University of Zurich[UZH]

Simon Frischknecht
Student ID number: 14-736-391

Determining the Optimal Number of Vowel Clusters in a Wide Range of Fundamental Frequencies using Unsupervised Learning

Master Thesis

Supervision

Prof. Dr. Martin Volk
Prof. Dr. Volker Dellow
Dr. Thayabaran Kathiresan

Department of Informatics
Department of Computational Linguistics

Date of submission: 01.08.2021

Contents

Abstract

Vowel detection is an important field of speech recognition. In this thesis, we focus on clustering, an unsupervised machine learning technique, and evaluate how these methods recognize vowel groups for different fundamental frequencies (f_o). We analyze the algorithms from a mathematical and computational point of view. The implementation results for different f_o levels up to 1 kHz are described and visualized. We use several internal and external cluster validation criterions to evaluate the outcomes of the clustering, because they are often needed to find the optimal cluster values. We show that certain external validation methods can recover the true number of vowel groups, independent of the f_o level, while internal validation methods struggle finding the correct number of groups.

Zusammenfassung

Die Erkennung von Vokalen ist ein wichtiges Gebiet in der Spracherkennung. In dieser Arbeit betrachten wir Clustering, eine unüberwachte Methode des maschinellen Lernens. Wir evaluieren, wie diese Methoden Vokalgruppen für unterschiedliche Grundfrequenzen erkennen. Wir analysieren diese Algorithmen von einer mathematischen und rechnerischen Perspektive. Die Resultate für verschiedene Grundfrequenzen bis zu einem kHz werden erläutert und visualisiert. Wir verwenden verschiedene interne und externe Cluster Validierungskriterien, um die Resultate zu evaluieren, da diese häufig benötigt werden, um eine optimale Clusteranzahl zu finden. Wir zeigen, dass gewisse externe Validierungskriterien die wahre Anzahl von Vokalgruppen erkennen können, unabhängig von der Grundfrequenz. Interne Validierungskriterien haben hingegen Schwierigkeiten, die korrekte Anzahl von Vokalgruppen zu finden.

Nomenclature

$\mathbb{R}$	The set of real numbers
$\mathbb{R}^d$	The set of real numbers in d-dimensions
n	The number of observations
k	The number of clusters
O	Algorithmic Big-O notation
$\|x\|$	ℓ_2 norm of x
$\nabla f(x)$	Gradient of function $f : \mathbb{R}^d \to \mathbb{R}$ at x
$\partial f(x)$	Partial derivative of the function $f : \mathbb{R}^d \to \mathbb{R}$ at x
$X \sim P$	A random variable X follows distribution P
$\langle x, y \rangle$	Inner or dot product of x and y
log	The natural logarithm

List of Figures

Introduction

With increasing fundamental frequency (f_o), vowels become less intelligible for humans [30]. Further, unsupervised learning can be seen as a related concept of how humans acquire phonemes [53]. Using the k-means clustering method to find groups without supervision in a speech corpus of isolated Standard German vowels, [30] shows that this machine learner finds an optimal number of mel-frequency cepstral coefficients (MFCCs) and signal bandwidth.
In this thesis, we use several clustering techniques and try to figure out how these methods recognize groups of vowels. We are interested if clustering algorithms find a similar solution as humans do, or if they are less sensitive to the f_o level. Of most interest is the behavior of the algorithms for high f_o levels around 1 kHz, where humans achieve a striking recognition performance of the three corner vowels /a i u/, while the performance for the non-corner vowels goes down, for some vowels even to chance level.
The structure of this thesis is as follows: In Chapter 2, we describe the vowel speech corpus and how we process the data by using the feature representation of MFCCs. These are used as the input for the machine learning algorithms. We introduce in Chapter 3 the concept of clustering as a branch of machine learning and analyze several clustering algorithms in detail. The goal is to provide a profound overview of these methods. In Chapter 4, we describe cluster validation criterions, where some of these utilize the true class labels and some do not. We need these validation methods to evaluate the results of the clustering algorithm. In Chapter 5, we implement the described clustering algorithms and validation criterions, and also analyze and visualize the results. Finally, we give some concluding remarks and an outlook for future applications in Chapter 6.

Speech Data and MFCC

2.1 Vowel Speech Data

In this thesis, we work with a dataset containing short recordings of eight isolated steady-state Standard German vowels /i y e ϕ ε a o u/ [30]. The data comes from a larger corpus [34]. These vowels are recorded by four professional female actresses at a wide range of f_o levels and three vocal efforts (low, medium, and high). f_o is the lowest frequency in a periodic signal [46]. The standard measurement unit of frequency is hertz, which is the number of cycles of a signal per second. Below, we describe another unit of frequency, frequently used in the context of speech data, called mel. We restrict the f_o range in our work to 10 levels (220, 330, 440, 523, 587, 659, 698, 784, 880, 988 Hz), similar to the work of [17]. The highest f_o level of 1046 Hz in the corpus is ignored because there is not enough data available needed by certain clustering algorithms. The recordings are produced without using any esthetic style, so that the vowels should be as intelligible as possible.

2.2 Mel-Frequency Cepstral Coefficients

MFCCs are a feature representation of a speech signal based on several transformations. It is the most used representation for speech data and is the standard in many modern speech recognition systems [46]. In this thesis, we use MFCCs as the data input for the cluster algorithms. In the following, we explain the steps to create this feature representation.

Programming Language and Analysis Framework

To create the MFCCs, we use Python 3.7.5 and the music and audio signal processing package librosa, version 0.8.0 [35]. librosa contains many functions to process signals, like filter-bank generation, computing spectrograms, feature extraction and manipulation, tempo estimation, and visualization.

Windowing

Given a speech signal, it is extracted over a short time period, usually 20 milliseconds, because we assume the signal is stationary only over a short period of time [42]. An often-used window function for these short time periods is Hamming or Hanning to taper the speech signal at the boundaries. The result is called a windowed frame or chunk. These frames usually overlap for a few milliseconds.

Discrete Fourier Transform

To extract the spectral information from the speech signal, that is, the frequency components of the signal, a discrete Fourier transform (DFT) is applied to every frame [42]. This means we map the signal from the time domain into the frequency domain. The DFT is mathematically defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp\left(\frac{-j2\pi nk}{N}\right), \tag{1}$$

where $x(n)$ is the signal in the time domain, j is the imaginary unit, defined as $j^2 = -1$, N is the length of the signal, k is the frequency variable, and $n \in \mathbb{N}$. The standard way to compute the DFT is by using the fast Fourier transform (FFT), an efficient algorithm that has a time complexity of only $O(n\log(n))$ [28].

Squared Magnitude

Next, we transform the signal $X(k)$ by using only the squared magnitude of it, multiplied by the weighting function $H_m(k)$:

$$s(m) = \sum_{k=0}^{N-1} \left(|X(k)|^2 H_m(k)\right), \tag{2}$$

where $0 \leq m \leq M-1$ and M is the total number of weighting filters.

Mel Spectrum

In a further step, the signal is mapped from the hertz scale to the mel scale. A mel is a unit that reflects how the human ear perceives frequencies, that is, in a non-linear way that can be logarithmically approximated. This makes the mel scale

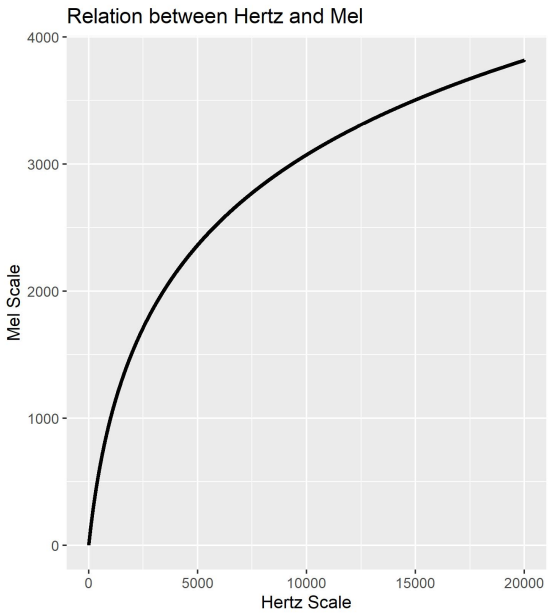

Figure 1: **Different frequency scales.** Logarithmic relationship between the traditional hertz scale and mel scale.

more appropriate for speech data than the classical hertz scale [4]. The reference point between hertz and mel is defined so that 1000 hertz corresponds to 1000 mel. The mel scale is defined as [57]:

$$f_{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{3}$$

where f is the frequency in hertz. f_{Mel} can be approximated with a linear relationship to the hertz scale below 1 kHz and logarithmically above 1 kHz. Figure 1 graphically representation Equation (3).

Discrete Cosine Transform

To decorrelate it, a discrete cosine transform (DCT) is applied to the signal [42]. The DCT is related to the DFT, but in contrast to the DFT, the DCT only uses real numbers. Typically, the type-II DCT is applied to create the MFCCs:

$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m))\cos\left(\frac{\pi n(m-0.5)}{M}\right), \tag{4}$$

where $n = 0, \ldots, C-1$ and C is the number of MFCCs.

Delta and Delta-Delta Coefficients

We can interpret these created coefficients as a static representation. For speech recognition applications, the first and second derivatives of the MFCCs are often created as well. These are then referred to as delta or speed coefficients and delta-delta or acceleration coefficients, respectively. It is typical to use 13 MFCCs for speech applications, although depending on the literature, various numbers of MFCCs are preferred. For instance, [30] shows that for vowel recognition using unsupervised machine learning, 5 is an optimal number for the MFCCs. Assuming we have created 13 MFCCs in the last step, adding the delta and delta-delta coefficients, there are 39 coefficients in total. We use the generated MFCCs for the clustering task of the vowel speech data.

In the next section, we describe what clustering is, why we utilize it, and explain the mathematical principles of several selected algorithms. The goal is to provide a sound overview of the foundations of clustering.

Clustering

3.1 Introduction

In this thesis, we focus on learning algorithms that work on a given dataset without any supervision. Concretely, we analyze and use clustering algorithms. These are unsupervised machine learning techniques that have the goal of partitioning a given dataset into different groups, also called clusters [27]. Unsupervised learners have in common that they work with datasets containing no labels, also called ground truth. Other examples of techniques in this branch of machine learning are dimensionality reduction, density estimation, and anomaly detection. In contrast to unsupervised learning, supervised learning deals with algorithms that work with labelled datasets. A labelled dataset is for instance a collection of images of animals, where each image is annotated with the name of the animal. Usually, supervised learning is divided into regression and classification methods. The third subfield of machine learning is called reinforcement learning, where agents act in an environment, getting feedback in form of rewards [49]. Applications of reinforcement learning are agents acting in a simulated gaming environment or the training of a self-driving car.

A loose definition of clustering is that observations in the same group should be as similar as possible, and observations in different groups should be as dissimilar from each other as possible [27]. To define what similarity and dissimilarity means is a big challenge in cluster analysis and depends on the context. One reason for this difficulty is the absence of labels or rewards [49]. This makes it hard to measure the success of a learner, in contrast to supervised learning, where the ground truth is available. Because of this, variety of clustering algorithms have been developed in the last decades.

The following example from [49] makes this difficulty more clear. Given four groups of points that should be clustered into two groups. It is possible to cluster these data points into two groups as illustrated in Figure 2, among other potential solutions. There is no clear way to decide which clustering result is more appropriate. This

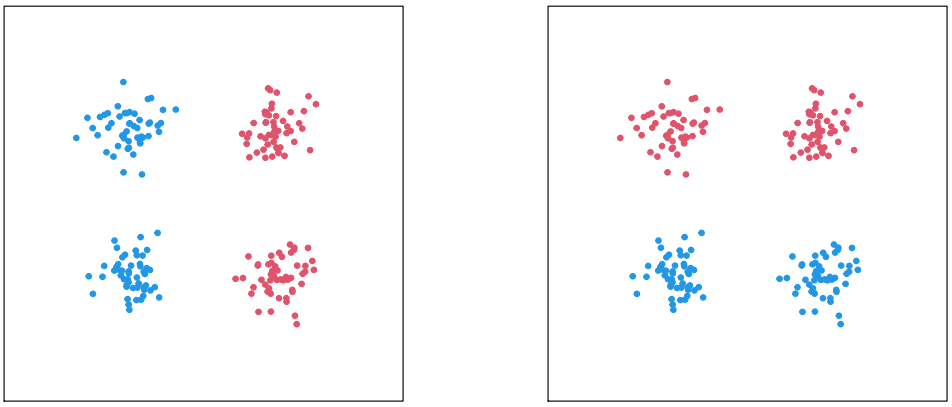

Figure 2: **Data points clustering into two groups.** Left: Horizontal clustering. Right: Vertical clustering. Both solutions are valid, neither the left nor the right solution is preferable.

may seem like an artificial example, but similar situations frequently appear in real-world applications, although in the context of higher dimensional problems.
A more formal definition of clustering is according to [49]: Given a clustering function F and a set of input data X with a dissimilarity or distance function $d : X \times X \to \mathbb{R}_+$ over X, where d is symmetric, that is, $d(x, x') = d(x', x)$, and $d(x, x) = 0 \; \forall x \in X$. Often, the triangle inequality is also fulfilled. Certain clustering algorithms require to define the parameter k that stands for the number of clusters. The resulting output of the clustering is a hard partition of X into k subsets $C_1, \ldots, C_k$ such that $\bigcup_{i=1}^{k} C_i = X$ and $C_i \cap C_j = \emptyset \; \forall i \neq j$. However, there are also clustering methods where the group assignments are probabilistic. We then refer to them as soft clustering.
A clustering function F should according to [31] optimally have three properties:

- Scale invariance. For any $\alpha > 0$, input set X, dissimilarity function d, and $(\alpha d)(x, y) \stackrel{\text{def}}{=} \alpha d(x, y)$, it should hold that $F(X, d) = F(X, \alpha d)$. Written out in words, the clustering should be independent with respect to the units of the distance measurements.

- Richness. For a given dataset X and every partition $C = (C_1, \ldots, C_k)$: $\exists d$ over X such that $F(X, d) = C$. This means that the clustering result is completely determined by the distance function d.
- Consistency. By reducing the within-cluster distances as well as increasing the between-cluster distances, the result of the clustering does not change.

However, it has been shown by [31] that no such function F exists that fulfills all these properties. This is called the impossibility theorem for clustering. Therefore, no algorithm presented in this thesis has all these desirable properties but is optimal only for specific tasks. It is not possible to talk about a universal best method for clustering.
Scientific fields, where clustering techniques frequently find applications, are image segmentation, document clustering, information retrieval, grouping customers, and analyzing genome data [26].
In the following sections, we explain several common clustering techniques that we use for the implementation of the vowel speech data.

3.2 k-Means

One of the most known and oldest clustering algorithms is k-means [21]. It works on the design matrix X, also called input or feature matrix, and tries to solve the following optimization problem:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 , \tag{5}$$

where k is the number of clusters, S_i is a disjoint set or cluster, and μ_i is the mean vector or centroid of cluster i [5]. This means the k-means algorithm has the goal of minimizing a cost function, sometimes referred to as objective or loss function, that is based on the squared ℓ_2-norm, also known as squared Euclidean distance [49]. In this process, the feature space is partitioned into k different sets $S_1, \ldots, S_k$ with $\bigcup_{i=1}^{k} S_i = X$ and $S_i \cap S_j = \emptyset \; \forall i \neq j$. It turns out that solving Equation (5) is NP-hard. Formulated slightly differently, it is often computationally infeasible to solve such a problem exactly. Therefore, an approximation method has to be used

to find a reasonable solution. Algorithm 1, based on the work of [37], shows the steps of the k-means implementation to solve Equation (5) approximately.
The user has to specify three parameters to apply the k-means algorithm: The number of clusters k, the distance measure, and the initialization approach [25].

Algorithm 1: k-means Algorithm

1. Initialize k cluster centers $\mu_1, \ldots, \mu_k$;
2. Repeat until converged:
   (a) Assign each data point to its closest cluster center: $z_i = \arg\min_k ||x_i - \mu_k||_2^2$
   (b) Update each cluster center by computing the mean of all points assigned to it: $\mu_k = \frac{1}{N_k} \sum_{i:z_i=k} x_i$

3.2.1 Initialization and Iterations

The cluster centers of k-means have to be initialized somehow. A common approach is to select k observations of the dataset at random. These points will then be defined as the centroids of the clusters [37]. Another possibility to select k initial values as cluster centers is the k-means++ method, invented in 2007. This method has the goal of distributing the cluster centers in the dataset as widely spread as possible. The first point will be chosen uniformly at random. The next point is selected with probability proportional to the distance squared to the point's cluster center that is closest, while ignoring the already selected first cluster center. The remaining $k - 2$ cluster centers are selected in the same way.
Next, the algorithm works iteratively to find k partitions of the dataset. Every data point is assigned to the closest cluster center μ_i. Then, the new cluster center μ_i is computed, using the arithmetic mean of the data points in the i'th cluster. These two steps are repeated and after some iterations, the algorithm usually converges, as the cluster centers do not change anymore. The resulting partition is the approximate solution of the k-means clustering problem.

3.2.2 Properties of k-Means

Finding only local optima is an important property of k-means. A local optimum means the algorithm has not found the best possible solution, called global optimum. A local optimum is not necessarily a poor outcome, but often the global optimum is the desired outcome. The local optimum can coincide with the globally best solution, but this does not have to be the case. It is therefore recommendable to run the algorithm several times with different initializations and compare the learned partitions.

As already explained, the number of clusters k has to be specified by the user before running the algorithm. This is probably the biggest challenge for k-means, as it is not straightforward to ascertain the right number of clusters, assuming such a number exists. Domain knowledge of the machine learning engineer can help to define k. Many other clustering algorithms require to define the number of clusters as well. Examples are Gaussian mixture models, hierarchical clustering, and spectral clustering, discussed in detail in the next sections.

Because the Euclidean norm is used, k-means is not able to handle categorical variables. This, however, is not an issue in the context of MFCCs, as they consist of continuous data.

k-means is a non-probabilistic clustering method. Hence, no distributional assumptions are made. This contrasts with Gaussian mixture models, explained in the next section, where every cluster is modelled by using a normal distribution.

There are several assumptions made by k-means. For example, using the ℓ_2-norm can be inappropriate in the presence of outliers, as the distances between the observations and the cluster centers are squared. Other metrics like the ℓ_1-norm may be more suitable in such situations. An implicit property that the k-means algorithm assumes is the convex form of the clusters. Non-convex shaped groups, like for instance the half-moon dataset, are overwhelming for k-means. A simple example that illustrates this problem can be found in Figure 3. Clustering techniques like spectral clustering or density-based methods are more appropriate for such cluster forms. Further, expecting that there are k clusters in a dataset, is also an assumption that has to be justified. To figure out the correct number of clusters k can be seen, as already mentioned, as one of the hardest parts in the

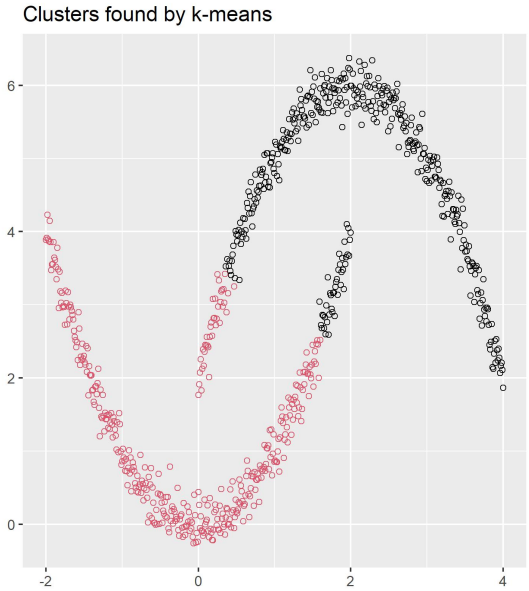

Figure 3: **k-means solution for half-moon data.** The solution found by k-means is clearly suboptimal, because k-means expects convex-shaped clusters. This dataset violates this assumption.

k-means procedure. This is also true for many other clustering algorithms.

3.2.3 Computational Aspects and Applications

In general, k-means efficiently handles large datasets, containing thousands of observations. In the case of very large datasets, with hundreds of thousands or millions of observations, or where the number of dimensions is large, computing often takes much time. To speed up the computation of the Euclidean distances of all data points to the cluster centers in every iteration, tree-based data structures [5] or the triangle inequality are used [13].

As a basic technique, k-means finds application in the context of lossy data compression, where it is known as vector quantization [3]. k-means also has usage in image segmentation [5]. Further, it is a frequently applied strategy to initialize the parameters of Gaussian mixture models.

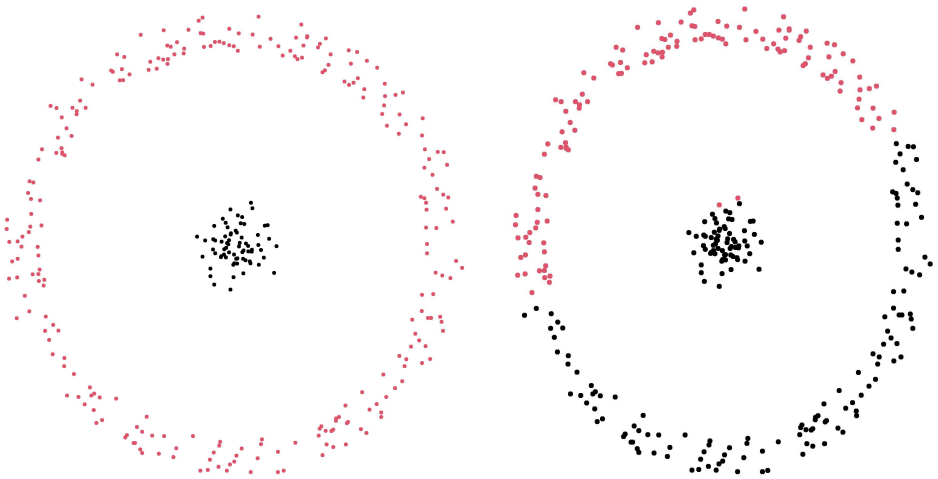

Figure 4: **Non-linear separable groups.** Two groups of data points that are not linearly separable. Left: Kernel k-means can make a meaningful clustering. Right: Clustering solution found by k-means with $k = 2$.

3.2.4 X-Means

X-means is an alternative to k-means that avoids defining the number of clusters k [40]. The user only determines the maximal number of clusters to be examined. X-means then finds the optimal k by minimizing the Akaike Information criterion (AIC) or the Bayesian Information criterion (BIC) [26]. We explain these information criterions in detail in the section about Gaussian mixture models.

3.2.5 Kernel k-Means

An extension of k-means is to utilize a kernel function to perform the clustering in a transformed feature space [18]. This feature space is possibly infinite-dimensional. Because the inner products between the observations are used, the computation is done by employing the kernel function in the input space. This approach makes it feasible to find more complicated shapes of clusters rather than with the classical k-means algorithm [10]. Support vector machines are examples from supervised learning that make use of the same concept.
For instance, k-means finds only linear separable clusters. Kernel k-means can cluster non-linear separable groups. Figure 4 visualizes this.

Kernel functions are explained in more mathematical detail in Section 3.5.

3.2.6 k-Medoids

The technique of k-medoids requires to find cluster centers that are actual observations [21]. It allows to use any distance measure, like Jaccard or Gower distance, not only the squared Euclidean distance [48]. The centroid plays the role of the object with the smallest dissimilarity d to the other objects in the cluster. Because k-medoids requires that centroids are represented by data points, it is a computationally more costly problem to solve than k-means. Algorithm 2 is from [21], page 516, and shows the detailed steps of k-medoids.

Algorithm 2: k-medoids Clustering

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_j^* = \underset{i:C(i)=j}{\arg\min} \sum_{C(i')=j} d(x_i, x_{i'})$$

   Then $m_j = x_{i_j^*}$, $j = 1, 2, \ldots, k$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $m_1, \ldots, m_k$, minimize the total error by assigning each observation to the closest cluster center:

$$C(i) = \underset{1 \leq j \leq k}{\arg\min}\, d(x_i, m_j)$$

3. Iterate steps 1 and 2 until the assignments do not change.

3.3 Gaussian Mixture Model

It is quite common in machine learning to assume a probabilistic model that involves the normal distribution, also called Gaussian distribution. Examples are linear and

quadratic discriminant analysis [21], Gaussian naive Bayes [37], probabilistic PCA [5], Gaussian Processes [43], and the Kalman filter [29]. A multivariate Gaussian distribution, named in honor of Carl Friedrich Gauss, has according to [5] the form:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}, \tag{6}$$

meaning that the data x is parametrized by μ and Σ. $\mu \in \mathbb{R}^d$ is the mean vector of the distribution. $\Sigma \in \mathbb{R}^{d\times d}$ a symmetric positive semi-definite covariance matrix, that is, $\Sigma = \Sigma^T$ and all eigenvalues λ_j of Σ are real-valued and non-negative, $\lambda_d \geq \ldots \geq \lambda_1 \geq 0$. $|\Sigma|$ represents the determinant of the covariance matrix. It is noteworthy that to identify a normal distribution, it suffices to know the mean vector and the covariance matrix. This makes it particularly comfortable to work with the Gaussian distribution.

An important motivation behind the wide usage of the normal distribution in statistics and machine learning is the central limit theorem (CLT) [24]. It can be shown that the sum of independent and identically distributed (i.i.d.) random variables follows approximately a normal distribution, regardless of the distribution of the random variables itself. First work of the CLT goes back to Jacob Bernoulli, Abraham de Moivre, and Pierre-Simon Laplace. There are also generalizations of the CLT, for example to dependent data. Other nice properties of the normal distribution are that the conditional distributions are also normally distributed, and it has maximum entropy among all probability distributions having finite mean and variance [5].

In the context of clustering, the normal distribution can be used as the distributional foundation of a Gaussian mixture model (GMM), also called mixture of Gaussians [37]. A GMM is basically a superposition of k normal distributions [5]. The probabilistic formulation has the form:

$$P(x|\pi,\mu,\Sigma) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x|\mu_j,\Sigma_j). \tag{7}$$

π_j are called mixing coefficients or mixture components and need to be probabilities, that is, $\pi_j \geq 0$ and $\sum_{j=1}^{k}\pi_j = 1$. μ_j and Σ_j are the mean vector and the covariance matrix of the j'th mixture component, respectively. Like all clustering methods

analyzed in this thesis, a GMM makes certain assumptions about the analyzed data, as already outlined in the context of the k-means algorithm. The assumption of normally distributed clusters can be appropriate in some cases and less appropriate on others. This circumstance is comparable to the supervised learning setting using linear or quadratic discriminant analysis, or a Gaussian naive Bayes classifier, where the model assumptions are either valid or violated.
Due to the underlying Gaussian distribution, the measured distance between the data points and the cluster centers is based on the squared Mahalanobis distance [51]:

$$D_M^2(x|\mu, \Sigma) = (x - \mu)^T \Sigma^{-1} (x - \mu). \tag{8}$$

For Σ being the identity matrix I, the Mahalanobis distance becomes the Euclidean distance [5].
From a statistical learning point of view, a GMM is a generative model [39]. This means that the GMM models the joint probability distribution $P(X, Y)$ for two random variables X and Y, and infers from this the posterior distribution of Y given X, $P(Y|X)$. For a GMM, X is the design matrix and Y are the latent cluster labels that are estimated. The way to infer the posterior is by using Bayes rule, which says that $P(Y|X) = P(X|Y)P(Y)/P(X) = P(X, Y)/P(X)$. Bayes rule is motivated by the definition of conditional probability for two random variables X and Y: $P(X|Y) = P(X, Y)/P(Y)$.

3.3.1 Expectation-Maximization Algorithm

To estimate the parameters of a GMM, the method of maximum likelihood is most appropriate [37]. The concept of a likelihood functions is explained later in this section. We often work with the log-likelihood function, as this makes computations easier than working with the likelihood function but gives the same solution. Estimating the model parameters is not straightforward, as we have unlabeled data. Because the labels are not observed, we refer to the parameters as hidden or latent. A powerful and common method to infer the parameters of this probabilistic model is the expectation-maximization (EM) algorithm. The EM algorithm is an iterative method that maximizes the likelihood and makes sure

that the covariance matrices of each Gaussian distribution are positive definite, meaning all their eigenvalues are larger than zero. This important property is not guaranteed by gradient-based optimizers.
Another commonly used statistical model, where the EM algorithm finds usage in learning the parameters, is the hidden Markov model (HMM) [5].
The EM algorithm finds only a local optimum of the likelihood function, but this is the case for all optimization techniques applied in practice [37]. Alternatives to the EM algorithm are Bayesian methods. An overview of such approaches using the Gibbs sampler to solve this problem and other sampling-based methods based on Markov chain Monte Carlo (MCMC) can be found in [5].
In the following, we describe the EM algorithm based on the explanations in [45]. It reminds roughly on the k-means algorithm from the previous section. At the beginning of this method, the model parameters are arbitrarily or heuristically initialized. Then, the EM algorithm consists of two steps that are performed iteratively:

1. Expectation (E-step): Compute the posterior distribution $P(C = i|x_j)$, that is, the probability of the i'th mixture component given the j'th observation. This is done by using Bayes rule: $P(C = i|x_j) \propto P(x_j|C = i)\pi_i$, where $P(x_j|C = i)$ is $\mathcal{N}(x_j|\mu_i, \Sigma_i)$.

2. Maximization (M-step): Compute the mean, covariance matrix, and mixture coefficient of the i'th component based on the result of the E-step.

   - $\mu_i = \sum_j p_{ij} x_j / n_i$
   - $\Sigma_i = \sum_j p_{ij}(x_j - \mu_i)(x_j - \mu_i)^T / n_i$
   - $\pi_i = n_i / n$

   This means the likelihood of the data given the parameters is maximized.

It can be shown that the EM algorithm increases the likelihood function of the data at every iteration. The EM algorithm will converge to a local optimum or in rare cases to a saddle point. By running the algorithm several times using different initialization parameters, we can make sure to overcome suboptimal solutions, in the same way as for k-means.

