Term Paper Data Science 1

Docent: Prof. Dr. Lena Wiese
Semester: Summer Term 2021

Institute of Computer Science
Goethe-Universität Frankfurt a. M.

Author: M. A.
⟨your Student ID⟩
⟨your.email@ddre.ss⟩
⟨branch of study (Bachelor/Master, semester count)⟩
L. B.
⟨your Student ID⟩
⟨your.email@ddre.ss⟩
⟨branch of study (Bachelor/Master, semester count)⟩
B. L.
⟨your Student ID⟩
⟨your.email@ddre.ss⟩
⟨branch of study (Bachelor/Master, semester count)⟩
J. M.
⟨your Student ID⟩
⟨your.email@ddre.ss⟩
⟨branch of study (Bachelor/Master, semester count)⟩

Date: June 9, 2021

Chosen Project Topic:

- T3 Clustering

Abstract

Not done yet.

Contents

1 Problem Description

Clustering is a long-known method in the field of Data Science and Machine Learning. It describes the process of grouping elements of a given dataset into a finite number of distinct segments such that the elements of one segment are as similar as possible to one another while maximizing the dissimilarity between different segments. The resulting segments are called clusters. Finding clusters is of interest to gain information on patterns or structures characterizing a given dataset [1]. Each of the given data points is interpreted as a multidimensional feature vector and clusters are found by calculating vector similarities. Similarity or dissimilarity is measured using different kinds of metrics which are chosen according to the underlying data space [2].

As no a priori knowledge about the given dataset is assumed, distinguishing it from classification contexts, clustering is an unsupervised learning method. There exist several different clustering techniques which differ in their embedded assumptions on the cluster shapes or the requirement of various parameters such as the number of clusters or the bandwidth.

There is a wide variety of applications for clustering methods in research and professional contexts ranging from social network analysis to image compression and even oil well operation.

In this report, K-Means Clustering, Affinity Propagation Clustering, Mean Shift Clustering and Spectral Clustering are investigated.

Each of these clustering algorithms is applied on each of the chosen datasets and clustering performance is evaluated using cluster validation indexes. In the end, results generated applying the different algorithms are compared to each other.

2 Description of Specific Methods and Algorithms

2.1 K-Means

written by B.L.

On the surface, the idea for K-Means clustering is straight forward: an initial grouping of observations closest to starting points whereafter the starting points are updated to be placed at the center of these groupings. A new grouping step is performed and centers updated until the groupings do not change any further [3,4]. This very brief introduction requires elaboration.

One way to look at clustering mathematically is to consider an objective function, similar to [5]

$$\sum_{k=1}^{K} \sum_{x \in C_k} w_x d(x, c_k) \tag{1}$$

where $K \in \mathbb{N}$ is the number of clusters $C_k, k \in \{1, \ldots, K\}$ of which the centers are c_k. Observations are $x \in \mathcal{X}$ and their weights w_x with $\mathcal{X} = \{x_1, \ldots, x_m\}$ being the entire data set. The data in $\mathcal{X}$ lives inside a feature space $\mathcal{F}^n$ with $n \in \mathbb{N}$ the number of different features. There also exists a measure on $\mathcal{F}^n$ that allows to calculate the distance d between points.

The most common specification of K-Means and often most practical in real world applications is the feature space being a euclidean space $\mathbb{R}^n$ and used for its distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, fittingly, the euclidean distance (written as $\|\cdot\|$). Using these, the optimization problem presents as

$$\min \sum_{k=1}^{K} \sum_{x \in C_k} \|x - c_k\|^2 \tag{2}$$

$$= \min \sum_{k=1}^{K} \sum_{x \in C_k} \left\| x - \frac{1}{|C_k|} \sum_{y \in C_k} y \right\|^2 \tag{3}$$

The inner sum in equation 2 can be seen as the sum of squared errors (SSE) for each cluster. Notably this version omits the weighting of the points. Introducing (equal) weights for all points $(\ldots) \sum_{x \in C_k} \frac{1}{|C_k|} \|x - c_k\|^2$ will not change the clustering result but allows for it to be seen as the intra-cluster variance, its minimizaiton across clusters directly reflecting one of the objectives mentioned in section 1.

