0,72 | 37681,21 | 0,45 |
| 4 | None | 0,55 | 28714,81 | 0,76 |
| 5 | None | 0,52 | 24953,72 | 0,79 |
| 6 | None | 0,53 | 23513,65 | 0,74 |

*Table 12: Clustering performance with 50% of Spectral feature selection features*

Lastly, the dataset was expanded to contain three-fourths of all features. The clustering quality has been illustrated below in table 13. K-means, Fuzzy C-means and K-medoids produced the same result when using two clusters. When the number of clusters was increased above two, the clustering results for K-medoids degraded significantly.

| 75% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,18 | 0,76 | 11110,01 | 0,33 |
| 3 | 0,16 | 0,78 | 9582,47 | 0,47 |
| 4 | 0,14 | 0,56 | 9364,04 | 0,65 |
| 5 | 0,12 | 0,57 | 9275,20 | 0,72 |
| 6 | 0,11 | 0,42 | 8306,50 | 0,84 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,18 | 0,76 | 11110,01 | 0,33 |
| 3 | 0,28 | 0,12 | 746,32 | 1,29 |
| 4 | 0,28 | 0,08 | 303,48 | 3,98 |
| 5 | 0,27 | 0,10 | 475,32 | 1,57 |
| 6 | 0,26 | -0,02 | 375,41 | 2,47 |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,18 | 0,76 | 11110,01 | 0,33 |
| 3 | 0,15 | 0,46 | 7247,43 | 1,03 |
| 4 | 0,14 | 0,47 | 5774,61 | 1,11 |
| 5 | 0,13 | 0,42 | 4527,17 | 1,34 |
| 6 | 0,10 | 0,42 | 8282,50 | 0,83 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,75 | 9953,77 | 0,31 |
| 3 | None | 0,78 | 9475,38 | 0,46 |
| 4 | None | 0,53 | 8632,23 | 0,61 |
| 5 | None | 0,56 | 8550,03 | 0,70 |
| 6 | None | 0,57 | 8229,55 | 0,58 |

*Table 13: Clustering performance with 75% of Spectral feature selection features*

An interesting pattern could be seen for the other clustering methods as well: the goodness of the algorithms degraded slowly as the number of clusters was increased, with Fuzzy C-means being a slight exception. The result of Fuzzy C-means deteriorated faster than K-means and hierarchical clustering, but improved slightly when the number of clusters was increased to six. Still, a clear result could be derived from the results: two clusters was optimal for Spectral feature selection. It was also previously noted, that the Spectral feature selection method ranked features that resembled categorical features highly. This resulted in a significant difference in the goodness of clustering evaluation metrics. The evaluation

methods tended to assess the Spectral feature selection method to be significantly better than the other feature selection methods.

### 4.3.4. Multi-Cluster feature selection

The last method to be tested was Multi-Cluster Feature Selection. Table 14 illustrates the results obtained using a dataset containing the best ranked 25% of all features. The first observation was, that K-medoids clustered all of the samples into the same cluster meaning, that it wasn't a viable option for clustering with the dataset used. This was once again caused by the initialization problems in the algorithm.

| 25% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,24 | 0,78 | 13342,15 | 0,38 |
| 3 | 0,20 | 0,79 | 10338,82 | 0,55 |
| 4 | 0,17 | 0,76 | 8745,81 | 0,51 |
| 5 | 0,16 | 0,77 | 8452,95 | 0,48 |
| 6 | 0,14 | 0,80 | 9649,71 | 0,59 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | X | X | X | X |
| 3 | X | X | X | X |
| 4 | X | X | X | X |
| 5 | X | X | X | X |
| 6 | X | X | X | X |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,22 | 0,78 | 13356,70 | 0,39 |
| 3 | 0,19 | 0,79 | 10338,82 | 0,55 |
| 4 | 0,16 | 0,76 | 8792,77 | 0,52 |
| 5 | 0,14 | 0,59 | 7207,66 | 0,83 |
| 6 | 0,12 | 0,62 | 7762,69 | 0,81 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,78 | 13356,70 | 0,39 |
| 3 | None | 0,78 | 10097,53 | 0,67 |
| 4 | None | 0,77 | 8514,46 | 0,62 |
| 5 | None | 0,78 | 8405,56 | 0,79 |
| 6 | None | 0,80 | 8920,09 | 0,73 |

*Table 14: Clustering performance with 25% of Multi-Cluster feature selection features*

The K-means and hierarchical algorithms provided very comparable results, with K-means being slightly more effective. Fuzzy C-means provided similar results to K-means and hierarchical clustering, but as the number of clusters was increased to 4 and over, the clustering performance typically degraded more than the other two methods. On a general level, the best clustering results were achieved when using two or six clusters.

