

University of Reading

Department of Computer Science

**Mining Comorbidity 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*

02/05/2023

**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

02/05/2023

**Abstract**

…

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

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

**Code repository**: <https://github.com/paperkatana/CS3IP16-comorbidity-cluster-analysis>

**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, Dad, Nanny and Wayne 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**

Table of Contents

1 Introduction1

1.1 Background2

1.2 Problem statement2

1.3 Aims and objectives2

1.4 Solution approach2

1.5 Summary of contributions and achievements2

1.6 Organisation of the report2

2 Literature Review4

2.1 Review of existing research2

2.1.1 Existing research into comorbidity scores3

2.1.2 Existing research into co-occurrence clustering3

2.2 Critique of the review2

2.3 Summary2

3 Methodology1

3.1 The MIMIC-III Clinical Database2

3.2 Data preparation2

3.3 Clustering algorithms2

3.3.1 k-Means Algorithm3

3.3.2 Mini-Batch k-Means Algorithm3

3.3.3 M-Algorithm3

3.4 Evaluation metrics2

3.4.1 Calinski-Harabasz Index3

3.4.2 Davies-Bouldin Index3

3.4.3 Sum of Squared Error3

3.4.4 Relative Risk3

3.4.5 Scatter and Line Plots3

3.5 Summary2

4 Results1

4.1 k-Means Algorithm2

4.2 Mini-Batch k-Means Algorithm2

4.3 M-Algorithm2

4.4 Summary2

5 Discussion and Analysis1

5.1 Interpreting the results2

5.1.1 k-Means Algorithm3

5.1.2 Mini-Batch k-Means Algorithm3

5.1.3 M-Algorithm3

5.2 The optimal value of k2

5.3 Significance of the findings2

5.4 Limitations2

5.5 Summary2

6 Conclusions and Future Work1

6.1 Conclusions2

6.2 Future work2

7 Reflection1

References 1

**Appendices**

A K-means clustering diagrams1

B Mini-batch k-means clustering diagrams1

C M-algorithm diagrams1

**List of Figures**

2.1 Srinivasan et al’s co-occurrence equation2

3.1 Flowchart representing the data cleaning process2

3.2 Relative risk equation2

3.3 Calinski-Harabasz Index equation2

3.4 Davies-Bouldin Index equation2

3.5 SSE equation2

3.6 Scatter plot representing the distribution of discharged/deceased admissions2

4.1 Metrics line plots for the k-means algorithm2

4.2 Metrics line plots for the mini-batch k-means algorithm, batch size 1002

4.3 Metrics line plots for the mini-batch k-means algorithm, batch size 5002

4.4 Metrics line plots for the mini-batch k-means algorithm, batch size 10002

4.5 Metrics line plots for the mini-batch k-means algorithm, batch size 50002

4.6 Metrics line plots for the M-algorithm2

5.1 Sub-sections of the six scatter plots for k=402

**List of Tables**

3.1 ICD-9 code classifications2

4.1 Metrics table for the k-means algorithm2

4.2 Metrics table for the mini-batch k-means algorithm, batch size 1002

4.3 Metrics table for the mini-batch k-means algorithm, batch size 5002

4.4 Metrics table for the mini-batch k-means algorithm, batch size 10002

4.5 Metrics table for the mini-batch k-means algorithm, batch size 50002

4.6 Metrics table for the M-algorithm2

**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-9/10 – International Classification of Diseases, ninth/tenth revision

LASSO – Least Absolute Shrinkage and Selection Operator

MBK – Mini-Batch K-Means algorithm

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**

Disease co-occurrence studies investigate the relationships between different diseases in order to develop more effective treatments, and to draw conclusions on causation. Studying disease co-occurrences helps develop an understanding of common relationships between diseases and conditions, as well as to identify patterns in diagnosis. It has the useful application of allowing us to make predictions about future health outcomes based on current trends in an individual’s conditions. Co-occurrence analysis can also highlight uncommon or unexpected relationships between diseases, or show a change in co-occurrence patterns over time, which can provide insights and breakthroughs in treatment or specialised care.

Co-occurrence can be classified as comorbidity and multimorbidity. Multimorbidity refers to an individual’s presentation of two or more conditions; comorbidity is the presence of an index, or primary condition, and two or more other conditions. As such, the difference between the two classifications is that comorbidity represents the relationship between the primary and secondary conditions, whereas multimorbidity represents the presence of multiple conditions, often without emphasis on a primary condition.

A common method for investigating co-occurrence is through clustering analysis; clustering groups data such that all data points within a cluster are similar, and dissimilar to data points in other clusters. Clustering diseases can demonstrate similarities and relationships between them, allowing us to further our understanding of why and how they occur and affect health outcomes. Typically, clustering analysis on diseases is performed on the population – i.e.: clustering individuals’ conditions, health factors, demographic and lifestyle information, etc. – or the condition – conditions diagnosed for an individual, shared symptoms and causes, similar health projections or evaluated ‘seriousness’, and so on.

Advances in the field of data mining has given way to comprehensive methods of clustering analysis. There exist a great number of clustering algorithms that allow for clustering of data for any size and form, and different characteristic outcomes depending on one’s desired result. Of the five main categories of clustering algorithm, model-based, partitioning and agglomerative algorithms the most commonly used in the field of disease clustering, due to their visual capabilities and ability to represent relationships and quantify their significance.

* 1. **Problem statement**

Existing research on clustering diseases for comorbidity typically has a focus on a subset of conditions, which could potentially ignore comorbidities from outside of the selected conditions. For instance, there is a recognised comorbidity between kidney disease and cardiovascular disease, despite belonging to separate classifications of disease, due to common risk factors such as diabetes and hypertension, and research that has shown a diagnosis of kidney disease puts an individual at greater risk of being diagnosed with cardiovascular disease, and vice versa. Despite this, there is limited published research into generalised clustering across multiple classifications of disease.

At present, model-based algorithms are most predominantly used in research relating to disease co-occurrence. However, the use of partitioning algorithms comparatively offers increased speed and scalability, as they can handle large amounts of data, and produce results in much less time. With a large dataset, a model-based algorithm can suffer negatively in terms of interpretability, as it often represents clusters graphically with connections between data points, which can become confusing or overpopulated for a large amount of data. Partitioning algorithms are much simpler; for data of a considerable size, clusters grouped by lists of data points is much easier to work with. In addition, partitioning algorithms can handle a combination of categorical or continuous data which can be useful in application to disease co-occurrence, as factors like symptoms and projections can be used as features in analysis alongside categorical diagnosis data.

* 1. **Aims and objectives**

The main aim is to perform a comprehensive analysis across all classifications of disease present in the dataset. The data used comes from admissions to an intensive care unit registry; all diagnoses are given in the form of ICD-9 diagnostic codes. Clustering diseases using admissions containing a diagnosis of any disease classification, excluding other classifications such as symptoms and complications, should demonstrate similarities between diseases without restricting them to the category they fall under. Clustering will be based on admission rather than patient, in order to cluster across conditions instead of across population. Additionally, I aim to consider how the outcome of the admission is associated with the identified comorbidity, in order to assess the severity of comorbid diseases in the context of individual mortality.

A secondary goal is to demonstrate the usefulness of partitioning algorithms in clustering data. In order to do this, I will evaluate the performance of several partitioning algorithms on the same data to determine their credibility as useful in the domain of disease co-occurrence research. Three variations of partitioning algorithm will be used: the k-means algorithm, the mini-batch k-means algorithm, and an adaptation of the M-algorithm proposed by Sieranoja & Fränti [11]. Performance will be quantified through multiple intrinsic measures of clustering, and discussion of the visual representation and logical comparison of the results versus expectations will provide additional insights into how well each algorithm has performed.

* 1. **Solution Approach**

Using tables from the MIMIC-III healthcare database, handled in Python by the pandas library, the resulting program should produce a feature array from the tables’ information, containing patient and admission identifiers, an indicator of whether the admission resulted in discharge or death, the primary diagnosis and any subsequent secondary diagnoses. Each of the three partitioning algorithms should be implemented, and perform clustering of primary-secondary diagnosis pairs. The algorithms should cluster the data multiple times with different numbers of clusters, the results of which will inform the decision on the optimal number of clusters present in the data.