Achieving the goal described above globally is an NP-hard problem but convergence to local optima can be achieved rather efficiently. This leads to the question of how an approximation can be achieved in practice. A method similar to the one described by [5, 6] works as follows:

The one parameter that has to be specified for this algorithm is K, the number of clusters one wishes to produce. Once a K is chosen, the method can start by picking K locations in the feature space $\mathcal{F}^n$ (each feature is a column in the data set). There are more advanced methods to be explored later but for the moment we assume these locations, so called centroids, c_k to be picked randomly. For every observation (row in the data) there is a centroid with the lowest distance (for equidistant centroids there is a method in place for assignment). In this step each observation is assumed to be part of the cluster C_k whose centroid c_k is closest to it. After this assignment the centroids are relocated to the ”center” of their respective first step clusters (more on this below). Hereafter the method starts again from the point of calculating the distance between all observations and centroids. Observations are newly assigned to clusters, centroids are relocated and the method starts again. This procedure is repeated until the centroids are stationary (again, a closer look into this follows below). At this point a clustering has been achieved which satisfies the described method, but cannot necessarily be assumed to be the best clustering possible given the inputs. The entire procedure shall be redone from the top, beginning with new starting points, until one is reasonably satisfied with the stability of the result given a predefined method of evaluation. This series of operations can be summarized in a short amount of pseudo-code

Algorithm 1 Pseudo K-Means

```
place initial centroids $c_i$ ①
while centroids not stationary ④ do
    if observations have labels then
        update centroids $c_i$ to center of current clusters $C_i$ ③
    end if
    calculate all distances $d_{n,k_i}$ ②
    assign each observation to min distant centroid via label $i$
end while
```

where each of the numbers ① to ④ refer to a topic mentioned above and further examined below.

Clustering with the described method is fast, commonly requiring fewer iterations than observations [7]. The naive upper bound is trivially $\mathcal{O}(K^n)$ (trying every possible configuration), with [8] showing worst case run time to be superpolynomial ($\mathcal{O}(m^{(K+2/n)})$) with common runtimes being polynomial.

① Initial placement of the centroids can be achieved in different ways. The most basic method is placing them at random. On singular runs of the algorithm this can lead to problems as can be seen in the example in figure 1.

Figure 1: Example with 1 dimension and 10 points bad starting position

It can be seen that this initial placement does allow for *a* clustering in the end, though it does not achieve what a human would describe as a fitting result. In two ways a solution to this problem can lie in repeating the method multiple times with different initialization. Firstly a new placement of initial centroids can lead to different clustering results which can be more fitting to the observer and secondly introducing a measure of fit and comparing it across runs to select the clustering that maximizes (minimizes) the measure can objectivize the selection of the "best" clustering. Commonly used as a measure are SSE and intra-cluster variance themselves, while other methods will be discussed more thoroughly in section 5. The repetition of clustering can therefore alleviate some problems introduced by randomly selecting starting points.

While computational capabilities are ever growing and trying more initializations should provide a higher chance of achieving better clustering results, it may still be useful to attempt improving on pure randomness.

One early proposition is selecting the center of the entire data set $\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} x$ as a first centroid and then randomly going through all observations and picking the subsequent points if they exceed a predefined distance threshold [9].

To ensure that initial points cannot fall far from the data [10, 11] propose going a step further by only selecting observations as initial points by randomly selecting observations. This can save some iterations required to "pull" centroids towards the data and can prevent the formation of empty clusters. Assuming no inner logic to the ordering of data in a data set (which might of course exist and presents a problem to this method) an implementation of K-Means in SPSS [12] selects the first i observations in the data set as starting points [13].

A number of methods expand on the idea of working with observations by first selecting a single observation and then seeking to select more not too close to the first one, which appears sensible computationally as well as considering the goal of identifying seperate clusters.

One such idea is to randomly select a starting point and subsequently selecting points with the maximum minimal distance to the previously selected [14], often called *maximin*.

A popular method going by the name *kmeans++* developed by [15] extends the idea by [13] described above. After arbitrarily selecting a first observation, the remaining $K-1$ initial points are picked in a random process. Instead of assuming an even distribution all observations are assigned probabilities proportional to the squared distance to their nearest previously picked initial centroid. Despite the additional computational aufwand in the beginning, this method shows reductions in terminal errors as well as time to convergence of up to half compared to naive random selection on $\mathcal{F}^n$. There are implementations of *kmeans++* for most popular machine learning implementations, including the one described in section 4.

This selection of methods is by no means exhaustive but something something ..

