Master's thesis
Master's Programme in Data Science

Customer Segmentation with Subscription-based Online Media Customers

Henri Haatanen

April 9, 2022

Supervisor(s): Professor Keijo Heljanko

Examiner(s): Professor Keijo Heljanko
PhD Ilari Maarala

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Henri Haatanen

Työn nimi — Arbetets titel — Title

Customer Segmentation with Subscription-based Online Media Customers

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	April 9, 2022	43

Tiivistelmä — Referat — Abstract

In the modern era, using personalization when reaching out to potential or current customers is essential for businesses to compete in their area of business. With large customer bases, this personalization becomes more difficult, thus segmenting entire customer bases into smaller groups helps businesses focus better on personalization and targeted business decisions. These groups can be straightforward, like segmenting solely based on age, or more complex, like taking into account geographic, demographic, behavioral, and psychographic differences among the customers. In the latter case, customer segmentation should be performed with Machine Learning, which can help find more hidden patterns within the data.

Often, the number of features in the customer data set is so large that some form of dimensionality reduction is needed. That is also the case with this thesis, which includes 12802 unique article tags that are desired to be included in the segmentation. A form of dimensionality reduction called feature hashing is selected for hashing the tags for its ability to be introduced new tags in the future.

Using hashed features in customer segmentation is a balancing act. With more hashed features, the evaluation metrics might give better results and the hashed features resemble more closely the unhashed article tag data, but with less hashed features the clustering process is faster, more memory-efficient and the resulting clusters are more interpretable to the business. Three clustering algorithms, K-means, DBSCAN, and BIRCH, are tested with eight feature hashing bin sizes for each, with promising results for K-means and BIRCH.

ACM Computing Classification System (CCS):
Information systems → Information systems applications → Data mining → Clustering
Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning → Cluster analysis
Computing methodologies → Machine learning → Machine learning algorithms → Feature selection

Avainsanat — Nyckelord — Keywords

Customer segmentation, Machine learning, Feature hashing, Clustering

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

This thesis focuses on using feature hashed article tags for customer segmentation on paid online media subscribers. The resulting segmentation of this thesis will be used for creating knowledge about the customer base and more than likely for personalization in the future. Feature hashing is performed on the article tags into 5, 10, 20, 50, 100, 200, and 500 hashed features. Three unsupervised learning algorithms, K-means [18] as centroid-based clustering, DBSCAN [12, 23] as density-based clustering, and BIRCH [28] as hierarchical clustering are tested for the segmentation and results are evaluated with Silhouette coefficient [22], Dunn index [10, 11], and Davies-Bouldin index [9].

Customer segmentation is used for discovering homogeneous groups from a heterogeneous mass of customers. Segmentation can be done with for example using behavior data like previous purchases or sociodemographic data like age. Uncovering these segments creates knowledge about the customer base and dividing the customer base into smaller groups enables more personalized marketing for the customers. Customer segments can also be used to analyze profit potentials within each segment and whether more or less money should be used to maintain or grow said segments.

Customer segmentation can be done with hard clustering or soft clustering, meaning one customer can belong to only one cluster or multiple clusters, respectively. This thesis uses hard clustering for creating knowledge about the customer base, overlapping segments might bring more confusion since the segments counts would not add up to the customer base. Soft clustering is beneficial in, for example, online advertising with thousands of soft clusters, where one might belong to a pet food cluster and a badminton cluster and get ads based on both.

Chapter 2 covers background and related work. Chapter 3 covers the data available for segmentation and feature engineering. Chapter 4 introduces the three algorithms tested for the segmentation. Chapter 5 introduces three evaluation metrics used for testing. Chapter 6 covers the results of the experiments and comparisons between them. Chapter 7 covers the conclusions of the thesis.

2. Background

The concept behind modern customer segmentation was introduced in 1956 by Smith as discovering homogeneous groups from a heterogeneous mass of markets [25]. Although the original paper was about market segmentation (segmenting the entire available market instead of just the customer base), the same basic principles also hold for customer segmentation. In market segmentation, the business segments the market and decides to focus on pleasing one or a few segments. Conversely in customer segmentation, the business segments the customer base and tries to identify segments for personalization or business strategy reasons.

Customer segments can be identified by examining geographic, demographic, behavioral, and psychographic differences among the customers as described by Kotler [16]. Geographic data refers to features such as region, city and climate, and demographic data means features such as age, gender, family size, and income. Demographic data can also include problematic features such as race and religion that should never be included as features in customer segmentation. Behavioral data focuses on the known behaviors of the customer such as usage activity, pageviews, and email sign-ups. Psychographic differences are rarely used in customer segmentation solely based on the difficulty of obtaining such data. Psychographic data includes features much deeper than demographic data such as interests, values, personalities and lifestyles. Sometimes the difference between behavioral and psychographic data is somewhat vague as is the case with pageviews. Pageviews are considered behavior data but pageviews on the same topic of articles can be considered psychographic data as the customer is interested in that topic.

Segmentation can be done on the current customer base, or on customers that have been part of the customer base within some time window. Both of these versions tells a different story to the business and creates knowledge about the selected group of customers. This thesis focuses on the second option in customers that have been part of the customer base within a time window from September 2021 to January 2022. This will hopefully help the business understand its current customer base better while also understanding which of the segments is "leaking" customers and has the highest churn rates, which is defined as the percentage of customers lost compared to the size

of the customer base within a month.

2.1 Feature Hashing

Feature hashing, otherwise known as the "hashing trick", is a low-memory and rapid way of vectorizing features [26]. Compared to vectorizers that first build a hash table based on the inputs of the training data, feature hashing applies a hash function directly to the features to determine the column values. This enables faster and more memory-efficient hashing with the downside of having no way of knowing the original inputs as no hash table is saved.

As hash collisions are inevitable when the number of features is much lower than the original input, a signed hash function is used to downsize the effect of the different features colliding to the same hashed feature. The sign function assigns each input string as positive or as negative, but deterministically for the string inputs so each of the same inputs is always positive or negative. In the case that the sign function is different between two colliding features, an expected value of zero is produced instead of two times the value, helping machine learning algorithms handle the collided features.

This thesis uses feature hashing with article tags. There are 12,802 unique tags in the data set used and some form of dimensionality reduction is required. The most important reason for using feature hashing with article tags compared to some other forms of dimensionality reduction is that new article tags can be created after the initial segmentation. When an article is written about a new topic, a new tag might be created. This tag can now be feature engineered with the same feature hashing into the same features. This property to be able to be introduced new tags is a big advantage when compared to other commonly used dimensionality reduction methods, where such property does not exist like Uniform Manifold Approximation and Projection (UMAP) [19] or Principal Component Analysis (PCA) [20].

2.2 Related Work

Chen et al. [7] performed customer segmentation on Recency, Frequency and Monetary values (RFM) using K-means, K-medoids and DBSCAN clustering methods. In this context, Recency means the time since the last purchase, Frequency means the frequency of purchases and Monetary value refers to the total spent money. RFM is closely related to Frequency, Recency and Volume (FRV) which will be discussed in Section 3.2 and used in this thesis. Chen et al. [7] found out that the best clustering with regards to evaluation metrics came from using two clusters, but only DBSCAN was able to find the "Golden class", which is the segment with the highest average

RFM. With four clusters the evaluation metrics were worse but all the algorithms found the "Golden class", meaning that using only the evaluation metrics was not the best solution for business purposes.

Hossain compared customer segmentation using centroid-based and density-based clustering algorithms [15]. They used K-means as the centroid-based algorithm and DBSCAN as the density-based algorithm. Although finding the distinct border for density-based algorithms can be difficult, Hossain was able to generate segments with both K-means and DBSCAN with their data set and claims density-based algorithms are worth considering applying.

Senuma experimented with K-means clustering feature hashed documents from a data set called twenty Newsgroups [24]. They chose six document classes and drew randomly 100 documents from each class. Senuma implemented feature hashing on the words of the documents and performed K-means clustering on the hashed features. Feature hashing proved high memory efficiency with memory usage being only 3.5% of the original memory while the score calculated from recall and precision was still good. In some cases, the hashed version performed even better than the original data, perhaps since the original data might have more local minimas and the hashed features might be more smooth and a lower minima is found.