The expected results are a number of metrics calculated for each algorithm for each number of clusters calculated, as well as the partitioned groups of diseases. The metrics to be used are the Calinski-Harabasz Index, Davies-Bouldin Index, sum of squared error, and a relative risk calculation based on Srinivasan et al’s co-occurrence correlation [6]. Using this information, decisions can be made on the appropriate number of clusters for grouping the diseases in the data, and evaluating how each algorithm performed in comparison to each other.

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

This project demonstrates the merit of partitioning algorithms for clustering disease comorbidity, and compares the performance of the k-means algorithm, mini-batch k-means algorithm, and an adapted implementation of the M-algorithm, on a large dataset.

In addition, [describe major results and their implications].

* 1. **Organisation of the report**

This report is organised into seven chapters, of which the introduction is the first. Chapter 2 discusses existing research in the area of clustering analysis for disease co-occurrence. In chapter 3, the project’s methodology is broken down into the dataset and data cleaning, and an explanation of each of the chosen algorithms and metrics. Chapter 4 displayers the results, with discussions made on them in chapter 5. Chapter 6 draws the report to its conclusion, with considerations of the project’s future applications, while chapter 7 offers a reflection of my learning experience. An additional appendix chapter is provided with the graphical results of each clustering algorithm for each number of clusters.

**Chapter 2**

**Literature Review**

**2.1 Review of existing research**

**2.1.1 Existing research into comorbidity scores**

Early research into clustering for co-occurrence of diseases by Cornell et al [1] 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” [1, pp. 163]. Their work was among the first to aim to cluster specific groups of diseases, rather than generalised disease clustering across a population. 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 [2] 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 [3] 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 [4] 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.

S. K. Ng [5] 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 [6] 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” [6, pp. 1970]. 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 [13, 14, 15, 16] demonstrate a bias towards rare diseases, so Srinivasan et al developed the co-occurrence correlation metric:

*Figure 2.1: 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**

García-Olmos et al [7] 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 [8] 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 [9]. 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 [10] 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 [4] 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 Sieranoja & Fränti [11]. 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 [12] 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**

Notably, a lot of the existing research into clustering comorbidities and multimorbidities aims to cluster based on individuals rather than based on diseases [2, 4, 6, 10, 12]. 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 [10], or for smaller datasets [8]. 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**

This chapter considers existing research within the domain of clustering analysis for disease co-occurrence. In particular, research relating to comorbidity scores, and research relating to methods of clustering, were discussed.

**Chapter 3**

**Methodology**

**3.1 The MIMIC-III Clinical Database**

The MIMIC-III Clinical Database [17] 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 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 preparation**

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

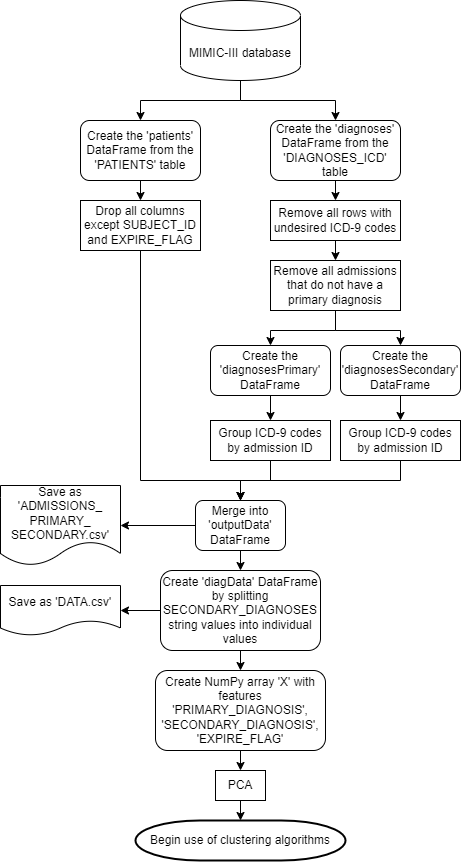
The information needed for clustering includes patient identifier, admission identifier, the primary diagnosis for the admission, a comma-separated list of subsequent secondary diagnoses and an entry indicating whether the patient was discharged or diseased at the end of the admission. Clustering analysis 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.

*Table 3.1: a table representing the grouping of ICD-9 codes [18]*

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

Considering disease comorbidities, it is necessary to remove any entries that contain diagnoses of any other category. Codes 290-319, 630-679, and 740-999, as well as the E and V codes, indicate conditions, symptoms and injuries rather than diseases. As the problem focuses on diseases, it is sensible to remove any entries containing these codes. This also results in a much smaller number of datapoints, from around 950,000 rows originally, and after removing unnecessary rows there are only 372,715. 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.



*Figure 3.1: a flowchart detailing the breakdown of steps for data cleaning*

**3.3 Clustering algorithms**

**3.3.1 k-Means Algorithm**

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 [19] and developed further by Hartigan & Wong [20], 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 |

Python’s scikit-learn library provides a simple and high-performing implementation of the k-means algorithm. For this project, a random state of 0 is used, and all other parameters keep their default values. A list of k values, K=[10, 40, 80, 100, 150, 200, 250, 300, 400, 500] is used to run the k-means algorithm with the dataset, producing 10 different groupings of clusters for the data.

**3.3.2 Mini-Batch k-Means Algorithm**

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 approach 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 |

Much like with the k-means algorithm, scikit-learn’s implementation of the mini-batch k-means algorithm is used, 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 an additional list batchSize = [50, 100, 500, 1000] for batch size.

**3.3.3 M-Algorithm**

The final partitioning algorithm implemented is a variation of Sieranoja & [Fränti](https://link.springer.com/article/10.1007/s10115-021-01623-y#auth-Pasi-Fr_nti)’s M-algorithm [11], which addresses the k-means algorithm’s tendency to get stuck on a local optimum. The M-algorithm is a hybrid model-based and partitioning algorithm; an adapted form of this algorithm is implemented, that is singularly a partitioning algorithm. It would be computationally 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.

The adapted 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 [12]. The sum of the SSE values for each data point is used 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, Srinivasan et al’s co-occurrence correlation [6] is also used, adapted to introduce a multiplication constant, d, to double the relative risk score if the patient was recorded as deceased at the end of a given admission. In this case, d is determined to be 2 if the admission has an event flag of 1, where 0 denotes the patient instead being discharged. On instances where the event flag is 0, d is 1. This change is included to allow one 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.

*Figure 3.2: the 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 the dataset has no weights, and therefore no densities to sort by, the centroids are instead initialised 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. The threshold is set 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, the overall SSE for the cluster set is computed at the beginning of the algorithm, as well as the overall relative risk. After splitting and merging clusters, the overall SSE and relative risk are calculated 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. The SSE values are compared first, as the goal is to minimise the distance between data points in each cluster. The relative risk is a secondary metric used to handle sets with the same or 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. The k-means algorithm is the same as in 3.3.1, as is K.

**3.4 Evaluation metrics**

The silhouette score is typically the metric used most to evaluate clustering results [21]. However, it has a great computational cost for a large amount of data, so it is not used. Instead, other intrinsic measures of clustering are utilised; namely, the Calinski-Harabasz Index, Davies-Bouldin Index, and SSE.

**3.4.1 Calinski-Harabasz Index**

The Calinski-Harabasz Index (or Variance Ratio Criterion) [21, 22] is the ratio of between-cluster dispersion against within-cluster dispersion, measuring dispersion as the total sum of squared distances from the centroid to each data point.

*Figure 3.3: Calinski-Harabasz Index equation, where nq is the number of data points in cluster q; cq is the center of cluster q; nE is the total number of data points; cE is the center of all data points; and k is the number of clusters.*

It is a useful metric for defining whether your clusters are well-separated, and has the additional benefit of being fast to compute, making it an efficient cluster measure. A higher value indicates that each cluster is far away from other clusters in the set, making for a well-defined cluster.

**3.4.2 Davies-Bouldin Index**

The Davies-Bouldin Index [21, 23] is an average of the measure of the similarity of each cluster with its most similar cluster, in terms of the size of each cluster against the distance between them. However, the index is limited in that it only computes the Euclidean distance between clusters.

