

University of Reading

Department of Computer Science

**Mining Co-Morbidity Patterns and Associations with Health Outcomes from an Intensive Care Unit Registry**

Leah Gourley

*Supervisor*: Yevgeniya Kovalchuk

A report submitted in partial fulfilment of the requirements of

the University of Reading for the degree of

Bachelor of Science in *Computer Science*

DATE

**Declaration**

I, Leah Gourley, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person’s work, except where the works of others have been explicitly acknowledged, quoted and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Leah Gourley

DATE

**Abstract**

…

**Keywords**: Python, clustering analysis, partitioning algorithms, data extraction

**Report’s total word count:** - words (excluding references and appendices)

**Acknowledgements**

Thank you to Yevgeniya for your support and guidance, and helping me quickly adapt to unexpected changes in plan. Thank you to Pat for lending me your ear and for your confidence in me. Thanks to Mum and Dad for always being a phone call away, and for your unconditional belief in me. And lastly, thanks to my housemates for not complaining about my laptop running at all hours of the day and night!

**Contents**

**List of Figures**

**List of Tables**

**List of Abbreviations**

CCI – Charlson Comorbidity Index

CCS – Clinical Classification System

CITI – Collaborative International Training Institute

CSV– Comma-Separated Value

HIPAA – Health Insurance Portability and Accountability Act

HPEPP – Hewlett-Packard Enterprise Power Protector

ICD-9 – International Classification of Diseases, ninth revision

LASSO – Least Absolute Shrinkage and Selection Operator

MIMIC-III – Medical Information Mart for Intensive Care, third revision

NCI – NSHAP Comorbidity Index

NSHAP – National Social Life, Health and Aging Project

PCA – Principal Component Analysis

SSE – Sum of Squared Errors

**Chapter 1**

**Introduction**

* 1. **Background**

Describe comorbidity; purpose of clustering analysis;

types of partitioning algorithms? Maybe this comes in methodology instead

Comorbidity is the presence of an index condition and one or more other conditions, with the emphasis on these conditions’ effects on the individual; multimorbidity is instead defined as the co-occurrence of two or more conditions [Batstra & Neeleman]. This distinction is … comorbidity is a disease-to-disease association, but multimorbidity is a general collection of diagnoses without emphasis on their relationship.

* 1. **Problem statement**

Describe the approach? And the main goal?

Describe why this research needs to be done

Outline the issues that need addressing (eg how to quantify success of algorithm, why the algorithms need comparing, how to handle data)

Discuss difference between comorbidity and multimorbidity and the selection of comorbidity for this project

As well as identifying algorithm for clustering, a comorbidity index/measure must be chosen.

* 1. **Aims and objectives [162]**

The aim of this project is to develop a working program that performs clustering analysis on a large dataset. The dataset is the MIMIC-III healthcare database. The resulting clusters should be able to be analysed to consider comorbidities between different diagnoses, and their impact on health outcomes. (Has this particular area been done yet?)

The project will utilise multiple clustering algorithms, such as K-Means, mini-batch K-Means, and \_. Additionally, an interpretation of the M-algorithm, created by S. Sieranoja and P. Fränti in ‘Adapting k-means for graph clustering’, will be implemented to handle the given data as opposed to graph data, as in their original research. The code should take selected tables from the database, extract the relevant features (primary/secondary diagnoses, and outcome (death/discharge)), format it in an appropriate manner for the clustering algorithms.

The goal is to compare the results of these different algorithms using different evaluation measures, to determine the appropriate number of clusters for this data, and the optimal clustering.

* 1. **Solution Approach [113]**

Explain how the algorithms/metrics/measures were picked…

The completed program should be able to accept multiple inputs of the tables from the MIMIC III database. Utilising the pandas library in Python, it should manipulate these tables into one DataFrame or array, which will be passed to the various clustering algorithms. The algorithms will be run in a loop(s) for a pre-defined set of variable values, such as k (number of clusters) and b (batch size). After fitting each algorithm with the data, intrinsic evaluation metrics such as silhouette score, calinski-harabasz/davies-bouldin index should be calculated, along with SSE and \_. As well as this, the clusters should be plotted on a scatter plot, coloured by cluster or by event flag (diseased/discharged).

* 1. **Summary of contributions and achievements**

Implemented an adapted M-algorithm (explain how adapted) and implemented \_ measure.

* 1. **Organisation of the report [52]**

Section 2 describes existing research in the area. In section 3, the methodology is explained. Section 4 details the results, with discussions made in Section 5, drawing to a conclusion in Section 6. An additional Appendix chapter is provided with the graphical results of each clustering algorithm for each value of k/b.

**Chapter 2**

**Literature Review**

**2.1.1 Existing research into comorbidity scores [756]**

Early research into clustering for co-occurrence of diseases by Cornell et al [] identified six “clinically useful multimorbidity clusters…a Metabolic Cluster, an Obesity Cluster, a Liver Cluster, a Neurovascular Cluster, a Stress Cluster and a Dual Diagnosis Cluster”. Their work was among the first to aim to cluster specific groups of diseases, rather than generalised disease clustering across a population [] or clustering patients []. They identified that effective clustering relied on some ‘index of proximity’ to measure the distance between diseases and/or clusters. Similarity coefficients for binary data, such as their use of the Jaccard coefficient, can be utilised to quantify the presence and strength of comorbidities among diseases.

The chosen measure of proximity affects results when aiming to optimise the clusters in your data. Some such measures include a relative risk index or odds ratio as a measure of association. Batstra and Neeleman [] evaluated these measures in the context of psychiatric epidemiology, focussing on the comorbidity of individuals rather than the comorbidity of diseases. The conclusion reached was that these are appropriate, but not ideal, measures of classification. This was because such measures cannot distinguish a genuine comorbidity from a coincidental comorbidity, which can have the effect of over-emphasising association between clustered diseases. Therefore, if the goal of clustering is to identify aetiological factors, a cluster coefficient should instead be used. However, relative risks and odds ratios can be well suited to ‘nosological classification’ – the classification of diseases.

There exist risk indexes that specifically serve the purpose of quantifying comorbidity within co-occurrence research. Two such indexes are the Charlson Comorbidity Index (CCI) and the NSHAP Comorbidity Index (NCI). Vasilopoulos et al [] utilised these indexes in order to construct a framework for clustering NSHAP Wave 2 chronic health conditions.

Measures that do not account for coincidental comorbidity often overestimate the comparative strength of detected comorbidities; through the use of weighted data, this can be resolved. A weighted cluster coefficient that is widely used in comorbidity clustering research is the Somers’ D statistic. Ng et al researched the usefulness of this statistic in the context of multimorbidity [] by evaluating the results of cluster analysis using pairwise concordance statistics. Their work proposes an asymmetric Somers’ D statistic to combat a prediction of comorbidity by chance:

*-: Somers’ D statistic, Ng et al; where P is the number of concordant pairs; Q is the number of discordant pairs; Wr = P + Q + Tr; Wc = P + Q + Tc; Tr is the number of tied pairs on the row ordinal variable; and Tc is the number of tied pairs on the column ordinal variable.*

S. K. Ng [] additionally developed a two-way clustering model using hierarchical clustering and model-based algorithms, in order to identify comorbid diseases among individuals on binary (present, not present) data. An averaged pairwise Somers’ D statistic was used to assess the strength of the identified clusters, by summing the Somers’ D result for each pair of conditions in the cluster, multiplied by an indicator function to detect overlapping clusters. The result is a co-occurrence metric that adjusts for coincidental morbidity and accounts for the chance of a condition pair belonging to multiple clusters.

Producing concordant pairs of diseases does not reflect comorbidity of greater than two diseases completely. For data with weighted connections present between diseases, a co-occurrence network can be. Srinivasan et al [] created model to predict which patients would be high-cost, by combining the domains of data mining and clustering analysis to construct a tree-based network model from features identified in the data, and supported by HPEPP models for “community formation and structural properties”. Their method results in a great number of connections between conditions, demonstrating all the identified relationships in the data.

In addition to producing a co-occurrence network framework, they developed their own comorbidity index. Previously used metrics in co-occurrence network research [hidalgo et al] [steinhaeuser and Chawla] [Klimek et al] [Liu et al] demonstrate a bias towards rare diseases, so Srinivasan et al developed the co-occurrence correlation metric:

*-: Co-occurrence correlation, Srinivasan et al; where CCxy is the co-occurrence of diseases x and y; Px is the prevalence of disease x; and Py is the prevalence of disease y.*

This measure is used to calculate and assign edge weights between diseases, for the co-occurrence network. It does not have the limitations present in its predecessors, and it does not require the data to be binary, such as with the Somers’ D statistic, making it ideal for larger datasets.