3. Data

In this chapter, we cover the data available for the segmentation and describe the feature engineering performed with the data. The order data consists of 27,809 active paid subscription-based orders and free trials of a Finnish media brand that have been active for at least a day between September 2021 and January 2022. This data is combined with click-data size for the users of the media website from September 2021 to January 2022. The click-data consists of around 14 million clicks for all users in the orders data.

3.1 Order Details

Each order has basic order details that can be used in the segmentation. These include for example order length, paid price, number of orders, number of payments and the order channel. Order details also has some variables that can not be used in segmentation like order number, order type and product type. Order number can not be used because it is unique for each order, and conversely order type and product type is the same for each order.

3.2 Frequency, Recency and Volume

Frequency, Recency and Volume (FRV) is a measure of activity for the user on the website. Frequency measures the number of days the user has logged on the website in the last 30 days, Recency is the inverse of the number of days since the user has last logged in and Volume is the square root of the number of clicks the user has performed on the website in the last 30 days. These three variables multiplied together form a single FRV value for each registered user that can be used to estimate the activity of the user on the website. A histogram of the FRV values for the customer base is visualized in Figure 3.1.

As the website has both free and paid articles, FRV can also be measured for both independently. Someone might have a high FRV value within free articles and not

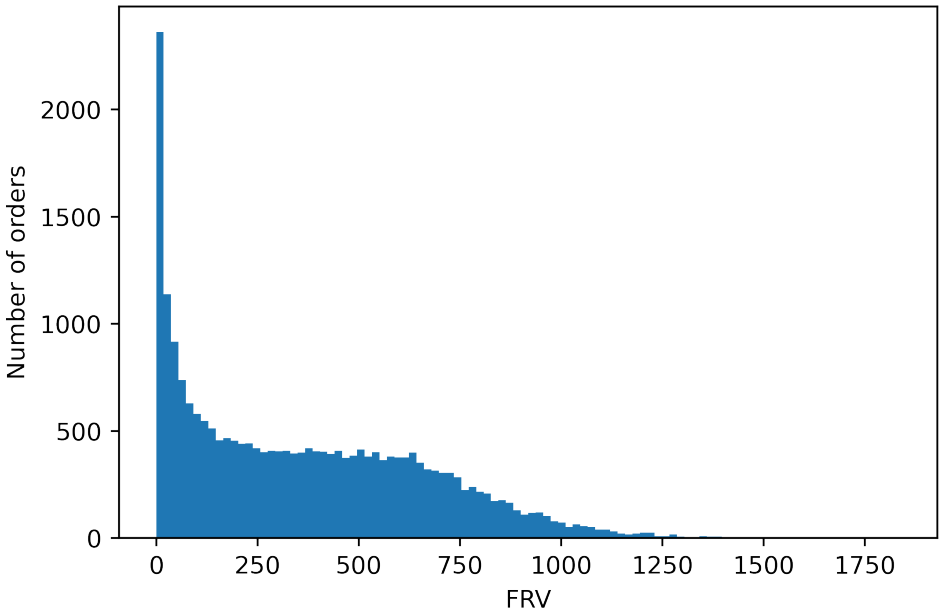

Figure 3.1: Histogram of FRV values for all customers within the order data set.

within paid articles and that might be the reason for churning. This might be due to the customer not finding enough interesting paid content to justify their subscription price, since they could consume all free articles without a subscription. In turn, someone with a fairly high FRV within paid articles but close to zero FRV within free articles might be less likely to churn, because the content they consume is available only with the subscription.

FRV is similar to a more well-known measure called RFM that has previously been used in customer segmentation [5, 7]. FRV is a website usage version of RFM, which consists of Recency and Frequency but volume is replaced by Monetary value. In this segmentation, Monetary value is taken into account as a variable called paid price.

3.3 Website Usage Details

The data includes each click performed on the website. This data can be aggregated to more meaningful variables for the orders like what page the user usually redirects from, usage time of day and what type of device the user has. As these categorical variables only have a few possible values they can be implemented as relative frequencies among the possible values.

3.3.1 Primary Categories

Each page on the website has a primary category and possible tags. There are 60 unique primary categories in the data set. Primary categories are feature engineered as relative frequencies of each user's activity and each category's relative interest making 60 new columns to the data. More specifically, for each user x and category y pair an interest value k_{xy} is calculated in Equation 3.1.

$$k_{xy} = \frac{\text{count}(x \text{ read articles from category } y)}{\max_X(\text{read articles from category } y)} : \frac{\text{count}(x \text{ read articles})}{\max_X(\text{read articles})} \tag{3.1}$$

The scaling in the interest value takes into consideration the general interest of each category and the general activity of each user with smaller categories emphasized among enthusiastic readers. This feature engineering also alters the data from discrete variables to continuous variables, simplifying the clustering as the difficulty of a mixed-data clustering (continuous and discrete variables) is avoided [1].

3.3.2 Article Tags

The article data includes 12,802 unique tags as keywords for the article provided by the journalist. As the number of unique tags is so large, using manual feature engineering similarly to primary categories or website usage details is not advisable. Tags are instead vectorized into features using feature hashing [26] and the resulting features are used in segmentation. This means losing the interpretability of the tags, however, such a large number of tags would be hard to interpret in any case. Using feature hashing enables the introduction of new unique tags never before seen by the feature hashing without having to hash the entire data again. This is due to the property of feature hashing being able to be introduced new values without having to do the entire hashing all over again. Feature hashing the article tag data into matrices of 5, 10, 20, 50, 100, 200 and 500 features is tested Chapter 6.

3.4 Final Data

After removing orders with zero clicks on the website are removed from the data, the final data includes 23,641 orders. The final data has 176 features from areas described in this section. The tag data with 23,641 rows and 12,802 columns is feature hashed into 5, 10, 20, 50, 100, 200 and 500 features and each of these hashed tag data sets is combined with the other 176 features for the experiments.

4. Algorithms

In this chapter three clustering methods are discussed: K-means [18], DBSCAN [12, 23] and BIRCH [28]. These three are selected as each represents a different type of clustering in centroid-based, density-based and hierarchical clustering, respectively. Each method is explained with pseudocode included and related work with regards to customer segmentation is discussed.

4.1 K-means

K-means is a clustering algorithm first proposed for signal processing [18]. The algorithm aims to divide N samples of data into K disjoint clusters C, where each cluster is described by its mean of samples μ, commonly called centroids. K-means algorithm attempts to choose the centroids to minimize the within-cluster sum of squares. The number of clusters K has to be determined before starting the clustering, and methods like the G-means algorithm [13] or Silhouette coefficient [22] can be used to help the determining process.

Determining the global optimum with K-means is NP-hard [2, 8], but K-means always converges to a local optimum. Converging, in this case, means that after n loops of the while loop, the cluster labels remain unchanged compared to the previous loop and the run is completed. Since the clustering is usually quite quick, the algorithm is commonly run with multiple initial conditions and the best local optimum among the results is selected. K-means is deterministic when the initial cluster centroids are the same, even if data is ordered differently. Pseudocode for the algorithm is provided in Algorithm 1.

4.1.1 Customer Segmentation

K-means has been widely used in customer segmentation [4, 7, 27]. K-means is one of the most popular clustering algorithms and in many cases, it is the first algorithm tested when performing clustering. K-means can give a good overall picture of the structure of the data with relatively easy implementation.

Algorithm 1 K-means, algorithm of Lloyd [18]

function K-MEANS(D, K)
  Assign initial random values for μ_1, μ_2, μ_3, ... μ_k.
  while Clusters remain unchanged **do**
    for `n = 1 to N` **do** $d_i \leftarrow \text{argmin}_k ||\mu_k - x_n||$
    end for
    for `k = 1 to K` **do** $\mu_k \leftarrow \text{MEAN}(\{ x_n : z_n = k \})$
    end for
  end while
end function

4.2 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a commonly used density-based clustering algorithm that separates clusters based on density areas [12, 23]. DBSCAN can produce clusters of any shape unlike mean-based clustering algorithms, making DBSCAN more viable when dealing with real-world data. DBSCAN is also robust to outliers, labeling points with no cluster points within ϵ range of it as noise. Compared to K-means requiring the number of clusters as an input, the two input variables for DBSCAN are minimum number of samples $minPts$ and the maximum distance between samples ϵ. Choosing the $minPts$ and ϵ well is critical and methods like OPTICS [3] or the K-nearest neighbor method [21] can be used to determine them. Generally, $minPts$ is chosen as two times the dimensions of the data, as

$$minPts = 2 \cdot n, \tag{4.1}$$

where n is the dimensionality of the data, and ϵ is chosen from the K-distance graph as the elbow of the graph as visualized in Figure 6.1. With too small ϵ, more clusters are created and likely more noise is generated, but with too large ϵ, small clusters merge as larger ones and information is lost.

