Project 2 - SF2868

A Clustering Approach to Detect Dependencies between Test Cases

Cecilia Battinelli, Magnus Orrsveden, Han Swee Yang

October 2018

Contents

1 Abstract

In this report a clustering approach is used to detect dependencies between a number of test cases. Two main files are provided: the test cases file, where each test case is represented by a 64-dimensional vector (this file was obtained using Doc2Vec algorithm) and the ground truth file, where the real dependencies between test cases are highlighted. After several assumptions are made, especially with respect to transitivity and bidirectionality, the clustering is done using Python and implementing two main clustering algorithms: K-Means and HDBSCAN. Once results are reported, they are evaluated comparing the returned clustering to the ground truth accordingly to a given metrics and evaluation indexes: Precision, Recall and F-Measure.

Both algorithm are implemented several times changing the pattern of parameters in order to get reasonable and fulfilling values for the indexes. Finally, a discussion and analysis of results, comprehensive analysis of the usability of PCA procedure, as well as of indexes' usefulness and results applicability are done.

2 An Introduction: Software testing

In many industries that are using software, the process of bug testing is an essential part. In the testing process, each system component with different properties is supposed to be evaluated in terms of response to different inputs, that it meets the requirements, is working in an acceptable amount of time, is safe and fulfils all the required results of the stakeholders. To test each one of these properties, test cases have to be defined. These test cases then have a specific objective and aim, for example, to test a single path on the program or to find a bug.

The number of possible test cases depends on the size and the complexity of the product/software and can in some cases be practically infinite. This fact requires a good testing strategy to test all the necessary properties without using too much of the available resources. Furthermore, the testing process is often a resource consuming activity and a good strategy can save the company a lot in terms of labour costs. Therefore, prioritising the tests and structure the tests in a clear way is often a very important step of the process.

One important feature to structure the test cases is to look at the dependencies of the individual cases. If two test cases are dependent of each other, they might have to be tested simultaneously or in a specific order. Without the knowledge of the dependencies, some tests are going to fail even though that the component actually works and a lot of time will be wasted trying to fix things that actually work. For instance, by converting the description of a test case into a n-dimensional vector, one can try to find dependencies by looking at the distances between the vectors, i.e. looking at their similarities. To find the dependencies based on the distances, different machine learning methods called clustering might be used.

3 Theoretical Background

In this section, the background of the project will be covered, both theoretically and mathematically.

3.1 Clustering

Clustering is a term that describes different methods used to group data into subgroups or clusters. Within these clusters, one aims for having observations that are quite similar to each other in some aspect, while observations in different clusters are different from each other. The observed data may contain several different features that can be used to cluster the data. For example, if grocery stores are to be clustered, features as store turnover, sales area, closeness to a big city, closeness to competitors and number of employees can be features that can be used to create the clusters.

It is noted that clustering is an unsupervised learning; for instance, the given data that must be clustered are not labeled, i.e. for each n-dimensional vector there is no output associated. As matter of fact, the goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. [1]

There are several different methods that can be used to cluster the data and two commonly used methods with different approaches will be described more in detail, *K-means* and *HDBSCAN*. For the sake of clarity, also *DBSCAN* is described as a requirement to describe *HDBSCAN*.

K-means
K-means is a well-known method used to create K non-overlapping clusters based on the features of the data. The objective of the method is to minimise the variation between the observations within each cluster. The most common approach is to use the within-cluster sum of squared also known as the sum of pairwise squared Euclidean distances between the observations in a cluster. This approach turns the K-means method into a optimisation problem described as:

[1] Machine Learning Mastery - Supervised and Unsupervised Machine Learning Algorithms

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \quad \frac{1}{|C_K|}\sum_{k=1}^{K}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 \tag{1}$$

Where $|C_K|$ is the number of observations in cluster k and p the number of features or dimensions. Furthermore, we have the following identity for each cluster k:

$$\frac{1}{|C_K|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2 \tag{2}$$

where $\bar{x}_{kj}$ is the mean of the samples in cluster k and with feature j. These points are often called *centroids* of the cluster and this equation shows that the objective is to minimize the squared Euclidean distance from each observation to the centroid of the cluster that the observation belongs to.

This optimization problem is computationally difficult to solve but a simplifying algorithm has been developed to solved the optimization problem and at least provide a local optimum. A remark about the first point of the algorithm below is that the initial cluster assignment of the observations to K clusters is either based on a an initial guess or completely random.

The algorithm is described here:[2]

Algorithm 1 K-Means

INPUT: Observation vectors **and** K = clusters
OUTPUT: Each observation vector is assigned to one and only one cluster.
Objective:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \quad \frac{1}{|C_K|}\sum_{k=1}^{K}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2$$

1. Randomly assign a number, from 1 to K, to each observations, These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster centroid.
   (b) Assign each observation to the cluster whose centroid is closest (closest is computed accordingly to Euclidean distance)

It must be highlighted that this method is extremely sensitive to the starting point; as matter of fact, it often wanted a so called "warm" start that guarantees a sufficiently good solution.

DBSCAN
Another, newer and more advanced, clustering method is called DBSCAN (Density Based Spatial Clustering of Applications with Noise). As the acronym indicates, the method is a density based clustering method which also allows data observations to be classified as noise, i.e. without belonging to any cluster. Instead of assigning observations that is closest, in terms of distance, to a centroid as in the K-means method, DBSCAN form clusters based on areas of high density separated by areas of low density. As a result of this, the resulting clusters can have any shape, compared to K-means which results in convex shaped clusters.

To define the method and the algorithm used to implement it, their are several key concepts that have to be defined:

- ***Eps-neighbourhood***: The *Eps-neighbourhood*, denoted $N_{Eps}(p)$, is defined as $N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$, for some distance function $dist(p,q)$, and is describing the distance between p and q.

[2]G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, 8th Edition 2017, Springer

- ***Core point***: An observation inside a cluster is defined as a *core point* if at least a minimum number (MinPts) of points are within the *Eps-neighbourhood* of it
- ***Boarder points***: Each cluster contains *boarder points* that are close to one or more core points without being one themselves
- ***Directly density-reachable***: A point q is said to be *directly density reachable* if it is within the *Eps-neighbourhood* of a core point p.
- ***Density reachable***: A point p is said to be *density reachable* from a point q with respect to Eps and MinPts if there is a chain of points (a path) $p_1, ..., p_n$ where $p_1 = q$ and $p_n = p$ such that p_{i+l} is directly density-reachable from p_i.

With these concepts, each cluster is formed by at least a core point p and all other points (core or boarder) that are density reachable from it. The points that are not reachable from any other point are outliers and are classified as noise, i.e. not belonging to any cluster.[3]

HDBSCAN

DBSCAN is seen as a very good clustering method, that is more robust than K-means as well as the fact that it gives clusters that does not have to be convex. Although, the method may in some cases fall short if the data is very density varying and makes the result differ extensively based on the input parameters. To provide an even better method, DBSCAN has been extended to a hierarchical method, called HDBSCAN.

This method is based on the same ground set of concepts as the DBSCAN and works in a similar way. To describe the differences, we once again, introduce some new concepts:

- ***Core distance***: Denoted $d_{core}(p)$, *core distance* is defined as the distance from an observation d to its MinPts-nearest neighbour.
- ***Mutual reachable distance***: The *core distances* are then used to compute the *Mutual reachable distance* between two points (e.g. p and q, defined as $d_{mreach}(p,q) = max\{dcore(xp), dcore(xq), d(xp, xq)\}$
- ***Mutual reachable graph***: A complete graph, denoted G_{MinPts}, in which the objects in the data set are vertices.

By finding the minimal spanning tree between all observations, where the mutual reachable distance is seen as the weight or cost on the edge between to observations. By including a threshold value that describes the maximum allowed weight, and let this threshold decrease, we will start to disconnect edges and form clusters from the connected components. By letting the weight threshold differ, we can detect on which level observations are connected and when an observation changes from being a core point to an outlier.

Additionally to fact that HDBSCAN is a hierarchical approach, the main difference between DBSCAN and HDBSCAN in practice is the input parameter, m_{clSize}, describing the minimum cluster size, i.e. the number of observation in each cluster. This is added as a parameter in the HDBSCAN algorithm and when an edge is disconnected, the resulting groups will only be labeled as a new true cluster if the number of points are greater than m_{clSize}. By then solving an optimization problem where the objective is to find the most "prominent"clusters, meaning to maximize the stability measure of the cluster (definition left out). In most programs for HDSCAN, one does only need to give a value for m_{clSize} for the algorithm to provide an optimal cluster for the data set.

In the original paper where the method is defined, the algorithm is described in the following way (cited but with our notations).[4]

[3]Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. No. 34. 1996.

[4]Campello, R. J., Moulavi, D., Sander, J., *Density-based clustering based on hierarchical density estimates*, 2013, Springer

Algorithm 2 HDBSCAN

1. Compute the core distance w.r.t. MinPts for all data objects in X.
2. Compute an Minimum spanning tree (MST) of G_{MinPts} , the Mutual Reachability Graph.
3. Extend the MST to obtain MST_{ext}, by adding for each vertex a "self edge" with the core distance of the corresponding object as weight.
4. Extract the HDBSCAN hierarchy as a dendrogram from MST_{ext}:
   (a) For the root of the tree assign all objects the same label (single "cluster").
   (b) Iteratively remove all edges from MSText in decreasing order of weights (in case of ties, edges must be removed simultaneously):
      i. Before each removal, set the dendrogram scale value of the current hierarchical level as the weight of the edge(s) to be removed.
      ii. After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label ("noise").

3.2 Principal Component Analysis - PCA

The PCA procedure is a tool to deal with what is (in the field of machine learning) commonly known as the curse of dimensionality. This refers to different phenomena that occur in high-dimensionality but not in small-dimensionality. For example, what is "close" in high-dimensions might not be "close" in small-dimensions. In other words, computing distance on high-dimensional vectors could lead to "close" vectors which may not be so close in reality.

This procedure aims to reduce dimensionality issues, by using a smaller number of dimensions (features). These dimensions are the principal components and they allow to summarise the high-dimensional set with a smaller number of representative variables that collectively explain most of the variability in the original set. [5]

Furthermore, PCA allows the noise - noise is meant as outliers (non-meaningful data) in this context - to be filtered and helps to get the underlying nature of the high-dimensional data.