**2.1.2 Existing research into co-occurrence clustering [584]**

García-Olmos et al [] attempted to cluster chronic diseases in their data to identify patterns of co-occurrence without use of a clustering algorithm. Instead, they split their characteristic data into various categories and dichotomised the categories, drawing summaries from the explanations of the data across 33 dimensions. They succeeded in identifying four main comorbidity patterns in their data, with associated comorbidity burdens. They also drew conclusions about the population of their data, such as 24.5% of individuals in the data had a present multimorbidity.

Wartelle et al [] used an agglomerative clustering method for multimorbidity clustering of ICD-10 codes. They identified that distance-based similarity measures are not applicable in the context of ICD-10 codes (and as such, ICD-9 codes), and instead used a simple relative risk metric to measure similarity. A relative risk index can be used on the statistical co-occurrences in the data; here, it was used on the probability of a patient receiving the diagnosis on a given visit to the Aube emergency department in France. Their results detailed 16 clusters, of which 5 were the most prevalent, present in 63% of the visits in the data.

A different approach by Dey et al uses a predictive model to cluster diagnoses in patients []. Working with ICD-9 codes, they aimed to develop a framework that could identify groups of codes that predict improvement in home-healthcare patients with urinary incontinence. They drew from existing information from the CCS, and, by adding their additional demographic, behavioural, psychosocial and physiological information, trained a LASSO-based predictive model to determine whether there would be improvement in the patients’ condition.

Ghosh et al [] sought to identify patterns of comorbidity in patients with a primary diagnosis of cardiovascular disease, through a combination of model-based prediction and k-means clustering methods. They used Ng et al’s asymmetric Somers’ D statistic [] on their binarized dataset, and clustered the diagnoses using model-based clustering. Further, they used a weighted k-means algorithm on each individual cluster, using the Silhouette score for each set of clusters on different values of k to determine the optimal number of sub-clusters within each cluster. They implemented logistic regression and SVM models as means of evaluating their clusters in predicting outcomes; their results show a 69% accuracy.

Model-based multimorbidity research, the likes of which carried out by Srinivasan et al, formed the foundation of work by Fränti et al [],…. They devised the K-algorithm, a form of k-means algorithm that could be applied to data in a graphical form. Further, they produced the M-algorithm, which is an optimised version of their K-algorithm that aims to counter the K- and k-means algorithms’ tendencies to get stuck on a local optimum. They evaluated the results of clustering on their diagnosis data through use of SVMs and logistic regression. As an alternative to a comorbidity index, they incorporated a delta function into their algorithm to assess the closeness of their clusters.

Fränti et al [] went on to apply the M-algorithm to patient diagnosis data from Finland. They utilised a relative risk index paired with the M-algorithm to identify information around disease co-occurrences within their data, that can be used to organise healthcare services around comorbid chronic diseases. Their data existed in the form of ICD-10 codes, so they constructed a co-occurrence network, with relative risk values as the weights on the connections. Out of all the existing research I have considered, this study’s dataset is the largest. As a result, they demonstrated the scalability of their model to large datasets.

**2.2 Critique of the review [359]**

Notably, a lot of the existing research into clustering comorbidities and multimorbidities aims to cluster based on individuals rather than based on diseases [Batstra and Neeleman] [Ng et al] [Srinivasan et al] [Ghosh] [Franti]. While this demonstrates how comorbidities present themselves across a population, it does not explain well how different diseases are related to each other – an area in the field that is lacking in both frameworks and research.

Much of the research makes use of a hierarchical or model-based clustering algorithm. These are effective approaches for clustering within a focused sub-section of disease diagnosis, such as cardiovascular disease [Ghosh], or for smaller datasets [Wartelle]. However, they are extremely computationally expensive for larger datasets. Use of k-means clustering (and variants, such as the K-algorithm) prove to be more robust in handling high quantities of data.

As well as this, a large number of studies make use of a binary matrix representation of their data. This results in an m\*n matrix, where m is the number of data entries (whether that is data regarding admission or patient) and n is the total number of possible diseases present in the data. For a large dataset and a broad range of diseases, this will result in a huge matrix, which will, like the choice of algorithm, make computation more complex. However, it is an efficient approach for smaller datasets or localised investigations into comorbidity.

With regards to measuring comorbidity, there is no agreed-upon measure that works best for quantifying a co-occurrence relationship between diseases or across a population. Indexes such as the CCI and NCI have the benefits of a uniform scoring system; however, this does not consider issues such as coincidental comorbidity. Various relative risk equations exist that aim to produce an index that can be used instead, but still have their own limitations, such as the lack of representation of overlapping clusters. Alternative distance-based metrics exist that attempt to account for overlaps without over- or underestimating relationships, such as Ng et al’s version of the Somers’ D statistic. However, each metric cannot be applied to all forms of data, so selection of a measure must take this into consideration.

**2.3 Summary**

**Chapter 3**

**Methodology**

**3.1 The MIMIC-III Clinical Database [415]**

The MIMIC-III Clinical Database [] is a large free-use database containing data taken from the Beth Israel Deaconess Medical Center in Boston, MA. Data was collected between the years 2001 and 2012 for 46,520 patients and 58,976 admissions to the critical care units of the hospital. The database consists of 26 tables. It encompasses a wide range of data, from patient demographics, discharge/mortality information, laboratory results and reports, medications and vital signs. Included are dictionary tables, denoted by the prefix ‘D\_’, which contain definitions for identifiers in the related table. For instance, the ‘DIAGNOSES\_ICD’ table has a corresponding ‘D\_DIAGNOSES\_ICD’ table containing a dictionary of all ICD9 code meanings present in the first table.

In line with ethical guidelines set by HIPAA standards, all personal information in the database has been deidentified. This involves shifting dates (such as date of birth, date and time of admission, etc) by a random offset, while preserving time of day/year; and removing any personally-identifiable information, including names, addresses and phone numbers. As such, all records in the database appear as between the years 2100 and 2200, and all patients with an age greater than 89 years instead have an age greater than 300 years.

This database was selected because of its use of ICD-9 diagnostic codes in documenting patient diagnosis for each admission. As my aim relates to identifying patterns in diagnosis, the use of ICD-9 codes provides ease in data handling as the diagnoses have already been categorised and tokenised. Additionally, the database is provided as a collection of CSV files, meaning the data will be easy to import into Python.

In order to access and use this data, a course in HIPAA requirements, the ‘Stage 1 Data or Specimens Only Research’ qualification provided by CITI, must be completed. As well as this, the data use agreement agreeing to data use and security standards needs to be signed.

Version 1.4 of the database was used, with it being the most recently released version at the time of this report. The tables within the database relevant to my problem are the ‘PATIENTS’ table, containing basic patient information such as date of birth/death, subject identifier and gender; the ‘ADMISSIONS’ table, containing a quantity of demographic information on the patient, patient and admission identifiers, and diagnosis information; and the ‘DIAGNOSES\_ICD’ table, containing a list of all diagnoses for a given admission, provided in the form of ICD-9 codes.

**3.2 Data cleaning [235]**

In order to use the data, it first needs to be handled to remove irrelevant columns and entries, and handle missing values. As well as this, the data needs to be adapted into a format appropriate for the clustering algorithms.

The information needed for clustering includes patient identifier, admission identifier, the primary diagnosis for the admission, a comma-separated list of subsequent secondary diagnoses and an entry indicating whether the patient was discharged or diseased at the end of the admission. Clustering analysis will be performed on diagnoses for a given admission, rather than for a given patient. This will ensure that the resulting clusters reflect relationships between the diseases, rather than the relationships between diagnoses a patient receives.

I am considering disease comorbidities, so it is necessary to remove any entries that contain diagnoses of any other kind. Below is a table representing the grouping of ICD-9 codes []:

|  |  |
| --- | --- |
| ICD-9 code range | Category |
| 001-139 | Infectious and parasitic diseases |
| 140-239 | Neoplasms |
| 240-279 | Endocrine, nutritional and metabolic diseases, and immunity disorders |
| 280-289 | Diseases of the blood and blood-forming organs |
| 290-319 | Mental disorders |
| 320-389 | Diseases of the nervous system and sense organs |
| 390-459 | Diseases of the circulatory system |
| 460-519 | Diseases of the respiratory system |
| 520-579 | Diseases of the digestive system |
| 580-629 | Diseases of the genitourinary system |
| 630-679 | Complications of pregnancy, childbirth, and the puerperium |
| 680-709 | Diseases of the skin and subcutaneous tissue |
| 710-739 | Diseases of the musculoskeletal system and connective tissue |
| 740-759 | Congenital anomalies |
| 760-779 | Certain conditions originating in the perinatal period |
| 780-799 | Symptoms, signs, and ill-defined conditions |
| 800-999 | Injury and poisoning |
| E and V codes | External causes of injury and supplemental classification |

Codes 290-319, 630-679, and 740-999, as well as the E and V codes, indicate conditions rather than diseases. As the problem focuses on diseases, it is sensible to remove any entries containing these codes. Further, the diagnoses need to be in one-to-one primary-secondary diagnosis pairs, in order to be able to perform clustering analysis. In order to improve the clustering algorithms’ performance, an additional step is to perform Principal Component Analysis on the primary and secondary diagnosis columns in order to reduce the dimensionality of the data.

**3.3 Clustering algorithms**

**3.3.1 k-Means Algorithm [173]**

Three partitioning clustering algorithms were used in order to perform a comparative clustering analysis of the data. The first algorithm selected was the k-means algorithm, a commonly-used partitioning algorithm. First proposed by MacQueen [] and developed further by Hartigan and Wong [], the algorithm seeks to group a shuffled dataset into k clusters by assigning k centroids within the data, and iteratively computing the SSE for each data point to all centroids, and assigning it to the closest centroid’s cluster. The centroids are represented by the mean of all data points within its cluster. The algorithm continues to iterate over each point until there is an iteration where no changes are made.

|  |  |
| --- | --- |
| **Algorithm 1**: K-Means clustering algorithm | |
|  | **Input**: data set D = [p1, p2, …, pn]; number of clusters k  **Output**: cluster centers C = [c1, c2, …, ck] |
| 1  2  3  4  5  6  7  8 | assign k initial centers C = [c1, …, ck]  **repeat**  **for** p in D **do**  calculate SSE for each c in C  assign p­i to c with smallest SSE  **for** c in C **do**  set ci to the mean of all p in c  **until** no change is made |

I have implemented the k-means algorithm through use of Python’s scikit-learn implementation, with a random state of 0 and all other parameters with their default values. A list of k values, K=[10, 40, 80, 100, 150, 200, 250, 300, 400, 500, 600] is used to run the k-means algorithm with the dataset, producing 11 different groupings of clusters for the data.

**3.3.2 Mini-Batch k-Means Algorithm [133]**

The k-means algorithm can take a very long time for a large dataset and/or a large value of k, given its iterative nature. An alternative, proposed by [], is the mini-batch k-means algorithm. Instead of passing over all the data on one iteration, it breaks the shuffled dataset down into b batches, and handles one batch per iteration. This algorithm runs much faster, and uses less memory, with an appropriate batch size.

|  |  |
| --- | --- |
| **Algorithm 2**: Mini-batch k-Means clustering algorithm | |
|  | **Input**: data set D = [p1, p2, …, pn]; number of clusters k; batch size b  **Output**: cluster centers C = [c1, c2, …, ck] |
| 1  2  3  4  5  6  7  8  9 | assign k initial centers C = [c1, …, ck]  **repeat**  select the next batch B of size b from D  **for** p in B **do**  calculate SSE for each c in C  assign p­i to c with smallest SSE  **for** c in C **do**  set ci to the mean of all p in c  **until** no change is made |

I have implemented this algorithm, again using scikit-learn, and with a random state of 0 and all other parameters with their default values. It uses the same list K for the number of clusters; and a list batchSize = [50, 100, 500, 1000] for batch size. I iterate over each value of k for each value in batchSize in my implementation.

**3.3.3 M-Algorithm [608]**

The final partitioning algorithm implemented is a variation of [Fränti](https://link.springer.com/article/10.1007/s10115-021-01623-y#auth-Pasi-Fr_nti) et al’s M-algorithm, which addresses the k-means algorithm’s tendency to get stuck on a local optimum. I have implemented an adapted form of this algorithm, as the original was created for graphical data, but my data is in the form of a NumPy array. It would be too costly to implement a co-occurrence network with such a large dataset, so the algorithm is adapted to simply calculate the ‘connections’ as needed, instead of creating a graph or matrix of connections.

My implementation of the algorithm still utilises a relative risk metric in place of a distance between data points to represent the weight of the relationship between diseases, such as in their application to a healthcare dataset []. I am using the total of the SSE values for each data point as the distance metric, as I am aiming to minimise the overall SSE value as an indicator of well-defined clusters.

Since the changes in SSE as a result of the M-algorithm can sometimes be minute, I have also used Srinivasan et al’s co-occurrence correlation, with the addition of a (what is it called?? That constant integer that you add/multiply by in an equation) to double the relative risk score if the patient was noted as deceased at the end of a given admission. This is represented by an event flag of 1, where 0 denotes the patient being discharged. I have chosen to do this since I want to consider the significance of comorbid diseases, rather than simply identify them; doubling the relative risk for a deceased data entry introduces the dimension of how fatal comorbid diseases could potentially be.

*-: Relative risk equation, where ∑A is the number of times diagnosis A appears in the dataset, ∑B is the number of times diagnosis B appears in the dataset; and d is a multiplication factor*

The original M-algorithm initialises the cluster centroids using a density-based initialisation, sorting data points by density and growing k clusters from the densest points. Since my dataset has no weights, and therefore no densities to sort by, I instead initialise the centroids by performing the k-means algorithm for k clusters.

The M-algorithm works by identifying the cluster with the largest SSE value and splitting it into two clusters; then it selects two random clusters, and if the probability is greater than a threshold, merge the clusters into one. I have set the threshold to 2\*1/k, as this will be adaptive to the number of clusters, but still great enough to ensure that merging the two selected clusters will have a significant impact. The probability is calculated as the total of all relative risks for data points in clusters A and B divided by the total relative risk for the whole cluster set, as per the original M-algorithm.

A decision needs to be made regarding whether the optimised cluster set should be kept. To do this, I compute the overall SSE for the cluster set at the beginning of the algorithm, as well as the overall relative risk. After splitting and merging clusters, I calculate the overall SSE and relative risk again. If the SSE is smaller after the algorithm, then keep the optimised set; else, if the relative risk is greater after the algorithm, then keep the optimised set; otherwise, discard the set. I check the SSE values first, as my goal is to minimise it. The relative risk is a secondary metric used to catch optimised sets with the same SSE or a larger SSE, but the clusters are better-defined.

|  |  |
| --- | --- |
| **Algorithm 3**: M-algorithm for clustering | |
|  | **Input**: cluster centers C [c1, c2, …, ck]; data set D = [p1, p2, …, pn]; number of clusters k  **Output**: optimised cluster centers CO |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | SSE = 0  **for** c in C **do**  **for** p in ci **do**  SSE += SSE for pj to ci  totalRR = 0  **for** c in C **do**  **for** p in ci **do**  totalRR += RelativeRisk(pj)  target = select c with largest SSE  CTemp = K-Means(D[all p in target], 2)  remove target from C  CTemp = concatenate C and CTemp  threshold = 2 \* 1/k  **repeat**  clusterA = randomly select c from C  clusterB = randomly select c from C != clusterA  clusterARR, clusterBRR = 0  **for** p in clusterA **do**  clusterARR += RelativeRisk(pi)  **for** p in clusterB **do**  clusterBRR += RelativeRisk(pi)  probability = (clusterARR + clusterBRR) / totalRR  **until** probability > threshold  CTemp CO = K-Means(D[all p in clusterA, clusterB], 1)  remove clusterA, clusterB from C  CTemp = concatenate C and CTemp  SSEnew = sum of SSE for all p in all c in CTemp  **for** c in CTemp **do**  **for** p in ci **do**  SSEnew += SSE for pj to ci  totalRRnew = 0  **for** c in CTemp **do**  **for** p in ci **do**  totalRRnew += RelativeRisk(pj)  **if** SSEnew < SSE **then**  CO = CTemp  **else if** totalRRnew > totalRR **then**  CO = CTemp  **else**  CO = C |

|  |  |
| --- | --- |
| **Algorithm 4**: Relative Risk | |
|  | **Input**: data point p, DataFrame DF = [columns: primaryDiagnosis, secondaryDiagnosis, eventFlag]  **Output**: relativeRisk |
| 1  2  3  4  5  6  7  8 | primaryDiagnosis = DF[p, 0]  secondaryDiagnosis = DF[p,1]  eventFlag = DF[p,2]  primaryCount = sum of times primaryDiagnosis appears in DF[:,0]  secondaryCount = sum of times secondaryDiagnosis appears in DF[:,1]  relativeRisk = ((primaryCount + secondaryCount) \* square\_root(2)) / square\_root(primaryCount\*\*2 + secondaryCount\*\*2)  **if** eventFlag == 1 **then**  relativeRisk \*= 2 |

The M-algorithm needs to be repeated a number of times (R) to have an effect, in case the first solution is discarded. [Fränti](https://link.springer.com/article/10.1007/s10115-021-01623-y#auth-Pasi-Fr_nti) et al recommended R=10, so this is what I will use. The k-means algorithm is the same as in 3.3.1, as is K.