*Figure 3.4: Davies-Bouldin index equation, where si and sj are the average distance between each point in clusters i and j respectively to cluster centers ci and cj respectively; dij is the distance between cluster centers ci and cj; and k is the number of clusters.*

One should aim for a lower score, preferably below 0, as this indicates a cluster is small compared to the other cluster. This should indicate a well-defined cluster, depending on the context of the application.

**3.4.3 Sum of Squared Error**

The Sum of Squared Error, or SSE, for the cluster set is the sum of all the squared distances between each data point and its centroid. It is a useful measure of how well-defined the clusters are in terms of how close the data points are to each other.

*Figure 3.5: SSE equation, where x is a data point within cluster ci­; and mi is the cluster centroid for cluster ci.*

A lower score indicates better clustering, as it reflects that all the data points are comparatively close to their cluster centroid.

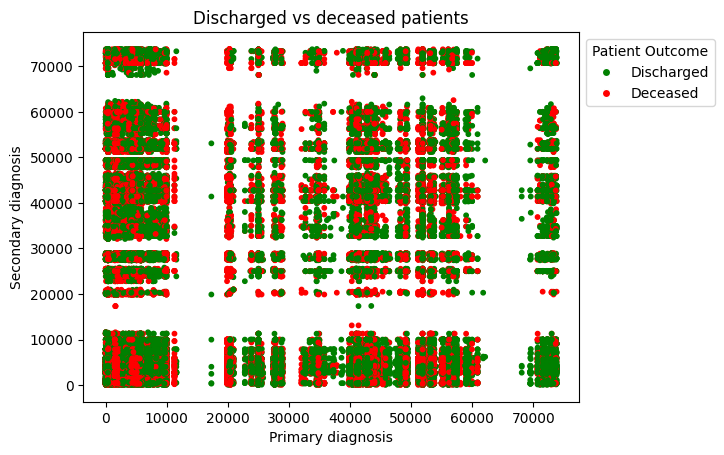
**3.4.4 Relative Risk**

Additionally, the relative risk equation implemented for the M-algorithm (figure 3.2) can be used to measure the total relative risk for the cluster set, as the sum of each primary-secondary diagnosis pair’s relative risk within each cluster. A higher value will indicate clusters that have greater relative risks, indicating the diagnoses within each cluster have a stronger relationship.

**3.4.5 Scatter and Line Plots**

As well as using mathematical measures to evaluate the clusters, it is also beneficial to visualise the results. For this, it is wise to create a scatter plot for each cluster set, distinguishing each cluster from each other by plotting them in different colours. In addition, the diagnosis pairs that resulted in a discharge are plotted with a ‘o’ marker, and those resulting in a death are plotted with a ‘x’ marker.

The following scatter plot shows the distribution of admissions resulting in discharge and death, in green and red respectively.



*Figure 3.6: a scatter plot showing the distribution of admissions that resulted in discharge or death*

In addition to plotting the cluster distributions, the metrics discussed in 3.4.1-4 can be plotted as a line plot against all values of k. This will serve the purpose of showing any trends in the metrics as the number of clusters changes.

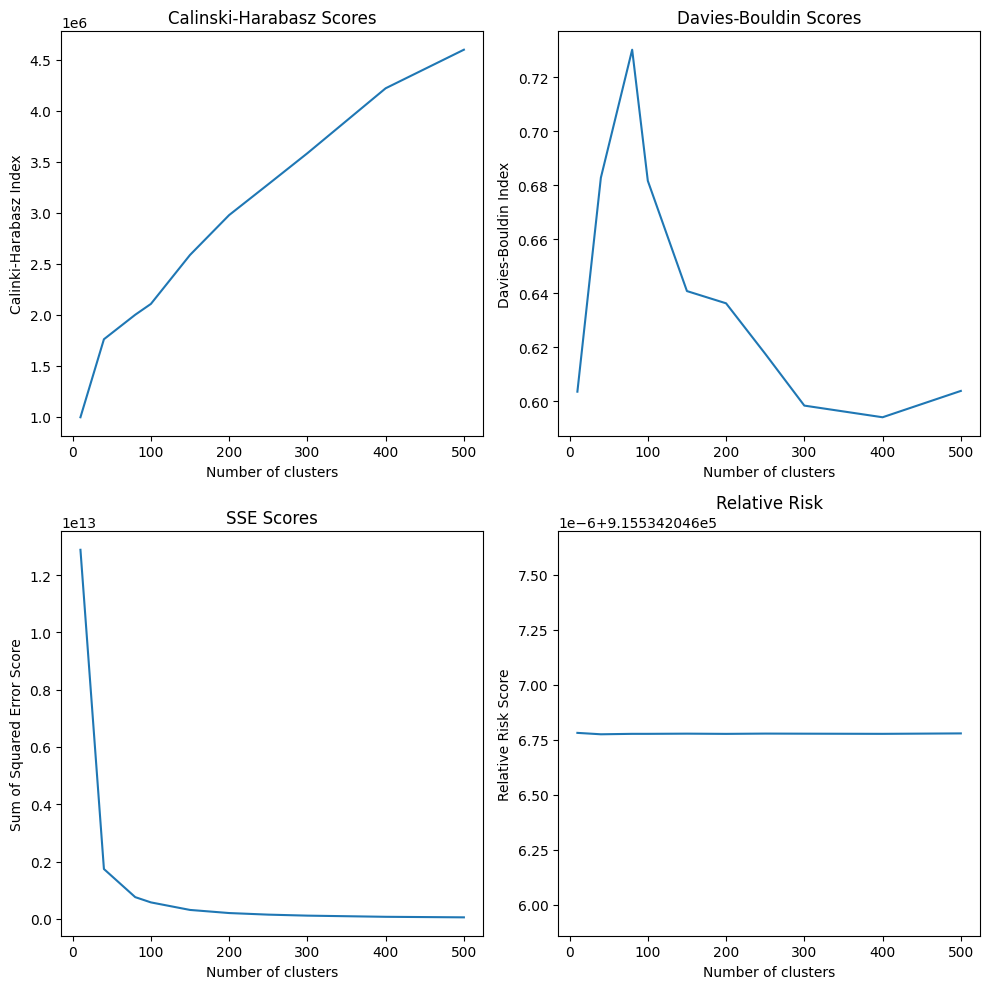
**3.5 Summary**

This section begins by providing details on the dataset used, the MIMIC-III clinical database, and how the desired data is gathered from this database to produce the dataset used for clustering analysis. It is followed by a breakdown of the three partitioning clustering algorithms used – the k-means algorithm, mini-batch k-means algorithm, and M-algorithm – with an explanation of how they work and pseudocode detailing the algorithm further. Finally, there is an explanation of the measures chosen to evaluate the results of each clustering algorithm – the Calinski-Harabasz Index, the Davies-Bouldin Index, the SSE, the relative risk, scatter plots and line plots.