This procedure can be implemented in two different ways: either defining how many components (dimensions) are wanted, or defining the percentage of how much variability is wanted to be explained by the principal components. This last one means that if the percentage is 100% then all the components are kept, otherwise if another percentage is introduced than only the most significant components to describe that percentage of variability are kept. In other words, this latter method does not require to define the number of components wanted.

[5] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, 8th Edition 2017, Springer

4 Problem Description & Assumptions

This section would cover the details of the Assignment, as well as some of the assumptions used, when tackling the Assignment.

4.1 Assignment Description

This assignment, assigned by RISE, aims to analyse the test cases and detect underlying dependencies among them. The raw data file includes the total number of test cases (1748 TCs), with each test case being a 64-dimensional vector. The objective is to investigate the best possible algorithm, to establish any linkage between each/multiple test cases. This may be through measuring correlation, covariance, or by measuring vectors' distances between each test case and grouping them into clusters. This algorithm aims to group the test cases using several clustering methods. Clustered data sets would indicate that the test cases in the data are dependent on each other, whereas non-clustered data points would mean the test cases are independent of each other. Later, a comparison to another data set is studied. This other data set shows the actual dependencies between test cases, which is given beforehand. This data set is also known as the 'ground truth'.

This information is provided in two different `csv` files:

- `dependencies.csv`: it provides the 'ground truth', as such:

TC	DependsOn
TC0000	TC0001
TC0000	TC0002
TC0001	TC0002
TC0018	TC0019
TC0018	TC0020
⋮	⋮

Table 1: Sample of Ground Truth

- `doc2vec.csv`: it provides the 64-dimensional test cases, as such:

TC	d_0	d_1	$\cdots$	d_{63}
TC0000	1.3540313243865967	-0.3458509147167206	$\cdots$	-0.526816725730896
TC0001	1.1611902713775635	-0.2490767240524292	$\cdots$	-0.6400560736656189
TC0002	1.0486541986465454	-0.11896729469299316	$\cdots$	-0.48657703399658203
⋮	⋮	⋮		⋮

Table 2: High dimensional vectors description

Input data - Vectors of Test Cases

The initial raw data, of 64 dimensional vectors, each representing a test case, is derived from the following:

1) The documents, listing out the instructions for each test case, are collated, and these documents are usually in a single language, such as English, to facilitate the later process.

2) The stack of documents would be processed via machine learning, using unsupervised processes. These processes might include recognition or identification of phrases, [6] or entire paragraphs of text

[6] Tomas Mikolov (2015). *Distributed Representations of Words and Phrases and their Compositionality.* Retrieved 28 September, 2018

[7] within the documents.

3) Such as process would then assign vectors to the documents given. the amount of dimensions may be specified during the process. All of the vectors, each representing one test case, would be compiled into one data sheet.

The algorithm just described, that is used for this particular assignment, is known as "Doc2Vec" algorithm. Briefly, this algorithm takes as input a large corpus of documents and the output is an n-dimensional vector for each of these documents. As explained in "Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications" article,[8] comparison between documents can be done. Using the corresponding vectors, vectors close to each other should be clustered together because they are similar to each other.

As it can be seen, there is some purpose to 'simplifying' the text documents into numerical vectors. There may be other projects or tasks, ranging from establishing the relationship of a few test cases, to a few thousand - In this particular assignment, there exists 1.748 test cases, for a single train prototype. Using manual analysis to determine the relative dependencies of each test case will take too much time, therefore such processes would be used, in order to convert words and paragraphs into vectors, in an effort to simplify the process of establishing dependencies. These high-dimensional vectors however, would still need to be further processed, to establish clusters/groups between each test cases.

Performance Metric of Results

There would be 4 main metrics that would determine if processed data matches up with the ground truth. Namely,

i) True Positive (TP): This means that the algorithm has clustered the test cases properly, and that the ground truth also shows that such is the case.

ii) False Positive (FP): This indicates that the algorithm has clustered the test cases, but that the ground truth shows that the test cases do not actually have any dependency on each other.

iii) True Negative (TN): This means that the algorithm has not clustered the test cases, and that the ground truth also shows that such is the case.

iv) False Negative (FN): This shows that the ground truth has shown that the test cases are dependent on each other, but the algorithm has failed to pick up on/clustered the data set together.

These evaluation metrics can be summarized as follows:

Metric	Ground Truth	Algorithm Output
TP	Yes	Yes
FP	No	Yes
TN	No	No
FN	Yes	No

Table 3: Metrics' summary. **Yes** means the existence of a dependency. **No** means the non-existence of a dependency.

The performance of the algorithm used will be based upon three main formulae, based on the 4 metrics that are discussed above.

The first one, called *Precision*, measures the percentage of correct clustering results, out of a sum of clustered results from the algorithm. The formula is given as such:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

[7] Andrew M. Dai (2013). *Document Embedding with Paragraph Vectors.* Retrieved 27 September, 2018

[8] Sahar Tahvili, Leo Hatvani, Michael Felderer, Wasif Afzal, Mehrdad Saadatmand, Markus Bohlin, *Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications*

Precision therefore measures the likelihood that a cluster of test cases is correctly identified (TP), against all of the identified test cases from the algorithm (TP+FP). The maximum value of precision is one, and this occurs when there is zero cases of non-clustered test cases being misidentified as in a cluster.

The second performance indicator, called *Recall*, measures the percentage of correct clustering results, out of a sum of actual clustered results from the algorithm. The formula is given as such:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Recall therefore measures the likelihood that our algorithm is able to identify correctly, in this case, clustered test cases (TP), from all actual clustered test cases (TP+FN). In other words, it represents the fraction of correctly detected dependencies among the ones described in the ground truth. The maximum value of recall is one, and this occurs when the is zero cases of actual clustered test cases being misidentified as independent test cases.

The last performance indicator, called *F-Measure* (also known as F_1 *Score*), is a measure that is resulted from the harmonic mean of both precision and recall. Its formula is given as such:

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

F-Measure would consider a balanced influence of Precision and Recall to determine the performance of the algorithm used. Generally, the higher the value of F-Measure, the better the performance of the algorithm in identifying, correctly, the clustered test samples. The maximum possible value that the F-measure can hold is one, when both Precision and Recall values are one. This only occurs when the algorithm is able to correctly identify clustered test cases from independent test cases, without any mistake. This rarely occurs in practicality, so our group would attempt to identify the algorithm used and to process the data, with satisfactory levels of performance, using the performance indicators as described earlier.

4.2 Assumptions of the Assignment/Work

Clusters in the Algorithm can never overlap (for all algorithms)

It is assumed that clusters should not overlap. Each test case would be exactly in one, and only one, cluster group. This should mean, that if there are cases where there exists a long chain of dependencies all the test cases should be grouped together into a single cluster. This is the case of what is depicted in the figure below. Though they are not all directly dependent on each other they are grouped in the same cluster, accordingly to the assignment presentation [9] that requires each dependent pair to be clustered together. In other words - let TCA and TCB be the dependent pair - this would imply by extension that all linked dependecies to both TCA and TCB are included in the clusters. This can be thought as a chain dependency clustering.

In particular, considering the pair TC0110 - TC0111 and their cluster, then also the dependency networks respectively for TC0110 and TC0111 are clustered together.

[9] A Clustering Approach to Detect Dependencies between Test Cases PPT - S. Tahavili

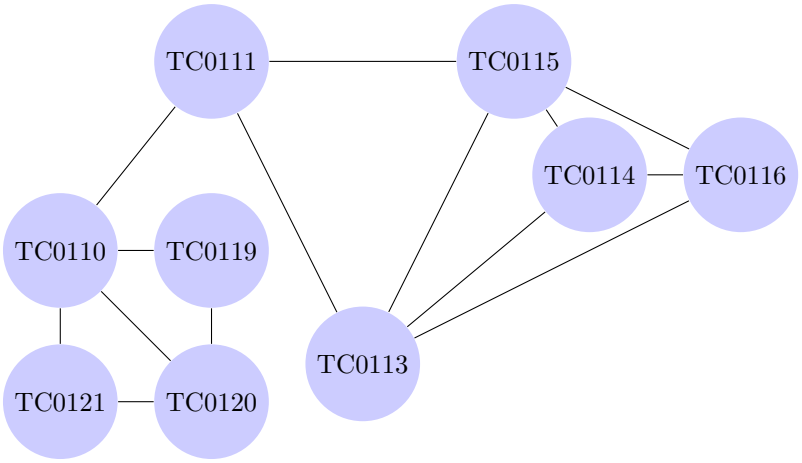


Figure 1: Example of a long-chain cluster extrapolated from the ground truth

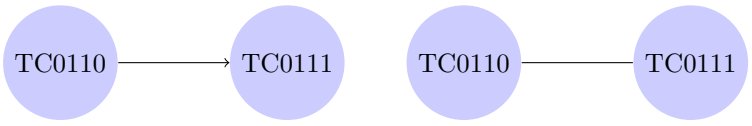
It can be noted that the arrows in the cluster are non-directional, it means that the dependency is valid on both directions, as better explained in next assumption. Finally, this assumptions will be enhanced by the transitivity assumption later on.

Bidirectional Dependency

As briefly described in the earlier assumption, it is assumed that the "being dependent on" feature is valid on both directions. What is described in the ground truth represents only one direction, but to get a complete understanding of the problem the dependency must be bidirectional and extends to the other direction.

For instance, referring to the Figure above, the arrow between TC0110 and TC0111 means that TC0110 depends on TC0111 as well as TC0111 depends on TC0110.

This can be represented as:



((a)) Directional ((b)) Bidirectional

Figure 2: Equivalent representations

However, our group acknowledges that bidirectional dependency may not always be the case in real life. For instance, consider the case of checking the functionality of the hydraulics of the braking system, and checking the functionality of the braking system whilst travelling at a certain speed. It is immediately obvious that the second case would be directly dependent on the first case, but the first case is not dependent on the second. Neither the algorithm nor the ground truth would indicate the direction of dependency of test cases, hence it must be known that after applying the algorithm (in real life test cases), additional steps must be taken to establish the direction of dependency between test cases.

This would also allow for scheduling of test cases to occur in precedence, rather than concurrence. Using the same example in the previous paragraph, it would be possible to test the hydraulics of the braking system first, then proceed with the checking of functionality of the braking system whilst travelling at speed. This would differ from another (example) test case, for instance, to test the hydraulics of the breaking system whilst travelling at speed. Now, it can be seen that these test cases (for travelling at speed) are bidirectional, and has to be tested concurrently instead.