3.3.2 Similarities Between k-Means and GMM

By comparing the EM algorithm for estimating the parameters of a GMM and the k-means algorithm, it becomes clear that these two methods have certain resemblances. For example, both techniques are based on an iterative procedure and both require to define the number of clusters k [5].
A big difference is the probabilistic model assumption for a GMM. Every cluster is modelled as a normal distribution with a mean vector μ_i and a covariance matrix Σ_i. This is not the case for k-means, where the only learned parameters are the mean vectors as the cluster centroids. k-means can be seen as a deterministic special case of a GMM by modelling every covariance matrix as an isotropic matrix $\Sigma_i = \sigma^2 I$ and considering $\pi_i = 1/k$ fixed [37]. Hence, a GMM can find the same clusters as k-means but is also able to learn more complicated cluster shapes.
It is noteworthy to mention that an often-applied strategy to initialize the parameters of a GMM is to first use k-means [5]. This is the default method in the Gaussian-Mixture function in Python's scikit-learn library.

3.3.3 Computational Aspects and Overfitting

The higher flexibility in modelling clusters by a GMM using arbitrary covariance matrices compared to k-means results in two possible problems. On the one hand, much higher computational costs arise for high-dimensional problems, as a $d \times d$ covariance matrix Σ contains $O(d^2)$ parameters [5]. On the other hand, overfitting can easily happen: Using more clusters and allowing for arbitrary covariance matrices can lead to a very flexible model, that does not allow for generalizations to unseen data based on the trained GMM.
Another common difficulty is that a learned cluster of a GMM contains only one observation. This collapsing of a single data point to a Gaussian component is called singularity problem. One possibility to prevent overfitting is to use a probability distribution for the mixing coefficients, as it is the case for Bayesian models [37].
Another method is to make constraints for the covariance matrix Σ. One approach is to restrict the covariance matrix to be isotropic, that means it has the form $\Sigma = \sigma^2 I$, where $\sigma^2 > 0$ is a from the data estimated scalar and I is the $d \times d$ identity matrix. Another possibility is to learn the same covariance matrix for all

clusters, that is, $\Sigma_i = \Sigma \; \forall i$. As already stated, by comparing to supervised learning algorithms, similar simplification techniques find application in the context of the Gaussian naive Bayes classifier and linear discriminant analysis.

3.3.4 Bayesian Information Criterion

An advantage of using a probabilistic clustering method like a GMM is the possibility to evaluate the likelihood function of the model based on information theoretical concepts [37]. One such technique is the Bayesian information criterion (BIC). The BIC can be motivated, as the name suggests, by a Bayesian point of view, where probabilities are treated in a subjective way [5]. The BIC is defined as:

$$BIC = p\log(n) - 2\log(\ell(\hat{\theta})), \tag{9}$$

where p are the number of parameters in the model, ℓ is the likelihood function, and $\hat{\theta}$ is the maximum likelihood estimator (MLE) of θ. The concept of a likelihood function goes back to the British statistician Ronald A. Fisher [22]. A likelihood function $L : \Theta \to \mathbb{R}$ has the form:

$$L(\theta) = \prod_{i=1}^{n} f(x_i|\theta), \;\; \theta \in \Theta, \tag{10}$$

that means it is a function of n observations $x_1, \ldots, x_n$ given the parameters θ out of the parameter space Θ. It is common to take the natural logarithm of the likelihood function, denoted as log-likelihood, to make computations easier, so that the likelihood function results in a sum rather than a product. As we deal with normal distributions in the context of GMMs, the univariate likelihood function has the form:

$$L(\theta) = L(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right), \tag{11}$$

where $\theta = (\mu, \sigma^2)$ represents the mean and the covariance, see also Equation (6). One method to estimate these parameters is to use the above-mentioned technique of maximum likelihood (ML) estimation. The ML estimator has some favorable statistical properties like asymptotically unbiasedness and consistency. The ML

estimator $\hat{\theta}$ is given by the expression:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta) = \arg\max_{\theta \in \Theta} \log L(\theta). \tag{12}$$

In case of a simple form of the likelihood function, like for a normal distribution, an exponential distribution, or a Poisson distribution, the MLE is computed by taking derivative of the log-likelihood with respect to the parameters, setting this expression to zero, and solve it:

$$\frac{\partial \log L(\theta)}{\partial \theta} \stackrel{!}{=} 0. \tag{13}$$

For the likelihood function in Equation (11), that is based on the normal distribution, the MLE for μ is derived, starting from the log-likelihood, as $\frac{\partial}{\partial \mu} \sum_{i=1}^{n} -\frac{(x_i-\mu)^2}{2\sigma^2} + \text{const} \Leftrightarrow \sum_{i=1}^{n} \frac{x_i-\mu}{\sigma^2} = 0 \Leftrightarrow \sum_{i=1}^{n} x_i = n\mu \Longrightarrow \mu_{MLE} = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}$. For the MLE of σ^2 we get $\sigma^2_{MLE} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$. Thus, we obtain the estimation of the parameters based on the given data. Likelihood functions can have local maxima, what has to be considered for a GMM. As explained above, the EM algorithm maximizes the likelihood function but is only able to find a local optimum.

The BIC is a quantity where the model with the smallest value is favored. In the literature, it is also known as the Schwartz criterion [5].

3.3.5 Akaike Information Criterion

An alternative approach to the BIC is the Akaike information criterion (AIC), that can be motivated from a frequentist view [22]. It is similarly defined as the BIC in Equation (9) as it also penalizes the log-likelihood function. It is given by:

$$AIC = 2p - 2\log(\ell(\hat{\theta})). \tag{14}$$

We see that for AIC, there is the term $2p$, where for the BIC, $p\log(n)$ is used instead. The AIC has the goal to utilize the in-sample prediction loss to approximate the out-sample prediction loss [11].

3.3.6 Bayesian Gaussian Mixture Model

As already mentioned, it is possible to extend a GMM and treat it in a Bayesian way. A common approach to model the mixing coefficient π_i is to use a Dirichlet distribution or a Dirichlet process [37]. The resulting method is then called Dirichlet process mixture model. The reason to make use of this prior distribution is that the Dirichlet distribution is a conjugate prior to the categorical distribution. This distribution results from having k clusters. Hence, the posterior distribution $P(C = i|x_j)$ belongs to the same distributional family as the prior distribution π_i, what usually leads to elegant mathematical properties and makes the computation more efficient. For Bayesian methods, an extension to maximum likelihood estimation is used, called maximum a-posteriori (MAP) estimation. Commonly preferred methods for MAP estimation are variational inference, which has similarities with the EM algorithm, and Markov chain Monte Carlo (MCMC) algorithms.

3.4 DBSCAN

DBSCAN stands for density-based spatial clustering of applications with noise [14]. It is a non-parametric clustering method, meaning that no distributional assumptions are made. Clusters are defined as high-density regions that are separated by low-density regions. This is a fundamentally different approach compared to k-means or a GMM. The clusters found by DBSCAN will have similar densities. This makes it possible to detect arbitrary shaped clusters, what is a big advantage in contrast to methods like k-means, which only find convex-shaped clusters. Different than k-means or a GMM, DBSCAN does not require to define the number of clusters by the user, and running it multiple times is not necessary, as local optima do not appear.

DBSCAN works on three types of points: Core points, reachable points, and noise points. Core points build up the clusters. For these points, there exists at least a certain number of points in its neighborhood, called minPts, based on a distance parameter ε. A reachable point is not a core point but is in a reachable distance ε from a core point. Points that are not reachable from any other point are noise points, also called outliers.

As mentioned above, DBSCAN requires two hyperparameters to be defined: The minimum number of neighborhood points minPts and the distance parameter ε. These hyperparameters are then used to find an optimal clustering. Because they determine the result, that is, how many clusters the algorithm finds, it is important to find optimal values for minPts and ε. However, figuring out optimal hyperparameters is not a trivial task. Often, different parameter values are used, and the results are qualitatively evaluated. This shows that, although it can be avoided to determine a priori the number of clusters k, we still have to define the hyperparameters that influence the clustering outcome.

A big challenge is to apply DBSCAN in high-dimensional spaces, because feature spaces get sparse if the number of dimensions becomes large. This makes it hard to distinguish between high-density regions and low-density regions [26].

3.5 Mean Shift

The clustering method mean shift is based on the statistical concept of kernel density estimation [51]. This is a non-parametric technique to estimate the density function $f = F'$ of some given data $x_1, \ldots, x_n \sim F$, where F is a distribution function. The data $x_1, \ldots, x_n$ is convolved with a kernel function K having bandwidth h. The goal is to make as few assumptions about f as possible [56]. There are two parameters for kernel density estimation: A kernel function K has to be chosen and the bandwidth h has to be estimated based on the data. Finding the optimal bandwidth h is considered as the most important part of density estimation. In general, a kernel function K is a probability density function with the following properties:

$$K(x) \geq 0, \int_{-\infty}^{\infty} K(x)dx = 1, \; K(x) = K(-x). \tag{15}$$

This ensures that K is non-negative, integrates to one, and is symmetric. The kernel K is usually a smooth function, like the Gaussian kernel [5], which has the form:

$$K(x) = \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi h^2}} \exp\left(-\frac{\|x - x_i\|^2}{2h^2}\right), \tag{16}$$

where $\|\cdot\|$ denotes the Euclidean norm such that $\|x\|^2 = \langle x, x\rangle = \sum_j x_j^2$. For the Gaussian kernel, h represents the standard deviation. Other examples of kernel functions are the Epanechnikov kernel $K(x) = \frac{3}{4}(1 - x^2)I(x)$ or the boxcar kernel $K(x) = \frac{1}{2}I(x)$, where $I(x) = 1$ if $|x| \leq 1$ and 0 otherwise [56]. These two kernels have, in contrast to the Gaussian kernel, finite support. Further, the Epanechnikov kernel is optimal with respect to the mean-squared error. It can be shown that the kernel density estimation of f converges in probability to the true density f as $n \to \infty$, independent of the choice of the kernel K. See for instance Chapter 6 of [56] for a proof.

The resulting d-dimensional density estimate has the general form:

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{17}$$

The bandwidth h determines how many clusters the algorithm finds, because for smaller h, the estimated density function becomes more wiggly [7]. Every local maximum can be considered as a possible cluster centroid and, thus, more clusters will be found. For a large enough bandwidth h, only one cluster will be found. We see that h controls the bias-variance tradeoff. There are several ways how to estimate the optimal value for h. One common approach is to use cross-validation [51].

Once the density function f is estimated, the goal is to find the stationary points of it. These are located where the gradient of f is zero: $\nabla f(x) = 0$. The mean shift procedure works by updating the estimate for the mode y_k at iteration k [51]:

$$y_{k+1} = y_k + m(y_k) = \frac{\sum_i x_i G(y_k - x_i)}{\sum_i G(y_k - x_i)}, \tag{18}$$

with

$$G(y_k - x_i) = -\frac{\partial}{\partial x} K\left(\frac{\|y_k - x_i\|^2}{h^2}\right). \tag{19}$$

This updating scheme converges to a local maximum of f, given a monotonically increasing kernel.
The previous analysis shows that mean shift does not require to define the number of clusters by the user but figures it out by itself based on the estimated density function f. The set of all points around a cluster centroid that converge to the same optimum is called basin of attraction. These points belong to the same cluster group.
In contrast to clustering methods like k-means, mean shift does not make any assumptions about the shape of the clusters. It does not need multiple restarts, as the algorithm will always find the same global solution. Thus, mean shift is a deterministic clustering algorithm, like DBSCAN.
Mean shift is a frequently used technique in computer vision for image segmentation and face tracking [51].

3.6 Hierarchical Clustering

One of the simplest and best-known clustering algorithms is hierarchical clustering [49]. It is based on heuristics and does not optimize an objective function [37]. This makes it hard to evaluate the quality of the clustering results. Hierarchical clustering is a method that does not produce flat clusters, as k-means or GMMs do, but rather creates nested clusters. It uses for the computation an $n \times n$ dissimilarity matrix, where n is the number of observations. For every data point x_i, the distance to all other data points is computed. Usually, the Euclidean distance is the preferred metric for this. Using a dissimilarity matrix is a different approach compared to clustering algorithms that utilize the design matrix X instead. We can apply hierarchical clustering without having access to the input matrix, but only knowing the dissimilarities.
One approach of hierarchical clustering, called agglomerative or bottom-up clustering, starts by considering every data point as a cluster, the trivial clustering [49]. Then, repeatedly, the clusters that are closest to each other are merged until one global cluster is defined. The questions are how to define closeness of data points and when to stop this merging process so that an optimal number of clusters results.
Another approach, called divisive or top-down clustering, starts at a single cluster

for all data points and then recursively splits the clusters. Computing the optimal split is hard to achieve for this type of technique. Therefore, divisive clustering is less often used in practice than agglomerative clustering [37].
The result of hierarchical clustering can be represented as a dendrogram [21], which is a tree that visualizes the splitting of the observations. This graphical representation of the clustering solution is independent of the number of dimensions and makes hierarchical clustering a popular method.
Hierarchical clustering has high computational costs of $O(n^3)$, but using certain data structures or different kinds of dissimilarity measures can reduce the calculations [37]. If the number of observations n is large, it is common to first run k-means and then apply hierarchical clustering. This is another method to speed up the process.

3.7 Spectral Clustering

An example of a graph-based clustering technique is spectral clustering [55]. It is a successful method with many applications. An explanation for this is that spectral clustering does not make strong assumptions about the shape of the data groups. Additionally, it is a mathematically rigorous method. Another appealing property of spectral clustering is that there are no local optima to worry about, in contrast to k-means or a GMM [38]. Because of this, we do not have to run the algorithm several times to get a good solution. Still, spectral clustering has some drawbacks, which lie in the parametrization of the algorithm, described below.
The name spectral refers to spectrum, the set of the eigenvalues of the modified adjacency matrix of the underlying graph that is used. It is based on concepts of spectral graph theory, a field at the intersection of graph theory and linear algebra. Spectral clustering uses a similarity measure computed from the data points $x_1, \ldots, x_n$, which is symmetric and non-negative. Using this data, a graph $G = (V, E)$ is constructed, consisting of a set of vertices, also called nodes, V and undirected edges E. V represents the n data points and E the connections between them. Clearly, the number of nodes in G is n, that is $|V| = n$. To the edges of G, weights are assigned based on the computed similarities. There are several possibilities to create such a graph G, for example by constructing a fully

connected graph, an ε-neighborhood graph, or a k-nearest neighbors graph. Based on this graph, a diagonal matrix $D \in \mathbb{R}^{n \times n}$, called degree matrix, containing the degree of each node of G and an adjacency matrix $W \in \mathbb{R}^{n \times n}$ are created. These matrices are then used to compute the so-called unnormalized graph Laplacian matrix L:

$$L = D - W. \tag{20}$$

The $n \times n$ graph Laplacian matrix L has several important mathematical properties [55]: It is symmetric, that is, $L = L^T$. This can be easily proved: As D is a diagonal matrix, hence, it is symmetric, and the adjacency matrix W of an undirected graph is always symmetric, $D - W$ is also symmetric. Further, it can be shown that $f^T L f = \frac{1}{2} \sum_{i,j} w_{ij}(f_i - f_j)^2 \geq 0 \; \forall f \in \mathbb{R}^n$. Therefore, L is a positive semi-definite matrix. From these two results, it follows that all eigenvalues of L are real-valued and non-negative, $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. It can also be proved that the smallest eigenvalue λ_1 of L is zero. Thus, the eigenvector f corresponding to this eigenvalue is given by the constant one vector, that is, a vector only containing ones. Because of this property, the eigenvector corresponding to the smallest eigenvalue is ignored in the context of spectral clustering. Two other common approaches in spectral clustering are to construct either the symmetric normalized graph Laplacian matrix $L_{sym} = I - D^{-1/2} W D^{-1/2}$ or the random walk normalized graph Laplacian matrix $L_{rw} = I - D^{-1} W$ [37]. Such extensions have the goal to create more balanced clusters [47]. Additional properties of the graph Laplacian matrix L are that it is singular, and all its off-diagonal elements are smaller or equal to zero.

