Term Paper Data Science 1

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

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

Authors: Franziska Hicking
6673525
franziska.hicking@stud.uni-frankfurt.de
Master Bioinformatics, 2
Jonas Elpelt
6673181
elpelt@stud.uni-frankfurt.de
Master Bioinformatics, 2
Julian Rummel
6673334
s9594673@stud.uni-frankfurt.de
Master Bioinformatics, 2
Niklas Conen
6599913
conen@stud.uni-frankfurt.de
Bachelor Computer Science, 8

Date: June 14, 2021

Chosen Project Topic:
T4 - DISTANCE MEASURES AND CLUSTERING

Abstract

Clustering algorithms can be important tools during the analysis of datasets. They divide a dataset into groups of items based on a certain measure of similarity such as the distances between each of the items. In this work, we implemented and compared four different clustering algorithms (K-Means, K-Medoids, K-Median, DBSCAN). For this, we selected four distinct datasets as well as multiple distance measures (Manhattan, Euclidean, Angular cosine, Chebyshev). For efficient comparison of the clustering results we made use of multiple clustering indices. Additionally, we implemented a web frontend which provides the ability to run all clustering algorithms with distance measures, datasets and clustering indices chosen by the user. The results will be visualized afterwards. After running all algorithms with each of the datasets respectively and all distance measures where they could be applied, we compared the resulting values of the clustering indices. Our results show that

Contents

1 Definition of Distance Measure

A distance measure is a function $d(x, y)$ that calculates a real value between two points in a space, containing two sets of points. If $d(x, y)$ satisfies the following three axioms the distance measure is classified as a *metric*:

$$d(x, y) = 0 \Leftrightarrow x = y \qquad \text{Identity of indiscernibles} \qquad (1a)$$
$$d(x, y) = d(y, x) \qquad \text{Symmetry} \qquad (1b)$$
$$d(x, y) \leq d(x, z) + d(z, y) \qquad \text{Triangle inequality} \qquad (1c)$$

The triangle-inequality imposes the condition that a distance reflects the shortest path between two points. Thus, it is not possible to achieve a distance improvement by traveling via an intermediate point z. [1]

Moreover all axioms enforce non negative distances as an additional condition.

$$d(x, y) \geq 0 \qquad \text{Non Negativity} \qquad (1d)$$

2 Different Distance Measurements

2.1 Manhattan Distance

To determine the distance between two items the Manhattan distance, also referred to as taxicab distance, may be used. This distance measure assumes a n-dimensional vector space with a fixed cartesian coordinate system. It is defined as following:

$$d(x, y) = ||x - y||_1 = \sum_{i=1}^{n} |x_i - y_i|$$

where x and y are vectors

$$x = (x_1, x_2, ..., x_n) \text{ and } y = (y_1, y_2, ..., y_n)$$

The Manhattan distance is, like the Euclidean distance, part of the $L_p - metrics$ (see 2.2), where the value for p is set to 1. Assuming a two dimensional space, the distance between two points is the shortest path between

them with the restriction of only being able to move vertically and horizontally. Prove of axioms described in section 1:

1. Identity of indiscernibles:
   Let $x = y$, then $|x - y| = 0$ and hence $\sum_{i=1}^{n} |x_i - y_i| = 0$
2. Symmetry:
   This axiom is fulfilled since $|x - y| = |y - x|$ for any x and y , which implies that $\sum_{i=1}^{n} |x_i - y_i| = \sum_{i=1}^{n} |y_i - x_i|$.

2.2 Euclidean Distances

For a variable $p \in \mathbb{N}$ the the *L_p-metrics* are defined as

$$d(x, y) = \sum_{i=1}^{n} (|x_i - y_i|^p)^{\frac{1}{p}} \tag{2}$$

Setting $p = 2$ expresses the Euclidean distance, which is defined as the positive square root of the sum of all squared distances in each dimension:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (|x_i - y_i|)^2} \tag{3}$$

The first two axioms defined in section 1 are easily shown to apply:

1. Identity of indiscernibles:
   For $x = y$ the value is obviously 0. Let $x = y$, then $(|x - y|)^2 = 0$ and $\sqrt{0} = 0$.
2. Symmetry:
   Symmetry is cleary given by the square of each distance.
   $(x - y)^2 = (y - x)^2$.

Non negativity is also shown quite easily. The square of any real number is always positive and the squareroot of any real positive number is always positive. Hence $d(x, y) \geq 0$.

The traingle inequality requires a more difficult proof. However, to keep it simple, the Euclidean space possesses the property that the sum of the lengths of Cathetus and Ancathetus is always longer than the length of the Hypothenuse. [1]

2.3 Angular Cosine Distance

The angular cosine distance gives the (normalized) angle between two points x and y represented as vectors in an n-dimensional space. It does not make a difference between a vector and a multiple of that vector. The cosine distance can be calculated by applying the arc-cosine function to the cosine of the angle θ between x and y [1].
It is based on the cosine similarity (cosine between two vectors x and y), which is definied as:

$$\text{cosine similarity} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sum_{i=1}^{n} y_i^2} \tag{4}$$

The cosine similarity, however, is not a distance as it is defined for positive values only. Therefore it has to be converted to the normalized angle between x and y as followed [2]:

$$\text{angular cosine distance} = \frac{\arccos(\text{cosine similarity})}{\pi} \tag{5}$$

Note, that if x or y are zero vectors, the cosine similarity would not be defined. To prevent a division by zero the cosine similarity is set to 1 in this special case (based on the implementation of the pairwise distance in scikit-learn [3]).

The axioms for a distance measure are fulfilled for the cosine distance [1]:

1. Identity of indiscernibles:
   Two vectors can have a cosine distance of 0 if and only if they are located in the same direction. (This applies also to vectors that are multiples of one another and therefore are in the same direction.)

2. Symmetry:
   Symmetry is obviously given by the equality to measure an angle between x and y and an angle between y and x.

3. Triangle inequality:
   A rotation from x to y can be explained by a rotation from x to z and then to y. Therefore a sum of these two rotations is always bigger or equal than the rotation directly from x to y

4. No negative distances:
   Regardless of the dimensionality of the space the values of the cosine distance are between 0 and 180 degrees, therefore no negative distances can occur.

2.4 Chebyshev Distance

The Chebyshev distance (also known as Tschebyscheff distance, Maximum Value distance or L_∞ distance) is the limit of the before mentioned L_p-metrics (equation 2). On a vector space this metric is induced by the Supremum Norm (also called Chebyshev Norm or Infinity Norm), which again is the limit of the L_p-norms.
Descriptively the Chebyshev metric is the greatest distance between two vectors on one axis. Formally it is defined as:

$$d(x, y) = \max(|x_i - y_i|) \tag{6}$$

which is the aforementioned limit of the L_p-metric and is therefore also called L_∞-metric:

$$d(x, y) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} (|x_i - y_i|^p)^{\frac{1}{p}} \right) \tag{7}$$

The three axioms for a metric (section 1) are proven below:

1. For $x = y$ all entries of a vector are identical and all differences between $x_i - y_i$ are 0. Thus: $d(x, x) = \max(|x_i - x_i|) = \max(0) = 0$

2. Symmetry is given because of the symmetry of the absolute value function: $|x_i - y_i| = |y_i - x_i|$

3. The triangle equation can be shown using some estimates:

$$\begin{aligned} \max(|x_i - y_i|) &= \max(|x_i - z_i + z_i - y_i|) \\ &\leq \max(|x_i - z_i| + |z_i - y_i|) \\ &\leq \max(|x_i - z_i|) + \max(|z_i - y_i|) \\ \Rightarrow d(x, y) &\leq d(x, z) + d(z, y) \end{aligned}$$

Non negativity also results from the non negativity of the absolute value function. Therefore the Chebyshev distance is classified as a metric.

3 Data Set Description

3.1 Housevotes

The housevotes dataset, created by Jeff Schlimmer in April 1987, was taken from the UCI Machine Learning Repository [4]. The dataset consists of voting results of the U.S. House of Representatives Congressmen on 16 key votes during the second session of Congress in 1984. The key votes and the voting results are identified by the Congressional Quarterly Almanac (CQA) documenting this session of Congress. The voting results are split into nine different types by the CQA, which are consolidated into three results used in the dataset.

Voted for, paired for, and announced for count as a yes vote. Voted against, paired against, and announced against count as a no vote. Voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known are denoted as a unknown state.

The set consists of two classes, 267 democrats and 168 republicans.

3.2 Wine recognition dataset

This dataset contains the chemical analysis results of Italian wines from 3 different cultivators. It is also taken from the UCI Machine Learning Repository [4]. The dataset consists of 178 instances, each of them having 13 numeric attributes according to different measurements taken for different constitutuents (alcohol, malic acid, ash, alcalinity of ash, magnesium, total

phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, proline). Each instance belongs to either one of three classes containing 59, 71 and 48 data points. It was created by R. A. Fisher in July 1988 [3].

3.3 Iris dataset

The Iris Dataset, introduced by Ronald Fisher in 1936, contains the petal and sepal measurements of three different species of Iris [5]. The considered Irises are Setosa, Versicolour and Virginica. For each Iris, 50 samples are included and for each sample the dataset contains the sepal length, sepal width, petal length and petal width in cm.

3.4 Diabetes dataset

The diabetes dataset contains various information as numeric values about 442 diabetic patients, namely age, sex, body mass index, average blood pressure, and six blood serum measurements (first 10 columns), as well as a quantitative measure of disease progression one year after baseline, i.e., the response of interest (11th column). All characteristics were standardized to standard deviation times n samples and also mean centered.
This dataset is taken from the diabetes study conducted by Efron et al. [6] with the main goal of constructing a model that predicts the response (column 11) from the covariates (column 1-10).

4 Clustering Algorithms

4.1 K-Means

The K-means algorithm aims to group together similar items of a given dataset into clusters. The total number of clusters is predefined and represented as the value for k. All considered items can be referred to as points, as this clustering algorithm assumes an Euclidean space. Hence, only distance measures which assume an Eulicdean space, such as the Manhattan distance or the Euclidean distance, are sensibly appicable. The K-means algorithm belongs to the point-assignment algorithms in clustering, as all points are considered successively and assigned to the most fitting cluster. The algorithms operates in the

following steps: [1]

1. Initially, the algorithm picks k points whose positions each represent one cluster centroid
2. All points are considered in turn:
   - Find the nearest centroid/mean of the considered point (Euclidean distance measure)
   - Assign point to cluster of that centroid
   - Adapt position of this centroid
3. (optional) fix all centroids and reassign all points with the inclusion of the initial k points

The essential first step of initializing the clusters requires k points that have a high chance of being in separate clusters. This can be achieved by different approaches. One possible approach consists of picking points which are as far away as possible from each other. This can be achieved by the conducting the following steps:

1. A random point is picked as the first of k cluster centorids
2. For k-1 passes:
   Pick the point whose minimum distance is the largest considering all previously chosen points

Another way of determining the cluster centroids is the K++ initializer, described in section 5, which we used in our implementation of the K-means algorithm.
After the K-means algorithm assigned all points, an optional step af reassigning the points with fixed centroids can be conducted. This can be sensible since it is possible that after a point has been assigned to cluster the centroids move so far that the point would now belong to a different cluster.

