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 20, 2021

Chosen Project Topic:

- T3 Clustering

Github Repository:

- `https://github.com/BenLucht/DS1_term_paper`

Abstract

The objective of this term paper is to provide an overview of clustering by the example of four different clustering algorithms applied to four different data sets. The algorithms used are K-Means, Mean Shift, Affinity Propagation, and Spectral Clustering. These provide different characteristics concerning e.g. metrics, handling of data set sizes and evenness of cluster size as well as the parameters driving them. Data sets on house pricing, mall customers, wheat seeds and red wine show vastly different properties in relation to the number of observations and features, types of features to exhibit the differences in clustering results for the algorithms used. Visualization of results and comparisons can be accessed through an online frontend.

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 start-

ing 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

```
1: place initial centroids c_i (1)
2: while centroids not stationary (4) do
3:     if observations have labels then
4:         update centroids c_i to center of current clusters C_i (3)
5:     end if
6:     calculate all distances d_{n,k_i} (2)
7:     assign each observation to min distant centroid via label i
8: 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. From a visual standpoint there should be three clusters, one to the left containing four observations, one in the middle containing two observations and one to the right with five observations. Initial random placement has resulted in two initial centroids far to the left and one near the center of the feature space. Performing the steps as described results in a final clustering of one cluster containing a single observation to the left, another comprising three observations also near the left and one big cluster with seven observations from two visually separated segments of the data. This is a valid result meeting stopping criteria but it is not meeting the expectations. Different starting positions can lead to different clustering results.

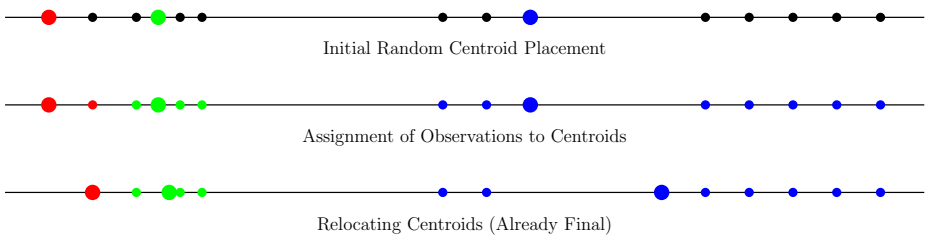


Figure 1: One Dimension With Ten Observations - Bad Starting Positions

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 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 $h > 0$ 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 large, 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 $h > 0$, 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 $\nabla \hat{f}(x) \stackrel{!}{=} 0$. 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. Thus, 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:

Algorithm 2 Mean Shift Clustering

for i=1, ..., 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 $\{C\}_{1...m}$

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 $O(T \cdot n^2)$, 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. Thus, the algorithm can 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 of the bandwidth parameter value. Therefore, Mean Shift Clustering is better applicable to low-dimensional datasets as well as datasets consisting of dense regions [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 with respect to 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 lower educational status. In addition to that, crime rate is included for neighborhoods of 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 binary variable encoded as 0 and 1, the remaining attributes are float numbers. The dataset does not contain any empty columns, thus no elimination or preprocessing of the

available data is necessary.
Table 1 provides some general statistics on the given dataset. For each of the columns, except the measurement of closeness to Charles River, mean, minimum and maximum value as well as the standard deviation are given. In addition to that, lower (0.25) and upper (0.75) quartiles and the 50%-quantiles are listed for each of the thirteen features. Quantiles yield information on how the data points are distributed within the high-dimensional feature space. It can be observed that for most of the features, data points lie relatively close together, proposing the dataset to consist of dense regions.

Column	Mean	Min	Max	Std	25%	50%	75%
CRIM	3.614	0.006	88.976	8.593	0.082	0.257	3.677
ZN	11.364	0.000	100.000	23.299	0.000	0.000	12.5
INDUS	11.137	0.460	27.740	6.854	5.190	9.960	18.100
NOX	0.555	0.385	0.871	0.116	0.449	0.538	0.624
RM	6.285	3.561	8.780	0.702	5.886	6.209	6.624
AGE	68.575	2.900	100.000	28.121	45.025	77.500	94.075
DIS	3.795	1.130	12.127	2.104	2.100	3.207	5.188
RAD	9.549	1.000	24.000	8.699	4.000	5.000	24.000
TAX	408.237	187.000	711.000	168.371	279.000	330.000	666.000
PTRATIO	18.456	12.600	22.000	2.163	17.400	19.050	20.200
B	356.674	0.320	396.9	91.205	375.378	391.44	396.225
LSTAT	12.653	1.730	37.970	7.134	6.950	11.360	16.955
MEDV	22.533	5.000	50.000	9.188	17.025	21.200	25.000

Table 1: General Statistics on Boston House Pricing Dataset

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. 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

A brief look into the data set in table 2 confirms this.

	Area	Perimeter	Compactness	Length	Width	Asymmetry	Groove	Label
1	15.26	14.84	0.871	5.763	3.312	2.221	5.220	1
2	14.88	14.57	0.881	5.554	3.333	1.018	4.956	1
3	14.29	14.09	0.905	5.291	3.337	2.699	4.825	1
4	13.84	13.94	0.896	5.324	3.379	2.259	4.805	1
5	16.14	14.99	0.903	5.658	3.562	1.355	5.175	1

Table 2: Seeds Data Set First Observations

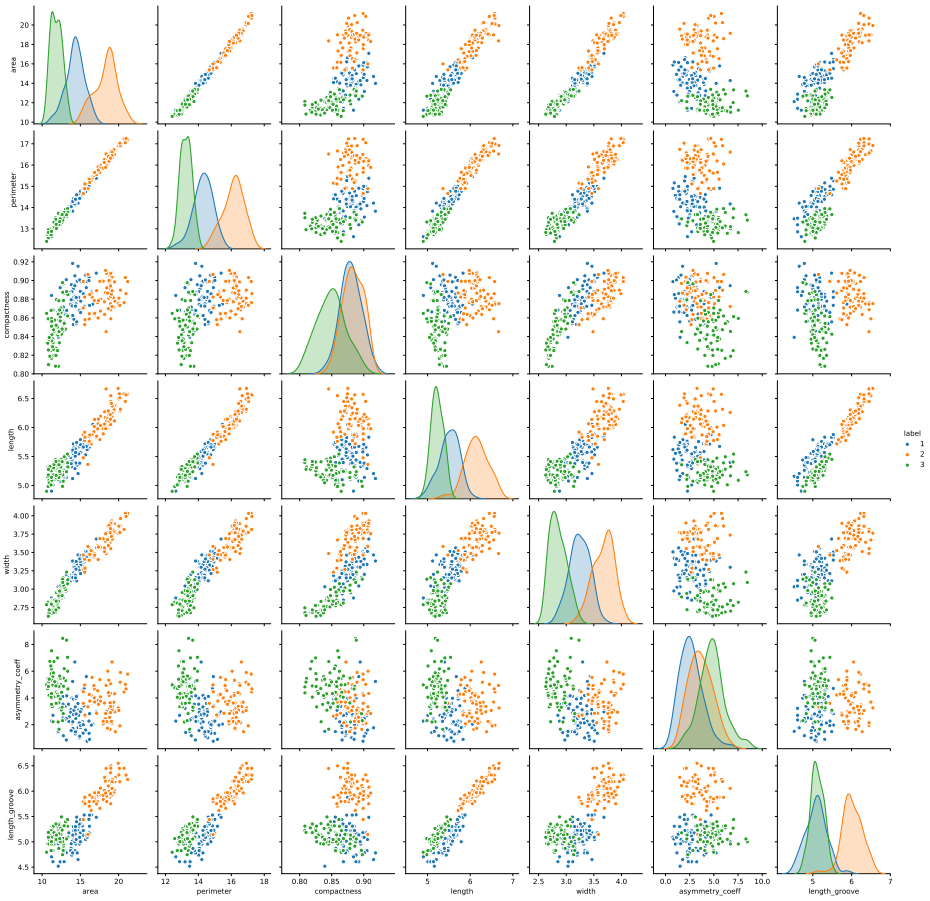


Figure 3: Seeds Pairplot

A few things can be noted: 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 correlated with width and length and the perimeter. These relationships among others can be seen in figure 3.3. Note that these plots have been colored according to their original labels. Density plots on the diagonal for example already show distinct characteristics for the different varieties of kernels for certain features.

Some key insights on the data set as a whole can be found in table 3 with information on the observation count, mean, standard deviation (std), minimum and maximum as well as 25, 50 and 75% quantiles as rows for each

feature in columns. There are a total of 210 observations which is on the lower end for clustering applications. The data is balanced at 70 observations for each kernel variety. There is no missing data in any of the features.

	Area	Perimeter	Compactness	Length	Width	Asymmetry	Groove
Count	210	210	210	210	210	210	210
Mean	14.848	14.559	0.871	5.629	3.259	3.700	5.408
Std	2.910	1.306	0.024	0.443	0.378	1.504	0.491
Min	10.590	12.410	0.808	4.899	2.630	0.765	4.519
25%	12.270	13.450	0.857	5.262	2.944	2.561	5.045
50%	14.355	14.320	0.873	5.524	3.237	3.599	5.223
75%	17.305	15.715	0.888	5.980	3.562	4.769	5.877
Max	21.180	17.250	0.918	6.675	4.033	8.456	6.550

Table 3: Seeds Data Set Statistics Summary

A few more things can be noted here: Rosa wheat kernels exhibit a mean area more than one standard deviation above the mean area for the entire data. Similarly, Canadian wheat kernels have a mean asymmetry coefficient far beyond one standard deviation above the overall mean. For each variety of wheat a normality test [31] has shown that the hypothesis of normality cannot be rejected at a 99% level for most features. This might bode well for at least K-Means clustering.

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 in part 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

written by B.L.

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 20, 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 [32] 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. Implementing a custom initialization function or providing a list of starting points is possible as well.

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

written by L.B.