Transitive Dependency between Clusters in the Ground Truth is Assumed

From the Ground Truth data, it can be seen that several clusters are formed, consisting of test cases. Some of these clusters have all test cases being directly dependent on all other elements in the clusters, but there are also instances where test cases in the cluster are only directly linked to one or few other elements in the cluster.

Consider the test cases of TC0059, TC0060, TC0061, and TC0062. In the ground truth, the first three test cases are directly dependent on test case TC0062, but are regardless not directly related to each other, otherwise, as shown in the figure below. In this scenario, since the chosen algorithms only assign test cases into clusters (but not establish direct relationships within the cluster), for performance analysis, we would also assume transitive dependencies between all elements in the cluster, in the results of the algorithms, as well as in the ground truth. This would mean that, in the case of the ground truth, TC0059 is transitively dependent on TC0060, and likewise for TC0060 and TC0061, as shown below.

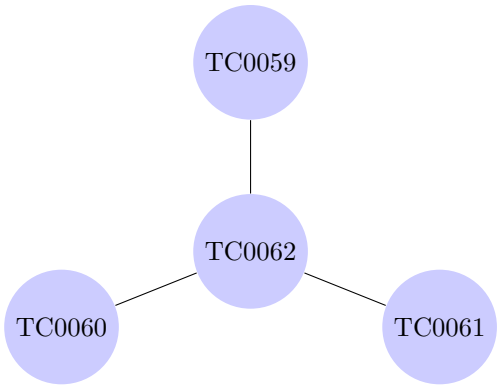


Figure 3: Example of a cluster where the TC do not depend all on each other from Ground Truth

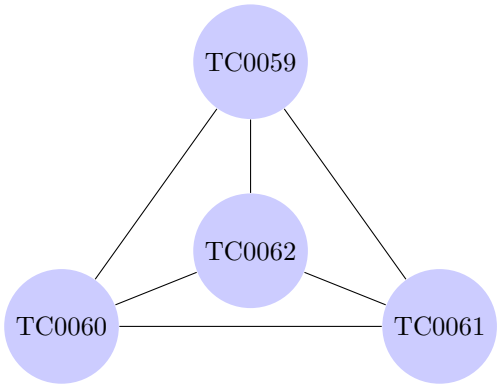


Figure 4: Example of the same cluster according to transitivity dependency assumption

This assumption is fundamental when it comes to analysis and evaluation - through the evaluation metrics - the results. In fact, it is assumed that all the test cases clustered together will have dependencies on each other either directly or by transitive property.

Again, for the sake of reality, in actual testing there would be further steps to determine the direct dependencies between test cases in each cluster.

Test Cases must be in at least one cluster (for K-Means Algorithm)

As stated in the first assumption, each test case is assigned into one and only one cluster. Indeed, there would be no instances where a test case would not be not allocated into clusters. This is due to the inherent process of algorithm clustering, such as K-means, that joins up one observation (test case) into others. There can not be any 'independent' or unique test cases not assigned to any cluster (in the case of K-means, where the number of clusters must be inputted in initial steps). This would mean that the algorithm would assign test cases into clusters, regardless of the actual dependencies on each other.

This would also mean that, should K-Means be used as an algorithm, further steps must be taken to establish unique/independent test cases, if any, from the clustering.
Alternatively, it may also be argued that clusters with only one test case may be interpreted as an independent test case. However, the algorithm for identifying True Negatives may differ, compared to when using other clustering methods.

5 Method

Two different clustering algorithms to process the data are implemented, of which there might be some intermediary processes required. The intermediary process is called Principal Component Analysis (PCA). The two clustering algorithms that are intended to be used would be K-Means Clustering, and the HDBSCAN algorithm.

A special insight concerns HDBSCAN and why it was chosen. This is a fairly recent algorithm able to manage outliers and non-clusterable data. This represents a main useful feature in this assignment since it allows to cluster all together the "independent" test cases. For instance, as in detailed described in the *Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications* [10], this algorithm detect dependencies evaluating the closeness of two vectors and identify a vector as non-clusterable the vector is relatively far from all the other vectors. A non-clusterable vector is then labeled as independent. The resulting cluster of non-clusterable test cases will be labelled as -1 in the results, and all elements in this cluster has to be considered as independent.

5.1 Interpretation of the Data, for Performance Analysis

The data from the following algorithms would be interpreted as such:

1) Format the ground truth data `dependencies.csv` to get a new `csv` file named `GT.csv` where test cases are grouped together into clusters, based on the direct dependencies.

2) Establish the assumption that all element in the cluster have transitive dependencies on every other element in the cluster, if there is no direct dependency already existing between the elements in the cluster. As a result, the `GT.csv` is such that each line corresponds to a cluster and all the test cases in a line are dependent on each other.

3) Compute a list `Indep` where all the test cases independent from everything else are grouped. For instance, test cases such as "TC0004" and "TC0005" belong to `Indep`, whereas test cases such as "TC0000" and "TC0018" do not belong to it, because they depend on some test cases even though they are independent from each other.

4) Obtain the results of the clustering algorithm - if needed, use the PCA procedure.

5) The results obtained should have test cases being assigned to clusters. This obtained results are reported in a `Result` matrix such that $R \in \mathbf{R}^{1748\times 2}$. This matrix is such that each row corresponds to a test case where the first element is the test case itself while the second element is the cluster it belongs to.

6) Compare the results (from step 3) to the data obtained from the raw data (from step 2) according to the evaluation metrics.

The implementation of how the validation was computed is here described through the algorithms used.

[10] Sahar Tahvili, Leo Hatvani, Michael Felderer, Wasif Afzal, Mehrdad Saadatmand, Markus Bohlin, *Cluster-Based Test Scheduling Strategies Using Semantic Relationships between Test Specifications*, 5th International Workshop on Requirements Engineering and Testing (RET'18), Sweden, pages 1-4, 2018, ACM.

K-Means Performance Analysis

These algorithms are valid for K-Means implementations.

Algorithm 3 Detection of TP and FN pairwise instances

```
Data: Results matrix and formatted Ground Truth GT.csv
Result: TP and FN
initialization to 0 for TP and FN
 for every TC do
    for every line in GT.csv do
        if TC is in line then
            for every element in line do
                if element != TC and element > TC then
                    if they have same label in Results then
                        TP += 1
                    else
                        FN += 1
                    end
                end
            end
        end
    end
end
```

The idea is to go through all the dependencies in the Ground Truth formatted file and check for each test case that appears in the file if it is clustered together to the test cases it should depend on, hence if it has the same label as the ones that are in the same line. If two test cases that are dependent in the Ground Truth formatted file have the same cluster label, then the counter for TP is increased by one; in other words, it increases by one for every correctly detected dependent pair. Otherwise, the counter for FN is increased by one.

Algorithm 4 Detection of FP pairwise instances

Data: Results matrix & formatted Ground Truth `GT.csv` & `Indep` list
Result: FP
initialisation to 0 for FP
for *every pair of TCs i,j* **do**
  if $j > i$ **then**
    if *i,j have same label in Result* **then**
      for *each line in* `GT.csv` **do**
        if *i is in the line but j not* **then**
          FP += 1
        end
      end
    end
  end
end
for *every pair of TCs i,j* **do**
  if $j > i$ **then**
    if *i,j have same label in Result* **then**
      if *i & j are both in Indep* **then**
        FP += 1
      end
    end
  end
end

When it comes to checking the FP two different cases must be taken into account. The first case concerns the test cases in the Ground Truth formatted file. In particular, all the test cases in the file should be independent from the test cases in the file who are not in the same cluster, for example TC0000 must be independent of TC0018. If this is not the case,thus if they are labelled the same, the counter for FP is increased by one. The second case concerns all the test cases that are not in the Ground Truth formatted file, hence that are expected to be independent from every other test case, Each pair of these test cases is checked and if they have the same label, thus if they are clustered together, the counter for FP is increased by one.

HDBSCAN Performance Analysis

These algorithms are specific for HDBSCAN and are obtained by extension from the ones valid for K-Means.

Algorithm 5 Detection of TP and FN pairwise instances

Data: Results matrix and formatted Ground Truth `GT.csv`
Result: TP and FN
initialization to 0 for TP and FN
for *every TC* **do**
  for *every line in* `GT.csv` **do**
    if *TC is in line* **then**
      for *every element in line* **do**
        if *element != TC and element > TC* **then**
          if *they have same label in Results different from -1* **then**
            TP += 1
          else
            FN += 1
          end
        end
      end
    end
  end
end

Algorithm 6 Detection of FP pairwise instances

Data: Results matrix & formatted Ground Truth `GT.csv` & `Indep` list
Result: FP
initialisation to 0 for FP
for *every pair of TCs i,j* **do**
  if $j > i$ **then**
    if *i,j have same label in Result & this label is not -1* **then**
      for *each line in* `GT.csv` **do**
        if *i is in the line but j not* **then**
          FP += 1
        end
      end
    end
  end
end
for *every pair of TCs i,j* **do**
  if $j > i$ **then**
    if *i,j have same label in Result & this label is not -1* **then**
      if *i & j are both in Indep* **then**
        FP += 1
      end
    end
  end
end

One must pay attention in algorithm 5 when checking if the test cases that are supposed to be dependent(i.e. in the same cluster) have the same label. It must be excluded the case when they are clustered together but in cluster -1, because being clustered in cluster -1 implies that the test case is regarded independent of all the others.

Another remark concerns the algorithm 6 when the if condition checks that the test cases are not labelled -1. This is because HDBSCAN is able to detect non-clusterable, i.e. independent, test cases. If a test case is detected as independent then it is clustered in a cluster whose label is -1; for instance, the test cases within this cluster are NOT dependent on each other. If this condition is missed, the amount of FP would considerably increase, hence distorting final indexes.

Considerations about Performance Analysis

A remark must be highlighted with respect to the $>$ condition for test cases in both algorithm. This is used to take into account that, as the dependency is bidirectional, the feature of being for example TP is referred to the pair, hence it must be counted only once. In other words if the dependency between $TC0000$ and $TC0001$ is correctly detected, the TP variable must only increase by one.