4.2 K-Medoids

The K-Medoids clustering method is related to the well-known K-means algorithm, but uses medoids (representative points for each cluster) instead of means to define new cluster centers, which makes it more robust to outliers [7]. It partitions the dataset by assigning each data point to the closest of k cluster

centers, which are defined by the most centrally located medoids. A medoid is a point with a minimal average dissimilarity to all other data points in the same cluster. The most commonly used algorithm to solve this NP-hard problem heuristically is the PAM (Partitoning Around Medoids) algorithm, that works as following: [8]

1. First initialize the algorithm by selecting k data points to be the medoids and assigning every data point to its closest medoid. The initial data points can be found by a k++ approach (similarly used by kmeans) [9].

2. Compare the average dissimilarity coefficient of a swap of each medoid m and a non-medoid data point $\bar{m}$. Find a swap between m and $\bar{m}$ that would decrease the average dissimilarity coefficient the most.

3. If no change of a medoid happened in the second step, terminate the algorithm, else re-assign the data points to the new medoids and go back to step 2.

4.3 K-Median

The K-Medians cluster algorithm is closely related to the K-Means algorithm, but is more robust to outliers because it uses the median as statistics in order to determine the center of each cluster. Its main approach is to cluster data by minimizing the absolute deviations, corresponding to the Manhattan distance, between each point and its closest cluster center, i.e., creating k disjoint cluster by minimizing the following function. [10]

$$Q(\{\pi_j\}_{j=1}^{K}) = \sum_{j=1}^{K} \sum_{x \in \pi_j} ||x - c_j||_1 \tag{8}$$

The geometric median is used for the minimization.

$$\underset{y \in \mathbb{R}^n}{arg\,min} \sum_{i=1}^{m} ||x_i - y||_2 \tag{9}$$

At the start of the algorithm, k cluster centers must be initialized. There are many different approaches to perform this task, such as Random Initialization,

Density Analysis, Single Dimension Subsets, and many more. In this work, the random approach was used because many of the other theories, while theoretically promising, are inferior or nearly equivalent in performance to the results produced by random initialization. [10] Achieving global optimization in k-medians is known to be NP-complete [11].
The algorithm works as following: [12]

1. Assign each dataset to a cluster, thus its nearest cluster center using the Manhattan distance as default.
2. Shift the cluster centers to the position of the vector whose elements are equal to the median value of each dimension of all instances in a cluster.
3. There is no guarantee to get the perfect cluster because the starting cluster centers were initialized randomly. There is the approach of reinitializing the algorithm many times and securing the best cluster center of all iterations.

4.4 DBSCAN

DBSCAN was developed by Martin Ester, Hans-Peter Kriegel, Jiirg Sander and Xiaowei Xu. All following definitions and descriptions are taken from their original publication [13] or their revisit of DBSCAN [14] and only apply to this algorithm.
Contrary to the aforementioned centroid-based partitioning algorithms (k-means, k-medoids and k-median) the DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) algorithm uses point densities to determine clusters.
To introduce the definition of the density of a cluster, first the Eps-neighbourhood of a point is defined:

Definition 1: *Eps-neighbourhood*
A point q is part of the Eps-neighbourhood N_{Eps} of point p if the distance between them is smaller than a threshold distance called Eps.
The Eps-neighbourhood therefore is defined as $N_{Eps} = \{q \in D \mid ||p, q|| \leq Eps\}$ with D denoting the entirety of points that are supposed to be clustered and $||p, q||$ being the distance between p and q for an arbitrary distance measure.

The Eps-neighbourhood fails at being a reliable measure for the point density if a point is located at the border of a cluster. These points are called *border point*. Points that are located on the inside of a cluster are called *core points*. Hence the following definition is made:

Definition 2: *directly density-reachable and density-reachable*
A point p is directly density-reachable from a point q when

1. $p \in N_{Eps}(q)$
2. $|N_{Eps}(q)| \geq$ MinPts

with MinPts being the minimal number of points that $N_{eps}(q)$ should contain so that q is considered a core point of a cluster.
A point is *density-reachable* if there is a chain of points between p and q so that all neighbouring points in the chain are directly density reachable.

To complete the definition of what is considered part of a cluster density-connectivity is defined:

Definition 3: *density-connected*
Two points p and q are considered density-connected if there is a common point o which is density-reachable from p and q.

Now a cluster can be described as:

Definition 4: *cluster*
A cluster is a non empty subset $C \in D$ so that:

1. $\forall p, q : p \in C \wedge q$ is density reachable from $p \Rightarrow q \in C$
2. $\forall p, q \in C :$ p is density-connected to q

Noise is easily defined as every point that is not part of a Cluster C_i.

Using these definitions DBSCAN can begin the clustering process with given values for Eps and MinPts. In the beginning all points are not labeled. Beginning with an arbitrary point p all points are iterated in a linear fashion. For each point a `RangeQuery` function is executed finding all density-reachable neighbours of p. If `RangeQuery` finds more than MinPts neighbours then p is a core point and is labeled as such. Otherwise p is marked as Noise.

In the next step every point in the Neighbourhood excluding p is expanded. Unlabled Points get checked for the core point condition (which equals a `RangeQuery` call). Points that got labeled as Noise before are labeled as core points. When the expansion comes to an end a cluster is yielded and the next unlabeled point is chosen as p.

Two clusters may be merged if their distance is below Eps. The distance between two clusters C_1 and C_2 is defined as $||C_1, C_2|| = \min\{||q, p|| \mid p \in C_1, q \in C_2\}$.

The runtime complexity of DBSCAN heavily depends on the runtime of the `RangeQuery` function and the distance measure. Thus the runtime can exceed $\mathcal{O}(n^2)$ depending on the chosen implementations. A detailed discussion of DBSCANs runtime can be found in [14].

5 Additional Methods Used

5.1 k++-Initializer

The K++ initializer is a method which determines the positions of the initial k cluster centroids before starting the K-means algorithm. The following steps outline how the K++ initializer operates:

1. Choose one centroid at random from all possible datapoints.
2. Loop through all remaining datapoints (k-1):
   determine the distance $D(x)$ of the datapoint to the nearest already chosen centroid using the Euclidean distance.
3. A weighted probability distribution is used to set the next cluster centroid. For each point, the probability to be chosen is proportional to $D(x)^2$, meaning points which are farther away are more likely to be chosen.
4. Steps 2 and 3 are repeated until all k cluster centroids have been set.

5.2 One-Hot-Encoding

Categorical data is represented by specific discrete values or labels. This is case for the housevotes dataset (see section 3.1), where voting results can have one of three values (y, n, ?). The distance measures described in section 2 need numerical data to work. The categorical data therefore needs to be converted (encoded) to numbers which accurately describe their distance to another. Using simple integer encoding where no is encoded as 0, yes is encoded as 1 and the unknown state is encoded as 2 results in a yes vote being classified as closer to the unknown state than a no vote by the distance measures.

For so called non ordinal data (data which has no known order) like the votes, One-Hot-Encoding provides better results when using distance based clustering.
With One-Hot-Encoding every attribute is represented as binary vector. Each element of this vector represents a category value. The corresponding value of a sample is set to 1 in the binary vector. This increases the dimensionality of the problem, but represents an equal distance between every value an attribute can have.

The scikit-learn library [3] was used to implement One-Hot-Encoding on the housevotes dataset.

5.3 t-SNE

t-SNE is a nonlinear dimensionality reduction technique, which was developed by Laurens van der Maaten and Geoffrey Hinton [15]. It can be used for visualizing high-dimensional data in a lower-dimensional (typically 2-dimensional) space such that more similar data points should be represented nearby in the lower-dimensional representation. This can lead to visual cluster formation based on the local structure of the data (and chosen parameters) [16].
The t-SNE algorithm first calculates the distances $d(x_i, x_j)$ (by default using the euclidean distance) between each of the N data points x_i and x_j [17]. Then it computes conditional probabilities $p_{j|i}$, " that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i." [15]

$p_{j|i}$ for $i \neq j$ is given as

$$p_{j|i} = \frac{exp(-d(x_i, x_j)^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-d(x_i, x_k)^2/2\sigma_i^2)} \tag{10}$$

and $p_{i|i} = 0$ is set.
The joint probability p_ij is defined by

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{11}$$

Note that the Gaussian distributions should have their standard deviations σ_i such that the perplexity of the conditional distribution is equal to a predefined perplexity parameter [17]. It basically measures the effective number of neighbours of the data point i, that can be found performing a binary search. In the next step t-SNE searches for an embedding of the data points considering the previously computed similarities [17]. This is achieved by minimizing the Kullback-Leibler divergence between the modeled Gaussian distributions of the high-dimensional data points X and a Student t distribution of the corresponding points Y in the lower-dimensional space. To do this we define q_{ij} for $i \neq j$ as followed

$$q_{ji} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_t\|^2)^{-1}} \tag{12}$$

and set $q_{i|i} = 0$.
Now the Kullback-Leibler divergence can be expressed as

$$KL(P||Q) = \sum_j \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{13}$$

The optimization procedure is performed by a gradient descent method to find a local minimum [17].
The final results may heavily depend on the chosen parameters, especially the perplexity value [16]. It is therefore recommended to compare different perplexity values to identify spurious clustering artifacts in the visualization.

5.4 Principal Component Analysis (PCA)

PCA is a dimension reduction technique to increase interpretability while minimizing information loss during the process. Working with a dataset containing p numerical variables and n entities, a nxp-Matrix X gets defined with p vectors as columns. Now linear combinations (see: 14) of the columns of X with maximum variance (see: 15) are searched for, given by:

$$\sum_{j=1}^{p} a_j x_j = Xa \tag{14}$$

$$var(Xa) = a^T Sa \tag{15}$$

with a as vector of constants $a_1, ..., a_p$ and S as sample covariance matrix associated with the dataset. [18]
These linear combinations are called principal components and are p uncorrelated, new variables for the initial variables. Most of the information of the original data is compressed in the first principal component, with reduced but maximized information in the following components. It is highly important to understand the correlation between variance and information. The greater the variance, the greater the dispersion and thus the greater the abundance of information. Eigenvectors and eigenvalues are needed to calculate the principal components, where the eigenvectors of the covariance-matrix S are the directions of the axes where most of the variance is present and the corresponding eigenvalues indicate the amount of variance contained in each principal component. [19] This produces the equation: [18]

$$Sa - \lambda a = 0 \Leftrightarrow Sa = \lambda a \tag{16}$$

Thus, ranking the eigenvectors in order of their eigenvalues, one obtains all principal components from 1 to p. In the following step, the user can decide whether to keep all principal components or discard some of them based on their calculated significance, by: [18]

$$\pi_j = \frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j} \tag{17}$$

This results in a matrix called feature vector, which contains all the remaining components as columns and forms the final dimension of the reduced data

set [19]. Usually, the requirements of graphical representation lead to keeping the first two to three principal components [18]. Finally, the original data gets reorientated from the original axes to the axes represented by the principal components.