For Mean Shift Clustering we use the implementation provided from [26] in version 0.24.2 which is the latest stable release as of June 20, 2021. The implementation offers the possibility to enter several additional parameters supplementary to the required bandwidth. Three related additional parameters can be used to speed up the clustering procedure as mentioned in section 2.3. These parameters are called `seeds`, `bin_seeding` and `min_bin_freq` and allow to define which of the given data points should be used for the initialization of candidate centroids. Exclusively if no array of initial seeds is passed over, the algorithm checks the `bin_seeding` parameter value. If

this parameter is `False`, all given data points are used to initialize kernels. In contrast to that, if the `bin_seeding` parameter is `True` , the algorithm creates a grid with cell size defined by the bandwidth parameter. The number of points within the created bins defines which data points are used as seeds. Therefore, `min_bin_freq` defines the minimum number of data points that have to occur in a bin to define a candidate centroid. Running the algorithm in its default mode, as it is done in this work, all the given data points are initialized as candidate centroids and hence used as initial kernel locations. In addition to that, the algorithm provides a parameter called `cluster_all`, which defines whether all the given data points have to be assigned to a cluster. If the parameter is `True`, all data points are clustered, even if they are not placed within any kernel. Those outliers are assigned to their nearest kernel. If the parameter is `False`, outliers outside any kernel's radius are not assigned to any cluster, but instead their label is `-1`. A technical parameter is called `n_jobs`. Users can pass an integer to determine the number of jobs which should be used for the computation.
In theory, the algorithm stops the mean shift iteration for each seed if it converges. The `max_iter` parameter can be used to fix a maximum number of iterations if a seed does not converge. In the default mode, the maximum number of iterations is set to 300.
If no value for the required bandwidth parameter is provided, the bandwidth is estimated using a built-in `estimate_bandwidth` function. This function calculates the median of all pairwise distances and ascertains a suitable bandwidth based on the underlying dataset.
The algorithm performs the mean shift procedure for all the seeds in parallel. In each iteration, all the neighbors within the kernel's radius are found by executing a nearest neighbor search. The seed is shifted to the mean of all the neighboring data points until it either converges or the maximum number of iterations is reached. After shifting seeds, some postprocessing steps are performed. The algorithm searches for seeds whose distance is smaller than the bandwidth. If duplicates like these are found, the one with fewer data points is removed. If `cluster_all` is `True`, each of the given data points is assigned to the cluster defined by the seed that is closest to the respective data point. If `cluster_all` is `False`, only those data points that are located within a kernel's bandwidth are assigned to the cluster defined by the nearest seed and the remaining data points are assigned to the artificial cluster with label `-1`.

4.4 Scikit-Learn Spectral Clustering

5 Description of Evaluation Module

Due to the nature of clustering as an unsupervised machine learning method, it is typically applied to datasets lacking prior information on grouping. Therefore measuring the performance of each method can only be performed on the data and the labeling produced by each clustering algorithm. These types of calculations are commonly called *internal cluster validation* indices.

Differentiating by the way compactness and separation are measured [33] split indices into two groups. 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, Pakhira-Bandyopadhyay-Maulik (PBM), Ratkowski-Lance, Ray-Turi, Xie-Beni, Wemmert-Gancarski.

We have decided to use two indices from each group: Dunn Index (DI) and Silhouette Score (SI), Calinski-Harabasz Index (CH), Davies-Bouldin Index (DB).

5.1 Dunn Index

written by M.A.

The Dunn Index (in short DI) is a metric for evaluating clustering algorithms. Its aim is to identify sets of clusters [34]. It is used for measuring an internal validation of clustering results [35].

For each cluster, compute the distance between each of the objects in the cluster and the objects in the other clusters. Use the minimum of this pairwise distance as the inter-cluster separation (min. separation). For each cluster, compute the distance between the objects in the same cluster. Use the maximal intra-cluster distance (i.e maximum diameter) as the intra-cluster compactness. Calculate the Dunn index (D) as follow:

$\mathrm{D} = \frac{min.separation}{max.diameter}$ [35]

The higher the value of the resulting Dunn index, the better the clustering result is considered, since higher values indicate that clusters are Compact.

That means the greater the better [36].

5.2 Calinski-Harabasz Index

written by B.L.

The Calinski-Harabasz criterion [37] is defined as a poportional measure of within cluster variance to between cluster variance. That is why it is also known as the Variance Ratio Criterion (VRC). Mathematically this means

$$CH = \frac{S_B}{S_W}\frac{m-K}{K-1} = \frac{\sum_{k=1}^{K}|C_k|\left\|c_k - \sum_{x'\in X} x'\right\|^2}{\sum_{k=1}^{K}\sum_{x\in C_k}\|x-c_i\|^2}\frac{m-K}{K-1} \tag{8}$$

utilizing the same notation as equation 3. Between-cluster variance is divided by within-cluster variance. Dividing the data into more partitions of course reduces within-cluster variance, which is why there is also a scaling factor (second fraction representing degrees of freedom) to account for it. In comparison to DB between-cluster variance is calculated in relation to the center of the data set in place of between individual cluster centers themselves.

Due to its construction (distance measure) this index performs well when clusters form mostly convex shapes. This can lead to mis-evaluation if the underlying data does not comprise of such globular clusters so a method capturing this might score lower while actually fitting the data better. It may therefore be better suited to compare different parameters for the same algorithm instead of comparing different algorithms. A comparative study of 30 cluster validation indices [38] ranks CH performance in evaluating fit near the top in many applications, particularly when evaluating K-Means results as expected.

5.3 Davies-Bouldin Index

written by J.M.

The Davies-Bouldin index is a metric to evaluate clustering algorithms and it was developed by David Davies and Donald Bouldin in 1979 [39]. It is used to check the validity of the clusters. To reach this, the Davies-Bouldin Index tries to maximize the intern-cluster distance and on the other side tries

to minimize the intern cluster distance. The Davies-Bouldin index is defined as

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

where N is the number of cluster and R_i is the maximum of R_{ij} where i≠j

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

where M_{ij} is the Minkowski metric which can be descibed as:

$M_{ij} = \{\sum_{k=1}^{N} |a_{ki} - a_{kj}|^p\}^{\frac{1}{p}}$

$S_{ij} = \{\frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^q\}^{\frac{1}{q}}$

If q =1 S_{ij} is the Euclidean distance between centroids.

For our evaluation we use the implementation from sklearn of this metric. This metrics calculate a score. Zero is the lowest score that can be reached. If a score is closer to zero, it related that there is a better seperation between the calculated clusters.

5.4 Silhouette Score

written by L.B.

Cluster validation indexes are used to assess clustering quality. That means it is investigated whether the resulting clusters reflect natural structures in the data or whether clusters rather determine artificial groups. In general, the ratio of dissimilarity between data points within the same cluster to dissimilarity between data points in different clusters is a measurement for clustering quality. The Silhouette Score is one of such cluster validation indexes which can be applied to evaluate clustering results if Euclidean distance measures can be calculated on the underlying dataset. According to that, a clustering result as well as proximity scores between all data points have to be provided to be able to apply the Silhouette Score. For a single data point, distances to all the data points assigned to the same cluster are calculated. Here, the mean of all these distances is called *a*. After that, the nearest cluster for each data point has to be found. This can be done by iteratively calculating the mean of distances between the single data point and all data points within another cluster until distances for all clusters have been computed. The minimum average distance determines the nearest cluster for

the single data point. The minimum average distance is called b here. The Silhouette Score for data point i is calculated using the following formula:

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

The mean of all the Silhouette Scores for each individual data point determines the Silhouette Score for the whole dataset. As the nearest cluster has to be found for each of the included data points, the Silhouette Score can only be computed as soon as more than one cluster exists [40]. Taking a closer look at extreme cluster results determines how to interpret the Silhouette Score values. Given a partition where the mean intra-cluster distance a is extremely low while the mean inter-cluster distance b is high, the Silhouette Score would tend to be `1`. Hence, a Silhouette Score of `1` corresponds to a well-partitioned dataset because the mean of distances between single points and other clusters than the one it is assigned to is large. Analogously, a value of `-1` corresponds to a partition where the inter-cluster distance b is much larger than the intra-cluster distance a. Thus, the data points are not assigned to their nearest clusters, resulting in a misclassification. If the Silhouette Score is equal to zero, means of inter- and intra-cluster distances are close to each other and it is unclear which cluster the point should be assigned to. In this case, the partition can also contain overlapping clusters [40]. For the calculations of Silhouette Scores, we utilize the implementation provided from sklearn [26]. We make use of the Silhouette Score as it can be applied on cluster results generated with any clustering technique. It therefore provides an intuitive way of comparing the results of different algorithms to each other. One disadvantage is that Silhouette Scores are higher for convex clusters. As some density-based clustering algorithms, like Mean Shift or Spectral Clustering, explicitly allow the creation of non-convex clusters, the obtained Silhouette Scores could possibly be lower than results created applying other clustering techniques.

5.5 Results

In the following there is a discussion of the results of all clustering efforts in detail. A brief overview of the evaluation results can be found in table 4. For each data set there is a row for each algorithm used. In columns are the results for each index. The cells represent the highest (lowest for DB respectively) value achieved by applying various parameter configurations.

It should be noted that comparing results across different methods does not show the entire picture but it can yield indications on performance. MORE WHEN RESULTS.

Data and Method	CH	DB	SI	DI
Housing				
K-Means	1621.90	0.326	0.721	0.525
Mean Shift	1338.65	0.326	0.721	0.111
Affinity	1803.31	0.350	0.372	0.185
Spectral	1456.86	0.546	0.689	0.525
Mall				
K-Means	301.02	0.766	0.479	0.843
Mean Shift	921.29	0.103	0.438	0.905
Affinity	314.89	0.282	0.371	0.382
Spectral	262.82	0.731	0.454	0.087
Seeds				
K-Means	369.55	0.691	0.492	0.091
Mean Shift	374.19	0.689	0.519	0.113
Affinity	374.12	0.642	0.529	0.086
Spectral	335.58	0.638	0.525	0.107
Wine				
K-Means	3116.82	0.617	0.603	0.049
Mean Shift	1346.44	0.366	0.768	0.079
Affinity	2633.03	0.927	0.261	0.041
Spectral	2616.13	0.668	0.534	0.007

Table 4: Comparing All Algorithms On All Data Sets For All Evaluation Indices

5.5.1 Seeds - K-Means

written by B.L.