② In theory the practical application of the algorithm described above can be adapted to a plethora of distance measures to varying degrees of success. As described the most natural is the euclidean distance as it maps directly to variance minimization. It has been shown by [16] that other L^p norms than L^2 (like euclidean) can be used successfully, L^1 proving particularly good when applied for high dimensional cases. It can be shown by [17], though, that using metrics other than euclidean distances may cause the method to fail to converge. This may in some cases be alleviated by a clever selection of thresholds as described in ④. The choice of distance function will affect the clustering result. Opting for the euclidean distance will lead to clusters tending towards *n-spherical* shapes, applying Manhattan distance *hypercubes* and so on. Befitting the K-Means designation the ideal use case is data containing mostly contiguous subsets distributed around a finite number of means. Illustration 2 depicts examples of data suitable (1-3) and not suitable (4-6) for K-Means application.

Figure 2: Data shapes and K-Means-clustering success

③ The K-Means name makes it sufficiently clear that the selected method of calculating the position of centroids is by taking the mean of the observations grouped in the particular cluster. However, other interpretations of the concept of a mean have been devised in order to tackle specific limitations of the described method. One of the best known is *K-Medoids*. A method described similar to K-Means by [18] proposes to always select the most central observations of each cluster by minimizing the sum of pairwise dissimilarities.

This method is generally more robust to outliers and can perform better when incorporating categorical data.

④ An important question to answer is when to stop iterating. There are mainly three options (cf. [19]): Execution can be stopped once the centroids stop changing. It is possible though to construct examples where this is unlikely to happen. Then this criterion can be expanded to stopping when the distance of centroids between iterations is below a threshold like [6] suggests. Another idea is to stop execution when intra-cluster variance (or another measure) stops improving, respectively improves less than a threshold, for a defined number of iterations. The third option is to stop when points remain in the same cluster for multiple iterations. In many software implementations there is an option to hard cap the number of iterations to prevent excessively long execution times or simply in place of one of the described criteria.

The elephant in the room remains: how to choose K? Looking at equation 2 makes it obvious that optimization improves when increasing K. It is also obvious that simply adding more clusters is not the goal. Domain knowledge about the data set can inform the decision but the application case for an unsupervised method like K-Means typically lacks this type of information. A brief overview of methods to determine K by [20, 21] include among others: visualizing the clustered data, density, internal validation measures (see section 5), gap statistic, or their own distortion measure. One common method of determining K from these statistical methods is then to apply the so-called *elbow rule*: clustering is performed for multiple different K and the corresponding statistics are calculated. The results are then plotted against each other. Often the resulting plot exhibits a ”kink” at a certain point after which its trajectory flattens out. Its position then is used as an approximation of K. Viability of this method varies depending on the underlying data.

2.2 Affinity Propagation

by Jonas Mertens

Affinity Propagation is a machine learning algorithm which is used to create clusters. It was developed by Brendan J. Frey and Delbert Dueck in 2007. This algorithm does not require an input which specifies how many clusters are created. The algorithm finds the clusters by itself. Like

other clustering algorithms, Affinity Propagation iteratively sends “messages” between data points. In detail, the algorithm calculates responsibility and availability in each iteration (this will be explained later on). It iterates until all data points converge. After this, the algorithm has found the clusters for the input dataset. A disadvantage of Affinity Propagation is the computation complexity. Assume there are n data points, it results in a complexity of $O(n^2)$ because of the loop through all data points in every iteration. The runtime depends on the number of iterations T until the algorithm converges. Because of this, the runtime is $O(T * n^2)$.
Algorithm description:
$x_1, x_2, \ldots, x_n$ is a set of datapoints and the input of the algorithm. s[i,k] describes the similarity between x_i and x_k. In the Scikit-Learn (sklearn) implementation, similarity is measured using the negative Euclidean distance, but this algorithm also works with other type of similarities.

The responsibility describes how is k suitable to be an exemplar for i:

$$r(i,k) \leftarrow s(i,k) - max\left[a(i,k') + s(i,k')\forall k \neq k'\right] \tag{4}$$

The availability describes what happens if i should be chosen k as an exemplar.

$$a(i,k) \leftarrow min\left[0, r(k,k) + \sum_{i' s.t. i' \notin \{i,k\}} r(i',k)\right] \tag{5}$$

I am making use of the sklearn implementation of the Affinity Propagation algorithm. An important parameter is the preference where you can chose which how many exemplars are used.

The values for r(i,k) and a(i,k) are updated that after t iteration they are convergent:

$$r\left[i,k\right] = (1-\lambda)r\left[i,k\right] + \lambda r'\left[i,k\right] \tag{6}$$

$$a\left[i,k\right] = (1-\lambda)a\left[i,k\right] + \lambda a'\left[i,k\right] \tag{7}$$

λ is a dumping factor to avoid numerical oscillation. λ can be a float between 0.5 and 1.0. In my implementation the damping factor is 0.9.