5.5 DBSCAN Parameter Estimation Heuristic

In the original publication of the DBSCAN algorithm [13] a simple way for approximating the minPts and eps parameters was proposed.

The described heuristic defines a function `k-dist` which calculates the distance d of each point to its k-nearest neighbour. The d-neighbourhood of each point then contains k+1 points in most cases. Theoretically it could happen that two points share the same distance d to a point. Then the d-neighbourhood could contain more than k+1 points though this case is most unlikely.

Sorting the resulting distances in decending order and plotting them against the points results in the so called sorted k-dist graph. This graph gives some insights about the datasets density distribution. Choosing any point in this graph, setting minPts to k and eps to the k-dist value of the point will result in DBSCAN classifing every point with the same or smaller k-dist value as core point. These points will therefore be assigned to a cluster. Points with a larger k-dist value could be, but do not have to be, classified as noise.

In the sorted k-dist graph a threshold point can be searched describing the maximal distance of the thinnest cluster. The DBSCAN paper [13] proposes a visual approach on finding such threshold points by looking for a kind of bend or "valley" in the graph. In figure 1 an example is shown for such a bend found in the 4-dist graph of the iris dataset using the euclidean distance.

While this method provides good results for some datasets, sometimes the estimated parameters do not produce a meaningful clustering or a bend may not be visible in the sorted k-dist graph. In some cases the distance chosen may result in one large cluster. The eps must then be reduced until DBSCAN can find multiple clusters, most likely resulting in more noise.

An interactive display of this heuristic with adjustable parameters for all datasets was also created using streamlit and altair and is hosted on streamlit sharing [1] The plot in figure 1 was created using this interface.

[1] `https://share.streamlit.io/elpelt/datascience1_group42/main/code/heuristic_web.py`

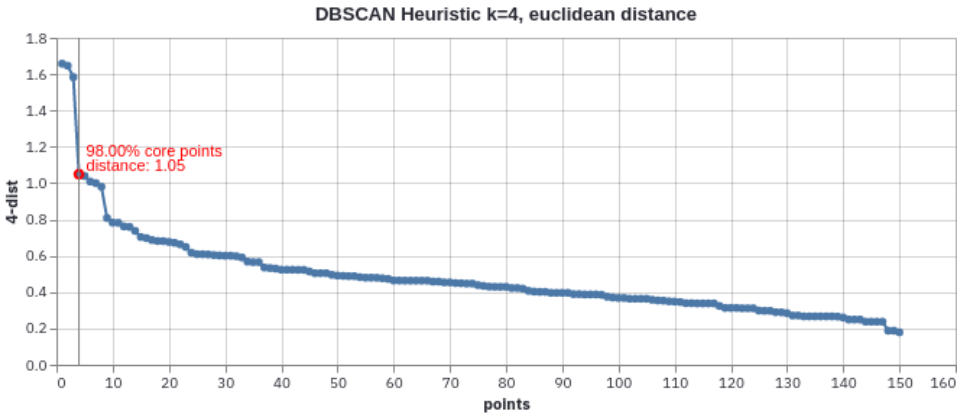


Figure 1: sorted 4-dist graph for iris dataset and euclidean distance

6 Implementation

6.1 Description

The application follows an object oriented approach. To unify the implementation of all cluster algorithms a base class, not meant for actual instanceing, was defined aggregateing all functions that the cluster algorithms need. The cluster algorithms themselves are implemented as child classes inheriting all methods and attributes from the base clustering class. This allows setting different parameters for every cluster algorithm while guaranteeing the existence and uniform execution of methods used by all algorithms.

Three different libraries were used for their implementations of the used cluster algorithms. For kmeans and kmedians the pyclustering [20] implementations were used, because of their flexibility in setting custom distance measures. For Kmedoids the scikit learn extra [9] implementation was used. DBSCAN is realised using the scikit learn implementation [3].

To reduce time spent on computing a class for organising already calculated resulsts as json files was created. All clustering results using the option of a predefined fixed seed are, if available, loaded from a file, or are calculated and then saved. Optionally cluster results can be bulk computed using a given script (*result_calculation.py*). Because of the large amount of possible parameter combinations of DBSCAN we decided to precompute only results for kmeans, kmedians and kmedoids with k ranging from 1 to 10 for all

distance measures.

The calculation of cluster indices is packaged into a Indices class and uses scikit learn implementations of the different scoring methods. Clustering results used for index calculations are saved using a data structure called SessionState. This structure is written by Thiago Teixeira [21] implementing a way of saving parameters per session. That is for example a browser tab in which the interface is opened. The variables saved in the SessionState are persistent until a session is closed, eventhough streamlits framework is designed in a way, where every page is a sequential python script being completely reloaded when an action is triggered.

Finally the webfrontend is implemented using streamlit and is hosted using their sharing service [2]. The cluster plots are generated using either seaborn or altair given a flag one can set in the frontend. The scoring results are visualised using matplotlib. The graphs which are visualised using altair are interactive in the way, that the user can freely move and zoom through the plots. By hovering over a point informations about this plot are shown in a small popup window. Where possible streamlits cacheing decorator is used, to reduce loading times when using any widget on the webpage.

Additionally the DBSCAN heuristic described in section 5.5 was also implemented using streamlit and interactive altair charts for displaying the results. All utility functions for the heruistic were packaged into a class and are then called from the script implementing the frontend. The finished page is also hosted on streamlit share [3] and is linked to from the projects webfrontend, when choosing DBSCAN as clustering algorithm.

All of the code base for the project is documented using the Doxygen style and an automatically generated documentation containing detailed description of every class, method and attribute can be accessed through the github projects page [4].

[2] `https://share.streamlit.io/elpelt/datascience1_group42/main/code/web_frontend.py`

[3] `https://share.streamlit.io/elpelt/datascience1_group42/main/code/heuristic_web.py`

[4] `https://elpelt.github.io/datascience1_group42`

6.2 Dependencies

Below all used libraries are described briefly.

6.2.1 pyclustering v0.10.1.2

pyclustering is a data mining library, focusing on clustering algorithms written in C++ and Python by Andrei Novikov [20]. The library contains a wide range of clustering algorithms implemented in Python with an optional C++ core. If possible pyclustering falls back to its C++ implementations utilising its efficiency and runtime benefits.

6.2.2 scikit learn v0.23.2

scikit learn is a wide rangeing data analysis tool kit for python encompassing algorithms not only for clustering but for classification, regression and in general Data Science tools [3].
Our implementations heavily depends on scikit learn implementations of distance measures, clustering algorithms, scoreing, data preparation like standardising and projecting results.

6.2.3 scikit learn extra v0.2.0

scikit learn extra is an extension to scikit learn spanning algorithms that do not satisfy the inclusion criteria of scikit learn [9]. This library is used for its implementation of K-Medoids that is fully compatible with all other scikit learn algorithms.

6.2.4 numpy v1.19.2

numpy is a fundamental library mostly used for its array structure implementing a C++/Fortran like way of saveing, organising and working with data while still being relatively easy to use [22]. Being a dependency of every other package used in this project we naturally use numpy arrays to store and work with our datasets.

6.2.5 matplotlib v3.3.2

matplotlib is a popular python library for generating and plotting various types of graphs [23]. For our project matplotlib is used for creating the graphs

for index comparision.

6.2.6 pandas v1.2.4

pandas is a powerful data analysis tool [24, 25]. It provides an efficient data structure called DataFrame. This structure is used for handling clustering results, storing them in an external csv-file and preparing data for plotting.

6.2.7 seaborn v0.11.0

seaborn is a powerful data visualisation library build on top of pythons matplotlib [26]. It simplyfies plotting by providing predefined templates. The `scatterplot` function is used for visualising the TSNE and PCA projections of the cluster results.

6.2.8 streamlit v0.82.0

streamlit is an easy-to-use library for building web apps [27]. The webfrontend for our application is implemented using streamlit. Additionally the streamlit hosting service is used for serving the webapp [5].

6.2.9 altair v4.1.0

Altair [28] is a data visualisation toolkit for python based upon the vega-lite project [29], implementing a simple way of generating interactive graphs. Plots built in altair can be easily integrated into streamlit and are therefore used when possible, to display results in a interesting fashion.

7 Evaluation Module

7.1 Implementation

All indices are implemented as a class function for the class Indices(). The code is part of the scikit-library ([3]) and expects two inputs for external cluster validation methods and just one input for internal methods. The input

[5] `https://share.streamlit.io/elpelt/datascience1_group42/main/code/web_frontend.py`

contains two arrays, namely the computed cluster labels and the expected cluster labels from the original data or only the second mentioned for internal methods. As output, a numerical value, usually between 0 and 1, is computed, depending on the selected cluster validation index.

7.2 Scoring Methods

7.2.1 Adjusted Rand Index

The Adjusted Rand Index (ARI) is an external cluster validation method with a value between 0 and 1, where a value close to 0 represents a random partition and a value close to 1 represents a nearly identical cluster compared to the original labels of the data. Thus, maximizing this score argues for perfect clustering.
It is defined by: [30]

$$ARI(P^*, P) = \frac{\sum_{i,j} \binom{N_{i,j}}{2} - [\sum_i \binom{N_i}{2} \sum_j \binom{N_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{N_i}{2} + \sum_j \binom{N_j}{2}] - [\sum_i \binom{N_i}{2} \sum_j \binom{N_j}{2}]/\binom{N}{2}} \tag{18}$$

N is the number of data points in the dataset and $N_{i,j}$ describes the number of data points in a class label $C_j^* \in P^*$ associated with cluster C_i in partition P.N_i represents the number of data points in cluster C_i of partition P, while N_j represents the number of data points in class C_j^*. [30]

7.2.2 Completeness Score

The completeness score is another external clustering index. Essentially, the completeness score determines how much items with the same labeling are also put into the same cluster. It is symmetric to the homogeneity score and is defined as following: [31]

$$c = 1 - \frac{H(K|C)}{H(K)}$$

where

$$H(C|K) = -\sum_{c,k} \frac{n_{ck}}{N} \log\left(\frac{n_{ck}}{n_c}\right)$$

Consistent to the homogeneity score, C represents the true cluster labels of the datapoints and K the labels predicted by the clustering algorithm. n_{ck} is

the number of items in a cluster k, which are labeled c, i.e. share the same label. n_c stands for the total number of points labeled c. When all items labeled c are put into one single cluster k, the completeness score is 1.

7.2.3 Homogeneity Score

The homogeneity score is an external clustering index to determine the quality of a calculated clustering in comparison to a preexisting grouping of items. It is defined as following: [31]

$$h = 1 - \frac{H(C|K)}{H(C)}$$

where

$$H(C|K) = -\sum_{c,k} \frac{n_{ck}}{N} \log\left(\frac{n_{ck}}{n_k}\right)$$

Here, C represents the true cluster labels of the datapoints and K represents the labels predicted by the clustering algorithm. n_{ck} is the number of items in a cluster k, which are labeled c, i.e. share the same label. n_k stands for the total number of labels present in cluster k. When each cluster k conains only items with the same label c, the homogeneity score equals one.