It must be observed that the sum of TP and FN instances is known in advance. This is because both labels refers to the *being dependent* feature. Indeed, this feature is known by the ground truth and revisited by the transitivity assumption. As a result, given the formatted ground truth introduced above, the sum of TP and FN is such that:

$$TP + FN = \sum_{i=1}^{NR} \binom{n_i}{2} = 1553$$

where NR is the number of clusters identified by the ground truth with the exception of the independent cluster and n_i is the amount of test cases in the i-th cluster. Each addend computes the number of possible non-ordered combinations without repetition. For example, if a cluster has test cases A, B, C then the possible dependencies are AB, BC and AC. This prevents double counting because, as assumed, the dependencies are bidirectional (AB is the same as BA) and meaningless dependencies as AA are not counted as well.

Moreover, TN instances do not need explicit computation. This is because, it can be deduced by taking the difference, all the other instances (TP, FP, FN), from the maximum number of pairs. In particular, let N be the number of total pairs, then it follows that

$$TN = N - (TP + FN + FP)$$

The number N can be computed by basic combinatorics as binomial coefficient of all test cases:

$$N = \binom{1748}{2} = 1526878$$

Finally, it must be highlighted that the amount of TN is not relevant for the purpose of indexes computations.

5.2 PCA Procedure

Whenever PCA procedure is used, the choice has been to maintain an 80% variability among the principal components. This is because, choosing an arbitrary number of components to maintain was pointless, and that choosing a too large percentage would have not be so efficient. Therefore, the choice of 80% variability represents a good compromise between variability, reduction and efficiency.

As a result, the cumulative variance for these data components is illustrated in the Figure below.

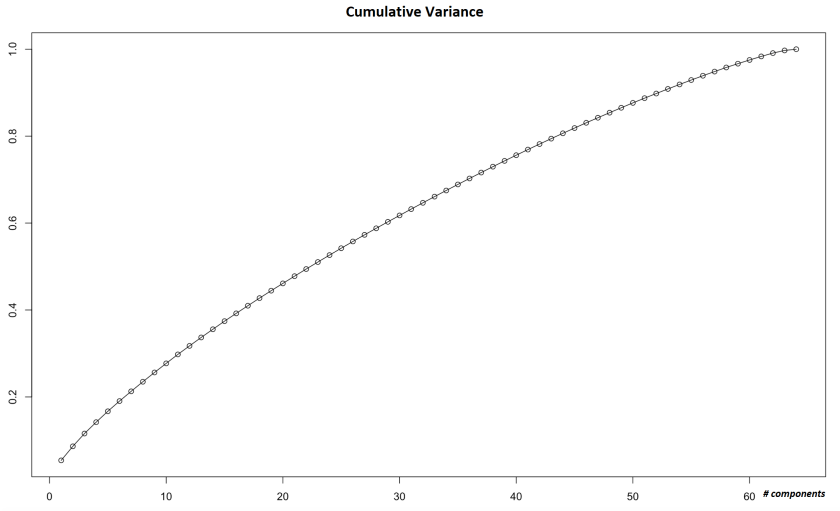


Figure 5: Cumulative Variance

Since the figure shows the cumulative variance for each one of the principal components it, as expected, monotonically increasing between 0% variance and 100%. In Figure 5 one can see that in this case, the variance increasing rate is almost linear. Usually, the cumulative variance has more of a square root increasing rate which means that the first variables are expected to describe almost all the variability. In this case, since the increasing rate is more linear, the previous effect is not happening. For instance, to get a large variability and still perform a reduction, several components are needed. This clearly depicts that to guarantee an 80% of variance, 44 components are needed. This means that to be able to explain 80% of the variance, approximately 69% of the components are needed. It is noted that this happens because the given data was - probably - already processed with respect to dimensionality reduction.

5.3 Software

Given all this, Python is regarded as suitable software. This is because Python is very flexible with respect to parameters' choice in HDBSCAN algorithm. This played a fundamental role to the purpose of this assignment. In fact, this allows defining numerous parameters' patterns for HDBSCAN that helped in the choice of best one, with respect to indexes.

In addition to this, K-Means in Python is performed multiple times and only the best results are reported. This is very beneficial due to sensitivity of initial centroids. This is done changing the `n-init` parameter, that will be from now on set to 10.

More detailed information about the libraries used in Python can be found in Appendix A.

5.4 Algorithms Chosen/Used for the Assignment

K-Means Algorithm, without PCA

The first algorithm that is tested is the K-Means clustering algorithm, without any PCA applied (The technical details can be found in the introduction of this report). A set of different number of clusters for this procedure are used as input, K={400,600,800,1000,1200}.

Analysis of formatted ground truth provides that the number of dependent test cases cluster is around 150. Hence, given that K-Means is not able to cluster all the independent test cases in one cluster but they are necessarily assigned to some specific cluster, it was assumed that there must be necessarily considerably more clusters in K-Means if compared to the amount of clusters provided by HDBSCAN. As a consequence, all the values of K listed above were tested for the sake of clearity and valid comparison.

K-Means Algorithm, with PCA

The second set of algorithms is K-Means, but unlike the first set, our group would experiment with using PCA applied in the raw data, before further processing by K-Means. Again, similar to the first set of algorithms, different numbers of K are used as input, where, K={ 400,600,800,1000,1200 }.

HDBSCAN Algorithm, without PCA

Under the algorithm of HDBSCAN, there exists multiple input parameters, that may affect the clustering algorithm and its results given (Technical details for HDBSCAN can be found in the introduction of this report). The input parameters that can be modified is as follows:

- Distance: Is representing the metric chosen to compute the distance, i.e. the closeness, of two test cases.
- Min Cluster Size: Is representing the minimum number of test cases within a cluster. Naturally, the value would be set to 2 accordingly to the ground truth analysis. However, another value such as 3 was also tested, to see if there is any significant changes, compared to when setting this parameter to 2.
- Cluster Selection Method: this parameter describes "how it selects flat clusters from the cluster tree hierarchy" [11]. It can be set to two different values:
  - `eom` : "Excess of Mass" and it tends to avoid too grained clustering, i.e. picking leaves
  - `leaf`:it selects leaf nodes from the tree, producing many small homogeneous clusters.
- Alpha: it is a tuning parameter for distances. The default value is set to 1.0, but two further values were tested as well, 0.75 and 1.2

An insight to the used distances is illustrated. Given the general Minkowski distance, then several can be derived picking different values of p. The Minkowski distance given two test cases TC such that $TC_1 \in \mathbf{R}^{64}$ and $TC_2 \in \mathbf{R}^{64}$ is:

$$D(TC_1, TC_2) = \Big(\sum_{i=1}^{64} |TC_{1,i} - TC_{2,i}|^p \Big)^{\frac{1}{p}}$$

In particular, if p is set to 1, then the Manhattan metrics is obtained; whereas if p is set to 2 the Euclidean metrics is obtained.

On the other hand, the Canberra distance is described by this formula:

$$D(TC_1, TC_2) = \sum_{i=1}^{64} \frac{|TC_{1,i} - TC_{2,i}|}{|TC_{1,i}| + |TC_{2,i}|}$$

Euclidean, Manhanttan and Minkowski (p=4) are here represented in 2D:

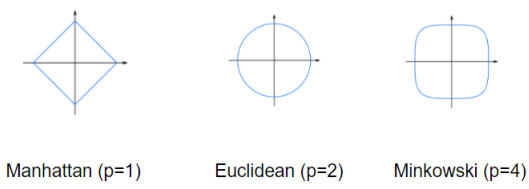


Figure 6: Graphical description of some of the used distances in 2D

[11] HDBSCAN Python Documentation - https://hdbscan.readthedocs.io/en/latest/parameter-selection.html

Several other distances are allowed to be used in HDBSCAN in Python; however, the ones described above were the most suitable since it is assumed to be in a 64-dimensional Euclidean space. For instance, distances - such as Hamming or Rogers Tanimoto - were not suitable because they are, for instance, measuring string similarity (bitwise) or other features different from vectors' distance.

These parameters will be modified to be able to observe the results, and any changes that may arise from the change of parameters.

HDBSCAN Algorithm, with PCA

This set of algorithms would use HDBSCAN Algorithm (Similar to the third case), but unlike the previous scenario, PCA procedure was applied before performing HDBSCAN. The same parameters would be modified as per the third scenario: 'Distance', Minumium Cluster Size, Cluster Selection Method, and 'Alpha' would be modified, again, to measure the effect of PCA, as well as the inputs, that is had on the results of the algorithm.

6 Results

In this section, only the most significant results are reported. The rest of the results may be found in Appendix B. In each table the best F-Measure within the table is highlighted in yellow; the best Precision in pink and finally the degenerate cases in green.

With respect to degenerate cases in HDBSCAN, this is the phenomenon that happens when few (only 1 to 4) clusters are identified and the performance analysis collapse because -obviously - too many FP are detected and the code becomes heavier. There are instances where K-Means can also degenerate. This is the case when a very bad start is chosen and most of the data are clustered in very few clusters among all the clusters. This results - again - in a vertiginous increase of FP.

Furthermore, tables where the amount of TP,FN,FP are reported. The amount of TN is not reported as it is not relevant to the purpose of indexes computations. It would have played a more relevant role if other kind of indexes were introduced, as better described in Discussion section.