2.3 Mean Shift

written by Leonie Bender

Mean Shift is an algorithm which can be used to find distinct clusters within a given dataset. The method was first proposed in 1975 by Fukunaga and Hostetler [22] and is categorized as density based clustering. It is assumed that data points of a given dataset are sampled from a distribution, the probability density function. The main idea is to define clusters in dense regions of the feature space. These clusters are defined around centroids corresponding to peaks, so-called *modes*, of the underlying probability density function. One can imagine that each data point is shifted to its nearest local peak of the density function and all the data points located at the same maximum are assigned to the same cluster around this mode. As the underlying distribution which determines the cluster centroids is not given, Mean Shift makes use of the concept of kernel density estimation. Local structures of the estimated probability density function determine both, the number of resulting clusters as well as which data points are assigned to which cluster [23].
The only input the user has to provide for Mean Shift Clustering is the *bandwidth* or *scale* [24]. The bandwidth parameter $h > \mathit{0}$ determines the kernel's radius. The larger the bandwitdh is chosen, the larger is the kernel's radius, resulting in a smoother estimated density surface. Analogously, the smaller the bandwidth is chosen, the smaller is the kernel's radius and hence this results in an estimated density surface containing a peak for each point. Accordingly, if the bandwidth parameter value is chosen too small, Mean Shift will assign each data point to it's own cluster. In contrast to that, if the bandwidth parameter is chosen too big, Mean Shift will assign all data points to one single cluster.

For a given dataset consisting of n data points in the d-dimensional space, the kernel density estimator, using kernel $K(x) = c_{k,d} \cdot k\left(||x||^2\right)$ with bandwidth $h > \mathit{0}$, is given through

$$\hat{f}_{h,K} = \frac{c_{k,d}}{n \cdot h^d} \sum_{i=1}^{n} k\left(\left|\left| \frac{x - x_i}{h} \right|\right|^2 \right).$$

As explained above, modes of this density have to be found as each mode determines one cluster centroid. Modes correspond to stationary points which fulfill the requirement that $\nabla \hat{f}(x) \overset{!}{=} 0$. Computing the gradient of the estimated density function $\hat{f}$ and afterwards performing some further algebraic manipulations yields the following formula:

$$\hat{\nabla} f_{h,K}(x) = \underbrace{\frac{2 \cdot c_{k,d}}{n \cdot h^{d+2}} \left[\sum_{i=1}^{n} g\left(\left|\left| \frac{x - x_i}{h} \right|\right|^2 \right) \right]}_{\text{proportional to } \hat{f}_{h,K}} \underbrace{\left[\frac{\sum_{i=1}^{n} x_i \cdot g\left(\left|\left| \frac{x-x_i}{h} \right|\right|^2 \right)}{\sum_{i=1}^{n} g\left(\left|\left| \frac{x-x_i}{h} \right|\right|^2 \right)} - x \right]}_{\text{mean shift}} .$$

While the first term is proportional to the above stated density estimate at point x, the second term describes the actual mean shift. It denotes the shift of point x, on which the kernel center is located, to the weighted mean of all the neighboring data points within the kernel's bandwidth. It can be shown that the mean shift vector always points in the direction of maximal increase of density of data points [23]. Local peaks of the probability density function are thus found by iteratively performing the mean shift as a gradient ascent method.
The iterative mean shift approach provides a way to determine modes without having to estimate the complete density function beforehand [23].

Initially, each of the given data points is chosen to be a candidate cluster centroid. In each iteration, all the candidate centroids are updated by shifting them to the mean of their respective neighboring data points. As the mean shift vector always points into the direction of maximal increase, repeating this procedure finally locates local maxima in the density function. The lower the density of data points, the larger the mean shift steps get. This means that data point density automatically regulates the step size and therefore, mean shift is an adaptive form of a gradient ascent method. After the mean shift of each data point is performed, the next iteration starts by placing the kernel on the shifted data point and again shifting it to the mean of its respective neighbors [25].

The algorithm stops if the shift of centroids falls below a predefined threshold. All data points that reside at the same stationary point are summarized into the same cluster where the local maximum of the underlying density function is used as the cluster centroid.

The procedure of Mean Shift Clustering can be summarized by the following pseudocode.