7.2.4 Normalized Mutual Index

The Adjusted Mutual Info Score (AMI) is an external cluster valdidation method, in order to calculate the informativeness of a partition computed by a clustering algorithm. It is an adjustment of the Mutual Information (MI) Score to account for chance [3]. This score is upper bounded by 1 and is expected to be 0 for random partitions. The score is calculated as following: [32]

$$AMI_{max}(U,V) = \frac{NMI_{max}(U,V) - E\{NMI_{max}(U,V)\}}{1 - E\{NMI_{max}(U,V)\}} \tag{19}$$

$$= \frac{I(U,V) - E\{I(U,V)\}}{max\{H(U),H(V)\} - E\{I(U,V)\}} \tag{20}$$

NMI stands for Normalized Mutual Index, which is the Mutual Index normalized to values in the range of 0 to 1 [3]. $E\{I(U,V)\}$ is the calculated expected index and $I(U,V)$ the actual Index [32].

8 Web Frontend and User Manual

The web frontend is designed to provide an intuitive exploration of the different datasets, clustering algorithms and distances. In an interactive interface multiple clustering settings can be chosen and a visualisation of the results is directly generated. A cluster-table is used to store previous calculated results, which can be plotted within the evaluation module.
A checkbox at the top of the page (see Figure 2 A), which is set by default, allows the use of precalculated clustering results, which have been computed beforehand (with a random seed value) and are stored in the github repository. This was done for reproduction purposes. The second checkbock (see Figure 2 B) allows the option for interactive projection plots.
The user can choose between four datasets (see Figure 2 C), four distance measures (see Figure 2 E) and four different algorithms (see Figure 2 D). If kmeans, kmedian or kmedoids is chosen a value for the parameter k between 1 and 10 has to be set (see Figure 2 F) The default value for k is 3. If DBSCAN is chosen the user has to define a value for epsilon and the minimal number of nearest points. The parameter settings can be adjusted with an interactive slider widget.

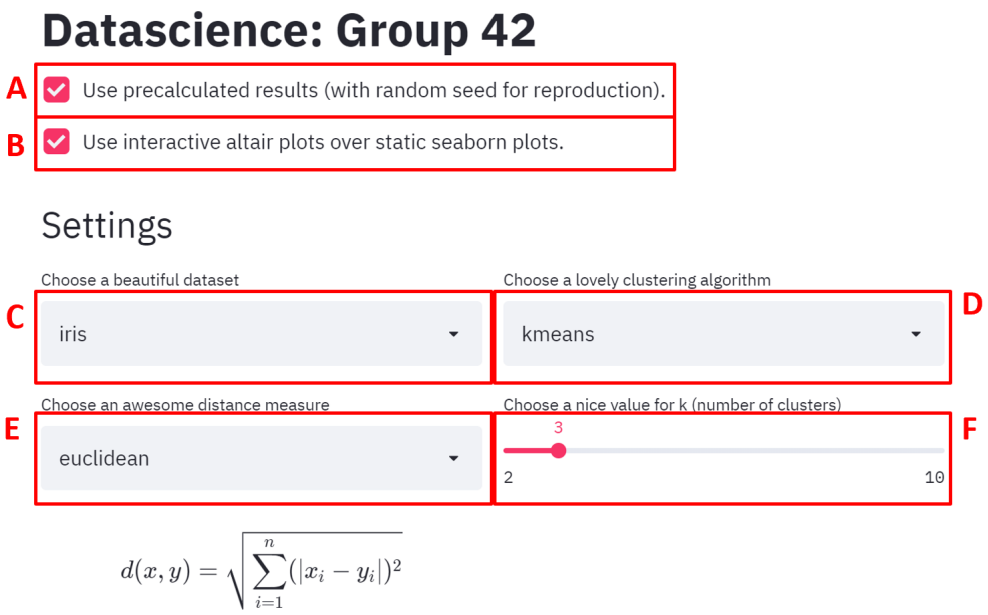


Figure 2: First part of the web frontend. Setting options for clustering parameters.

Moreover the perplexity value for the t-SNE projection can be set individually between 5 and 50 with a slider (see Figure 3 A). The default value is 25. The t-SNE and PCA plots are shown next to each other to allow a direct comparison of the lower-dimensional projections. To virtually interact with the plots, the button shown in Figure 3 C can be clicked. Please note that this option is only available when the checkmark shown in Figure 2 B is set. Furthermore, this button provides the ability to save the plot in different formats. The selected dataset can be viewed in a table format below the plots (see Figure 3 B). The frontend is reloaded entirely if a parameter value is changed, a different setting is made or a button is clicked.

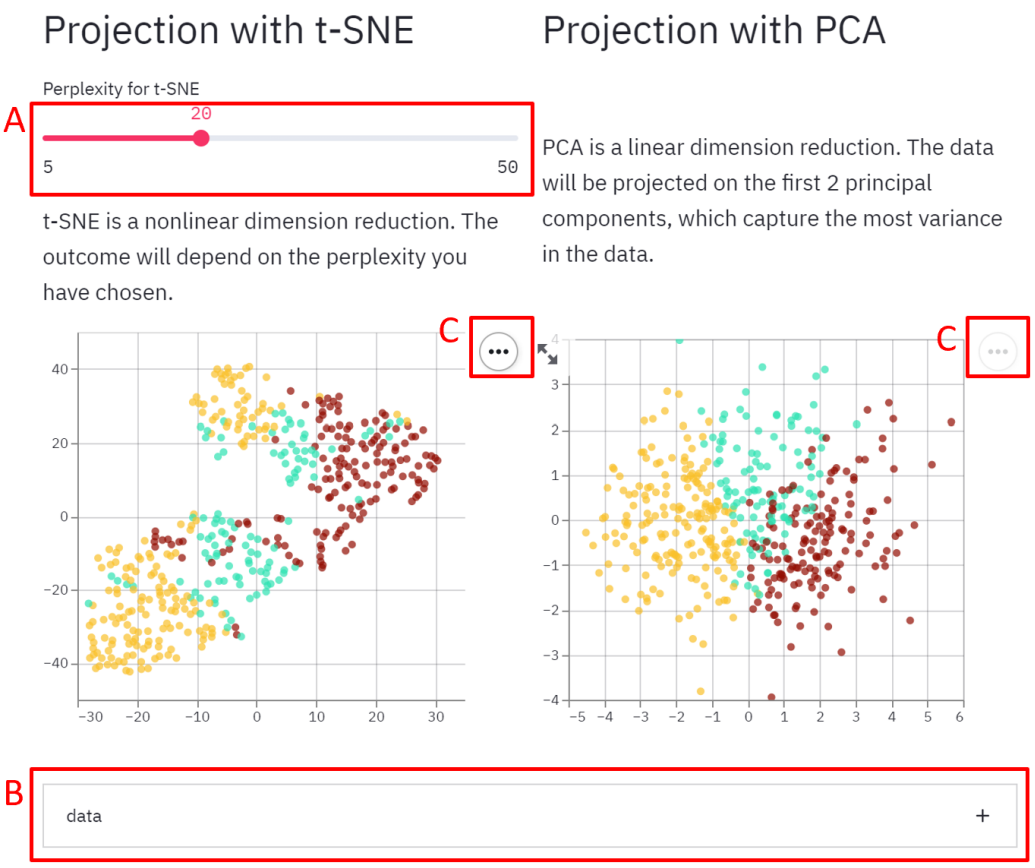


Figure 3: Second part of the web frontend. Projection results of a clustering.

For the evaluation of the clustering results, a clustering index can be chosen as shown in Figure 4 C. Afterwards, the clustering result can be saved through addition to the clustering table with the *add*-button (see Figure 4 A) and the selected index will be calculated. To compare this index to the resulting indices of clusterings with other settings, the *add*-button can be clicked repeatedly after desired adjustment of the clustering settings. To start over and compare further clustering indices, the *reset*-button (see Figure 4 B) can be clicked. The previous display of resulting indices will be cleared as well as the clustering table.

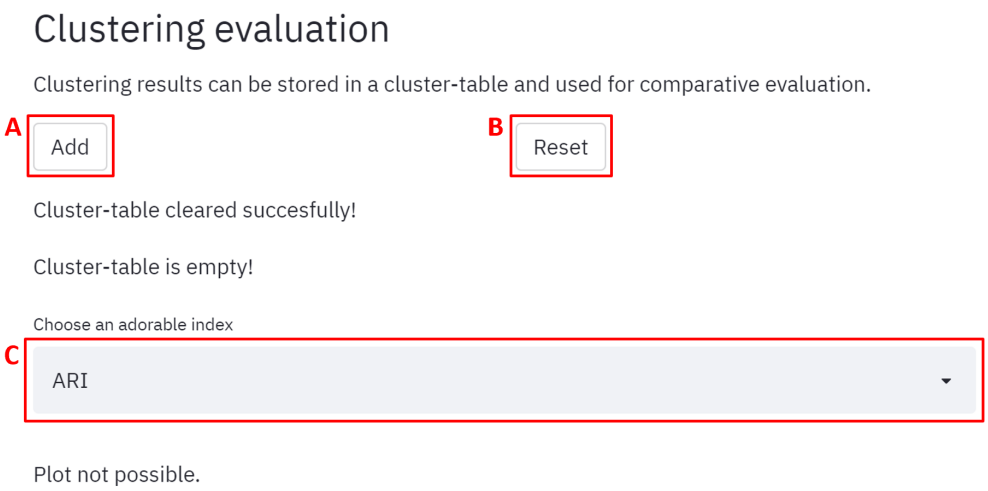


Figure 4: Third part of the web frontend. Evaluation of clustering results.

Ancillary streamlit settings can be found at the upper right corner. Additional explanatory texts provide help and more information.

9 Conclusion

A variety of additional algorithms, distance metrics, parameter settings and cluster evaluation measures could be considered.

We did not found a best clustering algorithm neither a superior distance measure, which would give the best results on every dataset. This problem is widely known as the "No Free Lunch Theorem" [33]. As specific requierements may differ for diverse use cases multiple algorithms and parameters should be explored and compared according to individual needs.

References

[1] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. Definition of a distance measure. In *Mining of Massive Datasets - THIRD EDITION*, page 97. Infolab Stanford EDU, 2020.

[2] Cosine distance, cosine similarity, angular cosine distance, angular cosine similarity, Jul 2017.

[3] 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.

[4] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017.

[5] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[6] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32:407–499, 2004.

[7] Xin Jin and Jiawei Han. *K-Medoids Clustering*, pages 564–565. Springer US, Boston, MA, 2010.

[8] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[9] scikit-learn-extra. `https://github.com/scikit-learn-contrib/scikit-learn-extra`, 2021.

[10] Christopher Whelan, Greg Harrell, and Jin Wang. Understanding the k-medians problem. *Int'l Conf. Scientific Computing*, pages 219–222, 2015.

[11] Sirion Vittayakorn, Sanpawat Kantabutra, and Chularat Tanprasert. The parallel complexities of the k-medians related problems. *IEEE Xplore*, 2008.

[12] K-means and k-medians, 2020.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases

with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[14] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.*, 42(3), July 2017.