K	CH	DB	SI	DI
2	351.235	0.688	0.519	0.038
3	375.804	0.753	0.471	0.085
4	327.439	0.825	0.412	0.020
5	310.331	0.915	0.361	0.091
6	302.344	0.917	0.366	0.106
7	294.385	0.939	0.350	0.077
8	297.502	0.937	0.362	0.083

Table 5: K-Means Seeds Data Set Indices by Number of Clusters

Table 5 contains the index scores for each of the indices selected as columns and the number of clusters as rows. An easier representation of this data is a line chart which can be seen in figure 5.5.1 and in the following description of results. More than simply a tool to compare scores between methods, these can help in identifying the number of clusters (or other parameters below) best fit to represent the data. So what can be learned from the data: If there is a clear maximum (minimum for DB) value for an index it indicates looking at that value to gain insight on the fit. If there is no clear maximum (minimum) value, one can try to employ the so called elbow criterion (see section 2.1) where a distinct change in direction for a graph might indicate a value worth examining. If there is neither there seems to be no information gain at hand from that particular index for this application.

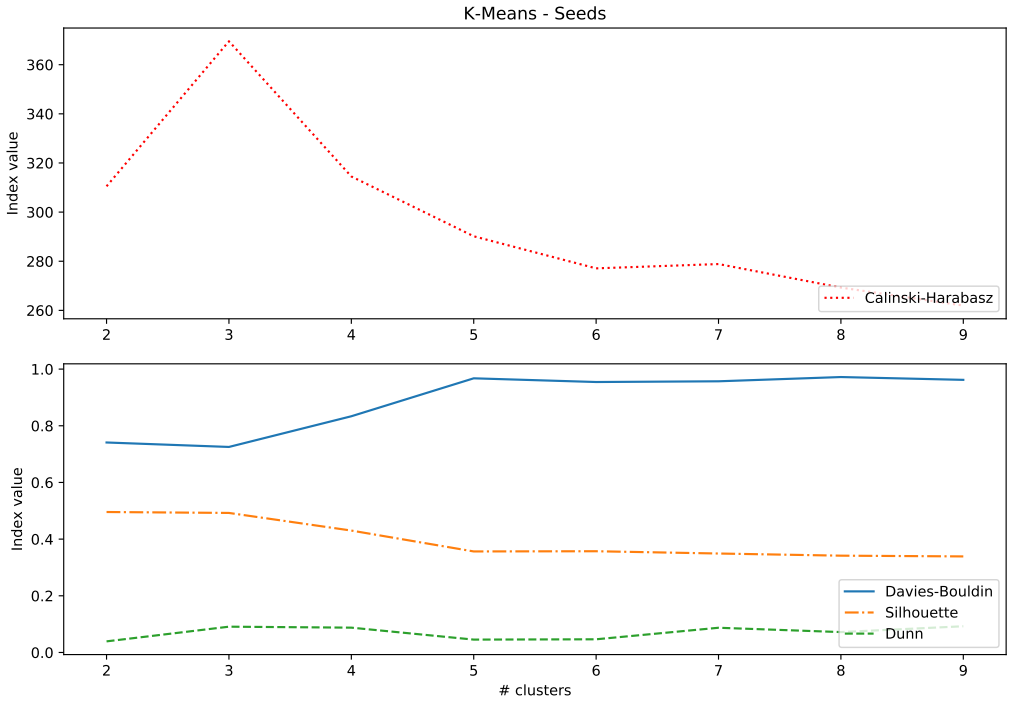


Figure 4: K-Means Seeds Data Set Indices by Number of Clusters

Looking at figure 5.5.1 (one line for each index, two panels because of the vastly different scale of CH) one can see a clear maximum in CH value and a local maximum in DI as well as a change in slope for SI at three clusters each. Following the described logic, DB also suggests considering three clusters.

As mentioned in section 3 this data set originally comes with labels which allows to use them as a benchmark to see how the clustering performs. Plotting (figure 5.5.1) two dimensional representations (see t-SNE in section 6) of the K-Means clustered result next to the original data shows a few things: the algorithm has, as expected, produced non-overlapping mostly contiguously shaped clusters in contrast to the original data. However using three clusters has generally produced a reasonably good fit, as is supported by an accuracy of 89.5%, precision of 90% and recall of 89.5% to name a few common metrics.

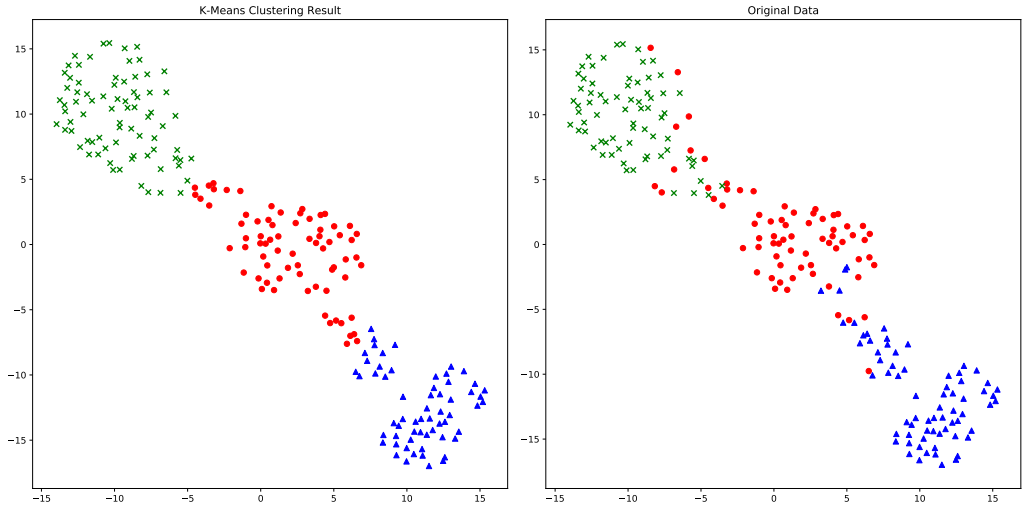


Figure 5: Seeds Data Set Comparison Clustering and Original Labels

5.5.2 Seeds - Mean Shift

written by L.B.

In the following two plots, values for the chosen cluster validation indexes that evaluate the clustering result generated applying the Mean Shift algorithm on the Seeds dataset for a range of bandwidth parameters are provided (Figure 5.5.2). To determine suitable bandwidth parameter values, ideally SI, DI and CH should be maximized while DB should be at a minimum. Reaching this goal is not that easy due to the way those scores are computed.

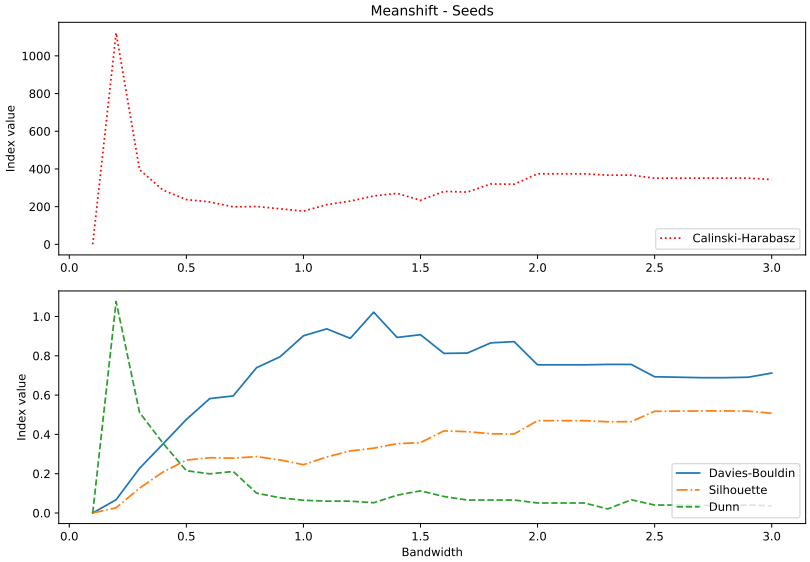


Figure 6: Cluster Validation Indexes: Means Shift on Seeds

Taking a look at DI, CH and DB, one might choose a bandwidth of 0.2 which indeed results in a very high number of small clusters and does not seem to represent a natural structure within the dataset as can be seen in Figure 7.

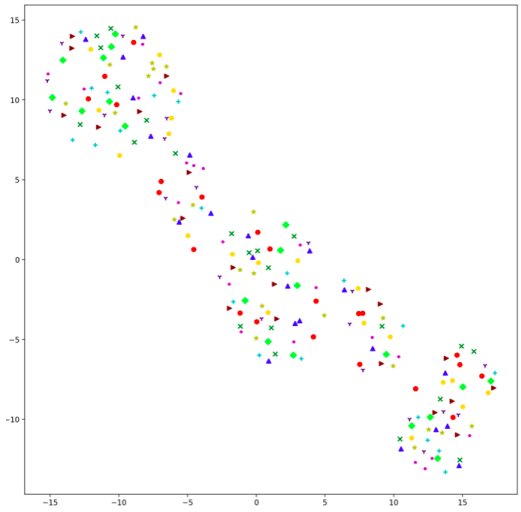


Figure 7: t-SNE projected result of Mean Shift with bandwidth = 0.2 on Seeds dataset

Instead, a balance between high values for DI, SI, CH and a low value for DB is found for bandwidth values of 2 or 2.5 which results in two respectively three distinct clusters (Figure 8).