6.1 K-Means, without PCA Applied

The results for using the K-Means Algorithm, without PCA applied, are as follows:

K	Recall	Precision	F-measure
400	0,093	0,036	0,052
600	0,156	0,313	0,216
800	0,176	0,009	0,016
1000	0,165	0,007	0,013
1200	0,189	0,008	0,015

K	TP	FN	FP
400	144	1409	3833
600	242	1311	444
800	274	1279	31414
1000	257	1296	38972
1200	293	1260	37343

6.2 K-Means, with PCA Applied

The results for using the K-Means Algorithm, with PCA applied, are as follows:

K	Recall	Precision	F-measure
400	0,086	0,187	0,118
600	0,155	0,310	0,207
800	0,171	0,008	0,016
1000	0,189	0,008	0,015
1200	0,170	0,474	0,250

K	TP	FN	FP
400	134	1419	583
600	241	1312	536
800	266	1287	31443
1000	294	1259	37665
1200	264	1289	293

6.3 HDBSCAN, without PCA Applied - Selected Data

The results for using the HDBSCAN Algorithm, without PCA applied, are as follows:

Distance	Alpha	Min Cluster Size	Cluster Selection Method	Recall	Precision	F-measure
Canberra	0,75 & 1,0	2	`eom`	0,261	0,269	0,265
Canberra	0,75 & 1,0	2	`leaf`	0,223	0,301	0,256
Canberra	0,75	3	`leaf`	0,280	0,252	0,265
Euclidean	0,75 & 1,0	2	`leaf`	0,233	0,449	0,307
Euclidean	0,75	3	`eom`	0,305	0,315	0,310
Euclidean	1,2	3	`eom`	0,957	0,002	0,005
Euclidean	0,75	3	`leaf`	0,290	0,391	0,333
Manhattan	0,75 & 1,0	2	`eom`	0,243	0,323	0,277
Manhattan	1,2	2	`eom`	0,217	0,444	0,292
Manhattan	1,2	2	`leaf`	0,201	0,627	0,304
Manhattan	0,75	3	`eom`	0,296	0,325	0,309
Manhattan	1,0	3	`eom`	0,256	0,394	0,310
Manhattan	0,75	3	`leaf`	0,270	0,405	0,324
Minkowski	0,75 & 1,0	2	`leaf`	0,233	0,449	0,307
Minkowski	0,75	3	`eom`	0,305	0,315	0,310
Minkowski	1,2	3	`eom`	0,957	0,002	0,005
Minkowski	0,75	3	`leaf`	0,290	0,391	0,333

Distance	TP	FN	FP	Clust
Canberra	405	1148	1103	371
Canberra	346	1207	805	387
Canberra	435	1118	1291	179
Euclidean	362	1191	444	340
Euclidean	473	1080	1027	113
Euclidean	1486	67	613809	2
Euclidean	451	1102	703	119
Manhattan	377	1176	789	313
Manhattan	337	1216	422	236
Manhattan	312	1241	186	241
Manhattan	459	1094	955	111
Manhattan	397	1156	610	75
Manhattan	420	1133	616	118
Minkowski	362	1191	444	340
Minkowski	473	1080	1027	113
Minkowski	1486	67	613809	2
Minkowski	451	1102	703	119

6.4 HDBSCAN, with PCA Applied - Selected Data

The results for using the HDBSCAN Algorithm with PCA applied to represent the 80% variability, are as follows:

Distance	Alpha	Min Cluster Size	Cluster Selection Method	Recall	Precision	F-measure
Canberra	0,75 & 1,0 & 1,2	2	`eom`	0,254	0,290	0,271
Canberra	0,75 & 1,0 & 1,2	2	`leaf`	0,227	0,278	0,250
Euclidean	0,75 & 1,0 & 1,2	2	`eom`	0,244	0,377	0,296
Euclidean	0,75 & 1,0 & 1,2	3	`eom`	0,997	0,002	0,003
Manhattan	0,75 & 1,0 & 1,2	2	`eom`	0,238	0,431	0,306
Manhattan	0,75 & 1,0 & 1,2	3	`eom`	0,999	0,002	0,003
Manhattan	0,75 & 1,0 & 1,2	2	`leaf`	0,184	0,391	0,250
Minkowski	0,75 & 1,0 & 1,2	2	`eom`	0,244	0,377	0,296
Minkowski	0,75 & 1,0 & 1,2	3	`eom`	0,997	0,002	0,003

Distance	TP	FN	FP	Clust
Canberra	394	1159	963	380
Canberra	352	1201	914	386
Euclidean	379	1174	626	97
Euclidean	1549	4	882122	3
Manhattan	369	1184	488	93
Manhattan	1552	1	970892	1
Manhattan	285	1268	443	97
Minkowski	379	1174	626	97
Minkowski	1549	4	882122	3

6.5 Short Consideration of Results

As it can be seen from the tables above, if one should pick one algorithm and its corresponding parameters to get the largest F-Measure, one could pick either Euclidean or Minkowski given that `alpha`=0,75, minimum cluster size is 3, cluster selection method is `leaf` and no PCA. This would allow to achieve an F-Measure of $0,33$. Obviously this consideration does not take into account any specific requirement one could ask for, such as specific values for some parameters or specific algorithms.

7 Analysis of Results

A detailed analysis of results is described, as well as comparisons and conclusions are done and drawn. In many cases when clustering algorithms are used, the visualization of the clusters might be useful to better understand and analyze the clusters. The most common approach is then to plot the multidimensional data in two dimension based on the most significant principal components. If these two explains a lot of the total variance, the plots are useful. In our case, the plot of the resulting clusters from K-means with K = 400 are shown below:

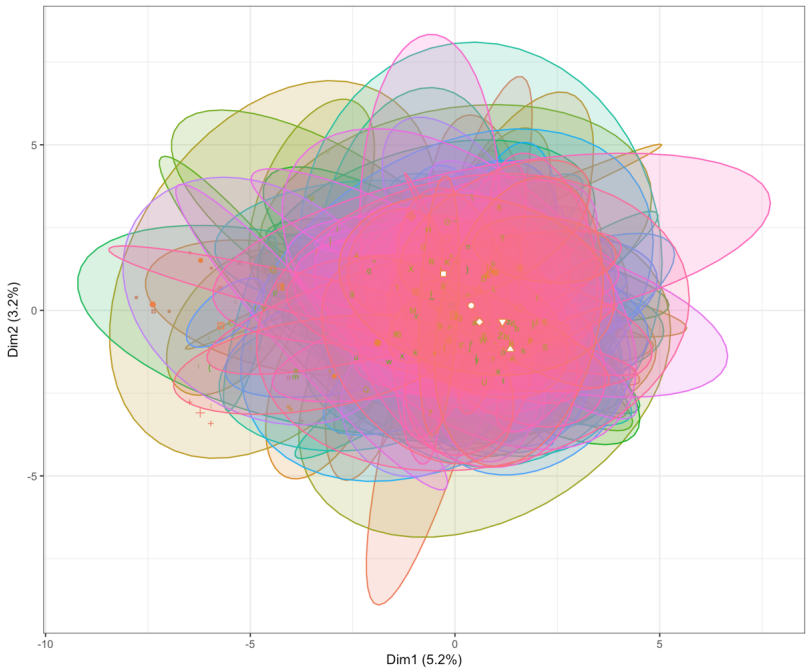


Figure 7: K-means plot

As one can see in figure 7 above, the information given by presenting this graph is very sparse, and almost useless. The different ellipses are the clusters and the geometrical figures describe the centroid of the figure. But as one can see on the axes, the first two principle components explains 8.4 % of the total variance and there are 62 additional dimensions that has been used for this clustering and therefore, in the following analysis, no additional plots of 2-dimensional clusters is presented.

7.1 Analysis of the results, derived from the K-Means Algorithm

As mentioned in the earlier section, the results given by K-Means Algorithm are as such: In all cases, K-Means seems to provide relatively poor results, in terms of overall F-measure score. The best case scenario was at a value of 0.250 (K=1200, with PCA).

K-Means provides a (relatively) fast method, in terms of processing the data, However, there seems to be several drawbacks:

i) It may be needed to run K-Means over several instances. Since the points at the front is chosen at random, it means that only local optimum of solutions are developed, but not global optimum for the data set. Unless there are codes/implementations in place to capture better results/iterations, an observer would have to repeat the procedure several times in order to get a clear understanding of the results from this algorithm. This is worsened by the fact that there is a larger number of clusters in the ground truth. Ideally, there should be a smaller number of clusters (in the ground truth) when using K-Means.

ii) Value of Number of Clusters must be known beforehand: the use of K-Means algorithm means that the number of clusters must be inputted beforehand. Having to varying the number of clusters as input can drastically change the results given. Additionally, it may be difficult to find out the exact number of clusters to be found, in the first place. Therefore, further testing and/or experimentation is required to get an accurate sense of what the number of clusters should be, to solve using this algorithm.

iii) Similar to point number ii), there is the possibility where the input for the number of clusters would force the data set into specific answers. This is known as 'overfitting'. This means that the algorithm fits the number of clusters into the data set, but is otherwise very prone to changes in the data set.

iv) K-Means is only ideal for 'globular'/spherical clusters: this would mean that clusters of different shapes may not be clustered well under this algorithm. Clusters that are elongated or spiral (in 2 dimensional space) in shape may exist. This spherical features is a direct consequence of the Euclidean distance used in this algorithm.
As only the raw data as input are available, this would mean that further work would have to done to ensure that the clusters are spherical, as much as possible, to make full use of the K-Means Algorithm.

v) K-Means algorithm does not capture clusters of differing variance well: there could be clusters which are grouped very close together (having very little variance) and other clusters which have test cases placed further apart (having more variance). This algorithm would tend to place test cases into clusters which are closer together, even though if these test cases might belong to a further, and more weakly-linked cluster.

vi) K-Means does not easily separate outliers/noise, and includes such data into the clusters: as mentioned earlier, the K-Means algorithm assigns ALL the test cases into clusters - it does not matter if the test cases are absurdly far from any clusters, it would be assigned into one of the clusters. This means that noise, or in this assignment, independent test cases, might be allocated into non-empty clusters, which implies that these independent test cases have some dependencies on other elements in the clusters.

To conclude, K-Means might be a very useful tool in cluster analysis. However, great work must be done to ensure that the raw data is compatible with the algorithm - even, spherical clusters, with all data points needing to be in a cluster. However, in our assignment, we would expect instances of dependencies as well as fully independent cases, and that K-Means would not be able to distinguish such information.

7.2 Analysis of the results, derived from the HDBSCAN Algorithm

As mentioned in the earlier section, the results given by HDBSCAN Algorithm are as such: HDBSCAN, on average, tended to yield much better results, in terms of F-measure score. The highest performing cases, where that of HDBSCAN, with PCA applied, with the distance parameter of 'Minkowski' or 'Euclidean' without PCA. This scenario resulted in a F-measure score of 0.333.

Our group feels that there are several advantages to using HDBSCAN for this particular problem:

i) HDBSCAN is able to filter out independent test cases. As mentioned earlier in the report, HDBSCAN is able to recognise such test cases as 'noise' in the data set, and would not assign them into clusters. This differs from K-Means, where the number of clusters has to be specified accurately to get good results from clustering.

ii) HDBSCAN is also able to identify clusters of varying sizes. This allows for it to identify (and distinguish) out smaller clusters, as well as large clusters.

iii) HDBSCAN only needs to be run once (as compared to K-Means). This means that the algorithm would develop its 'ideal' solution/results every single time. This allows for relative ease in writing the code, as the observer does not need to repeat the procedure over multiple times, and it allows for the result to be reproducible by external observers.