[15] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[16] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

[17] Mathworks documentation, t-sne, 2021.

[18] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *The Royal Societey Publishing*, April 2016.

[19] Zakaria Jaadi. A step-by-step explanation of principal component analysis (pca), April 2021.

[20] Andrei Novikov. PyClustering: Data Mining Library. *Journal of Open Source Software*, 4(36):1230, apr 2019.

[21] Thiago Teixeira. gist: Sessionstate.py. `https://gist.github.com/tvst/036da038ab3e999a64497f42de966a92`, 2021. Accessed: 10th June 2021.

[22] 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.

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

[24] 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.

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

[26] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[27] streamlit. `https://github.com/streamlit/streamlit`, 2021.

[28] Jacob VanderPlas, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, 3(32):1057, 2018.

[29] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2017.

[30] Yun Yang. *Temporal Data Mining Via Unsupervised Ensemble Learning*. Elsevier, 2017.

[31] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[32] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[33] David Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8, 03 1996.

[34] Moshe Lichman. UCI Machine Learning Repository, 2013.

[35] Harro Heuser. *Lehrbuch der Analysis : Teil 2*. Mathematische Leitfäden. B.G. Teubner, Stuttgart, 11. auflage edition, 2000.

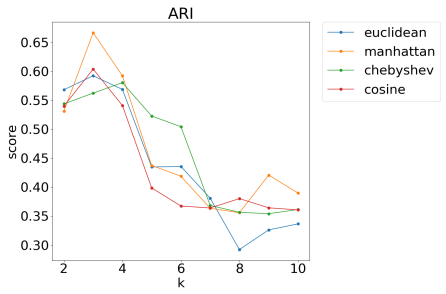
[36] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark 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.

Appendices

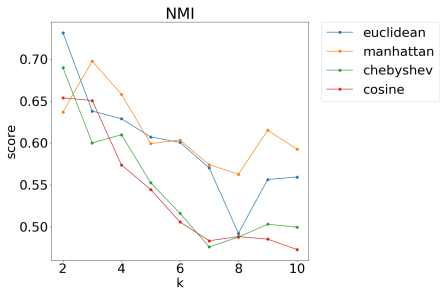
A Plots

A.1 Comparison of different k values

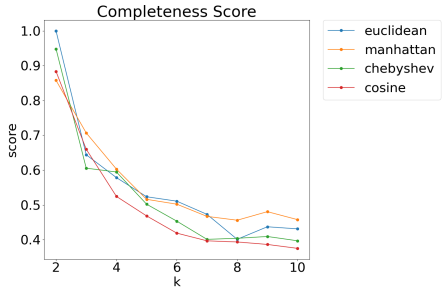
To estimate the number of meaningful clusters (clusterings with highest index scores) in the datasets and test the robustness of different distance measures we compared scorings as a function of k.



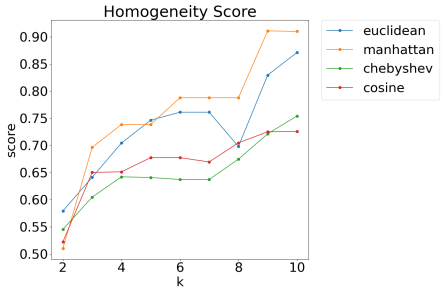
(a) ARI



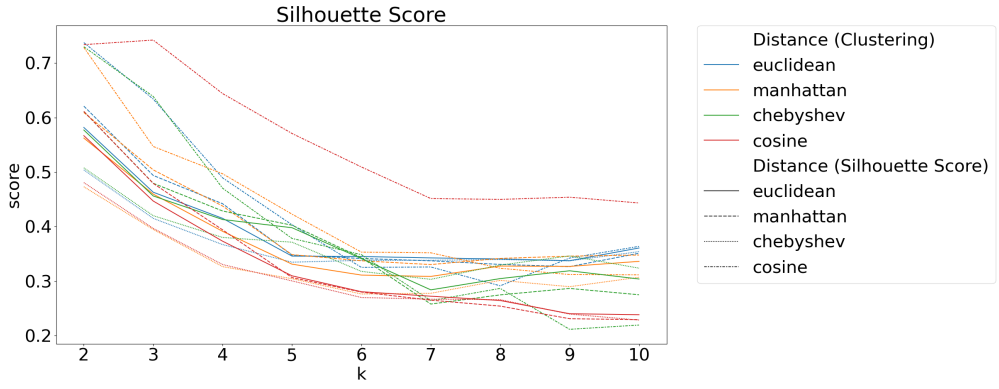
(b) NMI



(c) Completeness Score

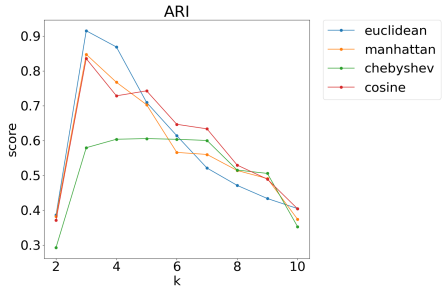


(d) Homogeneity Score

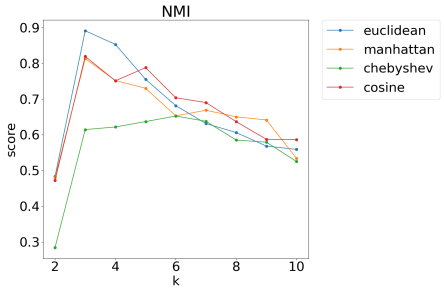


(e) Silhouette Score

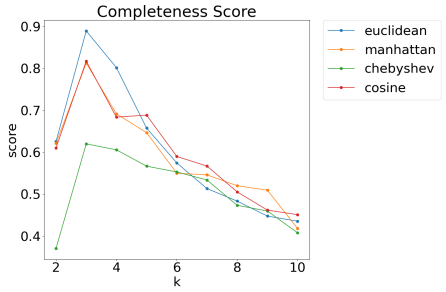
Figure 5: Comparison of clustering scores for kmeans-clustering on Iris dataset



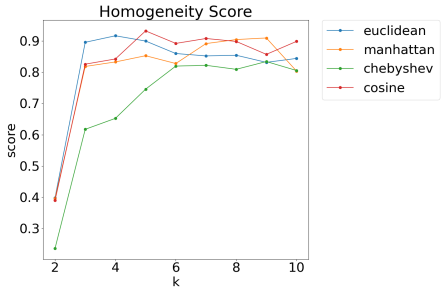
(a) ARI



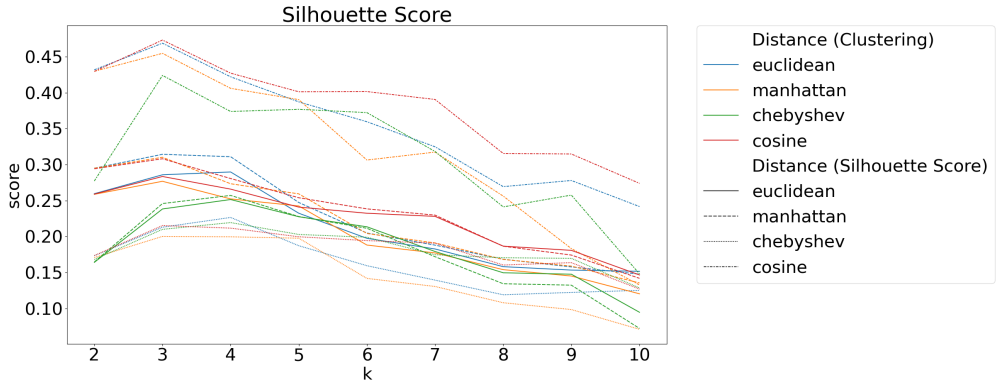
(b) NMI



(c) Completeness Score

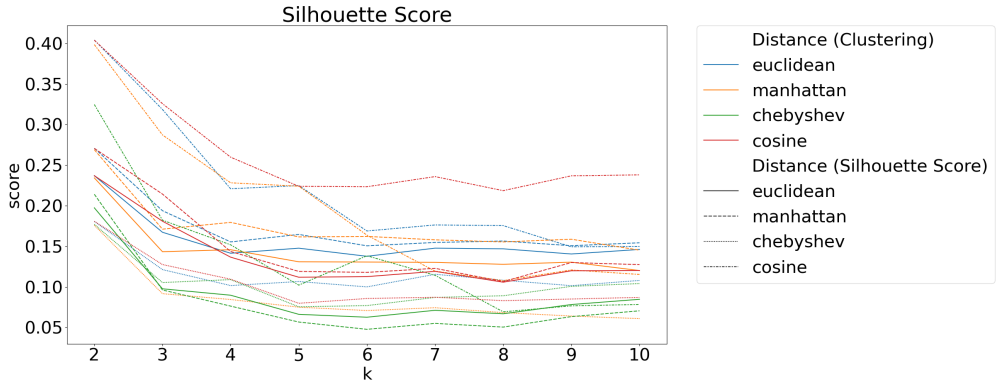


(d) Homogeneity Score



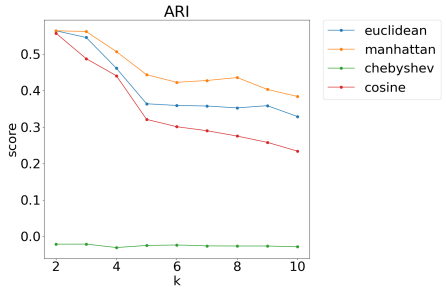
(e) Silhouette Score

Figure 6: Comparison of clustering scores for kmeans-clustering on Wine dataset

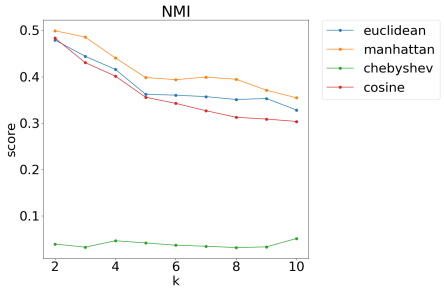


(a) Silhouette Score

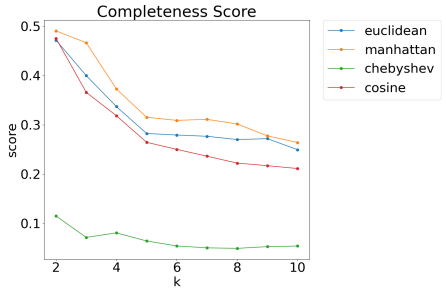
Figure 7: Comparison of clustering scores for kmeans-clustering on Diabetes dataset