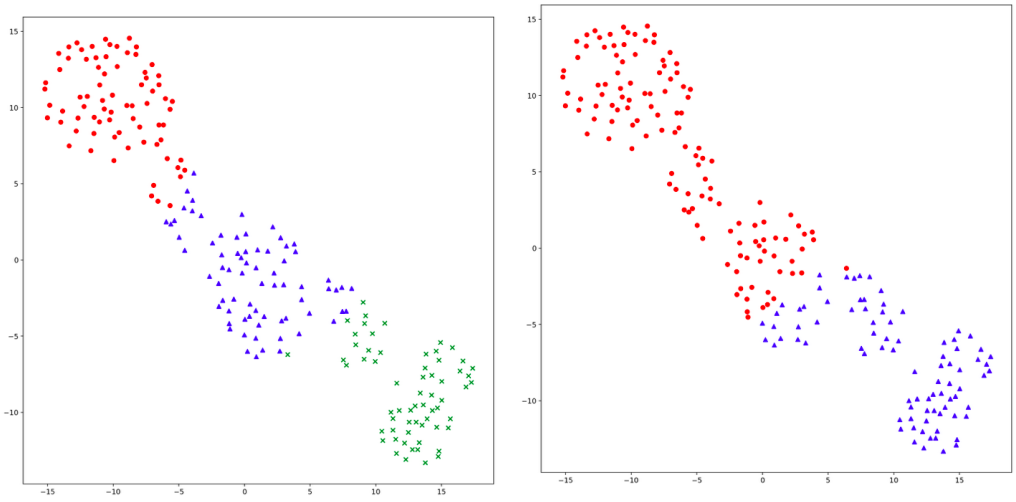


Figure 8: t-SNE projected result of Mean Shift with bandwidth = 2 (left) and bandwidth = 2.5 (right)w on Seeds dataset

As the dataset is labeled (see section 5.5.1), a comparison with the original labels would probably support choosing a bandwidth parameter value of 2.

5.5.3 Seeds - Affinity Propagation

written by J.M.

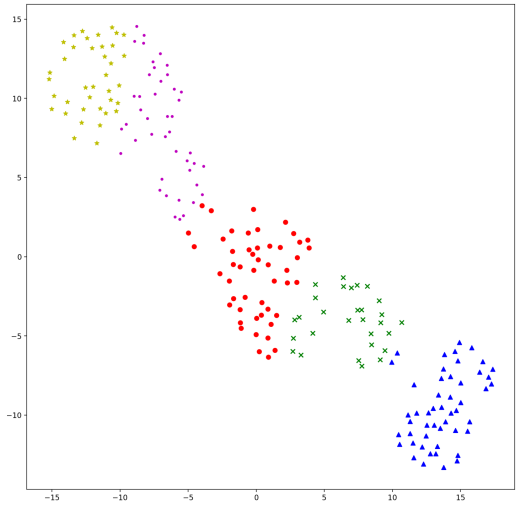


Figure 9: t-SNE plot of Affinity Propagation on Seeds dataset

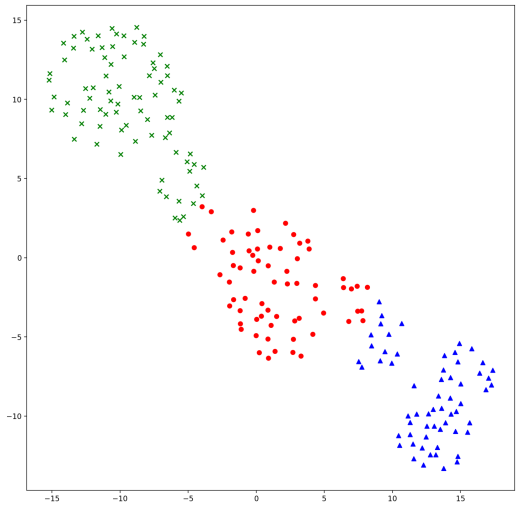


Figure 10: t-SNE plot of Affinity Propagation on Seeds dataset

On the seeds Dataset it performs good. And it finds some good cluster. Depending on the parameter more or less cluster are created which are shown 5.5.3 and 5.5.3. In 5.5.3 that input parameter is -78. It's a very low compared to the parameter that are used in the other datasets. If this parameter gets lower, affinity propagation will create less cluster like in5.5.3. In this figure the parameter is -247. If you compare the cluster validation metrics on these two plots, you can see that the second plot is better in all cluster validation metrics. If we look to other parameters, we don't find new insights, the parameter -247 is quite optimal for this dataset.

custer validation	plot1	plot2
Calinski-Harabasz	302.7425	374.6137
Davies-Bouldin	0.8784	0.7544
Silhouette	0.3644	0.4681
Dunn	0.0489	0.0495

Table 6: Comparison Cluster validation Affinity Propagation

5.5.4 Mall Customers - K-Means

written by B.L.

As one of the features is categorical (gender), this is not an ideal application of the K-Means clustering method. The categories can be resolved into discrete values so that calculations can be run, but the arbitrary nature of this does not really allow for good interpretation of the results. An exemplary three dimensional plot 11 shows the issue:

gender is represented on the z-axis and the two manifestations simply rip the data into two partitions. When gender is encoded as 0 and 1, this showed almost no effect. When using 0 and 100, it leads to a doubling in the index-recommended number of clusters. One attempt to deal with this is to scale all data between 0 and 1 but this aggravated the problem instead of improving on it. Extensions of the K-Means method have been devised to deal with this type of data (as described in section 2.1) but their application here seems beyond the scope of this work. Using the gender feature also does not improve scores in the evaluation methods used. Given these circumstances, considering plot 5.5.4 suggests considering the results for 2 (CH, DB, SI), 7 (DB) or 8 (CH) clusters.

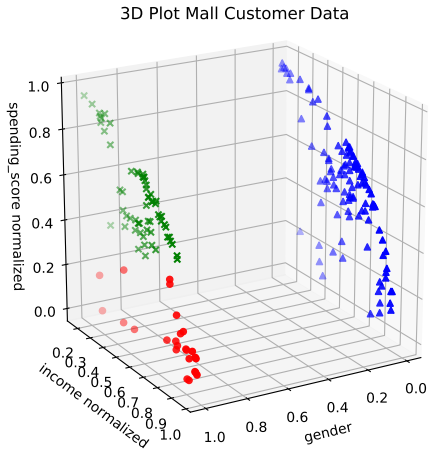


Figure 11: Split Data Due To Categorical Feature

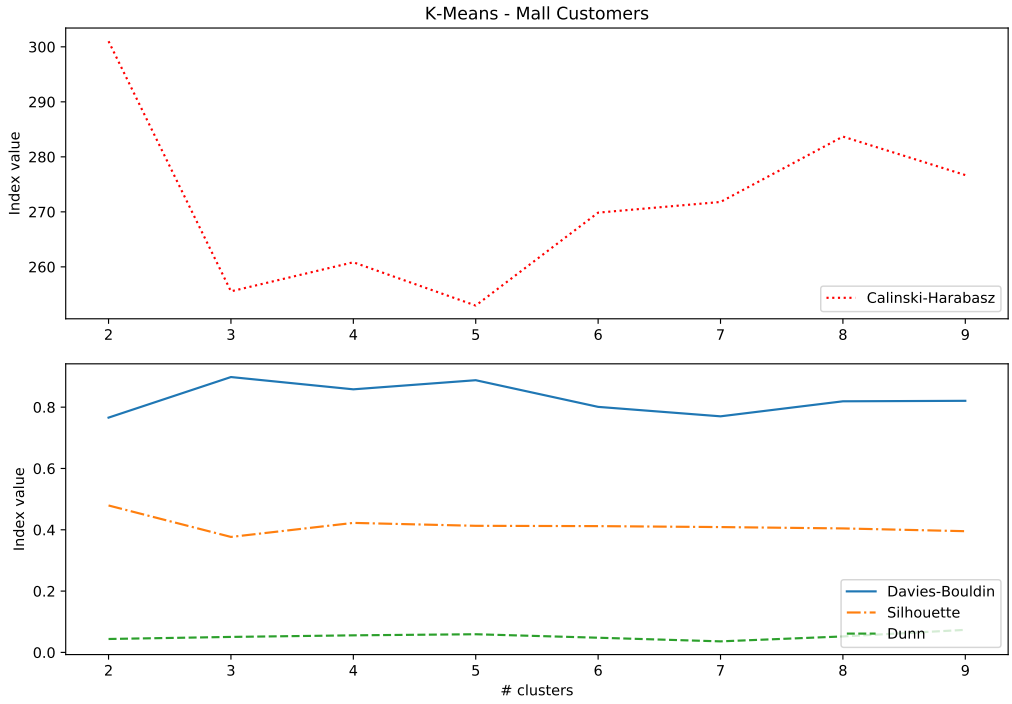


Figure 12: K-Means Mall Customers Data Set Indices by Number of Clusters

The data set contains four features, three of which have a measurable and interpretable impact on clustering, which lends itself to a visual comparison in three dimensional space. Figure 5.5.4 shows three plots, one for each suggested number of clusters. The highest overall evaluation scores were achieved for two clusters and purely from a visual standpoint this seems to hold up.

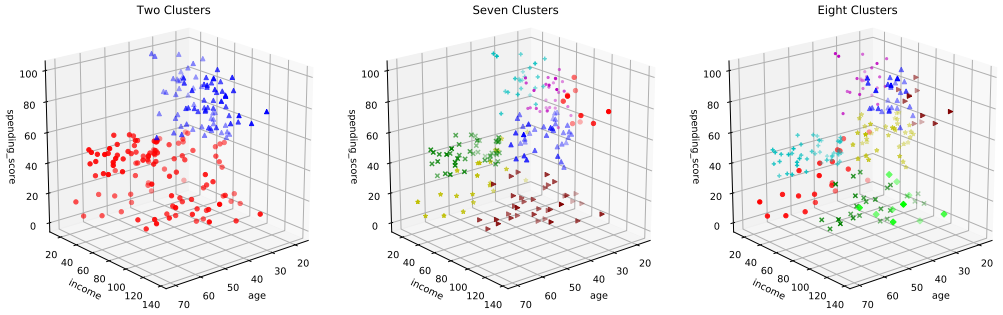


Figure 13: Mall Customers 3D Plot Clustering Options

5.5.5 Mall Customers - Mean Shift

written by L.B.

The influence of different bandwidth parameter values for the application of Mean Shift on the Mall dataset can be seen in Figure 5.5.5.