**Chapter 4**

**Results**

**4.1 k-Means Algorithm**

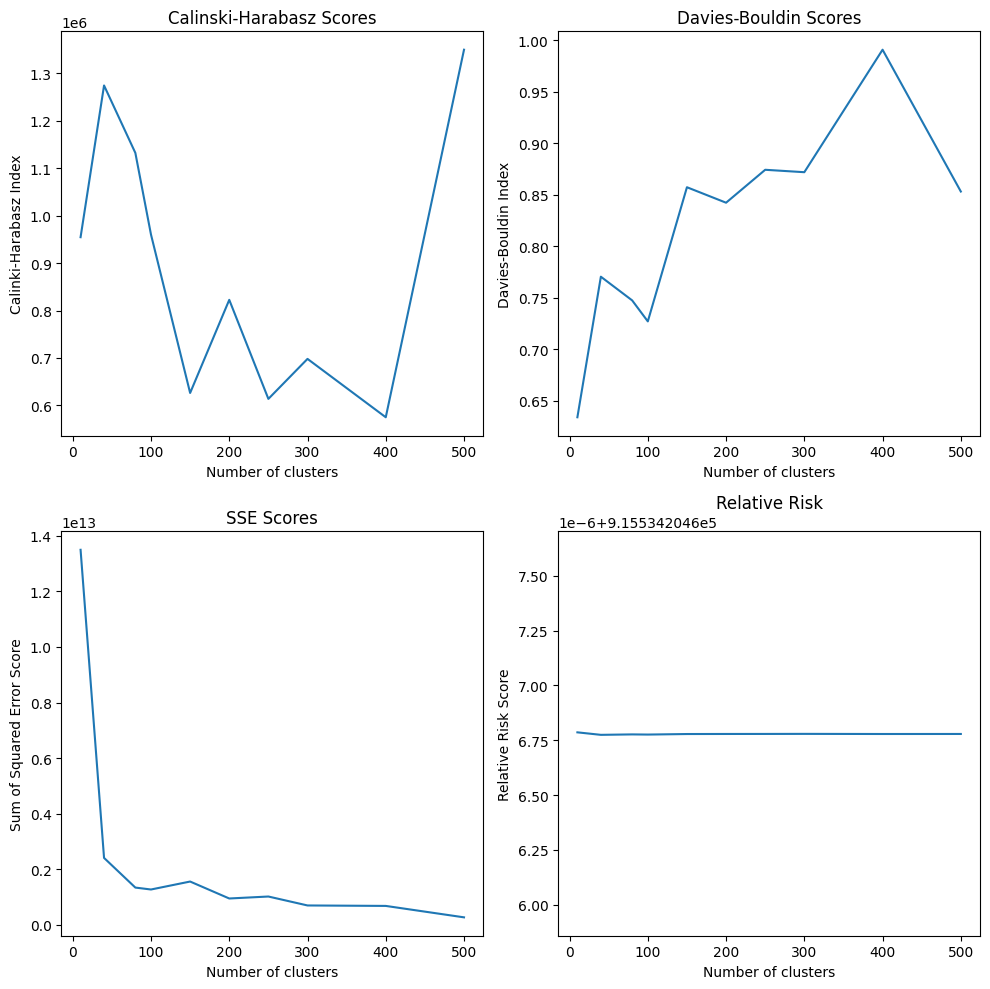
****

*Figure 4.1: line plots showing the metrics for each value of k for the k-means algorithm*

*Table 4.1: the metrics for each value of k for the k-means algorithm*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 992,686.025023 | 0.60357567 | 12,881,448,031,329.0 | 915,534.2045867814 |
| 40 | 1,757,993.53395 | 0.68277625 | 1,739,025,835,510.6216 | 915,534.2045867750 |
| 80 | 1,998,187.67795 | 0.73018170 | 757,588,269,307.8275 | 915,534.2045867770 |
| 100 | 2,104,401.37747 | 0.68164622 | 574,282,508,303.9340 | 915,534.2045867770 |
| 150 | 2,584,673.46494 | 0.64081820 | 310,883,870,013.4347 | 915,534.2045867778 |
| 200 | 2,975,852.49677 | 0.63629076 | 202,229,025,672.8697 | 915,534.2045867769 |
| 250 | 3,278,133.73005 | 0.61761544 | 146,719,968,214.9787 | 915,534.2045867781 |
| 300 | 3,582,512.69227 | 0.59841833 | 111,807,250,998.9380 | 915,534.2045867777 |
| 400 | 4,219,607.31778 | 0.59410182 | 71,117,384,220.8973 | 915,534.2045867770 |
| 500 | 4,597,972.63341 | 0.60385645 | 52,178,613,238.31792 | 915,534.2045867790 |

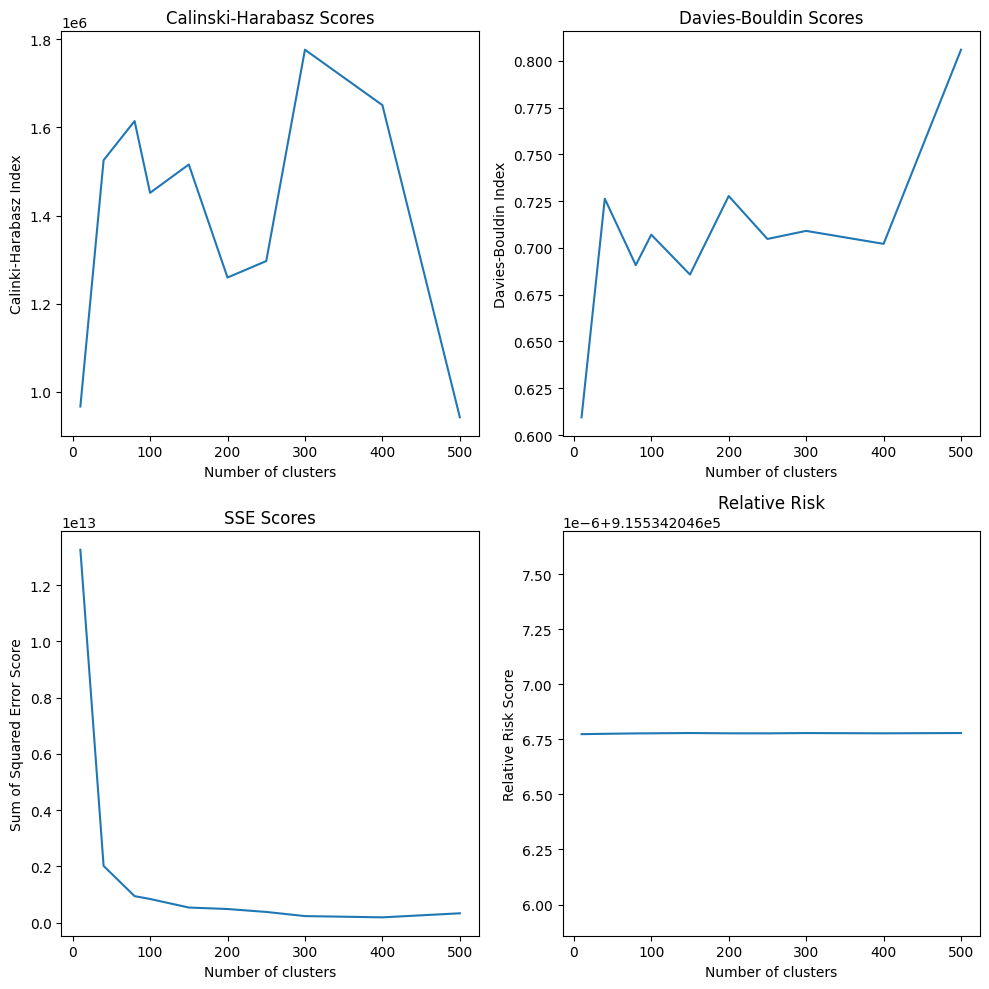
**4.2 Mini-Batch k-Means Algorithm**

****

*Figure 4.2: line plots showing the metrics for each value of k for the mini-batch k-means algorithm with 100 batch size*

*Table 4.2: the metrics for each value of k for the mini-batch k-means algorithm with 100 batch size*