Finally, there must be some considerations when using HDBSCAN as well. For instance, the computational time, compared to other algorithms, may be faster, it does not compare favourably to K-Means, whose faster computational time may result in it being favoured for very large data sets. For this particular assignment however, the difference in time taken is rather insignificant, as the data set obtained is relatively small. When executing the code in Python, an average HDBSCAN algorithm takes on average 4.5 seconds - except when degenerate cases occur -, whereas a K-Means algorithm performance takes on average 12 seconds (when running K-Means over 10 instances, 1,2 second average per instance) - except when degenerate performances occur.

7.3 Analysis of the results, Effectiveness of use of PCA

The questions that naturally arise are: "is PCA effective ? is PCA procedure a good option?".Firstly, it must be noted that these questions cannot be answered "yes" or "no". The answers are more complicated. For instance, the answers depend on the chosen patter of parameters.

This analysis will mainly focus on the value of the F-Measure, as it takes into account equally the other two indexes. To be able to analyse some of the results better, the following graphs has been arranged. The first left one shows the performance of the K-means without PCA and the right one with PCA.

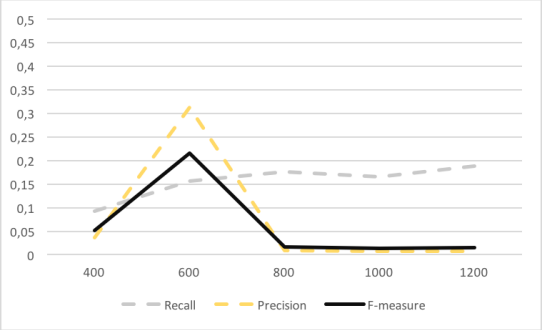


Figure 8: K-means performance without PCA

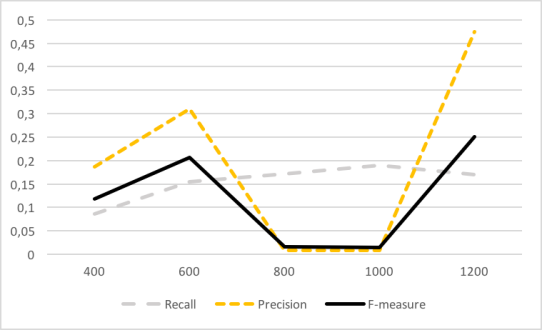


Figure 9: K-means performance with PCA

When K-Means is performed, an improvement using PCA can be seen when K is set to 1200 clusters; on the other hand, there is a worsening when K is set to 600. In fact, the largest F-Measure value when PCA is not applied is firstly reached when K is set to 600, while it is reached when K is set to 1200 when PCA procedure is used. It must be noted that the largest overall F-Measure is reached when PCA is applied. In conclusion, if one is only interested in reaching the largest F-Measure value applying PCA would represent the best way. However, it could be the case that one is interested in getting the largest F-Measure subject to certain constraints, for example using the smallest number of clusters. In this latter case, applying PCA would not represent the best way.

With respect to HDBSCAN the analysis is more intricate, this is due to the wide choice of parameters. In the graphs below, a chosen number of parameters are shown. Our group would consider `alpha` be equal to 1 and the minimum cluster size be equal to 2 in and let the distance measurement method and the cluster selection method change to see some of the influences that these parameters have on the F-Measure.

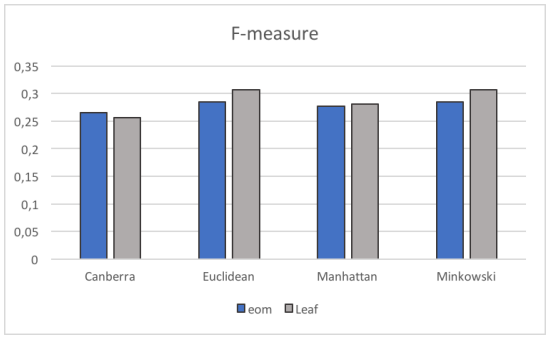


Figure 10: HDBSCAN performance without PCA

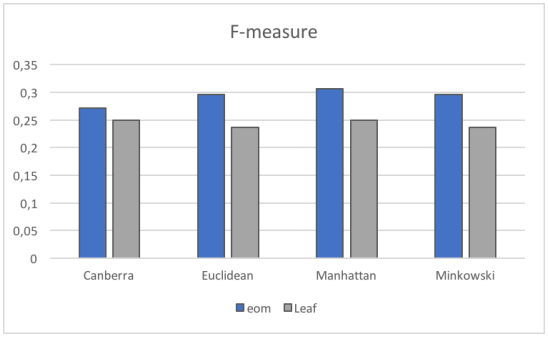


Figure 11: HDBSCAN performance with PCA

Referring to the figures, given the Manhattan distance, it can be seen that when the cluster selection method is `eom` using PCA implies an improvement with respect to F-Measure; however, if the cluster selection method is `leaf` than using PCA will entail a remarkable worsening. These phenomena occur similarly also when it is referred to Minkowski distance (p=4), Euclidean and Canberra. It must be noted that for Euclidean distance, using PCA implies a considerable worsening if the cluster selection method is `leaf`; indeed, F-Measure decreases by a 36% (it decreases from 0.33 to 0.21 ca.). This is the case also for Minkwoski distance. On the other hand, it must be noted that for Canberra distance both improvements and worsening are less remarkable, for instance if the cluster selection method is `eom` the improvement is only by a 7% whereas the worsening is by a 3%. However, one must also note that the returned best value of F-Measure when PCA is applied is lower than the one returned when PCA is not applied. Furthermore, it can be ascertained that when PCA is applied the algorithm degenerates more frequently. This usually happens when the minimum cluster size is 3, and cluster selection method is `eom`.

Finally, as explained above in detail, the procedure cannot be merely labelled neither as 'good' nor 'bad', but rather it must be analysed which are the requirements. For example, one could require that it *must* be used Canberra as distance parameter, or that a given cluster selection method must be avoided. Accordingly to these requirements then a proper and accurate choice can be made.

A last remark enhances what stated in the Method section when the PCA procedure was described. It can be noted that both improvements and worsening when PCA is applied are never overturning; this is with large probability due to the fact that the cumulative variance plot is mostly linear and to describe most of data variability a large number of components is required still.

8 Discussion

In this section, our group would look at the practical applications of the results obtained from the algorithms used, as well as discuss the importance of the results. Additionally, our group would look at improvements to the problem, as well as results, if any.

8.1 Application of results into Assignment

8.1.1 Issue of Algorithms to detect directional dependencies

Earlier in the assumptions, our group has made the assumption that bidirectional dependencies exist between elements, for the ease of comparison between the results, from the clustering algorithm, to the clusters in the ground truth. However, this may not be the case in the ground truth, or in reality - directional dependencies may exist between each test case. Suppose that test case B is directly dependent on test case A, but not vice versa. This would imply that test case A may be done in first, followed by test case B afterwards. In the event of the assumption, since the result would not indicate the direction of dependency, this would entail that test cases A and B must be done simultaneously. Therefore, there must be further checks to establish the direction of dependencies between test cases in reality.

8.1.2 Issue of Algorithms to detect direct dependencies

Another assumption made, for ease of comparison between the results and the ground truth, is that elements in the clusters in the ground truth has transitive dependencies between all other elements within the cluster. This assumption is made, as the algorithms only assigns test cases to clusters, but otherwise does not establish any relation, direct or otherwise, between test clusters. Suppose that test cases A, B, C and D are clustered together by the algorithm. What would be the relationship between each test case then? Is test case A (directly) dependent on test case B, C, D, or a combination of all three? The answer would not be immediately clear, just by looking at the results of the clustering algorithm. Therefore, there must be further checks to establish the direct dependencies between each test case.

8.1.3 Issue for algorithms to detect larger sized clusters

If one analyses the 40 elements cluster depicted in figure 12 the issue, due to transitivity assumption, for algorithm to detect this large cluster is clear. It must be noted that the cluster is depicted without the transitivity assumption but only using the original dependencies between test cases.

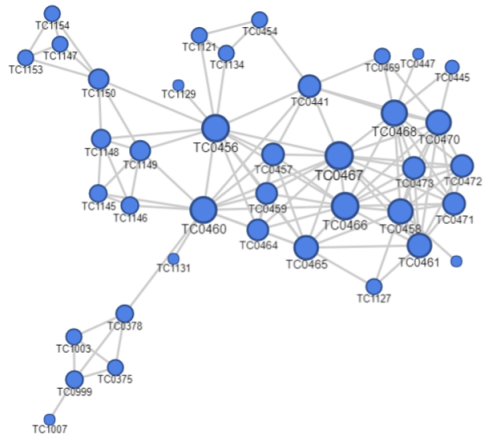


Figure 12: 40-Elements Cluster without transitivity assumption

If transitivity assumption holds, then TC1007 is assumed to be dependent on TC0447. It can be seen in the figure that between these two test cases there is a "long chain" dependency in reality. For instance, remembering the Doc2Vec algorithm that allows a later dependencies clustering accordingly to test cases closeness, it is clear that these two test cases will be hardly clustered together. Moreover, if for example TC1007 is clustered alone, then the amount of FN vertiginously increases. Whereas, in reality, a more accurate representation would be that it should increase only by one, because it should be dependent only on TC0999.

Finally, if one wants to keep the transitivity assumption and deal with this FP issue, then overlapping clustering could represent a valid solution. Then of course, the evaluation algorithms has to be changed to be able to take this into consideration. An idea of it could be is depicted in figure below.

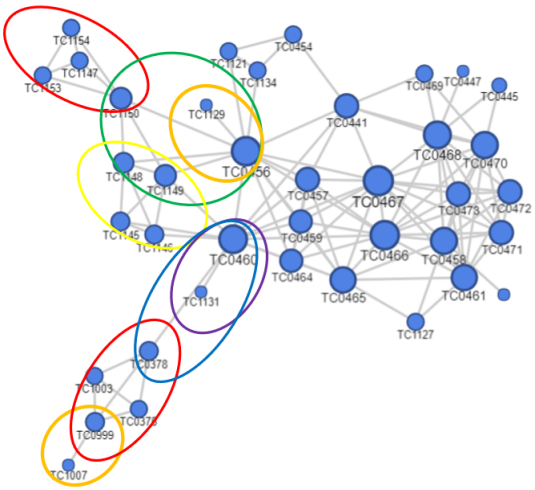


Figure 13: Idea of what overlapping clustering could be

For the sake of truth, it must be noted that if transitivity holds and a cluster is completely correctly identified then an increase of TP will occur. However, this increase is smaller if compared to the increase of FN instances if only one element is misclassified. In addition to this, it must be highlighted that these considerations should be investigated more in detail analysing the geometry, shape and number of each ground truth cluster without having the transitivity assumption being valid.

8.1.4 Evaluation of Performance Metrics

Initial observations of the performance metric, in particular, F_1 Score, indicate that it provides an equal importance of Precision, as well as Recall, from the algorithms given. However, this may not be the case - it may be possible for weigh in precision to be more important over recall, and vice versa. It may be possible to argue that there should be a higher emphasis on either recall or precision.

From the perspective that there would be additional post processing steps after clustering, it would make sense to focus more on recall rather than precision, as the time taken to sort out False Positives would be relatively less, than when compared to sorting our False Negatives from the independent test cases - If the assumption that only the clustered data would be further analysed and processed, whilst the independent test cases would be not processed/analysed further, it would be imperative that a good amount of dependent test cases is to be captured. This should imply a high(er) importance on recall (relative to precision).

However, it may also be argued that recall might be less important than precision, in this particular scenario. Most algorithms have around 1.000 to 1.200 for False Negatives. For this particular problem, it is observed that the largest clusters in this problem have 40 elements in it, with the next few having 16 and 14 elements. This may mean that, on the worst case scenario, where the 40 elements are clustered into clusters of 2 elements, This would mean that there would be a 760 instances of false negatives! This would mean that the performance metric would punish the algorithm very heavily in the instances where the cluster is not identified in its entirely. This would be true in the event that clusters have varying number of elements, which is true in this case. Therefore, the effect of False Negatives is over-represented in the performance of the algorithm - The Recall metric is artificially lower (when compared to data sets which have consistently small numbers of elements in clusters). In order to reduce the effect of this distortion, it would mean that there should be less importance in recall, and more focus on precision instead.

Finally, the performance metric does not accurately show the scale of positives (clusters of test cases, dependent on each other) and negatives (independent, non-clusterable test cases). In the case of the performance metric, it only utilises TP, FP, and FN cases - The amount of True Negatives is not factored in into the performance metric. This may be important in scenarios where there is a significant amount of independent elements. As the algorithm attempts to cluster the test cases, it may be equally important to place emphasis on identifying independent elements as well.

8.2 Improvements/Suggestions to improve the Scenario

8.2.1 Model improvements and other methods

In terms of model improvements to get better results, there are several other clustering techniques that can be used and implemented. One of these alternative methods is a method called Fuzzy C-Means (FCM).

FCM

The FCM method is a clustering method that allows data observations to belong to more than one cluster, i.e. it allows clusters to overlap. This is something that contradicts the main assumptions about the most common and classic clustering methods. The ability to belong to several clusters is described by a membership function which shows how much a data observation, 0 to 100%, belongs to a cluster center. This means that all data observation has a degree of membership to the cluster centers that sums to 1. The algorithm of FCM is similar to K-means in the way that it in an iterative way updates the centroids of the clusters and that it needs a determined number of clusters from the beginning. In addition to the centroids, also the membership are updated for each iteration and additional parameters, for example m, describing the fuzziness, i.e. the strictness and size of the membership function. This allows the method to be way more flexible compared to the strict boundaries of K-means and HDBSCAN.

In our case, the usage of FCM and the allowance of overlapping could have been a good idea. The possibility to label observations with multiple cluster belongings would possible make the clustering better, at least compared to K-means. Then we would have higher flexibility to handle the independent observations that are not supposed to be in a cluster. Based on the results, one could then have seen how the algorithm handles these points and draw conclusions on how to handle it. For example, an initial guess would be that the independent cases would have a membership that is low to multiple clusters rather than high for specific ones and therefore be assigned to multiple clusters. This could then improve the results, at least compared to K-means. [12]

Although, if this method would have been used, the definitions of the evaluation methods would have to be changed and the assumptions that we have made have to be revisited and discussed. Since we saw all dependencies as direct dependencies, the transitivity of the dependencies, the FCM method would probably result in some oddly looking results since dependent test cases could be in two clusters simultaneously. To do the evaluation in a correct way, then the assumption of the transitivity has to be changed. If for example A is dependent on B, and B is dependent on C in the ground truth, then B could be in two clusters and still have all cases correctly determined as TP but not if the assumption of transitivity is valid.

Subspace clustering

Additional methods and strategies to tackle the clustering and the high dimensional data is to use methods in subspace clustering, for example, SUBCLU which combines subspace clustering and DBSCAN. The method is useful for high-dimensional data and uses a bottom-up greedy strategy. SUBCLU is using the *downward-closure property* where it clusters data, using DBSCAN, in each 1-dimensional subspace of the data. Thereafter it tests these clusters in subsets of higher dimensions. This method results in clusters containing data observations that are equal in a specific number of dimensions. The clusters will have different number of dimensions and the total number of clusters will most likely increase if the method would be implemented. [13] The method uses the advantages of the density based DBSCAN but requires a lot of computations and can have a high runtime with many dimensions.

There are several other similar clustering methods based on subspaces and these methods are another way to handle the curse of dimensionality since clusters are based on subsets of the original dimensions. Although, the implementation are often more complex, time consuming and requires knowledge of the data set to be clustered to be able to interpret the data which made us skip these kind of methods in this project and instead focus on the methods that are more commonly used and which there are more theory around.

With that stated, it would be a good next step, if one wants even better results to further look into more advanced clustering methods, both subspace clustering methods as for example the previously mentioned SUBCLU, CLIQUE, DOC and MAFIA as well as projected clustering methods like PROFIT and PROCULUS. [14] [15]The area of machine learning and advanced statistical analysis is a pretty young field of study and new contributions in form of new algorithms and hybrid approaches based on previous methods are common, and so are the case of clustering methods as well.

[12]Bezdek, James C. *Objective function clustering. Pattern recognition with fuzzy objective function algorithms.* Springer, Boston, MA, 1981. 43-93.

[13]Kailing, Karin, Hans-Peter Kriegel, and Peer Kröger. *Density-connected subspace clustering for high-dimensional data.* Proceedings of the 2004 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2004.

[14]Parsons, Lance, Ehtesham Haque, and Huan Liu. *Subspace clustering for high dimensional data: a review.* Acm Sigkdd Explorations Newsletter 6.1 (2004): 90-105.

[15]Rajput, Dharmveer Singh, Pramod Kumar Singh, and Mahua Bhattacharya. *PROFIT: A Projected Clustering Technique.* Real World Data Mining Applications. Springer, Cham, 2015. 51-70.

8.2.2 Improvements of Performance Metrics

Use of different F-Measure Scores

There are other similarly related performance metrics that are linked to the F_1 Score. The general term is given as such, for a F_β Score. β indicates the importance of Recall, relative to Precision. For instance, if beta equals to 2, it would mean that the algorithms are evaluated, with Recall having twice the weightage of Precision. Conversely, having a beta of 0,5 means that Recall is only weighted half as much as Precision. The F_β Score is calculated as such:

$$F_\beta = (1 + \beta^2) \times \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{6}$$

For previous calculations, under results, β is equal to 1. It may be possible to change the value of β to see if we may get different performance results, using the same algorithm and settings. The table below illustrates some algorithms, with F_1 scores, $F_{0,5}$ scores, and F_2 scores, for comparison:

Distance	PCA	Alpha	MCS/K	CSM	F_1 Score	$F_{0,5}$ Score	F_2 Score
K-Means	Yes	N.A.	1200	N.A.	0,250	0,349	0,195
Minkowski	No	0.75	3	`leaf`	0.333	0,366	0,306
Manhattan	No	1,2	2	`leaf`	0,304	0,440	0,233
Manhattan	Yes	1,0	2	`eom`	0,306	0,371	0,261
Minkowski	No	0,75	3	`eom`	0,310	0,313	0,306
Canberra	Yes	1,0	2	`eom`	0,271	0,282	0,260

As it can be seen, changing the value of β could affect the performance algorithm. Therefore, one must consider the importance (and therefore, weightage) of Recall, as well as Precision, so as to compare the algorithms in a better fashion.

Use of Matthew's Correlation Coefficient

Another performance metric that can be used would be Matthew's Correlation Coefficient. This coefficient, introduced by Brian W. Matthews, aims to evaluate the performance of algorithms, by considering the problem as a binary assignment, rather than as a true test case/false case separation. Therefore, this algorithm considers equal importance in identifying dependent test cases (into clusters, the first class of the binary), as well as identifying independent test cases (into noise, the second class). This would be relevant in classifying data with imbalanced data, i.e. data sets which have significantly higher (or lower) positive elements.[16]

This algorithm uses all four elements of the confusion matrix, True Positive, True Negative, False Positive, and False Negative, and the formula is as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FN)(TN + FP)}} \tag{7}$$

Having a MCC value of 1 (which is the maximum possible value) means that our algorithm is able to distinguish independent test cases from dependent ones, and vice versa. Having a value of 0 means that the algorithm is no better than random guessing (assigning test cases randomly). Having a value of -1 (which is the lowest possible value) means that our clustering results are the complete opposite from the ground truth. Here, our group would sample some results, and compare the F_1 Score with its MCC score:

[16] Boughorbel, S., Jarray, F., El-Anbari, M. (2017). *Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric.* PLoS ONE, 12(6), e0177678. http://doi.org/10.1371/journal.pone.0177678

Distance	PCA	Alpha	MCS/K	CSM	TP	FP	FN	TN	F_1 Score	MCC Score
K-Means	Yes	N.A.	1200	N.A.	264	293	1.289	1.525.560	0,250	0,283
Minkowski	No	0,75	3	`leaf`	451	703	1.102	1.524.622	0,333	0,484
Manhattan	No	1,2	2	`leaf`	312	186	1.241	1.525.139	0,304	0,335
Manhattan	Yes	1,0	2	`eom`	369	488	1.184	1.524.837	0,306	0,396
Minkowski	No	0,75	3	`eom`	473	1.027	1.080	1.524.298	0,310	0,507
Canberra	Yes	1,0	2	`eom`	394	963	1.159	1.524.362	0,271	0,422

As it can be observed, using the MCC Score may affect the ranking of performance of the clustering algorithms. In particular, using of MCC Score has allowed some algorithm choices to be 'improved' in terms of performance ratings, in particular, the 5th (Minkowski, `eom`) and 6th (Canberra), had really good improvements. This shows that there are different metrics to evaluate the performance of the algorithms. Therefore, it is important to select the correct metric to check the performance of the clustering algorithms used.