The next step deals with the spectrum of L. The k eigenvectors corresponding to the smallest k eigenvalues of L, that build an orthogonal basis in $\mathbb{R}^{n \times k}$, are used [21]. This means we solve the eigenvalue problem $Lf = \lambda f$. The user defines k, that corresponds to the number of desired clusters. Due to the symmetry of the graph Laplacian matrix L, the eigenvectors and eigenvalues can be computed using for instance Lanczos algorithm, an efficient iterative numerical method for Hermitian matrices with good convergence properties [2]. These k eigenvectors are then used in the clustering step, where usually a common clustering method like the k-means algorithm is applied.

The big challenge in the practical usage of spectral clustering is to define the optimal hyperparameters. One such hyperparameter is the method to construct the graph, and, consequently, also the ε-distance or the number of neighbors, respectively. If a kernel function is used as a similarity measure, the kernel must be determined as well, for example a radial basis function, polynomial, cosine, or sigmoid kernel. Further, defining the numbers of clusters k is also a demanding task.
Spectral clustering is related to kernel principal component analysis (kernel PCA) [21], where a kernel function is utilized as a similarity measure to perform non-linear PCA in a reproducing kernel Hilbert space (RKHS). The same feature mapping concepts as for kernel k-means are used. In practical applications, spectral clustering produces better results than kernel PCA [37].

3.8 Affinity Propagation

Affinity propagation is a recently developed iterative clustering technique [16]. As other algorithms like mean shift or DBSCAN, it does not require to define the number of clusters k a priori. Moreover, affinity propagation does not make any probabilistic assumptions about the data. Every single data point has the potential to become an exemplar. An exemplar will at convergence of the algorithm be interpreted as a cluster center that is representative for similar data points. Therefore, affinity propagation has some similarity with k-medoids, where cluster centers are existing data points too.

3.8.1 Underlying Principle

Like hierarchical clustering, affinity propagation works on an $n \times n$ similarity matrix, where n is the number of observations [37]. Usually, the similarity between two points is measured using the negative Euclidean distance [16]. Concretely, for two points x and x', the distance is computed as $-\|x - x'\|^2$. Affinity propagation has two main parameters: Damping and preference. These two parameters are crucial to find a decent number of clusters. Using a suboptimal value for damping can lead to non-convergence of the algorithm. In such a situation, the method does not work as it should and will find only one cluster. A too large value for the preference

parameter results in finding too many clusters. The algorithm works by passing messages between the data points. There are two kinds of message passing, one is called responsibility, the other one availability. Both message passing methods build its own $n \times n$ matrix.
A drawback of the algorithm is that convergence is not guaranteed due to numerical oscillation [37]. Further, it has a high computational complexity of $O(n^2 \log(t))$, where t is the number of iterations until the algorithm converges. These high costs make affinity propagation not optimal for large datasets.

3.8.2 Algorithm

Algorithm 3 from [12], page 2, shows the steps of affinity propagation.

Algorithm 3: Affinity Propagation

1: **Initialization**
2: $r(i,k) = 0$, $a(k,i) = 0 \ \forall i,k$
3: **Responsibility updates**
4: $r(i,k) \leftarrow s(i,k) - \max_{j:j \neq k}(a(j,i) + s(i,j))$
5: **Availability updates**
6: $a(k,k) \leftarrow \sum_{j:j \neq k} \max\{0, r(j,k)\}$
7: $a(k,i) \leftarrow \min\left(0, r(k,k) + \sum_{j:j \notin \{k,i\}} \max\{0, r(j,k)\}\right)$
8: **Making assignments**
9: $c_i^* \leftarrow \arg\max_k \ r(i,k) + a(k,i)$

Cluster Validation Methods

In this section, we evaluate several common validation techniques for cluster analysis. In machine learning this is usually done to decide how good the resulting solution is. In unsupervised learning, however, there is no straightforward evaluation measure for this task [21]. All presented techniques are heuristics and are different to evaluation methods used in supervised learning.
One reason to validate the resulting clusters is that most clustering methods will find groups in an arbitrary dataset, independent if any grouping structures do exist or not [52]. As an illustration, we have a dataset with 1000 uniformly distributed data points in $\mathbb{R}^2$, see Figure 5. Applying k-means clustering with $k = 3$, the algorithm finds three clusters. Of course, the result does not make sense. For uniformly distributed data, we would expect to have only one cluster or no clusters at all. A density-based clustering algorithm, as another example, would find a different solution. This simple instance can be easily analyzed and evaluated by humans. But in the case of higher-dimensional data, what applies to the 13-dimensional MFCCs, the inspection has to be automated. Other reasons why to use evaluation methods for clustering are to determine the optimal number of groups in the data as well as to validate the resulting clustering with respect to the data, with or without using ground truth labels.
The following methods are either internal or external validation techniques. Internal evaluation methods do not include the class labels, also called ground truth, for the validation of the clustering. This contrasts with external validation methods that require the true labels of the clusters to be known. Both internal and external validation methods are independent of the clustering algorithm.

4.1 Internal Evaluation

In most real-world applications, the cluster labels are not available. Thus, internal evaluation methods can be seen as a more realistic approach for cluster validation than external evaluation measures [20]. However, refraining from using the ground

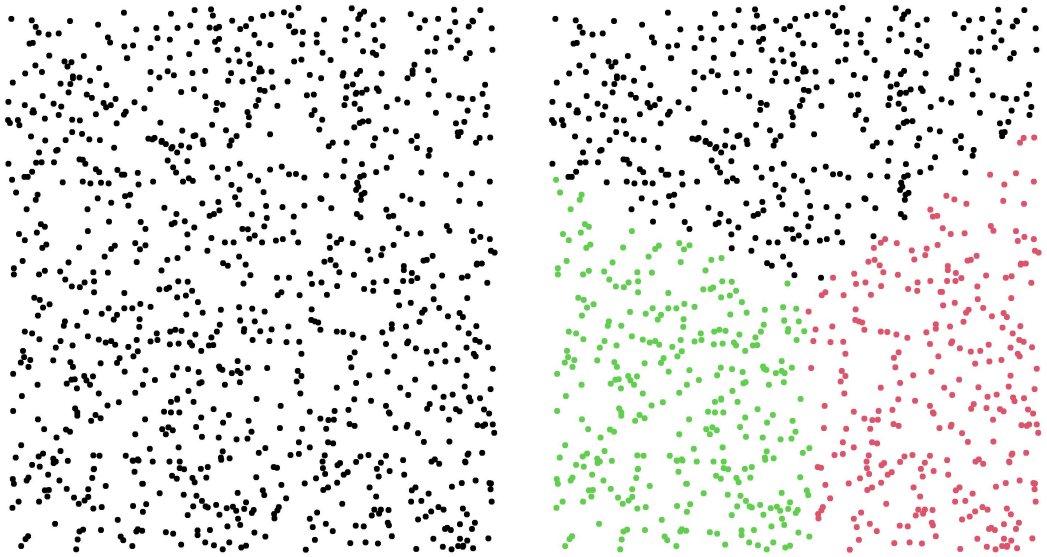

Figure 5: **Uniformly distributed data with no groups.** Left: 1000 uniformly distributed data points with no clear pattern. Right: The same data with clusters found by k-means with $k = 3$.

truth makes the task of cluster evaluation even harder, as important information is missing. We review four common internal validation techniques.

4.1.1 Davies-Bouldin Index

The Davies-Bouldin (DB) index is a widely used internal cluster evaluation method invented in 1979 [8]. Like for cluster evaluation methods in general, the user tries out different numbers of clusters, for instance $i = 1, 2, \ldots, 10$. Assuming a dataset is partitioned by an arbitrary clustering algorithm into k clusters, for example by using k-means or a GMM. The DB index measures the cluster cohesion of the points x in cluster C_i to its centroid a_i as well as the cluster separation between two centroids a_i and a_j for $i \neq j$ [1]. It is defined according to [19] as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} R_i, \tag{21}$$

with

$$R_i = \max_{i=1,\ldots,k, i \neq j} \left(\frac{S_i + S_j}{d_{ij}} \right), \tag{22}$$

$$S_i = \frac{1}{k} \sum_{x \in C_i} \|x - a_i\| . \tag{23}$$

S_i measures the cluster cohesion of cluster C_i, where a_i is its centroid. Further,

$$d_{ij} = \|a_{hi} - a_{hj}\| = \sqrt{\sum_{h=1}^{N} (a_{hi} - a_{hj})^2}. \tag{24}$$

d_{ij} is the distance between cluster C_i and C_j, that can be seen as a dissimilarity measure. a_{hi} is the h'th element of the centroid of cluster C_i.
The inventors of the DB index claim that the similarity function R_{ij} is defined to satisfy the following conditions [8]:

1. $R_{ij} \geq 0$
2. $R_{ij} = R_{ji}$
3. $R_{ij} = 0 \Leftrightarrow S_i = S_j = 0$
4. If $S_j = S_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$
5. If $d_{ij} = d_{ik}$ and $S_j > S_k$ then $R_{ij} > R_{ik}$

Therefore, R_{ij} is non-negative, symmetric, and zero if and only if all cluster cohesion measures are zero. The goal is to minimize the DB index. Thus, its minimum value indicates the optimal number of clusters.

4.1.2 Calinski-Harabasz Index

Calinski-Harabasz (CH) index, also called variance ratio criterion (VRC) [6], is another frequently used technique to find the optimal number of clusters k for a given dataset. The CH index is according to [33] mathematically defined as:

$$CH = \frac{\text{trace}(B)/(k-1)}{\text{trace}(W)/(n-k)}. \tag{25}$$

B is the so-called between cluster scatter matrix, also called between-group/cluster sum of squares (BGSS), with

$$B = \sum_{j=1}^{k} n_j (c_j - c_E)(c_j - c_E)^T. \tag{26}$$

W is the within cluster scatter matrix, also called within-group/cluster sum of squares (WGSS), with

$$W = \sum_{j=1}^{k} \sum_{x \in C_j} (x - c_j)(x - c_j)^T. \tag{27}$$

In Equation (25), trace(B) is the sum of the diagonal elements of B, that is, trace(B) $= \sum_i b_{ii}$, n is the number of data points, and k is the number of clusters. The higher the resulting index value, the better. The inventors of the CH index recommend to select the optimal number of clusters k by either choosing the maximum value resulting from Equation (25) or the value for which the CH index gives a rapid increase [6].
The CH index reminds of the analysis of variance F-statistic, where the variability between the groups and within the groups are compared [6].
Experiments show that the CH index performs well compared to other evaluation methods, and in some context, it has the ability to recover the true number of clusters [20], [33].

4.1.3 Dunn Index

The Dunn index (DI) tries to identify compact, also called dense, and well separated clusters [19]. This seems to be an intuitive and meaningful definition to evaluate clusters. The Dunn index is defined as follows:

$$DI = \min_{i=1,\ldots,k} \left(\min_{j=i+1,\ldots,k} \left(\frac{d(c_i, c_j)}{\max_{h=1,\ldots,k} \operatorname{diam}(c_h)} \right) \right). \tag{28}$$

$\operatorname{diam}(c_h)$ is the diameter of cluster c_h, that is, the maximal distance between any two points in the same cluster. The more compact a cluster is, the lower the diameter gets. It is defined as:

$$\text{diam}(c_h) = \max_{x,y \in c_h} d(x, y). \tag{29}$$

$d(c_i, c_j)$ in Equation (28) is the dissimilarity between two clusters c_i and c_j. The Euclidean or Manhattan distance are frequently used to measure the dissimilarity. The more separated two clusters are, the larger this value becomes. Hence, for the Dunn index, higher values are preferred over smaller values.

4.1.4 Silhouette Index

The silhouette s of a data point x is according to [9] defined as the following ratio:

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}, \tag{30}$$

where $a(x)$ is the average similarity of x in cluster C_i to all other points in this cluster, and $b(x)$ is the minimum of the average distance of the points to the nearest cluster of C_i.

The silhouette $s(x)$ is a measure that evaluates how appropriate the assignment of observation x is with respect to cluster C_i. It is a numerical value so that $-1 \leq s(x) \leq 1$. If the silhouette is negative, then the data point x is not optimal for the assigned cluster. Therefore, the clustering has some potential for improvement. A silhouette $s(x)$ of zero means that the data point x is a neutral point that has neither positive nor negative impact on the clustering. A silhouette $s(x)$ close to one means x is optimally assigned to the cluster C_i.

When all silhouette values of a cluster are determined, we can compute the silhouette index. It is defined as the mean of all silhouettes of a cluster C_i:

$$\frac{1}{n_{C_i}} \sum_{x \in C_i} s(x), \tag{31}$$

where n_{C_i} is the number of data points in cluster C_i. Then, we can use the silhouette index to evaluate every cluster in the dataset. Like the Dunn index, it prefers compact and well separated clusters. Constructing the silhouette index is computationally intensive and requires the squared number of operations depending on the total number of observations n of the dataset.

4.2 External Evaluation

In contrast to internal validation methods, external validation methods use the true class labels [20]. This allows us to evaluate the clustering outcome, using additional information compared to internal validation methods and, therefore, deduce potentially different conclusions. According to [15], using external evaluation methods is the optimal way to evaluate clustering results. They argue that internal evaluation methods, such as the techniques explained in the previous section, do not fairly evaluate the results of a clustering. However, in practical clustering applications, the ground truth class labels are often missing. This is usually the reason why unsupervised methods are applied. For the vowel speech data analyzed in this thesis, we know the true labels. Thus, it is possible to make use of external validation methods.

In supervised learning the algorithms are provided with the class labels, whereas evaluating the clustering results using the true labels is not as straightforward. Evaluating the produced clustering outcome is in general a hard task, even if the ground truth is provided [37]. This is the case because clustering methods do not output concrete class labels. In the context of our Standard German vowel corpus, there are eight classes in total, /i y e ϕ ε a o u/, so that every vowel represents one class. Clustering algorithms, as unsupervised learners in general, do not make use of these classes and will only return values like group 1, group 2, group 3, and so on. It is not possible to automatically map these groups to the true classes. Hence, we can not determine, as an example, if cluster 1 represents vowel a or vowel u. The following external evaluation measures take this into account. Still, we can define the quantities true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), but in a different way than in supervised learning. Computing TP, TN, FP, and FN is done by considering the pairwise correct clustering outcome. This leads to a much bigger number of comparisons than n, the number of data points. In total, we have to examine $\binom{n}{2}$ combinations. As an example, given $n = 100$ observations, $\binom{100}{2} = 4950$ comparisons are needed.

The following contingency table is inspired by [50] and defines the four possible categories that we use for the Rand index and Fowlkes-Mallows index:

	Pair in same group	Pair in different groups
Pair in same group	True positives (TP)	False positives (FP)
Pair in different groups	False negatives (FN)	True negatives (TN)

4.2.1 Rand Index

The Rand index (RI) is defined according to [37] as:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}, \tag{32}$$

where $|TP + FP + FN + TN| = \binom{n}{2}$ and n represents the number of data points. The Rand index takes on values between 0 and 1. A value close to 1 represents an optimal clustering result. The number of true classes does not have to be equal to the number of clusters estimated by the clustering algorithm. The Rand index has the property of assigning the same importance to both true positives and false negatives. This is usually not desired. Further, it is often close to 1 even if the clusterings are different. There are improved versions of the Rand index to correct this, for example the Hubert and Arabie adjusted Rand index (ARI) [50]:

$$ARI = \frac{RI - \text{Expected } RI}{\max RI - \text{Expected } RI}. \tag{33}$$

In our experiments, we use the adjusted Rand index to make sure that the evaluation is as reliable as possible.

4.2.2 Fowlkes-Mallows Index

The Fowlkes-Mallows index (FMI) is related to the Rand index, but it ignores the true negatives (TN). Therefore, the Fowlkes-Mallows index gives more weight to the true positives (TP). It ranges like the Rand index from zero to one and is defined according to [50] as:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}. \tag{34}$$

4.2.3 Mutual Information

Mutual information (MI) is different to the previous validation methods, as it has its origin in information theory [37]. Let U and V be two partitions of a dataset X with $|U| = |V| = n$. U can be seen as the result of a particular clustering with R groups and V is the ground truth containing C groups. In general, R and C do not have to be equal. MI is then defined according to [54] as:

$$MI(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{n} \log \frac{n_{ij}/n}{a_i b_j / n^2}. \tag{35}$$

We can interpret MI as a measure of dependence between two probability distributions. In the case of cluster analysis, the two sets U and V are treated as realizations of these distributions. MI can also be motivated such that it measures how much uncertainty we reduce for one distribution by knowing the other distribution [41]. MI is non-negative and symmetric, that is, $MI(U,V) = MI(V,U) \geq 0$. It has relations to other information theoretic quantities like entropy, conditional entropy, and the Kullback-Leibler divergence [36], [54].
A drawback of this evaluation criterion is that more clusters tend to lead to higher MI. An often-used extension to correct for this is the adjusted mutual information (AMI). It is defined as:

$$AMI = \frac{MI(U,V) - E[MI(U,V)]}{\max\{H(U), H(V)\} - E[MI(U,V)]}, \tag{36}$$

where $E[MI(U,V)]$ is the expected MI and $H()$ stands for entropy. Like for the adjusted Rand index, we use the AMI in our experiments.

4.2.4 V-Measure

The V-measure is a relatively recent concept to externally evaluate clusterings based on entropy and conditional entropy [44]. It is defined as the harmonic mean of homogeneity h and completeness c:

$$V = \frac{(1+\beta) \cdot h \cdot c}{(\beta \cdot h) + c}. \tag{37}$$

Homogeneity h means that all clusters contain data points from only one class. It is computed as:

$$h = \begin{cases} 1 & \text{if } H(U,V) = 0 \\ 1 - \frac{H(V|U)}{H(V)} & \text{else.} \end{cases} \tag{38}$$

Completeness c refers to a cluster that contains all data points from the same class. It is defined as:

$$c = \begin{cases} 1 & \text{if } H(U,V) = 0 \\ 1 - \frac{H(U|V)}{H(U)} & \text{else.} \end{cases} \tag{39}$$

$H(U,V)$ is the joint entropy of U and V, $H(U|V)$ stands for the conditional entropy of U given V. The fraction $H(U|V)/H(U)$ can be seen as the conditional entropy normalized by entropy. Homogeneity and completeness are orthogonal to each other: An increase of h often leads to a reduction of c, and vice versa. Hence, the goal is usually to achieve a good balance between homogeneity and completeness, evaluated by the V-measure.

The V-measure takes on values between zero and one, where values close to zero mean a bad clustering solution, and values close to one implies an optimal clustering. It is independent of the number of observations in the dataset and the number of constructed clusters.

Implementation

In this section we treat the implementation of the clustering algorithms explained in Section 3. To implement the algorithms, we use the programming language Python and apply it to the generated MFCCs of the vowel speech corpus. The evaluation of the clustering results is done, where needed, using internal and external validation criterions, as we outline in detail in Section 4: Calinski-Harabasz, Davies-Bouldin, silhouette, and Dunn index, as well as adjusted Rand index, Fowlkes-Mallows index, adjusted mutual information, and V-measure. Because there are eight vowels and some clustering algorithms do not allow to use only one cluster, we vary the range of clusters between two and eight.
We created the MFCC data using librosa, a Python-based library for analyzing sound data. [35]. We define 10 f_0 levels: 220, 330, 440, 523, 587, 659, 698, 784, 880, and 988 Hz. The number of MFCCs we favor is 13, a commonly used number in speech recognition. However, this number is not uniformly handled in the literature. Based on the findings of [30], we also try out 5 MFCCs in the experiments. Because the results do not change for most clustering methods we try out in our experiments, we often show only the results for 13 MFCCs.

5.1 k-Means

Internal Evaluation

It becomes clear from Figure 6 that the k-means clustering algorithm does not clearly identify a particular number of clusters for the vowel speech data using internal validation techniques. The result rather depends on the evaluation methods that are used. The Calinski-Harabasz index finds two as the optimal number of clusters for most f_o levels. The Davies-Bouldin index varies between two and eight clusters, with no clear pattern depending on the f_o level, that is, lower or higher f_o levels seem not to have an impact. The only exception is the proposed value for 988 Hz, which increases to eight clusters. One possible explanation is that

for 988 Hz there is fewer data available for the algorithm than for the other f_o levels. This might lead to a less stable result. We will see in the next sections a similar behavior for other clustering techniques. The silhouette index suggests the same results as the Calinski-Harabasz index. The Dunn index proposes reasonable results for the k-means algorithm, as it suggests mostly higher values of optimal number of clusters for all f_o levels than the other indices. Most of the proposed values are seven or eight clusters.

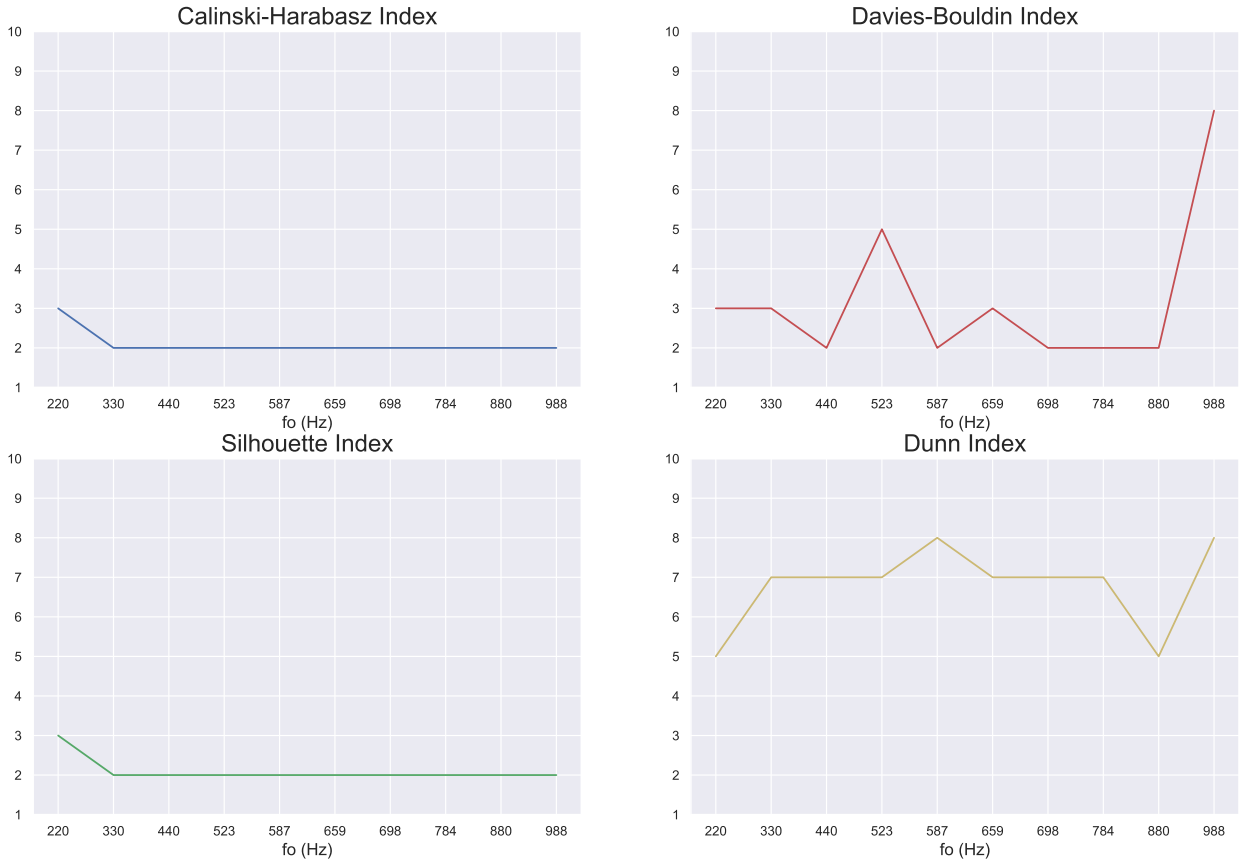

Figure 6: **k-means clustering evaluated with internal criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

External Evaluation

Figure 7: **k-means clustering evaluated with external criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

The external evaluation methods applied to k-means show a different picture than the previously described criterions. They seem to recognize more consistently that there are eight groups. This can be seen in Figure 7, showing the four external validation criterions. The adjusted Rand index varies between six and eight clusters. The V-measure suggests for nine f_o levels eight clusters. The Fowlkes-Mallows index is less consistent, it varies between two and seven optimal clusters. Adjusted

mutual information proposes mostly an optimal value between six and eight clusters, except for 880 Hz, where the number falls to three clusters. The reason for this drop is unclear to us.

5.2 Gaussian Mixture Model

Internal Evaluation

Internal validation criterions propose for the GMM a variation of results, like k-means. The Calinksi-Harabasz index proposes an optimal number of clusters between two and three. The Davies-Bouldin index and the silhouette index vary between two and eight clusters with no clear pattern regarding the f_o levels. The Dunn index proposes between four and seven clusters as optimal. The results are shown in Figure 8.

External Evaluation

As for the k-means results using external validation methods, we get more accurate suggestions for the optimal number of clusters for different f_o levels, meaning that these methods recover more reliable the true number of eight groups compared to internal criterions, see Figure 9. Except for the Fowlkes-Mallows index, all validation techniques suggest an optimal cluster number between six and eight. Like the results found for the k-means algorithm, Fowlkes-Mallows proposes for the GMM a wide range of optimal number of clusters between two and seven.

5.3 DBSCAN

Evaluating DBSCAN works different to the previous methods. As DBSCAN finds the optimal number of clusters by itself, we are not able to try out a various number of clusters. Thus, internal and external validation methods can not be used. The same is also the case for mean shift and affinity propagation. Our implementation shows that DBSCAN does not provide meaningful results, as it finds for all f_o levels only one cluster. We try out different numbers of MFCCs and different values for the two hyperparameters minPts and ε, but this does not affect the results either.

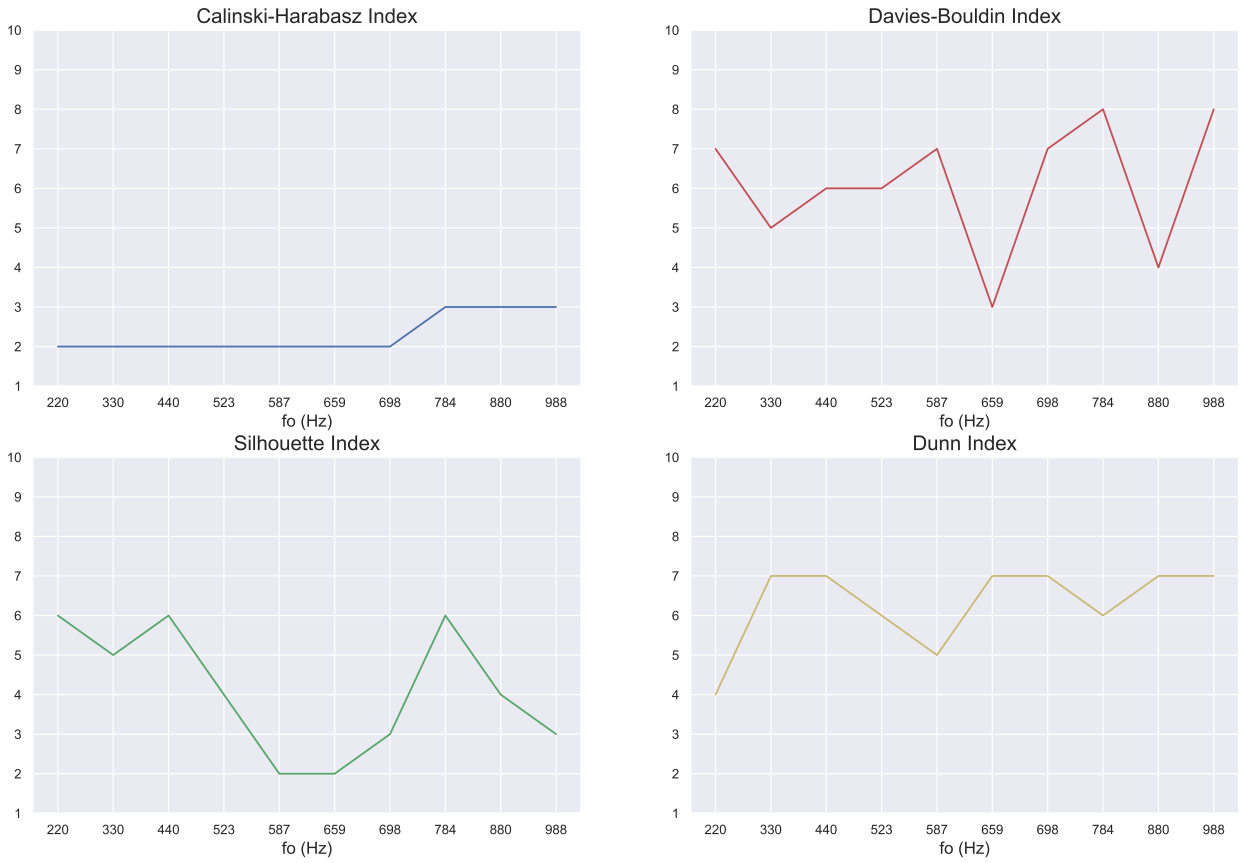

Figure 8: **GMM clustering evaluated with internal criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

This is the reason why we do not show a graphical result for DBSCAN, in contrast to all other clustering algorithms analyzed in this thesis.

A possible explanation for the poor estimations of DBSCAN is that the vowel speech data is not optimal for a density-based clustering method. This is an example, where some clustering algorithms are less appropriate for a problem, a general situation that occurs in machine learning. The other analyzed clustering algorithms in this thesis perform more appropriate for our vowel speech data.

Figure 9: **GMM clustering evaluated with external criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

5.4 Mean Shift

As explained in the previous section, we can not use internal and external evaluation methods to find the optimal number of clusters for the mean shift algorithm. Therefore, we get one single graph that shows the number of proposed clusters by mean shift, as is shown in Figure 10 and 11, where we show the 10 f_o levels for 5 and 13 MFCCs, respectively.

Mean shift finds a decent number of groups between four and eleven clusters for 5

MFCCs and two and five clusters for 13 MFCCs, respectively. Thus, the number of MFCCs has an impact on the outcome. This behavior is different to the findings for the other algorithms. We do not limit the maximal number of clusters that can be found by mean shift, as this is not possible, but we try out different values for the bandwidth to estimate the kernel density function. This hyperparameter has an impact on the cluster outcome. In the literature, there is no clear approach how to handle the bandwidth estimation in the context of clustering. We decided to use a bandwidth that leads to a reasonable outcome, that is, a cluster number neither overly low nor high.
An exception in the number of clusters that mean shift finds, occurs for the highest f_o level for both 5 and 13 MFCCs. At 988 Hz, the algorithm finds 15 and 17 clusters, respectively. We assume that this divergent behavior compared to the other f_o levels results because at 988 Hz f_o, less data is available for mean shift. Providing the algorithm with only a few observations can have an impact on the performance of the clustering process. That such a drastic change in the estimated number of clusters can be explain only due to an increase of around 100 Hz in f_o from 880 Hz to 988 Hz, remains unclear for us.

5.5 Hierarchical Clustering

Internal Evaluation

Figure 12 shows that all internal evaluation criterions applied to hierarchical clustering propose an optimal number of clusters between 2 and 8. There seems to be no clear pattern. The Calinksi-Harabasz index varies between two and eight clusters. Davies-Bouldin index starts with three clusters for 220 Hz, proposes two clusters for 330 Hz, and then increases up to 587 Hz. After this increase, it decreases once more and goes again up. The silhouette index suggests a low number at the beginning and then increases to a higher number of seven clusters, then it decreases again. The Dunn index is similar to the results of the Calinksi-Harabasz score, where the optimal number of clusters oscillates two and eight. Like for k-means and GMM, we see that the internal evaluation methods do not provide a clear result.

Figure 10: **Mean shift clustering evaluation.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

External Evaluation

The adjusted Rand index and the adjusted mutual information propose an optimal number of clusters around five to eight. Both measures start at 220 Hz with five optimal clusters and then increase to eight clusters, except for 659 Hz, where they propose seven clusters. The V-measure is able to consistently estimate the correct number of eight clusters for all f_o levels. As before, the Fowlkes-Mallows index is not able to find the true number of groups. It proposes for most f_o levels an optimal number of clusters of two, except for 784 Hz and 988 Hz. The results are shown in Figure 13.

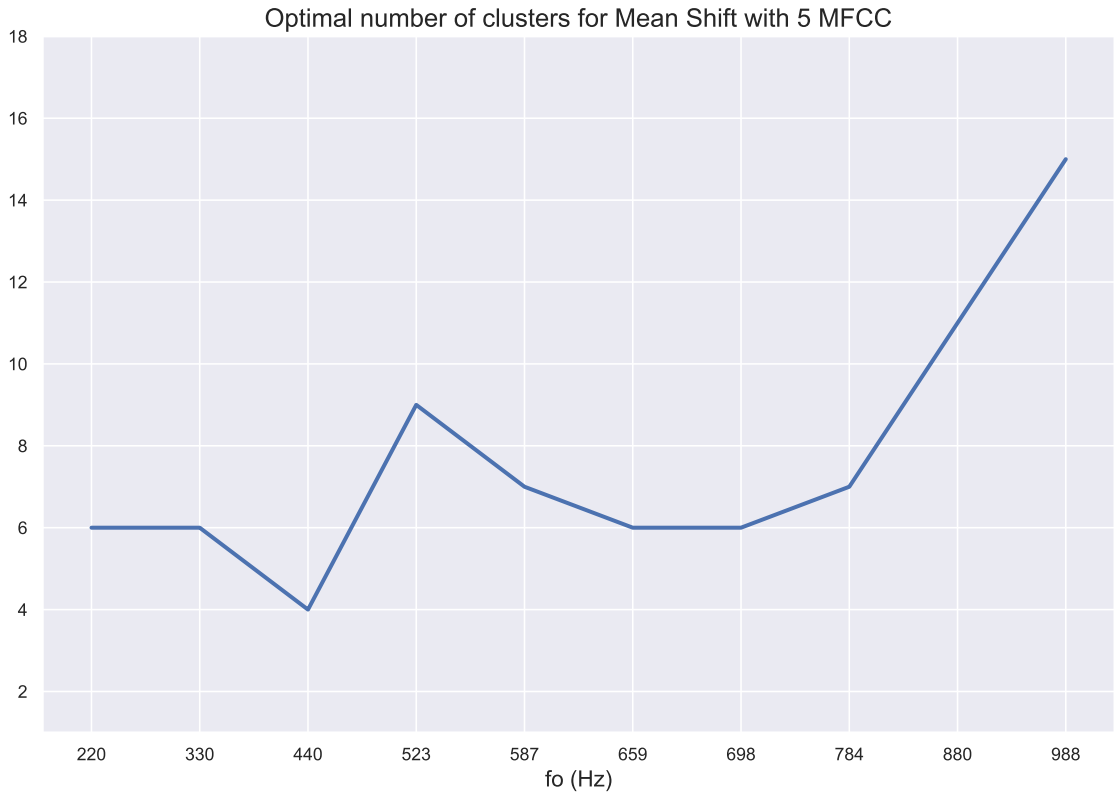

Figure 11: **Mean shift clustering evaluation.** The optimal number of clusters is evaluated for 10 f_0 based on 5 MFCCs.

5.6 Spectral Clustering

Internal Evaluation

All internal validation indices give different results for the optimal number of clusters for spectral clustering, see Figure 14. Calinski-Harabasz, Davies-Bouldin, and silhouette prefer a low number of clusters between two and four. The Dunn index switched in an almost arbitrary manner between two and eight clusters. Higher f_o levels do not affect the results. As noted for the other clustering algorithms, the internal evaluation criterions are not able to reconstruct the true number of groups, which is eight. For the vowel speech data, it seems that the

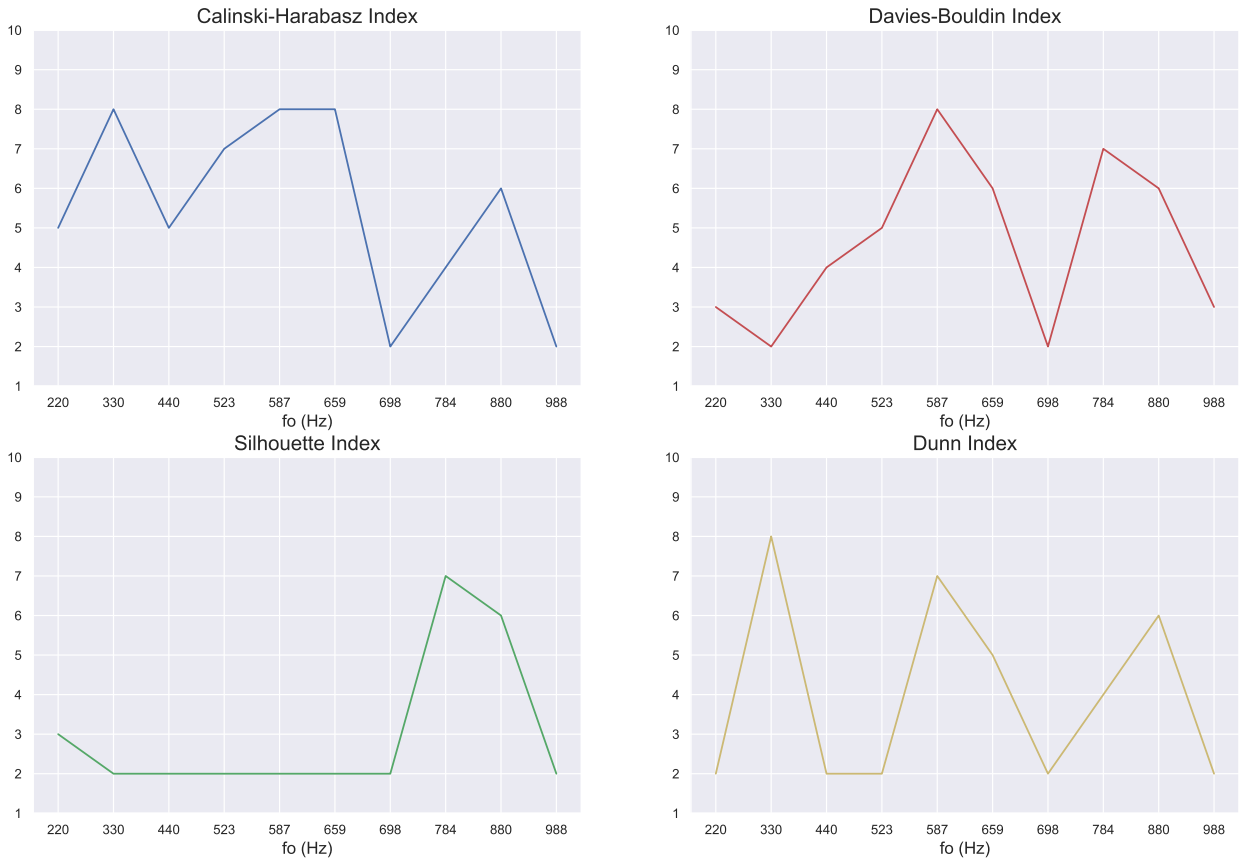

Figure 12: **Hierarchical clustering evaluated with internal criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

missing information of the ground truth class labels makes it hard for internal validation methods to estimate the correct number of clusters.

External Evaluation

The adjusted Rand index and the adjusted mutual information both vary around seven and eight clusters, except for 330 Hz and 988 Hz, see Figure 15. Like for the other clustering algorithms, Fowlkes-Mallows index suggest an optimal cluster

Figure 13: **Hierarchical clustering evaluated with external criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

number between two and eight. The V-measure is a stable metric that proposes six to eight clusters. As we describe in the other sections, this is also the case for most algorithms that we analyze in our experiments. It seems that the V-measure is an appropriate technique to find the true number of groups for the vowel speech data, independent of the clustering algorithm, as it always suggests a number of clusters close to the true number.

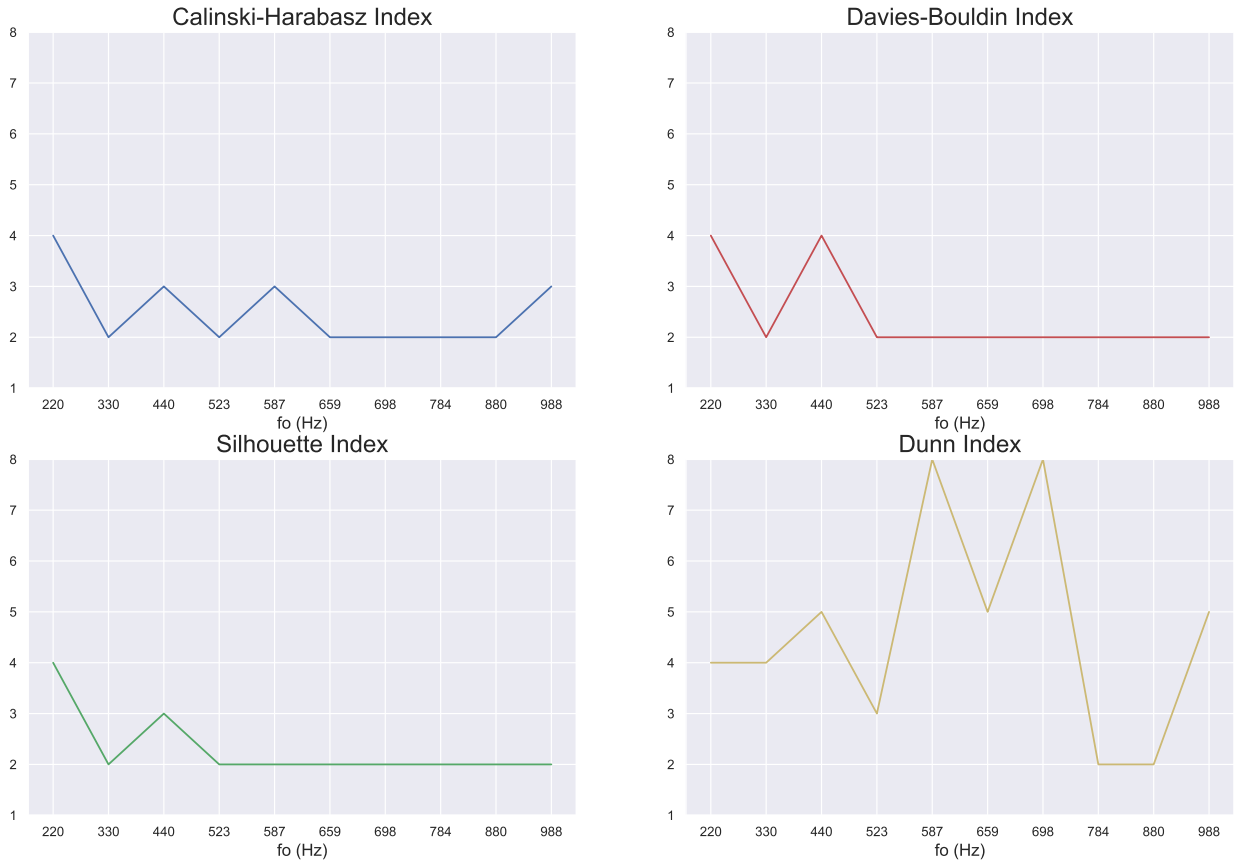

Figure 14: **Spectral clustering evaluated with internal criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

5.7 Affinity Propagation

Affinity propagation is a clustering algorithm that does not allow to define the numbers of clusters, like DBSCAN and mean shift. It finds the number of clusters on its own.

For 13 MFCCs, we see that affinity propagation finds a wide range of clusters depending on the f_o level, see Figure 16. There is a trend that for lower f_o levels, affinity propagation finds more clusters, between seven and ten clusters. For higher

Figure 15: **Spectral clustering evaluated with external criterions.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

f_o levels, less clusters are found. For 988 Hz, the estimation drops to two clusters. The same behavior appears for 5 MFCCs, as we see in Figure 17. For 880 Hz, affinity propagation estimates four clusters, for 988 Hz it estimates only two clusters.