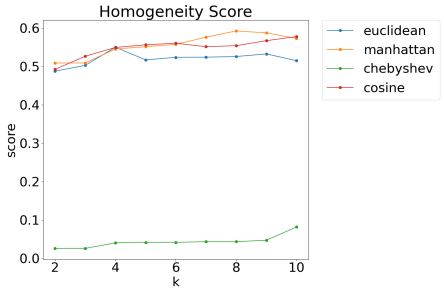
(a) ARI



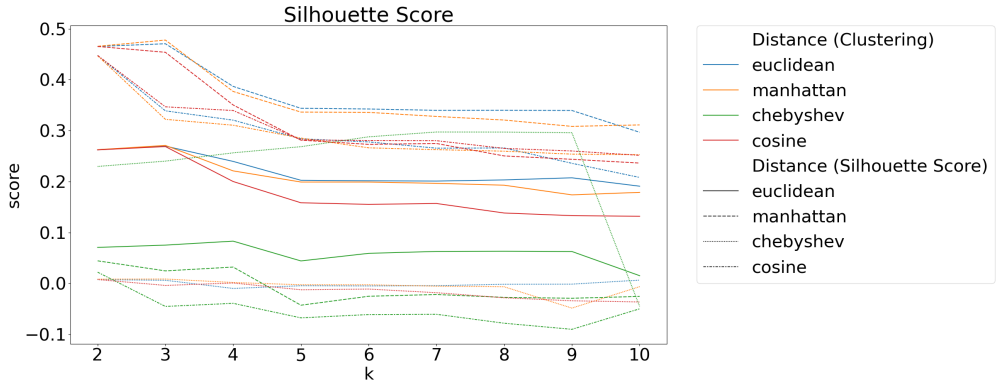
(b) NMI



(c) Completeness Score

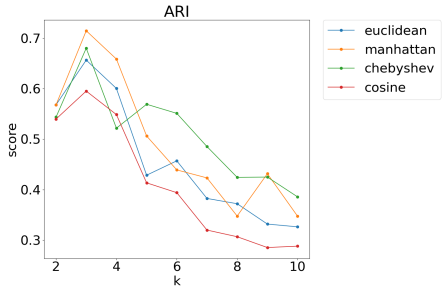


(d) Homogeneity Score

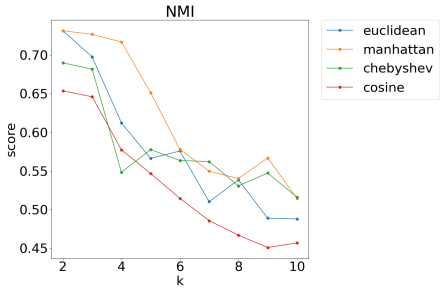


(e) Silhouette Score

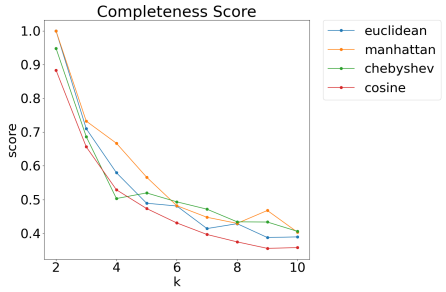
Figure 8: Comparison of clustering scores for kmeans-clustering on Housevotes dataset



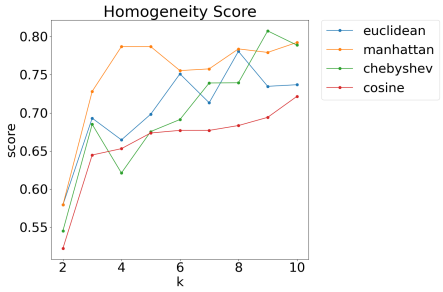
(a) ARI



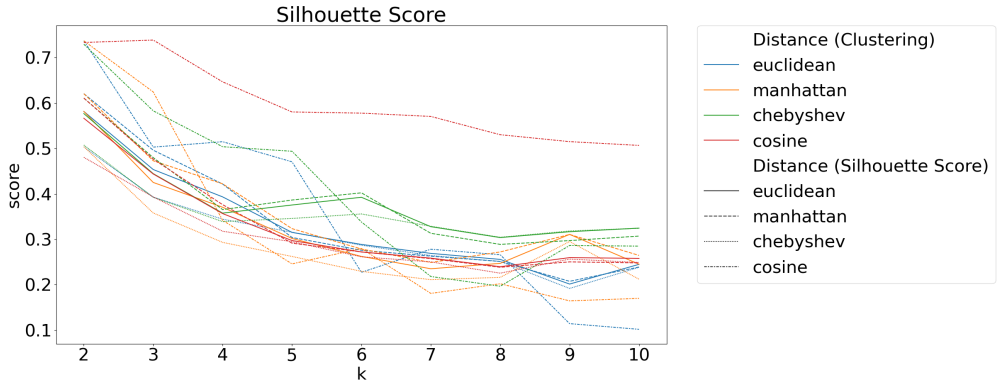
(b) NMI



(c) Completeness Score

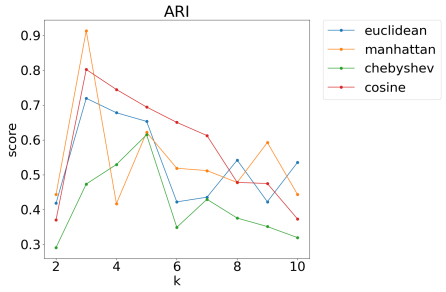


(d) Homogeneity Score

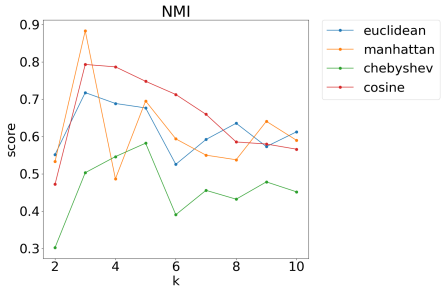


(e) Silhouette Score

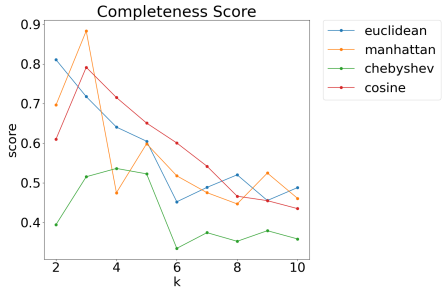
Figure 9: Comparison of clustering scores for kmedians-clustering on Iris dataset



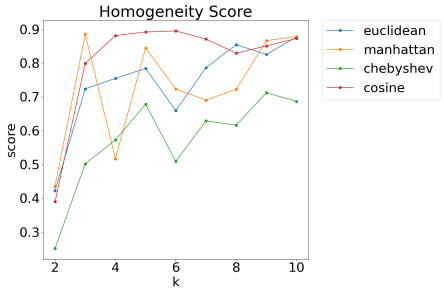
(a) ARI



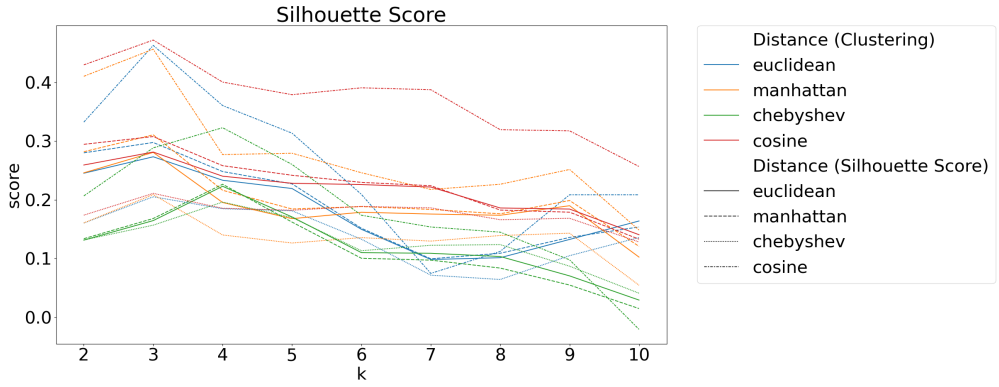
(b) NMI



(c) Completeness Score

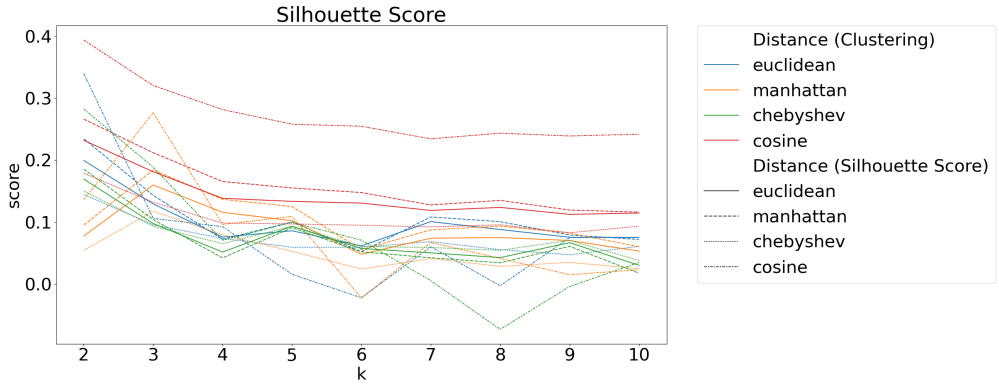


(d) Homogeneity Score



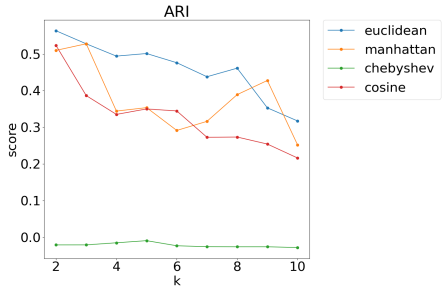
(e) Silhouette Score

Figure 10: Comparison of clustering scores for kmedians-clustering on Wine dataset

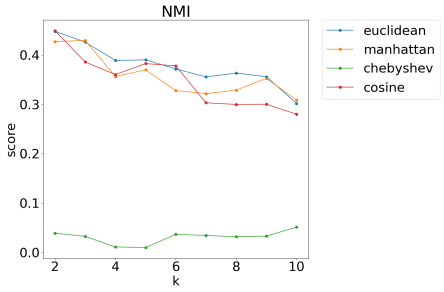


(a) Silhouette Score

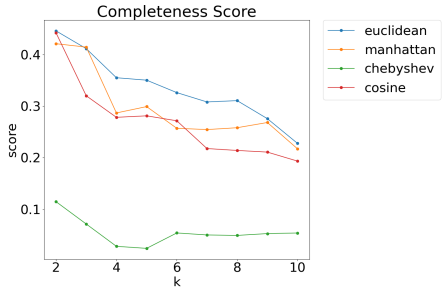
Figure 11: Comparison of clustering scores for kmedians-clustering on Diabetes dataset



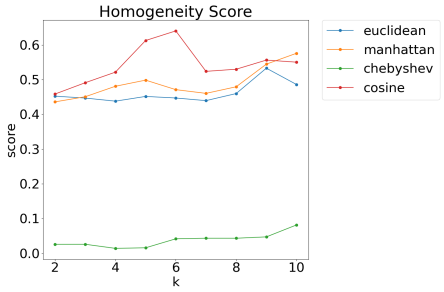
(a) ARI



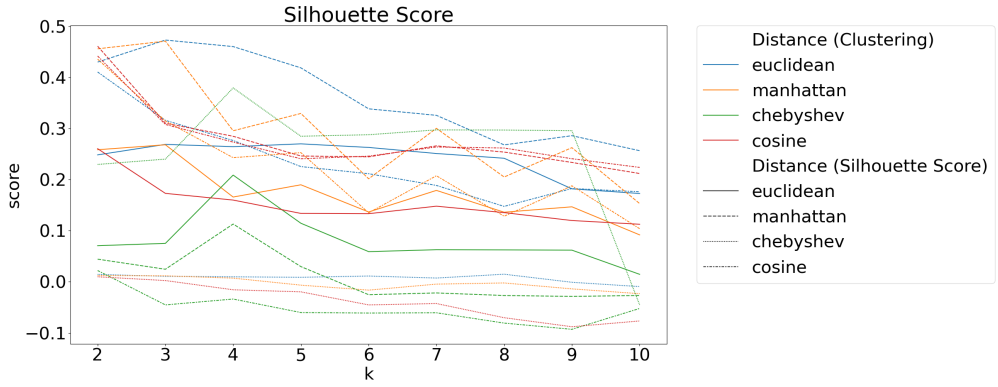
(b) NMI



(c) Completeness Score

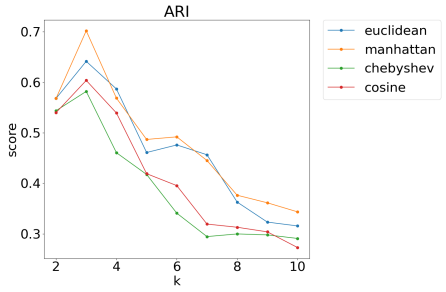


(d) Homogeneity Score

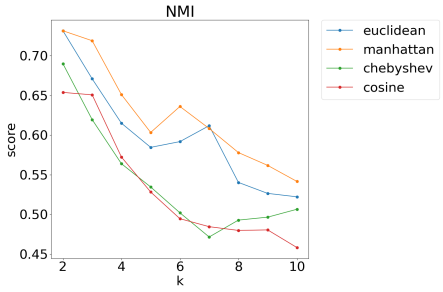


(e) Silhouette Score

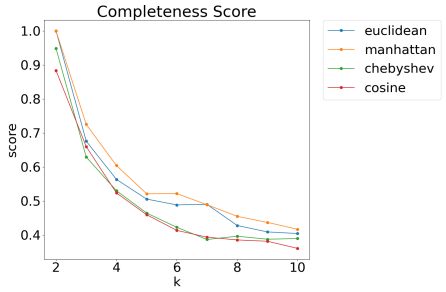
Figure 12: Comparison of clustering scores for kmedians-clustering on House-votes dataset



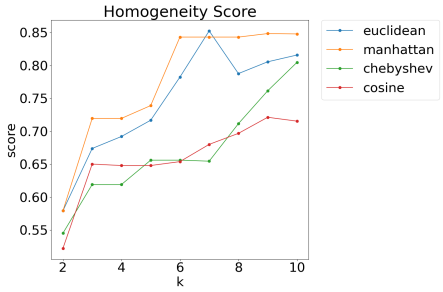
(a) ARI



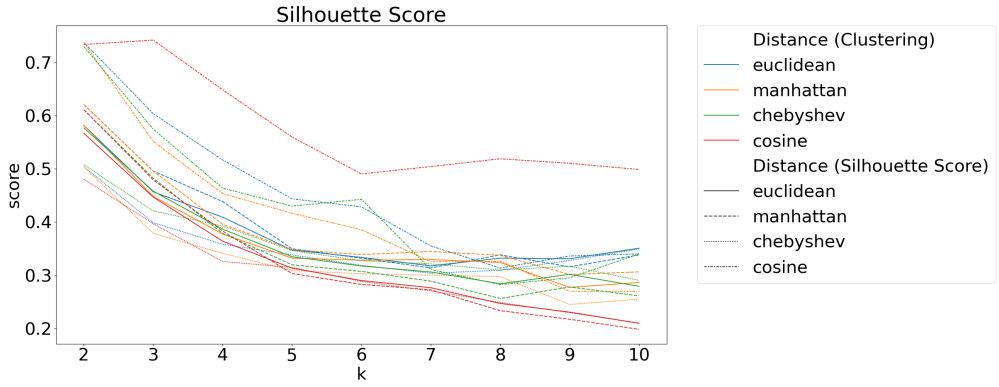
(b) NMI



(c) Completeness Score

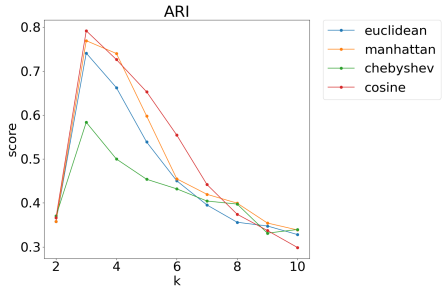


(d) Homogeneity Score

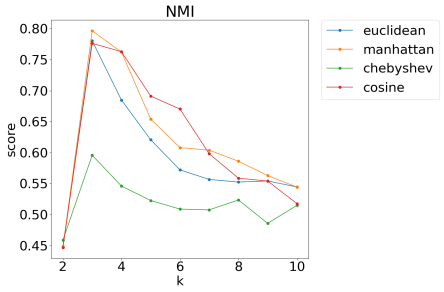


(e) Silhouette Score

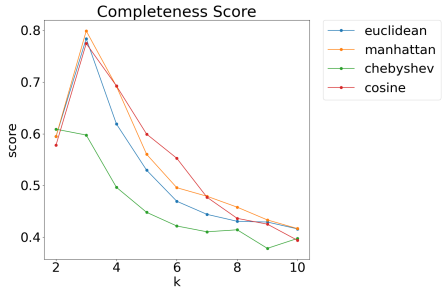
Figure 13: Comparison of clustering scores for kmedoids-clustering on Iris dataset



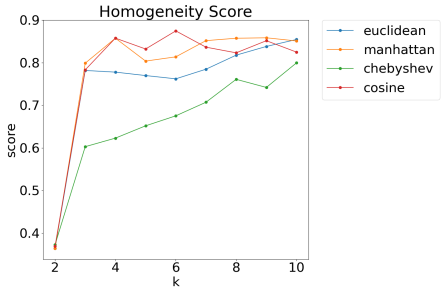
(a) ARI



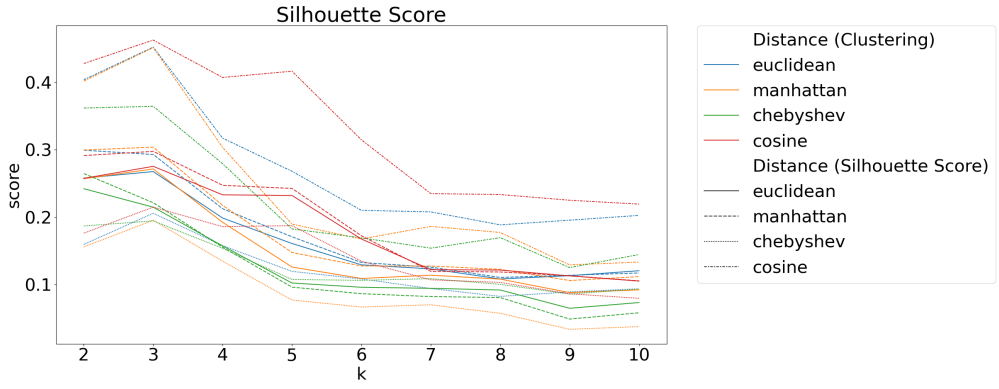
(b) NMI



(c) Completeness Score

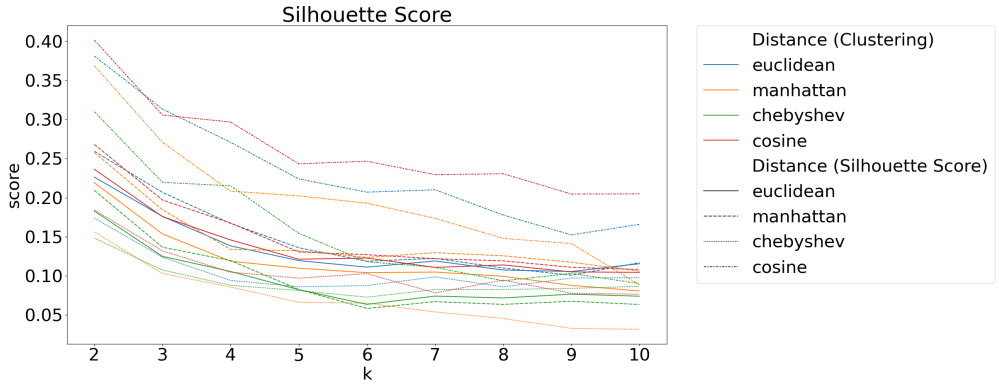


(d) Homogeneity Score



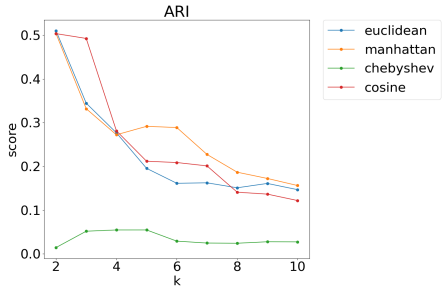
(e) Silhouette Score

Figure 14: Comparison of clustering scores for kmedoids-clustering on Wine dataset

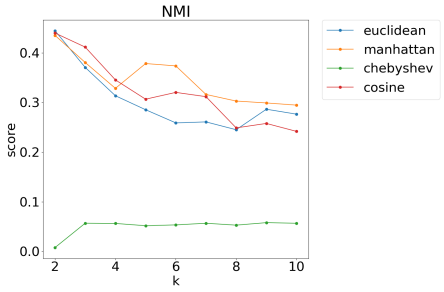


(a) Silhouette Score

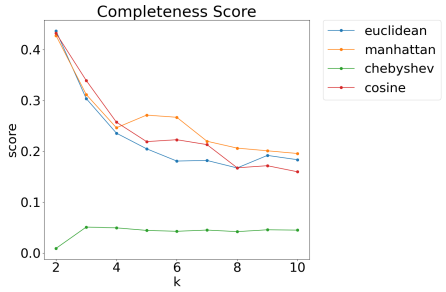
Figure 15: Comparison of clustering scores for kmedoids-clustering on Diabetes dataset