8.3 A final consideration: applicability of results

Presented results show a wide applicability if the user is aware of what assumptions imply, such as the increase in FN labels due to transitivity assumption. Equally, results must be applied knowing that sometimes dependencies are not bidirectional but rather directional, for example sequential dependencies in time could happen in reality.
The potential user should be aware that results provided are overall results, but it could be the case he wants something more specific, especially with respect to some parameters' value.
In addition to this, the user must be conscious that in this particular case data is provided already processed, and PCA procedure should not be a straight decision but rather a more reasoned one accordingly to the detailed analysis presented in the Analysis section.

A Libraries in Python

Python Libraries

An insight describing the libraries used in Pythoh is here illustrated.

- `pandas` and `csv` are used to manipulate and import data from `csv` files;
- `numpy` and `scipy` are used to implement matrices, arrays and mathematical function;
- `sklearn.decomposition` is used to apply PCA procedure to raw data;
- `hdbscan` is used to perform the clustering algorithm.
- `sklearn.cluster` is used to apply KMeans algorithm

B Complete Results of all algorithms, listed as a table

The complete list of results of all algorithms would be listed as a table. The first table would show results, without PCA applied, whereas the second table show results with PCA applied. Do note that MCS/K would mean the Minimum Cluster Size (for HBDSCAN) and K refers to the number of clusters, in the case of K-Means clustering. CSM refers to the Cluster Selection Method. Minkoski distance is set to P=4. The first table is as follows:

Distance	PCA	Alpha	MCS/K	CSM	Recall	Precision	F-measure
K-Means	No	N.A.	400	N.A.	0,093	0,036	0,052
K-Means	No	N.A.	600	N.A.	0,156	0,313	0,216
K-Means	No	N.A.	800	N.A.	0,176	0,009	0,016
K-Means	No	N.A.	1000	N.A.	0,165	0,007	0,013
K-Means	No	N.A.	1200	N.A.	0,189	0,008	0,015
Canberra	No	0,75	2	eom	0,261	0,269	0,265
Canberra	No	1	2	eom	0,261	0,269	0,265
Canberra	No	1,2	2	eom	0,173	0,383	0,238
Canberra	No	0,75	2	leaf	0,223	0,301	0,256
Canberra	No	1	2	leaf	0,223	0,301	0,256
Canberra	No	1,2	2	leaf	0,151	0,573	0,239
Canberra	No	0,75	3	eom	0,307	0,191	0,235
Canberra	No	1	3	eom	0,269	0,207	0,234
Canberra	No	1,2	3	eom	0,123	0,316	0,177
Canberra	No	0,75	3	leaf	0,280	0,252	0,265
Canberra	No	1	3	leaf	0,236	0,219	0,228
Canberra	No	1,2	3	leaf	0,092	0,305	0,141
Euclidean	No	0,75	2	eom	0,248	0,336	0,285
Euclidean	No	1	2	eom	0,248	0,336	0,285
Euclidean	No	1,2	2	eom	0,212	0,427	0,284
Euclidean	No	0,75	2	leaf	0,233	0,449	0,307
Euclidean	No	1	2	leaf	0,233	0,449	0,307
Euclidean	No	1,2	2	leaf	0,188	0,606	0,287
Euclidean	No	0,75	3	eom	0,305	0,315	0,310
Euclidean	No	1	3	eom	0,260	0,371	0,306
Euclidean	No	1,2	3	eom	0,957	0,002	0,005
Euclidean	No	0,75	3	leaf	0,290	0,391	0,333
Euclidean	No	1	3	leaf	0,260	0,381	0,309
Euclidean	No	1,2	3	leaf	0,185	0,454	0,263
Manhattan	No	0,75	2	eom	0,243	0,323	0,277
Manhattan	No	1	2	eom	0,243	0,323	0,277
Manhattan	No	1,2	2	eom	0,217	0,444	0,292
Manhattan	No	0,75	2	leaf	0,204	0,450	0,281
Manhattan	No	1	2	leaf	0,204	0,450	0,281
Manhattan	No	1,2	2	leaf	0,201	0,627	0,304
Manhattan	No	0,75	3	eom	0,296	0,325	0,309
Manhattan	No	1	3	eom	0,256	0,394	0,310
Manhattan	No	1,2	3	eom	0,171	0,441	0,246
Manhattan	No	0,75	3	leaf	0,270	0,405	0,324
Manhattan	No	1	3	leaf	0,240	0,389	0,297
Manhattan	No	1,2	3	leaf	0,171	0,441	0,246
Minkowski	No	0,75	2	eom	0,248	0,336	0,285
Minkowski	No	1	2	eom	0,248	0,336	0,285
Minkowski	No	1,2	2	eom	0,212	0,427	0,284
Minkowski	No	0,75	2	leaf	0,233	0,449	0,307
Minkowski	No	1	2	leaf	0,233	0,449	0,307
Minkowski	No	1,2	2	leaf	0,188	0,606	0,287
Minkowski	No	0,75	3	eom	0,305	0,315	0,310
Minkowski	No	1	3	eom	0,260	0,371	0,306
Minkowski	No	1,2	3	eom	0,957	0,002	0,005
Minkowski	No	0,75	3	leaf	0,290	0,391	0,333
Minkowski	No	1	3	leaf	0,260	0,381	0,309
Minkowski	No	1,2	3	leaf	0,185	0,454	0,263

The second table, which shows results, with PCA applied, is listed below. Do note that MCS/K would mean the Minimum Cluster Size (for HBDSCAN) and K refers to the number of clusters, in the case of K-Means clustering. CSM refers to the Cluster Selection Method. Minkoski distance is set to P=4. The first table is as follows:

Distance	PCA	Alpha	MCS/K	CSM	Recall	Precision	F-measure
K-Means	Yes	N.A.	400	N.A.	0,086	0,187	0,118
K-Means	Yes	N.A.	600	N.A.	0,155	0,310	0,207
K-Means	Yes	N.A.	800	N.A.	0,171	0,008	0,016
K-Means	Yes	N.A.	1000	N.A.	0,189	0,008	0,015
K-Means	Yes	N.A.	1200	N.A.	0,170	0,474	0,250
Canberra	Yes	0,75	2	eom	0,254	0,290	0,271
Canberra	Yes	1	2	eom	0,254	0,290	0,271
Canberra	Yes	1,2	2	eom	0,254	0,290	0,271
Canberra	Yes	0,75	2	leaf	0,227	0,278	0,250
Canberra	Yes	1	2	leaf	0,227	0,278	0,250
Canberra	Yes	1,2	2	leaf	0,227	0,278	0,250
Canberra	Yes	0,75	3	eom	0,250	0,242	0,246
Canberra	Yes	1	3	eom	0,250	0,242	0,246
Canberra	Yes	1,2	3	eom	0,250	0,242	0,246
Canberra	Yes	0,75	3	leaf	0,231	0,297	0,260
Canberra	Yes	1	3	leaf	0,231	0,297	0,260
Canberra	Yes	1,2	3	leaf	0,231	0,297	0,260
Euclidean	Yes	0,75	2	eom	0,244	0,377	0,296
Euclidean	Yes	1	2	eom	0,244	0,377	0,296
Euclidean	Yes	1,2	2	eom	0,244	0,377	0,296
Euclidean	Yes	0,75	2	leaf	0,187	0,326	0,237
Euclidean	Yes	1	2	leaf	0,187	0,326	0,237
Euclidean	Yes	1,2	2	leaf	0,187	0,326	0,237
Euclidean	Yes	0,75	3	eom	0,997	0,002	0,003
Euclidean	Yes	1	3	eom	0,997	0,002	0,003
Euclidean	Yes	1,2	3	eom	0,997	0,002	0,003
Euclidean	Yes	0,75	3	leaf	0,159	0,378	0,224
Euclidean	Yes	1	3	leaf	0,159	0,378	0,224
Euclidean	Yes	1,2	3	leaf	0,159	0,378	0,224
Manhattan	Yes	0,75	2	eom	0,238	0,431	0,306
Manhattan	Yes	1	2	eom	0,238	0,431	0,306
Manhattan	Yes	1,2	2	eom	0,238	0,431	0,306
Manhattan	Yes	0,75	2	leaf	0,184	0,391	0,250
Manhattan	Yes	1	2	leaf	0,184	0,391	0,250
Manhattan	Yes	1,2	2	leaf	0,184	0,391	0,250
Manhattan	Yes	0,75	3	eom	0,999	0,002	0,003
Manhattan	Yes	1	3	eom	0,999	0,002	0,003
Manhattan	Yes	1,2	3	eom	0,999	0,002	0,003
Manhattan	Yes	0,75	3	leaf	0,150	0,362	0,212
Manhattan	Yes	1	3	leaf	0,150	0,362	0,212
Manhattan	Yes	1,2	3	leaf	0,150	0,362	0,212
Minkowski	Yes	0,75	2	eom	0,244	0,377	0,296
Minkowski	Yes	1	2	eom	0,244	0,377	0,296
Minkowski	Yes	1,2	2	eom	0,244	0,377	0,296
Minkowski	Yes	0,75	2	leaf	0,187	0,326	0,237
Minkowski	Yes	1	2	leaf	0,187	0,326	0,237
Minkowski	Yes	1,2	2	leaf	0,187	0,326	0,237
Minkowski	Yes	0,75	3	eom	0,997	0,002	0,003
Minkowski	Yes	1	3	eom	0,997	0,002	0,003
Minkowski	Yes	1,2	3	eom	0,997	0,002	0,003
Minkowski	Yes	0,75	3	leaf	0,159	0,378	0,224
Minkowski	Yes	1	3	leaf	0,159	0,378	0,224
Minkowski	Yes	1,2	3	leaf	0,159	0,378	0,224