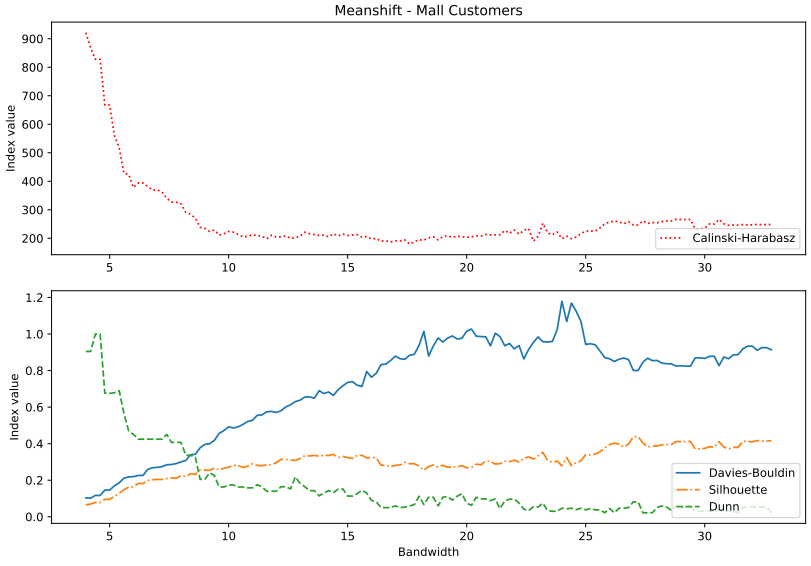


Figure 14: Cluster Validation Indexes: Mean Shift on Mall Customers

Again, as for the Seeds dataset, ignoring SI and only trying to maximize DI and CH while minimizing DB, one might choose a bandwidth of 4.6. But again, this rather introduces artifacts than reflecting any natural structures (Figure 5.5.5).

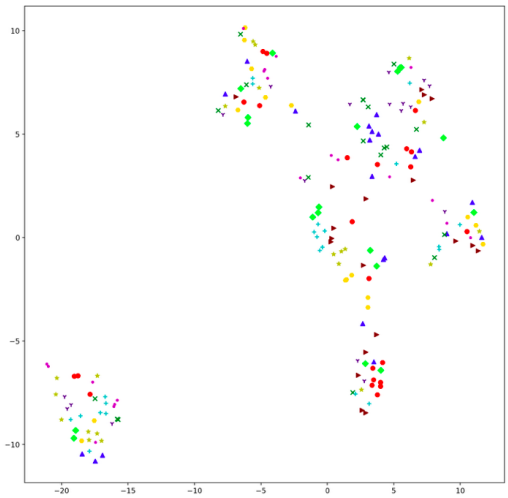


Figure 15: t-SNE projected result of Mean Shift with bandwidth = 0.2 on Mall Customers dataset

A more suitable clustering result is generated using a bandwidth of 27.5 (Figure 5.5.5). In the plot, one can see that this value compensates between all four indexes. When taking a look at the clustering result, there still are some data points which do not seem to be clustered correctly. One possible explanation concerns the gender column which is encoded as 0 and 1 (see section 5.5.4). It is not possible to calculate a meaningful distance between binary variables which may have an influence on the clustering result.

5.5.6 Mall Customers - Affinity Propagation

written by J.M.

On the customers Dataset it also performs good, but for this dataset the parameter is smaller as for the seeds dataset. In 5.5.6 it computed with -2870. Here you see that the Clustering is not good, there exists many clusters. To

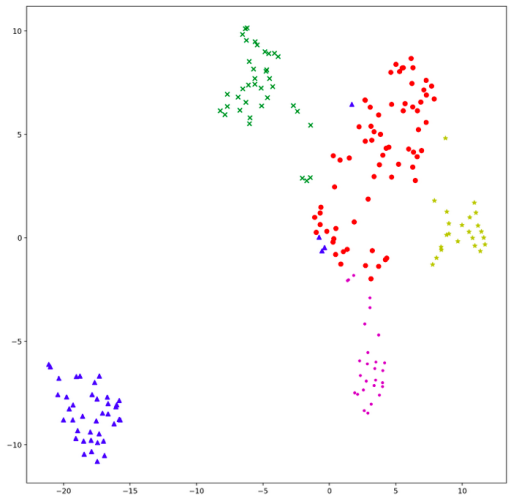


Figure 16: t-SNE projected result of Mean Shift with bandwidth = 27.5 on Mall Customers dataset

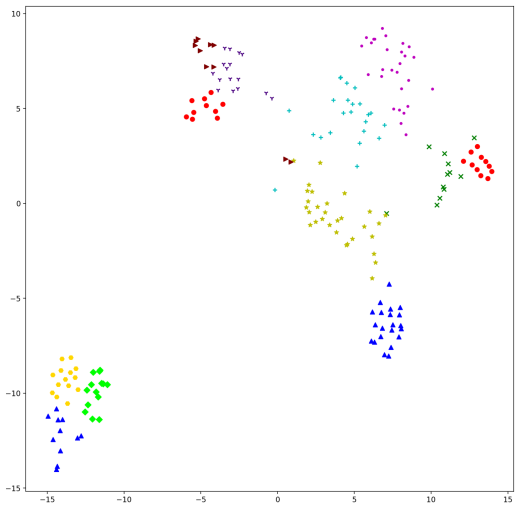


Figure 17: t-SNE plot of Affinity Propagation on Mall Customers dataset

check if this algorithm works with this dataset the parameter is change to a parameter where to clusters look good. This happened when the parameter is around 8778. This plot is in 5.5.6. If you compare these two plots with the

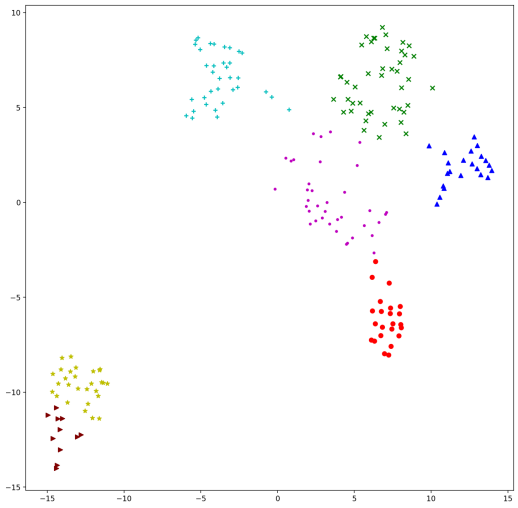


Figure 18: t-SNE plot of Affinity Propagation on Mall Customers dataset

cluster validation indexes, it confirms the obvious the 5.5.6 is better. This parameter is also near to the optimum for this dataset.

5.5.7 House Pricing - K-Means

written by B.L.

This data set is challenging for K-Means clustering due to the high number of features. As described in section 2.1 the concept of distance does not hold up well in high dimensional cases. Attempts to transform the data into lower dimensional representations through principle component analysis did not yield an improvement in clustering performance. Data in figure 5.5.7 suggests to consider 2 (DI), 3 (SI, DB) or 4 (CH) clusters.

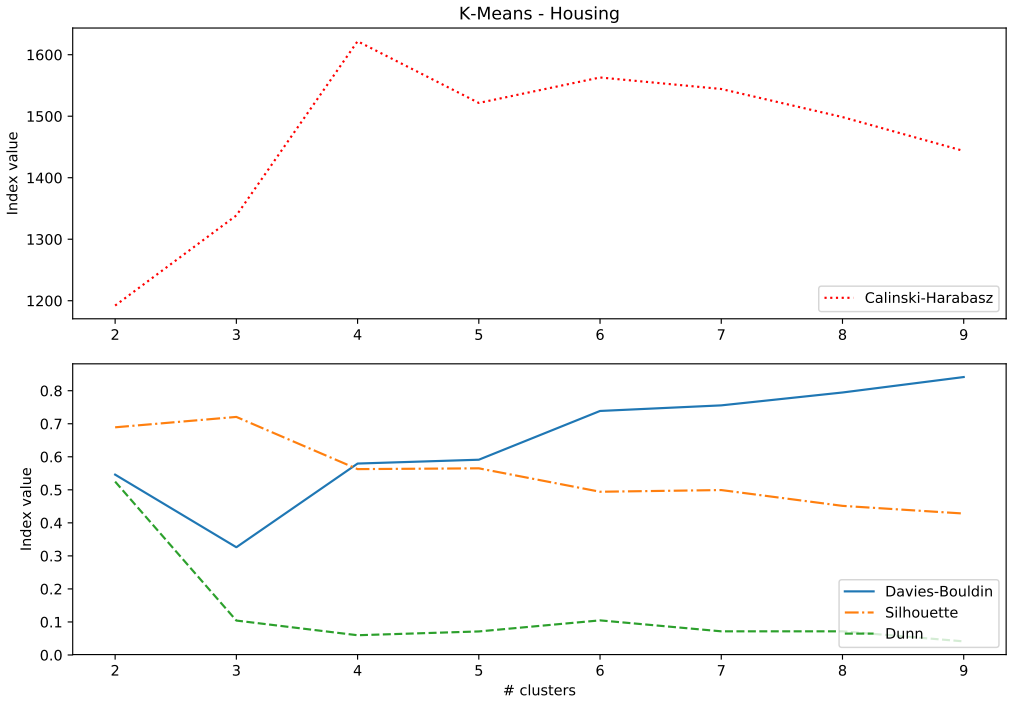


Figure 19: K-Means Housing Data Set Indices by Number of Clusters

Due to the high dimensionality of the data, looking at different angles in three dimensions to get an understanding is not feasible. Therefore a two dimensional transformation (t-SNE, see section 6) will help. Figure 5.5.7 shows the case of two, three and 15 as representation for a significantly higher number of clusters. From a visual standpoint, two and three clusters seem to capture the visibly separation of parts of the data quite well, while the high cluster count suggests it might be reasonable to consider the idea of there being a much more complex underlying structure to the data which is not well captured through the method at hand.