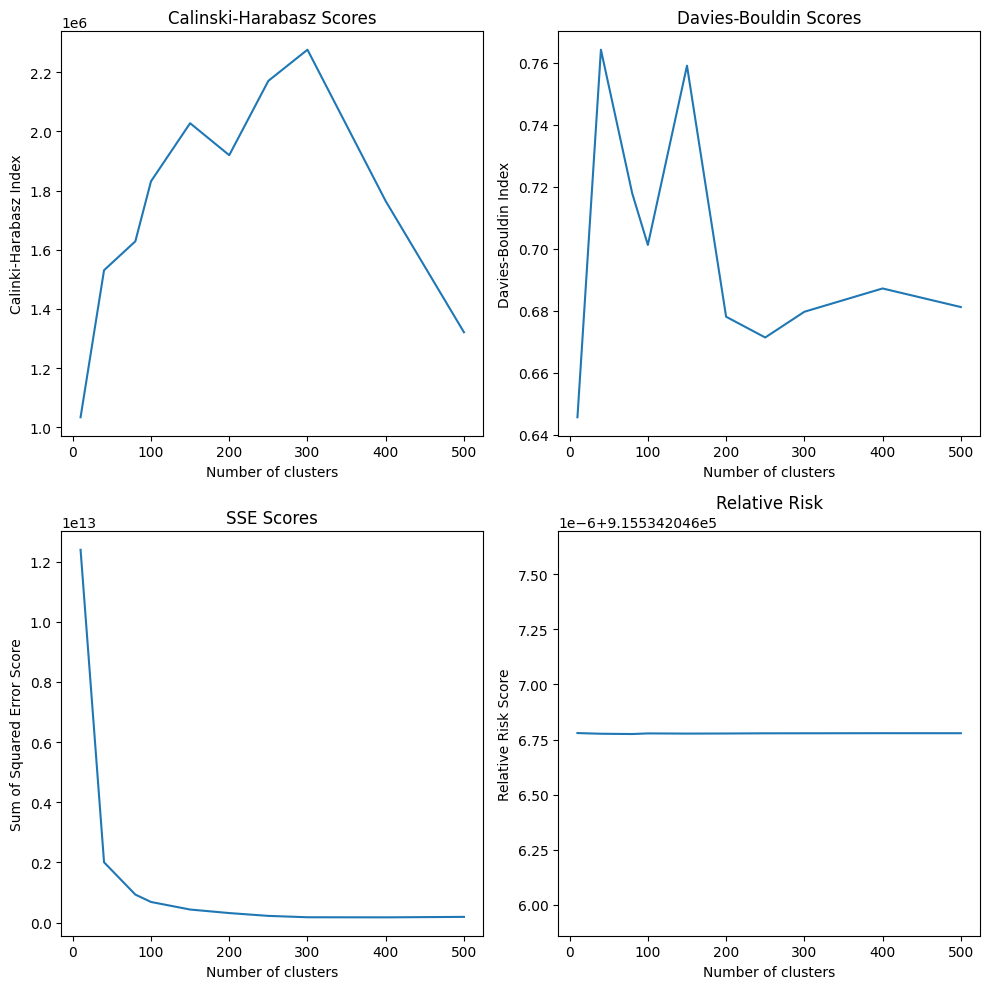
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 954,588.12629 | 0.63420883 | 13,498,610,976,458.8770 | 915,534.2045867860 |
| 40 | 1,274,373.40718 | 0.77066988 | 2,404,733,888,236.4287 | 915,534.2045867746 |
| 80 | 1,132,242.87737 | 0.74767325 | 1,336,602,121,415.7358 | 915,534.2045867766 |
| 100 | 960,035.93655 | 0.72728516 | 1,267,421,354,197.5317 | 915,534.2045867757 |
| 150 | 626,261.56460 | 0.85753277 | 1,554,018,109,365.5903 | 915,534.2045867782 |
| 200 | 822,528.28010 | 0.84250773 | 943,802,948,251.1692 | 915,534.2045867785 |
| 250 | 613,772.35317 | 0.87447480 | 1,014,661,824,310.7354 | 915,534.2045867787 |
| 300 | 698,071.91161 | 0.87212620 | 692,818,711,501.8611 | 915,534.2045867790 |
| 400 | 575,035.71482 | 0.99104875 | 676,297,518,512.3323 | 915,534.2045867784 |
| 500 | 1,349,754.31397 | 0.85335697 | 264,875,030,404.5039 | 915,534.2045867785 |

****

*Figure 4.3: line plots showing the metrics for each value of k for the mini-batch k-means algorithm with 500 batch size*

*Table 4.3: the metrics for each value of k for the mini-batch k-means algorithm with 500 batch size*

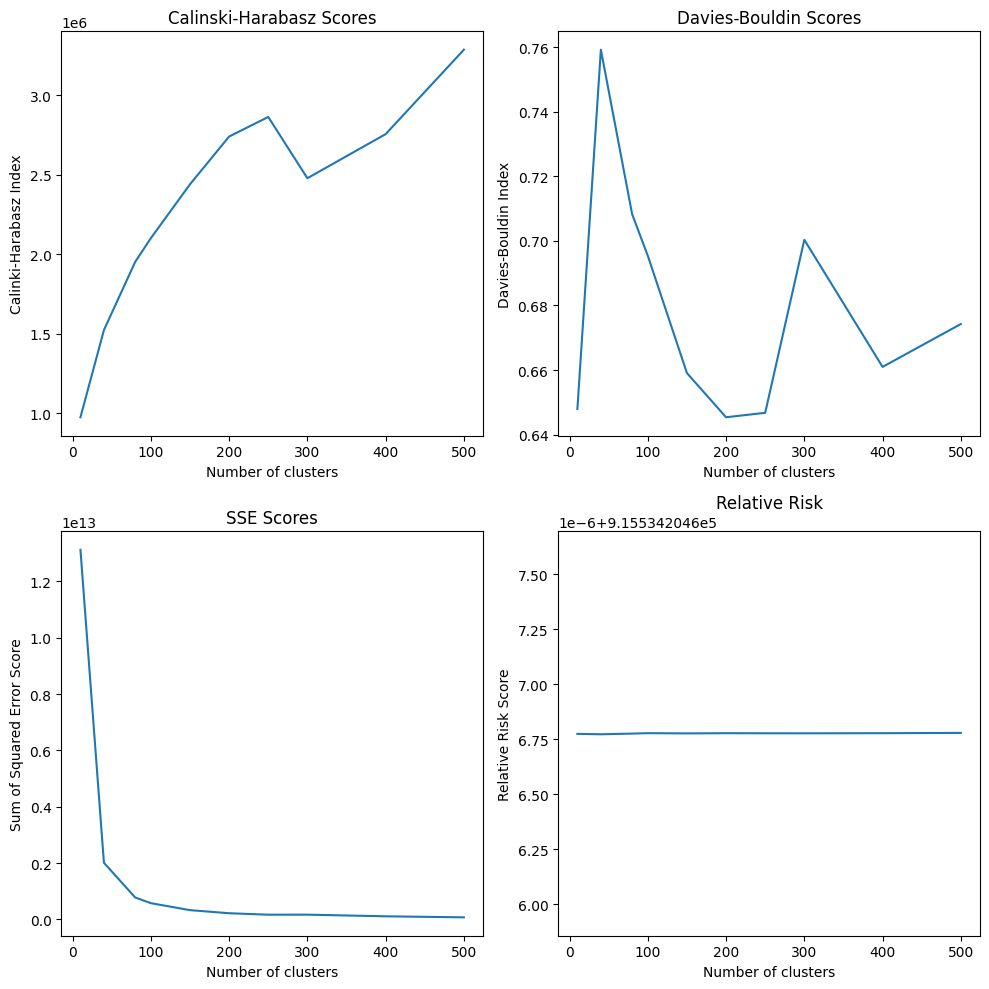
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 966,240.61877 | 0.60951355 | 13,261,036,930,300.8 | 915,534.2045867739 |
| 40 | 1,525,670.13846 | 0.72629784 | 2,013,810,159,912.8520 | 915,534.2045867754 |
| 80 | 1,614,498.44004 | 0.69080140 | 937,997,301,167.7832 | 915,534.2045867773 |
| 100 | 1,451,491.43655 | 0.70706615 | 837,213,450,973.9342 | 915,534.2045867776 |
| 150 | 1,515,808.32285 | 0.68577128 | 532,819,300,756.0957 | 915,534.2045867789 |
| 200 | 1,259,132.09641 | 0.72772611 | 479,917,837,827.8639 | 915,534.2045867776 |
| 250 | 1,296,531.30207 | 0.70479041 | 375,916,802,433.6248 | 915,534.2045867775 |
| 300 | 1,776,506.19508 | 0.70910035 | 228,391,279,148.0872 | 915,534.2045867788 |
| 400 | 1,650,637.68000 | 0.70217064 | 183,769,060,646.1679 | 915,534.2045867776 |
| 500 | 941,817.44117 | 0.80590884 | 327,732,716,801.8762 | 915,534.2045867789 |

****

*Figure 4.4: line plots showing the metrics for each value of k for the mini-batch k-means algorithm with 1000 batch size*