All the data points in DBSCAN clustering are considered one of three types: Core point, Border point or Noise point. Core points are points with at least $minPts$ number of points within its ϵ neighborhood. This means that each cluster produced by DBSCAN has at least one core point within it. All the unvisited points within the neighborhood are also considered part of the same cluster. Unvisited points are points that the algorithm has not yet processed, as the points are visited in a loop. If other unvisited core points are within the ϵ neighborhood of the first core point considered, also the unvisited points within the ϵ neighborhood of the second core

point are considered part of the same cluster. As the visitation order of the points might affect the computed clusters, DBSCAN is considered nondeterministic if the data is ordered differently. Points that are not core points, but have at least one core point within its ϵ neighborhood are considered border points, and points that are not core points and have no core points within ϵ distance are considered noise points. Pseudocode for DBSCAN is provided in Algorithm 2.

4.2.1 Customer Segmentation

Density-based clustering algorithms like DBSCAN have been found effective for customer segmentation when compared to centroid-based algorithms like K-means clustering [5, 15]. Advantages of using DBSCAN for customer segmentation include not having to determine the number of clusters in advance and robustness to outliers. Moreover, DBSCAN's ability to identify clusters of any shape is beneficial when dealing with real-world data. A drawback of using density-based clustering with customer segmentation is that a density drop is expected to determine the border of the cluster. With real-world data, such a density drop is not always found and the clusters that are desired can be of varying density.

4.2.2 OPTICS

Ordering points to identify the clustering structure (OPTICS) [3] is a density-based clustering algorithm similar to DBSCAN. The biggest upside of using OPTICS is that there is no need to define the ϵ value before clustering. The OPTICS algorithm uses a range of values as ϵ, although a maximum value for ϵ can be defined for runtime efficiency. This means that OPTICS can detect clusters of varying density compared to DBSCAN. The OPTICS algorithm builds a Reachability graph from which the user can cluster the points accordingly.

OPTICS algorithm introduces two more terms in addition to $minPts$ and ϵ introduced in DBSCAN for each point p: Core distance and reachability distance. Core distance for point p is the minimum radius from p so that at least $minPts$ points are within the radius. If the radius is greater than the defined maximum ϵ, the core distance is undefined. As with DBSCAN, points within a defined core distance are called core points. Reachability distance is calculated from one point to another. Reachability from point q to point p is defined as the maximum of the core distance of p and the distance between points q and p. Similarly to core distance, reachability distance is not defined if p is not a core point. Core distance and reachability distance are formally defined in Equations 4.2 and 4.3, respectively, where $N_\epsilon(p)$ is defined as the group of

Algorithm 2 DBSCAN, algorithm of Ester et al. [12, 23]

```
function DBSCAN(D, ϵ, MinPts)
    C = 0
    for each unvisited point P in the data set D do
        mark P as visited
        NeighborPts = regionQuery(P, ϵ)
        if sizeof(NeighborPts) < MinPts then
            mark P as NOISE
        else
            C = next cluster
            expandCluster(P, NeighborPts, C, ϵ, MinPts)
        end if
    end for
end function
function expandCluster(P, NeighborPts, C, ϵ, MinPts)
    add P to cluster C
    for each point P' in NeighborPts do
        if P' is not visited then
            mark P' as visited
            NeighborPts' = regionQuery(P', ϵ)
            if sizeof(NeighborPts') >= MinPts then
                NeighborPts = NeighborPts joined with NeighborPts'
            end if
            if P' is not yet member of any cluster then
                add P' to cluster C
            end if
        end if
    end for
end function
function RegionQuery(P, ϵ)
     return all points within P's ϵ-neighborhood (including F)
end function
```

points within the ϵ neighborhood of point p, including itself.

$$\text{core-dist}_{\epsilon,MinPts}(p) = \begin{cases} UNDEFINED & \text{if } |N_\epsilon(p)| < MinPts \\ MinPts\text{-th smallest distance in } N_\epsilon(p) & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{reach-dist}_{\epsilon,MinPts}(q,p) = \begin{cases} UNDEFINED & \text{if } |N_\epsilon(p)| < MinPts \\ \max(\text{core-dist}_{\epsilon,Minpts}(p), \text{dist}(p,q)) & \text{otherwise} \end{cases} \tag{4.3}$$

4.2.3 HDBSCAN

Hierarchical density-based spatial clustering of applications with noise (HDBSCAN) is a hierarchical version of DBSCAN [6]. HDBSCAN is a successor to DBSCAN and OPTICS. It avoids defining the ϵ value, or even the maximum ϵ value defined in OPTICS. Core distance is defined similarly to OPTICS, but Reachability distance is a symmetric version of the OPTICS version called Mutual Reachability distance. Mutual Reachability distance between points q and p is the maximum of the Core distance of p, Core distance of q and the distance between points p and q.

4.3 BIRCH

Balanced iterative reducing and clustering using hierarchies (BIRCH) is a hierarchical data clustering method that is especially suited for large data sets [28]. BIRCH clusters data points incrementally and dynamically and can usually produce a reasonably good clustering with just one scan of the data, while being able to improve with few additional scans. The main idea behind BIRCH is to first generate a compact summary of the original large data set while retaining as much information as possible using the Clustering Feature Tree (CFT). Then as each leaf of the CFT contains a subcluster, these leaf nodes can be combined with another clustering algorithm more efficiently. BIRCH utilizes agglomerative hierarchical clustering as the secondary clustering method. Similarly to DBSCAN, BIRCH is also considered nondeterministic if the data is ordered differently.

4.3.1 Clustering Feature

Clustering Feature (CF) is the core concept behind BIRCH. Clustering Feature is defined for N data points within a cluster as a triple $CF = (N, \vec{LS}, SS)$, where N is

the number of data points, $\vec{LS}$ is the linear sum of the N data points and SS is the square sum of the N data points. Clustering Features offer efficiency by using a small set of features to represent a larger data set.

4.3.2 Clustering Feature Tree

Clustering Feature Tree (CF Tree) is a height-balanced tree with two parameters: branching factor B and threshold T. Branching factor B determines the maximum number of CF subclusters in each node and threshold T is the maximum diameter requirement that all entries within a leaf must follow. Each non-leaf node consists of no more than B pairs $[CF_i, child_i]$ where CF_i is the Clustering Feature of the i-th child node and $child_i$ a pointer to it. Each leaf node is represented by at most L occurrences of triple $[CF_i, prev, next]$ where $prev$ and $next$ point to the previous and next leaf node respectively for a more efficient scan over all the leaf nodes. Compared to non-leaf nodes, all entries in a leaf node must follow the threshold requirement T. As a node is required to fit in a single memory page, a leaf's node size L is determined by the combination of the available page size P and the Branching factor B determined by the user.

4.3.3 Customer Segmentation

Research papers for customer segmentation with BIRCH are noticeably rarer compared to DBSCAN or K-means, but some work with customer segmentation with BIRCH has been done [14, 17]. The rarity might be explained by BIRCH being best suited for large data sets without categorical variables and thus is not possibly the first choice to test clustering customer data, which likely contains many categorical variables and is rarely in the millions for an average business.