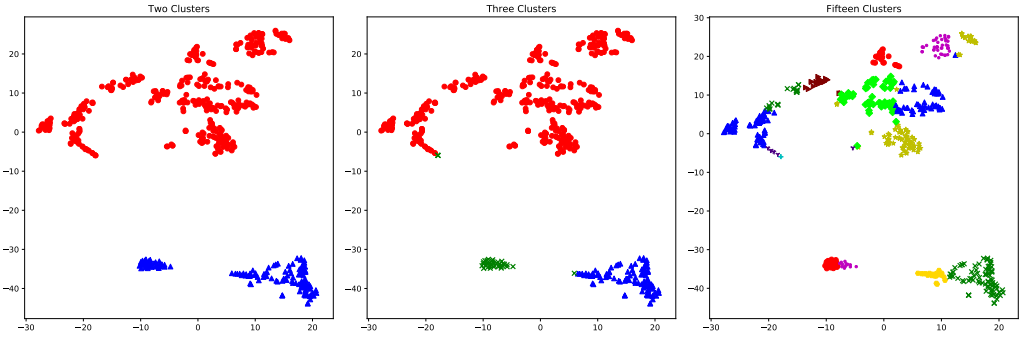


Figure 20: K-Means Housing Data Set t-SNE Transformed Data Plot

5.5.8 House Pricing - Mean Shift

written by L.B.

The third dataset the Mean Shift algorithm was applied to is the Boston House Pricing dataset. In contrast to the Seeds and the Mall Customers dataset, it contains fourteen columns and hence requires the handling of fourteen dimensions. As mentioned in section 2.3, runtime results in a complexity of $O(n^3)$ in high dimensions. Nevertheless, Mean Shift seems to be able to handle the dataset and produces reasonable clustering results.

In Figure 5.5.8 one can clearly see a maximum of DI, SI and CH and at the same time a minimum for DB in the range of 118 to 156 or between 159 and 185. All of the four chosen cluster validation indexes do not change for either of the values within the mentioned ranges. This is due to the large distance between data points of the red cluster and the remaining datapoints as it can be seen in Figure 5.5.8.

The bandwidth value determines the kernel's radius and thus influences which data points are considered for which cluster. If data points are far away from each other, as it is the case for the Boston House Pricing dataset, increasing the bandwidth parameter will not include any more data points within the kernel's radius and thus not have an influence on the cluster generation.

Furthermore, the dataset consists of three relatively dense regions far away from each other, which is a desirable situation when applying density-based clustering algorithms. Due to this, Mean Shift algorithm performs well on

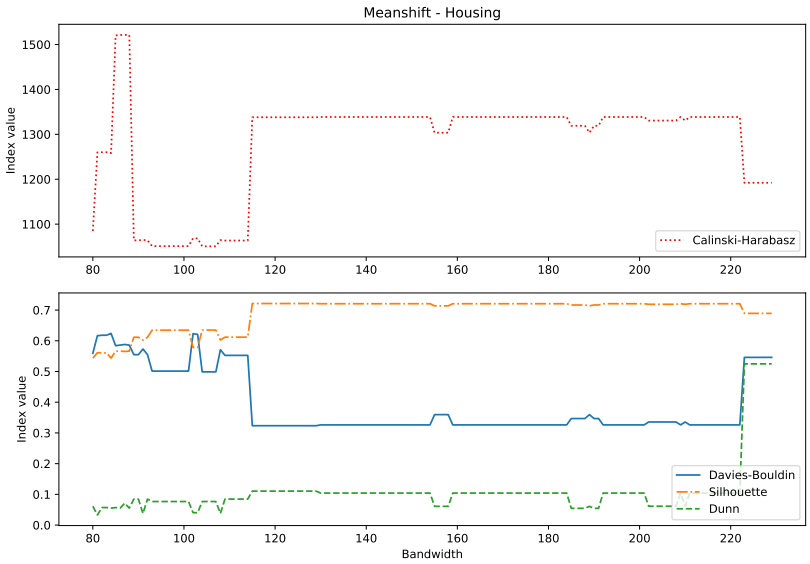


Figure 21: Cluster validation indexes: Mean Shift on House Pricing dataset

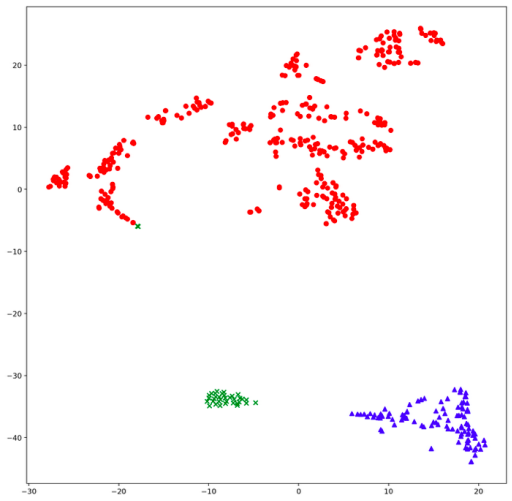


Figure 22: t-SNE projected result of Mean Shift with bandwidth = 120 on House Pricing dataset

the dataset, even if it is high-dimensional.

5.5.9 House Pricing - Affinity Propagation

written by J.M.

On the house - pricing dataset affinity propagation performs not good. It takes some time to find a good parameter for this dataset. This is shown in 5.5.9. The parameter for this plot wars -71433. This parameter is so low because this dataset has the most dimensions. If you use a higher parameter the cluster is not so good. This is shown in 5.5.9.

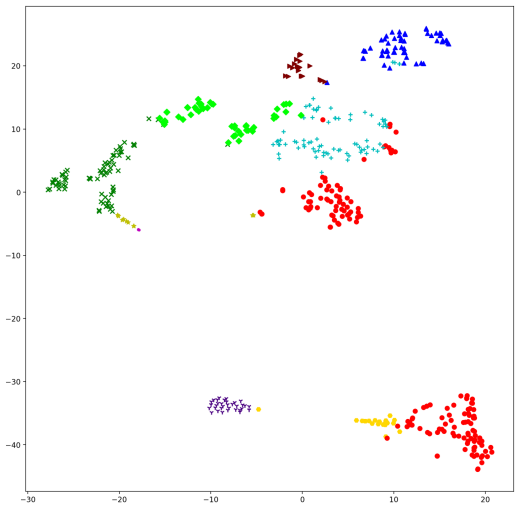


Figure 23: t-SNE plot of Affinity Propagation on House Pricing dataset

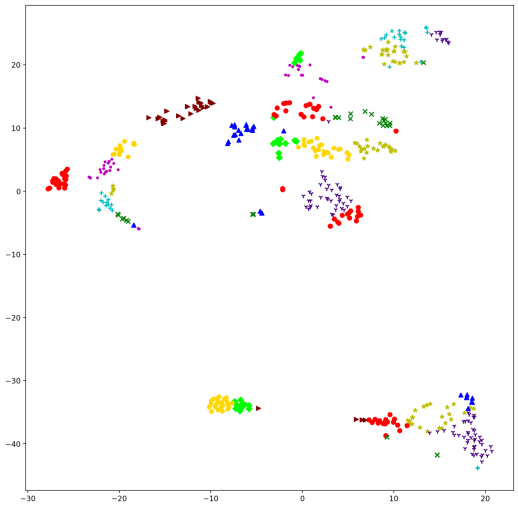


Figure 24: t-SNE plot of Affinity Propagation on House Pricing dataset

5.5.10 Wine Quality - K-Means

written by B.L.

The wine data set again presents a challenge due to its high feature count as well as having one discrete feature. Dimensionality reduction helps to speed up calculation but does not show an improvement in evaluation index performance. The plot of evaluation indices in figure 5.5.10 gives an indication for 2 (SI, DB), 3 (CH), or 6 (CH, DB) clusters

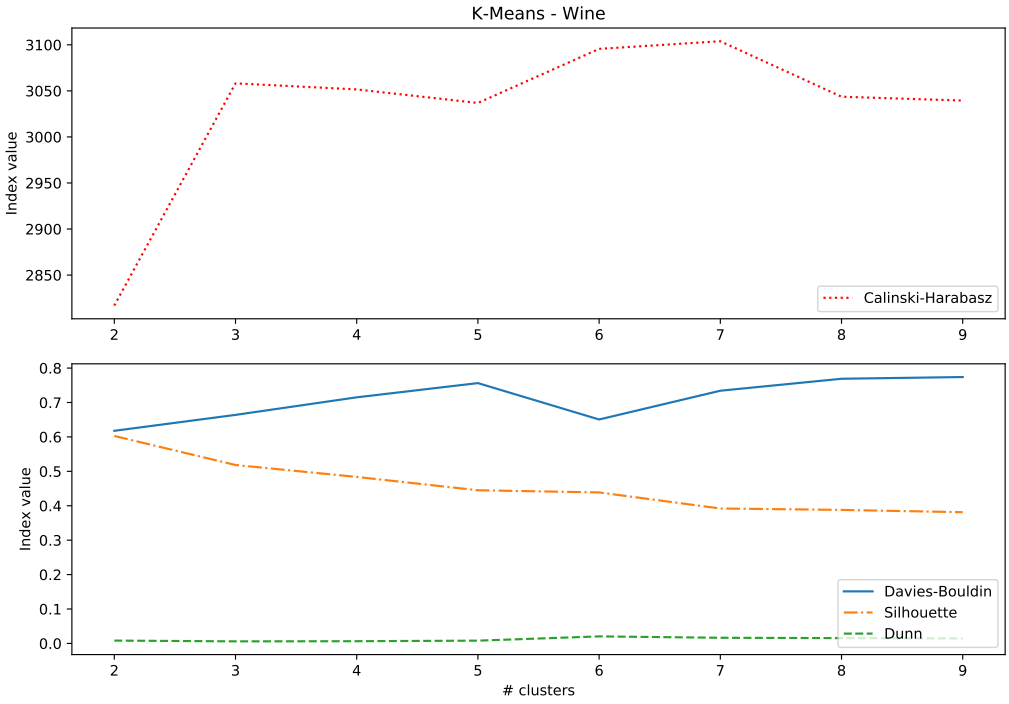


Figure 25: K-Means Wine Data Set Indices by Number of Clusters

but looking at the two dimensional transformed data in figure 5.5.10 for three and six clusters compared to the original labeling does not allow for any interpretation due to the fact that little about structure - if it exists - can be learned from the original labels.