Table 15 presents the clustering results when including half of the features as ranked best by the feature selection algorithm. Once again, due to the initialization problems in the algorithm, K-medoids grouped all samples into the same cluster. However, the clustering results were very different for the other methods as the number of features was increased.

| 50% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,62 | 0,53 | 4617,08 | 0,90 |
| 3 | 0,37 | 0,62 | 6141,80 | 0,63 |
| 4 | 0,29 | 0,66 | 6314,08 | 0,59 |
| 5 | 0,27 | 0,68 | 6054,19 | 0,75 |
| 6 | 0,24 | 0,66 | 5765,68 | 0,72 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | X | X | X | X |
| 3 | X | X | X | X |
| 4 | X | X | X | X |
| 5 | X | X | X | X |
| 6 | X | X | X | X |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,62 | 0,53 | 4612,18 | 0,90 |
| 3 | 0,36 | 0,62 | 6141,80 | 0,63 |
| 4 | 0,27 | 0,66 | 6314,08 | 0,59 |
| 5 | 0,24 | 0,62 | 5523,26 | 0,69 |
| 6 | 0,22 | 0,66 | 5765,68 | 0,72 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,53 | 4616,53 | 0,90 |
| 3 | None | 0,60 | 5334,01 | 0,69 |
| 4 | None | 0,64 | 5233,08 | 0,65 |
| 5 | None | 0,67 | 5446,69 | 0,84 |
| 6 | None | 0,66 | 5131,17 | 0,81 |

*Table 15: Clustering performance with 50% of Multi-Cluster feature selection features*

In comparison to the results in table 14, only relatively small changes were noted as the number of clusters was increased. K-means, Fuzzy C-means and hierarchical all provided similar results. For most cases, either four or five clusters seemed to produce the best results.

The final clustering evaluation was conducted on a dataset that contained the best ranked 75% features, as ranked by the Multi-Cluster Feature Selection algorithm. The results were summarized into table 16. Even as the number of features was increased to contain nearly all features available, the K-medoids algorithm still had trouble initializing properly.

| 75% Features | | | | |
|---|---|---|---|---|
| **K-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,76 | 0,43 | 3202,64 | 1,06 |
| 3 | 0,60 | 0,44 | 3383,70 | 1,04 |
| 4 | 0,49 | 0,50 | 3462,99 | 0,84 |
| 5 | 0,42 | 0,53 | 3559,71 | 0,83 |
| 6 | 0,35 | 0,59 | 3783,69 | 0,74 |
| **K-medoids** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | X | X | X | X |
| 3 | X | X | X | X |
| 4 | X | X | X | X |
| 5 | X | X | X | X |
| 6 | X | X | X | X |
| **Fuzzy C-means** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | 0,77 | 0,43 | 3192,93 | 1,07 |
| 3 | 0,59 | 0,44 | 3383,70 | 1,04 |
| 4 | 0,48 | 0,50 | 3462,99 | 0,84 |
| 5 | 0,40 | 0,53 | 3556,33 | 0,83 |
| 6 | 0,34 | 0,59 | 3783,69 | 0,74 |
| **Hierarchical** | | | | |
| **Clusters** | **Within-Cluster Sum of Squares** | **Silhouette Score** | **Calinski-Harabasz Score** | **Davies Bouldin Score** |
| 2 | None | 0,43 | 3181,35 | 1,08 |
| 3 | None | 0,40 | 2803,45 | 1,15 |
| 4 | None | 0,47 | 2999,38 | 0,88 |
| 5 | None | 0,50 | 2929,53 | 0,89 |
| 6 | None | 0,55 | 2964,80 | 0,79 |

*Table 16: Clustering performance with 75% of Multi-Cluster feature selection features*

The Fuzzy C-means and K-means algorithms produced very similar results, since clustering performance improved as the number of clusters was increased. Hierarchical clustering improved slightly as the number of clusters was increased throughout the iterations. K-means and Fuzzy C-means clustering algorithms generally provided the best results, followed by hierarchical clustering. K-medoids was judged to be unsuitable for clustering with features selected by Multi-Cluster feature selection. On a broad level, six clusters produced the best clustering results.