Therefore, the number of MFCCs influences the estimated number of clusters, but both results are similar. Like for mean shift, we assume that the drop at 988 Hz f_o is because less data is available for the algorithm.

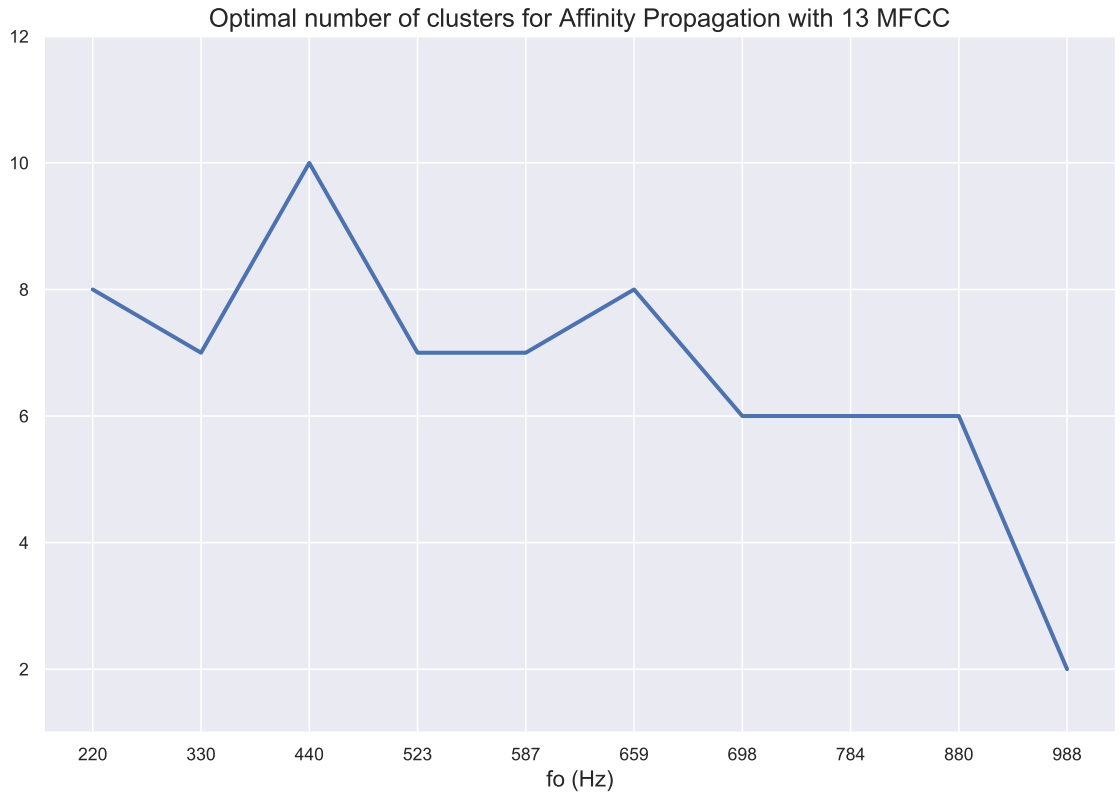

Figure 16: **Evaluation of affinity propagation.** The optimal number of clusters is evaluated for 10 f_0 based on 13 MFCCs.

5.8 Summary

In the previous sections, we evaluate several clustering algorithms for 10 f_o levels. The resulting outcomes do not depend on the clustering algorithm, but more on the validation criterions that are used. Exceptions are the clustering methods mean shift, affinity propagation, and DBSCAN, as these algorithms estimate the number of clusters on its own and do not need a separate evaluation technique.

Internal evaluation criterions frequently underestimate the true number of clusters for the vowel speech data, or they alternate between a low and a high value. The reason for this is probably the missing class labels in the evaluation process. They

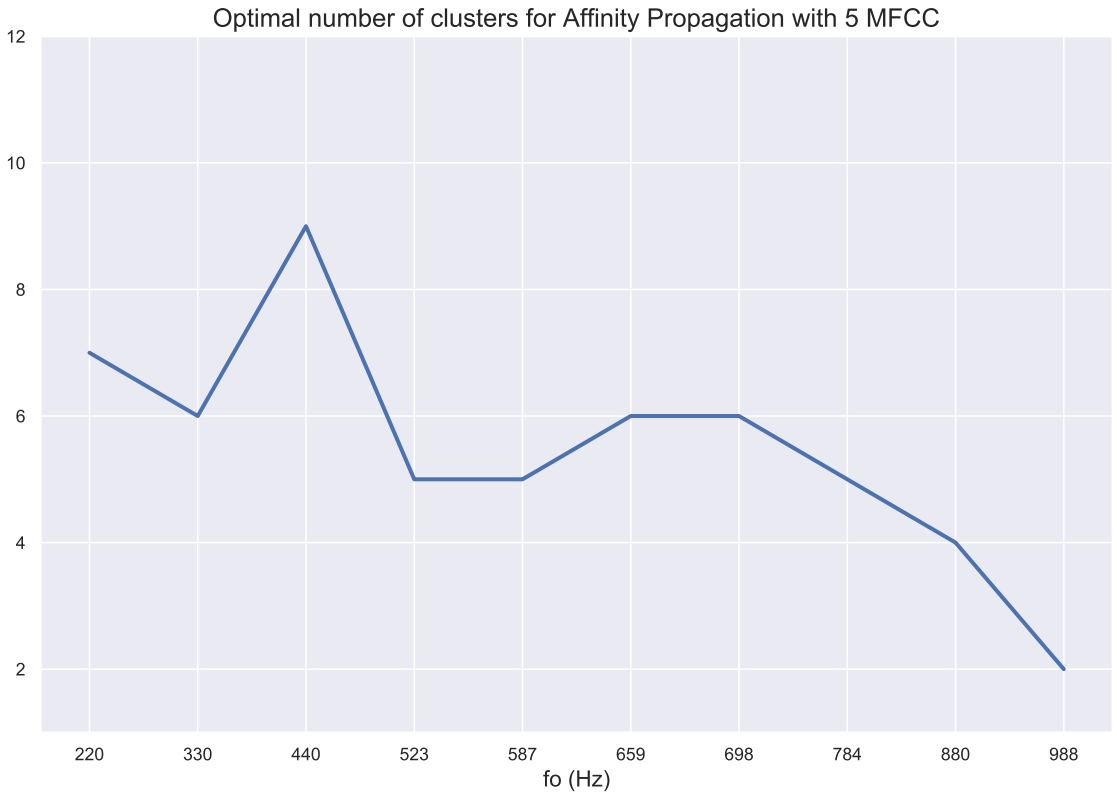

Figure 17: **Evaluation of affinity propagation.** The optimal number of clusters is evaluated for 10 f_0 based on 5 MFCCs.

seem to be an important aid for the validation criterions. Hence, internal validation methods appear to be an inappropriate measure for cluster evaluation in this context.

External validation methods which make use of the ground truth labels are the preferred way to evaluate clusters, because they can partially suggest the correct number of groups for the vowel speech data. Mentioning a convincing and a less appropriate criterion, the V-measure, which is a combination of cluster homogeneity and completeness, performs remarkably well and almost always propose seven or eight clusters. The Fowlkes-Mallows index on the other hand, which gives emphasis to the true positives and ignores the true negatives, does not perform satisfying for

most algorithms and f_o levels, as it proposes an arbitrary number of clusters. In practice, the true class labels are often not available, what is the disadvantage of using external validation techniques. However, because we are in possession of the ground truth data, we can make use of it.
Internal and external evaluation criterions do hardly depend on the number of MFCCs for our vowel speech data. DBSCAN, which does not make use of validation methods, is not influenced by the number of MFCCs, as the estimated number of clusters remains constant for all f_o levels. Mean shift and affinity propagation are slightly influenced by the number of MFCCs, but the overall tendency remains the same.
The f_o level seems to have no impact on the outcome, that is, most algorithms do not behave differently if the f_o level is low or high. Mean shift and affinity propagation make an exception, as these algorithms appear to be influenced by a high f_o level. A possible explanation for this may be that we have fewer data available at the highest f_o level of 988 Hz compared to the other f_o levels.

Conclusion

We describe and analyze clustering algorithms, that is, unsupervised machine learning methods, applied to a speech corpus containing short recordings of the eight Standard German vowels /i y e ϕ ε a o u/.

There exist center-based clustering algorithms like k-means and GMMs, which both prefer convex-shaped clusters. These methods require to define the number of clusters k before running the algorithm. Algorithms like mean shift and DBSCAN estimate a density of the data to decide how many clusters there are in the dataset. Thus, the user does not have to define the number of clusters for these methods. However, abolishing the necessity to determine k groups does not make the clustering task easier, because usually, several hyperparameters have to be defined. This is a similar challenge as figuring out the correct number of clusters. Some algorithms work on graph theoretical concepts. The most prominent one is spectral clustering. Affinity propagation works by passing messages between the data points. We show that certain clustering techniques are able to find an optimal number of groups on its own, for instance mean shift and affinity propagation. Other clustering algorithms rely on evaluation methods to determine the optimal number of clusters. Some clustering methods, like k-means, are not deterministic and need several runs of the algorithm to find an optimal solution. Algorithms like spectral or hierarchical clustering are deterministic and always find the same results. All these methods have strength but also limitations. We can deduce that there exists no best clustering algorithm. The user tries out different methods and decides which one is most appropriate for the analyzed problem.

The evaluation shows that there are clustering algorithms and evaluation criterions that are more appropriate for the vowel speech data than others. Internal validation methods produce less consistent results and frequently underestimate the true number of groups. Certain external validation criterions, particularly the V-measure, often reproduce the correct number of clusters.

The f_o level does not have an impact on the performance of the clustering methods or the validation criterions, except for mean shift and affinity propagation. Some

criterions make wrong suggestions with no clear pattern, while others propose the correct number of groups up to 1 kHz.

6.1 Future Work

Future work based on our thesis could be analyzing MFCC data using classification methods, that is, supervised learning algorithms. This would be a different approach to the unsupervised focus of this thesis.
A further possibility could be evaluating the clustering solutions using a stability-based approach, which is different to internal and external validation criterions [32].
Another attempt could be to use vowel recordings of other speakers, for example of men and children, and different datasets of speech recordings, and comparing the results with our findings.

Appendix

We visualize in this section the 13-dimensional MFCC data using linear and non-linear dimensionality reduction algorithms to get more insights into the clustering results. We use spectral clustering because it can find arbitrary shaped clusters, but other methods would also be appropriate. Clustering algorithms do not provide the true classes, that is, they do not know if one group is vowel a or vowel u. They only provide groups with no naming, for instance group 1, group 2, and so on.

Visualization of MFCC Data using PCA

Principal component analysis (PCA) applied on the MFCC data. PCA can separate the groups, but not in an optimal manner. For the graph with the true labels, PCA does not provide an expressive 2-dimensional visualization, see Figure 18.

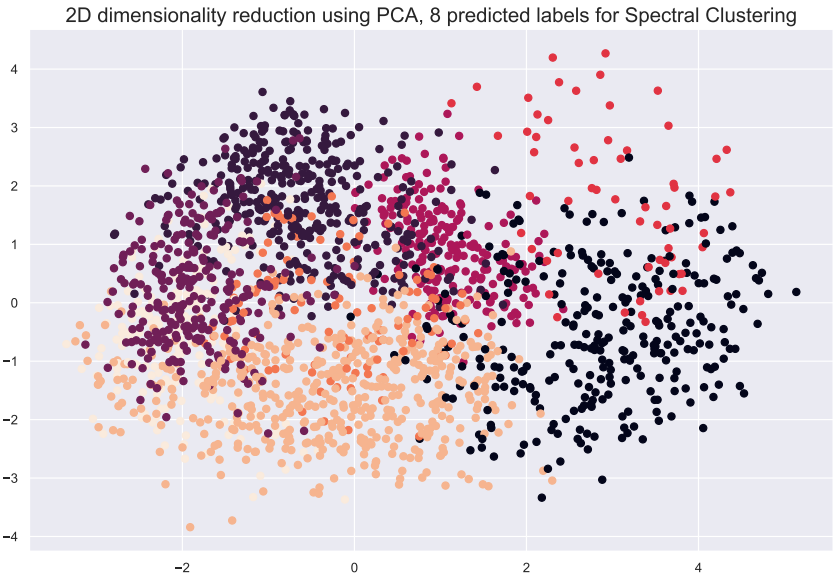

Figure 18: Results of PCA for 13-dimensional MFCC data.

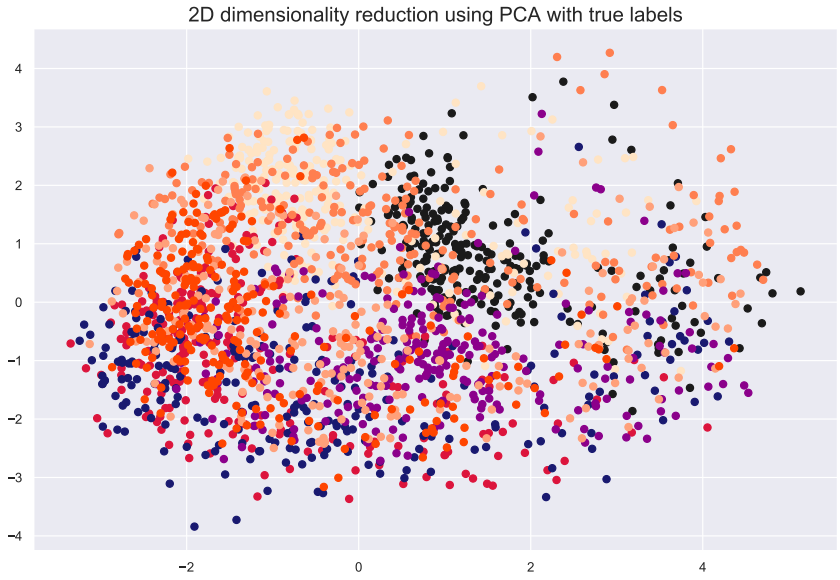

Figure 19: Results of PCA for 13-dimensional MFCC data.

Visualization of MFCC Data using t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimension reduction method, that can reconstruct more complicated shapes than PCA. There is some similarity with the true labels, see Figure 20.

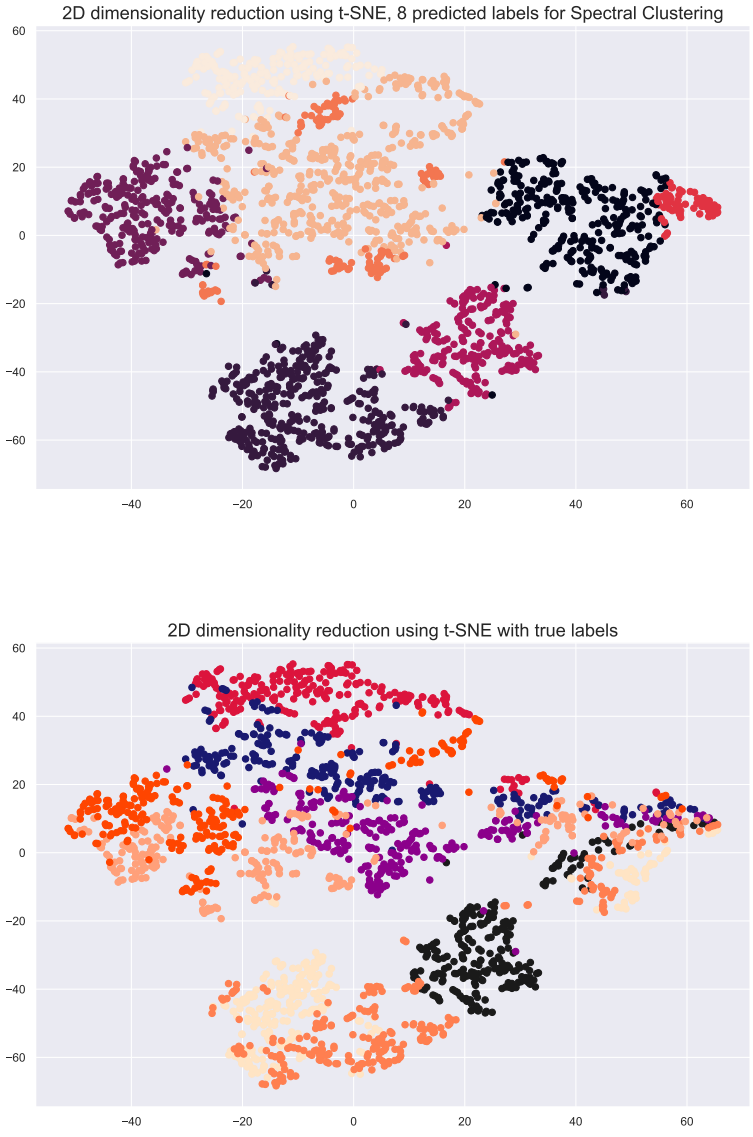

Figure 20: Results of t-SNE for 13-dimensional MFCC data.

Visualization of MFCC Data using Isomap

Isomap is like t-SNE another non-linear dimensionality reduction method, that tries to reconstruct a lower-dimensional manifold. The results agree with some of the true labels, see Figure 21.

Figure 21: Results of Isomap for 13-dimensional MFCC data.

Python Implementation of k-Means

We show here the Python code for creating MFCC data and running the k-means algorithm with a number of clusters from 2 to 8. The code is similar for other clustering methods. The scikit-learn package is used for the clustering algorithm and for the evaluation metrics. NumPy and pandas are packages for data handling. We make use of seaborn and matplotlib for the visualization of the data.

```
import os
import re
import numpy as np
import pandas as pd
import librosa.display
import seaborn as sns
import matplotlib.pyplot as plt
from validclust.indices import dunn
from sklearn.cluster import MiniBatchKMeans
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score
from sklearn.metrics import silhouette_score, pairwise_distances

np.random.seed(1)
Hz = [220, 330, 440, 523, 587, 659, 698, 784, 880, 988]
opt_clusters_ch = []
opt_clusters_db = []
opt_clusters_sil = []
opt_clusters_dunn = []

iterations = 25

for j in range(1,iterations+1):
    for hz in Hz:
        os.chdir("...Sound\\N\\Normalized\\female")
        info = np.loadtxt("MediaArchiveSelection.txt", dtype=np.str_,
```

```
delimiter="\t")
df = pd.DataFrame(info)
df = df.loc[(df[6] == "N")]  # select only "non-style"
df = df.loc[(df[14] == str(hz))]  # extract Hertz
df = list(df[0])

os.chdir("...Sound\\N\\Normalized\\female\\all")

files = os.listdir()
r = re.compile(".*wav")
files = list(filter(r.match, files))

files_corner = []

for file in files:
    if file[:6] in df:
        files_corner.append(file)

files = files_corner
files = list(np.random.choice(files, int(0.7*len(files)),
replace=False))

n_mfcc = 13

x, sr = librosa.load(files[0], sr=44100)
mfcc = librosa.feature.mfcc(x, sr=sr, n_mfcc=n_mfcc)
mfcc = np.transpose(mfcc)
center = int(mfcc.shape[0]/2)  # Compute +/- 350 ms from the center
mfcc = mfcc[center-35:center+35, :]
mfcc = np.mean(mfcc, axis=0)  # Average the data to 1 value

files = files[1:]
```

```
        for file in files:
            x, sr = librosa.load(file, sr=44100)
            melfcc = librosa.feature.mfcc(x, sr=sr, n_mfcc=n_mfcc)
            melfcc = np.transpose(melfcc)
            center = int(melfcc.shape[0]/2)
            melfcc = melfcc[center-35:center+35, :]
            melfcc = np.mean(melfcc, axis=0)
            mfcc = np.vstack((mfcc, melfcc))

        score_ch = []
        score_db = []
        score_sil = []
        score_dunn = []
        n_cl = 8
        for k in range(2, n_cl+1):
            clust = MiniBatchKMeans(n_clusters=k, batch_size=50).fit(mfcc)
            labels = clust.labels_
            score_ch.append(calinski_harabasz_score(mfcc, labels))
            score_db.append(davies_bouldin_score(mfcc, labels))
            score_sil.append(silhouette_score(mfcc, labels))
            mfcc_dist = pairwise_distances(mfcc)
            score_dunn.append(dunn(mfcc_dist, labels))
        opt_ch = np.argmax(score_ch)+2
        opt_db = np.argmin(score_db)+2
        opt_sil = np.argmax(score_sil)+2
        opt_dunn = np.argmax(score_dunn)+2
        opt_clusters_ch.append(opt_ch)
        opt_clusters_db.append(opt_db)
        opt_clusters_sil.append(opt_sil)
        opt_clusters_dunn.append(opt_dunn)

db_sampling = []
```

```
for j in range(len(Hz)):
    count = 0
    l = []
    for i in range(iterations):
        l.append(opt_clusters_db[i+count+j])
        count += len(Hz)-1
    db_sampling.append(np.median(l))

ch_sampling = []
for j in range(len(Hz)):
    count = 0
    l = []
    for i in range(iterations):
        l.append(opt_clusters_ch[i+count+j])
        count += len(Hz)-1
    ch_sampling.append(np.median(l))
dunn_sampling = []
for j in range(len(Hz)):
    count = 0
    l = []
    for i in range(iterations):
        l.append(opt_clusters_dunn[i+count+j])
        count += len(Hz)-1
    dunn_sampling.append(np.median(l))

sil_sampling = []
for j in range(len(Hz)):
    count = 0
    l = []
    for i in range(iterations):
        l.append(opt_clusters_sil[i+count+j])
        count += len(Hz)-1
```

```
    sil_sampling.append(np.median(l))

sns.set_theme()
sns.set_context(rc={"font.size":12, "axes.titlesize":20, "axes.labelsize":14})
x_labels = list(range(len(Hz)))

window = "221"
ax1 = plt.subplot(int(window), title="Calinski-Harabasz for k-means with {}
MFCC".format(n_mfcc))
ax1.plot(x_labels, ch_sampling, color="b")
plt.ylim(1, 10)
plt.xlabel("fo (Hz)")
plt.xticks(x_labels, Hz)

window = "222"
ax1 = plt.subplot(int(window), title="Davies-Bouldin for k-means with {}
MFCC".format(n_mfcc))
ax1.plot(x_labels, db_sampling, color="r")
plt.ylim(1, 10)
plt.xlabel("fo (Hz)")
plt.xticks(x_labels, Hz)

window = "223"
ax1 = plt.subplot(int(window), title="Silhouette for k-means with {}
MFCC".format(n_mfcc))
ax1.plot(x_labels, sil_sampling, color="g")
plt.ylim(1, 10)
plt.xlabel("fo (Hz)")
plt.xticks(x_labels, Hz)

window = "224"
ax1 = plt.subplot(int(window), title="Dunn for k-means with {}
```

```
MFCC".format(n_mfcc))
ax1.plot(x_labels, dunn_sampling, color="y")
plt.ylim(1, 10)
plt.xlabel("fo (Hz)")
plt.xticks(x_labels, Hz)
plt.rcParams['figure.figsize'] = [18,12]
```

References

[1] Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Perez, J. M., & Perona, I. (2013). An extensive comparative study of cluster validity indices. In *Pattern Recognition*, 46(1), 243-256.

[2] Ascher, U. M., & Greif, C. (2011). A First Course on Numerical Methods. Society for Industrial and Applied Mathematics.

[3] Barber, D. (2012). Bayesian Reasoning and Machine Learning. Cambridge University Press.

[4] Benesty, J., Sondhi, M. M., & Huang, Y. (2007). Springer Handbook of Speech Processing. Springer.

[5] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[6] Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. In *Communications in Statistics-theory and Methods*, *3*(1), 1-27.

[7] Carreira-Perpinan, M. A. (2015). A review of mean-shift algorithms for clustering. arXiv:1503.00687.

[8] Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. In *IEEE transactions on pattern analysis and machine intelligence*, (2), 224-227.

[9] De Amorim, R. C., & Hennig, C. (2015). Recovering the number of clusters in data sets with noise features using feature rescaling factors. In *Information Sciences*, 324, 126-145.

[10] Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 551-556.

[11] Ding, J., Tarokh, V., & Yang, Y. (2018). Model selection techniques: An overview. In *IEEE Signal Processing Magazine*, 35(6), 16-34.

[12] Dueck, D., & Frey, B. J. (2007). Non-Metric Affinity Propagation for Unsupervised Image Categorization. In *2007 IEEE 11th International Conference on Computer Vision*, 1-8.

[13] Elkan, C. (2003). Using the Triangle Inequality to Accelerate k-Means. In *Proceedings of the 20th International Conference on Machine Learning*, 147-153.

[14] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 96(34), 226-231.

[15] Farber, I., Gunnemann, S., Kriegel, H. P., Kroger, P., Muller, E., Schubert, E., & Zimek, A. (2010). On using class-labels in evaluation of clusterings. In *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*, 1.

[16] Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. In *Science*, 315(5814), 972-976.

[17] Friedrichs, D., Maurer, D., Rosen, S., & Dellwo, V. (2017). Vowel Recognition at Fundamental Frequencies up to 1 kHz Reveals Point Vowels as Acoustic Landmarks. In *The Journal of the Acoustical Society of America*, 142(2), 1025-1033.

[18] Girolami, M. (2002). Mercer kernel-based clustering in feature space. In *IEEE transactions on neural networks*, 13(3), 780-784.

[19] Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. In *Journal of intelligent information systems*, 17(2), 107-145.

[20] Hassani, M., & Seidl, T. (2017). Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. In *Vietnam Journal of Computer Science*, 4(3), 171-183.

[21] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

[22] Held, L., & Sabanes Bove, D. (2014). Applied Statistical Inference. Springer, Berlin Heidelberg.

[23] Hubert, L., & Arabie, P. (1985). Comparing Partitions. In *Journal of Classification*, 2(1), 193-218.

[24] Jacod, J., & Protter, P. (2012). Probability Essentials. Springer Science & Business Media.

[25] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. In *ACM computing surveys (CSUR)*, 31(3), 264-323.

[26] Jain, A. K. (2010). Data Clustering: 50 Years Beyond K-Means. In *Pattern recognition letters*, 31(8), 651-666.

[27] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

[28] Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics and Natural Language Processing. Upper Saddle River, NJ: Prentice Hall.

[29] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. In *Journal of Basic Engineering*.

[30] Kathiresan, T., Maurer, D., & Dellwo, V. (2019). Highly spectrally undersampled vowels can be classified by machines without supervision. In *The Journal of the Acoustical Society of America*, 146(1), 1-7.

[31] Kleinberg, J. (2003). An Impossibility Theorem for Clustering. In *Advances in Neural Information Processing Systems*, 463-470.

[32] Lange, T., Roth, V., Braun, M. L., & Buhmann, J. M. (2004). Stability-based validation of clustering solutions. In *Neural computation*, 16(6), 1299-1323.

[33] Maulik, U., & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. In *IEEE Transactions on pattern analysis and machine intelligence*, 24(12), 1650-1654.

[34] Maurer, D., Heureuse, C., Suter, H., Dellwo, V., Friedrichs, D., & Kathiresan, T. (2018). The Zurich Corpus of Vowel and Voice Quality, Version 1. 0. In *Proceedings of Interspeech 2018, Hyderabad, India*, 1417-1421.

[35] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. In *Proceedings of the 14th Python in Science Conference*, 18-25.

[36] Meila, M. (2007). Comparing Clusterings - An Information Based Distance. In *Journal of Multivariate Analysis*, 98(5), 873-895.

[37] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

[38] Ng, A., Jordan, M., & Weiss, Y. (2001). On Spectral Clustering: Analysis and an Algorithm. In *Advances in Neural Information Processing Systems*, 14, 849-856.

[39] Ng, A. Y., & Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems*, 841-848.

[40] Pelleg, D., & Moore, A. W. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *International Conference on Machine Learning*, 1, 727-734.

[41] Pfitzner, D., Leibbrandt, R., & Powers, D. (2009). Characterization and evaluation of similarity measures for pairs of clusterings. In *Knowledge and Information Systems*, 19(3), 361-394.

[42] Rao, K. S., & Manjunath, K. E. (2017). Speech Recognition Using Articulatory and Excitation Source Features. Springer.

[43] Rasmussen, C. E. & Williams, C. K. I (2006). Gaussian Processes for Machine Learning. Springer, Berlin, Heidelberg.

[44] Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 410-420.

[45] Russell, S., & Norvig, P. (2002). Artificial Intelligence: A Modern Approach. Addison Wesley.

[46] Sahidullah, M., & Saha, G. (2012). Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. In *Speech communication*, 54(4), 543-565.

[47] Schubert, E., Hess, S., & Morik, K. (2018). The Relationship of DBSCAN to Matrix Factorization and Spectral Clustering. In *LWDA*.

[48] Schubert, E., & Rousseeuw, P. J. (2019). Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms. In *International Conference on Similarity Search and Applications*, Springer, Cham, 171-187.

[49] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

[50] Steinley, D. (2004). Properties of the Hubert-Arable Adjusted Rand Index. In *Psychological Methods*, 9(3), 386.

[51] Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer Science & Business Media.

[52] Tan, P. N., Steinbach, M., & Kumar, V. (2016). Introduction to Data Mining. In *Pearson Education India*.

[53] Vallabha, G. K., McClelland, J. L., Pons, F., Werker, J. F., & Amano, S. (2007). Unsupervised learning of vowel categories from infant-directed speech. In *Proceedings of the National Academy of Sciences*, 104(33), 13273-13278.

[54] Vinh, N. X., Epps, J., & Bailey, J. (2010). Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. In *The Journal of Machine Learning Research*, 11, 2837-2854.

[55] Von Luxburg, U. (2007). A Tutorial on Spectral Clustering. In *Statistics and Computing*, 17(4), 395-416.

[56] Wasserman, L. (2006). All of Nonparametric Statistics. Springer Science & Business Media.

[57] Xu, M., Duan, L. Y., Cai, J., Chia, L. T., Xu, C., & Tian, Q. (2004). HMM-Based Audio Keyword Generation. In *Pacific-Rim Conference on Multimedia*, Springer, Berlin, Heidelberg, 566-574.