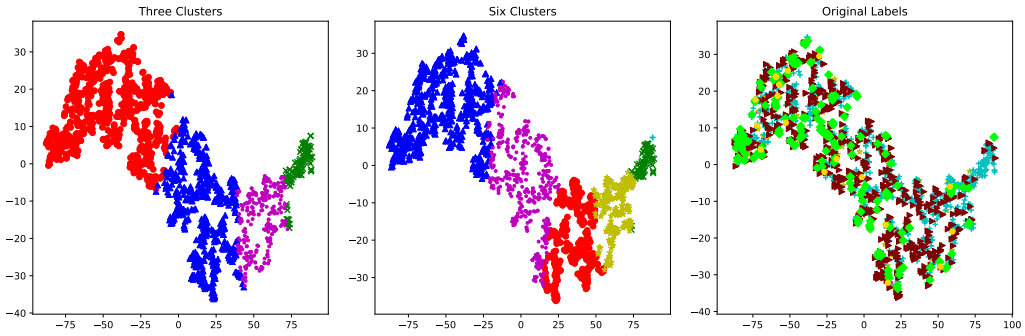


Figure 26: K-Means Wine Data Set Indices by Number of Clusters

Looking back at the PCA step mentioned before, the picture becomes clearer when utilizing it as a visualization support and looking at a three dimensional representation of the transformed data in figure 27 gives a clearer understanding of how the clustering has worked out. The component driving the differentiation between clusters called *PCA 1* captures 94% of the variance and loadings show it being driven in large part by the two *sulfur* concentration features.

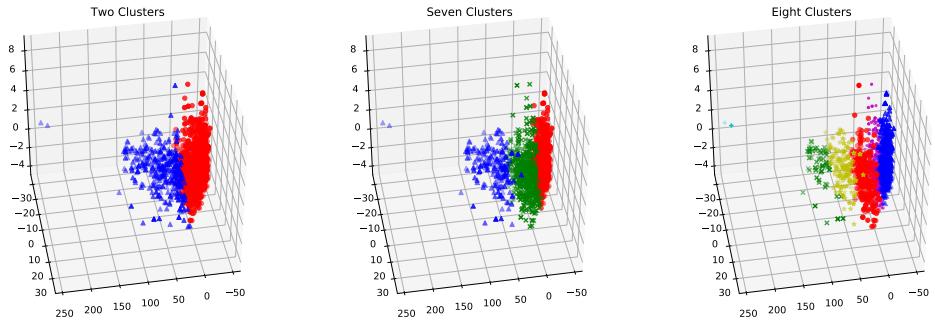
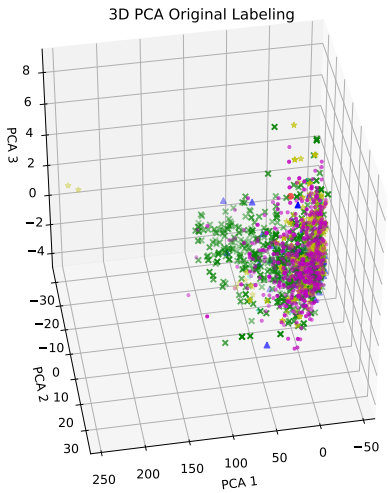


Figure 27: Wine Data Set 3D Plot Clustering Options

Looking at the original labeling of the data however shows the limitations of clustering and especially K-Means clustering for this data set. Figure 28 shows the same projection as figure 27 but this time with



the original labeling. The math on accuracy and recall at 19.8% and precision somewhat better at 58.2% also shows the less than optimal performance. However this is not unexpected when considering the strengths and weaknesses of K-Means described in section 2.1: contiguous, non-overlapping, convex clusters can be captured well while at least this representation of the data displays a lack of these characteristics in the data set.

5.5.11 Wine Quality - Mean Shift

written by L.B.

Applying Mean Shift algorithm to the last dataset shows that density-based clustering is also able to identify non-convex clusters. The following plots demonstrate how Mean Shift clustering performed on the Redwine Quality dataset along different values for the required bandwidth parameter.

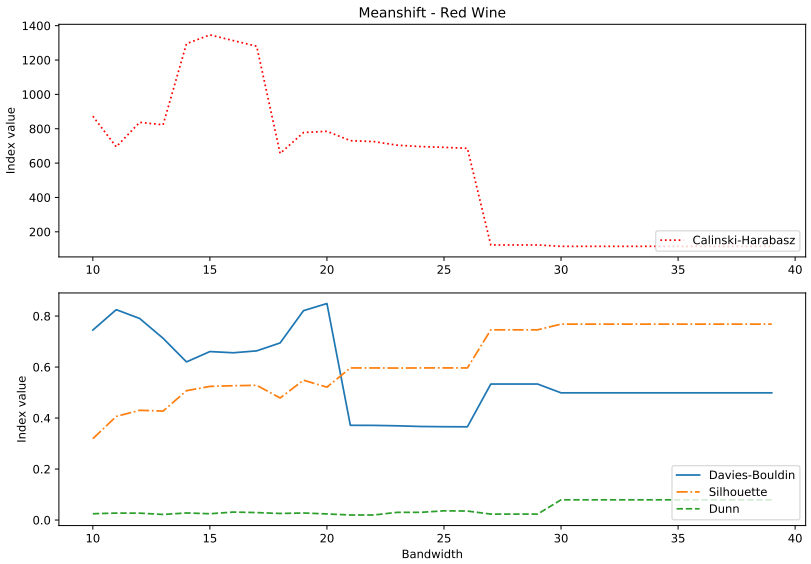


Figure 29: Cluster validation indexes: Mean Shift on Redwine Quality

The cluster validation indexes suggest that a bandwidth parameter of 22.5 generates the best clustering results. However, the overall performance on this dataset is worse compared to the House Pricing dataset, even though consists of less dimensions. This is the case as data points of the Wine Quality dataset lie relatively close together and no distant and clearly separated dense regions can be identified (Figure 5.5.11).

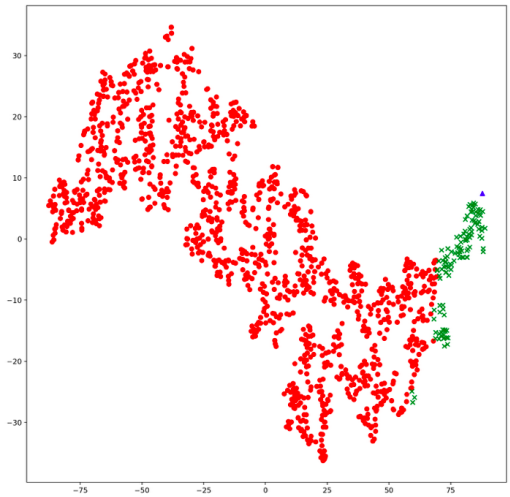


Figure 30: t-SNE projected result of Mean Shift with bandwidth = 22.5 on Redwine Quality dataset

5.5.12 Wine Quality - Affinity Propagation

written by J.M.

One the wine dataset affinity propagation works not good. The calculation to build cluster needs significantly longer because this dataset has more datapoints than the other datasets. After it computes the clusters it builds many clusters. 5.5.12 show a plot of the clustering. There are many cluster in it. The parameter for this examples wars – 6264. It don't work a good as the same reason as the house pricing dataset, because the dimensions are very high. The Silhouette Index is also low, only 0.32.

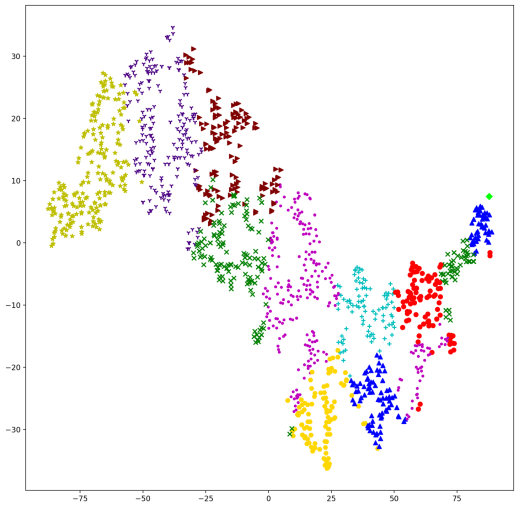


Figure 31: t-SNE plot of Affinity Propagation on Wine Quality dataset

6 Web Frontend and User Manual

written by L.B. and B.L.

The frontend has been developed in Streamlit [41] in its latest version as of June 20, 2021, 0.82.0 and can be accessed at `https://clustering.goethe.tech)`. On the left there is a sidebar containing two collapsible content groups. One of them is called `Regular` and comprises of controls for a single clustering method, the other named `Comparison` contains controls for a workflow that lets the user compare the different methods. First off there is the `Regular` group. It provides a menu containing two dropdown lists. While the upper dropdown list can be used to select the dataset to be clustered, the second dropdown contains four different clustering algorithms to be applied: K-Means, Mean Shift, Spectral Clustering and Affinity Propagation. According to the selected clustering algorithm, the required input parameter can be set by dragging the slider on the respective widget. For this purpose, a message displays which kind of input parameter is required for the selected algorithm. Default values for different parameters vary with regard to the chosen dataset. We therefore provide slider widgets with different value ranges and varying default values according to the dataset to be clustered. Once dataset and

clustering technique are chosen, the `Calculate` button can be clicked to run the clustering procedure. After the dataset is clustered, a two-dimensional projection plot of all data points is displayed in the main panel, distinguishing between different clusters with the help of distinct symbols and colors.

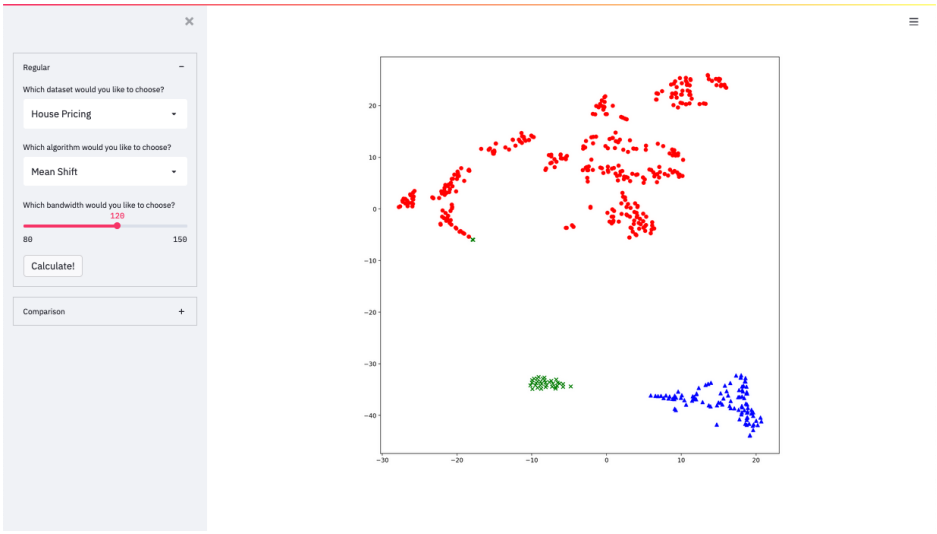


Figure 32: Frontend Regular Mean Shift House Pricing Example

The `Comparison` group contains a singular drop down list which can be used to select the data set of interest. Below this menu there are the same type of sliders with the same functionality as described above, but in this case for all the methods at the same time. Clicking the `Compare` button runs all clustering methods with the specified parameters and displays them next to one another in the main panel, similar to the method described above. Located below these plots is a table containing the index values for each index described in section 5 (rows) for each clustering method (columns).

	K-Means	Mean Shift	Affinity	Spectral
Calinski-Harabasz	375.8050	374.1991	331.3587	335.5825
Davies-Bouldin	0.7533	0.7540	0.7241	0.7694
Silhouette	0.4719	0.4697	0.4984	0.4484
Dunn	0.0855	0.0509	0.0487	0.0797

Figure 33: Frontend Comparison Seeds Example

Our code implementation is divided into segments to isolate clustering procedures from projecting the data into 2D and from visualizing the results.

Each of the clustering techniques that can be selected for grouping data points is implemented in a separate file which is named according to the technique. To run one of the clustering algorithms, the dataset has to be handed over in form of an $n \times m$ matrix () consisting of n data points with m features. Moreover, a value for the algorithm specific parameter has to be provided. The `returns` parameter can be used to define what output format is expected. We provide the possibility to distinguish between returning the fitted estimator, an $n \times (m + 1)$ - matrix consisting of n data points with m features and a column containing a label for each data point (a task facilitated by `numpy` [42]) or an $n \times 1$ vector, which only includes a label for each data point. These options exist in order to make it possible to return the right format for the task at hand, be it calculations for this work, for evaluation or plotting or inside the frontend itself. Running the clustering in its default mode, each technique returns the $n \times 1$ label vector. For the K-Means algorithm, the additional parameter `state` can be set. This value is used to initialize the random number generator such that always the same results are yielded [26].

In the top level `app.py` file, we implement all the functionalities concern-

ing the frontend elements as well as choice of dataset and clustering technique. After reading the four different datasets using the `read_csv` method provided by pandas [43, 44] non-numeric values in the data are converted to numeric and we calculate a version of the datasets transformed into 2D and cache it to speed up all following interactions because this step can take long for large data sets. The `tsne_transform` function performing this step also takes a state to ensure the resulting plots later will always look the same to ease comparability. To transform the datasets into 2D, inside `tsne_transform` we make use of the t-distributed Stochastic Neighbor Embedding (t-SNE) method provided by sklearn [26].

`t-SNE` is a method that facilitates displaying high-dimensional data in a 2D or 3D plot while nevertheless preserving local and global structures. This is done converting Euclidean distances between all data points in high dimensions into conditional probabilities such that high probabilities are allocated to similar data points while dissimilar data points are assigned a low probability. Afterwards, the same is done for all the data points projected into lower dimensions. The optimal projection of high-dimensional data points is finally found by minimizing the mismatch between calculated probabilities between points in high and in low dimensions [45].

As described earlier, we not only provide dropdown lists to enable the choice of a dataset as well as a clustering technique, but also adapt the slider widget label according to the required input parameter for the chosen algorithm. We define parameter value ranges along with default values appropriate to the chosen dataset. To run the clustering procedure, an event handler listens for click events on the `Calculate` button. According to the selected clustering algorithm, we call the respective clustering function and hand over the dataset as an $n \times m$ - matrix as well as the technique specific parameter value which was selected with the help of the slider widget.

Largely the same happens when using the `Comparison` workflow. The main difference is in the fact that selecting an algorithm is not required because all of them will be run one after the other and plotted next to each other accordingly. The comparison table is then calculated by using the `evaluation` module, returning index values and transforming them into an overview.

The plotting as described above is based on t-SNE transformation and executed by calling the `plot_tsne_2` from the plotting module. The resulting figure comprises of scatter plots from the Matplotlib visualization function `pyplot` [46] overlayed atop each other for each cluster. This results in one

single plot containing all data but each cluster differentiated by color and marker shape making it easy to distinguish between clusters.

7 Conclusion

After brief conclusions on each clustering method follows a more general conclusion discussing observations and particularities that stand out when looking at the functioning and results for all methods on all data sets with all evaluation methods.

K-Means - Generally K-Means clustering is a fast, easy to use and reliable method to cluster data. For house pricing, seeds and mall customer data it works well at segmenting into separate partitions which can be inspected visually to be what a human would describe as distinct clusters. When the underlying structure of the data lends itself to the globular type of cluster shapes K-Means is known for, like the seeds data set, the result is dependable as the use of classification scores has shown. House pricing and wine quality data show that high dimensional and overlapping data poses a challenge because especially in the wine data set (barely any separation) K-Means *will* find clusters of similar size and shape even if those do not represent the underlying data well.

Mean Shift - All in all, Mean Shift algorithm is known to work best on low-dimensional datasets but nevertheless, it performed better on our high-dimensional datasets about Boston House Pricing and Redwine Quality in contrast to the low-dimensional datasets about Seeds and Mall Customer Data. As Mean Shift is density-based, this phenomenon can be explained taking not only dataset dimensionality but also density into account. For the best Mean Shift performance, datasets consisting of dense regions which are distant from each other are required. The optimal value for the bandwidth parameter has to be evaluated for each dataset individually.

Affinity Propagation - Overall, we would rate affinity propagation as a medium cluster algorithm because it can compute a good clustering, but this depends on the given dataset and on the parameter values. Affinity Propagation clustering results for the first and second dataset seem to be suitable, but those for the third and fourth dataset do not represent plausible structures. Applying the Affinity Propagation algorithm without fitting the parameters will not result in a good clustering. This happened in all four datasets. Sometimes, the algorithm does not converge which makes it rather

unstable. An advantage of Affinity Propagation is that it does not require the number of clusters beforehand. Because of this property, Affinity Propagation is good if you don't know much about the dataset, but the calculation is slowly because it depends on the datapoints in the dataset. It also works better with datasets consisting of low dimensions like the Mall Customers and the Seeds dataset. For a dataset which has many dimensions, Affinity Propagation will find a lot of cluster.

As a high level overview it can be said that no one single algorithm outperforms the others in all or most circumstances. Each has particular advantages which could be seen for the different datasets. Some particularities can however be noted. Using K-Means and Mean Shift can regularly produce remarkably similar clustering results for the datasets at hand despite their different methods of arriving there.

For visually reasonable clusterings, CH often sees highest values for K-Means clustering, which makes sense given its construction (see section 5). However, its meaningfulness does seem to break down when there is an excessive number of clusters, possibly skewing the perception of clustering success when solely relying on this type of evaluation as can be seen in table 4. On this topic it should be reiterated that using evaluation indices on the same algorithm for different parameter configurations might be more meaningful than comparing different algorithms when not taking additional information into account. Also relying on a single measure can lead to one sided conclusions.

A few observations on the application of the clustering algorithm implementations: Affinity Clustering seems to be somewhat unstable when trying out different preference parameter values, regularly failing to converge even when significantly increasing the iteration limit. Runtime for Affinity Clustering and Spectral Clustering can be notably longer than Mean Shift and especially the utilized fast method for K-Means.

Given the opportunity to compare clustering results with original labelling for two of the data sets confirms the expectation that accurately assigning observations to groups is harder when separation is lacking. This is particularly evident in the results for wine quality data where all algorithms do find clusters but the actual data is highly dispersed and overlapping.

All in all, it is highly dependent on the given dataset in terms of its dimensionality and distribution of data points which algorithm should be applied to find a reasonable clustering result.

Acronyms

CH	Calinski-Harabasz Index.
DB	Davies-Bouldin Index.
DI	Dunn Index.
SI	Silhouette Score.
sklearn	Scikit-Learn.
t-SNE	t-distributed Stochastic Neighbor Embedding.
VRC	Variance Ratio Criterion.

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] RALPH D'AGOSTINO and Egon S Pearson. Tests for departure from normality. empirical results for the distributions of b 2 and b. *Biometrika*, 60(3):613–622, 1973.

[32] 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.

[33] Simona Balbi, Michelangelo Misuraca, and Maria Spano. A cosine-based validation measure for document clustering. In *JADT*, volume 2016, page 13ème, 2016.

[34] Douglas Rizzo. Dunn sklearn. `https://gist.github.com/douglasrizzo/cd7e792ff3a2dcaf27f6`, 2016.

[35] Chiheb-Eddine Ben Ncir, Abdallah Hamza, and Waad Bouaguel. Parallel and scalable dunn index for the validation of big data clusters. *Parallel Computing*, 102:102751, 2021.

[36] Douglas Rizzo. Dunn index for clusters analysis. `https://douglasrizzo.com.br/dunn-index/`, 2019.

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

[38] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M Pérez, and Inigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

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

[40] 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.

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

[42] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer

Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[43] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[44] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[45] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[46] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.