As the search for all the neighboring data points is computationally costly and in each iteration a large number of such searches has to be performed, the algorithm is not highly scalable. While in lower dimensions, complexity tends towards $O(T \cdot n \cdot log(n))$, in higher dimensions complexity tends towards

Algorithm 2 Mean Shift Clustering

```
for $i$=1,2,..., N do
    while $mean\ shift\ exceeds\ threshold$ do
        $m(x_i) \leftarrow compute\ mean\ of\ neighboring\ data\ points\ of\ x_i$
        $x_i \leftarrow m(x_i)$
    end while
    Store $z_i \leftarrow x_i$
end for
Identify clusters $\{C\}_{1...m}$ grouping together all nearby $z_i s$
Return clusters $\{C\}_{1...m}$
```

$O(T \cdot n^2)$, where n denotes the number of samples and T denotes the number of data points. If all given data points are used as samples, this results in a complexity of $O(n^2 \cdot log(n))$ in lower dimensions and analogously to a complexity of $O(n^3)$ in high dimensions. This complexity can be improved by initializing only a fraction of data points as candidate cluster centroids or by only taking a subset of all the given data points as samples [26]. Details on this are provided in section 4.3.

Unlike other clustering algorithms, Mean Shift is a nonparametric method, meaning it neither makes any assumption on the number of resulting clusters nor on their shape. The algorithm can thus be used to find clusters of arbitrary, even non-elliptical shapes. This makes Mean Shift applicable to real world datasets where clusters of convex shapes would possibly introduce severe artifacts [23].

In addition to that, Mean Shift Clustering is robust against outliers. This can be verified as outliers would only lead to the definition of a cluster consisting of a single data point instead of shifting the centroid of another cluster. At the same time, this can also be seen as a disadvantage of the Mean Shift algorithm. The resulting clusters caused by outliers are not meaningful and do not have any reasonable interpretation when analyzing the resulting clustering. The rationale of another drawback of Mean Shift algorithm is based on the concept of kernel density estimation. In high dimensions, kernel density estimation may react sensitively to changes in the bandwidth parameter value. Therefore, Mean Shift Clustering is better applicable to low-dimensional datasets [27].

2.4 Spectral Clustering

written by M.A.

Spectral clustering is an unsupervised learning algorithm. It treats each data point as a graph-node and performs the clustering problem into a graph-partitioning problem. The affinity rather then the absolute location (i.e. k-means) determines what points are set in which cluster. The algorithm consists of four basic steps:

1. The construction of a similarity graph:
During this step the Similarity Graph is being built in the form of an adjacent matrix, which is presented by A. The adjacency matrix can be built as an Epsilon-neighbourhood Graph, K-Nearest Neighbours strategy or a fully-connected graph.

The Epsilon-neighbourhood Graph sets the parameter epsilon already. Afterwards each point is connected to all the points which lie in the epsilon -radius. The graph which is built in this case is an undirected and unweighted graph.

By using the k-nearest neighbours strategy a parameter k is set as well. Then, for two vertices u and v, an edge is directed from u to v only if v is among the k-nearest neighbours of u. To build the fully-connected graph, each point is connected with an undirected edge-weighted by the distance between the two points to every other point. Since this approach is used to model the local neighbourhood relationships thus typically the Gaussian similarity metric is used to calculate the distance.

2. Projecting the data onto a lower Dimensional Space: This step is done to account for the possibility that members of the same cluster may be far away in the given dimensional space. Thus the dimensional space is reduced so that those points are closer in the reduced dimensional space and thus can be clustered together by a traditional clustering algorithm. It is done by computing the Graph Laplacian Matrix . To compute it though first, the degree of a node needs to be defined. The degree of the ith node is given by (Formel noch einfügen)

Note that $w_{i,j}$, is the edge between the nodes i and j as defined in the adjacency matrix above. The degree matrix is defined as follows (noch einfügen)

This Matrix is then normalized for mathematical efficiency. To reduce the dimensions, first, the eigenvalues and the respective eigenvectors are calculated. If the number of clusters is k then the first eigenvalues and their eigenvectors are taken and stacked into a matrix such that the eigenvectors are the columns.

3. Clustering the Data: This process mainly involves clustering the reduced data by using any traditional clustering technique – typically K-Means Clustering. First, each node is assigned a row of the normalized of the Graph Laplacian Matrix. Then this data is clustered using any traditional technique. To transform the clustering result, the node identifier is retained.

3 Data Set Description

3.1 Boston House Pricing Data

written by L.B.

The Boston House Pricing Dataset was originally published in 1978 by Harrison and Rubinfeld [28]. In their study the authors used the dataset to investigate how people's willingness to pay for clean air is correlated to different measurements of house data around the area of Boston. In total, 506 samples are included within the dataset, containing fourteen different attribute columns. Six of those attribute values originate from the U.S. Census Service, the remaining originate amongst others from the FBI, the Metropolitan Area Planning Commission, the Massachusetts Taxpayers Foundation, the Massachusetts Department of Education and the MIT Boston project. All data was sampled in 1970. The attributes of each data can be separated into different types, providing information on structure, neighborhood, accessibility or air pollution.

Structural attributes yield information on the state of the house in terms of year of construction or spaciousness. While the *RM* variable holds the numeric value for the average number of rooms, the *AGE* attribute describes the proportion of houses that were built before 1940. Both values are assumed to have a positive correlation on housing values since owning more rooms or owning houses with modern structures is perceived as increasing life quality.

Neighborhood attributes hold details about the socioeconomic status of the environment. This includes the fraction of colored people in the whole population as well as the *LSTAT* attribute, which denotes the amount of people being of a lower educational status. In addition to that, crime rate is included for neighborhoods of the Boston area. The latter attribute is supposed to have a negative effect on housing values as crime rate influences people's level of danger. Another neighborhood attribute stands for the sum of square feet available for residential zoning where constructing buildings like factories is prohibited. Next, the *INDUS* attribute comprises the proportion of industry which comes along with noise, traffic and dirt and is therefore negatively correlated with housing values. Moreover, property tax rate as well as the ratio between pupils and teachers are included. The last socioeconomic attribute classifies whether the respective city area adjoins Charles River.

Accessibility attributes characterize the infrastructure measured by closeness to employment centers and to radial highways.

In order to estimate air quality, the concentration of Nitrogen Oxid in parts per hundred million is measured.

The last attribute is the dependent variable which describes the median value of houses that are occupied by private owners.

While the index of highway accessibility is an integer value and the closeness to Charles River is measured using a boolean variable, all the other attributes are float numbers. The dataset does not contain any empty columns, thus no elimination or preprocessing of the available data is necessary.

3.2 Mall Customer Segmentation

by J. M.

The Data Set Mall Customer Segmentation Data includes basic customer data. It contains a unique Id for each customer, gender, age, the annual income, and the spending Score. In this Score you can have a value between 1 and 100. The distribution of gender of the customer is 56% female and 44 % male. In the preprocessing every entry of the gender column gets a number 0 or 1 depending on the gender is male or female. I do this because than every entry of this dataset contains only numbers. The Age of the Customers is between 18 und 70. The Annual Income of the customers is between 15 and 127.

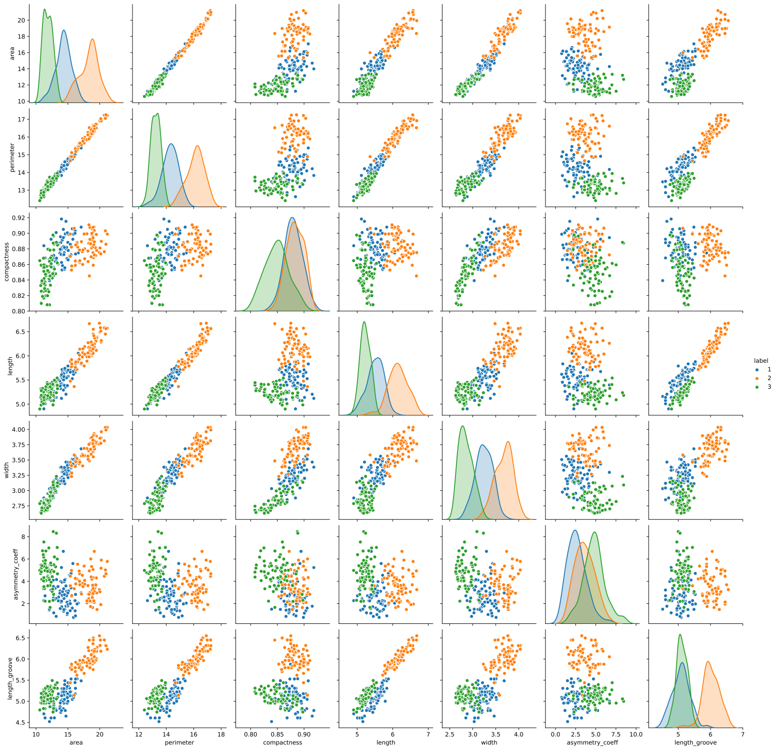
3.3 Seeds Data

written by B.L.

This data set has been used in the work of [29]. The data comprises information on different features of wheat kernels. There are seven species with a total of about 20 varieties of which three can be found in the data: Kama, Rosa and Canadian wheat. There is a total of 210 data points, evenly split between the varieties. The kernels for which data was collected were selected randomly. They were then examined through X-ray imaging. A software called *GRAINS* for this specific application [30] was used to extract the features for each observation:

- area A
- perimeter P
- compactness $C = 4\pi A/P^2$
- length (along groove)
- width
- asymmetry coefficient
- length of kernel groove
- label

Figure 3: Seeds pairplot WIP



A few things can be noted here: compactness is linearly dependent on two other features in the data set, calling into question what value this information can bring to a clustering method. Furthermore the area is not computed from other measurements, but is highly correlated with width and length and the perimeter. These relationships among others can be seen in figure 3. Note that these plots have been colored according to their labels. Density plots on the diagonal already show distinct characteristics for the different varieties of kernels for certain features. For the clustering step of course the labels are removed from the data (this data set was included despite not being the typical application of clustering because it can serve as a sort of ”control” for the clustering methods when looking at evaluation and conclusions).

4 Description of Python libraries used

4.1 Scikit-Learn KMeans

For K-Means clustering we use the implementation provided from [26] in version 0.24.2 which is the latest stable release as of June 9, 2021.

One of the parameters of this function is the choice of algorithm used for the optimization: `full` provides the option to run a method mimicking [6] described exhaustively in section 2.1, while `elkan` has become the default which uses a more efficient algorithm described by [31] to improve run time. This method takes advantage of the triangle inequality in multiple ways, most prominently by reducing the number of distance calculations for each iteration. After updating, distances between centroids are calculated. Then, starting with the previously assigned centroid, distances from observations to centroids are calculated. In the `full` case, this would be done for all observations with all centroids. With the `elkan` , thanks to triangle inequality, the algorithm knows it is not necessary to perform more distance calculations for a particular observation if its distance to the current centroid is below half the distance between the centroid and next closest centroid.

Another option is to specify how to initialize the algorithm. It is possible to specify `random` which selects random observations from the data, to provide a list of starting positions, a custom function, or to use the default which is `kmeans++` as described in section 2.1.

Stopping condition for the algorithm is movement of centroids below a certain threshold, as described in section 2.1, determined by calculation of the Frobenius norm of the difference in position of centroids of two consecutive iterations. The threshold can be set as a parameter. There is also the possibility to specify how many runs with new initializations will be run as well as a maximum number of iterations per run.

4.2 Scikit-Learn Affinity

4.3 Scikit-Learn MeanShift

4.4 Scikit-Learn Spectral Clustering

5 Description of Evaluation Module

There are graph-based indices: C Index, Dunn, Gamma, G+, McClain-Rao, Point Biserial, Silhouette, Tau, and there are prototype based indices: Calinski-Harabasz, Davies-Bouldin, PBM, Ratkowski-Lance, Ray-Turi, Wemmert-Gancarski, Xie-Beni.

So far we use the Dunn Index (DI), Calinski-Harabasz Index (CH), Davies-Bouldin Index (DB), and Silhouette Score (SS), maybe also Gap Statistic (GS).

- Dunn [32] brief explanation and reasons.
- Calinski-Harabas [33] brief explanation and reasons.
- Davies-Bouldin [34] brief explanation and reasons.
- Silhouette [35] brief explanation and reasons.
- Gap [36] brief explanation and reasons.

Explanation of evaluation results, using the big table 1, also using plots. Furthermore brief explanation on the result of each method separately.

Data and Method	DI	CH	DB	SS
Housing				
K-Means	1.567	2.346	3.457	4.856
Mean Shift	1.567	2.346	3.457	4.856
Affinity	1.567	2.346	3.457	4.856
Spectral	1.567	2.346	3.457	4.856
Mall				
K-Means	1.567	2.346	3.457	4.856
Mean Shift	1.567	2.346	3.457	4.856
Affinity	1.567	2.346	3.457	4.856
Spectral	1.567	2.346	3.457	4.856
Seeds				
K-Means	1.567	2.346	3.457	4.856
Mean Shift	1.567	2.346	3.457	4.856
Affinity	1.567	2.346	3.457	4.856
Spectral	1.567	2.346	3.457	4.856
Wine				
K-Means	1.567	2.346	3.457	4.856
Mean Shift	1.567	2.346	3.457	4.856
Affinity	1.567	2.346	3.457	4.856
Spectral	1.567	2.346	3.457	4.856

Table 1: Comparing everything example

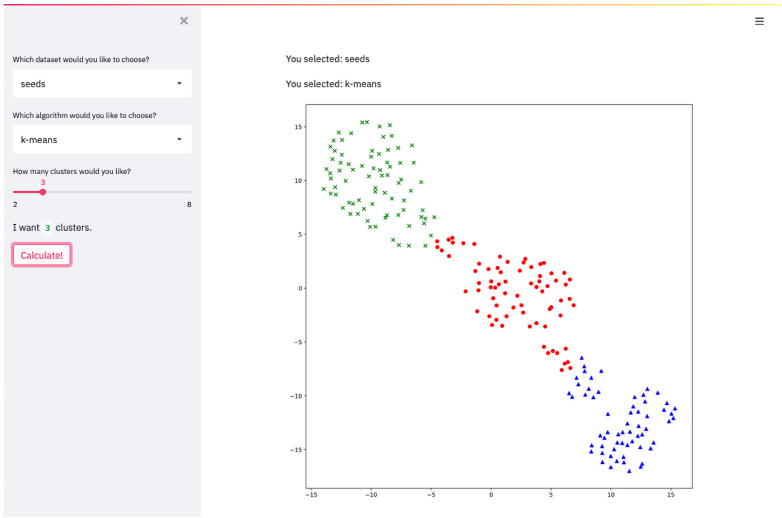
6 Web Frontend and User Manual

- Describe the implementation and write a brief user manual with screenshots.

The frontend has been developed in Streamlit [37] in its latest version as of June 9, 2021, 0.82.0. Things are moving fast but not yet finished (working beta can be accessed at `https://clustering.goethe.tech`).

Some screenshots like image 4 with explanation etc.

Figure 4: Frontend example screenshot



Which dataset would you like to choose?

seeds

Which algorithm would you like to choose?

k-means

How many clusters would you like?

3

I want 3 clusters.

You selected: seeds

You selected: k-means

7 Conclusion

- Summarize the main points and achievements
- Add your own assessment/criticism on the topic

Acronyms

CH	Calinski-Harabasz Index.
DB	Davies-Bouldin Index.
DI	Dunn Index.
GS	Gap Statistic.
sklearn	Scikit-Learn.
SS	Silhouette Score.

References

[1] T. Soni Madhulatha. An overview on clustering algorithms, 2012.

[2] L Rokach and O. Maimon. Clustering methods. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, chapter 15, pages 321–352. Springer, Boston, MA, 2005.

[3] Michael Wiedenbeck and Cornelia Züll. Klassifikation mit clusteranalyse: Grundlegende techniken hierarchischer und k-means-verfahren. 2001.

[4] Ludwig Fahrmeir, Alfred Hamerle, and Gerhard Tutz. *Multivariate statistische verfahren*. Walter de Gruyter GmbH & Co KG, 2015.

[5] Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

[6] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[7] Sariel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

[8] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153, 2006.

[9] Geoffrey H Ball and David J Hall. A clustering technique for summarizing multivariate data. *Behavioral science*, 12(2):153–155, 1967.

[10] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[11] RC Jancey. Multidimensional group analysis. *Australian Journal of Botany*, 14(1):127–130, 1966.

[12] MJ Norušis. Ibm spss statistics 19 advanced statistical procedures, 2011.

[13] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[14] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[15] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[16] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[17] Shokri Z Selim and Mohamed A Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on pattern analysis and machine intelligence*, (1):81–87, 1984.

[18] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[19] Guillaume Cleuziou. An extended version of the k-means method for overlapping clustering. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[20] Duc Truong Pham, Stefan S Dimov, and Chi D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[21] Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J—Multidisciplinary Scientific Journal*, 2(2):226–235, 2019.

[22] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. In *Transactions on Information Theory*, pages 32–40. IEEE, 1975.

[23] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. In *Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619. IEEE, 2002.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[26] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[27] M. A. Carreira-Perpiñán. A review of mean-shift algorithms for clustering. volume abs/1503.00687, 2015.

[28] David Harrison and Daniel Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 03 1978.

[29] Małgorzata Charytanowicz, Jerzy Niewczas, Piotr Kulczycki, Piotr A Kowalski, Szymon Łukasik, and Sławomir Żak. Complete gradient clustering algorithm for features analysis of x-ray images. In *Information technologies in biomedicine*, pages 15–24. Springer, 2010.

[30] A Strumillo, J Niewczas, P Szczypinski, P Makowski, and W Wozniak. Computer system for analysis of x-ray images of wheat grains. *International agrophysics*, 13(1), 1999.

[31] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)*, pages 147–153, 2003.

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

[33] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[34] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[35] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[36] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[37] Adrian Treuille, Thiago Teixeira, and Amanda Kelly. Streamlit. `https://github.com/streamlit/streamlit`, 2018. e1301bf6bb8f3a847d20ed9f51ff10585016f780.