*Table 4.4: the metrics for each value of k for the mini-batch k-means algorithm with 1000 batch size*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 1,033,855.51721 | 0.645587756 | 12,389,858,252,008.023 | 915,534.2045867792 |
| 40 | 1,530,902.47044 | 0.764131420 | 2,003,941,133,868.7893 | 915,534.2045867762 |
| 80 | 1,628,597.59286 | 0.717802599 | 930,519,720,297.5264 | 915,534.2045867749 |
| 100 | 1,831,406.06068 | 0.701165583 | 684,926,494,280.8176 | 915,534.2045867777 |
| 150 | 2,027,855.70596 | 0.758978420 | 433,465,420,917.1153 | 915,534.2045867769 |
| 200 | 1,919,786.33225 | 0.678011575 | 315,978,697,667.8048 | 915,534.2045867774 |
| 250 | 2,171,054.05830 | 0.671316340 | 222,174,814,447.3959 | 915,534.2045867782 |
| 300 | 2,276,343.04868 | 0.679611815 | 176,578,400,971.2640 | 915,534.2045867783 |
| 400 | 1,764,472.86465 | 0.687128651 | 173,780,605,492.2547 | 915,534.2045867787 |
| 500 | 1,033,855.51721 | 0.645587756 | 12389858252008.0230 | 915,534.2045867792 |

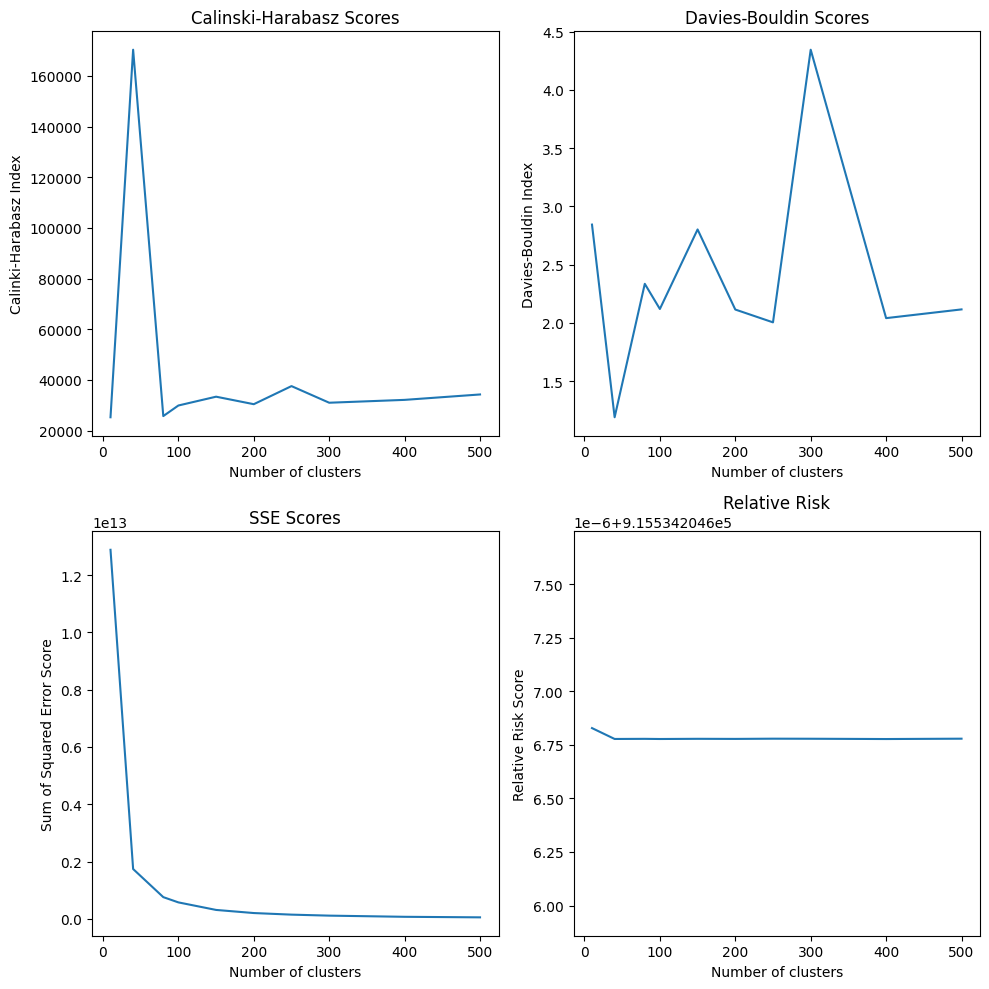
****

*Figure 4.5: line plots showing the metrics for each value of k for the mini-batch k-means algorithm with 5000 batch size*

*Table 4.5: the metrics for each value of k for the mini-batch k-means algorithm with 5000 batch size*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 974807.78783 | 0.64793088 | 13123072353201.078 | 915,534.2045867749 |
| 40 | 1523126.50683 | 0.75920506 | 2011732377897.8325 | 915,534.2045867731 |
| 80 | 1952372.14456 | 0.70823936 | 778293685708.7711 | 915,534.2045867761 |
| 100 | 2101272.34382 | 0.69537794 | 577988578523.9803 | 915,534.2045867782 |
| 150 | 2440116.07899 | 0.65905153 | 330130667411.8332 | 915,534.2045867770 |
| 200 | 2740250.73999 | 0.64536727 | 220450324095.919 | 915,534.2045867780 |
| 250 | 2863400.53647 | 0.64674435 | 168715052761.9589 | 915,534.2045867776 |
| 300 | 2478185.06665 | 0.70032895 | 170150346390.1688 | 915,534.2045867774 |
| 400 | 2755425.69826 | 0.66096458 | 111386600126.2731 | 915,534.2045867781 |
| 500 | 3286360.71174 | 0.67425116 | 74618423006.0993 | 915,534.2045867793 |

**4.3 M-Algorithm**



*Figure 4.6: line plots showing the metrics for each value of k for the M-algorithm*

*Table 4.6: the metrics for each value of k for the M-algorithm*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| 10 | 25,245.95350 | 2.84361198 | 12,881,448,031,329.0 | 915,534.2045868279 |
| 40 | 170,332.66634 | 1.19089891 | 1,739,025,835,510.6213 | 915,534.2045867771 |
| 80 | 25,711.80608 | 2.33554245 | 757,588,269,307.8274 | 915,534.2045867777 |
| 100 | 29,887.30135 | 2.11965555 | 574,282,508,303.9341 | 915,534.2045867770 |
| 150 | 33,366.08032 | 2.80197646 | 10,883,870,013.4348 | 915,534.2045867778 |
| 200 | 30,403.75221 | 2.11503260 | 202,229,025,672.8698 | 915,534.2045867775 |
| 250 | 37,543.26085 | 2.00502199 | 146,719,968,214.9786 | 915,534.2045867785 |
| 300 | 30,982.34389 | 4.34385665 | 111,807,250,998.938 | 915,534.2045867782 |
| 400 | 32,110.03744 | 2.04125943 | 71,117,384,220.8973 | 915,534.2045867769 |
| 500 | 34,244.58071 | 2.11615730 | 52,178,613,238.3179 | 915,534.2045867785 |

**4.4 Summary**

This chapter shows the corresponding Calinski-Harabasz Index, Davies-Bouldin Index, SSE and relative risk for each value of k (and batch size 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 Interpreting the results**

To review, a higher Calinski-Harabasz score indicates clusters that are well-separated and far from each other, as measured by the dispersion of data points within each cluster and against the other clusters in the set. As for the Davies-Bouldin score, a lower value is preferred. A good score is smaller than 1.0, indicating the clusters to be small in comparison to each other. The SSE represents how far apart data points within a cluster are from the cluster centroid, and as such, from each other. Good clusters have all the data points within a cluster close to each other and distant from other clusters, so a small SSE value is better. The relative risk is a measure of how often the diseases represented by each data point’s primary-secondary diagnosis pair appear together and overall, and the value is affected by whether the data point’s admission outcome was death or discharge. A higher relative risk score indicates a more significant relationship.

Therefore, what would be deemed a “good” set of clusters is one with a high Calinski-Harabasz index, a low Davies-Bouldin index, a comparatively small SSE, but not one too close to zero, and a large relative risk value.