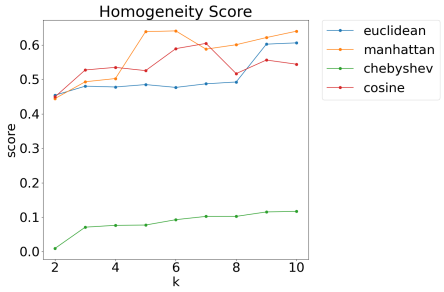
(a) ARI



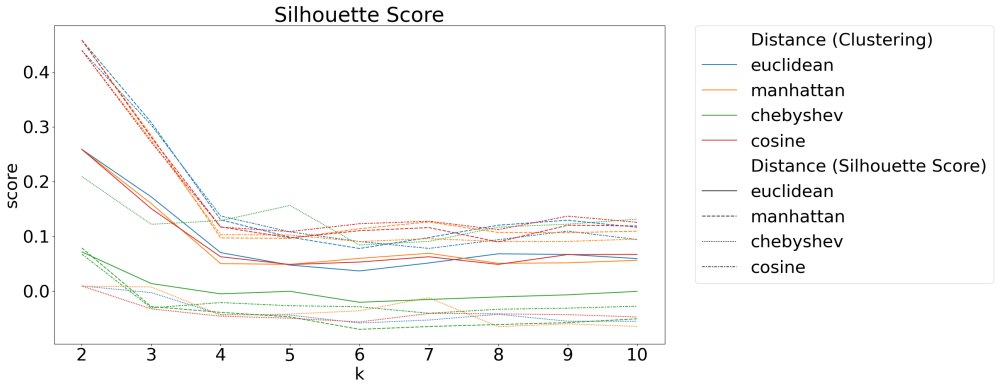
(b) NMI



(c) Completeness Score



(d) Homogeneity Score

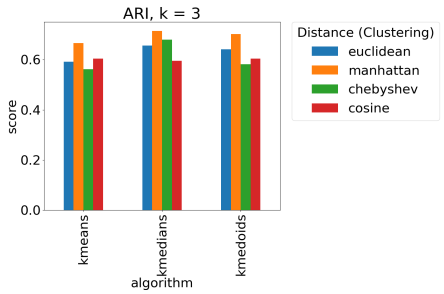


(e) Silhouette Score

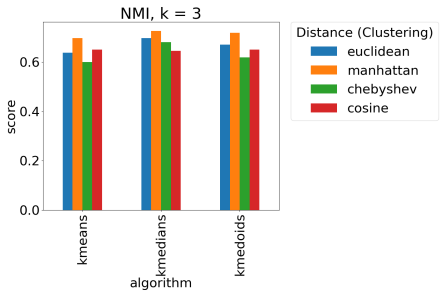
Figure 16: Comparison of clustering scores for kmedoids-clustering on House-votes dataset

A.2 Comparison for given k value

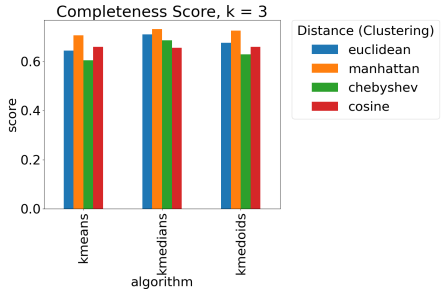
For each dataset we independently compared clustering scores of different algorithms for multiple distances. For the algorithms that require a predefined number of clusters we set the value k to be the number of different classes in the dataset (except for the diabetes dataset, which does not have categorical labels).



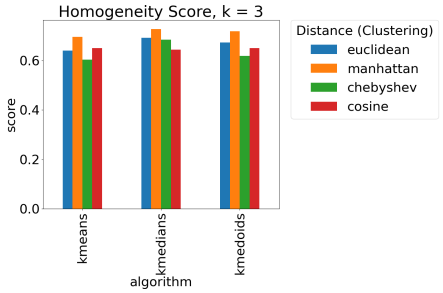
(a) ARI



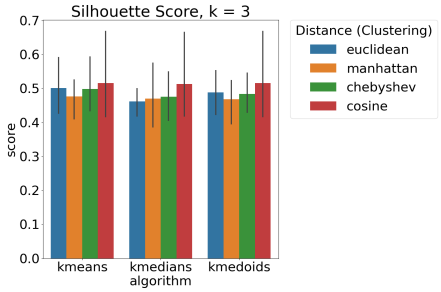
(b) NMI



(c) Completeness Score

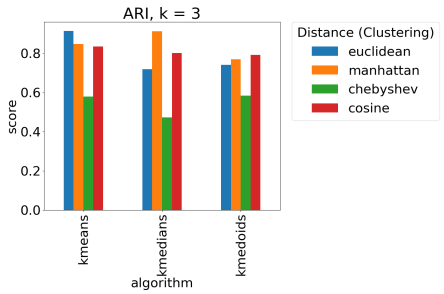


(d) Homogeneity Score

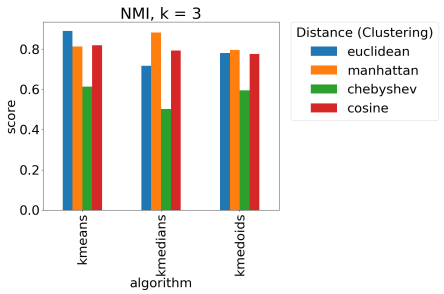


(e) Silhouette Score

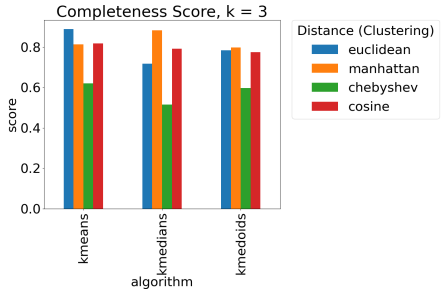
Figure 17: Comparison of clustering scores for Iris dataset (given k=3)



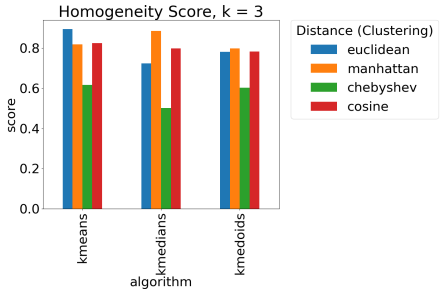
(a) ARI



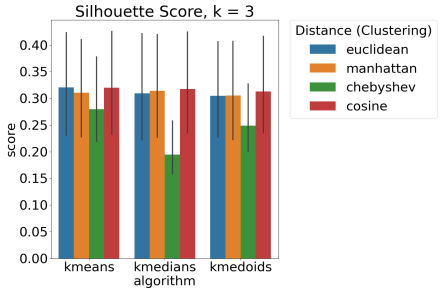
(b) NMI



(c) Completeness Score

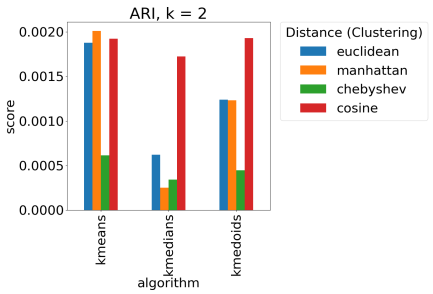


(d) Homogeneity Score

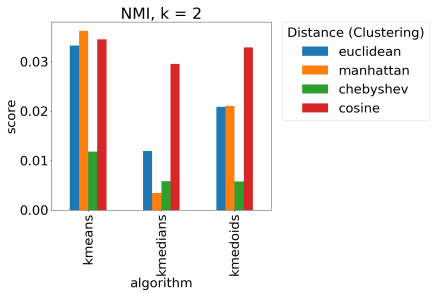


(e) Silhouette Score

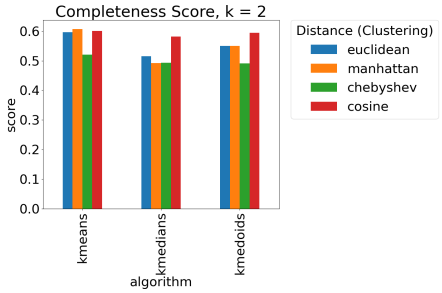
Figure 18: Comparison of clustering scores for Wine dataset (given k=3)



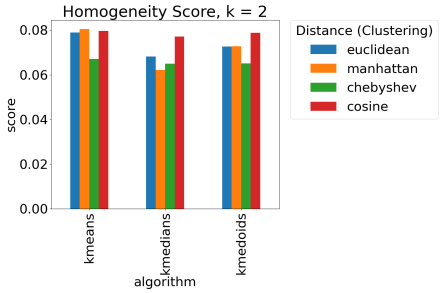
(a) ARI



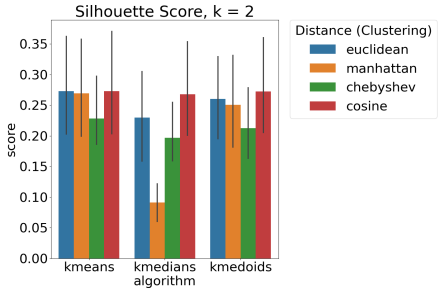
(b) NMI



(c) Completeness Score

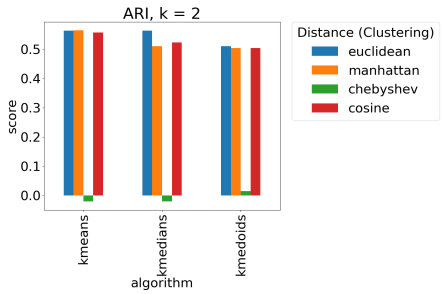


(d) Homogeneity Score

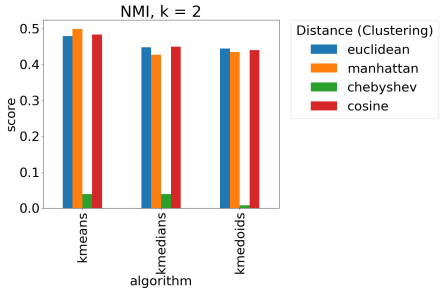


(e) Silhouette Score

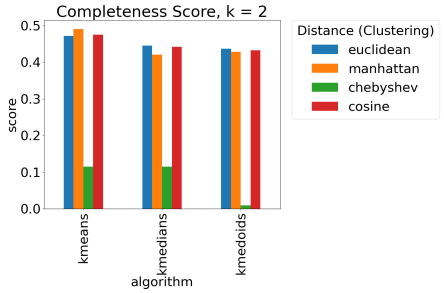
Figure 19: Comparison of clustering scores for Diabetes dataset (given k=2)



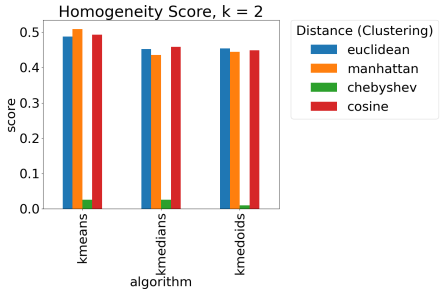
(a) ARI



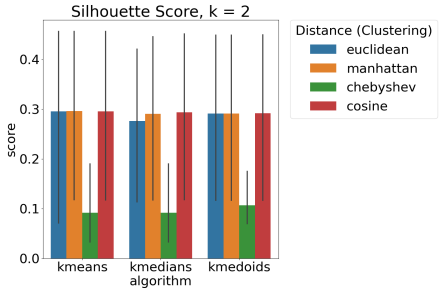
(b) NMI



(c) Completeness Score



(d) Homogeneity Score



(e) Silhouette Score

Figure 20: Comparison of clustering scores for Housevotes dataset (given k=2)