**3.4 Evaluation metrics**

**3.4.1 Calinski-Harabasz Index**

The silhouette score is typically the metric used to evaluate

**3.4.2 Davies-Bouldin Index**

**3.4.3 Sum of Squared Error**

**3.4.4 Relative Risk**

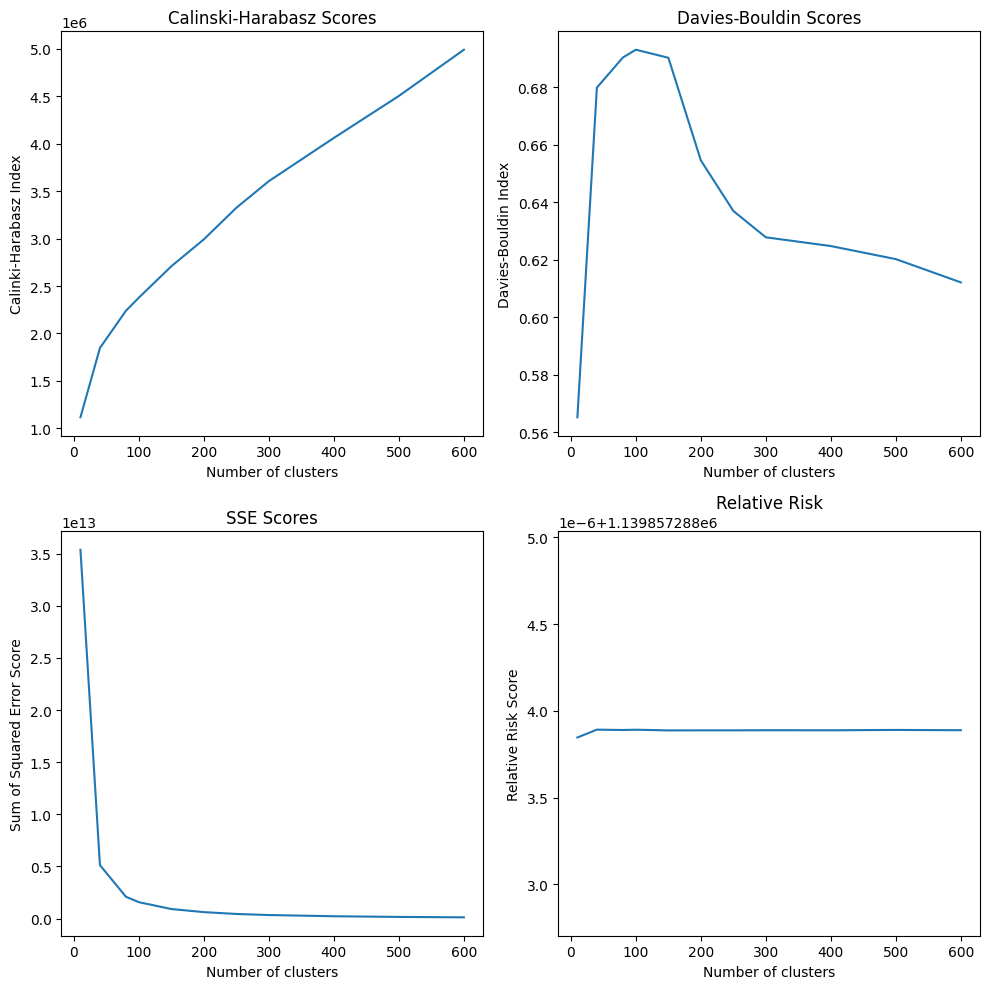
**3.4.5 Scatter and Line Plots**

**3.6 Summary**

**Chapter 4**

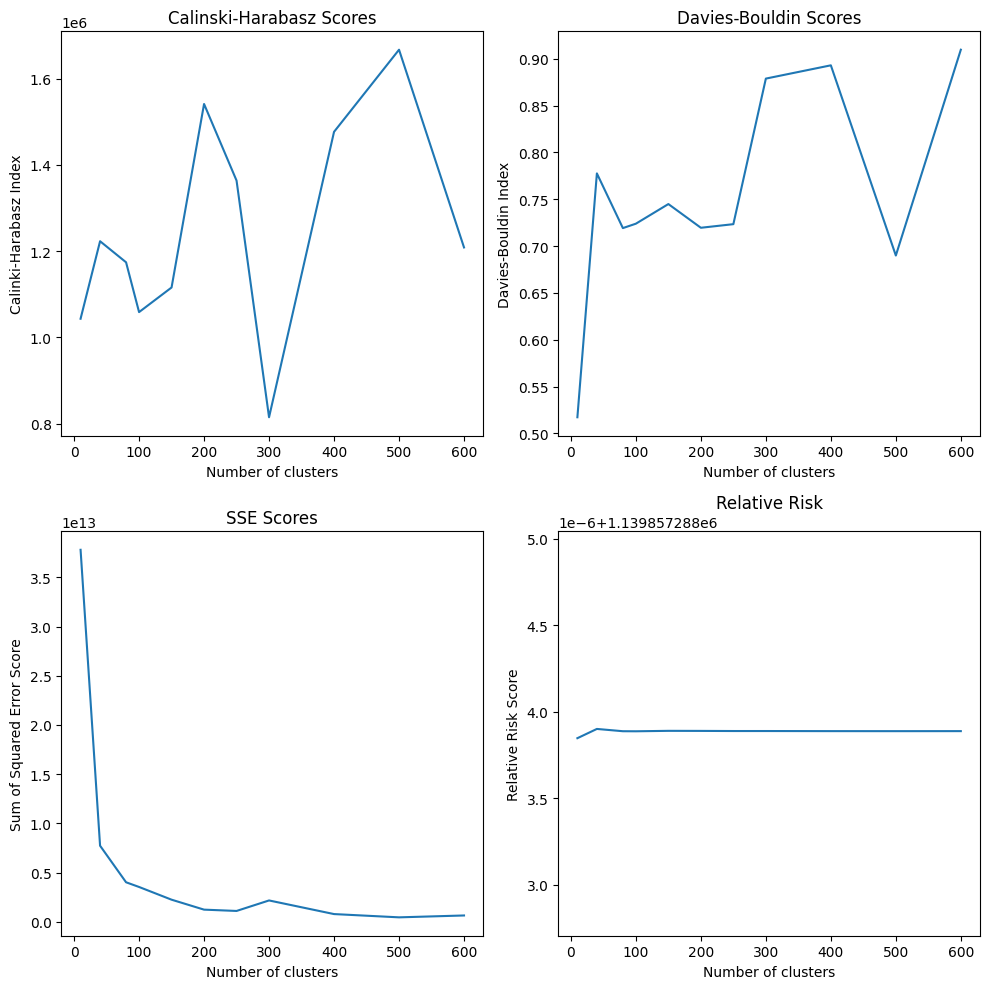
**Results**

**4.1 k-Means Algorithm**

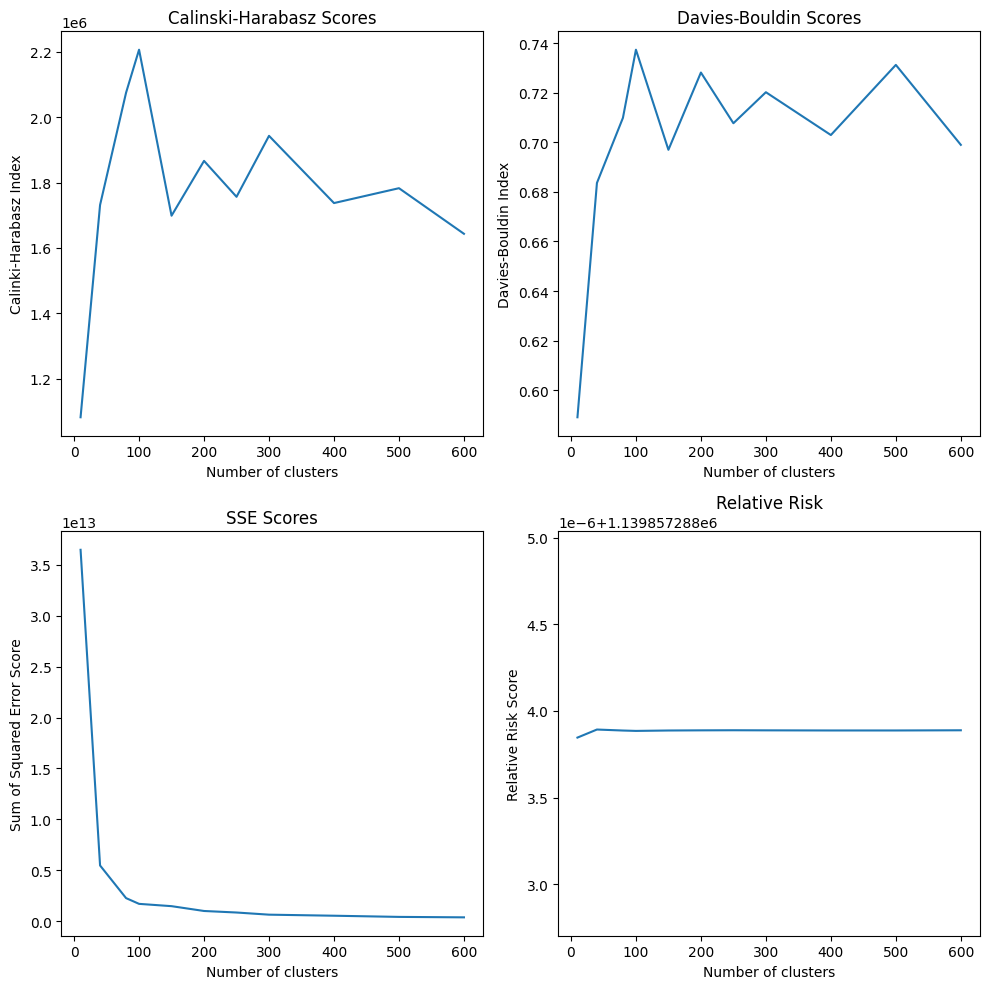
****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | Calinski-Harabasz | Davies-Bouldin | SSE | Relative Risk |
| 10 | 1,116,437.52198 | 0.565229 | 34,369,428,446,877.086 | 1,139,857.2877238467 |
| 40 | 1,846,742.46581 | 0.679878 | 5,131,423,163,801.025 | 1,139,857.2877238917 |
| 80 | 2,239,206.03226 | 0.690357 | 2,097,123,175,754.703 | 1,139,857.2877238898 |
| 100 | 2,378,199.35695 | 0.693070 | 1,576,627,154,205.349 | 1,139,857.2877238875 |
| 150 | 2,708,827.19333 | 0.690266 | 920,379,124,801.024 | 1,139,857.2877238875 |
| 200 | 2,992,592.20299 | 0.654557 | 623,952,704,753.612 | 1,139,857.2877238877 |
| 250 | 3,325,227.26365 | 0.636976 | 448,855,021,987.101 | 1,139,857.2877238877 |
| 300 | 3,605,876.29413 | 0.627819 | 344,709,187,368.826 | 1,139,857.2877238884 |
| 400 | 4,061,042.14943 | 0.624780 | 229,347,362,488.898 | 1,139,857.2877238882 |
| 500 | 4,503,506.51406 | 0.620233 | 165,336,876,332.255 | 1,139,857.2877238900 |
| 600 | 4,990,790.39230 | 0.612146 | 124,275,696,980.532 | 1,139,857.2877238840 |

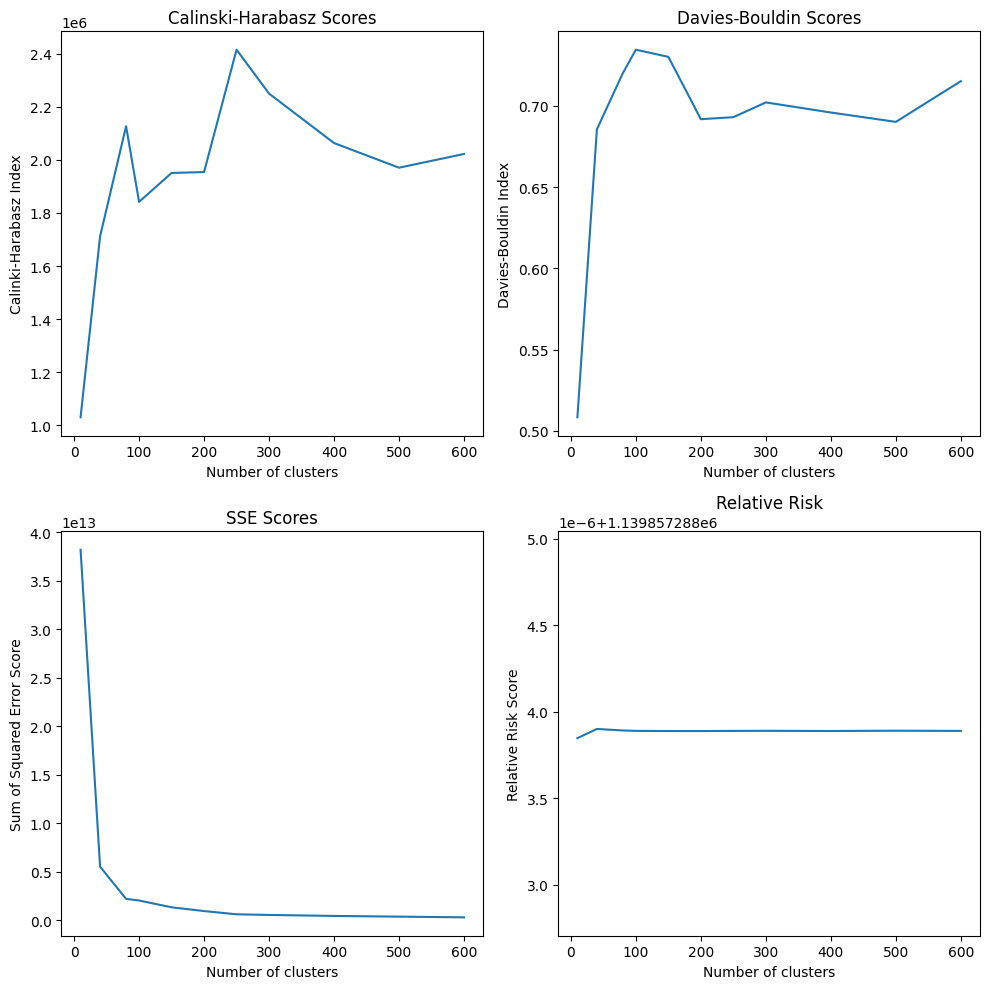
**4.2 Mini-Batch k-Means Algorithm**

****

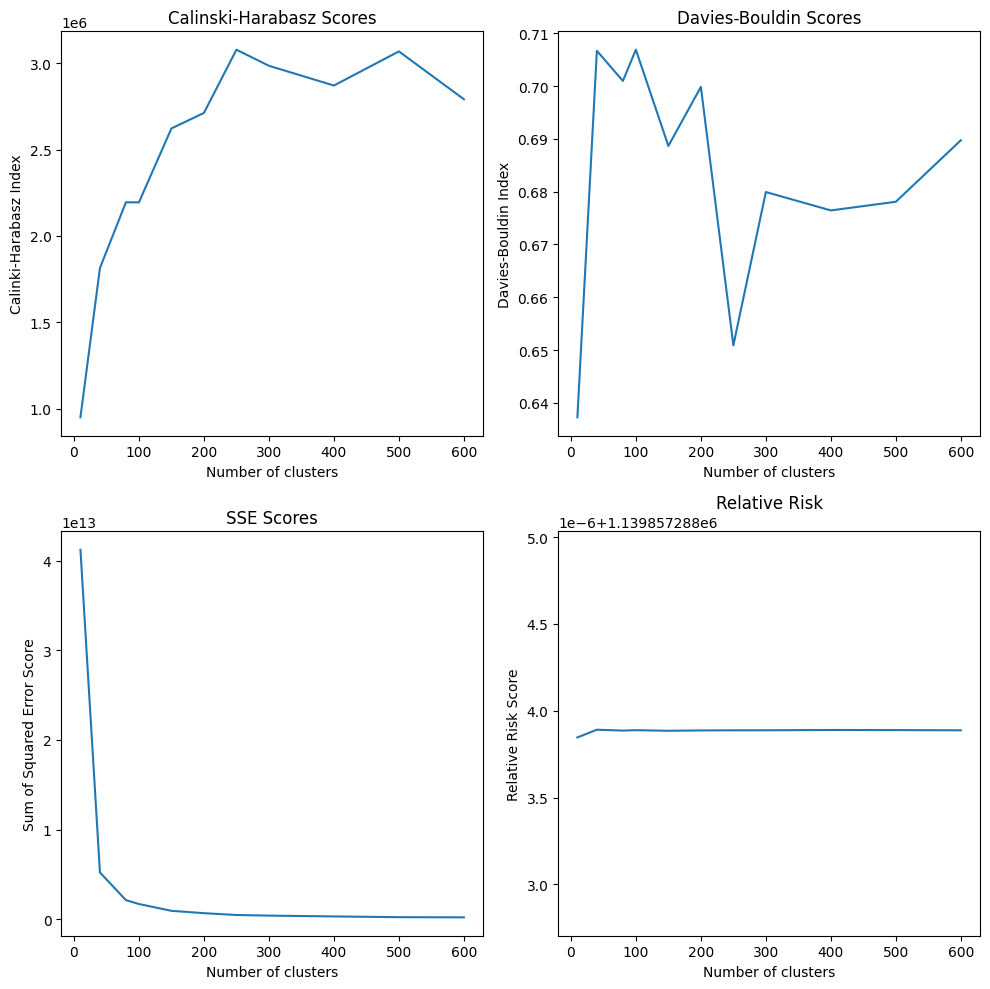
^100

****

^ 500

****

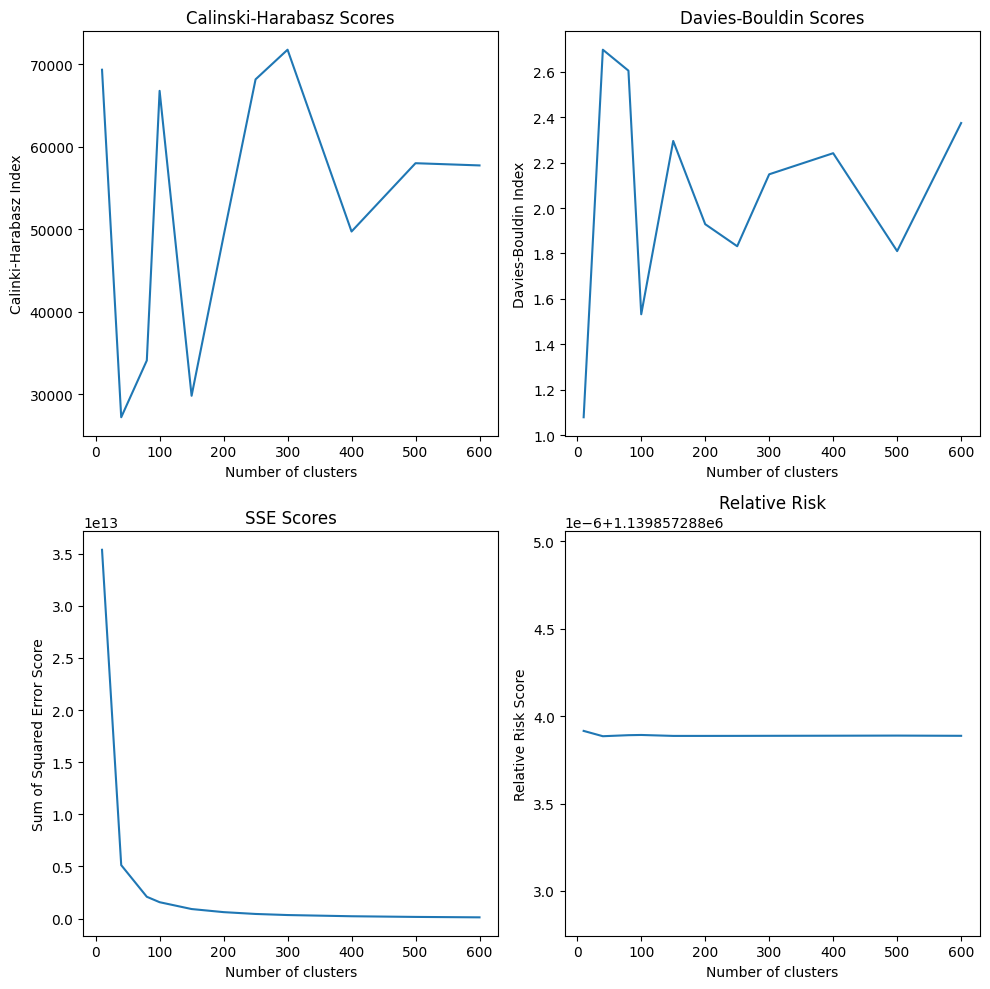
^1000

****

^5000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| K | Batch size | Calinski-Harabasz | Davies-Bouldin | SSE | Relative Risk |
| 10 | 100 | 1,043,008.63568 | 0.517288 | 37,792,643,452,599.170 | 1,139,857.2877238467 |
| 500 | 1,081,849.52661 | 0.589078 | 36,464,789,544,251.414 | 1,139,857.2877238463 |
| 1000 | 1,029,644.90263 | 0.508431 | 38,207,718,124,279.830 | 1,139,857.2877238463 |
| 5000 | 950,981.44569 | 0.637266 | 412,04,253,036,128.664 | 1,139,857.2877238465 |
| 40 | 100 | 1,222,765.60204 | 0.777565 | 7,729,834,917,594.727 | 1,139,857.2899239003 |
| 500 | 1,731,413.22514 | 0.683622 | 547,432,448,524.054 | 1,139,857.2877238928 |
| 1000 | 1,711,698.43436 | 0.685556 | 5,537,099,135,055.886 | 1,139,857.2877238996 |
| 5000 | 1,814,644.51167 | 0.706663 | 5,229,810,857,439.185 | 1,139,857. 287723891 |
| 80 | 100 | 1,173,913.07607 | 0.719186 | 4,015,665,394,415.526 | 1,139,857.287723887 |
| 500 | 2,074,936.52977 | 0.709884 | 2,270,153,300,281.581 | 1,139,857.287723887 |
| 1000 | 2,127,172.61460 | 0.720192 | 2,212,639,126,297.813 | 1,139,857.287723891 |
| 5000 | 2,194,914.23574 | 0.700967 | 2,140,595,843,377.004 | 1,139,857. 2877238858 |
| 100 | 100 | 1,058,402.67733 | 0.723886 | 3,538,151,730,354.357 | 1,139,857.2877238868 |
| 500 | 2,206,523.16840 | 0.737375 | 1,705,233,854,049.206 | 1,139,857.287723885 |
| 1000 | 1,841,944.45904 | 0.734561 | 2,045,484,607,209.991 | 1,139,857.2877238886 |
| 5000 | 2,194,466.75690 | 0.706877 | 1,711,342,750,748.854 | 1,139,857. 2877238882 |
| 150 | 100 | 1,115,491.23456 | 0.744837 | 2,247,341,505,054.188 | 1,139,857.2877238893 |
| 500 | 1,698,588.75111 | 0.696992 | 1,479,034,286,004.637 | 1,139,857.2877238872 |
| 1000 | 1,950,833.01562 | 0.730170 | 1,344,056,797,214.499 | 1,139,857.2877238877 |
| 5000 | 2,622,384.51866 | 0.688652 | 951,961,475,077.590 | 1,139,857. 287723885 |
| 200 | 100 | 1,541,064.58625 | 0.719535 | 1,235,766,292,625.600 | 1,139,857.2877238889 |
| 500 | 1,866,309.78042 | 0.728166 | 1,005,633,376,019.376 | 1,139,857.2877238882 |
| 1000 | 1,954,364.13754 | 0.691804 | 957,795,386,552.418 | 1,139,857.2877238875 |
| 5000 | 2,711,894.50127 | 0.699829 | 690,029,810,899.464 | 1,139,857. 2877238868 |
| 250 | 100 | 1,363,467.47567 | 0.723374 | 1,102,682,244,116.245 | 1,139,857.2877238882 |
| 500 | 1,756,422.42308 | 0.707708 | 855,656,250,812.552 | 1,139,857.2877238886 |
| 1000 | 2,415,700.53288 | 0.693059 | 621,287,939,247.358 | 1,139,857.2877238884 |
| 5000 | 3,077,699.59973 | 0.650884 | 486,696,823,212.219 | 1,139,857. 2877238875 |
| 300 | 100 | 814,506.05542 | 0.878742 | 2,169,333,866,201.457 | 1,139,857.2877238882 |
| 500 | 1,942,986.43197 | 0.720226 | 644,250,932,298.549 | 1,139,857.2877238882 |
| 1000 | 2,250,196.91635 | 0.702141 | 556,492,359,209.177 | 1,139,857.2877238889 |
| 5000 | 2,984,734.97631 | 0.679922 | 417,751,259,002.250 | 1,139,857. 2877238877 |
| 400 | 100 | 1,476,491.98934 | 0.892908 | 785,313,498,524.569 | 1,139,857.2877238877 |
| 500 | 1,737,258.60035 | 0.702948 | 540,712,378,515.371 | 1,139,857.2877238875 |
| 1000 | 2,063,778.77762 | 0.695925 | 453,723,957,233.290 | 1,139,857.287723888 |
| 5000 | 2,870,382.90459 | 0.676438 | 326,328,381,129.320 | 1,139,857. 2877238896 |
| 500 | 100 | 1,666,973.18988 | 0.689914 | 453,038,178,478.893 | 1,139,857.2877238875 |
| 500 | 1,782,764.18323 | 0.731248 | 425,444,069,905.175 | 1,139,857. 2877238875 |
| 1000 | 1,970,761.80700 | 0.690162 | 381,896,129,055.196 | 1,139,857. 2877238896 |
| 5000 | 3,067,896.79569 | 0.678079 | 244,562,431,893.423 | 1,139,857. 2877238889 |
| 600 | 100 | 1,208,371.20664 | 0.909572 | 642,962,938,408.732 | 1,139,857.2877238877 |
| 500 | 1,643,238.56453 | 0.698984 | 381,456,132,637.156 | 1,139,857. 2877238884 |
| 1000 | 2,022,544.26748 | 0.715213 | 308,883,431,063.382 | 1,139,857. 2877238884 |
| 5000 | 2,791,205.54388 | 0.689727 | 224,539,482,742.496 | 1,139,857. 2877238875 |

**4.3 M-Algorithm**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| K | Calinski-Harabasz | Davies-Bouldin | SSE | Relative Risk |  |
| 10 | 69,333.80250 | 1.079009 | 35,369,428,446,877.086 | 1,139,857.287723916 | DSESEDDDED |
| 40 | 27,179.92605 | 2.697155 | 5,131,423,163,801.024 | 1,139,857.2877238854 | DEESESSDDS |
| 80 | 34,066.77479 | 2.604925 | 2,097,123,175,754.703 | 1,139,857.2877238914 | EESESDDESS |
| 100 | 66,773.64468 | 1.532385 | 1,576,627,154,205.349 | 1,139,857.2877238926 | EDSESSESSD |
| 150 | 29,786.95842 | 2.295449 | 920,379,124,801.024 | 1,139,857.2877238872 | DDDDSSDSDD |
| 200 | 49,160.24750 | 1.928984 | 623,952,704,753.612 | 1,139,857.2877238872 | DDSSSESEES |
| 250 | 68,166.91766 | 1.831956 | 448,855,021,987.101 | 1,139,857.2877238875 | DSESDSSDSS |
| 300 | 71,761.85910 | 2.148672 | 344,709,187,368.826 | 1,139,857.2877238877 | DDESSSDSSD |
| 400 | 49,710.23076 | 2.241549 | 229,347,362,488.898 | 1,139,857.2877238884 | DESESDSESD |
| 500 | 57,995.78084 | 1.810434 | 165,336,876,332.255 | 1,139,857.287723889 | DDDSSSDSDS |
| 600 | 57,727.15318 | 2.374340 | 124,275,696,980.532 | 1,139,857.2877238877 | DDSDDSSSDS |

**4.4 Summary [23]**

This chapter shows the corresponding Calinski-Harabasz Index, Davies-Bouldin Index, SSE and relative risk (figure) for each value of k (and b in 4.2), applied to the k-means algorithm (4.1), mini-batch k-means algorithm (4.2) and M-algorithm (4.3).

**Chapter 5**

**Discussion and Analysis**

**5.1 The optimal value of k**

K means

* From the k-means ch scores, there is a steady progression. From this, it suggests that k should be as high as possible.
* For the db scores, it increases sharply between 10 and 50, continues increasing until 100, declines quite fast until around 300, where it begins to decline more slowly. This suggests k should be greater than 300.
* The SSE score drops sharply until about 100 clusters, and then peters out slowly. From this, it suggests that k should be greater than 100.
* Ch: 600, 500, 400, 300, 250, 200, 150, 100, 80, 40, 10
* Db: 10 600 500 400 300 250 200 40 150 80 100
* Sse: 10, 40, 80, 100, 150, 200, 250, 300, 400, 500, 600
* Rr: 40, 500, 80, 300, 400, 200/250, 100/150, 600, 10

For the mbk 100 clusters

* ch scores get higher until 40, drop again until 100 to about the same score as for 10; then sharply rise at around 200, then drop to the lowest score by a long shot for 300 clusters, then sharply back up at 400, higher at 500, then sharply back down at 600. This suggests k should be between 150-200, or 300-500, but biased away from 300.
* We want db scores to get smaller, but for this the values get higher. The only time there is a decline is between 400-500. In fact, they largely increase at around 300, like the ch scores decline there.
* The SSE curve is the same shape for all variations except this one, which peters out at a much higher value and also has a bump in the curve at around 300. This suggests that 300 is most definitely the wrong value.
* Ch: 500 200 400 250 40 600 80 150 100 10 300
* Db: 10 500 80 200 250 100 150 40 300 400 600
* Rr: 40 (by a large amount), 150, 200, 250/300, 600/400, 500, 80, 100, 10

Mbk 500

* Inconsistently jagged ch scores, and also doesn’t match with the db scores. From the main peaks in ch, I would draw the conclusion that k should be around 100, or 300.
* For db scores, I would avoid values greater than 100, although there is a relatively significant decrease at around 400 and again at 600. These values are inconsistent.
* SSE curve much like k means curve.
* Ch: 100, 80, 300, 200, 500, 40, 250, 400, 150, 600, 10
* Db: 10, 40, 600, 100, 500, 200, 300, 80, 250, 400, 150
* Rr: 40, 250, 600, 200/300, 400/500, 150, 80, 100, 10

Mbk 1000

* The ch values increase sharply towards 80 clusters, then drops rapidly at 100. Most significant increase shown between 250-300; from there it drops out slowly again. Suggests the best value of k to be around, but before, 300.
* Again db scores increase instead of decrease – really good low values at the beginning, steep increase but then a quite decline between 150-200, stays around that level before increasing marginally again at 300. Suggests a value of k should be found between 150-300.
* SSE curve no deviations.
* Ch: 250, 300, 80, 400, 600, 500, 200, 150, 100, 40, 10
* Db: 10, 40, 500, 200, 250, 400, 300, 600, 80, 150, 100
* Rr: 40, 80, 500, 300, 100, 250/600, 400, 150, 200, 10

Mbk 5000

* Ch scores show good upward tend, sharply until 250, dips a little between 300 and 400 and picks up again. This tells that k should be found after 200 but not around 300.
* Db scores climb high and bounce up and down between 80-200, with a significantly lower value at 150 but high again at 200. Most notable drop, and best db value is at 250, although picks up again at 300 but lower than 200. Values stay lower than 200’s value but higher than 300’s for the remainder. 250 would be ideal value for k, but second best option is greater than 300.
* SSE curve consistent.
* Ch: 250, 500, 300, 400, 40, 600, 200, 150, 80, 100, 10
* Db: 10, 250, 400, 500, 300, 150, 600, 200, 80, 40, 100
* Rr: 40, 400, 500, 100, 300, 250/600, 200, 80, 150, 10

M algorithm:

* High ch for small values of k, crops sharply at 40, up again sharply at 100, drops way down again at 200, high again at 250 and stays high. Drops relatively lower but by no means the lowest at 450. Best k values probably found between 200-400.
* Db levels up and down around the same range. Probably best k could be selected between 100-500 but not between 300-400.
* Ch: 300, 10, 250, 100, 500, 600, 400, 200, 80, 150, 40
* Db: 10, 100, 500, 250, 200, 300, 400, 150, 600, 80, 40
* Rr: 10, 100, 80, 500, 400, 300/600, 250, 150/200, 40

K means, Mbk 5000, mbk 500, mbk 1000, m algorithm, mbk 100

**5.2 Interpreting the results**

**5.3 Significance of the findings**

**5.4 Limitations**

Cannot handle the possibility of overlapping clusters (Ng et al)

Scalability challenge – extremely computationally expensive for large data set

American study should not be generalised to other populations (bias)

Does not consider other factors impacting comorbidity, such as age, gender, lifestyle.

May not reflect the actual relationship between diseases, as only information comes from visits to the hospital

All the partitioning algorithms create spherical clusters, would the data look like that?

**5.5 Summary**

**Chapter 6**

**Conclusions and Future Work**

**6.1 Conclusions**

**6.2 Future work**

A SVM could be trained in accordance with the cluster results, and on strings of sequential diagnoses (both primary and secondary) in order to make predictions on future health outcomes for a patient.

**Chapter 7**

**Reflection**

**References**

[https://medinform.jmir.org/2022/5/e35422 - clustering with icd10](https://medinform.jmir.org/2022/5/e35422%20-%20clustering%20with%20icd10) codes

<https://link.springer.com/article/10.1007/s10115-021-01623-y> - k algorithm and m algorithm

<https://github.com/uef-machine-learning/gclu/blob/main/graphclu.cpp> - the code from ^

<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000353> - how i could change the data points to graph nodes

[] A. Johnson, T. Pollard, L. Shen, et al. (May 24, 2016). “MIMIC-III, a freely accessible critical care database”, *Sci Data*, vol. 3, pp. 160035 doi: <https://doi.org/10.1038/sdata.2016.35>

[] National Center for Health Statistics, (Mar 3, 2022). “ICD-9”, *Centers for Disease Control and Prevention (CDC)*. [Online]. Available: https://www.cdc.gov/nchs/icd/icd9.htm

[] K. J. Wong (Dec 9, 2022). “7 Evaluation Metrics for Clustering Algorithms”, *Towards Data Science*. [Online]. Available: <https://towardsdatascience.com/7-evaluation-metrics-for-clustering-algorithms-bdc537ff54d2>

[] J. Cornell, J. Pugh, J. W. Jr. Williams, et al. (Jan 13, 2009). “Multimorbidity Clusters: Clustering Binary Data From Multimorbidity Clusters: Clustering Binary Data From a Large Administrative Medical Database”, *Applied Multivariate Research*, vol. 12, issue 3, pp. 163-182. doi: [10.22329/amr.v12i3.658](http://dx.doi.org/10.22329/amr.v12i3.658)

### [] L. García-Olmos, C. H. Salvador, Á. Alberquilla, et al. (Feb 16, 2012). “Comorbidity Patterns in Patients with Chronic Diseases in General Practice”, *PLoS One*, vol. 7, issue 2. doi: <https://doi.org/10.1371/journal.pone.0032141>

[] A. Wartelle, F. Mourad-Chehade, F. Yalaoui, *et al*. “Clustering of a Health Dataset Using Diagnosis Co-Occurrences”, *Applied Sciences,* vol. 11, issue 5, pp. 2373. (Mar, 2021). doi: <http://dx.doi.org/10.3390/app11052373>

[] S. Dey, G. Simon, B. Westra, *et al*. “Mining Interpretable and Predictive Diagnosis Codes from Multi-source Electronic Health Records”, in *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, vol. 2. M. Zaki, Z. Obradovic, P. Ning-Tan, *et* al, Eds. Philadelphia: Society for Industrial and Applied Mathematics Publications, 2014, pp. 1055-1063.

[] S. K. Ng, L. Holden, J. Sun. (Jun 19, 2012). “Identifying comorbidity patterns of health conditions via cluster analysis of pairwise concordance statistics”, *Statistics in Medicine,* vol. 31, issue 27, pp. 3393-3405. doi: <https://doi.org/10.1002/sim.5426>

[] S. K. Ng. (May 21, 2015). “A two-way clustering framework to identify disparities in multimorbidity patterns of mental and physical health conditions among Australians”, *Statistics in Medicine*, vol. 34, issue 26, pp. 3444-3460. doi: <https://doi.org/10.1002/sim.6542>

[] L. Batstra, E. H. Bos, J. Neeleman. (2002). “Quantifying psychiatric comorbidity: Lessons from chronic disease epidemiology”, *Social Psychiatry and Psychiatric Epidemiology*, vol. 27, pp 105-111. doi: <https://doi.org/10.1007/s001270200001>

[] T. Vasilopoulos, A. Kotwal, M. J. Huisingh-Scheetz, *et al*. (Oct 29, 2014). “Comorbidity and Chronic Conditions in the National Social Life, Health and Aging Project (NSHAP), Wave 2”, *The Journals of Gerontology: Series B*, vol. 9, issue 2, pp. 154-165. doi: <https://doi.org/10.1093/geronb/gbu025>

[] K. Srinivasan, F. Currim, S. Ram. (Dec 13, 2017). “Predicting High-Cost Patients at Point of Admission Using Network Science”, *IEEE J Biomed Health Inform.,* vol. 22, issue 6, pp. 1970-1977. doi:  10.1109/JBHI.2017.2783049

[] D. Ghosh, J. Cabrera, T. N. Adam, *et al*. (Jun 1, 2020). “Comorbidity Patterns and Its Impact on Health Outcomes: Two-Way Clustering Analysis”, *IEEE Transactions on Big Data,* vol. 6, issue 2, pp. 359-368. doi: <http://dx.doi.org/10.1109/TBDATA.2016.2623323>

[] P. Fränti, S. Sieranoja, K. Wikström, T. Laatikainen. (Apr 4, 2022). “Clustering Diagnoses From 58 Million Patient Visits in Finland Between 2015 and 2018”, *JMR Medical Informatics*, vol. 10, issue 5. doi: <https://doi.org/10.2196/35422>

[] C. A. Hidalgo, N. Blumm, A. L. Barabási and N. A. Christakis, (Apr 4, 2009). “A dynamic network approach for the study of human phenotypes”, PLoS Computational Biology, vol. 5, issue 4. doi: <https://doi.org/10.1371/journal.pcbi.1000353>

[] K. Steinhaeuser and N. V. Chawla. (2009). “A network-based approach to understanding and predicting diseases”, *Social Computing and Behavioral Modeling*, pp. 1-8. doi: <https://doi.org/10.1007/978-1-4419-0056-2_26>

[] P. Klimek, A. Kautzky-Willer, A. Chmiel, *et al*. (2015). “Quantification of diabetes comorbidity risks across life using nation-wide big claims data”, *PLoS Computational Biology*, vol. 11, issue 4, pp. 1-16. doi: https://doi.org/10.1371/journal.pcbi.1004125

[] J. Liu, J. Ma, J. Wang, *et al*. (Jan 29, 2016). “Comorbidity Analysis According to Sex and Age in Hypertension Patients in China”, *International Journal of Medical Sciences*, vol. 13, issue 2, pp. 99-107. doi: <https://doi.org/10.7150%2Fijms.13456>

[] J. MacQueen. “Classification and analysis of multivariate observations”, *5th Berkeley Symposium on Mathematical Statistics and Probability,* vol. 4. L. M. Le Cam and J. Neyman, Eds. Los Angeles, USA: University of California, 1967, pp. 281-297.

[] J. A. Hartigan and M. A. Wong. (1979). “Algorithm AS 136: A K-Means Clustering Algorithm”, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, issue 1, pp. 100-108. doi: https://doi.org/10.2307/2346830

https://link.springer.com/article/10.1007/s10115-021-01623-y

<https://experts.umn.edu/en/publications/mining-interpretable-and-predictive-diagnosis-codes-from-multi-so> - book ( or <https://epubs.siam.org/doi/epdf/10.1137/1.9781611973440.120> )