To decide on the optimal number of clusters, the best approach is to consider the individual results of each algorithm and draw conclusions on the most appropriate number(s) of clusters at this level; then, consider the conclusions drawn from each algorithm to make an overall decision on the best solution.

The k-means algorithm appears to have performed the best on this data, from visually comparing the metrics’ plots for each algorithm. The M-algorithm performed the least when considering the Calinski-Harabasz and Davies-Bouldin Indexes, as it produced the lowest and highest values, respectively. This is to be expected, as the algorithm aims to improve the SSE and relative risk scores by splitting and merging clusters. As a consequence of improving the intra-cluster scores, the inter-cluster scores will decline.

Considering each of the four batch sizes implemented with the mini-batch k-means algorithm, the greater the batch size, the better the algorithm performed at clustering the data, shown by comparing each of the line plots and noting the trend of the line is smoother as the batch size increases.

Considering the scatter plots (see Appendix), roughly the same groupings were made, with a few small differences in data points towards the edges of clusters. This is particularly true when comparing the performance of the mini-batch k-means algorithm for different batch sizes. For instance, consider the top left corner of all six scatter plots for 40 clusters:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

*Figure 5.1: a section of the k=40 scatter plots for the k-means algorithm, mini-batch k-means algorithm with batch size 100, 500, 1000 and 5000, and the M-algorithm, respectively*

The colours themselves are not significant, due to the nature of the ColorMap used picking 600 random colours with which to plot different clusters. However, the shapes of each cluster, denoted by the different colours, is significant, as it is demonstrated that no sets of results are identical. There is a a collection of data points at the top of the figures, which all algorithms labelled as part of the same set. However, the k-means algorithm decided that additional datapoints underneath also belong to the same set, which no other algorithms did. The MBK algorithm with batch sizes of 100 and 1000 produces more diagonal groupings, but the other algorithms keep the clusters in a more uniform shape. For the set of data points in the centre of this section, on either side of the horizontal line with no data points present, the k-means and M-algorithms decided these points all belong to the same cluster set, but the MBK algorithm decided each time that those points must be in two sets. A consideration of the ICD-9 codes represented by these data points, for each cluster, could draw conclusions on which of these clusters are the most appropriate.

However, it is notable that as the number of clusters increases, new clusters are found by splitting the larger clusters, rather than forming new clusters from the boundaries of multiple existing clusters. This demonstrates a dataset with relatively well-defined groupings in the data.

**5.1.1 The k-Means Algorithm**

Looking at the plotted results for this algorithm’s performance (figure 4.1), the graphs demonstrate that the k-means algorithm performed the most consistently. The Calinski-Harabasz index trends almost consistently upwards as k increases, suggesting that the larger the value of k, the better the clusters. The SSE demonstrates similarly consistent results, with a very high value for k less than 40, but dropping sharply and then consistently decreasing as the value of k gets larger. This agrees with the results demonstrated by the Calinski-Harabasz scores.

The relative risk does not change by a large amount for different values of k, as the data being calculated is the same, and the only changes come from how the data is grouped, which results in a negligible change of relative risk value for all algorithms, around 2 to 3 decimal places difference. However, these differences are still indicative of a better or worse performance. The values of k producing the greatest relative risk scores are the extremes of 10 and 500, followed by 250, 300 and 100 clusters. This suggests the best value of k can be found around these values.

As for the Davies-Bouldin index results, the value is large for around k=100. The values are much lower after this, with the best results between 300 and 400. This suggests the number of clusters should be greater than 100, and preferably around 300-400, for clusters that are dissimilar to each other.

Drawing these considerations together, the results of the k-means algorithm evaluation suggests that the optimal value of k is around 300 clusters.

**5.1.2 The Mini-Batch k-Means Algorithm**

The MBK algorithm generally performed better with a larger batch size, as shown by larger overall Calinski-Harabasz scores and lower overall Davies-Bouldin and SSE scores as the batch size increases. The relative risk score is consistent for all batch sizes investigated.

The relative risk was consistently high across all batch sizes for values of k between 200-300, and poorly for values around 40-100.

The trend of SSE scores for each of the four batch sizes was consistent, although the MBK algorithm with a batch size of 100 performed much worse, with overall larger values, and the values tended to go up and down rather than simply decreasing. However, like the k-means algorithm, the trend suggests a greater value of k is ideal.

The two smaller batch sizes did not perform well for the Davies-Bouldin index. As the number of clusters increased, so did this score. This is opposite to what one would expect. However, the algorithm performed much better for a greater batch size. The index was high for values less than 200, and dropped sharply for a larger value of k. For a batch size of 1000, the index stayed low for larger values; for a batch size of 5000, the index peaked again at around 300 clusters. Overall, for a better Davies-Bouldin index using the MBK algorithm, a larger batch size is preferred. Then, the best value of k should be found for values greater than 200.

Each batch size produced a widely different pattern of results for the Calinski-Harabasz index. A batch size of 100 performed poorly, with the lowest values. Its highest values were seen around 40-100 clusters. As for a batch size of 500, the scores were inconsistent but higher, with the greatest values around 300-400 clusters, but also performing well for 40-150 clusters. Otherwise, the scores were low. For 1000 batch size, the results were high for all values except 10 and 500. The higher the value of k, the better the score. The pattern for a batch size of 5000 was similar, except for a drop in score for 300 clusters; however, this algorithm performed well for 500 clusters. Considering all of this, the best values of k are likely to be found at around 300 clusters, unless the batch size is 100.

The greater the batch size, the better the performance. The consideration for the optimal value of k should be biased towards the results for a larger batch size. Drawing conclusions from the results and discussion, the number of clusters should either exceed 200, and should be around 300.

**5.1.3 The M-Algorithm**

The M-algorithm aims to optimise clusters at runtime using the intra-cluster measures of SSE and relative risk. Therefore, it is expected that this algorithm performed very well for these scores, achieving the lowest SSE values and highest relative risk values. However, it does not perform so well, or consistently, for the other two metrics. This indicates clusters that are too similar to each other – likely due to the fact that there was repeated cluster splitting and merging, affecting these indexes.

The graphs in figure 4.6 show low Calinski-Harabasz scores, and irregular Davies-Bouldin scores. This algorithm performed particularly badly for these indexes, with only one Calinski-Harabasz index exceeding 100,000, and all Davies-Bouldin indexes are greater than 1. However, these indexes show the best performance to be for 40 clusters, with the respective highest and lowest index values. Conversely, this number of clusters achieves the worse SSE score. These results are poor enough that it is difficult to draw conclusions from them regarding the optimal number of clusters.

Considering the SSE, which was the algorithm’s target score to reduce, the larger the number of clusters, the better the performance. This is mostly reflected by the relative risk scores too, although the best relative risk can be found for 250 clusters.

**5.2 The optimal value of k**

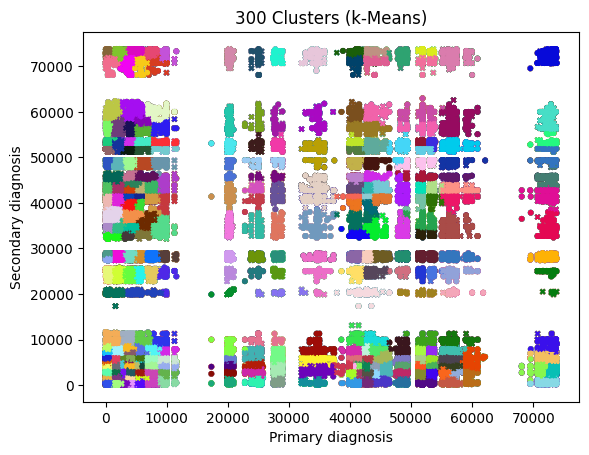
*Table 5.1: a table of summarised conclusions from 5.1*

|  |  |
| --- | --- |
| Algorithm | Conclusions around the best k value |
| K-means algorithm | 250<k<400; a large value of k |
| Mini-batch k-means algorithm | k>200; 300<k<400 |
| M-algorithm | K=250; a large value of k |

It is clear from the conclusions drawn from the results of each algorithm’s performance that the optimal number of clusters for this dataset lies at around 300. As such, the results for k=300 is as follows:

*Table 5.2: the metrics for each algorithm with a value of k=300*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Calinski-Harabasz Index | Davies-Bouldin Index | SSE | Relative risk |
| K-means | 3,582,512.69227 | 0.59841833 | 111,807,250,998.938 | 915,534.2045867777 |
| MBK, b=100 | 698,071.91161 | 0.87212620 | 692,818,711,501.8611 | 915,534.2045867790 |
| MBK, b=500 | 1,776,506.19508 | 0.70910035 | 228,391,279,148.0872 | 915,534.2045867788 |
| MBK, b=1000 | 2,276,343.04868 | 0.679611815 | 176,578,400,971.2640 | 915,534.2045867783 |
| MBK, b=5000 | 2478185.06665 | 0.70032895 | 170150346390.1688 | 915,534.2045867774 |
| M-Algorithm | 30,982.34389 | 4.34385665 | 111,807,250,998.938 | 915,534.2045867782 |

****

*Figure 5.3: a scatter plot showing the results of the k-means algorithm for 300 clusters*

The metrics in figure 5.2 clearly indicate that the k-means algorithm performed the best, with the largest Calinski-Harabasz index and smallest SSE and Davies-Bouldin index. Figure 5.3 shows the distribution of clusters for the k-means algorithm for k=300.