Algorithm 3 BIRCH, algorithm of Zhang et al. [28]

```
function CF Tree(D, B, T)
    Calculate Clustering Features (CF) for all data points d in D
    for each data point d calculated calculated do
        Insert d as a CF to the root CF Tree
        while current node is not a leaf node do
            Merge with the child subcluster with the smallest radius after merging
        end while
        Find the nearest subcluster within the leaf node
        if threshold condition T is not violated then
            Add d to the cluster and update CF triplets
        else
            if there is room to insert d as own cluster then
                Insert d as a single cluster and update CF triplets
            else
                Select two most distant subclusters and divide the subclusters into two
                    groups based on the distance to the two subclusters
                while the parent node has no room to split based on B or
                    root is reached do
                    Split parent node in two and go to their parent node
                end while
                if parent nodes that can be splitted are found then
                    Split parent nodes in two
                end if
            end if
        end if
    end for
end function
```

5. Evaluation

In this chapter, evaluation metrics used for the experiments are described in detail. Evaluating results by unsupervised learning, like clustering, is substantially more difficult compared to supervised learning. In supervised learning, the ground truth is usually known for at least for the test data and plenty of metrics based on the ground truth exist like specificity, sensitivity and mean squared error. For clustering, despite the absence of ground truth, some internal methods for evaluation still exist and a few of those are highlighted in this chapter. Also, the possible difference between the mathematically best clusters based on these metrics and the best clusters based on future business decisions will be discussed.

5.1 Silhouette Coefficient

Mean Silhouette coefficient is an internal evaluation metric for determining the robustness of the clustering [22]. In Silhouette analysis, a Silhouette coefficient is calculated for each point in the data set based on the distance to its cluster and to its closest neighboring cluster. The score ranges from -1 to 1, where a low value indicates poor labeling from the clustering algorithm and a high value indicates good labeling. Values near 0 indicate that the point is on the border of at least two clusters and could belong to either. The Silhouette coefficient can be also used in determining the optimal number of clusters in the data set if the clustering algorithm in question needs that information as an input.

5.1.1 Calculating Silhouette Coefficient

Silhouette coefficient calculation starts using a clustered data set that needs evaluation. For each data point i in the cluster C_I, let

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j) \tag{5.1}$$

be a measure of how well data point i is assigned to its cluster C_I, where $|C(i)|$ is the size of the cluster and $d(i,j)$ the distance between data points i and j. Also, let

$$b(i) = \min_{I \neq J} \left(\frac{1}{|C_J|} \sum_{j \in C_J} d(i,j) \right) \tag{5.2}$$

be a measure of average dissimilarity between the data point i and all data points in the closest neighboring cluster J. Now, the Silhouette coefficient $s(i)$ for data point i is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{if } |C_I| > 1 \tag{5.3}$$

and

$$s(i) = 0, \text{if } |C_I| = 1.$$

Thus, Silhouette coefficient $s(i)$ ranges between -1 and 1.

As Silhouette coefficient is calculated for each point separately, more calculation is needed for obtaining a goodness score for the entire clustered data set. Mean Silhouette coefficient is defined as the mean value over all Silhouette coefficients $s(i)$ for the data points i in the data set D as

$$s(D) = \frac{\sum_{i \in D} s(i)}{|D|} \tag{5.4}$$

and approximates how the data is clustered overall.

This thesis uses mean Silhouette score as one of the evaluation metrics. Euclidean distance is used as the distance metric as it will also be used as the distance metric in the clustering.

5.2 Dunn Index

Dunn index is an internal evaluation metric for determining the robustness of the clustering [10, 11]. Dunn index ranges from 0 to infinity and similarly to the Silhouette coefficient, a larger Dunn index indicates better clustering. Dunn index is defined as the minimum inter-cluster diameter divided by the maximum intra-cluster diameter. As Dunn index takes into consideration the maximum diameter of the clusters, the index might give a worse value for such a clustering where one of the clusters has a high variance while the other clusters are tight. Considering this fact, while Dunn index can be used to determine the optimal number of clusters for algorithms requiring the number of clusters as an input, it should not perhaps be the only indicator.

5.2.1 Calculating Dunn Index

Calculating Dunn index is not straightforward. The calculation starts with determining the intra-cluster diameter metric for the index. Let i and j be data points belonging to the same cluster C_I. Intra-cluster diameter metrics are defined as

$$\Delta_I = \max_{i,j \in C_I} d(i,j) \text{ as the maximum distance,} \tag{5.5}$$

$$\Delta_I = \frac{2}{|C_I|(|C_I|-1)} \sum_{i,j \in C_I, i \neq j} d(i,j) \text{ as the mean distance and} \tag{5.6}$$

$$\Delta_I = \frac{\sum_{i \in C_I} d(i,\mu)}{|C_I|},\ \mu = \frac{\sum_{i \in C_I} i}{|C_I|} \text{ as the mean distance of all points to the centroid.} \tag{5.7}$$

Similarly, the inter-cluster diameter metric between two different clusters has multiple possible variations. Let C_I and C_J be different clusters. Inter-cluster diameter metrics between the clusters are defined as

$$\delta_{IJ} = min_{i \in I, j \in J} d(i,j) \text{ as the single linkage distance,} \tag{5.8}$$

$$\delta_{IJ} = max_{i \in I, j \in J} d(i,j) \text{ as the complete linkage distance,} \tag{5.9}$$

$$\delta_{IJ} = \frac{\sum_{i \in I, j \in J} d(i,j)}{|C_I||C_J|} \text{ as the average linkage distance and} \tag{5.10}$$

$$\delta_{IJ} = d(\mu_I, \mu_J),\ \mu_K = \frac{\sum_{k \in K} k}{|C_K|} \text{ as the centroid linkage distance.} \tag{5.11}$$

With the intra-cluster dimater metrics in Equations 5.5, 5.6, and 5.7 and the inter-cluster diameter metrics in Equations 5.8, 5.9, 5.10, and 5.11 determined, Dunn index for m clusters is defined as

$$DI_m = \frac{\min_{1 \leq I \leq J \leq m} \delta_{IJ}}{\max_{1 \leq k \leq m} \Delta_k}. \tag{5.12}$$

For this thesis, Dunn index is used with the mean distance of all points to the centroid in Equation 5.7 as the intra-cluster diameter metric and centroid linkage distance in Equation 5.11 as the inter-cluster diameter metric. The distance metric is Euclidean distance as it will be used in the clustering.

5.3 Davies-Bouldin Index

Davies-Bouldin index is an internal evaluation metric for determining the robustness of the clustering [9]. Similarly to Dunn index, Davies-Bouldin index ranges from 0 to infinity. Conversely to both Dunn index and Silhouette coefficient, a lower Davies-Bouldin index value indicates better clusters. The index estimates the average similarity between each cluster with its closes neighboring cluster. Compared to Dunn index, Davies-Bouldin index is not affected as drastically when just one cluster has a high variance and Davies-Bouldin index is rather an evaluation metric comparing all clusters with its closest neighboring cluster.

5.3.1 Calculating Davies-Bouldin Index

For calculating the Davies-Boulding index, the same intra-cluster diameter metrics Δ_I and inter-cluster diameter metrics δ_{IJ} as for the Dunn index in Section 5.2.1 can be used. Davies-Boulding index for m clusters is defined as

$$DB = \frac{1}{m}\sum_{I=1}^{m} \max_{I \neq J} \frac{\Delta_I + \Delta_J}{\delta_{IJ}}. \tag{5.13}$$

Similarly to Silhouette coefficient and Dunn index, in this thesis Euclidean distance is used as the distance metric for Davies-Bouldin index as well. Also similarly to Dunn index, Davies-Bouldin index is used with the mean distance of all points to the centroid in Equation 5.7 as the intra-cluster diameter metric and centroid linkage distance in Equation 5.11 as the inter-cluster diameter metric.

5.4 Interpretability

Interpretability plays a large role in successful customer segmentation. While the previously introduced evaluation metrics are a great way of evaluation clustering, the results of a customer segmentation must also have some interpretability for decision-making. For example, the number of clusters must be manageable for business actions and the cluster sizes must be large enough for decisions to be based on them that matter in the big picture. Preferably the cluster differences are such that business actions can be made based on them, but this should be handled already when choosing the data that is clustered.

When presenting the customer segments to the business decision-makers, interpretability plays a key role in how the segmentation is perceived. As most decision-makers might not be experts in the field of machine learning, presenting the segments

as understandable groups of customers that the decision-makers might already recognize helps to tell the story better. This might make the segmentation more likely to be used in decision-making in the future.

6. Results

In this chapter, the experiments are carried out, the results of the experiments are covered and comparisons between the results are discussed. Customer segmentation is tested with 0, 5, 10, 20, 50, 100, 200 and 500 feature sets from feature hashing the original 12,802 distinct tags. Three algorithms are tested: K-means, DBSCAN and BIRCH. Each algorithm is tested with each feature set from feature hashing the tags. Each algorithm, feature hashing and feature set combination is run 10 times over with slightly different initial conditions depending on the algorithm to avoid randomness caused by the initial conditions. For example, with K-means, a feature set of size 200 from feature hashing and $k = 4$ is run 10 times with different initial centroids. The best score with the Silhouette coefficient is selected from these 10 runs.

6.1 K-means

As K-means requires the number of clusters as an input, it is tested with the value of 2, 3, 4 and 5 clusters for each of the tag feature sets. Each parameter combination is run ten times with different initial cluster centroids to reduce the randomness of the results. From the ten runs, the best clustering based on Silhouette coefficient is selected.

Results for K-means clustering with increasing k from 2 to 5 clusters are in Tables 6.1, 6.2, 6.3 and 6.4, respectively. With each k, more tag features included result in a better Silhouette coefficient and Dunn index. The only exception for the highest Silhouette coefficient is with $k = 4$, where 200 tag features has a value of 0.147986 and 500 tag features has the second highest score of 0.112559. As this is the only exception, it might be due to randomness. For Davies-Boudlin index, where a lower score is better than a higher one, the best score for $k = 2$ is achieved with 500 tag features. For every other k, the best Davies-Bouldin index is achieved with 200 tag features.

Without using tag features, the best Silhouette coefficient is achieved with $k = 3$, the best Dunn index with $k = 2$ and the best Davies-Bouldin index with $k = 5$. In this respect, there seem to be no distinct clusters in the data as determining the optimal k is not trivial with these metrics. With tag features included, the best result with

each metric is achieved with $k = 2$ and 500 tag features. Controversially, even though that is the best result out of the three metrics, more than two segments are needed out of the customer segmentation in a real business case. In all experiments, clustering performs better regarding all three evaluation metrics with the increasing number of tag features. Using five to twenty tag features already achieves better results from the evaluation metrics compared to not using any while retaining more interpretability with most of the features not coming from hashed variables like with the 200 and 500 feature sets.

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.040059	0.012970	5.394044	163s
5	0.046428	0.008635	4.939160	166s
10	0.054590	0.016993	4.342015	167s
20	0.068074	0.020334	3.800444	168s
50	0.129171	0.025999	2.934559	168s
100	0.189285	0.032917	2.563176	195s
200	0.249661	0.048056	2.297429	202s
500	0.307343	0.084165	2.280315	250s

Table 6.1: Experimental results with K-means ($k = 2$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.080873	0.003360	4.431149	119s
5	0.088908	0.005446	3.996263	119s
10	0.107413	0.009075	3.461698	121s
20	0.122683	0.008789	3.183790	123s
50	0.094491	0.018664	3.223542	132s
100	0.124532	0.023113	2.925882	132s
200	0.156570	0.037952	2.702455	156s
500	0.190881	0.057037	2.777074	213s

Table 6.2: Experimental results with K-means ($k = 3$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.044132	0.000960	3.950845	86s
5	0.071489	0.007326	3.692432	118s
10	0.065487	0.008870	3.887458	123s
20	0.067505	0.008266	3.588587	125s
50	0.094659	0.018237	3.010459	130s
100	0.111363	0.024832	2.767905	142s
200	0.147986	0.036618	2.739884	166s
500	0.112559	0.047381	3.449514	236s

Table 6.3: Experimental results with K-means ($k = 4$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.043133	0.000960	3.282948	120s
5	0.071567	0.007326	3.123208	122s
10	0.053211	0.007266	3.638320	122s
20	0.070667	0.009912	3.420308	128s
50	0.076531	0.016902	3.199783	138s
100	0.067169	0.021568	3.181996	146s
200	0.085420	0.031412	3.037760	183s
500	0.103534	0.047381	3.426314	238s

Table 6.4: Experimental results with K-means ($k = 5$).

6.2 DBSCAN

DBSCAN requires initial values for ϵ and $minPts$. K-nearest neighbor method is used for determining these values [21]. The data has 176 dimensions without the tags. For each experiment, the value of $minPts$ with t number of tag features is

$$minPts = 2 \cdot (176 + t),$$

as defined in Function 4.1. Value for ϵ is selected from K-nearest neighbor graph as the "elbow" of the graph as defined previously in Section 4.2. As an example, this elbow is visualized for 5 tag features in Figure 6.1 where the optimal value is $\epsilon = 16.5$. As the elbow is sharp and located near the maximum value of 23,641 at 23,300, the clustering will not likely produce distinct clusters, which is also the case here. The graphs for each of the different tag features are very similar and the results from the method are displayed in Table 6.5.

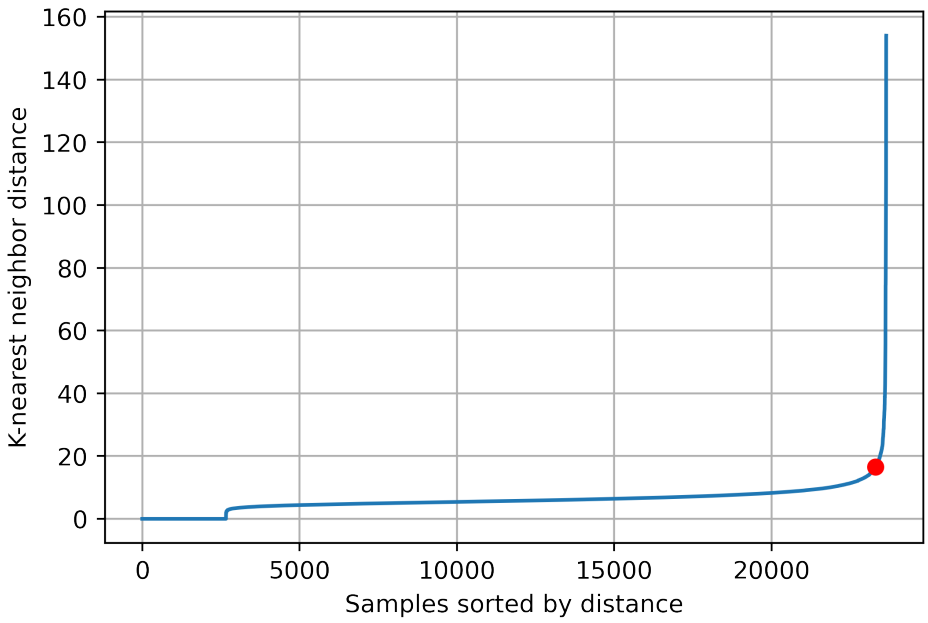

Figure 6.1: K-nearest neighbor method example with 5 tag features and optimal value $\epsilon = 16.5$.

DBSCAN is run ten times for each number of tag features with slightly different values for $minPts$, within a range of 50, and ϵ, within a range of one, from the optimal values displayed in Table 6.5. The results for all the tag features are displayed in Table 6.6, where it is fairly obvious that DBSCAN does not work for this experiment, as the elbow is located in the far bottom right corner. Although the evaluation metrics show promising results, DBSCAN produces only one cluster with the other cluster being the "noise" cluster. The data set likely has so much overlap that a density-based algorithm can not distinguish areas of high density.

As DBSCAN with the optimal $minPts$ and ϵ did not produce clusters, the algorithm was also tested with many drastically different initial values for $minPts$ and ϵ. Despite all these efforts, DBSCAN was unable to produce multiple clusters except for some clusters of size less than twenty units.

Tag features	$minPts$	ϵ
0	352	16.4
5	362	16.5
10	372	16.6
20	392	17.2
50	452	18.5
100	552	21.9
200	752	26.9
500	1352	41.1

Table 6.5: Optimal values for $minPts$ and ϵ from K-nearest neighbor method.

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.626655	0.014204	8.657057	247s
5	0.599496	0.014279	8.718333	246s
10	0.572874	0.007064	8.702586	326s
20	0.604694	0.012146	8.657516	324s
50	0.604884	0.018971	7.094666	297s
100	0.599080	0.021351	4.638849	305s
200	0.577713	0.049231	3.032385	351s
500	0.554129	0.085466	2.137706	457s

Table 6.6: Experimental results with DBSCAN.

6.3 BIRCH

As BIRCH requires the number of clusters as an input, the experiments are run with the same parameters as with the K-means. BIRCH is run ten times with each parameter combination while shuffling the order of the data points on each run. BIRCH is sensitive to the order of the data input, thus we generate ten different results from where the optimal one is selected. Similarly to DBSCAN and K-means, the optimal clustering based on Silhouette coefficient is selected.

Results for BIRCH clustering with increasing k from 2 to 5 clusters are in Tables 6.7, 6.8, 6.9 and 6.10, respectively. Generally, the more tag features are included, the better the Silhouette coefficient and Dunn index are. Despite this, the best overall score for Silhouette coefficient and Davies-Bouldin index is achieved with $k = 2$ and 200 tag features with the scores of 0.276220 and 2.336433, respectively. With the Dunn index, the best score is achieved with $k = 4$ and 500 tag features scoring 0.053714.

Interestingly, while generally more tag features seem to indicate a better Silhouette score, with $k = 3$, the best score is achieved with 20 tags. For the same conditions also Davies-Bouldin index is fairly good at 3.431294 being the third best score with the $k = 3$ test after 500 and 200 tags.

Davies-Bouldin index is interesting with BIRCH. For $k = 2$ and 3, more tag features indicate a better score, but for $k = 4$ and 5, the result is the opposite. For runtimes, BIRCH is by far the slowest of the three algorithms, although DBSCAN can not be compared justly. Two clusters with 500 tag features took almost 40 minutes for ten iterations. The rest of the tests with BIRCH did not outperform the slowest of the competing algorithms.

When excluding the tags from the clustering, the results are as scattered as with K-means clustering. The best score for the Silhouette coefficient is achieved when $k = 3$, the best score for the Dunn index when $k = 2$, and the best score for the Davies-Bouldin index with $k = 4$.

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.019968	0.013278	6.981064	962s
5	0.080078	0.017595	5.877116	948s
10	0.061751	0.015733	5.660380	973s
20	0.074864	0.021151	4.802301	990s
50	0.076732	0.020809	3.977928	1105s
100	0.226333	0.037524	2.581129	1231s
200	0.276220	0.045463	2.336433	1631s
500	0.169205	0.047058	2.714882	2229s

Table 6.7: Experimental results with BIRCH ($k = 2$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.082839	0.007794	4.998483	821s
5	0.010847	0.006115	4.926454	821s
10	0.037318	0.006760	4.543373	846s
20	0.131779	0.016468	3.431294	888s
50	0.101156	0.014488	3.657805	967s
100	0.113119	0.020197	3.620746	1082s
200	0.075254	0.022752	3.371427	1388s
500	0.087866	0.026029	3.295253	1810s

Table 6.8: Experimental results with BIRCH ($k = 3$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.014738	0.006479	2.976180	779s
5	0.013576	0.005719	4.427011	817s
10	0.021995	0.006760	4.188930	838s
20	0.045468	0.007766	4.194280	878s
50	0.089739	0.014488	3.625134	962s
100	0.068459	0.020841	3.680564	1082s
200	0.061873	0.023399	3.412109	1377s
500	0.148624	0.053714	4.016621	1815s

Table 6.9: Experimental results with BIRCH ($k = 4$).

Tag features	Silhouette	Dunn	Davies-Bouldin	Runtime
0	0.023738	0.006479	3.142627	820s
5	0.013703	0.006277	3.554501	815s
10	0.022241	0.007303	3.363554	846s
20	0.032783	0.010944	3.734834	871s
50	0.041363	0.013775	3.820094	950s
100	0.071521	0.020841	3.377201	1062s
200	0.099580	0.025743	3.461046	1365s
500	0.112533	0.033803	3.905745	1843s

Table 6.10: Experimental results with BIRCH ($k = 5$).

6.4 Comparison

As discussed in Section 6.2, DBSCAN does not perform well with this data set. The data set likely has too much overlap for DBSCAN to produce multiple clusters, which might not be the case for all future experiments. The overlap hypothesis is supported by Silhouette scores with all the tests with K-means and BIRCH, where scores are relatively close to zero indicating overlapping data. As DBSCAN failed to produce comparable results to K-means and BIRCH, the rest of this section will focus on the comparison between K-means and BIRCH, with an emphasis on the three and four cluster experiments.

Comparing the Silhouette coefficient and Dunn index from clustering with K-means to clustering with BIRCH show that both are usable for the task with three clusters. These comparisons are best seen in Table 6.11 and Figures 6.2, 6.3, 6.4 and

6.5, which combine results from Tables 6.2 and 6.8. As seen in Figure 6.2, K-means is better in terms of Silhouette coefficient with 5, 10, 100, 200, and 500 tags and BIRCH is better with 0, 20 and 50 tags with the three clusters, meaning that for some combinations of the number of clusters and the number of tag features one performs slightly better than the other. Still, K-means seems much more consistent with the scores while BIRCH has a lot of variation between similar numbers of hashed features.

With Dunn index in Figure 6.3, K-means is better with 10, 50, 100, 200, and 500 features and BIRCH with 0, 5, and 20. It seems that for Dunn index, K-means starts off worse with 0 clusters, but scales better when the number of hashed features is increased. For the Davies-Bouldin index in Figure 6.4, K-means performs better in all the tests with three clusters and overall better in all but a few tests. Also, for runtime in Figure 6.5, K-means is significantly faster when compared to BIRCH.

Tag	Sil K	Sil B	Dunn K	Dunn B	DB K	DB B	R K	R B
0	0.080873	0.082839	0.003360	0.007794	4.431149	4.998483	119s	821s
5	0.088908	0.010847	0.005446	0.006115	3.996263	4.926454	119s	821s
10	0.107413	0.037318	0.009075	0.006760	3.461698	4.543373	121s	846s
20	0.122683	0.131779	0.008789	0.016468	3.183790	3.431294	123s	888s
50	0.094491	0.101156	0.018664	0.014488	3.223542	3.657805	132s	967s
100	0.124532	0.113119	0.023113	0.020197	2.925882	3.620746	132s	1082s
200	0.156570	0.075254	0.037952	0.022752	2.702455	3.371427	156s	1388s
500	0.190881	0.087866	0.057037	0.026029	2.777074	3.295253	213s	1810s

Table 6.11: Experimental results with K-means and BIRCH ($k = 3$). Number of tag features (Tag), Silhouette coefficient with K-means (Sil K), Silhouette coefficient with BIRCH (Sil B), Dunn index with K-means (Dunn K), Dunn index with BIRCH (Dunn B), Davies-Bouldin index with K-means (DB K), Davies-Bouldin index with BIRCH (DB B), runtime with K-means (R K) and runtime with BIRCH (R B).

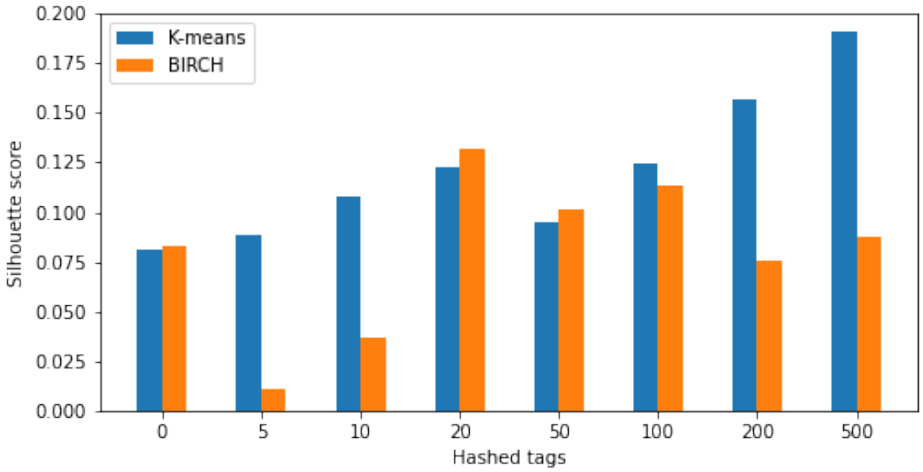

Figure 6.2: Silhouette score comparison between K-means and BIRCH with 3 clusters.

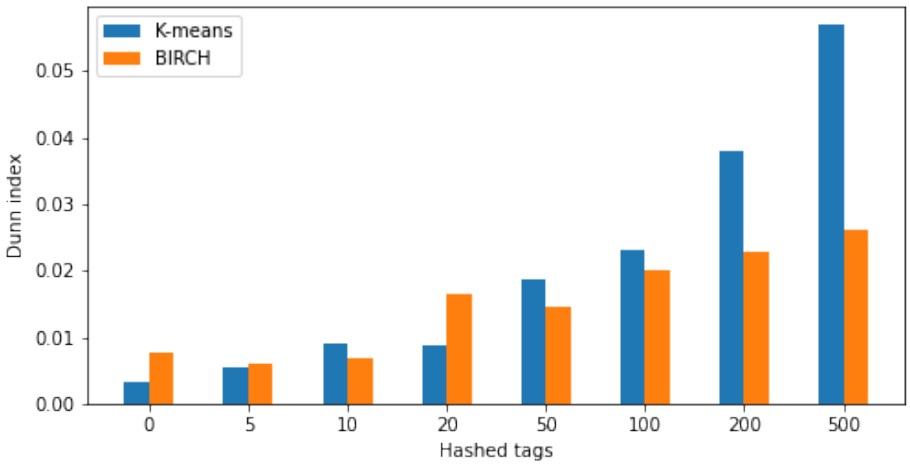

Figure 6.3: Dunn index comparison between K-means and BIRCH with 3 clusters.

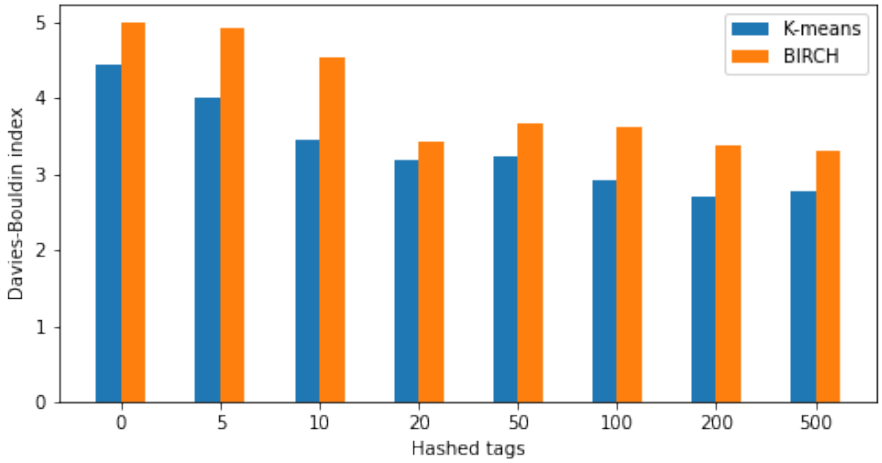

Figure 6.4: Davies-Bouldin index comparison between K-means and BIRCH with 3 clusters.

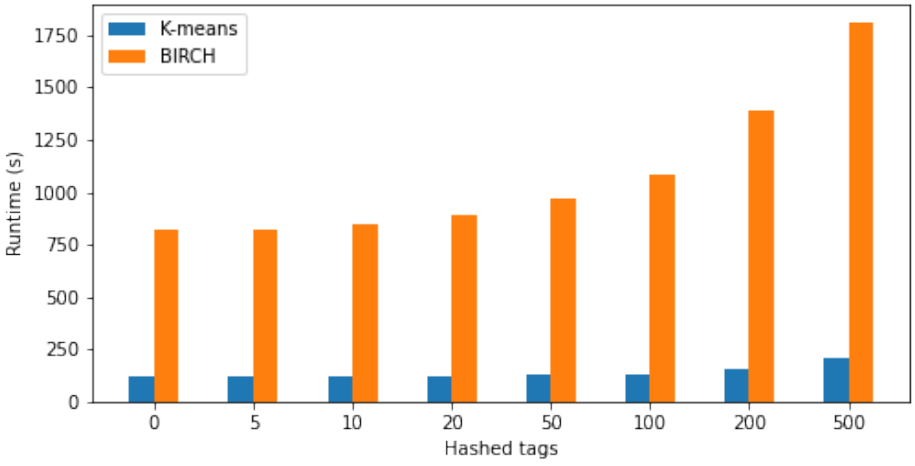

Figure 6.5: Runtime comparison between K-means and BIRCH with 3 clusters.

With four clusters in Table 6.12 and in Figures 6.6, 6.7, 6.8, and 6.9, which combine the previous results from Tables 6.3 and 6.9, K-means outperformed BIRCH with regards to the Silhouette coefficient and Davies-Bouldin index in all cases but with 500 tag features. It seems that the run with BIRCH and 500 tag features is an outlier, especially when looking at the Dunn index in Figure 6.7. The results from the Dunn index are in line with the results with three clusters, where BIRCH is initially better with a lower number of tag features but scales worse when increasing the number of hashed features. The only exception is the run with 500 hashed features where the Dunn index for BIRCH jumps significantly compared to the run with 200 features. Based on the results from evaluation metrics combined with the result that BIRCH is 7-10 times slower to run, I would prefer K-means over BIRCH for customer segmentation with hashed article features.

Tag	Sil K	Sil B	Dunn K	Dunn B	DB K	DB B	R K	R B
0	0.044132	0.014738	0.000960	0.006479	3.950845	2.976180	86s	779s
5	0.071489	0.013576	0.007326	0.005719	3.692432	4.427011	118s	817s
10	0.065487	0.021995	0.008870	0.006760	3.887458	4.188930	123s	838s
20	0.067505	0.045468	0.008266	0.007766	3.588587	4.194280	125s	878s
50	0.094659	0.089739	0.018237	0.014488	3.010459	3.625134	130s	962s
100	0.111363	0.068459	0.024832	0.020841	2.767905	3.680564	142s	1082s
200	0.147986	0.061873	0.036618	0.023399	2.739884	3.412109	166s	1377s
500	0.112559	0.148624	0.047381	0.053714	3.449514	4.016621	236s	1815s

Table 6.12: Experimental results with K-means and BIRCH ($k = 4$). Number of tag features (Tag), Silhouette coefficient with K-means (Sil K), Silhouette coefficient with BIRCH (Sil B), Dunn index with K-means (Dunn K), Dunn index with BIRCH (Dunn B), Davies-Bouldin index with K-means (DB K), Davies-Bouldin index with BIRCH (DB B), runtime with K-means (R K) and runtime with BIRCH (R B).

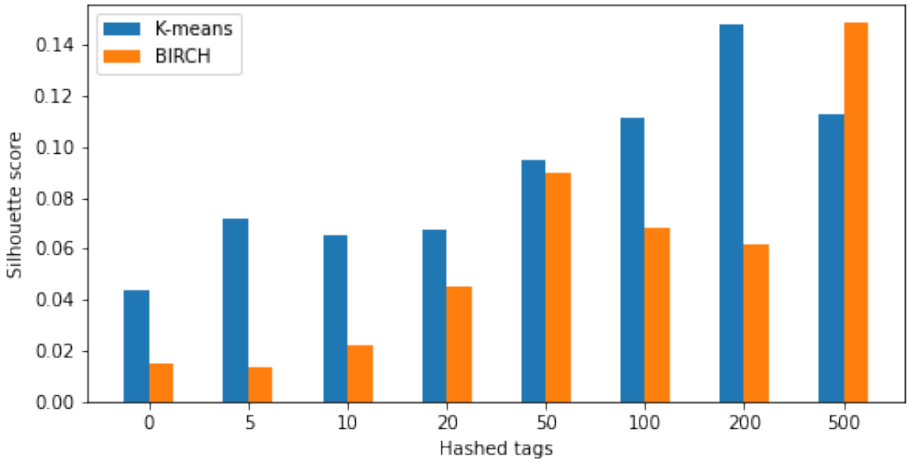

Figure 6.6: Silhouette score comparison between K-means and BIRCH with 4 clusters.

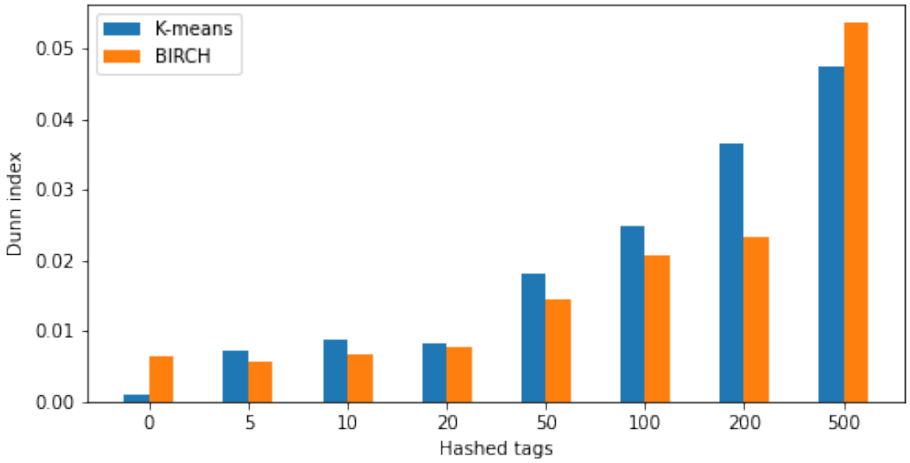

Figure 6.7: Dunn index comparison between K-means and BIRCH with 4 clusters.

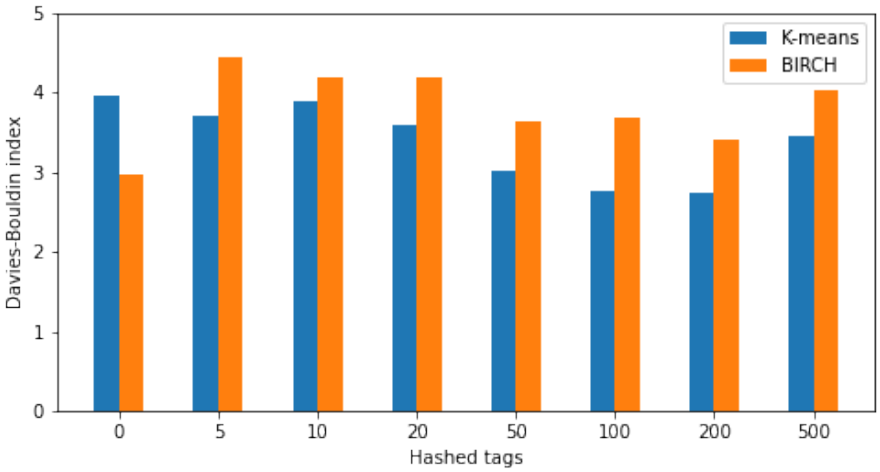

Figure 6.8: Davies-Bouldin index comparison between K-means and BIRCH with 4 clusters.

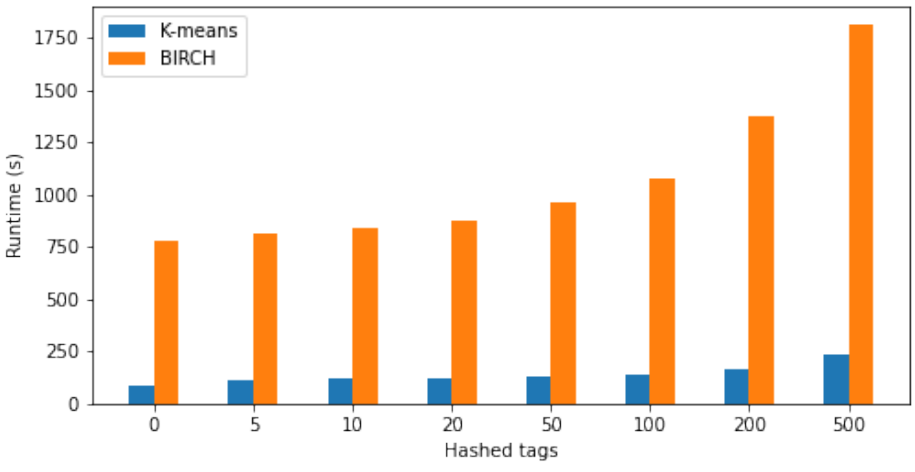

Figure 6.9: Runtime comparison between K-means and BIRCH with 4 clusters.

7. Conclusions

This thesis focused on customer segmentation with hashed features from 12,802 unique article tags using a method called feature hashing. One centroid-based, one density-based, and one hierarchical clustering algorithm as K-means, DBSCAN, and BIRCH, respectively, was used for experimentation with 0, 5, 10, 20, 50, 100, 200, and 500 features from feature hashing the article tags for each. Results were discussed based on the Silhouette coefficient, Dunn index, Davies-Bouldin index and interpretability.

Using tags as feature hashed features in customer segmentation is found functional with K-means and BIRCH. Experiments with DBSCAN were not performing as expected, which was possibly due to the data being too overlapping and without distinct borders. With K-means and BIRCH, reducing 12,802 unique tags into just five to twenty features is already an improvement compared to not using the tags at all. Using the tags as 200 or 500 features shows the best results with regards to the evaluation metrics, but interpreting customer segments with features mostly comprised of hashed features is more difficult. The evaluation metric results with just the five to twenty hashed features outperformed compared to when hashed features were not used. This is the most important result from this experiment as those clusters retain more interpretability for use in business.

Subjectively speaking, the clusters with five to twenty hashed features are as interpretable as the clusters without any hashed features and much more interpretable than the clusters with 500 features. Since the evaluation metrics were mostly in favor of using five to twenty tag features compared to not using any, hashed tag features are recommended. Even though each data set is different and the same results do not hold for every case, using feature hashing with article tags should be considered when such data is available.

Bibliography

[1] A. Ahmad and S. S. Khan. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*, 7:31883–31902, 2019.

[2] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009.

[3] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 49–60. ACM Press, 1999.

[4] M. Aryuni, E. Didik Madyatmadja, and E. Miranda. Customer segmentation in XYZ bank using K-means and K-medoids clustering. In *2018 International Conference on Information Management and Technology (ICIMTech)*, pages 412–416, 2018.

[5] R. S. Brahmana, F. Mohammed, and K. Chairuang. Customer segmentation based on RFM model using K-means, K-medoids, and DBSCAN methods. *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, 11(1):32–43, 2020.

[6] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data*, 10(1):5:1–5:51, 2015.

[7] D. Chen, S. Sain, and K. Guo. Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing and Customer Strategy Management*, 19:197–208, 2012.

[8] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *IEEE Trans. Inf. Theory*, 55(7):3229–3242, 2009.

[9] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[10] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[11] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[12] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press, 1996.

[13] G. Hamerly and C. Elkan. Learning the K in K-means. In *Advances in Neural Information Processing Systems 16*, pages 281–288. MIT Press, 2003.

[14] M. Hassan and M. Tabasum. Customer profiling and segmentation in retail banks using data mining techniques. *International journal of advanced research in computer science*, 9(4):24–29, 2018.

[15] A. S. Hossain. Customer segmentation using centroid based and density based clustering algorithms. In *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*, pages 1–6, 2017.

[16] P. Kotler. *Marketing management*. Pearson Italia Spa, 2007.

[17] J. Li, K. Wang, and L. Xu. Chameleon based on clustering feature tree and its application in customer segmentation. *Ann. Oper. Res.*, 168(1):225–245, 2009.

[18] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[19] L. McInnes and J. Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018.

[20] K. Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[21] N. Rahmah and I. S. Sitanggang. Determination of optimal epsilon (eps) value on DBSCAN algorithm to clustering data on peatland hotspots in Sumatra. *IOP Conference Series: Earth and Environmental Science*, 31:012012, 2016.

[22] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[23] E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems*, 42(3):19:1–19:21, 2017.

[24] H. Senuma. K-means clustering with feature hashing. In *Proceedings of the ACL 2011 Student Session*, pages 122–126, 2011.

[25] W. R. Smith. Product differentiation and market segmentation as alternative marketing strategies. *Journal of marketing*, 21(1):3–8, 1956.

[26] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 1113–1120. ACM, 2009.

[27] L. Ye, C. Qiu-ru, X. Hai-xu, L. Yi-jun, and Y. Zhi-min. Telecom customer segmentation with k-means clustering. In *2012 7th International Conference on Computer Science Education (ICCSE)*, pages 648–651, 2012.

[28] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In H. V. Jagadish and I. S. Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114. ACM Press, 1996.