### 4.3.5. Selecting Number of Features

As a result of the findings presented previously, it was decided that Spectral Feature selection would be used in combination with K-means clustering. The reasoning was, that the quality of clustering was generally better than for other methods. Spectral feature selection also produced stable clusters meaning that it consistently provided the same cluster centroid locations. Since the best clustering metrics were often gained with two clusters, it was decided that the optimal segmentation results would be obtained by grouping the active users into two groups. However, the optimal number of features was still unclear. An experiment was set up to gain better insight on how the number of features affected the clustering performance. The goal was to find the optimal number of features.

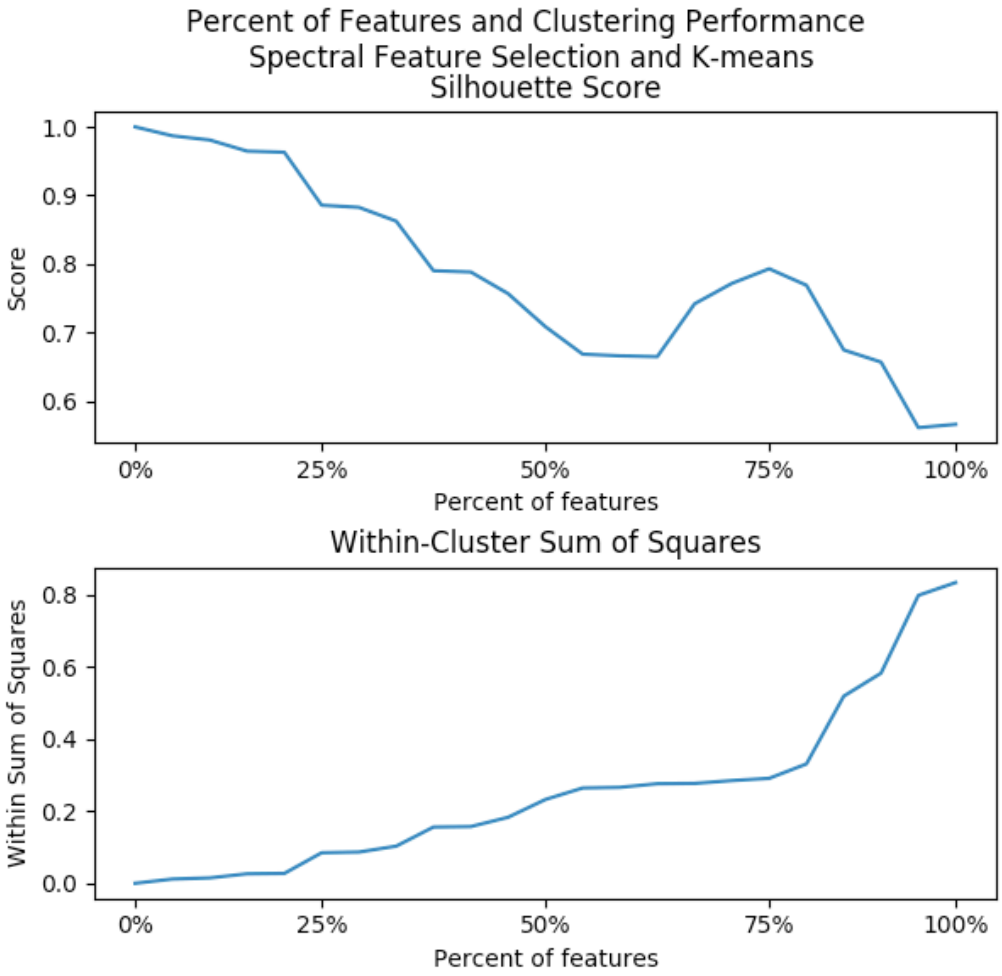


*Figure 15: Number of features and clustering performance*

Figure 15 presents the Silhouette score and within-cluster sum of squares as a function of the number of features. A high silhouette score indicates better clustering performance, whereas a low within-cluster sum of squares implies good cluster quality. The silhouette score showed a clear pattern: clustering performance degraded as the number of features was increased up until around 60%, after which the performance improved notably until 75% of the features were in use for clustering. After reaching 75%, the performance started to degrade once again. A similar pattern could be depicted for within-cluster sum of squares: the sum of squares increased steadily up until 75%, after which the metric started to degrade noticeably. This gave indication, that increasing the number of features up until 75% plausible, after which the performance started to degrade rapidly indicating, that it didn't appear justified to increase the number of features anymore. Since the goal was to optimize the clustering performance by getting rid of unnecessary features while still keeping the clusters as granular and descriptive as possible, a decision to use 75% of the best-ranked features was made.

## 4.4. Customer segments

The main goal of this thesis was to segment customers according to data on their software use patterns. Figure 16 shows the composition of different customer segments. The percentages in the figure depict the percent of all companies that belong to the segment. First, as a product of insights gained from data exploration, an initial split between users was made to split companies with active users and companies with inactive users into two separate groups. This was due to the fact, that inactive companies had little to no activity in the database, making the results of clustering suboptimal, thus leading to homogenous clusters when comparing them to each other.

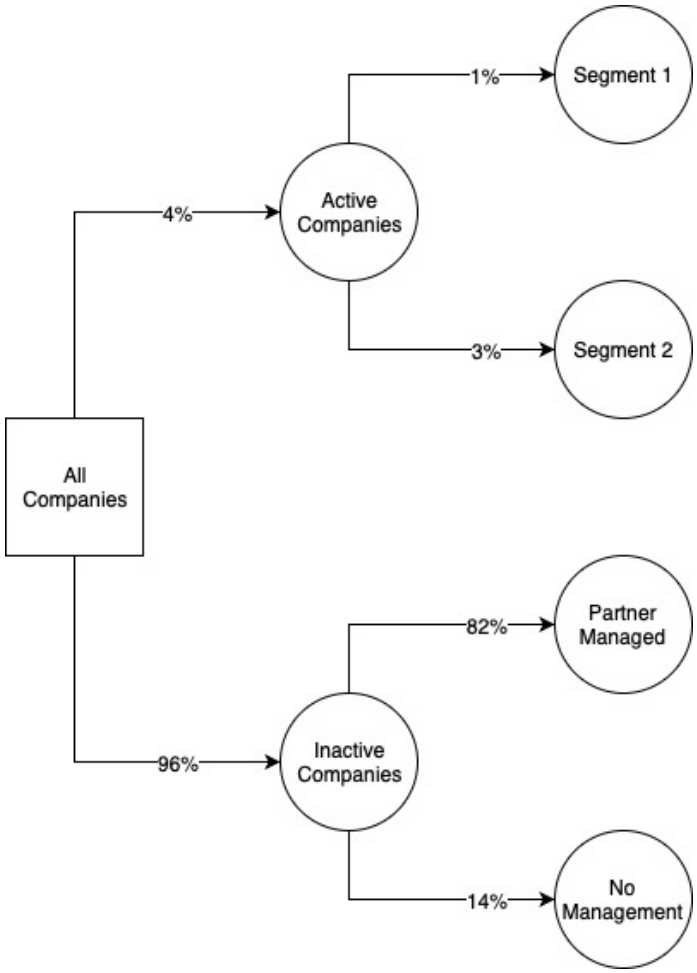


*Figure 16: Customer segments*

Since activity was the main feature considered throughout the project, the segmentation for inactive companies was made according to whether any activity was logged that might be considered to have an effect on the company. In practise, this meant that the companies were split according to whether the partner that sold the software to the company had activity in the portal. If the partner was active in the portal, the company was deemed partner managed. This meant, that even if the company itself wasn't active, it might still be managed and maintained by the partner. If there wasn't any activity detected by the partner or the company itself, it was concluded that the company wasn't managed by anyone.

For active companies, the segmentation was conducted using clustering. As a result of the investigation conducted previously, it was decided to use Spectral feature selection with 75% of the best rated features in combination with K-means clustering. These two groups are

presented in figure 16 as segment 1 and segment 2. The main differences between the segments have been described below.

Around one-fourths of all active companies were clustered into *segment 1*. Segment 1 has companies that generally have over 10 but under 100 devices. They have one to two administrators that are typically active one to four times a month. One of the main differentiating factors between the two segments was, that the partners for companies in segment 1 were completely inactive. On average, the partners in the segment had sold from hundreds to thousands of licenses to numerous companies.

Three-fourths of all active companies ended up being clustered into *segment 2*. The companies in segment 2 typically have from 5 to around 50 devices. One to two administrators are active one to three times a month. The partners for companies in segment 2 were active several times a month. The partners in this segment have sold licenses from tens to hundreds of licenses to companies and can therefore be concluded to be smaller than the partners in segment 1.

# 5. Conclusions

This section will outline and go through the main topics of this thesis. The methods and results are summarized, and a model based on the results of the project is presented. In the end of this section topics for further research and improvement are considered.

To start with, the aim and scope of the thesis was set. The goal of the project was to investigate how the software is used, and segment customers according to common software use patterns. The significance of the study was evaluated and a short examination of existing studies on the subject was carried out. Possible limitations, assumptions and risks were evaluated and taken note of.

To increase knowledge on the subject and create a basis for the study a literature review was conducted. First, different feature selection algorithms, methods and processes were investigated. As a result, a list of potential algorithms and related Python libraries was produced. The second subject to be investigated was customer analysis. Various methods to utilize logged software use data and map customer journeys were examined. The use of clustering for customer segmentation was explored. Different clustering algorithms were studied and related Python libraries were collected. In addition, various methods to evaluate clustering results and compare feature selection as well as clustering methods to each other were found and rated according to relevance to the application in this thesis.

Next, the research setting was described. The approach for the experimental phase of the thesis was introduced on a general level. The various tools, applications, software and software packages used were introduced and various use cases were clarified. Two processes were then introduced: one for data preprocessing and one for evaluating clustering performance with different combinations of feature selection and clustering algorithms. An introduction to the reporting and visualization of the thesis and other relevant findings was given.

The results of the project were presented. The main results of data exploration on the preprocessed dataset were introduced and examined. Findings made from various groupings, visualizations and clustering experiments were explored. The most significant discovery was

concluded to be that most customers are not active companies and rely on automated services. Nevertheless, while only a small segment of customers was active, they still accounted for over one-fifths of all licenses sold.

After, the results of the feature selection process were presented. It was noticed, that only filter feature selection algorithms could be applied to the dataset, as wrapper feature selection methods were computationally too heavy. Out of the tested feature selection methods, four were chosen to conduct further experimentation with: variance threshold, Laplacian score, Spectral feature selection and Multi-Cluster feature selection. The 15 best-ranked features for each feature selection method were presented in descending order. A clear pattern could be seen, where the feature selection algorithms favoured and ranked the same features highly, but often in differing order.

To gain numerical insight on different clustering and feature selection algorithms, various ways to experiment with different parameters were set up. Four clustering algorithms were selected as candidates: K-means, K-medoids, Fuzzy C-means and Hierarchical clustering. The goal was to find the best combination of a feature selection algorithm and a clustering algorithm. All combinations of feature selection algorithms and clustering algorithms were tested with a varying number of clusters and features.

*Variance Threshold* feature selection provided relatively consistent results. In most cases, the best results were obtained when using six clusters. K-means and Fuzzy C-Means clustering often provided the best quality clusters. *Laplacian Score* feature selection regularly provided better quality clusters than the variance threshold algorithm. The best clustering results were achieved fairly consistently with the K-means algorithm and six clusters. *Spectral feature selection* provided the most consistent results and best appraisals from the goodness of the clustering algorithms. Two clusters provided the best quality clusters in almost all cases. K-means clustering was most likely to provide the best quality clusters. *Multi-Cluster feature selection* provided varying clustering results. With 25% of features, a lower number of clusters seemed to bring better results. On the other hand, when the number of features was increased, the quality of clusters was often better with four to six clusters.

The best quality clustering was generally achieved with the *K-means* algorithm. It often produced the best quality clusters for all feature selection methods, quantities of features and numbers of clusters. *Fuzzy C-means* often produced results that were comparable or slightly inferior to the results provided by K-means. However, Fuzzy C-means often provided a better within-cluster sum of squares metric. *Hierarchical* clustering produced stable results, and clear patterns could be observed. Nevertheless, the results were often outperformed by the K-means and Fuzzy C-means algorithms. The *K-medoids* clustering algorithm often provided poor and inconsistent results. This was largely due to initialization problems that the algorithm had.

The model was decided to be created using Spectral feature selection and K-means clustering. It was decided to set the number of clusters to two, since in the majority of cases it provided the best quality clusters. The optimal number of features was investigated by increasing the number of features and taking note of the changes to the within-cluster sum of squares and Silhouette score metrics. As a result, it was noticed that using 75% of all features provided good quality clusters while keeping the dataset granular.

Lastly, the results of the customer segmentation process were presented. Customers were segmented into four groups: ‘no management’, ‘partner managed’, ‘segment 1’ and ‘segment 2’. The ‘no management’ segment contained companies, where neither the partner nor the company managed the company. The ‘partner managed’ segment contained companies, where the partner potentially managed the company. ‘Segment 1’ and ‘segment 2’ were created by clustering active companies. The main differentiating factors between the clusters were the size of the companies, and how active the companies and partners were.

The case company has plans to improve the current data collection framework in the future. As a result, the quantity of features and data would increase. However, this might cause the feature selection algorithms to slow down or not work at all. To overcome this problem, further research could be conducted on how to create PySpark compatible feature selection methods. By utilizing parallel computing, the problem posed by the increase in data could be overcome. In addition, it could potentially enable the use of wrapper and embedded feature selection methods.

Variations and potentially improved versions of the algorithms implemented in this thesis could be used to better the results. For example, there are more modern variations of the Fuzzy C-means algorithm that could have potentially produced better results. In addition, various process mining algorithms could be used to further enhance clustering. By analysing and mapping out more software use patterns, additional features and descriptive values could be derived for each user segment.

# 6. References

Alelyani, S., Tang J. & Liu H. 2013. Feature selection for clustering: a review. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.8115&rep=rep1&type=pdf

Apache Software Foundation. 2019a. Apache Hadoop. [Website] Retrieved 29.8.2019. Available: https://hadoop.apache.org

Apache Software Foundation. 2019b. Apache Hadoop. [Website] Retrieved 29.8.2019. Available: https://spark.apache.org

Azkaban. 2019. Open-source Workflow Manager. [Website] Retrieved 29.8.2019. Available: https://azkaban.github.io

Baesens, B. 2017. Customer Journey Analytics. Available: https://search-proquest-com.ezproxy.cc.lut.fi/docview/1906048575?pq-origsite=primo

Bernard, G. & Andritsos, P. 2017b. CJM-ex: Goal-oriented Exploration of Customer Journey Maps using Event Logs and Data Analytics. Available: https://pdfs.semanticscholar.org/db1a/bbd4764924d1c052fb62a8aeb87eab219f12.pdf

Bernard, G., & Andritsos, P. 2017a. A Process Mining Based Model for Customer Journey Mapping. Available: http://ceur-ws.org/Vol-1848/CAiSE2017_Forum_Paper7.pdf

Bezdek, J. 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Available: https://www.researchgate.net/publication/233932672_Pattern_Recognition_With_Fuzzy_Objective_Function_Algorithms

Cai, D., Zhang, C. & He, X. 2010. Unsupervised Feature Selection for Multi-Cluster Data. Available: http://www.cad.zju.edu.cn/home/dengcai/Publication/Conference/2010_KDD-MCFS.pdf

Calinski, T. & Harabasz, J. 1974. A Cluster Separation Measure. Available: https://www.researchgate.net/publication/233096619_A_Dendrite_Method_for_Cluster_Analysis

Canny, J. Introduction to Data Science. 2018. Available: https://bcourses.berkeley.edu/courses/1377158/files/62044813/download?verifier=U8wZ1kcGxU7ZqpRQ7f0A28UP2uhjTZ4EzOKgWJ58&wrap=1

Databricks. 2019. Pyspark. [Website] Retrieved: 28.8.2019. Available: https://databricks.com/glossary/pyspark

Davies, D. & Bouldin, D. 1979. A Cluster Separation Measure. Available: https://www.researchgate.net/publication/224377470_A_Cluster_Separation_Measure

Demšar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic, M., Možina, M., Polajnar, M., Toplak, M., Staric, A., Stajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M. & Zupan, B. Orange: Data Mining Toolbox in Python. 2013. Available: http://www.jmlr.org/papers/volume14/demsar13a/demsar13a.pdf

Dy, J. & Brodley, C. 2004. Feature selection for unsupervised learning. Available: http://www.jmlr.org/papers/volume5/dy04a/dy04a.pdf

Fichter, D. 2015. Customer Journey Mapping. Online Searcher. Available: https://search-proquest-com.ezproxy.cc.lut.fi/docview/1702929689/fulltextPDF/E2E5F8C3C4B54B49PQ/1?accountid=27292

Greenacre, M. Hierarchical cluster analysis. 2008. [Online Article] Retrieved 23.8.2019. Available: http://www.econ.upf.edu/~michael/stanford/

Gutierrez-Osuna R. 2019. L11: sequential feature selection. [Online article] Retrieved 20.8.2019. Available: http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf

Hamerly, G. & Elkan, C. 2002. Alternatives to the k-means algorithm that find better clusterings. Available: http://people.csail.mit.edu/tieu/notebook/kmeans/15_p600-hamerly.pdf

Herrera, F. Nápoles, G., Arco, G. & Vanhoof, K. 2018. Customer Segmentation Using Multiple Instance Clustering and Purchasing Behaviors. Available: https://www.researchgate.net/publication/327794279_Customer_Segmentation_Using_Multiple_Instance_Clustering_and_Purchasing_Behaviors

Kashwan, K. 2013. Customer Segmentation Using Clustering and Data Mining Techniques. Available: http://www.ijcte.org/papers/811-E365.pdf

Kaufman, L. & Rousseeuw, P. 1987. Clustering by means of medoids. Available: https://www.researchgate.net/publication/243777819_Clustering_by_Means_of_Medoids

Kotsiantis, S., Kanellopoulos, D. & Pintelas P. 2006. Data Preprocessing for Supervised Learning. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.8413&rep=rep1&type=pdf

Lamont, J. 2016. Web analytics: Insights into the customer journey. Available: https://search-proquest-com.ezproxy.cc.lut.fi/docview/1844587635/?pq-origsite=primo

Li, J. 2017. Challenges of Feature Selection for Big Data Analytics. Available: https://ieeexplore.ieee.org/document/7887649

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R., Tang, J. & Liu J. 2016. Feature Selection: A Data Perspective. Available: https://arxiv.org/abs/1601.07996

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R., Tang, J., & Liu, H. 2019. Scikit-feature. [Website] Retrieved: 28.8.2019. Available: http://featureselection.asu.edu/index.php

Liu, H. & Motoda, H. 2007. Computational Methods of Feature Selection. Available: https://pdfs.semanticscholar.org/4541/7badfeb7b9f974b706f265afd66a94bf9060.pdf

Liu, Y., Zhongmou L., Hui X., Xuedong G. & Junjie W. 2010. Understanding of Internal Clustering Validation Measures. Available: http://datamining.rutgers.edu/publication/internalmeasures.pdf

Macedo, F. 2017. Theoretical Foundations of Forward Feature Selection Methods based on Mutual Information. Available: https://arxiv.org/abs/1701.07761

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. Available: https://projecteuclid.org/euclid.bsmsp/1200512992

McKinney, W. 2013. Python for Data Analysis. Available: https://www.cin.ufpe.br/~embat/Python%20for%20Data%20Analysis.pdf

Park, S. & Jun, C. A Simple and Fast Algorithm for K-medoids Clustering. 2009. Available: http://isiarticles.com/bundles/Article/pre/pdf/79087.pdf

Pedregosa R. et al. 2011a. Scikit-learn: Machine Learning in Python. Available: https://scikit-learn.org/stable/modules/feature_selection.html

Pedregosa R. et al. 2011b. Scikit-learn: Machine Learning in Python. [Website] Retrieved: 28.8.2019. Available: https://scikit-learn.org/stable/#

Pohjalainen, J. & Räsänen, O. & Kadioglu, S. 2013. Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits. Computer Speech & Language. Available: https://users.aalto.fi/~jpohjala/publications/csl14stc.pdf

Pratt, M & White, S. 2019. What is a business analyst? A key role for business-IT efficiency. [Online article] Retrieved: 24.9.2019. Available: https://www.cio.com/article/2436638/project-management-what-do-business-analysts-actually-do-for-software-implementation-projects.html

Project Jupyter. 2019. Jupyter. [Website] Retrieved: 28.8.2019. Available: https://jupyter.org

Python Software Foundation. 2019. The Python Tutorial. [Website] Retrieved: 27.8.2019. Available: https://docs.python.org/3/tutorial/

Qiuru, C., Ye, L., Haixu, X., Yijun L. & Guangping, Z. Telecom customer segmentation based on cluster analysis. 2012. Available: http://ieeexplore.ieee.org.ezproxy.cc.lut.fi/stamp/stamp.jsp?tp=&anumber=6309069&isnumber=6308775

Renaud, K., and Gray, P. 2004. Making sense of low-level usage data to understand user activities. Available: https://interruptions.net/literature/Renaud-SAICSIT04.pdf

Rousseeuw, P. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Available: https://www.researchgate.net/publication/222451107_Rousseeuw_PJ_Silhouettes_A_Graphical_Aid_to_the_Interpretation_and_Validation_of_Cluster_Analysis_Comput_Appl_Math_20_53-65

Rudnitckaia, J. 2015. Process Mining. Data science in action. Available: https://pdfs.semanticscholar.org/7e6b/ac4de2adda5f5e993644126d8cbdf6839f39.pdf

Saghafi, L. 2017. A Tutorial on Clustering Algorithms. [Online Article] Retrieved 22.8.2019. Available: https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html

Saitta, S., Raphael, B. & Smith I. 2008. A Comprehensive Validity index for Clustering. Available: https://pdfs.semanticscholar.org/d933/0f539f2c96c43a2967ab9b1db8cdbfc7f572.pdf

SciPy Developers. 2019a. Scientific Computing Tools for Python. [Website] Retrieved: 28.8.2019. Available: https://www.scipy.org/about.html

SciPy Developers. 2019b. NumPy. [Website] Retrieved: 28.8.2019. Available: https://numpy.org

SciPy Developers. 2019c. Python Data Analysis Library. [Website] Retrieved: 28.8.2019. Available: https://pandas.pydata.org/index.html

SciPy Developers. 2019d. Matplotlib. [Website] Retrieved: 28.8.2019. Available: https://matplotlib.org

Thinsungnoen T., Kaoungku N., Durongdumronchai P., Kerdprasop K. & Kerdprasop N. 2015. The Clustering Validity with Silhouette and Sum of Squared Errors. Available: https://pdfs.semanticscholar.org/8785/b45c92622ebbbffee055aec198190c621b00.pdf

Tripathi, S. 2018. Approaches to Clustering in Customer Segmentation. Available: https://www.sciencepubco.com/index.php/ijet/article/download/16505/7053

Tukey, W. 1993. Exploratory Data Analysis. Available: https://www.stat.berkeley.edu/~brill/Papers/EDASage.pdf

Väätäjä, H. 2016. Opportunities and needs for logged usage data analytics of complex industrial systems. Available: https://uta-fi.academia.edu/HeliV%C3%A4%C3%A4t%C3%A4j%C3%A4/Papers

Wang, S., Tang, J. & Liu, H. 2016. Feature Selection. Available: http://www.public.asu.edu/~swang187/publications/Feature_Selection.pdf

Ward, J. 1963. Hierarchical Grouping to Optimize an Objective Function. Available: https://grid.cs.gsu.edu/~wkim/index_files/papers/ward63.pdf

Whittle, P. 1983. Prediction and regulation by linear least-square methods. Available: https://www.sciencedirect.com/science/article/pii/0169207086900439?via%3Dihub

Xiaofei, H., Deng, C. & Partha, N. 2006. Laplacian Score for Feature Selection. Available: http://papers.nips.cc/paper/2909-laplacian-score-for-feature-selection.pdf

Xie, K. 2012. Using log mining to analyze user behavior on search engine. Available: https://link-springer-com.ezproxy.cc.lut.fi/article/10.1007%2Fs11460-011-0177-4

Yamaguchi, K. 2013. Leveraging Advertising Data For Behavioral Insights. [Online article] Retrieved 24.9.2019. Available: https://marketingland.com/leveraging-advertising-data-for-behavioral-insights-46467

Zhang, J. 2004. K-means Cluster Analysis. [Online Article] Retrieved 26.8.2019. Available: http://www.cs.uky.edu/~jzhang/CS689/PPDM-Chapter3.pdf

Zhao Z., Liu H. 2007. Spectral Feature Selection for Supervised and Unsupervised Learning. Available: http://www.public.asu.edu/~huanliu/papers/icml07.pdf

Zhao, A. & Liu, H. 2011. Spectral Feature Selection for Data Mining. Available: https://doc.lagout.org/Others/Data%20Mining/Spectral%20Feature%20Selection%20for%20Data%20Mining%20%5BZhao%20%26%20Liu%202011-12-14%5D.pdf

*Commercial sources:*

Amazon. 2019a. Cloud computing with AWS. [Website] Retrieved 29.8.2019. Available: https://aws.amazon.com/what-is-aws/

Amazon. 2019b. Amazon EC2. [Website] Retrieved 29.8.2019. Available: https://aws.amazon.com/ec2/

Amazon. 2019c. Amazon S3. [Website] Retrieved 29.8.2019. Available: https://aws.amazon.com/s3/

Amazon. 2019d. Amazon Redshift. [Website] Retrieved 29.8.2019. Available: https://aws.amazon.com/redshift/

Redash.io. 2019. What's Redash? [Website] Retrieved 29.8.2019. Available: https://redash.io/help/faq/general