**5.3 Significance of the findings**

Within this project’s associated repository, a CSV file containing all primary and secondary diagnoses for each cluster can be found. The majority of clusters contain codes from the same ICD-9 classification. This indicates good groupings, as you would expect codes from the same classification to form clusters.

[NOTABLE RESULTS]

Evidence that the clustering has not been performed so well is demonstrated for ICD-9 codes on the boundaries of the classification’s range of codes. For instance, cluster [] first five primary diagnoses are []. Is there truly a relationship between [what are these diseases?], or has the algorithm grouped them because the actual numeric values are similar?

M Algorithm performed poorly – the adapted implementation was not a success. However, did I meet the aim to demonstrate that partitioning algorithms are good for this application?

Also consider the other aim – are the diseases clustered well? Also talk about how they’re clustered in comparison to health outcome

Is it actually significant? This code only sees numbers, not the meaning behind the codes. The only information it has to form relationships is the value of the number and the pair of primary and secondary codes – this is the only way that interesting cluster results will happen. The algorithms place too much value on the number, when that is only a small factor.

**5.4 Limitations**

One of the most considerable limitations regarding partitioning algorithms is that their clusters are typically spherical in shape. However, not all data clusters are spherical; this would imply uniform groupings of data. This may not be the case for the disease diagnosis data, as they are complicated and comorbidity can occur due to many risk factors, symptoms and complications. In general, this problem can be counteracted by reviewing the clustering results and applying common-sense to determine whether the groupings make sense, investigating any anomalies or unexpected results.

The implementation does not consider the possibility of overlapping clusters, where existing research has succeeded to do so [4, 12]. In the real-life context of disease comorbidity, there inevitably will be overlaps in clusters due to shared causes, risk factors and effects. For instance, liver disease is a comorbid condition of both hepatitis B and C, and also cardiovascular disease, which you would expect to find in different clusters due to an undefined relationship between these diseases. Failing to consider or implement overlapping clusters means this implementation of clustering analysis can only have limited applications.

The clusters have been defined based on pairs of primary and secondary diagnoses for a given admission, and paired with the outcome of the admission (deceased or discharged). While this is useful for a general analysis of disease comorbidity, it does not take into account other factors of comorbidity such as age, gender and lifestyle. As well as this, the data only represents diseases that were identified and diagnosed, but there may well be underlying conditions not identified in the patients included in the registry. All of this means that the resulting clusters must not be considered wholly accurate, as they are too generalised and do not account for information not present within the data.

**5.5 Summary**

**Chapter 6**

**Conclusions and Future Work**

**6.1 Conclusions**

Typically a conclusions chapter first summarizes the investigated problem and its aims and objectives. It summaries the critical/significant/major findings/results about the aims and objectives that have been obtained by applying the key methods/implementations/experiment set-ups. A conclusions chapter draws a picture/outline of your project’s central and the most signification contributions and achievements. A good conclusions summary could be approximately 300–500 words long, but this is just a recommendation.

**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**

There was a point in my project where I came completely undone due to decisions I was making and how I felt about making them. As part of my methodology, I originally decided to do my data extraction steps in PostgreSQL and the main process in Python, as the Python code was provided for my original project and what I was demonstrating was the data extraction section. Despite spending weeks trying to set up a virtual PostgreSQL environment on my laptop, trying every solution I could find and asking multiple people for their advice and recommendations, by the time I had successfully set up the environment and began the data extraction, I found I couldn’t figure out how to get the data into the format I had envisioned in my head.

I decided to just disregard the entire approach and instead do data extraction in Python. I was disappointed in myself for what felt like wasted time, and for just giving up on the method I had planned out. I couldn’t shake it off and I kept putting off making progress with my project as a result. I know that I made the decision to give up on the original method because I wasn’t confident in using PostgreSQL and I didn’t want to have to learn from scratch how to reach my goal. An option would have been that I could have tried harder to find a solution in PostgreSQL in order to avoid disappointing myself.

I didn’t start off in Python because at the beginning of the project I had never worked with Python at this level; at the point I switched method I had completed my Python for data science module and was now confident in working with pandas. At the time, I felt like I was wasting my own time and giving myself less work to present. Retrospectively, I was playing to my strengths, as I was more knowledgeable in one area and I would be better off enhancing that skill instead of trying to learn a new one. In reality, the cost of my own inward disappointment came as the price for changing my plans in order to perform better, and develop an existing skill rather than trying to form a foundation for a new skill. I can learn from this that changing my plan isn’t always a bad thing or something to be negative about, as I’m sure I wouldn’t change from the original intentions without good reason. The next time I feel like this, I will keep this particular situation in mind, because it turned out okay in the end.

When I started this project in September, I went into it with the notion that I was contributing to published research and ongoing further research. My project was building on a previous student’s project, which he further went on to publish with our supervisor, so I felt like that was my end goal too. I placed so much pressure on myself to produce something I considered “worthy” of being in the public domain, and my report, in my mind, had to look like the research papers I was reading for my literature review. It was making me feel ill that I couldn’t achieve my own impossibly high standards in a field I had previously never worked on, despite nobody else putting this pressure on me.

When I had to change my project because I wouldn’t have reached a conclusion on the original one, I adjusted my mindset to the new domain I was working in and realised I had made up the pressure. There didn’t actually exist this need to be perfect and academic, I just needed to get a good grade on my project in order to do well overall in my degree. That is to say, the consequences of a poor final year project would have no impact on my life outside of my degree, except maybe in future job applications I would have one less thing to talk about. I felt much better when I realised this, and it made it easier for me to work on the project. A lesson I need to learn is to think about my situation from outside of my own head, as my perception of the situation can be skewed by my ideals and standards.

**References**

[1] 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)

[2] 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>

[3] 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>

[4] 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>

[5] 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>

[6] 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

### [7] 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>

[8] 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>

[9] 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.

[10] 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>

[11] S. Sieranoja and P. Fränti, (Dec 4 2021). “Adapting k-means for graph clustering”, *Knowledge and Information Systems*, vol. 64, pp. 115-142. doi: https://doi.org/10.1007/s10115-021-01623-y

[12] 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>

[13] 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>

[14] 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>

[15] 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

[16] 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>

[17] 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>

[18] 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

[19] 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.

[20] 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>

[21] 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>

[22] T. [Caliński](https://www.tandfonline.com/author/Cali%C5%84ski%2C+T) and J. Harabasz. (1974). “A dendrite method for cluster analysis”, *Communications in Statistics*, vol. 3, issue 1, pp. 1-27. doi: <https://doi.org/10.1080/03610927408827101>

[23] D. L. Davies and D. Bouldin. (May, 1979). “A Cluster Separation Measure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, issue 2, pp. 224-227. doi: <http://dx.doi.org/10.1109/TPAMI.1979.4766909>

**Appendix**

**Chapter A – k-Means clustering diagrams**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Chapter B – Mini-batch k-means clustering diagrams**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Chapter C – M-algorithm clustering diagrams**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |