USING ENSEMBLE CLUSTERING TO IDENTIFY PHENOTYPES OF DIABETES PATIENTS FOR EVALUATING DISEASE PROGRESSION

by

Kayalvizhi Thanigainathan

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2022

To Mom, Dad and Sister!

Thank you for your constant support that made me see this adventure through to the end.

Table of Contents

List of Tables

List of Figures

Abstract

Diabetes Mellitus (DM) is a chronic health condition that affects multiple organs and is associated with significant morbidity and mortality. The reasons underlying the relative progression rate of DM remain poorly understood, and as such, its impact on different organs needs to be closely monitored. The management of diabetes requires periodic pathology investigations and physician examinations to manage the disease's progression. The ability to predict the temporal progression of the disease for a patient can significantly impact therapeutic choices and improve outcomes. This thesis presents investigations in stratifying diabetes patients based on their pathology test results in terms of temporally salient patient clusters. We investigated machine learning-based clustering methods and applied them to time-series data of pathology tests and their results to generate patient clusters. We applied an ensemble clustering approach where multiple base clustering algorithms were combined to generate a cluster ensemble that resulted in grouping the patients in three top-level clusters. Using the clusters, we generated patient phenotypes comprising clinical characteristics that are then used to develop classification-based prediction models to predict the disease's temporal progression and suggest the potential pathology tests. A key feature of this research is the generation of patient clusters without patient demographics and only using pathology test data for identifying the phenotypes. The ability to predict and understand disease progression will lead to novel personalized medicine for managing patients with diabetes.

Acknowledgements

I thank my supervisor Dr. Syed Sibte Raza Abidi for his constant guidance and encouragement, without which I would not have completed this thesis. His valuable suggestions and insightful comments have navigated my thesis in the right direction. I am extremely grateful to Dr. Patrice Roy for readily being available and addressing all my questions patiently. I would like to thank my parents Thanigainathan, Poonguzhali and my sister Sudarvizhi for their constant encouragement and support in every moment of my life. Finally, a special mention to Amruth Sagar Kuppili for bring my constant.

Chapter 1

Introduction

Diabetes is a global epidemic growing rapidly worldwide. As per the International Diabetic Federation, 463 million individuals were affected by diabetes in 2019, a figure that has significantly multiplied over the last 20 years, with projections of 578 million by 2030 and 700 million by 2045 [41]. Diabetes has the potential to cause numerous health complications, including ketoacidosis, Nerve disease, Kidney disease, eye disease, stroke, cardiovascular disease, and other complications [23]. Furthermore, persons with diabetes are at a higher risk of contracting infections. Many people with diabetes can delay the onset of complications with proper treatment and lifestyle modifications. To keep their health in check, diabetic individuals are subjected to a multitude of checkups regularly, including blood tests, eye, and dental exams, kidney functionality tests, and electrocardiogram [77]. Furthermore, the frequency of these tests may vary from patient to patient, depending on the physician's judgment and the patient's visit habits. Healthcare facilities collect these test reports digitally as Electronic Health Records(EHRs).

A typical EHR may contain demographic data, lab results from blood work or other tests, diagnostic imaging reports, medication history, immunization records, and discharge notes from any hospital stays. The adaption of EHR has brought an unprecedented amount of clinical information available for research. Analyzing EHRs provides the potential to improve patient care, improve the understanding of disease and clinical phenotypes, and generate knowledge for clinical decision support [49, 83]. These vast data require the use of data-driven approaches that can make sense of such large amounts of data, efficiently explaining possible latent associations and disease characteristics.

Moreover, the growing diabetes population demands the effective utilization of these EHRs in detecting the adverse effects of diabetes at an early stage. Many patients lose their life due to diabetic complications that may lead to multi-organ

and systemic injury, which impairs the quality of their life. The main issue with diabetic complications is that they are undetected in the early stages and can only be diagnosed if the patient exhibits symptoms [82]. Identifying the population at high risk of developing such complications can help intervene in preventative care early, thereby reducing the death rate. One possible way of identifying such individuals is by closely monitoring their disease progression using their pathological reports. For instance, a test called A1C can hint at the onset of micro-vascular complications [5]. Arguably, predicting these health consequences among the diabetic population before the patients get these tests done can reduce the healthcare burden of diabetes and its long-term complications and improve care pathways and prevention.

A significant volume of longitudinal data is accumulated for a diabetes patient spanning multiple years. Machine learning algorithms provide a viable approach to analyzing patient data to determine the progression of the disease and stratify patients accordingly to streamline their treatment. Machine learning aims to develop computer systems that can learn and respond from their prior observation. Machine learning is applied in various aspects of diabetes evaluation ranging from early prediction of the disease and complications to its management and other related clinical traits. Its insights also help clinicians promote disease awareness to afflicted individuals and establish peer support groups where they may closely monitor patients with similar health concerns and provide tailored care to those who are afflicted. The primary step to acquiring these benefits using machine learning is subtyping or phenotyping the patients using the data available.

EHR phenotyping is the process of identifying the individuals with a specific clinical condition or trait that may be ascertained via a computerized query from large quantities of imprecise clinical patient data and classifying them accordingly into retrospective cohorts. Phenotyping is the basis of translational research, comparative effectiveness studies, clinical decision support, and population health analyses using routinely collected EHR data [26]. Various machine learning approaches, especially cluster analysis, are applied for effective phenotype identification [98]. Cluster analysis assists in discovering key trends among patients through effective grouping using the key features available in the data. There are numerous successful applications of clustering in identifying distinct diabetes phenotypes with unique risk

profiles [39, 113, 22, 19]. Our study uses various clustering algorithms to identify the underlying pattern in the diabetes EHRs and group the patients to determine disease progression at a group level.

1.1 Research Objective

The project's primary objective is to analyze the information contained in EHRs to assist in the long-term management of diabetic patients; in particular, this thesis aims to group patients based on their clinical characteristics to identify patient phenotypes that can help understand the progression of diabetes and the incidence of diabetes-related complications in different patient groups. Diabetes patients undergo routine pathology tests to monitor the severity of the disease. The underlying approach is to perform a data-driven analysis of the time-series of pathology test results, conducted over a period of time for a patient, to abstract a temporal progression pattern of the disease and then group patients with similar disease trajectories to understand the underlying clinical characteristics of each patient group in order to develop patient phenotypes for different patient groups. We have also extended the research to predict the value of pathology test results before taking the test itself using the phenotypes identified and the knowledge obtained from the given data using classification techniques. This thesis will attempt to answer the following clinical and technical questions:

Diabetes related clinical questions:

- Is it possible to generate clinically meaningful patient groups on the basis of pathology test results without the use of demographic information?
- What pathology tests distinguish the different patient groups?

Machine learning related technical questions:

- In detecting clusters from diabetes data, is the consensus function more resilient and efficient than individual clustering models? If that is the case, which consensus function is beneficial?
- Which imputation approach effectively substitutes the best value for the diabetic patients' missing temporal characteristics?

- Which classification model can successfully learn the diabetes dataset's clustering structure and forecast the clusters of unknown patients?

1.2 Research Problem

A significant amount of research is conducted on a wide range of EHRs that comprise vital signs, test ordering trends, or discharge summaries. The majority of the data also includes demographic information about the patients. Despite the lack of patient specifics, our primary goal is to identify diabetes phenotypes using only routine laboratory results of diabetes patients—i.e. without the use of clinical attributes; we aim to group diabetes patients. After finding distinct phenotypes, we aim to predict the phenotype of a new incoming patient based on their pathology test results in order to identify the disease trend for this patient and also the pathology tests that need to be prescribed based on a cohort level analysis.

1.3 Research Approach

To generate patient phenotypes based on a time series of pathology test results, we have investigated machine learning methods for clustering and classification. Our research approach is (a) to investigate and apply ensemble clustering methods, i.e. a combination of multiple clustering methods to generate consensus-based clusters to derive patient phenotypes based on time-series data, and (b) to investigate and apply classification methods to the clustered data to develop diabetes progression prediction models that can be used on new patients to stratify their diabetes progression.

Our approach involves (a) processing time-series data, comprising multiple data types and collection frequency, for clustering algorithms; (b) imputation of missing values in a time-series; (c) exploring ensemble clustering methods and investigating the efficacy of different consensus functions, and (d) exploring various classification models and identifying the algorithm that can predict the clusters of patients with greater accuracy.

After preprocessing the data, the data is split into two cohorts where cohort1 or the base cohort, possesses patients who have taken all the considered pathological tests and cohort2, or the incomplete set, includes patients who have taken at least

one of the considered tests. The cohorts are separated so that none of the patients overlap in both cohorts. Meaning if ten tests are considered, the base cohort will have records of the patients who have undergone all the ten tests more than once, whereas an incomplete set will contain patients who have taken up to nine of the ten considered tests leaving nulls at the results of the tests that are not taken. Various imputation techniques are explored to replace the nulls in the incomplete set, and a feasible method is selected. After cohort separation, the base cohort is used for finding phenotypes, whereas patients in the incomplete set are assigned to groups using classification models. A broad range of clustering models was investigated to identify a superior one that can find clusters from the first cohort's data. Together with the labels created, the first cohort data serves as the training source for the classification model, which subsequently assigns patients from the second cohort and new patients to the clusters.

As an outcome, we can identify diabetes patient phenotypes from test results and predict disease progression trends in new patients based on their available pathology test results.

1.4 Contribution

The main contribution of this thesis include:

- We have introduced a pipeline for discovering patient characteristics and evaluating disease progression from unstructured diabetes data that combines unsupervised and supervised learning approaches, namely consensus clustering and classification models.

- Investigation and identification of a feasible consensus function to cluster high-dimensional time-series diabetes data to uncover the crucial tests that distinguish patients into different peer groups.

- Assessing various classification algorithms and identifying the best one that effectively learns the properties of the clusters formed by consensus function and can assign the new patients to appropriate clusters.

1.5 Thesis Organization

The remainder of the thesis is structured as follows: Chapter 2 starts by explaining phenotypes and the application of machine learning in it, followed by the explanation of previous works done on diabetes diagnosis and progression. It also presents the literature review of essential topics employed in the study, such as cluster analysis, cluster ensembles, cluster validation metrics, imputation, and classification. Chapter 3 presents the dataset employed along with the methods followed in the study. Chapter 4 gives a detailed description of the methods used to achieve our aim. The parameter settings and outcomes obtained at each stage are described in Chapter 5. Finally, Chapter 6 wraps up the work by highlighting limitations and future work.

Chapter 2

Background and Related Work

This chapter reviews phenotyping in general and the use of machine learning algorithms for phenotyping. Next, we will present an overview of the application of machine learning models for diabetes management. We will discuss cluster analysis and consensus clustering methods and review data science methods applied in our research.

2.1 Phenotyping in Machine Learning

A phenotype is the physical or biological representation of a particular attribute in an organism, such as a disease, height, or blood type, that is impacted by genetic information and environmental circumstances [90]. In biomedical research, a phenotype is a clinical condition, characteristic or a set of features that are found more often in individuals with a disease or condition than in the general population. The primary source for analyzing these phenotypes in healthcare is electronic health records (EHRs). Several national efforts, including eMERGE, SHARP and PGRN, have demonstrated the applicability of EHR towards high-throughput phenotype extraction [70]. The eMERGE (Electronic Medical Records and Genomics) network is an academic medical centre collaboration that develops generalizable EHR phenotypic definitions for genome-wide association research using shared clinical data sets [26]. eMERGE is responsible for an extensive catalogue of phenotypes, including hypothyroidism, type 2 diabetes, atrial fibrillation, and multiple sclerosis [26].

The earliest approach to phenotyping is via manual chart review, typically performed by thorough searching of clinical documents, laboratory, and medication information by individuals with medical domain knowledge. With the rapid increase in the adaption of EHRs, a large volume of data becomes available to the experts making the analysis process complex and time-consuming. Efforts were made to reduce the manual intervention made by the experts in identifying the phenotypes from the

EHRs. Identifying the phenotypes solely from the ancillary data sources without the review or interpretation of clinicians is called a computable phenotype [90].

Machine learning (ML) is the most sought-after approach to identifying computable phenotypes from structured and unstructured EHRs [21]. ML is a computational discipline that creates algorithms that allow computers to use large amounts of data to build predictive models or recognize complex patterns or characteristics within data. Computable phenotype definitions using electronic health record data (EHR) have been published on common conditions like chronic obstructive pulmonary diseases (COPD), bipolar disorders, heart diseases, and more. There are two types of machine learning approaches: supervised and unsupervised. The input data in supervised learning are labelled, and the system must learn a pattern to predict the intended output from these data. Support Vector Machine (SVM), Decision Tree, and logistic regression approaches are the most commonly utilized supervised techniques in the literature [21].

In 2007, one of the first uses of machine learning for phenotyping was published by Huang et al. [59]. Using a cohort of diabetic patients and controls, the authors employed FSSMC (feature selection via supervised model construction). They manually distilled an initial set of 410 structured variables down to 47 features. They ran FSSMC to rank the features in order of importance. They then evaluated the performance of three machine learning classifiers, namely naïve Bayes, C4.5 and IB1, in identifying diabetic patients. They determined the features that had the highest performance and selected those features as the most relevant ones.

Another example of supervised learning in phenotype classification is the work of Wright et al. [114], who have used the SVM-based approach for classifying EHR progress notes about diabetes. They retrieved 2000 EHR progress notes from patients with diabetes at the Brigham and Women's Hospital (1000 for training and 1000 for testing) and another 1000 notes from the University of Texas Physicians (for validation). They manually annotated all notes, trained an SVM using a bag of words approach and then used the SVM on the testing and validation sets. The evaluation of their model was carried out in the Area under the curve (AUC) and F statistics. Their model was highly generalizable to different datasets.

In another phenotyping work carried out by Zhou [117] and his colleagues, decision

trees identified patients with rheumatoid arthritis. They identified eight distinct predictors related to diagnostic codes for rheumatoid arthritis, medication codes for disease-modifying anti-rheumatic drugs, and the absence of alternative diagnoses such as psoriatic arthritis. The proposed data-driven method was performed, and the expert clinical knowledge-based methods.

Supervised strategy, however, is dependent on annotated training and testing sets for model building and validation, which are costly, complicated, and time-consuming to create by domain experts. Labelled data also has the disadvantage of not being transferable across institutions and being unable to be shared according to HIPAA (Health Insurance Portability and Accountability Act) restrictions [26]. Without the requirement for expert input or labelled samples, unsupervised learning approaches can uncover patterns that, when combined, produce a compact and helpful portrayal of the original data. However, due to the absence of stated ground truth in these groups, result verification for phenotypic groups with this technique is difficult. With the availability of massive volumes of unlabeled data, this approach has lately gained favour.

Hripcsak et al. [49] used the phrase "high-throughput phenotyping" in 2013, describing a fundamental change toward customized phenotypes carefully created from individual data sets and toward the automated synthesis of thousands of phenotypes in a scalable manner with minimum human supervision. An example of phenotyping on unlabeled data is Siroux et al. [99], where they used a model-based clustering method to identify asthma phenotypes among the adult population and reported four distinct phenotypes. Using the latent class analysis approach, they used 19 factors to determine four asthma phenotypes. These included personal characteristics, asthma symptoms, exacerbations, therapy, age of asthma onset, allergic features, lung function, and airway hyperresponsiveness. They stated that their phenotypes discriminated populations in terms of quality of life and blood eosinophil and neutrophil counts, leading to better identification of asthma risk factors.

Jess [97] and his colleagues ensembled three clustering algorithms and identified four distinct subtypes of Pervasive Developmental Disorder(PDD). They applied kmeans, hierarchical clustering and EM clustering on 358 PDD patients and identified four clusters that roughly correspond to three main subtypes of PDD: Autism,

PDD-NOS and Asperger's syndrome.

Even though the unsupervised method tried to form groups among the patient population, the experts need minimal intervention to validate the correctness of the phenotypes. To reduce the need for intervention, Henderson et al. [58] created PheKnow- Cloud, a platform that lets researchers analyze phenotypes generated using previously accessible methodologies utilizing medical literature. It matches the candidate phenotype with the synonyms of related biomedical vocabularies and performs a co-occurrence search on PubMed Central's open access subset, using them to build sets of evidence for user-supplied candidate phenotypes. Thus, reducing the number of candidate phenotypes must be manually reviewed by medical experts.

Thus, ML, be it supervised or unsupervised is the most predominant approach applied to EHRs for identifying valuable patterns from the data. These approaches can minimize the work required from clinical domain specialists since it eliminates the necessity for their intervention throughout the process [21]. Because of their ability to identify patients using observational research and clinical trials, computable phenotypes are desirable to the rare disease community. However, the quality of the phenotypes discovered is limited by the available data quality.

2.2 Diabetes Progression

Diabetes is a chronic disease in which the body cannot absorb and utilize sugar (glucose) from the diet, resulting in elevated blood glucose levels (hyperglycemia). Complications caused by diabetes often lead to physical, psychological, and functional impairments. As a serious health concern, diabetes has been declared a global epidemic by the WHO due to its rapidly increasing incidence [17]. Owing to its consequences, early-stage detection of diabetes is needed, and in this context, EHRs play a crucial role in keeping track of patients' health conditions. Recently, many research studies have been carried out on these clinical data predominantly by applying machine learning and data mining techniques. Machine learning is a sub-field of artificial intelligence that uses algorithms to generate predictions based on previously collected data. Clustering and classification are the two popular machine learning techniques commonly used for analyzing medical records. These techniques are comparatively more efficient in handling complex data than traditional statistical approaches. It is

applied to diabetes EHRs to estimate disease probability prediction, screening, diagnosis, treatment guidance, and complication management [17]. While most problems require professional guidance, some diabetes management challenges can be handled with data processing corresponding to the research's aim. ML can be applied extensively to genomic data EHR data of diabetic patients for diagnosing the disease and predicting the onset of complications like nephropathy and retinopathy. This can be done by grouping the patients based on their medical records using various phenotyping algorithms.

Numerous studies have focused on identifying subgroups in the diabetes population. This phenotyping not only assists in the identification of subgroups within the population but also aids in the identification of patient groups that require targeted interventions [63, 112]. The dataset in all research generally comprises n patient records, each represented by a p-dimensional feature vector. These feature values reflect patient characteristics that may differ depending on the data source. In this part, we reviewed prior research that used cluster analysis and classification methods on diverse diabetes datasets.

The research carried out by Deepti and Dilip [100] identified the patients with the probable onset of diabetes. They conducted experiments on Pima Indians Diabetes Database by comparing three classification algorithms, including SVM, NB and DT, to forecast the likelihood of diabetes in patients with maximum accuracy. The attributes number of time pregnant, plasma glucose concentration, diastolic blood pressure, skinfold thickness, 2-hour serum insulin, BMI, diabetes pedigree function, age and class' 0' or' 1' were used. Their experiments concluded that NB predicted the chance of diabetes with maximum accuracy. Another research by Perveen et al. [86] AdaBoost and bagging ensemble with decision tree as a base learner to identify diabetes mellitus using diabetes risk factors. They utilized demographic information, vital signs and lab values for their analysis and applied these classification models to three ordinal adult groups in the Canadian Primary Care Sentinel Surveillance network. Experimental result shows that the overall performance of the AdaBoost ensemble method is better than bagging and standalone decision tree.

The investigation of different diabetes datasets conducted by Ahlqvist [19] and his colleagues revealed five repeatable clusters. Each cluster differed in disease course

and risk of diabetic complications. They utilized k-means and hierarchical clustering on 8980 newly diagnosed diabetes patients from three independent diabetes cohorts: the Scania Diabetes Registry, ANDIU, and DIREVA, using six variables as features: GAD-antibodies, age at diagnosis, BMI, HbA1c, HOMA2-B, and HOMA2-IR. Anjana et al. [22] divided the Asian Indian population's type 2 diabetes patients into four groups. They chose 19,084 people with type 2 diabetes (aged 10–97 years) who had diabetes for less than five years and used k-means clustering with the following variables: age at diagnosis, BMI, waist circumference, glycated hemoglobin, serum triglycerides, serum high-density lipoprotein cholesterol, and C peptide (fasting and stimulated). Both studies stated that their findings could identify individuals at a higher risk of diabetic kidney disease and retinopathy. They concluded that this stratification might eventually aid in early treatment for patients who would benefit the most, thereby representing a first step toward precision medicine in diabetes.

Apart from phenotyping the patients based on various characteristics related to diabetes, researchers have used the information related to the complications caused by diabetes like heart disease, eye damage, kidney damage, and nerve damage to classify the patients. Ernande and her colleagues [39] established distinct categories of type II diabetic (T2DM) people based on echocardiographic characteristics because they suspected that T2DM would change heart structure and functions. They utilized agglomerative hierarchical clustering on echocardiograph characteristics from type II diabetes individuals who did not have overt heart disease. Their research revealed three distinct groups of individuals with varying cardiac problems. Like Ernande, Chul Won et al. [113] used hierarchical clustering to identify three patient categories based on diabetic peripheral neuropathy. Sawacha et al. [93] concentrated on diabetic foot problems, a severe consequence of diabetes caused mainly by peripheral neuropathy. They categorized the data among patients using walking patterns (biomechanical data). K-means cluster analysis categorized 20 nondiabetic and 46 diabetic patients with and without peripheral neuropathy.

According to Abhari et al. [17], classification is the most sought-after method for diabetes analysis, especially SVM and NB. They also state that the most common application of machine learning for diabetes was screening and diagnosis, followed by complication diagnosis. Also, there is only a little application of clustering in diabetes

research, predominantly using single clustering methods. Our research evaluates the performance of the cluster ensemble techniques with conventional single clustering models for finding the phenotypes of diabetes patients. Also, from the past research, it is evident that most of the research was carried out using the original features available in the dataset. Our research stands apart by utilizing the temporal characteristic of the pathological report as a feature instead of using the report itself. Thus, we have built a novel method for analyzing diabetes data.

2.3 Cluster Analysis

Cluster analysis is a powerful statistical technique of data processing that unveils the hidden pattern by grouping data objects with similar characteristics into homogeneous clusters while improving heterogeneity across the clusters [72]. These data objects are usually represented by features or attributes, which is key to identifying similarities and dissimilarities. Cluster analysis has been widely applied in a wide range of disciplines, including data mining, pattern recognition, machine learning, image processing, bioinformatics, web cluster engines, and many more [29]. Exploratory Data Analysis(EDA), statistical analysis, anomaly detection, customer segmentation, computer vision, recommendation systems, and spatial clustering are some instances where clustering is helpful [74].

Over the years, several clustering algorithms have been developed using different measurements of distance or similarity and objective functions that produce a different results on the same data [43, 107]. Some of the typical cluster models include connectivity models, centroid models, distribution models, density models, fuzzy models and model-based clustering.

Connectivity or hierarchical clustering is based on the fundamental principle that objects are more connected to items close to them than objects further away. These algorithms use distance to connect "objects" to generate "clusters." There are two approaches to this process. In the first technique, the algorithm divides data elements into distinct clusters and arranges them based on distance criteria. Another way is for the algorithm to select all data entities into a particular cluster and then aggregate them based on the distance criterion because the distance function is a subjective choice based on user criteria. Hierarchical clustering is simple and does not require

apriori knowledge of the number of groups. Easy to implement across various forms of data and known to provide robust results for data generated via multiple sources. Hence it has a wide application area. The main output, a dendrogram, is more appealing and easier to understand. The biggest downside of this approach is that it requires a considerable number of arbitrary judgments and struggles to deliver the optimal outcome when the input data is vast and multidimensional as it has high time complexity. This algorithm also fails to handle outliers and noise. The algorithms that follow hierarchical clustering are agglomerative and divisive clustering algorithms.

Centroid-based clustering is one of the most successful and basic methods of constructing clusters and allocating data points. Each cluster is defined and represented by a central vector, which may or may not be a part of the dataset, and data points near these vectors are assigned to the relevant clusters. Using various distance metrics, these clustering algorithms iteratively estimate the distance between the clusters and the characteristic centroids. The critical drawback here is that we must specify the number of clusters or the central vectors before allocating data points to the vectors. Also, this model cannot be applied to categorical data and fails to handle outliers. Despite its shortcomings, Centroid-based clustering has shown to be superior to Hierarchical clustering when working with massive datasets and is much faster than other algorithms. Furthermore, due to their ease of implementation and interpretation, these algorithms have a broad range of applications, including market segmentation, customer segmentation, text topic retrieval, picture segmentation, and much more. Some of the centroid models are k-means, k-medoids and affinity propagation.

Compared to the prior models dependent on distance metrics, the *Density-based clustering* model considers the density. It considers clusters as the densest region in a data space, separated by areas of lower object density. While most clustering models assume that the data is devoid of noise and the shape of the cluster formed is geometrical, the density-based model forms clusters in arbitrary shapes by successfully handling outliers or noise. The drawback of this model is that it does not apply to datasets with varying densities or if the data is too sparse. Some example algorithms that are density-based include OPTICS and DBSCAN.

Distribution Models consider probability as a metric for clustering the data. It

creates and groups data points based on their likelihood of belonging to the same probability distribution in the data. It is closely related to statistics as data points are studied under this method to trace their generation or origination point. These models have a prominent advantage over density and centroid models in terms of flexibility, correctness and shape of the cluster formed. Distribution models are not preferred if the distribution of the input data points is unknown or not clear. A prominent example of a distribution model is the gaussian mixture model.

Fuzzy clustering makes a distance-based cluster assignment similar to centroid-based clustering but with the difference that it assigns data points to more than one cluster. It is based on the idea that some data inputs may have features that overlap and so assign a data point to more than one cluster based on the parameters of the different clusters. It has a greater convergence rate and works well on highly correlated or overlapped data where centroid models cannot offer ideal results. However, the model gets slower if the data is large and cannot handle outliers. Like centroid-based models, this model also requires the number of centroids to be specified apriori.

Among various clustering models mentioned above, we have equipped k-means, k-medoids, affinity propagation and the Gaussian mixture model in our study. The reason for choosing these particular algorithms is explained in section 4.2. A detailed explanation of each algorithm selected is provided below:

K-means clustering

K-means clustering is a fundamental and widely used unsupervised learning technique, particularly in data mining and statistics. As a partitioning method, it aims to arrange data points into groups depending on the number of clusters, denoted by the variable k. The value of k is typically unknown in advance and must be chosen by the user. K-means produces its final grouping using an iterative refinement approach based on the number of clusters provided by the user. Initially, k-means selects k mean values of k clusters, known as centroids, at random and finds the nearest data points of the selected centroids to generate k clusters. The approach repeatedly recalculates the revised centroids for each cluster until it reaches a single optimum value. Because numerical data is utilized to derive the mean value, K-means clustering is best suited for numerical data.

The k-means method splits a dataset into k groups using the following steps [115]:

1. Based on the predetermined value of k, K points are randomly initialized as cluster centroids.

2. Each data point in the data set is allocated to the nearest centroid by distance to generate the k clusters. In order to calculate the distance between each data point and the initialized centroids, the Euclidean distance is employed. Although there are many different metrics for determining the closest distance, we chose the Euclidean distance since earlier research on clustering analysis produced excellent results using the Euclidean distance.

3. The centroid of each cluster is computed based on the points assigned to that cluster.

4. Steps (2) and (3) are repeated until no data points change their clusters.

Suppose $N = \{x_1, ..., x_n\}$ is the data set to be clustered. The Euclidean distance between two data points X_1 and X_2, each represented by a p-dimensional vector, $X_1 = (X_{1_1}, X_{1_2}, ...X_{1_p})$ and $X_2 = (X_{2_1}, X_{2_2}, ...X_{2_p})$ is defined as follows:

$$dist\_Euc(X_1, X_2) = \sqrt{\sum_{i=1}^{p}(X_{1_i} - X_{2_i})^2} \tag{2.1}$$

The algorithm iteratively shifts data points across clusters by reducing the sum of squared distances from each point to its centroid, represented by J. We can calculate the sum of squared distances of the i^{th} among C clusters as follows:

$$D_i = \sum_{X \in C_i} dist\_Euc(X, Y_i)^2 \tag{2.2}$$

where, $dist\_Euc(X, Y_i)$ is the Euclidean distance from a data point X in C_i to C_i's centroid Y_i. Thus the sum of squared distances for all the k clusters is defined as :

$$D = \sum_{i=1}^{k} D_i \tag{2.3}$$

K-means is a widely used unsupervised learning algorithm as it is fast, robust, and easier to understand. Many researchers utilized k-means in their study and achieved significant results[40, 108].

Partitioning Around Medoids

K-medoids also referred to as Partitioning Around Medoids (PAM), is a centroid-based partitioning method similar to k-means that searches the data for k-centroids (or k-medoids) assigns each data object to the nearest medoid to construct clusters. This approach needs the user to define the number of clusters before executing the algorithm. In contrast to the k-means method, k-medoids select real data points as centroids, allowing for higher interpretability of cluster centers than k-means, which does not need the center of a cluster to be one of the input data points. It includes greedy search, which is faster than exhaustive search [89]. A dissimilarity matrix is used to reduce the total dissimilarity between the centroids of each cluster and its members. The dissimilarity of the medoid (C_i) and an object (P_i) can be computed using $E = |P_i - C_i|$. The cost of k-medoids is given as

$$c = \sum_{C_i} \sum_{P_i \in C_i} |P_i - C_i| \tag{2.4}$$

The PAM algorithm's overall procedure is divided into the Build Phase (steps 1-3) and the Swap Phase (steps 4 & 5). This algorithm comprises the following steps:

1. Choose k data objects to become the medoids or utilize the objects supplied as the medoids.

2. Compute the dissimilarity matrix if it was not provided;

3. Assign every object to its closest medoid;

4. For each cluster, determine whether any of the cluster's objects reduces the average dissimilarity coefficient; if so, choose the entity that reduces this coefficient the most as the medoid for this cluster.

5. If at least one medoid has changed, go to (3); otherwise, stop the algorithm.

Some of the research that benefited from this algorithm are [27, 34, 64, 18]. This algorithm's key advantages are that it is quick, simple, and less sensitive to outliers [10].

Affinity Propagation

Affinity Propagation(AP) is based on the concept of "message passing" that does not require the number of clusters to be provided prior to starting the process [47]. It represents each data point as a node in a network, with all data points sending messages to all other data points. The willingness of the points being exemplars is the focus of these communications. Exemplars are data points that best describe the other data points and are the most important in their cluster. There can be just one exemplar in a cluster. Two types of real-valued signals are repeatedly exchanged between data points during the clustering process until a high-quality collection of exemplars and appropriate clusters forms [47]. These messages are saved in two separate matrices:

1. The "responsibility" matrix R ,r(i, k), indicate how well-suited x_k is to serve as the example for x_i in comparison to other possible exemplars for x_i.

2. The "availability" matrix A , a(i, k), shows how "appropriate" it would be for x_i to choose x_k as its exemplar, taking otherpoints' preferences for x_k as an exemplar into consideration.

Both matrices are initialized to all zeroes and are updated by the algorithm iteratively.

$$\textbf{Responsibility update: } r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\} \tag{2.5}$$

where, s(i,k) is the similarity between the samples i and k.

$$\textbf{Availability update: } a(i,k) \leftarrow min(0, r(k,k) + \sum_{i' \notin i,k \ \&\ i \neq k} \max(0, r(i',k)) \tag{2.6}$$

Iterations are performed until the cluster boundaries stay intact after a certain number of iterations or a preset number of iterations is attained. The exemplars are selected from the final matrices as those with a positive 'responsibility + availability' for themselves [1]. Some of the previous works that successfully incorporated this model are [116, 38, 37].

Gaussian Mixture Model

Gaussian Mixture Model(GMM) with Expectation-Maximization(EM) is a probabilistic model-based clustering algorithm that assumes the data to be a mixture of various Gaussian distributions and tries to group the data points that belong to the same distribution. Here, each cluster is modelled according to a different Gaussian distribution instead of selecting clusters by "nearest" centroids [50]. Then, for each cluster, we estimate gaussian distribution characteristics such as mean and variance, as well as cluster weight. We can determine the probability of each data point belonging to each cluster after learning the parameters for each data point. This approach is flexible because, unlike other methods, it soft-assigns objects to clusters rather than hard-assigning them, implying that each data point might have come from any of the distributions with a matching probability. A latent variable is introduced to estimate such a model, identifying the gaussian that created the given data point. The EM method is used to estimate this latent variable, which consists of two steps [9]: The Expectation step, the E step, computes the expectation of the component assigned to the cluster for the given mean, variance, and weight and a Maximization step, M step, that maximizes the expectation calculated in the E step. The overall process of this model is as follows:

1. Set the parameters (mean and variance) for each k Gaussian distribution.

2. Calculate the probability density for each feature vector under each of the k clusters, using the Gaussian distribution's probability density function for each cluster.

3. Using the probability densities derived in Step (2) re-estimate the parameters for each of the k Gaussian distributions.

4. Iterate through Steps 2 and 3 until convergence is reached.

The data's probability(or likelihood), given the parameters, is maximized by updating the mean and variance of the Gaussian distribution for each cluster in step (3). The better the clustering model fits the data, the higher the likelihood. The starting values determine the EM algorithm's maximum convergent value supplied to the parameters in step (1). The optimum found by EM dramatically depends on the

initialized parameters and picking the solution that yields a higher likelihood. One of the main advantages of this model is that it is flexible and performs soft clustering. Some of the successful research that used GMM can be found in [88, 111, 109].

All these algorithms have their strength and weakness due to the complexity of the information and differ mainly in input parameters, like the number of clusters, optimization/construction criterion, termination condition, proximity measure, and much more. These clustering algorithms produce different clustering results for the same dataset due to differences in their clustering technique. [20]. According to the "no-free-lunch theorem," every algorithm performs equally well when averaged across all possible situations meaning there is no single clustering algorithm that performs best for every dataset [91, 68]. Also, there are no clear guidelines to follow for choosing individual clustering algorithms for a given problem [20]. According to the impossibility theorem [66], no single clustering method meets the fundamental assumptions of data clustering, notably scale-invariance, consistency, and richness, thus making it impossible to develop a generalized framework of clustering methods. In order to overcome these hurdles, researchers have proposed an exceptional concept of clustering called ensemble clustering, which will be elaborated in section 2.4.

2.4 Cluster Ensemble

Ensemble learning is a generic machine learning method that tries to improve performance by integrating outcomes from several models. Ensemble members, the models that contribute to the ensemble, can be of the same or various kinds, and they can be trained on the same or separate training data. Ensemble learning aims to combine weak learners to make one strong learner. The results of the ensemble members can be aggregated using statistics like the mode or mean or more advanced approaches that decide how far to consider each member and under what situations. The primary reason for utilizing ensemble models is to minimize generalization error while boosting model performance and robustness. The ensemble method was firstly introduced and well-studied in supervised learning fields [20]. Inspired by the success of the ensemble approach in the supervised learning field, numerous research has been performed in recent years to create clustering ensemble techniques [102, 103]. Ensemble clustering is a technique of combining several clustering results into the final clusters, using a

consensus function, without gaining access to the algorithms or features [54]. It is also known as Consensus clustering or Clustering Aggregation. The cluster ensemble problem is more complex than the classifier ensemble problem. One reason is that there is no previous information with which the algorithm can identify the actual cluster structure, and there is no "ground truth" to confirm the clustering outcome. Furthermore, no cross-validation approach can be used to tweak the clustering algorithm's parameters; therefore, there are no suggestions to help the user choose the best clustering algorithm for a specific dataset. Another issue is that the number of clusters created by different clustering methods may differ.

J Ghosh and A Acharya [52] highlighted that there are numerous justifications for implementing clustering ensembles, which are substantially greater than those for employing a classification ensemble, where the primary goal is to improve prediction accuracy. Some of these motivations include:

1. Improving cluster quality compared to clusters produced by different clustering methods.

2. Knowledge reuse: In some applications, various partitions may exist, which may be merged to provide a final clustering result. This produces a more consolidated clustering result; Several examples are provided in [101]

3. Generating robust clustering solution, i.e., being able to provide good results across an extensive range of datasets.

Among these objectives, improving the quality of the clustering result is the most widely accepted one. The clustering quality is usually measured with a numerical measurement to assess different aspects of cluster validation [67]. Section 2.5 reviews some of these in detail. The ensemble clustering has the capacity to produce a better outcome in terms of consistency, Robustness, Novelty, and stability [20, 107].

Cluster ensemble is a two-step process: the generation step, where the clusterings need to be combined, is generated, and the consensus step, where certain consensus functions are applied to combine the clustering results. Figure 2.1 shows the general architecture of ensemble clustering where the input is the dataset, and the output is the ensembled cluster assignments. Vega-Pons and Ruiz-Shulcloper [107] stated that since there are no restrictions on how the generating stage should be carried out,

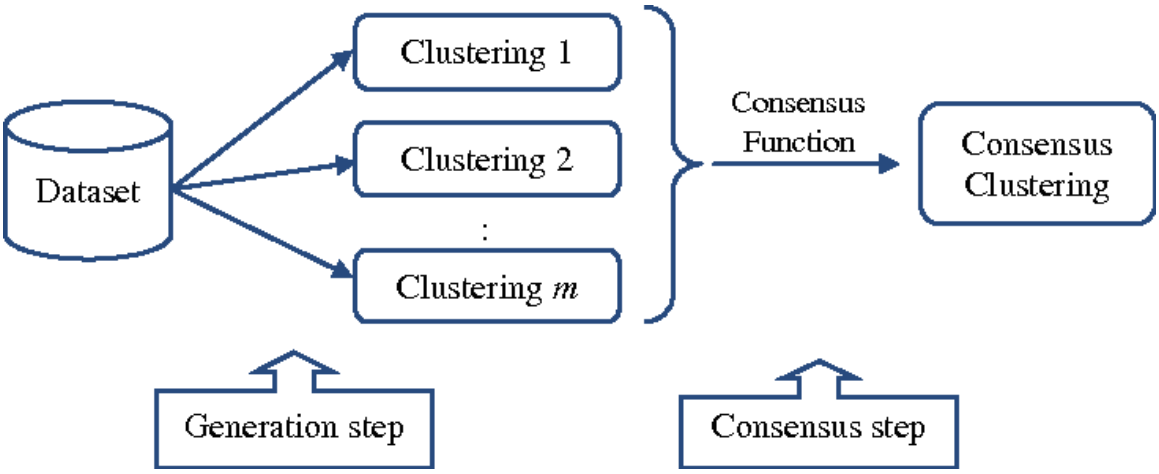

Figure 2.1: General process of cluster ensemble [107]

different strategies can be used. Several generation approaches have been utilized in the literature to create members for an ensemble; further explanation of these techniques can be found in section 2.4.1. A clustering ensemble's success is dependent on selecting a consensus function that can increase the quality of the final clustering solution. Several consensus functions have been proposed in the literature, some of which will be described in section 2.4.2

2.4.1 Ensemble Generation Methods

A generative mechanism comprises several techniques that might result in various individual clusters of a given dataset. Regardless of obtaining base clusterings with the required diversity and quality, main diversity is discovered in many ways. There are no restrictions on how the base clusters must be collected, such as if various single clustering methods or one algorithm with variable parameter settings must be used. Researchers select the generation technique depending on the type of data or the application being used. Depending on the purpose of the research, the generation mechanism can be carried out primarily using any of the following strategies [101, 71]:

1. For the same data set, employ the same algorithm with the different parameters or initial conditions.

2. For the same data set, employ the different algorithms.

3. For the subsets, carry out the clustering respectively.

4. For the same data set, carry out the clustering in different feature spaces based on different kernels.

On the other hand, N. Iam-On and S. Garrett [61] classified the cluster generation step into five categories, namely homogeneous ensemble, random-k, data subspace/ subsample, heterogeneous ensemble, and mixed heuristics. While the first four types are similar to the strategies stated in [101, 71], the mixed heuristics is the combination of any of the strategies mentioned above.

The majority of the research on ensemble clustering equip one of the ensemble generation methods mentioned above. One such example is the work of Strehl and Ghosh [101] where they used random feature subspaces for high-dimensional data and created members for each of the data subspaces. Members were also formed by picking distinct subsets of items for each member. This approach was denoted as object distribution, and it was used for large amounts of data. Fern and Brodley [42] created members by randomly projecting items into distinct subspaces and using the Expectation-Maximization method (EM) on these subspaces. Topchy et al. [79] proposed a robust and stable resampling(or bootstrapping) method, especially sampling with replacement, by investigating the effectiveness of a bootstrapping technique in conjunction with several combination algorithms. They have chosen k-means for bootstrapping with random restart due to their lower computational complexity. On the other hand, Monti et al. [81] used the bootstrap technique with different clustering algorithms, including k-means, model-based Bayesian clustering, and self-organizing map. The main reason was to determine the optimal number of clusters obtained while running the clustering algorithms multiple times to examine the boundaries, cluster number, and membership. K-means seems to be the most popular algorithm for generating members with random-k initializations due to their low computational complexity [20]. Another popular strategy is using different clustering algorithms for each member of ensemble data where all the algorithms may complement one another. H Mannila et al. [53] used the Single, Average, Ward, and Complete linkage methods and k-means to generate ensemble members.

In conclusion, as shown here, no one clustering technique is commonly employed, and there are no generally agreed-upon criteria for selecting the best one [20]. Our

study utilizes subsampling techniques on heterogeneous as well as homogeneous clustering algorithms for generating the ensemble data.

2.4.2 Consensus Function

A consensus function is the most crucial component in ensemble clustering as it combines the output of n individual clusters {C1, C2,...Cn} to a single final clustering result C. According to Vega-Pons and Jose Ruiz-Shulcloper [107], object co-occurrence and median partition are the two main approaches in consensus functions. Median partition-based approach groups the data points of different partitions by their similarity index and forms the new partitions based on the medians of the data points of previous partitions. It treats the ensemble as an optimization problem. Some of the approaches based on this method include the genetic-based method, Kernel-based method, and Non-Negative Matrix Factorization-based method [20]. The co-occurrence-based technique counts the presence of an item in one cluster or the occurrence of two objects in the same cluster and creates the final clustering result through a polling system among the objects. Some of the techniques that use this approach are Relabelling and Voting method, Co-association matrix, and the Hypergraph partitioning based method [20].

Hypergraph partitioning method proposed by Strehl and Ghosh [101] where the clusters are picturized as the edges of the graph and vertices correspond to the items to be grouped, and each hyperedge specifies a set of objects belonging to the same clusters. The consensus clustering problem is thus simplified to determining the minimum cut of a hypergraph. Based on this concept, they proposed three consensus functions: the cluster-based similarity partitioning algorithm (CSPA), the hypergraph partitioning algorithm (HGPA), and the meta-clustering algorithm (MCLA). While the CSPA generates a graph from the similarity matrix and clusters it using the METIS method, the HGPA and MLCA generate hypergraphs with each member represented by a hyperedge [51]. The main difference between HGPA and MLCA is that HGPA directly partitions the hypergraph at the minimal possible hyperedge number. In contrast, MLCA generates the similarity score using the Jaccard index, represents the scores as edges, and then partitions the graph using these newly computed similarity scores using HMETIS method [20].

The voting approach, also called the direct or relabeling approach, attempts to solve the correspondence problem between the labels of the generated cluster results and then applies the simple voting procedure to determine the final cluster assignments. The technique resolves the correspondence problem by relabeling the labels based on a defined reference point, which might be drawn from one of the ensembles created or from a new clustering of the dataset [51]. The study by Dudoit and Fridly [36] deduced a consensus function that is similar to the bagging process(plurality voting) used in classification ensembles. They assumed that all members have the same number of clusters. By applying the plurality voting process, they obtained the final clustering result, with the same number of clusters as the members.

The co-association matrix is a pairwise similarity-based method. Here, the label correspondence problem is avoided by mapping the ensemble members to the new representation, the similarity matrix computed between pairs of objects in terms of their togetherness across all ensemble members [46]. Fred and Jain obtained the final partition by applying single and average linkage hierarchical algorithms to the co-association matrix. While the co-association matrix seems to help capture the information available in the ensemble members, it only captures the pairwise relationship between the objects. Wang et al. [110] proposed Probability Accumulation (PA) which extends the Co-association method by considering the cluster size and the dimensions of the objects within the data when calculating the Co-association matrix. Iam et al. [61] proposed a link-based similarity measure(LCE) where they used various similarity matrices like Connected–Triple-based similarity (CTS) and SimRank based similarity (SRS).

Apart from the consensus functions mentioned above, there are numerous other functions out in the literature. Among all, we have equipped five specific consensus functions, namely CSPA, LCA, LCE, kmodes, and majority voting, which are discussed below:

Cluster-based Similarity Partitioning Algorithm(CSPA)

The cluster-based similarity partitioning algorithm (CSPA) developed by Strehl and Ghosh [101] signifies clustering as a relationship between the objects in the same cluster by measuring the pairwise similarity between the objects, which is then used

to recluster the objects. It creates a hypergraph from the cluster labels, with the number of frequencies of two objects gathered in the same clusters treated as the weight of each edge. A n x n binary similarity matrix is built from the network for each grouping, with two items having a similarity of 1 if they are in the same cluster and 0 otherwise. A single sparse similarity matrix S from r such similarity matrices of various clusterings is obtained by:

$$S = \frac{1}{r} H H^{+} \tag{2.7}$$

The final similarity matrix is used to recluster the objects, which is done using METIS(Multiway Spectral Clustering Method) algorithm due to its robust and scalable property.

K-modes

The K-Modes algorithm, presented by Haung [60] extends the K-Means approach to both numerical and categorical domains. The k-modes approach employs a basic matching dissimilarity measure to minimize the clustering cost function, replaces cluster means with modes, and employs a frequency-based mechanism to update modes throughout the clustering process. These changes ensure that the clustering process converges on a local minimum outcome. The matching dissimilarity matrix is defined as: Let X and Y be two data objects with m attributes each. The total mismatches of the respective characteristics of two objects define the dissimilarity measure d(X, Y) between X and Y. Lower mismatch indicates that the objects are similar. Mathematically, it can be defined as:

$$d(X, Y) = \sum_{j=1}^{m} \delta(x_i, y_i) \tag{2.8}$$

where,

$$\delta(x_i, y_i) = \begin{cases} 0, & (x_j = y_j) \\ 1, & (x_j \neq y_j) \end{cases} \tag{2.9}$$

The overall procedure followed in k-modes is as follows:

1. Choose K unique objects as initial modes at random, one for each cluster.

2. Allocate an object to the cluster whose mode is the closest to it in terms of dissimilarity.

3. Each cluster's mode is updated based on the frequency of the data items in the same cluster.

4. Retest the dissimilarity of objects against the current modes once all objects have been assigned to clusters. If it is discovered that an item's closest mode belongs to a different cluster than its current one, reallocate the object to that cluster and update the modes of both clusters.

5. Repeat (3) until no object causes the clusters to change.

Latent Class Analysis(LCA)

LCA is a promising statistical strategy for examining multidimensional data. It allows researchers to capture the association that exists among observed categorical indicators through a set of unobserved latent classes [78]. In the literature, LCA is referred to in different ways: Latent Structure Analysis, Mixture Likelihood Clustering, Model-based Clustering, Mixture-model Clustering, Bayesian classification, and Latent Class Cluster Analysis [95]. This model assumes that by considering measurement error, categories of latent variables could explain the correlation between indicator variables. Using several iterations, LCA compares the frequency of response patterns for each identified latent class. As a result, LCA calculates some statistics that help to select the best model. These statistics are G2, Akaike information criterion (AIC), Bayesian information criterion (BIC) [16]. For these statistics, the smaller value shows a better model fit. Finally, a model with the smallest values of AIC or BIC might be selected. Like k-means, LCA divides the data to maximize intra-cluster differences and minimize inter-cluster differences. Despite the models like k-means where the decision is arbitrary or subjective, LCA facilitates statistical evaluation leading the result to be less subjective [95].

A significant difference between standard cluster analysis techniques and LCA is that LCA is a model-based clustering approach where a statistical model is postulated for the population from which the sample under study is taken [55]. It assumes that a

mixture of underlying probability distributions generates the data. When maximum-likelihood is used for parameter estimation, the model maximizes the log-likelihood function and finds the optimal solution. Another advantage of LCA is that the model is flexible to distribution and scaling of the variables [55] i.e., it can handle both simple and complicated distributional forms of variables, and its result remains the same irrespective of whether the variables are normalized or not.

Link-Based Cluster Ensemble(LCE)

Iam-On et al.[61] analyzed the problem of degradation associated with clustering quality and presented a link-based algorithm that improves the matrix computation for finding the unknown elements in the dataset using cluster ensemble similarity. It significantly extends the hybrid bipartite graph formulation (HBGF) technique by applying a graph-based consensus function to an improved cluster association matrix instead of the conventional binary cluster-association matrix. The focus has shifted from revealing the similarity among data points to estimating those between clusters accurately and inexpensively. For evaluating the similarity between the objects, Iam-on established three different similarity measures, which can perform with substantially fewer unknown entries as compared to conventional co-association matrix [62]. They are connected-triple-based similarity (CTS), SimRank-based similarity (SRS), and approximate SimRank-based similarity (ASRS). SRS extends the scope of similarity estimation beyond the local context of adjacent neighbours, with the assumption that neighbours are similar if their neighbours are similar as well. Here, a similarity measure between any pair of vertices can be computed through the iterative refinement process. Let SRS(a,b) be the initial SRS matrix, representing the similarity between any pair of data points or any two clusters in the ensemble. For a=b, SRS(a,b)=1. Otherwise,

$$\text{SRS(a,b)} = \frac{DC}{|N_a||N_b|} \sum_{a' \in N_a} \sum_{b' \in N_b} \text{SRS(a',b')}, \tag{2.10}$$

where DC is the constant decay factor within the interval (0, 1], N_x is the neighbour set whose members are directly linked to vertices.

ASRS, a variation of the SRS, improves the applicability and efficiency of the SRS

approach by removing the iterative process of similarity refinement.

Clustering the data using LCE involves the following main steps[61]:

1. Creating base clusterings to form a cluster ensemble
2. Generating a refined cluster-association matrix using a link-based similarity algorithm.
3. Producing the final data partition by employing the spectral graph partitioning technique as a consensus function.

Majority voting

Hanan G.Ayad and Mohamed S.Kamel [24] proposed a consensus clustering technique of cumulative voting in which they compared with a bipartite-based consensus algorithm with the inference that the voting technique had better accuracy, stability, and estimation of the actual number of clusters than the other. They considered this cumulative voting as a regression problem with linear computational complexity. Their method provided a probabilistic vote for each object on which cluster they should belong in the ensemble result. They are then thresholded to identify each object's membership in the ensemble clusters. This approach necessitates the employment of a mapping function between the selected reference member and the other members for which the information bottleneck concept was applied. The primary function of cumulative voting follows two approaches relabeling and aggregation of the consensus partition [24]. In relabeling, the most appropriate relabeled partitions are taken. The problem was viewed as a supervised learning problem with continuous response variables leading to a soft relabeled partition. In aggregation, the ensemble partition is run multiple times with random ordering.

2.5 Cluster Validation

Even though numerous clustering techniques are used in data mining, there is a need to evaluate the quality of the cluster to find the one that performs best for the input data despite the unavailability of class information. Validation methods are required to resolve the following situations [67]:

1. Determining correct the number of clusters;

2. Evaluating how well the clustering solution fits the given data without any external reference;

3. Comparing the results of cluster analysis to externally known results, such as labels;

4. Comparing two sets of cluster results to determine the better;

Internal and external validation techniques are the two sorts of validation procedures [57]. External validation compares the clustering result to a reference result that serves as the ground truth. If the result is comparable to the reference, we consider it to be a better clustering. This validation is easy when the similarity between two clusterings is well-defined; nevertheless, the fundamental drawback is that the reference result is not supplied in most real-world situations. As a result, external assessment is primarily utilized for synthetic data and fine-tuning clustering methods. Internal validation assesses the validity of the clustering by comparing it to the result itself, i.e., using just internal data and no external reference. It may also be used to calculate the optimal number of clusters for a given piece of data. This is considerably more realistic and efficient in many real-world circumstances since it does not rely on any presumed outside references, which are not always possible to get. With the massive rise in data quantity and dimensionality, it is not easy to assert that ground truth knowledge is available. In this study context, we did not have the ground truth partition of the data to which we could compare our solution.

Most of the internal validation indices are based on two criteria: compactness and separation. [67, 57]. The Compactness metric evaluates how closely data points in a cluster are clustered. The clustered points are meant to be connected by sharing a similar characteristic representing a meaningful practice pattern. Distances between in-cluster nodes are typically used to determine compactness. The variance, i.e., the average distance to the mean, is a typical technique for evaluating compactness to determine how objects are bound together with their mean as its centre [57]. The Separation metric evaluates how distinct the discovered clusters are from one another. A separate cluster that is dispersed among the rest correlates to a particular pattern.

Distances between objects, like compactness, are commonly employed to quantify separation, for example, pairwise distances between cluster centers or pairwise minimum distances between items in distinct clusters [57]. The internal validation metrics used in the study that is based on the above criteria are the Silhouette index, Dunn index, Calinski Harabarz, Compactness, and Connectivity. These indices are discussed briefly below:

Calinski Harabarz(CH)

The CH Index is a measure of how similar an object is to its cluster (Compactness) compared to other clusters (separation) [57]. The distances between data points in a cluster and its cluster centroid are used to assess compactness. In contrast, the distance between cluster centroids and the global centroid estimates separation. The degree of separation, or how far away the cluster centers are, is represented by the numerator of the CH. At the same time, compactness, or how close the in-cluster objects are grouped around the cluster center, is represented by the denominator. The CH index for K number of clusters on a dataset D =$[d_1, d_2, d_3, ..., d_N]$ is defined as,

$$\text{CH} = \left[\frac{\sum_{k=1}^{K} n_k \parallel c_k - c \parallel^2}{K-1}\right] / \left[\frac{\sum_{k=1}^{K}\sum_{i=1}^{n_k} \parallel d_i - c_k \parallel^2}{N-K}\right] \tag{2.11}$$

where, n_k and c_k are the no. of points and centroid of the k^{th} cluster respectively, c is the global centroid, N is the total no. of data points. Higher the value of this index indicates that the clusters are well separated.

Dunn Index

Dunn index identifies sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance [62]. Dunn index ranges from zero to infinity and higher the value, better is the clustering.

$$\text{Dunn Index} = \min_{1\le i\le c}\left\{\min_{1\le j\le c, j\neq i}\left\{\frac{\delta(X_i, X_j)}{\max_{1\le k\le c}\{\Delta(X_k)\}}\right\}\right\} \tag{2.12}$$

where, $\delta(X_i, X_j)$ is the intercluster distance between the cluster X_i and X_j and $\Delta(X_k)$ is the intracluster distance of the cluster X_k.

Silhouette Index

Similar to the previous metrics, the Silhouette index also evaluates the compactness and separation of the clusters. The silhouette index indicates the average distance to objects in the same cluster and the distance in the alternate clusters. The silhouette index is calculated as:

$$S(i) = (b(i) - a(i))/(\max\{a(i), b(i)\}) \tag{2.13}$$

where a(i) represents the average dissimilarity of i^{th} item to all other objects in the same cluster and b(i) represents the average dissimilarity of i^{th} object to all objects in the nearest cluster.

The silhouette range lies between -1 and 1; high values represent good clustering with no overlaps between the clusters [13]. If the silhouette value is close to 1, the sample is well-clustered and already assigned to a very appropriate cluster. If the silhouette value is close to zero, the sample is located equally far from both clusters, indicating overlapping clusters. If the silhouette value is close to –1, the sample is misclassified and placed between the clusters.

Compactness

This metrics measures the closeness of the objects with the same cluster. It has value between 0 and infinity and the lower value of compactness indicates better clustering. Nguyen and Caruana[84] computed compactness as follows:

$$\text{Compactness} = \frac{1}{N}\sum_{k=1}^{K} n_k\Big(\sum_{\frac{x_x, x_j \xi \text{iC}_k}{d(x_i, x_j)}} n_k(n_k - 1)/2\Big) \tag{2.14}$$

where $d(x_i, x_i)$ is the distance between X_i and x_j, N is the number of data points, whereas K denotes the number of clusters.

Connectivity

Connectedness refers to the amount to which observations are grouped in the same cluster as their nearest neighbours in the data space, and it is assessed in this context by the connectivity. [30]. The value of connectivity is computed using the below equation:

$$\text{Connectivity} = \sum_{i=1}^{N} \sum_{j=1}^{L} x_{i,nn_{i(j)}} \tag{2.15}$$

Where N is the total number of records considered, L denotes the number of nearest neighbours considered, and $(nn_{i(j)})$ is the distance of the j^{th} nearest neighbour to the object i.

The value of $x_{i,nn_{i(j)}}$ will be zero if i and j are from the same cluster and $1/j$ if other wise.

2.5.1 Finding optimal number of clusters

In an unsupervised learning problem, the target variable is not known, so it is essential to determine the optimal number of clusters that can be formed in the given dataset. This metric is evaluated using the silhouette method, elbow method, BIC and PAC scores. These measures are introduced below in detail.

The elbow method is the most common method for determining the optimal number of clusters. It works by calculating the Sum of Squared Errors(SSE) for each value of k and then plotting a line chart with the calculated values.SSE is the sum of the squared differences between each observation and its group's mean. It may be measured using any distance metric to measure variance within a cluster. If all examples inside a cluster are identical, the SSE is equal to zero.[6]. If the line chart looks like an arm, then the "elbow" on the arm is the optimal value of k as it represents the k value where the SSE is low.

Bayesian information criterion (BIC) score is a method for scoring a model which is using the maximum likelihood estimation framework(e.g., GMM). The BIC statistic is calculated as follows:

$$\text{BIC} = (k * \ln(n)) - (2\ln()L) \tag{2.16}$$

where L is the model's maximal likelihood function, k is the number of parameters, and n is the number of records. The BIC considers both the fit of the model to the data and the complexity of the model. The lower the BIC score, the better is the model.

Cumulative Distribution Function(CDF) plots histogram ranging from 0 to 1 using consensus matrix to identify the optimal number of clusters and assess the stability of the clusters[81]. For the histogram given, the CDF defined over the interval [0,1] is defined as follows:

$$\text{CDF(c)} = \frac{\sum_{i<j} l\{\mathcal{M}(i,j) \leq c\}}{N(N-1)/2} \tag{2.17}$$

where 1... denotes the indicator function, $\mathcal{M}$(i, j) denotes entry (i, j) of the consensus matrix $\mathcal{M}$, and N is the number of rows (and columns) of$\mathcal{M}$.

The CDF graph curves illustrate a step function across 0, a flat line passing between 0 and 1, and a second step function around 1. If the curve gradually climbs and constitutes a different shape, then the clusters formed lack stability characteristics. If the CDF curves form a bimodal shape, it estimates the presence of significant clusters. The lower left portion represents sample pairs rarely clustered together; the upper right portion represents those almost always clustered together, whereas the middle portion represents those with occasional co-assignments in different clustering runs. To capture the features of CDF curves, Senbabaoğlu et al. [96] introduced a new index called Proportion of Ambiguous Clustering(PAC) which is defined as the fraction of sample pairs with consensus index values falling in the intermediate sub-interval (x1, x2) $\in$ [0, 1]. A low value of PAC indicates a flat middle segment, allowing inference of the optimal K by the lowest PAC.

2.6 Data Imputation

Many clinical research datasets have a significant percentage of missing values, which are generally represented as blanks, NaNs, or other placeholders, which directly impacts their use with machine learning algorithms to provide better results. As a result, it is critical to analyze the impact of missing data [80]. Rubin [92] formulated three possible missing data mechanisms: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). MCAR is the

highest level of randomization, implying that the missing value pattern is entirely random and does not depend on any variable that may or may not be included in the analysis. The observed information in the dataset determines the likelihood of missing data in MAR. It indicates that the likelihood of missingness is determined by the observed information but not by the unseen component. In this case, the pattern of missing data may be deduced from the observed values in the dataset. Missingness in MNAR is influenced by unobserved data rather than observable data. When data is MNAR, the likelihood of missing data is proportional to the value of the missing data [65]. The missing data pattern is not random and is not predictable based on the observed values of the other variables in the dataset.

One fundamental approach for working with incomplete datasets is to ignore/delete entire rows or columns with missing values. However, this comes at the price of potentially crucial data being lost. When the dataset includes a limited number of missing values, this strategy is suitable. There are two methods for disregarding missing data: listwise deletion and pairwise deletion [65]. The listwise deletion method eliminates any observations that have missing values for any variable of interest. As a result, this technique restricts the analysis to instances for which all values are observed, which frequently leads to skewed estimates and loss of precision. [94]. In pairwise deletion, we analyze all instances that include the variables of interest. It does not exclude the whole record but instead uses as much data from each unit as feasible. The advantage of this technique is that it preserves as much data as possible for analysis, even if some of its variables have missing values. The disadvantage of this technique is that it employs a different sample size for each variable. [94].

A preferable method is to impute the missing values or predict them from the known portion of the data [56]. Numerous imputation approaches seek to offer an accurate estimation of population characteristics to maintain the power of data mining and data analysis tools. The amount of missing data determine the optimal solution for missing data. Although there is no rule for determining what proportion of missing data is unacceptable, it is always best to compare findings before and after imputation if more than 25% of the data is missing. [65].

There are two types of data imputation methods: single imputation methods and multiple imputation methods. The Single Imputation Method entails imputing

one reasonable value for each missing value of a specific variable in the dataset and then analyzing as if all data had been seen initially. Imputation with the constant, Mean Imputation, Imputation with distributions, Regression Imputation, and kNN Imputation are some common single data imputation methods. [65]. The precision is exaggerated in the single imputation technique since it is assumed that the single imputation value is correct. There can, however, never be complete confidence regarding the correctness of imputed data. As a result, uncertainty about these imputed values must be included in missing data techniques [73]. Instead of replacing each missing observation with a single value, multiple imputations replace several probable values to represent uncertainty about the correct values to impute. As a result, the Multiple Imputation technique produces "m" distinct complete datasets with observed and imputed values. One such method is Multiple Imputation by Chained Equations(MICE), which works under the assumption that the data are missing at random. Implementing MICE when data are not MAR could result in biased estimates [25].

Several research has been conducted to determine the most effective data imputation strategy. Kyureghian et al. [69] performed a study that evaluated imputation strategies by assessing the inaccuracy of predicting missing data and parameter estimations from later regression analysis. The paper's findings demonstrate that multiple imputation approaches provide the best coverage of parameter estimations and dependent variable prediction. For gene expression data, Troyanskaya et al. [103] compared k-nearest neighbour imputation to mean imputation and singular-value decomposition (SVD). Their analysis revealed that the KNN impute method outperforms the mean imputation and SVD methods. Malarvizhi and Thanamani [75] discovered that median or standard deviation substitution outperforms mean substitution in a comparative analysis of single imputation techniques. Penone et al. [85] evaluated the performance of four approaches for estimating missing values in trait databases (K-nearest neighbour, multivariate imputation by chained equations (MICE), missForest and Phylopars) and tested whether imputed datasets retain underlying relationships in the life-history trait datasets.

Although there are various ways for data imputation, it is also critical to understand and assess the performance of different imputation methods on the source

data to utilize the right approach when doing data mining tasks. Though some work has been done to examine the performance of various imputation methods, we intend to examine the performance of different imputation methods that use single and multiple imputation methods, namely mean imputation, median imputation, mode imputation, kNN imputation, constant imputation, and multiple imputations with various regression models on our data in this study. A detailed explanation of the imputation techniques used in this study is presented in the rest of this section.

2.6.1 Univariate Imputation

Simple (or) univariate imputer imputes the values in the i-th feature dimension using only non-missing values in that feature dimension. We have utilized SimpleImputer class from sklearn for this purpose. This class imputes the missing value with the provided constant value or uses the column's statistics in which the missing values are located. The commonly used statistics in SimpleImputer are: mean, median, and mode. This study evaluated the effect of imputing the missing value with mean, median, mode, and constant values.

2.6.2 KNN Imputation

KNN imputer scans a dataset for k nearest rows to the row with missing values. The neighbours' features are averaged evenly or weighted by distance to each neighbour. If a sample lacks more than one attribute, the neighbours may change based on the particular feature being imputed. When fewer than n neighbours are available, the training set average is used during imputation. Suppose at least one neighbour with a known distance, the remaining neighbours' weighted or unweighted average will be utilized during imputation. We have utilized KNNImputer class from sklearn for this purpose. After evaluating several values for the number of neighbours, we have selected five as the number of neighbours.

2.6.3 Multivariate Imputation

Multiple imputations estimate imputation by modelling each feature with missing values as a function of other attributes. It does so in an iterated round-robin method,

with each step selecting a feature column with null as output y and the remaining feature columns as inputs X. For knowing y, a regressor is fitted on (X, y). The regressor is then used to anticipate the missing y values. This process is done iteratively for each feature and then repeated for n imputation rounds provided by the user. The last imputation round results are returned. We have used IterativeImputer class with various regressors such as Bayesian ridge, decision trees, and gaussian process in our study for evaluation. The function of this class is inspired by the MICE package [25], but it varies in that it returns the outcome of the most recent imputation rather than multiple imputations.

2.6.4 Performance Evaluation

The most common evaluation metrics used are raw Bias, percent bias, coverage rate, average width, and root mean square error(RMSE) [105]. Among them, we have utilized RMSE to evaluate the performance of the imputation techniques. It evaluates the accuracy and precision of the predicted value by comparing the predicted value with the actual value. RMSE is computed as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(predicted_i - actual_i)^2}{N}} \tag{2.18}$$

where N is the number of non-missing data points.

2.7 Data Classification

Classification is a supervised pattern recognition problem in which unknown examples are categorized into pre-existing categories or classes. In other words, each data item in the supplied dataset has a class label, which is often assigned by an expert or derived from prior knowledge. As a result, the entire set is divided into various subgroups known as classes. With the help of these pre-categorized training datasets, classification models estimate the likelihood or probability that the future dataset will fall into one of the predetermined categories. Similar to clustering, classification has a wide range of application domains, including computer vision, handwriting recognition, speech recognition, geostatistics, drug discovery and development, document classification, and much more [35]. There are several classification models used in

the literature. We chose Support Vector Machines, Gaussian Naive Bayes, Decision Tree, Random Forest and KNeighbors classifier in this study. A detailed description of these algorithms is present below:

The rest of this section provides a detailed description of the classification algorithms used in this study.

2.7.1 Support Vector Machines

Support Vector Machine is a supervised machine learning model used for classification and regression. In SVM, each data point is mapped to an n-dimensional space, where n is the number of features. The goal is to create the best line or decision boundary that can segregate n-dimensional space into classes while maximizing the marginal distance for both classes and minimizing the classification errors[104]. This best decision boundary is called a hyperplane. A class's marginal distance is the distance between the decision hyperplane and its nearest instance, a member of that class. Thus for classifying the data using SVM, each data point is plotted as a point in an n-dimension space, with the value of each feature being the value of a specific coordinate. Then a hyperplane that differentiates the classes by a maximum margin is identified to group the data.

2.7.2 Gaussian Naive Bayes

The Bayes theorem is the foundation of the Naive Bayes classification algorithm. This theorem can characterize the likelihood of an occurrence depending on prior knowledge of the event's circumstances[7]. One assumption taken in this model is the strong independence assumptions between the features. That is, these classifiers assume that the value of one characteristic is independent of the value of any other feature and that the presence or absence of one feature has no effect on the presence or absence of any other feature. It forecasts membership probabilities for each class, such as the likelihood that a given record or data point belongs to a specific class. The most accurate class is the one with the highest probability. This classification model is used when the dimensionality of the inputs is high. Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. This approach creates a simple model by assuming that the data is

described by a gaussian distribution with no covariance between the dimensions. This model is fit by comparing the mean and standard deviation of the points within each label. The probability of an attribute x for the class y with mean σ_y^2 and variance μ_y is computed as follows:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.19}$$

2.7.3 KNeighbors Classifier

The k-Nearest Neighbor (KNN) algorithm is one of the most widely used classification algorithm due to its simplicity and easy implementation. It is an instance-based learning algorithm as it does not learn weights from training data like model-based algorithms but uses the entire training set to predict the class of the unseen data[8]. It is also called a lazy learning algorithm as it postpones the learning process from the training data until the prediction is required for the new instances[8]. This method assumes that similar objects exist nearby and assumes the similarity between the new data and the available neighbours before assigning the new data to the class based on the majority vote from its neighbours. The similarity is measured in terms of distance, where the distance metric like euclidean, Minkowski, and manhattan can be used. The K in KNN represents the K neighbours for voting, and selecting different K values produces different classification results.

2.7.4 Decision Trees

A decision tree is a graphical depiction of the design process used to decide the class of a given feature. Each tree node can be either a leaf node containing a class name or a decision node containing an attribute test with many branches to another decision tree for each potential attribute value. In short, it is a graphical representation for getting all the possible solutions to a problem based on given conditions. A decision tree asks a yes or no question and divides the tree into sub-trees based on the answer. It is named a decision tree because, like a tree, it begins with the root node and then develops on subsequent branches to form a tree-like structure. The essential advantage of employing decision trees is that they are simple to grasp since they resemble human decision-making capacity. The decision tree is implemented using

the ID3(Iterative Dichotomiser 3) algorithm, which divides the dataset based on the attribute and the right place to stop splitting[4]. This algorithm uses a top-down greedy approach to build the trees where it selects the best feature at each step using information gain. Information Gain computes the decrease in entropy (the measure of disorder in the target feature of the dataset) and assesses how effectively a particular feature separates or classifies the target classes. The feature with the most significant Information Gain is chosen as the best. The entropy is computed as follows:

$$\text{Entropy(S)} = -\sum_{i=1}^{n} p_i(\log_2 p_i) \tag{2.20}$$

where p_i is the probability of choosing class i and n is the number of classes. Using the entropy the information gain of the feature A is computed as :

$$\text{IG(S,A)} = Entropy(S) - \sum((|S_v|/|S|) * Entropy(S_v)) \tag{2.21}$$

Where S_v is the set of rows in S for which the feature column A has value v, $|S_V|$ is the number of rows in S_v, and likewise |S| is the number of rows in S. The procedure of determining the feature with the highest IG and making it a node is repeated until we have exhausted all features or the decision tree includes all leaf nodes.

2.7.5 Random Forest

A random forest (RF) is an ensemble classifier made up of numerous Decision Trees. Deep decision trees frequently result in overfitting the training data, leading to a significant variance in classification results for a bit of change in the input data[104]. They are pretty sensitive to their training data, which renders them prone to making mistakes on the test dataset. A random forest's decision trees are trained using different sections of the training dataset. The sample's input vector must be sent down with each decision tree in the forest to categorize a new sample. Each decision tree then evaluates a distinct segment of the input vector and returns a categorization result. The forest then chooses the categorization with the most' votes'. Since this algorithm considers the outcomes from many different decision trees, it can reduce the variance resulting from considering a single decision tree for the same dataset.

There have been numerous researches involving classification models for pattern

recognition, especially with the diabetes dataset. Some of them are listed as follows: Amol et al. [28] used MLP to classify retinal images in detecting diabetic retinopathy, one of the severe conditions caused by diabetes. The model classified the retinal images as normal or abnormal. The training and cross-validation rates by the MPL were 100% for the detection of normal and abnormal retinal images. Pethunachiyar [87] classified diabetes patients and identified patients with early stages of diabetes using the support vector machine model(SVM). The author used kernel-based SVM with linear, polynomial, and radial kernel parameters and found SVM with linear function to be more accurate in classifying the data. Maniruzzaman et al. [76] used a machine learning paradigm to classify and predict diabetes. They utilized four machine learning algorithms, i.e., naive Bayes, decision tree, AdaBoost, and random forest, for diabetes classification. They applied the models to a US-based National Health and Nutrition Survey data of diabetic and nondiabetic individuals and achieved promising results for a Random Forest classifier.

The presence of various classifiers in the literature necessitates the evaluation metrics to assess their performance in finding the optimal classifier for the particular dataset. The correct evaluation of learned models is one of the most critical issues in pattern recognition. One side of this evaluation can be based on statistical significance and confidence intervals when we claim that one method is superior to another. The other side of evaluation relies on which metric is used to evaluate a learned model. These measures are calculated by comparing the expected class label to the problem's predicted class label. Evaluation metrics are critical in measuring classification performance and directing classifier modelling. Standard metrics for assessing classification prediction models, such as classification accuracy or classification error, are frequently employed. Standard metrics perform well for the majority of issues, but they make certain assumptions about the problem.[14]. As a result, an assessment metric that best conveys what is relevant about the model or forecasts must be chosen, which makes selecting model evaluation metrics difficult. When the class distribution is skewed, this issue becomes significantly more difficult. The reason for this is that when classes are uneven and exceptionally substantially lopsided, many of the conventional measurements become inaccurate or even deceptive.[14]. Therefore we have carefully chosen precision, recall,f-score and AUC-ROC curves as the evaluation

metrics that work well with imbalanced classes. Ferri et al. [45] classified various evaluation metrics into three sets, namely qualitative, ranking, and probability metrics. Qualitative metrics(e.g., accuracy, F1-measure, kappa statistic) are used to minimize the number of errors caused by the model. Probabilistic metrics(e.g., mean absolute error, mean squared error, LogLoss) are primarily used to check the reliability of the classifiers where it not only measures when the model fails but also checks whether it has selected the wrong class with a high or low probability. Rank metrics(e.g., ROC curve, ROC Analytics) are more concerned with evaluating classifiers based on how effective they are at separating classes. We used precision, recall, f1-score, and AUC-ROC to evaluate the classification models among these evaluation metrics.

Precision is a measure that quantifies the amount of valid optimistic predictions made, i.e., it informs us how many predicted samples are relevant, i.e., our errors in identifying a sample as correct if it is not valid. When reducing false positives is the goal, precision is calculated. It is calculated as the ratio of accurately anticipated positive instances divided by the total number of positive examples predicted.

$$\text{precision} = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.22}$$

Recall, sensitivity, or true positive rate is a measure that quantifies the amount of correct positive predictions produced out of all possible positive predictions. In contrast to precision, which only comments on the accurate positive predictions out of all positive predictions, recall indicates missing positive predictions. When reducing false negatives is the goal, recall is calculated. In the case of uneven learning, recall is often employed to assess minority class coverage.

$$\text{recall} = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.23}$$

The F-score combines accuracy and recalls into a single metric that incorporates both features in a balanced manner. This measure is appropriate for unbalanced classification, and its value spans from [0,1], where 1 represents a perfect result and 0 represents a catastrophic failure.

$$\text{f-score} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.24}$$

The metrics precision, recall, and f-score are computed using the functions provided by the sklearn metrics package.

The ROC (Receiver Operating Characteristic) curve, which is a valuable tool for forecasting the likelihood of a binary event, represents the false positive rate vs. the true positive rate (sensitivity) for various threshold values ranging from 0.0 to 1.0. The false-positive rate is also known as the inverted specificity. Specificity is defined as the sum of true negatives and false positives divided by the total number of true negatives.

$$\text{specificity} = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{2.25}$$

$$\text{False Positive Rate} = 1 - specificity \tag{2.26}$$

The Area Under the Curve (AUC) measures a classifier's ability to discriminate between classes and summarise the ROC curve. The greater the AUC, the better the model's ability to differentiate between positive and negative classifications. When AUC = 1, the classifier can successfully distinguish all Positive and Negative class points. If, on the other hand, the AUC was 0, the classifier would forecast all Negatives as Positives and all Positives as Negatives. When AUC1 is 0.5, there is a good possibility that the classifier will discriminate between positive and negative class values. This is due to the classifier's ability to recognize more True positives and True negatives than False negatives and False positives. When AUC = 0.5, the classifier cannot differentiate between Positive and Negative class points. The classifier predicts either a random class or a constant class for all data points. The ROC curve, which is often used for binary classification issues, may be extended to multi-class situations by employing the one vs. all technique[2]. This method fits one classifier per class, with each classifier fitting against all other classes.

2.8 Summary

We covered the history of the critical ideas utilized in this chapter. A variety of problems with clustering algorithms have been described in the literature, including that various clustering structures can be obtained by single clustering methods with varied parameters or many algorithms. The clustering ensemble approach is developed to solve the inherent problems with single clustering methods. It is the

technique of merging a collection of partitions created from the same data to produce a single better data partition. The clustering ensemble's core process consists of two significant steps: creation and consensus. The generation phase generates several ensemble members from the same data, merged in the consensus step using a consensus function. According to the review, the consensus function is the essential component in a clustering ensemble since it decides whether or not an ensemble is successful. We looked at some of the most frequent consensus functions in the literature. This chapter also covers the necessity for cluster validation and the many validation approaches utilized in the literature. We then overviewed the basics of classification models and their evaluation metrics. We have also examined the impact of missing values in datasets and methods for improving the loss. We end the chapter by discussing various clustering applications in diabetes control.

Chapter 3

Data Description and Methodology

This chapter presents the dataset being used and the research methodology for this study.

3.1 Dataset Description

The dataset used in this study contains 1,209,502 rows that present the results of around 940 unique laboratory tests performed on 7726 diabetic individuals over six months from January to June of 2017. It is time-series data containing the date and time when the patients took a particular test. It also includes other information such as the unique patient number, medical record number, patient visit number, examination code, examination name and hospital information. We decided to move forward with just the patient ID, exam name, result of the tests, and the timestamp from all the available features. A detailed description of the features present in the dataset, along with a sample record, is provided in the table 3.1

Attribute	**Description**	**Type**	**Sample**
ETPR_PT_NO	Unique patient identity number	Numbers	832176
EXM_NM	Name of the test	Characters	Urea Nitrogen, Blood(BUN)
BRFG_DTM	Date and time of the test taken	Timestamp	2017-02-20 09:30:36
RESULT	Result of the test taken	Numbers	10.5

Table 3.1: Description of the Dataset

The distribution of the tests taken across the timeframe is presented in Figure 3.1. From the plot, we could infer that the highest and the least amount of tests were taken in May and June months, respectively. Figure 3.2 shows the distribution of the count of unique tests taken by the patients. By visualizing this plot, we could infer that not

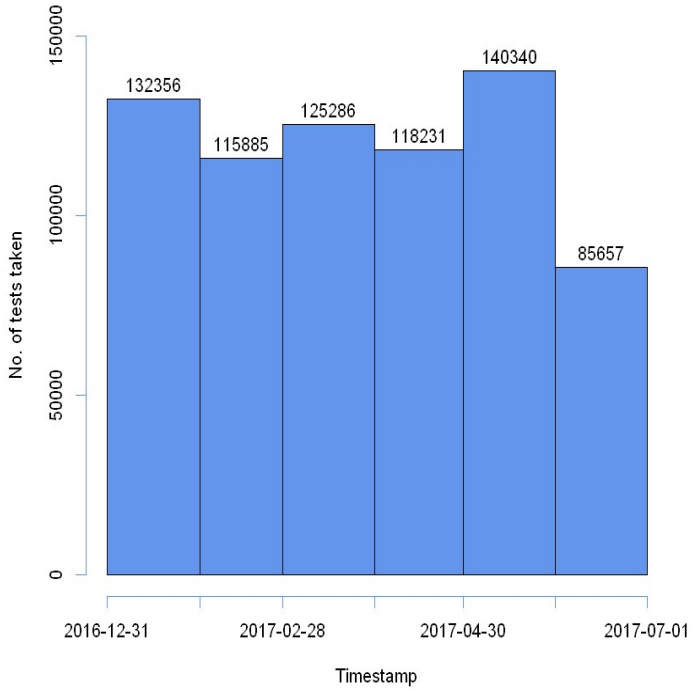

Figure 3.1: Distribution of tests over time

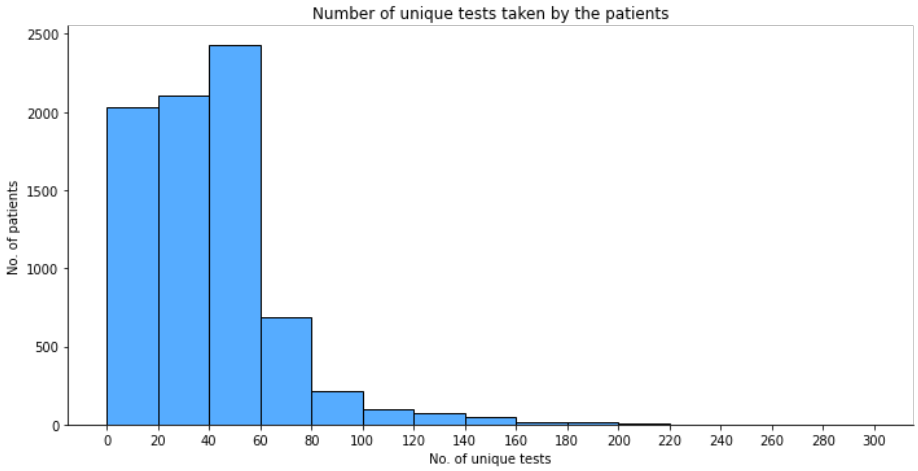

Figure 3.2: Distribution of the number of unique tests taken by a patient

all the patients have taken all the 940 tests as the maximum number of unique tests taken by an individual did not exceed 220. Figure 3.3 shows the patient numbers who have taken a higher number of unique tests.

We observed that most of the patients had taken specific tests repetitively daily by analyzing the dataset further. Some patients have taken the same test multiple times on the same day. Furthermore, the number of tests performed on each patient

ETPR_PT_NO	count
228940	215
424145	212
514232	201
831178	204
3025200	204

Figure 3.3: Patient numbers and the maximum number of unique tests taken by them

and their frequency in the observed period differ from person to person. Thus, these observations depict that the data points are sparse in a time series.

Test	**Patient count**	**Repeats**
AGAP	5254	233
Alanine Transferase(SGPT)	4635	176
Aspartate Aminotransferase (SGOT)	4653	176
Carbon Dioxide Level (CO2)	5267	237
Chloride Level, Serum (Cl Level)	5267	234
Creatinine Level, Serum	5414	237
Haematocrit (Hct)	5007	208
Hemoglobin (Hgb)	5007	210
MCHC	5004	208
MCV	5004	208
MPV	5004	208
Magnesium(Mg) Level, Serum	4639	181
Platelet Count (Plt Count)	5003	208
Potassium Level, Serum (K Level)	5338	238
RBC	5005	208
RDW	5004	208
Sodium Level, Serum (Na Lvl)	5267	239
Urea Nitrogen, Blood(BUN)	5365	236
White Blood Cell Count (WBC Count)	5004	206
eGFR	4764	277

Table 3.2: Twenty most common tests among the patients with the count of the maximum number of unique patients attempted it and its highest repetitions on a single patient.

Since most of the tests were rarely taken by the patients, we have narrowed down the dataset and worked with up to 20 most frequently taken tests which are listed in

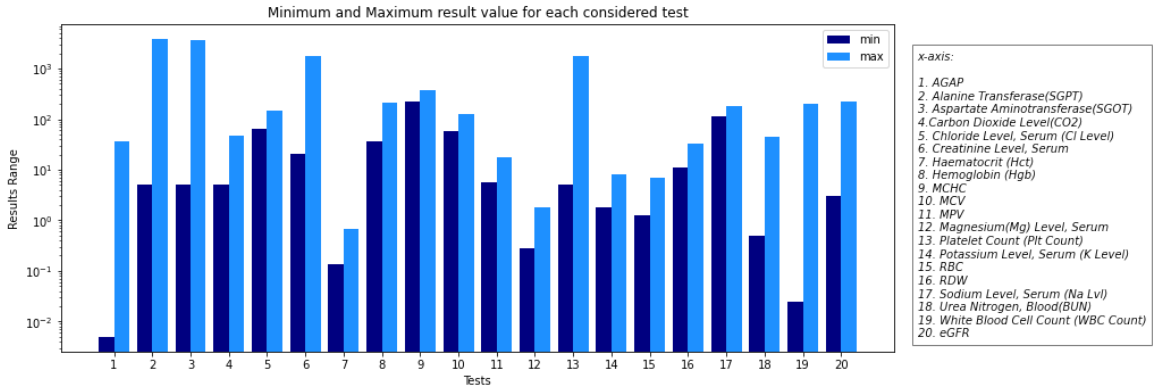

Figure 3.4: Minimum and Maximum result observed for considered tests

table 3.2. We chose 20 tests because further tests reduced the overall count of the patients who took the filtered tests. That is, when we considered 20 frequent tests, we were left with 1001 patients who have taken all the 20 tests, whereas if we consider the top 30 frequently taken tests, we are left with just 81 patients who have taken all the considered 30 tests. Table 3.2 also contains the maximum number of times an individual took the test. The minimum and maximum RESULT values of each of the considered tests can be seen in Figure 3.4. Using these top 20 tests, we created three distinct data subsets: Dataset 1 includes patients who have taken at least one of the top ten tests, Dataset 2 includes patients who have taken at least one of the subsequent ten top tests, and Dataset 3 includes patients who have taken at least one of the top twenty regularly taken tests. These three subsets are used for identifying patient phenotypes. Thus, the final dataset considered contains 20 most commonly taken tests and the records of the patients who have taken them. Figure 3.5 presents sample of the final data.

3.2 Research Methodology

The study comprises four phases: data preparation, clustering, data imputation, and classification. Figure 3.6 depicts the methodology followed in the project.

The *data preparation* phase comprised the following tasks:

- **Data pre-processing:** The raw time-series laboratory data of diverse diabetic patients are analyzed in this step for any discrepancies such as undesired columns, duplicate, irrelevant, or incomplete records, which are then removed

	ETPR_PT_NO	MRN	PACT_ID	EXM_CD	EXM_NM	BRFG_DTM	RESULT
0	3092	520	17226003	L3900102	AGAP	2017-01-02 12:06:42	15.00
1	3092	520	10994098	L3900102	AGAP	2017-03-05 14:14:17	12.00
2	3092	520	20490933	L3900102	AGAP	2017-05-21 16:17:26	11.00
3	3092	520	17226003	L3900109	Alanine Transferase (SGPT)	2017-01-01 10:00:54	58.00
4	3092	520	17226003	L3900109	Alanine Transferase (SGPT)	2017-01-02 10:26:54	55.00
...	...	...	...	...	...	...	...
265889	99999916	2001951	24991721	L2000008	RBC	2017-06-12 14:33:13	4.68
265890	99999916	2001951	14078621	L2000014	RDW	2017-02-09 12:46:20	13.20
265891	99999916	2001951	24991721	L2000014	RDW	2017-06-12 14:33:13	13.10
265892	99999916	2001951	14078621	L2000016	White Blood Cell Count (WBC Count)	2017-02-09 12:46:20	4.16
265893	99999916	2001951	24991721	L2000016	White Blood Cell Count (WBC Count)	2017-06-12 14:33:13	3.07

265894 rows × 7 columns

Figure 3.5: A glimpse of the final dataset being used in the study

from the dataset. The outcome of this step is a cleaned dataset with no discrepancies in the data.

- **Data Transformation:** Since the distribution of data points in the time series is sparse, the cleaned data are resampled and interpolated. These preprocessed and transformed data are then used to extract characteristics and are considered the dataset's new features. The characteristics computed are mean, slope, peak to peak distance, variance and mean absolute change.

- **Data partitioning:** Once the data is regularized and transformed, the next step is to partition the data into two cohorts: base set(cohort1) and incomplete set(cohort2). The cohorts are separated based on the tests considered. The base set contains data of patients who have undergone all the considered tests more than once, whereas an incomplete set contains records of patients who have taken at least one of the considered tests. For example, If we have considered 20 tests, a patient in the base set would have taken all the 20 tests at least once, whereas a patient in the incomplete set would have many null entries as its patients have not have undergone all but at least one of the 20 tests. The base set is used for identifying patient phenotypes as it possesses results for all the considered tests, whereas the incomplete set is used for cluster prediction, where the missing outcomes of the unattempted tests are imputed using the

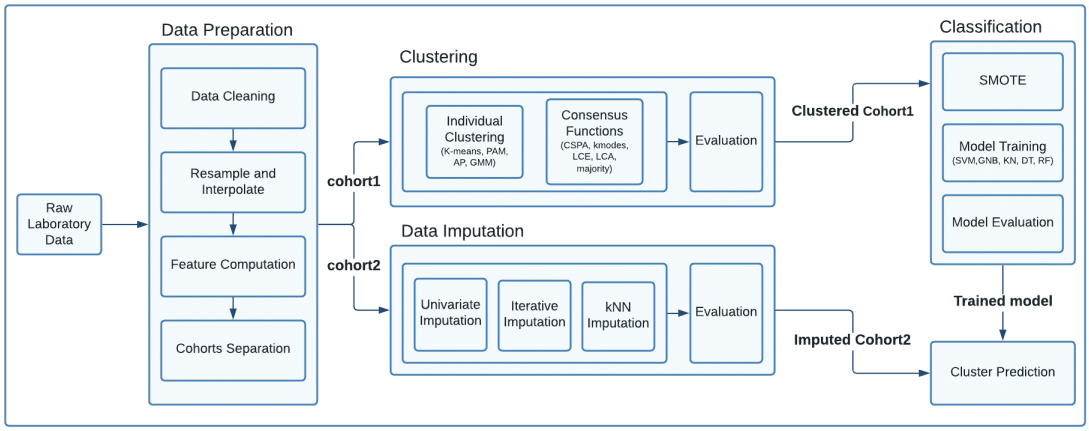

Figure 3.6: Research methodology of the project

known values. The main reason for data partitioning is to utilize the available data effectively.

The *clustering phase* possess the following steps:

- **Conventional clustering:** In this step, clustering algorithms such as K-means, Affinity Propagation, Partitioning Around Medoids, and Gaussian Mixture Models are applied to the base set for grouping the data based on their temporal characteristics.

- **Ensemble clustering:** Ensemble clustering: Here, we subsample the base set by randomly picking 80% from it and then apply the above-mentioned clustering methods iteratively for n number of times. The cluster assignments made by individual algorithms for each recursion are ensembled using various consensus functions like CSPA, kmodes, majority, LCA, and LCE. This step is carried out to improve the performance of the model.

- **Cluster Validation:** The quality of the clusters formed by individual clustering algorithms and various consensus functions are evaluated in this step. Since the actual labels of the data are unknown, we used internal evaluation metrics to assess the quality of the data partitions as it relies only on the information in the data. We also investigated and applied various assessment metrics to identify the optimal number of clusters 'k'. The 'k' groups formed by the optimal clustering method represent the phenotype of the diabetes patients for the

considered tests. After identifying the patient phenotypes, the clustered first cohort data is subjected to training the classification models.

The *imputation phase* involves the following steps:

- **Data Imputation:** The second cohort data cannot be grouped due to the vast number of missing entries. To prepare the data for cluster prediction, we employed the imputation approach to replace nulls with predicted values. We used a variety of imputation approaches for this goal, such as basic imputation techniques (zero, mean, or median imputation), iterative imputation (with various regression models), and k- nearest neighbour imputations.

- **Evaluation:** To impute the incomplete set close to the base set data, we have evaluated the considered imputation strategies on the base set by randomly removing its entries and identifying the one which imputes the missing value of the base set with minimal error. The efficacy of each technique is measured using Root Mean Square Error (RMSE), and the option with the lowest error rate is chosen. Finally, the selected imputation technique is applied to the incomplete set to fill its missing values. This non-null dataset is then sent to the project's next step to categorize then into relevant classes.

The final step, *Data classification*, involves the below steps:

- **Model Training:** Since the values in the second cohort are not 100 percent accurate, the imputed data from the second cohort cannot be grouped in the same way as the data from the first cohort. As a result, we employ a classification strategy to separate the patients in the second cohort, where the clustered first cohort is utilized for training and validating the models. One significant difficulty with training classification models with data from the first cohort is that the clusters formed are imbalanced, implying that the class distribution is not uniform. Most machine learning algorithms perform poorly, particularly when learning minority classes, since there are too few instances for the model to understand the decision boundary successfully [28]. We used the Synthetic Minority Oversampling Technique(or SMOTE) to oversample minority classes to address this issue. Before oversampling, the base set data is split into the

train and test sets and the SMOTE is applied just on the training set. The oversampled training data is then used to train Support Vector Machine, Gaussian Naïve Bayes, KNeighbors classifier, Decision Trees and Random Forest classification models.

- **Model Evaluation and prediction:** The models trained on the base set are then evaluated by making it to predict the test set clusters, and its prediction correctness is evaluated using precision, recall, f1-score and ROC-AUC validation measures. The identified best model is then used to forecast the disease progression of incomplete set patients by assigning them to appropriate clusters.

Thus, as a result, all the available patients in the dataset are grouped based on their temporal characteristics. While base set patients serve as the base for identifying distinct phenotypes, assigning the incomplete set patients will help us predict a value for the test that the incomplete set patient has not taken yet. This predicted value, in turn, will help decide if the patient needs to take a particular that he missed taking, i.e. a patient might need immediate attention if the predicted value of a test is beyond its normal range. Thereby, this will help us predict the progression of the disease.

Chapter 4

Methods

This chapter discusses the machine learning methods employed in our research. Section 4.1 outlines the data preparation steps carried out to make the data ready for analysis. Following this, Section 4.2 discusses our cluster analysis methods to identify the patient phenotypes among diabetes patients. Finally, sections 4.3 and 4.4 illustrates how imputation techniques are applied to the base dataset to perform classification for tracking disease progression among diabetes patients.

4.1 Data Preparation

This section discusses the steps taken to prepare the data for analysis.

4.1.1 Data Preprocessing

Data preprocessing is the preliminary step in data mining that takes all of the available information and transforms it into cleaner information that is more suitable for the study. Data-collection procedures are frequently poorly managed, resulting in out-of-range values, missing data, etc. Analyzing data that has not been thoroughly checked for such issues can produce deceptive results [44]. Thus, analyzing data quality should be the foremost step before running any analysis. The most crucial stage of preprocessing is to get rid of bad records. Bad records here refer to duplicates, irrelevant entries, and missing values. Deleting random data without proper knowledge of the dataset might affect the overall informativeness of the data. Hence, care must be taken while cleaning the data.

The diabetes dataset considered in this study contains some duplicate entries dropped during cleaning. The features like patient number (ETPR_PT_NO), examination name (EXM_NM), timestamp (BRFG_DTM), and Results (RESULT) are the main focus of the study. The features other than the primary ones are dropped from the dataset. The RESULT column being the main feature, seems to possess

many null entries. These patients with no RESULT values are removed from the dataset. Also, a few duplicate exam names are suffixed with special characters '%' and '#'. While looking into it, we found that both of the values represent the test result except that '%' is the value in percentage whereas '#' is the numerical value. Since the remainder of the test results is represented using numerical, we have dropped the records suffixed with '%' and removed the '#' character from the EXM_NM. After cleaning the dataset, we are now left with 7485 unique patients and 510 examination names.

As discussed earlier in section 3.1, since many of the tests are not taken by a majority of the patients, we have shrunk our dataset by selecting up to 20 most commonly present tests. A test is considered common if it occurs at least once in a maximum number of individuals. Using these popular tests, we have filtered the dataset by retaining the patients that have taken at least one of these tests. Thus, after filtering the data based on EXM_NM, we had 3281 patients who took at least one of the considered tests. The tests considered are mentioned in Table 3.2. The next step in preparing the data would be to compute temporal characteristics from the dataset. To do so, the data should be in a regularized time series as is required by the package that we use for temporal feature computation. However, the cleaned dataset possesses sparse data in the time series as the time when the patients took the test is irregular. Thus, we used resampling and interpolation techniques to regularise the frequency of the tests in the data.

Changing the frequency of time series observations is referred to as resampling, and there are two types of it: upsampling and downsampling [31]. Upsampling is used when the data frequency has to be increased, such as from minutes to seconds, and downsampling is used when the data frequency needs to be decreased, such as from days to months. Our data have some tests are taken daily and some monthly. Reducing the frequency from daily to monthly may eliminate many entries, so we decided to increase the frequency by converting the data to a daily basis. The resampled data has an entry for each day between the first and the last date a particular test was taken. After resampling, we could see that the resampled rows have a null value in the RESULT column, which are then filled using interpolation.

Interpolation is a technique for generating new data points from a set of data

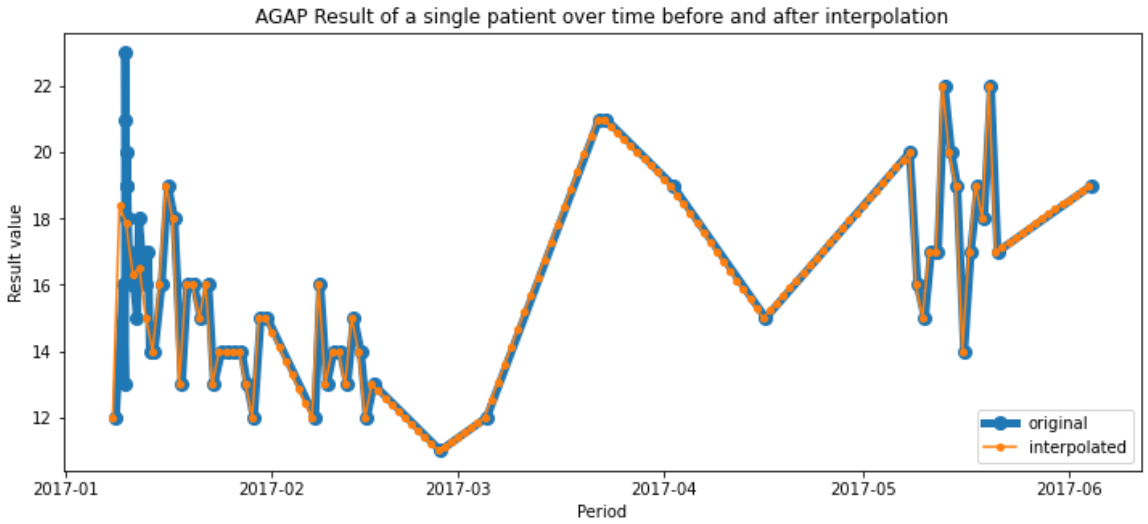

Figure 4.1: Trend comparison of a patient's AGAP result before and after resampling and interpolation

points [31]. There are various types of interpolation, namely Linear, Multivariate, Nearest Neighbor, Polynomial, and Spline. We have used linear interpolation to fill out NaNs in the data. Linear interpolation creates a continuous function of discrete data by creating a straight line between the available data. The data is interpolated so that if there are multiple occurrences of the same test on the same day, they are combined using the median of the results. If there is a big gap between the tests, a straight line is drawn, and points on the line are considered the result. In this way, we tried to maintain the original trend of the results after processing. We used resample and interpolate function from the Pandas.series library. Figure 4.1 shows the frequency of AGAP tests taken by a patient before and after resampling and interpolation. Care must be taken while regularizing the data as it may add noise while computing the features like variance and mean absolute change. We may use the original sparse data to compute the features if the calculated feature does not assume equal spacing in time. In our case, we equipped 'linear_trend' function to compute the slope from the result, which assumes the data to be uniformly sampled. So, we used interpolated data to compute features with such assumptions, whereas we used the original data to compute the other features like mean, mean absolute change, peak to peak distance and variance. Thus, this preprocessed data is then used to calculate temporal features, which are then used in the subsequent processes.

4.1.2 Computing Temporal Characteristics

The most crucial step in the data preparation is computing temporal characteristics as they are used as the new features of the dataset in the subsequent processes. The primary purpose of computing temporal characteristics is to extract the pattern and trend in the result column over time in numerical format. Also, from the perspective of pattern recognition, it is much more efficient and effective to characterize the time series concerning the distribution of data points, correlation properties, stationarity, entropy, and nonlinear time series analysis [48]. For this purpose, we have equipped a python based machine learning library named 'tsfresh'(Time Series FeatuRe Extraction on the basis of Scalable Hypothesis tests) [33]. This package automatically calculates 794 meaningful time series characteristics from a typical time-series pandas data frame. Apart from these predefined features, the package also provided the facility to include a custom feature. We have extracted five different features for each of the unique tests taken by the patients, among which four are predefined, and one is a custom user-defined characteristic. The features calculated in this study are described below:

1. **Slope:** This feature is computed using the 'linear_trend_timewise' module from the library. It calculates a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one [15]. This feature uses the time series index to fit the model, a DateTime datatype. Hence, we have restructured our base data frame and made BRFG_DTM the index. It returns five different attributes, among which we have picked 'slope' for further analysis;

2. **Peak to Peak distance:** This is a custom feature computed using the 'simple' feature property of the custom feature generation facility provided by the package. It is the absolute value of the difference between the maximum value (max) and minimum value (min) in the time series (i.e. the absolute difference between max peak and min peak);

3. **Mean:** It is the mean value of each observed test result;

4. **Variance:** It is the variance observed on individual test results;

5. **Mean Absolute Change:** It provides the mean over the absolute differences between subsequent time points that is the mean of daily differences;

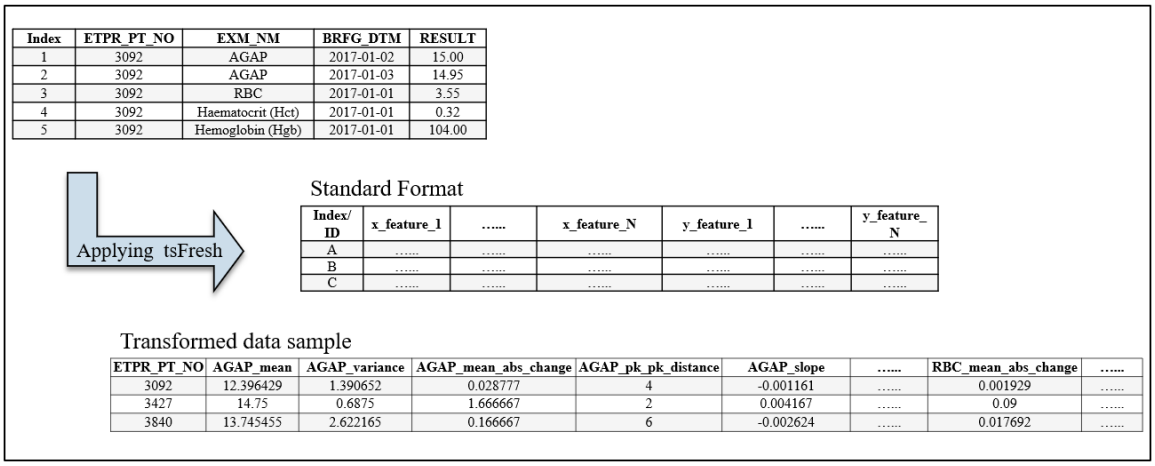

Figure 4.2: Data frame before and after feature calculations.

Similar to slope, the values mean, variance and mean absolute change are computed using predefined functions. The extracted features are represented by a design matrix where rows represent individual samples and columns represent the extracted features. In our case, the rows represent individual patients, and the columns represent the temporal features computed for each test taken by the patients, i.e., if we are to consider the top 10 tests, we would end up having 10*5 columns for each patient. Using this package, we computed the temporal features for all the three subsets, i.e., top 10 tests subset, next 10 tests subset and top 20 tests subset. Figure. 4.2 presents the sample of the original and transformed dataset. The next step after data frame transformation is separating the data into cohorts.

4.1.3 Cohort Separation

The new dataset with temporal characteristics as features has few null entries in various columns for all three subsets because the considered patients do not take those specific tests. Retaining these records with missing values in the dataset during analysis can lead to incorrect results. Hence, we divided the dataset into two cohorts where one cohort owns patients with no null entries meaning retaining the patients who have undergone all of the selected tests. To understand this cohort separation better, let us consider a sample where we are analyzing AGAP, RBC, and WBC tests

in the dataset. The corresponding pre-processed dataset will contain the temporal features of individuals who have had all three tests at least once within the period under consideration. This cohort is the primary source for identifying the typical phenotypes of patients by applying clustering models, as these models perform better with complete cases. Hence, we refer to this cohort as a base set.

After removing the base set from the data, we can see that each patient misses at least one of the test outcomes, leaving null entries for those who have not performed tests. Another cohort is separated from this missing data called the incomplete set, which contains the records of patients who have taken at least one of the considered tests. Let us consider the same three tests under consideration and two patients, one who has taken just AGAP but not RBC and WBC and another who has not taken any of the considered tests. In this case, we include the patient with just AGAP results in this second cohort and filter the other patient as including him will leave us with just the empty row for the considered tests. Figure 4.3 represents the sample discussed. Thus, the data in both cohorts are not overlapping and are distinct from one another. Similar to the sample explained above, we divided the data in each subset into two non-overlapping cohorts.

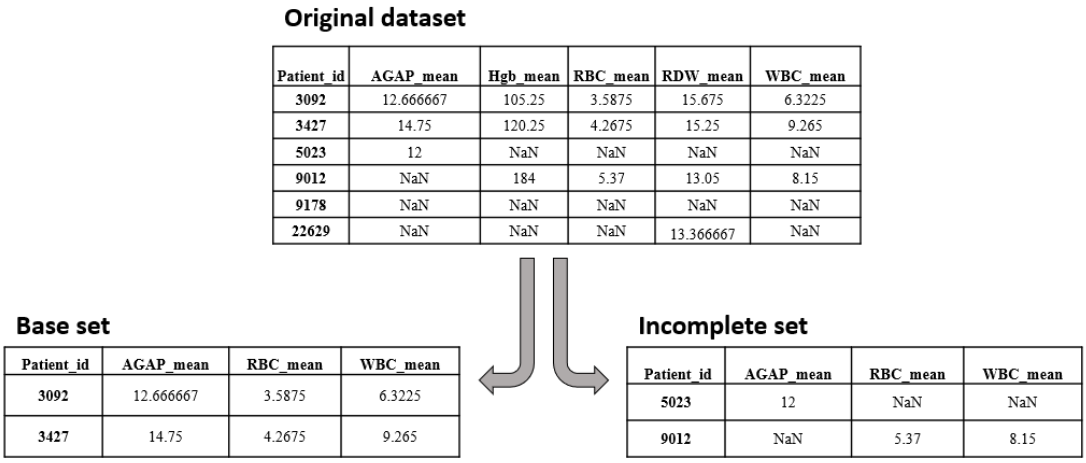

Figure 4.3: Cohort separation sample

After splitting the data, the data in each cohort is standardized, removing the mean and scaling to unit variance. Standardization/feature scaling is said to be a common requirement for many machine learning models as they might act up if

the individual features do not more or less look like standard normally distributed data [3]. To standardize our data, we have utilized the scikit-learn Standard Scaler library, which normalized our data so that the features are centred around 0 and have a standard deviation of 1. After standardization, the cohorts with no null entries, i.e., base set, are used for identifying the typical phenotypes of patients by applying clustering models. On the other hand, the data in the incomplete set is imputed and then sent for classification.

4.2 Data Clustering

After separating the data into cohorts, the cohort with no null values, i.e., base set, undergoes the clustering process to group the patients based on temporal characteristics of the results from the considered tests. We evaluated the performance of individual clustering algorithms and cluster ensembles and used the best to group our data. For this purpose, we have equipped the DiceR(diverse cluster ensemble in R) package [32]. This package guides the user through generating diverse clusterings, ensemble formation, and algorithm selection to attain the final consensus solution.

DiceR performs clustering using the subsampling technique, where the clustering algorithms are applied iteratively to a random portion of the data. It provides predefined functions to access around 15 clustering algorithms and five different consensus functions. The consensus functions available in this framework are Cluster-based Similarity Partitioning Algorithm(CSPA), K-modes, Latent Class Analysis(LCA), Link-based Cluster Ensemble(LCE), and majority voting.

After carefully evaluating various basic clustering algorithms, we selected k-means, k-medoids(partitioning around medoids), affinity propagation, and the Gaussian mixture model for our cluster analysis. The evaluation was done using various internal evaluation indices to see how separated the clusters were. Additionally, we have also checked the stability of the groups by running the same clustering models multiple times on the same data to see if the clusters assigned are consistent. We have evaluated other algorithms, including DBSCAN, spectral clustering, fuzzy clustering, and agglomerative clustering. We dropped DBSCAN as it marked most of the data points as outliers. The other not selected algorithms had poor internal validation metrics compared to the algorithms chosen for our data. We have equipped all the

five consensus functions in our study.

Since poor-performing algorithms can degrade the performance of a cluster ensemble [32], the package includes a method for including just the top N performing algorithms in the ensemble. It also provides visual and analytical evaluation strategies that assist the user in assessing their final result. The package includes numerous internal and external validity metrics for determining the quality of the clusters. Since our data is unsupervised, we have evaluated the cluster outputs using just the internal evaluation metrics. The package provides 16 internal evaluation metrics imported from clValid, clusterCrit, and LinkClue packages. Our study used silhouette score, Dunn index, Compactness, Connectivity, and Calsinki Harabarz from the package. Additionally, it assists the user in finding the optimal number of clusters using the proportion of ambiguous clustering (PAC) metric. It provides graphical displays of cumulative distribution function (CDF) graphs, the relative change in area under CDF curves, heatmaps, and cluster assignment tracking plots for further analysis of the clusters.

This framework incorporates the whole clustering process into a single function called 'dice()' that wraps the cluster analysis with the evaluation process. Before processing the data using this function, the user must select the clustering algorithms, distance metrics, consensus functions, and cluster sizes they desire to evaluate. This function takes the data frame with rows as samples and columns as features as input and processes it as mentioned in the Figure4.4.

A detailed explanation of the flowchart is given below:

1. Every base algorithm selected by the user is iteratively applied to several subsets of data, each consisting of 80% of the original observations. The user must specify the number of iterations that need to be carried out. The result is an array of clustering assignments computed across cluster sizes, algorithms, and subsamples of the data.

2. As a result of subsampling, not every sample is included in each clustering process, leaving some records without any assignments, which are then completed using the k-nearest neighbour imputation predefined in the function.

3. The completed array with multiple cluster assignments is then evaluated using

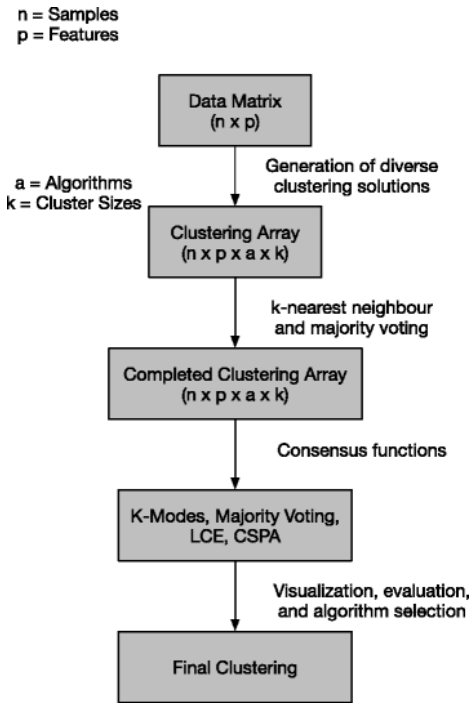

Figure 4.4: Flowchart of the process carried out by dice function[32]

internal evaluation metrics, and the results of top N algorithms are selected for the ensemble process.

4. The selected algorithm's cluster assignments are then combined using the consensus functions.

5. The cluster assignments made by the consensus functions are then evaluated using the internal evaluation metrics.

6. After evaluation, the metrics are returned in output and a rank of the consensus functions based on their performance.

From the output of the dice function, we can compare the performance of classical clustering algorithms with the ensemble functions and infer the best algorithm for our data. After identifying the best algorithm for our data, we have applied the algorithm to our data to find the patient phenotypes. The parameter description of the dice function is provided in the table 4.1. The parameter setting of this function is presented in section 5.1.

Parameter	Description
data	dataframe with rows as samples and columns as variables
nk	number of clusters ; can be a single integer or a range of integers
reps	number of iterations i.e. number of times random subsamples should be picked by the function
algorithms	single or a vector of clustering algorithms for performing consensus clustering
k.method	a method to choose k when no reference class is given. The default method used is PAC to choose the best k(s).
nmf.method	specify NMF-based algorithms to run. Default are the "brunet" and "lee" algorithms.
hc.method	agglomeration method for hierarchical clustering. By default, the "average" approach is utilised.
distance	a vector of distance functions like Euclidean, Manhattan, spearman, minkowski. Defaults to "euclidean".
cons.funs	consensus functions to use.
sim.mat	similarity matrix to be used . default are are "cts", "srs", "asrs"
seed	random seed for knn imputation reproducibility
trim	logical; if TRUE, algorithms with low internal indices will be pruned
reweigh	logical; if TRUE, after removing bad performers, each algorithm is reweighted based on its internal indices.
n	a number indicating the top n methods to maintain after removing the weak performers via Rank Aggregation
evaluate	logical; if TRUE internal validity indices are returned
plot	logical; if TRUE, heatmap of ranked algorithms vs. internal validity indices is plotted.

Table 4.1: Parameter description of dice function[11]

4.2.1 Performance Evaluation

Since the data we use for the analysis are not provided with actual labels, we have chosen five internal evaluation metrics to evaluate our final clusters. They depend only on the information in the data. The metrics assessed are Calinski Harabarz, Dunn index, Silhouette score, Compactness, and Connectivity. The metrics selected best evaluate the clusters' Compactness, Connectivity, and separation. Also, we have used the proportion of ambiguous clustering(PAC), elbow method, BIC scores and silhouette method to determine the best number of clusters that can be formed from the data. Finally, after evaluating and identifying the best model for the data, we

visualized the clusters obtained by reducing the data to two dimensions.

Finding optimal K value

Because the number of clusters discovered in the dataset is unknown, it is necessary to determine the ideal number of clusters that best organize the unsupervised data. For this purpose, we have used the elbow method, silhouette method, and BIC(Bayesian Information Criterion) score along with the PAC score provided by diceR on the considered clustering algorithms. We considered the k values ranging from 2 to 8 for our evaluation. We ran the elbow and silhouette methods using k-means and PAM to determine optimal k values and used BIC to find the optimal k-value for the GMM model. Table 4.2 shows the various methods used for each clustering model to find the optimal number of clusters. We used python implementation of these functions from the sklearn library for evaluation. The silhouette method considers the silhouette score calculated for each k value by the clustering models. It considers the k-value with a higher silhouette score optimal as it better groups the data into compact and well-separated clusters.

Model	**Method**
k-means	silhouette method , elbow method, PAC
PAM or k-medoids	silhouette method , elbow method, PAC
GMM	BIC, PAC
AP	PAC

Table 4.2: Methods used to identify optimal k-value for each models

Cluster visualization

After identifying the best model and clustering the data, we visualized the clusters on a two-dimensional space using t-distributed stochastic neighbour embedding(t-SNE). It is a nonlinear dimensionality reduction approach for displaying high-dimensional data by assigning a position to each data point in a two- or three-dimensional map. [106]. This technique models each high-dimensional object to a lower-dimensional point so that nearby points model similar objects, and dissimilar objects are modelled by distant points with high probability. This process would help us visualize how compact and separate each cluster are from one another.

4.3 Data Imputation

The data in the incomplete set have null entries across the dataset as it contains data of the patients who have not taken at least one of the considered tests. As a result, while computing the temporal characteristics, the tests that the patient did not take will have a void entry. Since the dataset with the null entries cannot produce efficient results, we processed incomplete set with imputation techniques to fill the null values. Imputation is a strategy for retaining most of the data/information in a dataset by replacing missing data with some substitute value. There are various imputation techniques available in the literature, among which we have chosen univariate, multivariate, or iterative and kNN imputation techniques. We evaluated the performance of these selected techniques on the base set data before applying it to the incomplete set. We used the cross-validation technique(repeated kfold) to utilize the available data efficiently. The k-fold cross-validation procedure divides a limited dataset into k non-overlapping folds. Each fold in the k folds is allowed to be used as a held back test set, while all other folds collectively are used as a training dataset. A total of k models are fit and evaluated on the k hold-out test sets, and the mean performance is reported. In repeated kfold, the kfold process is repeated n times, and the mean of the runs is reported as performance. The entire evaluation process, which is carried out using python functions provided by the scikit learn's package, is presented in Figure 4.5.

The evaluation process is carried out using python's sklearn package and is as follows:

1. The base set data with no null values is standardized and is split into train and test sets using repeated fold.

2. The test set is masked where a random percent of entries are changed to null entries.

3. The train set is used to fit the model where the model learns the pattern in the data.

4. The masked test set is then imputed using the trained model.

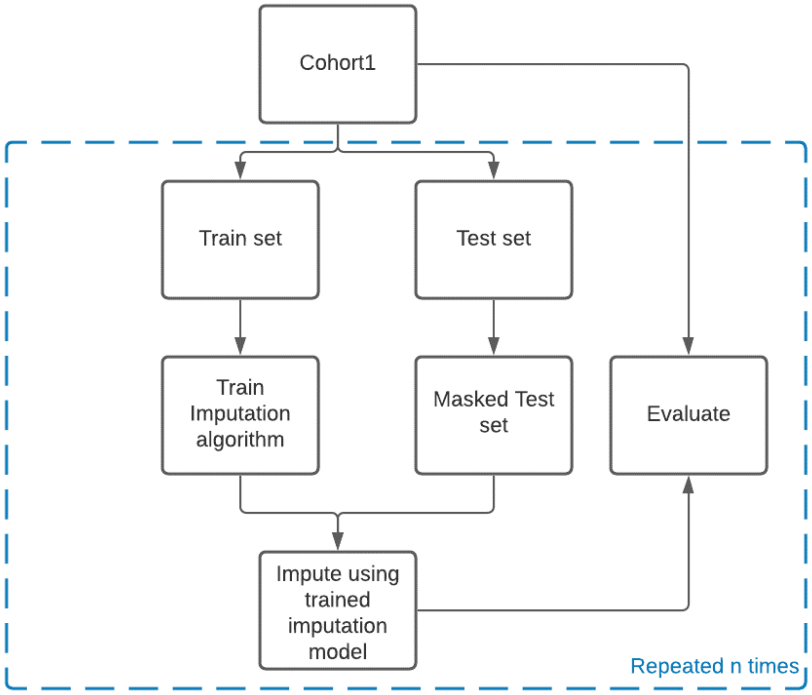

Figure 4.5: Evaluation process of Imputation models

5. The masked imputed test set is then evaluated against the original test set, i.e., before masking, and the error rate is computed using Root Mean Square Error(RMSE).

6. Steps (2) to (5) are repeated five times as the number of repeats in repeated kfold was set to 5.

The process from steps (1) to (5) is repeated thrice as we evaluated the imputation models against 10%,20%, and 30% masked data and carried out for all the considered imputation techniques. The final error rate is computed by averaging the RMSE of all three runs. The algorithm with the least error rate is trained on the base set and applied to the incomplete set data. For computing the RMSE scores, we have equipped the mean_square_error function from sklearn's metrics library.

4.4 Data Classification

The imputed incomplete set, which is not assigned to any clusters defined by the clustering models, is now set to the appropriate cluster using the classification models. The classification models considered are support vector machine(SVM), Gaussian Naive Bayes, KNeighbous classifier, decision trees, and random forest. The main advantage of using imputation and classification is to understand the progression of the disease among the patients by guessing the result of the tests and assigning them to appropriate peer groups. Similar to imputation techniques, the classification models are evaluated and trained on a base set and applied to an incomplete set. The classification process is performed using the scikit learn package implementations. The data classification process in this study follows the below flow:

1. Base set is split into train and test sets using repeated stratified Kfold.
2. The training set is oversampled using SMOTE as it is imbalanced.
3. The oversampled training set is grouped using considered classification models.
4. The assignment by the classification model is evaluated against the true cluster assignment(here, assumptions made by the clustering model) using several evaluation metrics like precision, recall, f-score, and ROC-AUC curves.
5. The model with better evaluation metrics is selected and then applied to the incomplete set data.

We split the data into train and test sets using the repeated stratified kfold strategy, which separates the dataset in that each fold has the same proportion of observations with the given label. After retaining the same proportion of observations, we have applied SMOTE(Synthetic Minority Oversampling Technique) to oversample the minority classes per training set of each repeat. These techniques are beneficial as we deal with imbalanced class distributions. The classification models, evaluation metrics, and the kfold methods were imported from the predefined python's sklearn packages, whereas SMOTE is imported from the imblearn python package. We have equipped the ROCAUC function that is provided by the Yellowbrick library in our study for evaluation[12].

Chapter 5

Experimental Results

This chapter presents the results of all of the procedures depicted in Figure 3.6 on these subsets and explanations for different decisions made along the way. We examined the top 20 commonly discovered tests to categorize the patients, as mentioned in Section 4.1. After preprocessing the entire dataset, we ran three different experiments with subsets of data partitioned by these specified tests. The first subset includes patients who have taken at least one of the first ten tests among the top 20, whereas the second subset would possess the records of the patients who have taken the last 10 of the top 20 commonly occurring tests, as mentioned in table 3.2. The third subset retains the data of patients who took at least one of the 20 tests considered. The total number of patients present in these subsets is 2252, 2314, and 2381. The tables 5.1,5.2 and 5.3 presents the summary of each test in each of the three subsets divided.

Tests	**Top 10 tests subset**				
	Mean	**SD**	**Min**	**Median**	**Max**
AGAP	12.87	3.23	0	12	42
Carbon Dioxide Level (CO2 Level)	22.51	4.06	5	23	47
Chloride level, Serum(Cl Level)	104.4	6.05	54	104	149
Creatinine Level, Serum	126.21	155.8	21	73	1733
Haematocrit(Hct)	0.333	0.072	0.11	0.33	0.66
Hemoglobin(Hgb)	108.06	23.89	36	105	213
Potassium Level, Serum(K Level)	4.15	0.59	1.8	4.1	8.1
RBC	3.73	0.89	1.26	3.65	6.98
Sodium Level, Serum(Na Level)	135.68	4.9	114	136	180
Urea Nitrogen, Blood(BUN)	7.49	6.13	0.5	5.5	44.6

Table 5.1: Statistics of the result of the tests present in the top 10 tests subset

We subjected each subset to the clustering procedure from which we selected the subset with better phenotypes to undergo further processes. Section 5.1 presents the results of the cluster analysis which is followed by the results of imputation and classification in Sections 5.2 and 5.3 respectively.

Tests	Next 10 tests subset				
	Mean	**SD**	**Min**	**Median**	**Max**
Alanine Transferase(SGPT)	51.1	127.4	5	24	3882
Aspartate Aminotransferase(SGOT)	48.26	134.76	5	23	4031
MCHC	323.71	13.84	252	324	366
MCV	91	7.9	58	90.9	124
MPV	8.75	1.37	5.7	8.7	16.6
Magnesium(Mg) Level, Serum	0.77	0.12	0.28	0.76	1.53
Platelet count (Plt Count)	205.55	132.71	5	189	1812
RDW	16.5	2.8	10.9	15.9	33.1
White Blood Cell Count(WBC Count)	7.3	7.9	0.05	5.98	200.52
eGFR	80.85	40.2	3	81	220

Table 5.2: Statistics of the result of the tests present in the next 10 tests subset

Tests	Top 20 tests subset				
	Mean	**SD**	**Min**	**Median**	**Max**
AGAP	12.76	3.17	0	12	42
Alanine Transferase(SGPT)	52.6	131.45	5	24	3882
Aspartate Aminotransferase(SGOT)	49.78	138.98	5	23	4031
Carbon Dioxide Level (CO2 LeveL)	22.53	3.87	6	23	39
Chloride level, Serum(Cl Level)	104.25	5.83	78	104	149
Creatinine Level, Serum	130.08	154.58	28	78	1733
Haematocrit(Hct)	0.332	0.07	0.11	0.32	0.66
Hemoglobin(Hgb)	107.44	23.89	36	104	213
MCHC	323.7	13.93	252	324	366
MCV	91.13	7.96	58	91	124.2
MPV	8.78	1.38	5.7	8.7	16.6
Magnesium(Mg) Level, Serum	0.77	0.12	0.28	0.76	1.48
Platelet count (Plt Count)	200.95	131.2	5	183	1812
Potassium Level, Serum(K Level)	4.16	0.59	1.8	4.2	8.1
RBC	3.67	0.87	1.26	3.58	6.98
RDW	16.5	2.8	10.9	15.9	33.1
Sodium Level, Serum(Na Level)	135.39	5	114	136	180
Urea Nitrogen, Blood(BUN)	7.82	6.17	0.5	5.9	44.6
White Blood Cell Count(WBC Count)	7.31	8.08	0.05	5.96	200.52
eGFR	80.3	40.74	3	80	220

Table 5.3: Statistics of the result of the tests present in the top 20 tests subset

5.1 Clustering Results

Once the data preprocessing is carried out on the whole dataset, we divided the dataset into subsets and computed temporal characteristics for each subgroup. Based on the selection of tests to be considered and the availability of their results, the cohorts are separated. The clustering process is carried out solely on base set data: the non-imputed data with no null entries. It is the temporal features of the patients who have taken all the considered tests of each subset. The number of unique patients present in the base set of the three subsets is 1466, 1110 and 1001, respectively. We carried out the entire clustering process using the dice function from the R package 'DiceR'. This function, by default, splits the data 80:20 ratio and applies clustering algorithms on 80% of the data for the number of repetitions mentioned by the user. We considered the 20 repetitions in the experiment where the considered clustering algorithms cluster 20 different fragments of the same data. We have used this subsampling method to generate our cluster ensembles. In contrast, we obtained the individual clustering results by applying the algorithm to the data in a conventional way using the function' consensus_cluster'. This function outputs cluster assignments across subsamples and algorithms for different clusters where the user can predefine the subsampling ratio. To make the clustering conventionally, we have chosen the sampling ratio as 100:0. Throughout the process, we have considered the potential cluster numbers ranging from 2 to 8. We have compared four different ensemble cluster assignments, each formed upon the top 1,2,3 and 4 clustering algorithms. The parameter setting of the dice function is presented in the table 5.4. The remaining parameters were set to their default values. Following are the cluster analysis results for the various subsets:

5.1.1 Result of top 10 tests subset

This subset contained the top 10 most commonly taken tests by the patients. The tests considered are AGAP, Carbon dioxide level, serum chloride level, serum creatinine level, hematocrit, hemoglobin, serum potassium level, RBC, serum sodium level and blood urea nitrogen. After preprocessing and separating the cohorts, we assessed the base set data for identifying the optimal k-value using the elbow method,

Parameter	Values
data	subset1 or subset2 or subset3
nk	2 to 8
reps	20
algorithms	km, pam, ap, gmm
nmf.method	brunett, lee
distance	euclidean, manhattan.
cons.funs	majority, CSPA, LCE, kmodes, LCA
sim.mat	cts, srs, asrs
seed	1
trim	TRUE
reweigh	TRUE
n	1,2,3,4
evaluate	TRUE
plot	TRUE

Table 5.4: Parameter setup of dice function

BIC scores. Figures 5.1and 5.2 show the elbow curve and BIC scores indicating the number of clusters to use for the dataset.

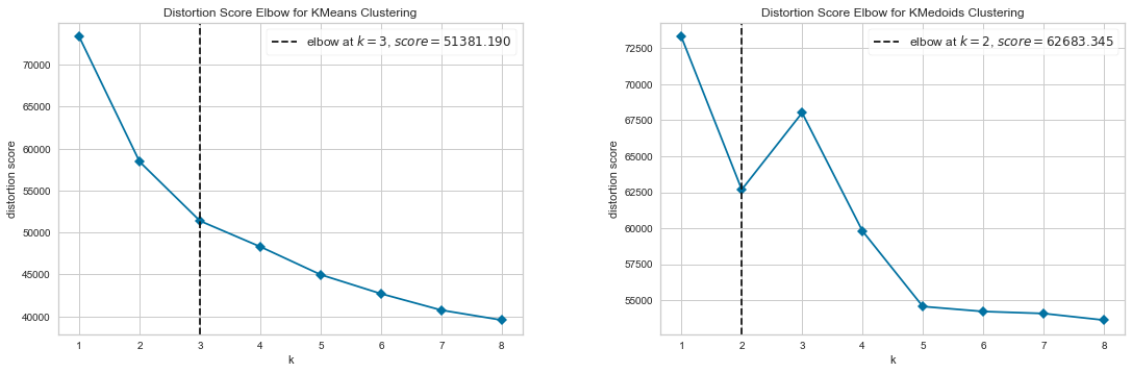

Figure 5.1: Elbow method indicating optimal k value using kmeans and kmedoids on the top 10 tests subset

From the plots, we could see that the kmeans and GMM model have the optimal value at k=3, whereas the kmedoids have the optimal k value at k=2. To further evaluate the optimal number of clusters, we computed PAC scores for the range of k values using the four considered clustering algorithms. Table 5.5 shows that k=2 has the lowest average PAC score, followed by k=3, indicating that they are the better number of clusters for the dataset. The next step was to compare the silhouette score of all the clustering methods across various k values and find the optimal clusters.

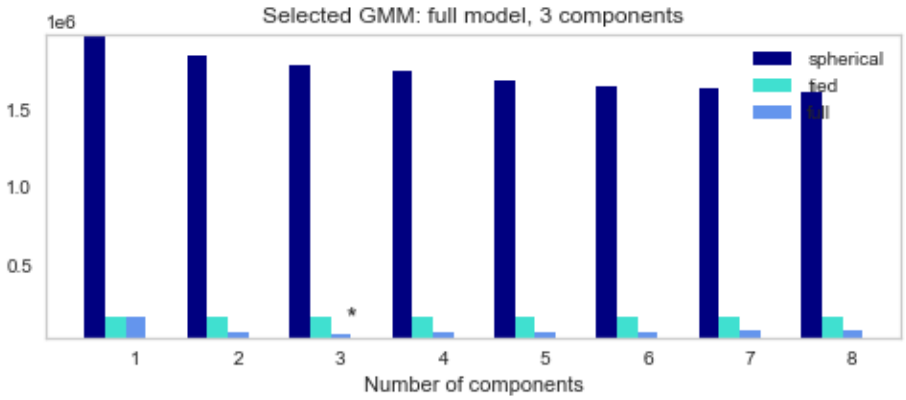

Figure 5.2: Bayesian Information Criterion(BIC) score of Gaussian Mixture Model(GMM) indicating optimal k value for top 10 tests subset

Table 5.6 shows the internal evaluation metrics like calinski harabasz, Dunn index, silhouette score, compactness and connectivity of the clustering algorithms for various k values. From the table, we could infer that k=2 has the highest silhouette score indicating that dividing the data into two clusters forms a better clustering solution.

K	**PAM Euclidean**	**PAM Manhattan**	**AP**	**GMM**	**KM**	**Average PAC**
2	0.06	0.20	0.23	0.08	0.02	0.12
3	0.14	0.19	0.32	0.11	0.05	0.16
4	0.16	0.27	0.29	0.11	0.09	0.18
5	0.17	0.29	0.30	0.16	0.18	0.22
6	0.19	0.28	0.26	0.18	0.20	0.22
7	0.15	0.21	0.24	0.20	0.19	0.20
8	0.15	0.15	0.21	0.23	0.24	0.20

Table 5.5: Proportion of Ambiguous Clustering(PAC) scores for each k on top 10 tests subset

Thus, from all the considered measures for finding an optimal number of clusters, we concluded the best groups for the top 10 tests subset to be k=2 and k=3. Using these k values, we have applied the consensus functions on clustering results and compared their performance to the conventional clustering performance. We have generated cluster ensembles by combining the results of n different algorithms executed for 20 repetitions on the same data using consensus functions. Table 5.7 show

K	best alg	CH	D	S	CP	C
2	KM	372.46	0.03	0.31	6.76	170.41
3	KM	228.27	0.02	0.25	6.47	538.19
4	PAM_E	152.27	0.02	0.10	6.66	553.36
5	KM	118.98	0.01	0.08	6.40	543.52
6	AP	97.00	0.01	0.02	6.32	1081.67
7	PAM_E	82.45	0.01	-0.11	6.24	1439.25
8	PAM_E	67.13	0.01	-0.06	6.71	1675.66

Table 5.6: Best algorithm for each k values on top 10 tests subset

the internal evaluation metrics of the two best consensus functions for the considered k values and n values. Here, n is the number of clustering algorithms combined to form the ensembled result, and it ranges from 1 to 4. For instance, if the n value is 1, the consensus function considers the results of only the top-performing clustering algorithm and combines its results from 20 repetitions. When the value of n is 4, the consensus function combines the cluster assignments of all the four algorithms we considered and generates a single cluster assignment from it. Along with the n value, we have also mentioned the best performing algorithms considered by the consensus functions for generating ensembled results.

n	K	Best consensus	CH	D	S	CP	C
1[km]	2	majority	429.71	0.04	0.32	6.77	158.58
		LCE	416.15	0.04	0.32	6.77	161.24
	3	majority	307.91	0.04	0.27	6.60	192.76
		LCA	274.77	0.04	0.27	6.63	207.39
2[km,ap]	2	LCE	372.15	0.04	0.28	6.76	170.49
		majority	359.76	0.03	0.28	6.76	204.43
	3	majority	223.84	0.03	0.20	6.70	226.69
		kmodes	229.91	0.02	0.21	6.54	524.69
3[km,ap,pam]	2	majority	342.54	0.03	0.26	6.76	247.17
		kmodes	342.39	0.03	0.25	6.76	248.70
	3	majority	213.26	0.03	0.17	6.66	526.75
		kmodes	229.55	0.02	0.15	6.52	748.92
4[all]	2	majority	354.48	0.03	0.24	6.75	232.08
		kmodes	354.48	0.03	0.24	6.75	232.08
	3	majority	229.87	0.02	0.11	6.45	525.55
		LCE	219.10	0.02	0.10	6.45	521.95

Table 5.7: Performance metrics of various consensus functions for each n and k values considered on the top 10 tests subset

From the tables 5.6 and 5.7, we concluded the following:

- The quality of the ensemble decreases with the increase in the number of best-performing algorithms considered(i.e., n), indicating that combining the results of different algorithms reduced the quality of the ensembles.
- The consensus function 'majority' is the best performing function as it tops almost all the experiments.
- We inferred the order of the algorithms based on their performance using the order in which the function considered the algorithms. The rank of the algorithm is as follows: kmeans, AP, PAM/kmedoids and GMM. This indicates that kmeans is the best individual clustering algorithm that effectively separates the data into clusters.
- By comparing the performance of individual and ensemble clustering, we could say that the ensembles have the upper hand in forming clusters within the data.
- Though the cluster value k = 2 produces better internal evaluation indices, the centroid models such k-means are convex and isotropic because of inertia, always converge and remain optimal at lower cluster values. Thus, we have considered k=3 as the optimal number of clusters.

Thus, we have considered the ensemble results generated by the majority consensus function at k=3 and n=1 as the optimal solution. To visualize the clusters formed by this method, we have then reduced the dimensions of the top 10 tests subset using the t- distributed stochastic neighbour embedding(tSNE) technique from python. Figure 5.3 presents the clusters using the two-dimensional plot along with the proportion of patients available per cluster.

Using the groups formed by the optimal cluster model, we plotted a heat map figure 5.4 that represents the characteristics of the patients in each cluster. Here, the features refer to the pathology test results that distinguish the groups. This heatmap presents the average of the results of each considered test. From the plot, we can see that the tests serum creatinine level, hemoglobin and blood urea nitrogen are the key attributes that distinctly identify the clusters. The results of the other seven tests remain the same across the groups.

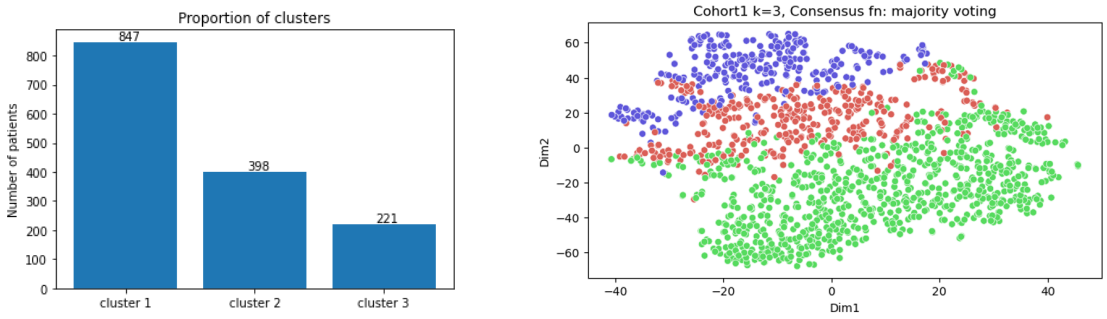

Figure 5.3: Proportion and Visualization of clusters formed on top 10 tests subset

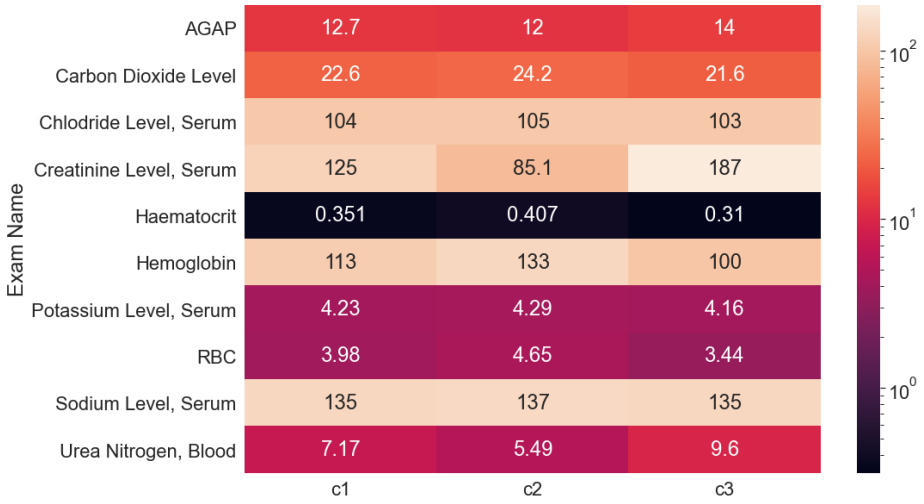

Figure 5.4: Comparison of feature values across the 3 clusters formed on top 10 tests subset

5.1.2 Results of next 10 tests subset

This subset contains the next ten commonly occurring tests among the patients. The tests included are Alanine Transferase, Aspartate Aminotransferase, MCHC, MCV, MPV, serum magnesium level, platelet count, RDW, white blood cell count and eGFR. The processing of the subset with the next 10 tests is similar to the process carried out on the top 10 tests subset. Figures 5.5 and 5.6 shows the optimal number of clusters possible on the data using kmeans, kmedoids and GMM models. We can observe that the elbow method indicates the optimal k value as 3 and 5 while applying it using kmeans and kmedoids, respectively. In contrast, the BIC score is the lowest at k=3, indicating a better k value while applying the GMM model.

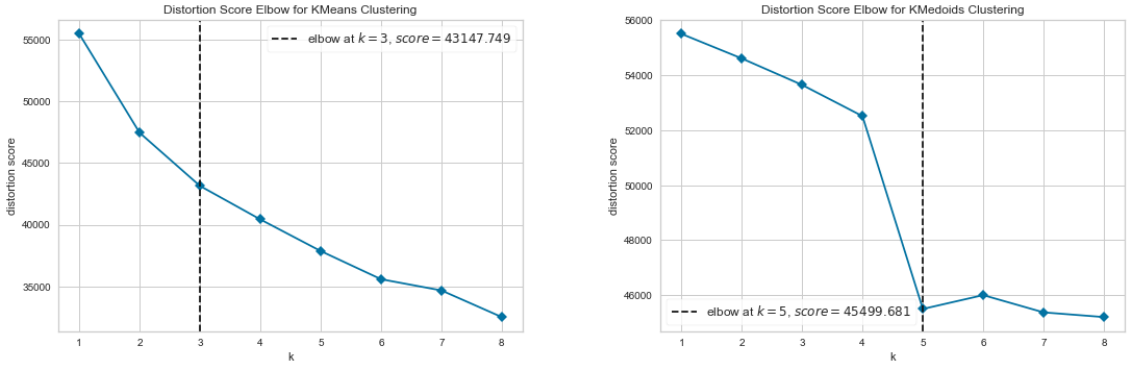

Figure 5.5: Elbow method indicating optimal k value using kmeans and kmedoids on the next 10 tests subset

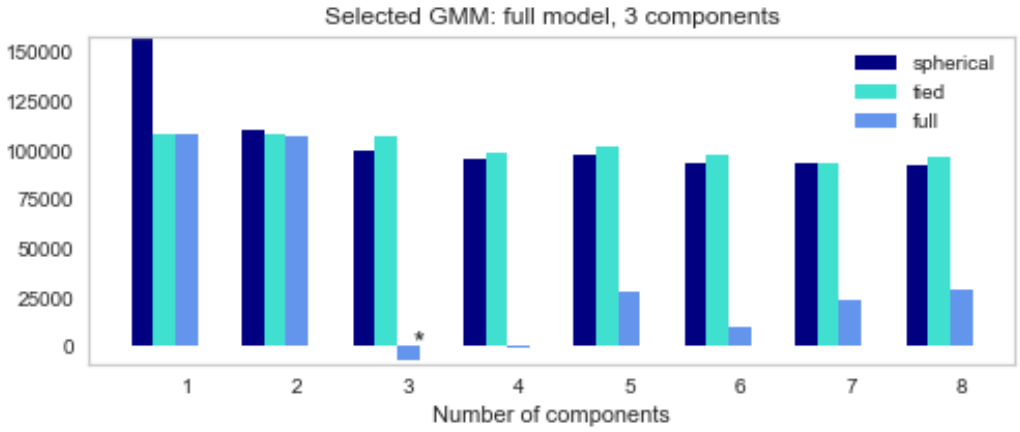

Figure 5.6: Bayesian Information Criterion(BIC) score of Gaussian Mixture Model(GMM) indicating optimal k value for next 10 tests

K	**PAM Euclidean**	**PAM Manhattan**	**AP**	**GMM**	**KM**	**Average PAC**
2	0.09	0.23	0.14	0.07	0.07	0.12
3	0.22	0.24	0.22	0.18	0.05	0.18
4	0.33	0.27	0.25	0.20	0.14	0.24
5	0.28	0.16	0.35	0.24	0.19	0.24
6	0.29	0.26	0.29	0.17	0.24	0.25
7	0.28	0.26	0.29	0.23	0.18	0.25
8	0.25	0.27	0.27	0.18	0.31	0.26

Table 5.8: Proportion of Ambiguous Clustering(PAC) scores for each cosidered k values on next 10 tests subset

Since we received different k values for other algorithms, we investigated it further by evaluating all the algorithms' PAC scores and silhouette scores for all the considered k values. Table 5.8 shows the PAC scores from which we could infer that the PAC scores increase with the increase in the k value. We could see that the smallest PAC for the PAM with euclidean distance, PAM with Manhattan distance, AP, GMM, and KM are at k=2,k=5,k=2,k=2, and k=3, respectively.

K	**Best alg**	**CH**	**D**	**S**	**CP**	**C**
2	KM	185.80	0.03	0.30	6.76	170.41
3	KM	106.04	0.03	0.27	6.77	390.23
4	KM	97.94	0.02	0.12	6.68	452.43
5	KM	66.78	0.02	-0.07	6.62	428.66
6	GMM	47.42	0.02	-0.10	6.61	1436.26
7	GMM	39.39	0.01	-0.11	6.60	1646.42
8	KM	41.16	0.02	-0.09	6.39	956.14

Table 5.9: Best algorithm for each k values using next 10 tests data

From the table 5.9 that shows the metrics of the best clustering algorithm for each considered k value, we could say that the silhouette score is the highest when the number of clusters is small and vice versa. We could also see that the kmeans algorithm performed better for most k values. Thus, from all the measures of identifying the optimal number of clusters, we narrowed down our k values to be 2,3 and 5.

Following the evaluation for the k value, we applied the dice function to the data for generating cluster ensembles. Table 5.10 shows the internal metrics of the consensus functions that performed better for the k values that were narrowed down. Apart from evaluating the process for just the filtered k value, we have assessed the ensemble performance of all the k values that were initially considered.

n	**K**	**Best consensus**	**CH**	**D**	**S**	**CP**	**C**
1[km]	2	majority	186.09	0.04	0.31	6.94	204.02
		LCA	186.81	0.04	0.31	6.92	204.96
	3	majority	135.37	0.03	0.29	6.74	291.37

		LCA	135.07	0.03	0.27	6.63	332.30
	5	majority	120.85	0.04	-0.08	6.50	361.89
		LCE	102.71	0.02	-0.06	6.54	443.95
2[km,ap]	2	majority	181.75	0.03	0.31	6.91	226.95
		kmodes	173.99	0.03	0.30	6.90	241.95
	3	majority	100.02	0.03	0.20	6.83	453.29
		kmodes	105.62	0.02	0.18	6.78	534.62
	5	majority	77.44	0.03	-0.07	6.63	412.89
		kmodes	68.67	0.02	-0.04	6.62	812.00
3[km,ap,pam]	2	majority	188.90	0.03	0.23	6.80	259.87
		kmodes	187.51	0.04	0.22	6.94	292.05
	3	majority	102.66	0.02	0.11	6.77	519.40
		kmodes	106.79	0.02	0.11	6.74	519.30
	5	majority	69.20	0.02	-0.07	6.63	563.53
		LCE	70.98	0.02	-0.08	6.56	787.09
4[all]	2	majority	176.41	0.03	0.19	6.92	237.92
		kmodes	175.68	0.03	0.19	6.89	258.51
	3	majority	95.03	0.02	0.09	6.57	519.98
		LCA	93.43	0.02	0.06	6.56	665.86
	5	majority	70.16	0.02	-0.05	6.59	588.45
		LCE	71.40	0.02	-0.08	6.55	786.79

Table 5.10: Performance metrics of various consensus functions for each n and k values considered on the next 10 tests subset

Following are the inferences made from the clustering results:

- Similar to the top 10 tests subset, the quality of the clusters decreases as we include more algorithms for ensemble clustering. That is, the performance is better for the lower values of n.

- 'Majority' and 'kmodes' are the most successful consensus functions that produced better results.

- Based on the order of trimming of the poor algorithms made by the dice function, we could say that GMM was the least performing algorithm as the function did not pick it unless all the four algorithms were included. The rank of the algorithms based on the trimming made by the dice function is kmeans, AP, PAM and GMM, respectively.
- As seen in the previous experiment, ensemble clustering has higher internal metrics than conventional clustering.
- Among the filtered k values, we have decided to go forward with k = 3 as it has better metrics than k = 5 and slightly decreases its performance compared to k=2.

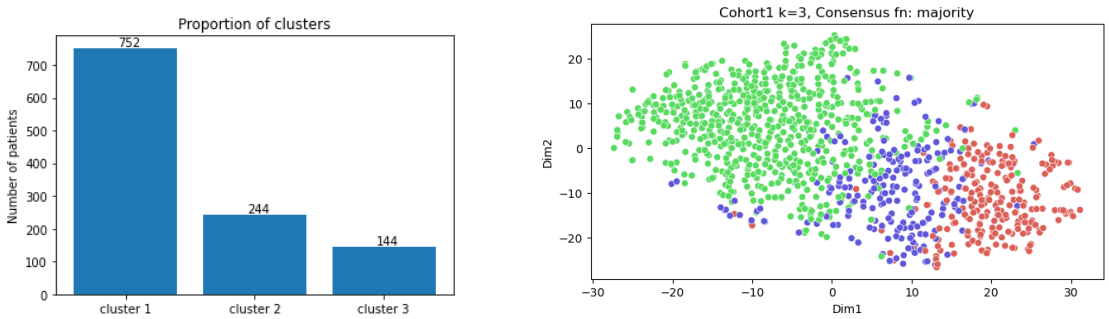

Figure 5.7: Proportion and Visualization of clusters on next 10 tests subset

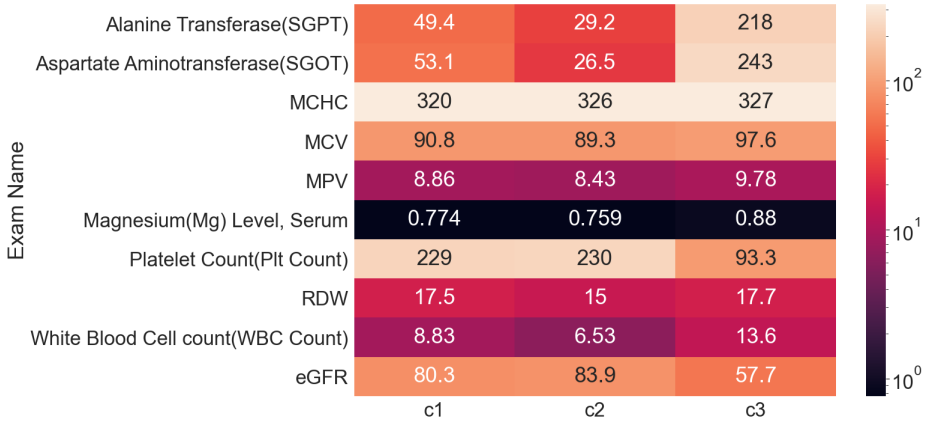

Figure 5.8: Comparison of feature values across the 3 clusters formed on next 10 tests subset

Thus we grouped the data with the next 10 tests into three clusters by applying the majority voting consensus function on the results of kmeans algorithms that were

iteratively applied 20 times. Figure 5.7 presents the proportion of the clusters formed as well as the visual representation of the clusters using dimensionally reduced next 10 tests data.

From the clusters formed, we identified the pathological tests that uniquely differentiate each cluster by averaging the results of each test. Figure 5.8 presents the average result values of each test under each cluster. The heatmap shows that the tests Alanine Transferase, Aspartate Aminotransferase and platelet count have significant changes across the clusters. In contrast, the white blood cell count and eGFR have slight differences across the clusters.

5.1.3 Results of top 20 tests subset

This subset is the combination of both the top 10 tests and the next 10 tests subset in terms of the pathology tests, i.e., the patients in the base set of this subset will have taken all the tests selected in the top 10 tests subset and next 10 tests subset, thus making this subset a refined version of other subsets with fewer patients than them. As mentioned earlier, since the same procedure is followed across various subsets, the first step is to identify the optimal number of clusters for the data. Figures 5.10 and 5.10 portrays the selection of optimal k value by elbow method and BIC scores computed using kmeans, kmedoids and GMM models. Table 5.11 presents the PAC scores of unique algorithms for various k values. The average PAC score is low for k=2, indicating a better number of clusters. We have obtained the optimal k values from all three methods as 2 and 3. We then applied each of the clustering methods to the data separately for each value of k and computed the internal metrics using its output which is presented in the table 5.12. From this table, we concluded that kmeans is the better performing algorithm for most of the k values and the quality of the clusters. Also, based on the silhouette score mentioned in the table, we could infer the better clustering at k=2. Thus, after summarising the results of all the methods that obtained optimal k values, we concluded the optimal number of clusters as 2 and 3.

With these observed k values, we have generated cluster ensembles using the dice function. This function combined the cluster results of various algorithms repeatedly

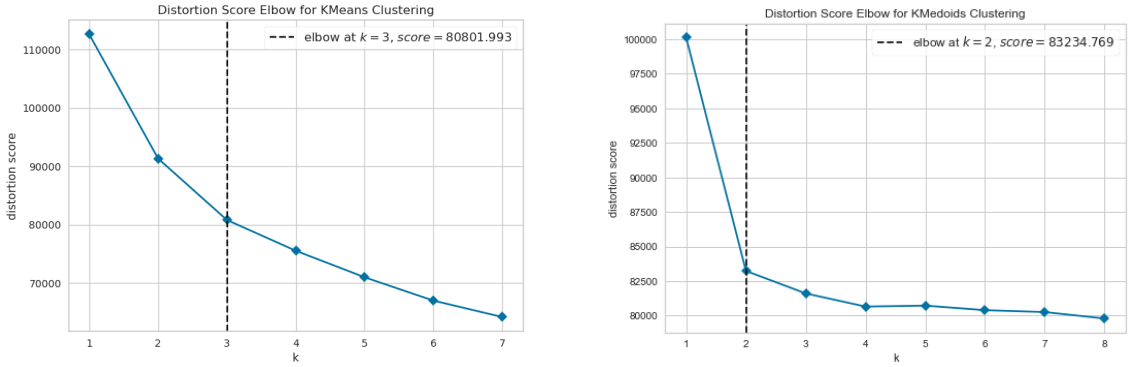

Figure 5.9: Elbow method indicating optimal k value using kmeans and kmedoids on top 20 tests subset

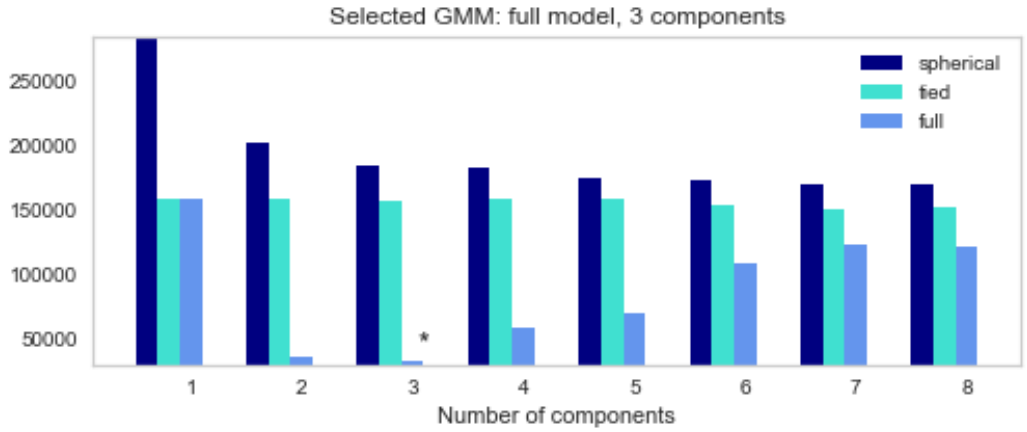

Figure 5.10: Bayesian Information Criterion(BIC) score of Gaussian Mixture Model(GMM) indicating optimal k value for top 20 tests subset

K	**PAM Euclidean**	**PAM Manhattan**	**AP**	**GMM**	**KM**	**Average PAC**
2	0.03	0.01	0.10	0.13	0.00	0.05
3	0.28	0.27	0.15	0.20	0.02	0.19
4	0.35	0.21	0.34	0.11	0.08	0.22
5	0.27	0.24	0.25	0.15	0.08	0.20
6	0.26	0.19	0.20	0.30	0.18	0.28
7	0.26	0.16	0.20	0.48	0.06	0.23
8	0.25	0.16	0.21	0.47	0.25	0.27

Table 5.11: Proportion of Ambiguous Clustering(PAC) scores on for each k values on the top 20 tests subset

K	Best alg	CH	D	S	CP	C
2	KM	185.80	0.03	0.36	9.84	139.79
3	KM	106.04	0.03	0.22	9.63	223.37
4	KM	93.94	0.02	0.12	9.76	452.43
5	KM	88.17	0.03	0.06	9.39	557.43
6	PAM_E	53.45	0.03	0.04	9.62	545.11
7	PAM_M	51.29	0.02	-0.08	9.60	977.20
8	KM	42.37	0.03	-0.03	9.47	775.04

Table 5.12: Best algorithm for each k values using top 20 tests subset

applied to various data fragments and combined it according to the value of n provided. Here, n is the number of the top algorithms to be considered for generating ensembles. We ran this experiment by varying the n value from 1 to 4 and compared the performance of various consensus functions. Table 5.13 displays two best performing consensus functions for each k and n values.

n	K	Best consensus	CH	D	S	CP	C
1[km]	2	majority	218.89	0.04	0.38	10.05	137.95
		kmodes	219.13	0.05	0.38	10.04	131.91
	3	majority	150.13	0.05	0.25	9.77	226.55
		kmodes	149.84	0.05	0.25	9.78	226.67
2[km,ap]	2	majority	182.71	0.04	0.34	6.98	197.76
		kmodes	180.79	0.03	0.33	6.90	223.67
	3	LCA	119.80	0.04	0.21	10.00	167.63
		majority	131.18	0.04	0.18	9.84	280.21
3[km,ap,pam]	2	majority	176.31	0.03	0.32	6.91	243.43
		kmodes	175.68	0.03	0.32	6.89	258.51
	3	majority	111.11	0.04	0.12	9.98	238.43
		LCE	121.81	0.03	0.07	9.79	401.73
4[all]	2	majority	176.41	0.03	0.32	6.92	237.92
		LCE	174.90	0.03	0.19	6.90	241.73
	3	majority	106.80	0.04	0.08	9.59	416.96
		LCA	102.97	0.03	0.10	9.55	514.80

Table 5.13: Performance metrics of various consensus functions for each n and k values considered on top 20 tests subset

Hence, from the individual and ensemble clustering results, we had a similar conclusion when compared with the previous subsets and are the following:

- Combining the results of kmeans from multiple runs had better performance

than combining the results of all the algorithms.

- The consensus function 'majority' outperformed all other consensus functions in all the experiments.

- kmeans is selected as the top-performing algorithm for each increment of n.

- By comparing the performance metrics of individual clustering and ensembles, we could see that the ensembles have the upper hand over the conventional clustering algorithms.

- Among k=2 and k=3, we considered k = 3 as the optimal value for the same reason mentioned in the inference of top 10 tests subset results.

Finally, we clustered the top 20 tests' data using the majority voting function at k=3 and n = 1. The proportion of the clusters and two-dimensional representation of the clustered data is presented in Figure 5.11. The clusters obtained are then used to extract each cluster's key features. For this, we took the cluster assignments and mapped them with the original data to obtain the actual result value of each pathological test. These original values are then averaged per test and cluster and are then represented as a heatmap in figure 5.12 for easier comparison. This plot shows that the tests Alanine Transferase, Aspartate Aminotransferase, serum creatinine level, hemoglobin, platelet count, blood urea nitrogen and eGFR have significant changes across the clusters.

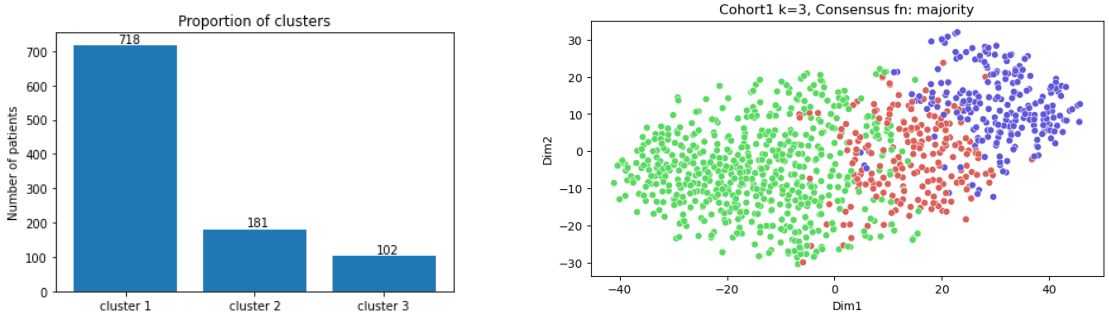

Figure 5.11: Proportion and Visualization of clusters on top 20 tests subset

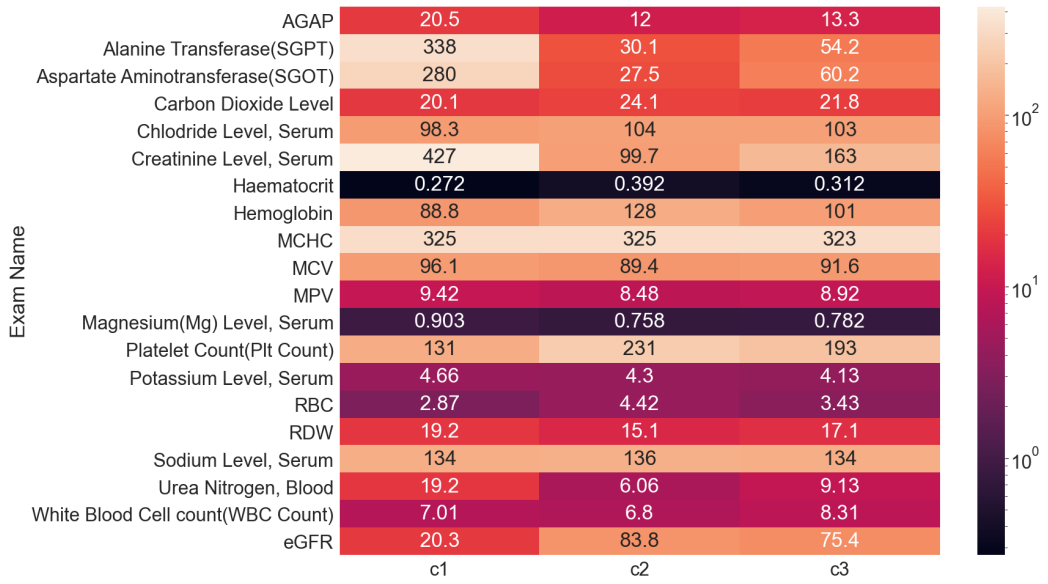

Figure 5.12: Comparison of feature values across the 3 clusters formed on top 20 tests subset

5.1.4 Summary of cluster analysis

From all the three subsets experimented with, we made the following conclusions from the whole clustering experiment:

- 'kmeans' is the best suited individual clustering algorithm, whereas' majority voting' is the best performing consensus function for the data.
- The best number of clusters in all the three subsets is k = 3.
- The most significant tests that differentiate the clusters in the top 10 tests' subset are AGAP, Carbon dioxide level, serum creatinine level, hematocrit, hemoglobin, RBC and blood urea nitrogen.
- In the next subset, i.e., the next 10 tests subset, Alanine Transferase, Aspartate Aminotransferase, MCV, MPV, platelet count, RDW, WBC and eGFR tests can be used to distinctly identify patients of each cluster.
- The tests AGAP, Alanine Transferase, Aspartate Aminotransferase, carbon dioxide, serum creatinine level, hematocrit, hemoglobin, MCV, MPV, magnesium level, platelet count, RBC, RDW, blood urea nitrogen, WBC and eGFR can all clearly distinguish the subgroups of the top 20 tests.

- Because all the tests that made a difference in the top 10 and next 10 tests subsets are included in the top 20 tests subset, we may consider the outcomes of the top 20 considered tests to be the prominent phenotypes because they mainly encompass the patterns seen in the other two subgroups.

- The target clusters formed at each subset have an uneven distribution of patients making the clusters imbalanced.

- Among the three clusters formed in the top 10 tests' subset, the patients in the third cluster seem to have the highest mean values for serum creatinine level and blood urea nitrogen and the lowest value for the Hemoglobin test. In contrast, the patients in the second cluster are contradictory to the third cluster with the lowest creatinine and urea nitrogen levels and highest hemoglobin levels.

- We can infer from the clusters formed on the subset of the next 10 tests that patients in cluster 3 have higher alanine and aspartate levels and lower platelet count and eGFR, which appears to contradict the second cluster, which has lower alanine and aspartate levels and higher platelet count and eGFR.

- Among the three clusters formed in the top 20 tests' subset, the patients in the first cluster have the highest mean values significantly for Alanine Transferase, Aspartate Aminotransferase, serum creatinine level, and blood urea nitrogen, the lowest value for eGFR and hemoglobin test.

- In contradiction to the first cluster of the top 20 tests' subset, the second cluster has the lowest average values for Alanine Transferase, Aspartate Aminotransferase, serum creatinine level and blood urea nitrogen. It has the highest average results for platelet count, hemoglobin and eGFR tests.

- We might deduce that the top 20 tests subset covers the patterns of the top 10 and next 10 tests' subsets based on the cluster patterns found from the mean of the actual test results of all three subsets.

Hence, we decided to move forward with the top 20 tests' subset with these observations as it is a combined inference of all the subsets. Some observations from the phenotypes noted for the top 20 tests are:

- Patients in cluster1 may face serious diabetic complications like ketoacidosis(higher AGAP), liver problem(more elevated alanine transferase and aspartate aminotransferase), kidney problem(higher creatinine) and anemia(Higher MCHC, MCV) comparing patients from other groups.
- Patients in cluster2 may be anemic(higher MCHC), may have heart problems(more elevated hematocrit, lung problem, hemoglobin, platelet count, RBC and RDW) and may have mild kidney damage(higher eGFR) when compared to the patients in other clusters.
- Patients in cluster3 may face mild kidney damage due to higher eGFR results.
- We could also make an assumption from the frequency of the tests taken by patients in cluster1 is that they may be inpatients(i.e., patients admitted within the facility) as they have taken almost all the considered tests multiple times daily for several successive days.

Apart from investigating just the 20 common tests, we extended our investigation of phenotypes to other tests available especially focusing on the common diabetics-specific tests. Following are the commonly ordered tests we considered where the first two are the glucose-related tests, and the last two are the urine-related tests :

1. Glucose Level, Random (Blood Sugar)
2. Glucose Level, Fasting (FBS)
3. Hemoglobin A1C (Hgb A1C)
4. UA Blood
5. UA Glucose

Figure 5.13 presents the mean, minimum value, maximum value, median value and standard deviation of the results of these selected diabetes-related tests. This figure shows that the patients in each cluster are distinguishable using these diabetes-related tests.

By further analyzing the nature of these test results on the identified patient groups, we inferred the following observations on the identified phenotypes:

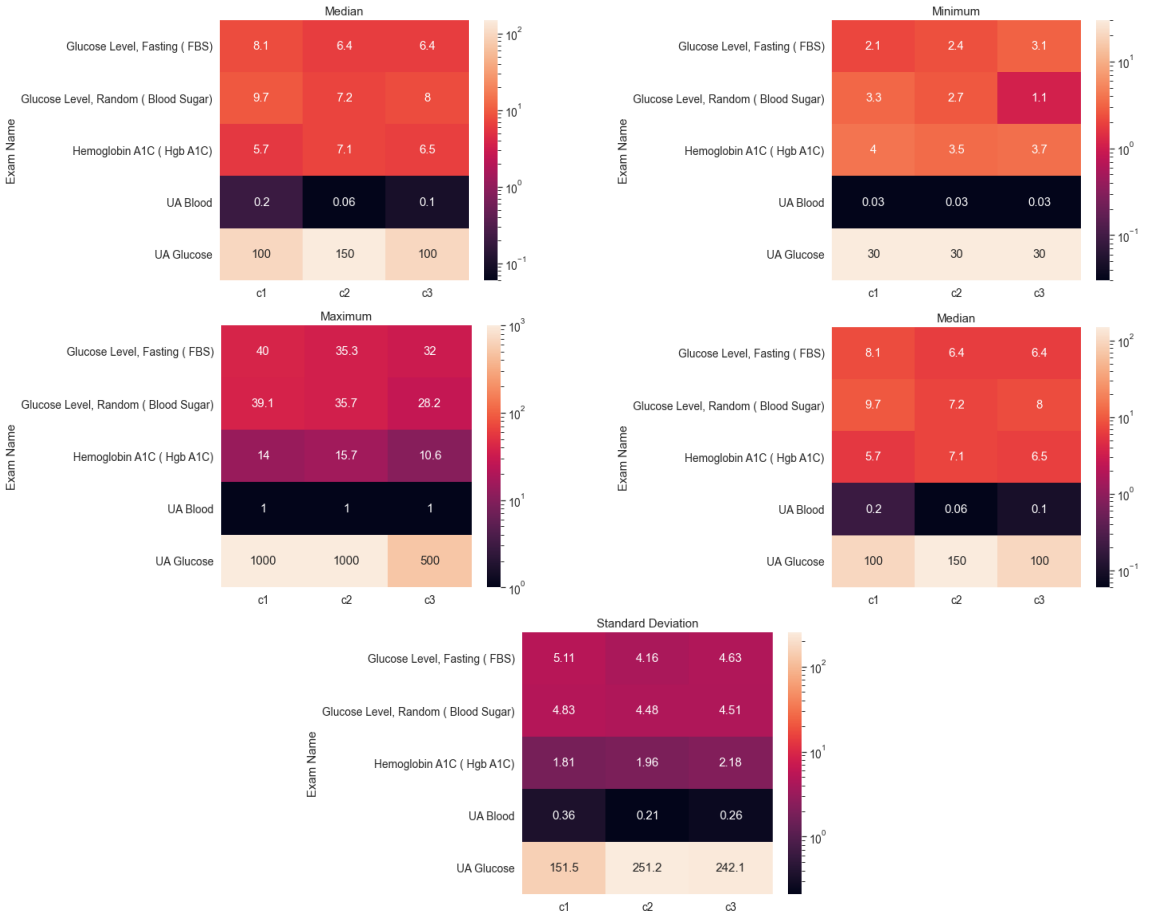

Figure 5.13: Statistics of the diabetes-related tests for each clusters

- The random glucose level seems to increase in the cluster1 patients, whereas it declines with time for the cluster2 and cluster3 patients.

- The fasting glucose level decreased over time for most of the patients in cluster1, whereas it increased in cluster2 and cluster3.

- The hemoglobin A1C value increased with respect to time for most of the patients in cluster1, which is the opposite for the patients in cluster2. We couldn't observe any specific trend among the patients in cluster3 as they seemed to have not taken this test frequently, with most of the patients attempting the test exactly once.

- The Urine Analysis(UA) Blood seems to have the same range of value in all three clusters. But it seems to increase with time for the patients in cluster2

and decrease for those in cluster1 and cluster3.

- The Urine Analysis(UA) Glucose results seemed to increase with time for individuals in cluster1. In contrast, its trend is inconclusive in the other two clusters, as most patients appeared to have taken the test exactly once.

Finally, from the temporal characteristics of the top 20 tests' subset, we inferred the following:

Tests	Cluster1	Cluster2	Cluster3	Indicator
AGAP	-0.003049	-0.000642	-0.000526	
SGPT	0.446229	-0.001684	-0.00035	
SGOT	0.311024	-0.001097	0.002485	
CO2 Level	0.004044	-0.000145	0.001067	
Cl Level	0.00433	0.000876	0.001833	
Creatinine Level	-0.25583	0.003503	-0.020396	
Hct	-0.000205	-0.000004	-0.000001	
Hgb	-0.061218	-0.000567	-0.000726	
MCHC	-0.002604	0.001887	-0.000012	
MCV	0.002953	0.000003	0.000702	
MPV	0.00248	0.000479	0.000547	
Mg Level	0.000342	-0.000009	-0.00001	
Plt Count	-0.088616	-0.008769	-0.01723	
K Level	-0.001862	-0.000015	0.000019	
RBC	-0.002182	-0.000046	-0.000045	
RDW	0.003256	0.000181	0.000638	
Na Lvl	0.007667	0.000153	0.002343	
BUN	-0.01033	0.000065	0.000311	
WBC Count	0.014698	0.000185	-0.00028	
eGFR	0.02119	-0.00167	0.000986	

Figure 5.14: Table indicating increase or decrease in the mean of slopes of top 20 tests' subset

We used the slope computed for individual patients for each of the 20 tests considered to identify the change in the results of the tests concerning time. By averaging the slopes of each patient in each cluster, we were able to deduce an overall rise or reduction in each of the test outcomes. Figure 5.14 shows the average slopes of each test across each cluster. The indicator column indicates the overall increase or decrease in the slope of each test using blue and red blocks, respectively.

From the indicators, we could see that the tests AGAP, Hct, Hgb, Plt count and RBC have a decline in the slope across clusters, indicating that the patients in the considered period showed an overall decrease in the test results irrespective of their clusters. On the other hand, the tests Cl level, MCV, MPV, RDW and Na Level had an increase in the value of the results over time irrespective of the clusters. The remaining tests, SGPT, SGOT, CO2 level, creatinine level, MCHC, Mg level, K level, BUN, WBC count, and eGFR, are crucial for each cluster since it differs among the clusters.

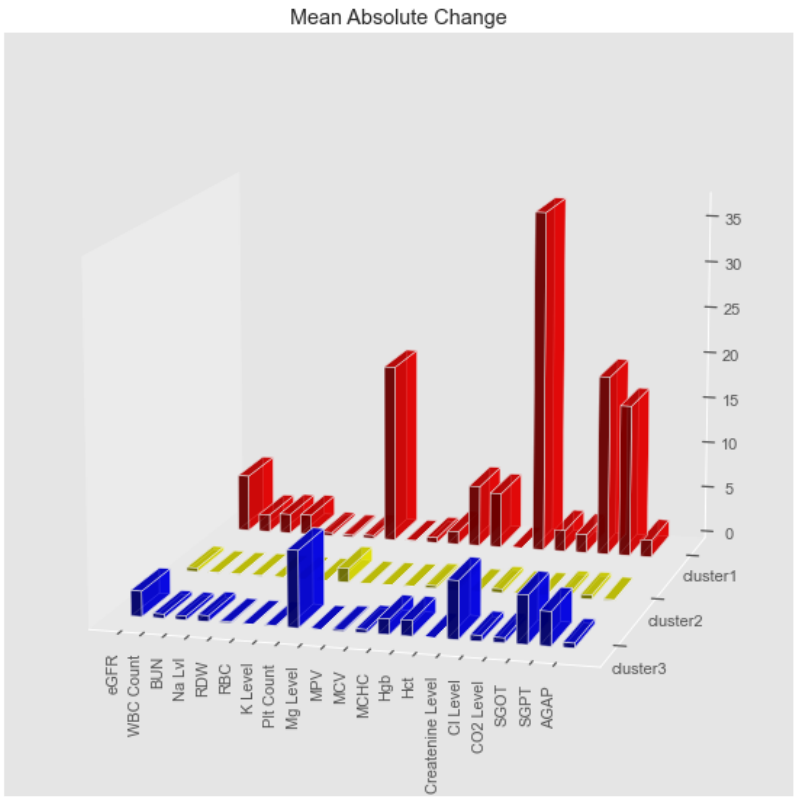

Figure 5.15: Average of the amount of change in the results over time in each cluster of top 20 tests' subset

The following inference is obtained using each patient's mean absolute change values to identify the overall change in the results concerning time. For this, similar to slopes, we computed the mean absolute change of each patient's results over time, which are then averaged for the cluster assignment of the patients. Figure 5.15 shows the average change in the results over time in each cluster using a three-dimensional bar plot.

This chart shows that the patients in cluster 1 have the most significant change in most tests, especially for SGPT, SGOT, creatinine level, MCHC, Plt count, and eGFR. In contrast to cluster 1, patients in cluster 2 faced the lowest change in their test results than patients in other clusters.

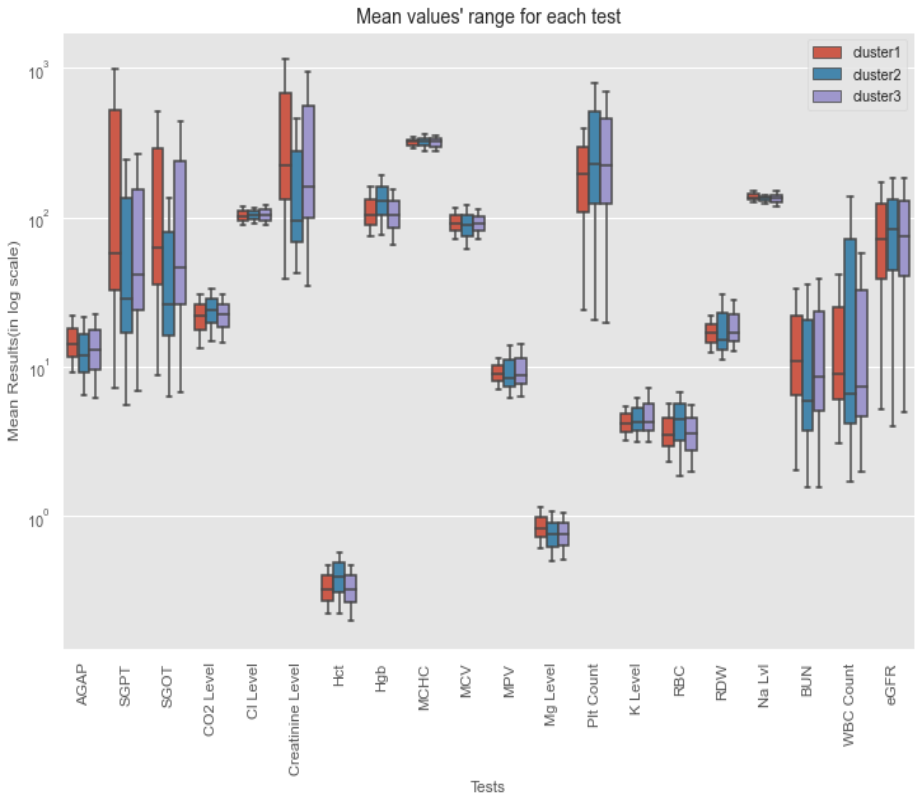

Figure 5.16: Range of the average result values over time in top 20 tests' subset

We also compared the overall range of results by averaging the mean of the result value of each patient. We did this by first calculating the mean of each patient's test results, then determining the lowest and maximum of each test across each cluster. Figure 5.16 represents the range of the mean result values of each cluster using a grouped box plot. The figure shows that the tests AGAP, CO2 Level, Cl Level, MCHC, MCV, MPV, Mg Level, Na Level and eGFR have almost the same results value across the clusters. The tests SGPT, SGOT, creatinine level, Hgb, Plt count and WBC count are comparable across clusters as the range of the results have significantly differed among the cluster.

Apart from these temporal characteristics, we have tabulated the maximum time an individual repeated a test in the considered interval in Table 5.14. The table

Tests	**Cluster1**	**Cluster2**	**Cluster3**
AGAP	233	56	110
SGPT	69	56	92
SGOT	69	56	94
CO2 Level	237	56	111
Cl Level	234	56	111
Creatinine Level	237	56	111
Hct	105	56	141
Hgb	105	56	141
MCHC	105	56	141
MCV	105	56	141
MPV	105	56	141
Mg Level	137	57	99
Plt Count	105	56	133
K Level	238	56	112
RBC	105	56	141
RDW	105	56	141
Na Level	239	56	111
BUN	236	56	111
WBC Count	105	56	141
eGFR	277	56	113

Table 5.14: Maximum repeats of each test by a single patient in the top 20 tests' subset

shows that eGFR is the maximum repeated test taken 277 times by a patient in cluster 1. In contrast, Hct, Hgb, MCHC, MCV, MPV, RBC, RDW and WBC counts were the most frequently taken tests taken 141 times by a patient in cluster 3 in the considered interval. The patients in cluster2 have undergone almost all of the tests for a maximum of 56 repetitions.

We also used the incomplete set of the top 20 tests' subset to find the percentage of patients who have missed taking each of the considered tests. Figure 5.17 compares the missing percentage of each test across each cluster.

From the plot, we can see that the bars of the tests Hct, Hgb, MCHC, MCV, MPV, Plt count, RBC, RDW and WBC look identical. After additional investigation, we discovered that these tests are ordered together regardless of clusters, i.e., a patient who missed MCV is likely to have missed MPV, MCHC, Hgb, Hct, platelet count, RBC, RDW, and WBC tests.

Figure 5.17: Percentage of patients who have missed taking each test per cluster in incomplete data of top 20 tests subset

Thus, from all the observations, we conclude that we have successfully divided the data into meaningful clusters. These clusters are then equipped for our further analysis.

5.2 Imputation Results

The imputation process is carried out on the top 20 tests' subset, i.e., the data of patients who have taken all the top 20 most frequently occurring pathological tests. After processing the base set of this data and identifying the phenotypes, we are then processing the incomplete set of data to estimate the disease progression of its patients. Since there are many void entries in the top 20 tests' subset, we imputed the null values with non-null entries using univariate multivariate and knn imputation

techniques. In particular, they replaced the void entries using mean, median, and zero strategies of univariate imputation. In the case of multivariate imputation, we have used Bayesian ridge, decision tree and gaussian process regressors with iterative imputation function in python. To evaluate the accuracy of these techniques before applying them to incomplete set data, we have used these imputation techniques to the randomly masked base set data. As the highest proportion of missing values in a column of this subset's incomplete set is 27 percent, we randomly voided the base set records in 10, 20, and 30 percent. We repeated the entire evaluation process using RepeatedKFold with five repetitions and splits, and the metric used for evaluation is Root Mean Square Error.

Model	**Strategy**	**RMSE for each percentage missing**		
		10%	20%	30%
Simple Imputation	Mean	0.27	0.40	0.50
	Median	0.28	0.41	0.51
	Zero	0.27	0.40	0.50
Iterative imputation	Bayesian Ridge	0.13	0.24	0.32
	Decision Tree	0.23	0.37	0.47
	Gaussian Process	0.26	0.39	0.48
kNN imputation		0.18	0.27	0.36

Table 5.15: Root Mean Square Error(RMSE) values computed by applying various imputation techniques on the masked test set

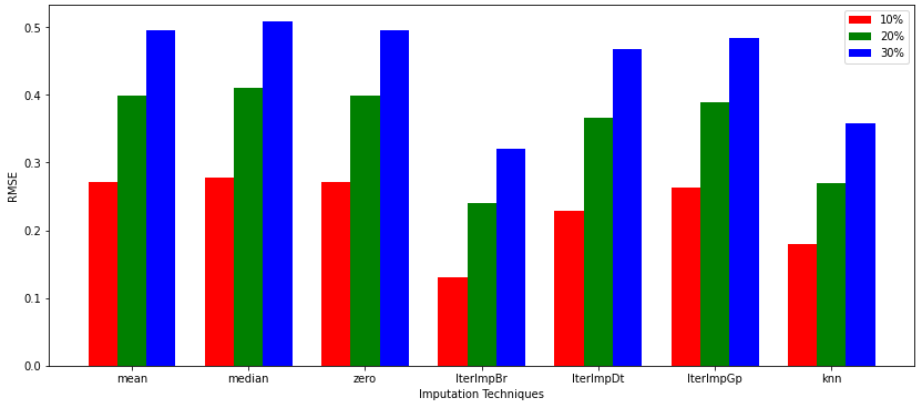

Figure 5.18: Comparison of the performance of different imputation techniques

Table 5.15 and Figure 5.18 presents the RMSE values of each imputation techniques for various percent of missing values. From them, we could infer that the

iterative imputation with Bayesian ridge regressor performed better than any other imputation technique. Hence, using this technique, we trained the imputer with the base set data, transformed the incomplete set data into non-null data and made the data ready for the classification process.

5.3 Classification Results

The main aim of the classification task was to assign the imputed incomplete set to appropriate clusters that are formed on the base set. This process aims to identify the cluster to which a particular patient might belong if he had attended the considered tests. We might also use this assignment to predict the approximate result value of the considered pathology tests that the patient did not encounter in the given period. We carried out the classification process using the entire data of the top 20 tests' subset, i.e., we have utilized the base set and the incomplete set for classification. In order to train a classification algorithm for predictive analysis, we need a dataset labelled by experts. We treated the labels acquired using the clustering models as the original labels and utilized them for training the classification models because the data we used for our research was not labelled.

We tested SVM, GNB, KN, DT, and RF models on our labelled base data to find the best classification model for our data. Using a repeated stratified kfold cross validator, we divided the data into train and test sets. The base set was divided into five-folds and repeated three times. Once the data is split, each training fold is oversampled using SMOTE to balance the dataset. The oversampled train data is then used to train the selected models, and the test data are used for prediction. The predicted labels of the test set by the model and the actual label are used to evaluate the models' performance. The metrics used for evaluation are precision, recall and score. The results of each fold and repeat were recorded and then averaged to determine the models' overall performance.

Table 5.16 shows the precision-recall and score obtained by applying the considered algorithms on the top 20 tests' subset base set. In addition to these scores, we compared the performance of the models using the confusion matrix and the ROC curves. We observed that the SVM outperformed all other models from these evaluation metrics. Hence, we considered SVM for classifying our data. Figure 5.19 shows

Models	Precision	Recall	fscore
SVM	97.35	97.00	97.07
GNB	92.95	90.91	91.50
KN	96.36	95.90	96.01
DT	91.83	91.61	91.67
RF	95.52	95.40	95.40

Table 5.16: Performance comparison of classification models

the confusion matrix and ROC curve of SVM model.

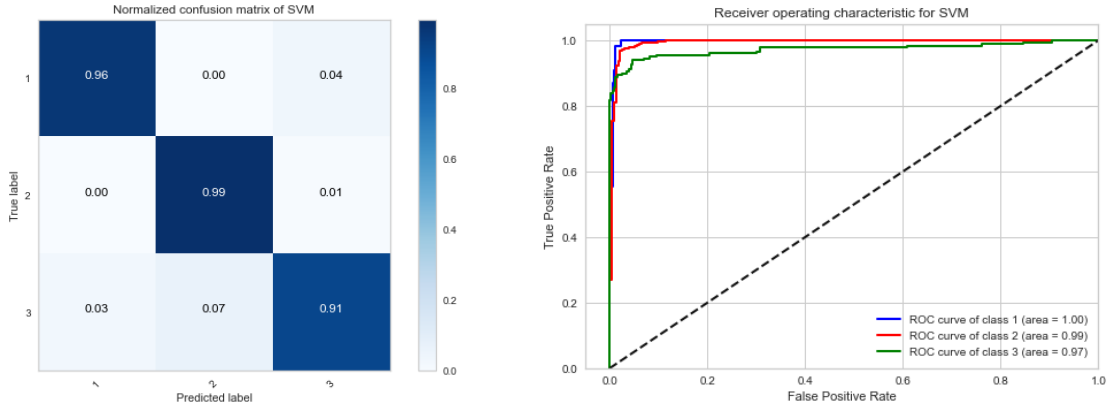

Figure 5.19: Confusion Matrix and ROC curve of Support Vector Machine(SVM) model on base set of top 20 tests subset

For cluster prediction, we trained the SVM model using the entire data of the base set and predicted the clusters of the imputed incomplete set. Figure 5.20 shows the proportion of the incomplete set of patients assigned to each cluster via the predictions of classification models.

From the labels obtained using classification, we can tell if a patient needs to take a particular test based on the average value in the cluster to which he is assigned. This can, in turn, help the doctors predict the disease's progression among the patients. This average test result also reduces the unnecessary laboratory utilization as the patients attend the tests that are assumed to be essential. Below are some examples from the incomplete set explaining the significance of predicting the clusters of the patients.

Sample 1:

Let us consider the patient with patient number 5023. This patient has taken 11 out of 20 tests that we considered. Figure 5.21 and Table 5.17 show the original

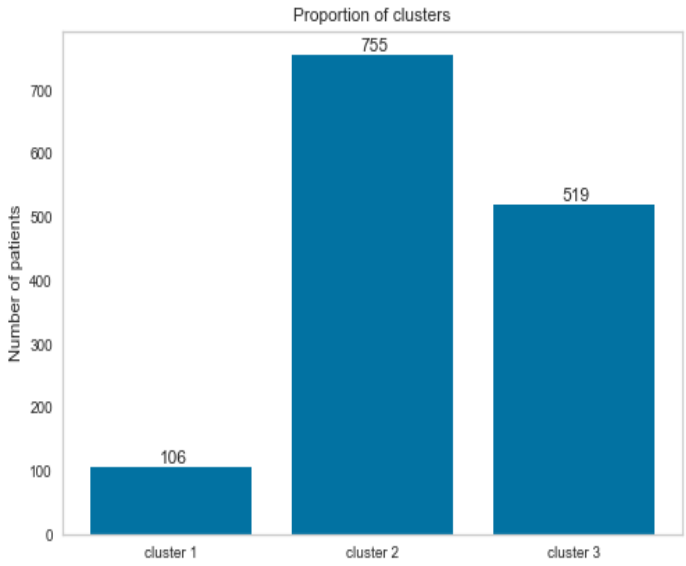

Figure 5.20: Proportion of incomplete set patients per cluster

data and the list of the tests along with the number of repetitions done by the patient, respectively. The temporal characteristics are calculated using the tsfresh package from the original data. Figure 5.22 is the snippet of the computed temporal characteristics.

	ETPR_PT_NO	MRN	PACT_ID	EXM_CD	EXM_NM	BRFG_DTM	RESULT
445	5023	2483	5203832	L3900102	AGAP	2017-02-28 12:25:21	12.00
446	5023	2483	9409880	L3900102	AGAP	2017-06-06 11:29:18	12.00
447	5023	2483	5203832	L3900109	Alanine Transferase (SGPT)	2017-02-28 12:25:21	14.00
448	5023	2483	9409880	L3900109	Alanine Transferase (SGPT)	2017-06-06 11:29:18	18.00
449	5023	2483	5203832	L3900111	Aspartate Aminotransferase (SGOT)	2017-02-28 12:25:21	14.00
450	5023	2483	9409880	L3900111	Aspartate Aminotransferase (SGOT)	2017-06-06 11:29:18	14.00
451	5023	2483	5203832	L3900231	Carbon Dioxide Level (CO2)	2017-02-28 12:25:21	24.00
452	5023	2483	9409880	L3900231	Carbon Dioxide Level (CO2)	2017-06-06 11:29:18	23.00
453	5023	2483	5203832	L3900234	Chloride Level, Serum (Cl Level)	2017-02-28 12:25:21	105.00
454	5023	2483	9409880	L3900234	Chloride Level, Serum (Cl Level)	2017-06-06 11:29:18	104.00
455	5023	2483	5203832	L3900104	Creatinine Level, Serum	2017-02-28 12:25:21	70.00

Figure 5.21: Sample original records with result of patient 5023

	AGAP__linear_trend_timewise__attr_"slope"	AGAP__mean	AGAP__mean_abs_change	AGAP__pk_pk_distance	AGAP__variance	Alanine Transferase (SGPT)__linear_trend_timewise__attr_"slope"	Alanine Transferase (SGPT)__mean	Alanine Transferase (SGPT)__mean_abs_change	Alanine Transferase (SGPT)__pk_pk_distance	Alanine Transferase (SGPT)__variance	...	White Blood Cell Count (WBC Count)__linear_trend_timewise__attr_"slope"	White Blood Cell Count (WBC Count)__mean	White Blood Cell Count (WBC Count)__mean_abs_change	White Blood Cell Count (WBC Count)__pk_pk_distance	White Blood Cell Count (WBC Count)__variance	eGFR__linear_trend_timewise__attr_"slope"	eGFR__mean	eGFR__mean_abs_change	eGFR__pk_pk_distance	eGFR__variance
5023	0	12	0	0	0	0.001701	16	0.040816	4	1.360544	...	NaN	NaN	NaN	NaN	NaN	0.00085	82	0.020408	2	0.340136

Figure 5.22: Snapshot of the temporal characters computed by tsfresh for 5023 patient

From Figure 5.22, we could see that the temporal features for WBC count are

EXM_NO	No. of times taken
AGAP	2
Alanine Transferase (SGPT)	2
Aspartate Aminotransferase (SGOT)	2
Carbon Dioxide Level (CO2)	2
Chloride Level, Serum (Cl Level)	2
Creatinine Level, Serum	2
Magnesium(Mg) Level, Serum	2
Potassium Level, Serum (K Level)	2
Sodium Level, Serum (Na Lvl)	2
Urea Nitrogen, Blood(BUN)	2
eGFR	2

Table 5.17: Tests taken by patient 5023 and the number of times the tests are repeated

populated as NaN by tsfresh as this patient did not attempt this test. We could also see that the slope, mean absolute change, peak to peak distance and variance computed for AGAP is '0' as this patient did not see any change in their result. These computed temporal features are then standardized and used to predict the cluster this patient belongs to. The classification model assigned this patient to cluster 2. Using the inference Figure 5.12, we could say that this patient might get the following results for the unseen tests:

Tests	Avg. result in cluster 2
Haematocrit (Hct)	0.392
Hemoglobin (Hgb)	128
MCHC	325
MCV	89.4
MPV	8.48
Platelet count (Plt Count)	231
RBC	4.42
RDW	15.1
White Blood Cell Count (WBC Count)	6.8

Table 5.18: Forecasted results for tests unseen by patient 5023

This predicted value may help the physicians decide if the patient needs to undergo these tests. It may also help them to personalize the care given to the patient as they can foresee the onset of any complications using the predicted values and cluster characteristics.

Sample 2:

Let us consider a patient with patient number 92255. This patient has seen all the 20 tests of our interest. Based on our illustrations, this patient must belong to the base set. However, this patient has taken nine tests just once, making the temporal characteristics except for the mean value NaN as there is no change in the results in the considered interval. Since the base set was filtered with non-null constraint, this patient was left out of the base set and was assigned to clusters using the classification model. Figure 5.23 and Table 5.19 are the sample and statistics of this patient's data respectively.

	ETPR_PT_NO	MRN	PACT_ID	EXM_CD	EXM_NM	BRFG_DTM	RESULT
22446	92255	90710	14027419	L3900102	AGAP	2017-01-18 13:03:45	11.00
22447	92255	90710	11586282	L3900109	Alanine Transferase (SGPT)	2017-01-17 10:54:18	23.00
22448	92255	90710	14027419	L3900109	Alanine Transferase (SGPT)	2017-04-11 11:21:09	22.00
22449	92255	90710	11586282	L3900111	Aspartate Aminotransferase (SGOT)	2017-01-17 10:54:18	22.00
22450	92255	90710	14027419	L3900111	Aspartate Aminotransferase (SGOT)	2017-04-11 11:21:09	21.00
22451	92255	90710	14027419	L3900231	Carbon Dioxide Level (CO2)	2017-01-18 13:03:45	24.00
22452	92255	90710	14027419	L3900234	Chloride Level, Serum (Cl Level)	2017-01-18 13:03:45	106.00
22453	92255	90710	14027419	L3900104	Creatinine Level, Serum	2017-01-18 13:03:45	65.00
22454	92255	90710	11586282	L2000011	Haematocrit (Hct)	2017-01-17 12:05:32	0.33
22455	92255	90710	14027419	L2000011	Haematocrit (Hct)	2017-04-11 13:04:40	0.51
22456	92255	90710	11586282	L2000009	Hemoglobin (Hgb)	2017-01-17 12:05:32	93.00
22457	92255	90710	14027419	L2000009	Hemoglobin (Hgb)	2017-04-11 13:04:40	145.00
22458	92255	90710	11586282	L2000018	MCHC	2017-01-17 12:05:32	278.00
22459	92255	90710	14027419	L2000018	MCHC	2017-04-11 13:04:40	287.00
22460	92255	90710	11586282	L2000015	MCV	2017-01-17 12:05:32	83.30
22461	92255	90710	14027419	L2000015	MCV	2017-04-11 13:04:40	80.50
22462	92255	90710	11586282	L2000010	MPV	2017-01-17 12:05:32	7.00
22463	92255	90710	14027419	L2000010	MPV	2017-04-11 13:04:40	7.70
22464	92255	90710	11586282	L3000007	Magnesium(Mg) Level, Serum	2017-01-17 10:54:18	0.69
22465	92255	90710	11586282	L2000006	Platelet Count (Plt Count)	2017-01-17 12:05:32	110.00
22466	92255	90710	14027419	L2000006	Platelet Count (Plt Count)	2017-04-11 13:04:40	102.00
22467	92255	90710	14027419	L3900232	Potassium Level, Serum (K Level)	2017-01-18 13:03:45	3.90
22468	92255	90710	11586282	L2000008	RBC	2017-01-17 12:05:32	4.00

Figure 5.23: Snapshot of original records of the patient 92255

After applying the tsfresh, we used the temporal features to assign this patient to the clusters. Figure 5.24 is a glimpse of the temporal features computed for this patient.

	AGAP__linear_trend_timewise__attr_"slope"	AGAP__mean	AGAP__mean_abs_change	AGAP__pk_pk_distance	AGAP__variance	Alanine Transferase (SGPT)__linear_trend_timewise__attr_"slope"	Alanine Transferase (SGPT)__mean	Alanine Transferase (SGPT)__mean_abs_change	Alanine Transferase (SGPT)__pk_pk_distance	Alanine Transferase (SGPT)__variance	...	White Blood Cell Count (WBC Count)__linear_trend_timewise__attr_"slope"	White Blood Cell Count (WBC Count)__mean	White Blood Cell Count (WBC Count)__mean_abs_change	White Blood Cell Count (WBC Count)__pk_pk_distance	White Blood Cell Count (WBC Count)__variance	eGFR__linear_trend_timewise__attr_"slope"	eGFR__mean	eGFR__mean_abs_change	eGFR__pk_pk_distance	eGFR__variance
92255	NaN	11	NaN	0	0	-0.000496	22.5	0.011905	1	0.085317	...	0.000025	2.755	0.000595	0.05	0.000213	NaN	113	NaN	0	0

Figure 5.24: Snapshot of the temporal characters computed by tsfresh for patient 92255

EXM_NM	No. of times taken
AGAP	1
Alanine Transferase (SGPT)	2
Aspartate Aminotransferase (SGOT)	2
Carbon Dioxide Level (CO2)	1
Chloride Level, Serum (Cl Level)	1
Creatinine Level, Serum	1
Haematocrit (Hct)	2
Hemoglobin (Hgb)	2
MCHC	2
MCV	2
MPV	2
Magnesium(Mg) Level, Serum	1
Platelet count (Plt Count)	2
Potassium Level, Serum (K Level)	1
RBC	2
RDW	2
Sodium Level, Serum (Na Lvl)	1
Urea Nitrogen, Blood(BUN)	1
White Blood Cell Count (WBC Count)	2
eGFR	1

Table 5.19: Tests taken by the patient 92255 and the number of times the tests are repeated.

Our classification model assigned this patient to the cluster 2. From Figure 5.12 we could see that the average values of each test and the original values of this patient mentioned in Figure 5.23 is approximately equal to each other proving the patient to be a good fit for the cluster.

Sample 3:

Let us consider the patient with the patient number 3029990. This patient has just taken one test out of twenty considered pathological tests throughout the time window. Figure 5.25 shows the entire data of this patient. Computing temporal features resulted in the data with nulls for all other tests except eGFR. Figure 5.26 is the snapshot of the temporal features.

These temporal characters are then imputed using the base set and then assigned to a cluster by the classification model. This patient was assigned to cluster 1 by our trained classification model. Using the inferences of cluster 1, the physicians can

	ETPR_PT_NO	MRN	PACT_ID	EXM_CD	EXM_NM	BRFG_DTM	RESULT
323576	3029990	2815513	21736245	L3000002	eGFR	2017-03-13 09:16:43	56.0
323577	3029990	2815513	21736245	L3000002	eGFR	2017-03-14 03:51:49	38.0
323578	3029990	2815513	21736245	L3000002	eGFR	2017-03-15 06:13:03	38.0
323579	3029990	2815513	21736245	L3000002	eGFR	2017-03-16 06:17:55	36.0
323580	3029990	2815513	21736245	L3000002	eGFR	2017-03-17 06:23:23	29.0
323581	3029990	2815513	21736245	L3000002	eGFR	2017-03-18 06:14:49	18.0
323582	3029990	2815513	21736245	L3000002	eGFR	2017-03-19 07:50:17	11.0
323583	3029990	2815513	21736245	L3000002	eGFR	2017-03-20 07:32:05	8.0
323584	3029990	2815513	21736245	L3000002	eGFR	2017-03-21 04:38:15	7.0
323585	3029990	2815513	21736245	L3000002	eGFR	2017-03-22 05:23:06	6.0

Figure 5.25: Original data of patient 3029990

	AGAP__linear_trend_timewise__attr_"slope"	AGAP__mean	AGAP__mean_abs_change	AGAP__pk_pk_distance	AGAP__variance	Alanine Transferase (SGPT)__linear_trend_timewise__attr_"slope"	Alanine Transferase (SGPT)__mean	Alanine Transferase (SGPT)__mean_abs_change	Alanine Transferase (SGPT)__pk_pk_distance	Alanine Transferase (SGPT)__variance	...	White Blood Cell Count (WBC Count)__linear_trend_timewise__attr_"slope"	White Blood Cell Count (WBC Count)__mean	White Blood Cell Count (WBC Count)__mean_abs_change	White Blood Cell Count (WBC Count)__pk_pk_distance	White Blood Cell Count (WBC Count)__variance	eGFR__linear_trend_timewise__attr_"slope"	eGFR__mean	eGFR__mean_abs_change	eGFR__pk_pk_distance	eGFR__variance
3029990	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	-0.22803	24.7	5.555556	50	265.41

Figure 5.26: Snapshot of the temporal characters computed by tsfresh for patient 3029990

assess the patient's condition and take the necessary steps to check the complications.

Sample 4:

Let us consider the patient with patient number 71916. This patient has taken just the eGFR test, similar to the patient in sample 3, but unlike sample 3, this patient has taken the test just once. Figure 5.27 is the data of this patient.

	ETPR_PT_NO	MRN	PACT_ID	EXM_CD	EXM_NM	BRFG_DTM	RESULT
19103	71916	70000	22240777	L3000002	eGFR	2017-03-22 09:27:57	96.0

Figure 5.27: Original data of patient with patient number as 71916

	AGAP__linear_trend_timewise__attr_"slope"	AGAP__mean	AGAP__mean_abs_change	AGAP__pk_pk_distance	AGAP__variance	Alanine Transferase (SGPT)__linear_trend_timewise__attr_"slope"	Alanine Transferase (SGPT)__mean	Alanine Transferase (SGPT)__mean_abs_change	Alanine Transferase (SGPT)__pk_pk_distance	Alanine Transferase (SGPT)__variance	...	White Blood Cell Count (WBC Count)__linear_trend_timewise__attr_"slope"	White Blood Cell Count (WBC Count)__mean	White Blood Cell Count (WBC Count)__mean_abs_change	White Blood Cell Count (WBC Count)__pk_pk_distance	White Blood Cell Count (WBC Count)__variance	eGFR__linear_trend_timewise__attr_"slope"	eGFR__mean	eGFR__mean_abs_change	eGFR__pk_pk_distance	eGFR__variance
71916	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	96	NaN	0	0

Figure 5.28: Snapshot of the temporal characters computed by tsfresh for patient 71916

The temporal features computed using this data will possess most of the values as

NaN as we have very minimal data. To assign this patient to a cluster, we imputed the temporal features using a base set to make the data similar to the base set. The classification model assigned this patient to cluster 2.

Thus, these samples show us how a new patient or existing patient can be assigned to the clusters despite the tests not attempted by them.

Chapter 6

Conclusion

This chapter gives an overall view of the thesis, including the outcomes, limitations and possible future works.

6.1 Summary

The main focus of this thesis is to identify the phenotypes of diabetes patients by effectively utilizing the diabetes pathology data without any demographic information. For this purpose, we have investigated various clustering models and used ensemble clustering to group the patients. We have utilized the temporal laboratory data of diabetes patients from King Fahad Hospital. This data held the results of various pathology taken by diabetes patients from January 2017 to July 2017. Despite the absence of additional information about the patients, we proceeded with the research with just four original features: ETPR_PT_NO (patient number), EXM_NM (exam name) and BRFG_DTM (date-time when the test was taken) and RESULTS (outcome of lab test conducted). Among all the exams available in the data, we used the top 20 frequently occurring tests among the patients since the minimal population just took many tests. We then filtered the entire data and retained the patients who had witnessed at least one of the 20 commonly occurring tests. We considered 20 as taking more than 20 common tests reduced the data available for research.

The next step we carried out was to split the data into three subsets based on the tests taken by the patients' top 10, next 10 and top 20 tests. There was a need to regularize the subsets as the data was sparse. To normalize the filtered time-series data using resampling and interpolation technique which was followed by computing the temporal features using the tsfresh package from the regularized data. The computed temporal features seemed to have many null entries as there were a missing few tests. This temporal feature set was then divided into the base set and incomplete set depending on the presence of void entries in a patient's record, i.e., if

a patient has taken all the tests of interest more than once in the considered interval, he is considered under base set, and if a patient at least one of the considered tests or has not taken all the tests more than once, the details of this patient is saved under incomplete set.

The base set is then grouped and analyzed using AP, k-means, GMM and PAM clustering algorithms. Several consensus functions such as CSPA, k-modes, LCA, LCE, and majority voting were applied to the base set. Their performance was compared against each other and the single clustering algorithms using several internal evaluation metrics. From the metrics, we found that the best k value for our data was k= 3. We also discovered that consensus clustering had better performance over individual clustering algorithms. For our data, k-means was the best individual clustering algorithm, and majority voting was the best consensus function. Based on our analysis, we used the cluster assignments made by the majority voting function as our diabetes phenotypes. The pathology that distinguishes each cluster are Alanine Transferase, Aspartate Aminotransferase, serum creatinine level, hemoglobin, platelet count, blood urea nitrogen and eGFR.

The next goal of our thesis was to predict the progression of the disease among the new patients by utilizing the observed phenotypes. For this, we have utilized the incomplete set of data. This incomplete set was imputed using the base set by iterative imputation technique with Bayesian Ridge regressor as it excelled in performance compared to others we considered. The imputed data was then assigned to appropriate clusters using classification models. We evaluated SVM, GNB, KNC, DT and RF classifiers and utilized SVM as this had higher performance than others. By combining the classification and clustering, we utilized the available dataset and effectively grouped all the patients of interest. Using the characters of the clusters, the physician can predict the progression of the disease among the patients in the incomplete set. The physician can also order the tests that the patient might not have taken, depending on the cluster he falls under. This, in turn, will prevent overutilization or improper utilization of laboratories. Thus, using our methodology, we could conclude that we could generate meaningful phenotypes using EHRs without demographic information.

6.2 Limitations

Even though our model had some significant advantages, it faced the following drawbacks:

- Using diceR for cluster analysis was computationally expensive as it needed more space and time to cluster our data. The system crashed several times while running multiple algorithms together and increasing our k value. We had to run each experiment individually, which cost us time.
- The interpretation of the clusters was limited to the observations we did on data available for the research but was not focused on the clinical aspects of diabetes as we had limited features available.

6.3 Future Work

One possible future work could be to interact and gain knowledge from the domain experts in interpreting our clusters and revising the phenotypes' outline based on the clinical significance. We could also get the domain experts to provide class labels to the patients to tune the model's performance by comparing our assignments with their labels using external validation metrics.

Bibliography

[1] Affinity propagation. https://www.machinecurve.com/index.php/2020/04/18/how-to-perform-affinity-propagation-with-python-in-scikit/. Accessed: 2021-11-30.

[2] Auc-roc curve in machine learning clearly explained. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/. Accessed: 2021-12-10.

[3] Data standardization. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed: 2021-11-25.

[4] Decision trees: Id3 algorithm explained. https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1. Accessed: 2021-12-10.

[5] Definition, classification and diagnosis of diabetes, prediabetes and metabolic syndrome. https://guidelines.diabetes.ca/cpg/chapter3#sec3. Accessed: 2022-01-06.

[6] Error sum of squares (sse). https://hlab.stanford.edu/brian/error_sum_of_squares.html. Accessed: 2021-12-06.

[7] Gaussian naive bayes. https://iq.opengenus.org/gaussian-naive-bayes/. Accessed: 2021-12-10.

[8] Knn classification using scikit-learn. https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn. Accessed: 2021-12-10.

[9] Ml | expectation-maximization algorithm. https://www.geeksforgeeks.org/ml-expectation-maximization-algorithm/. Accessed: 2021-11-30.

[10] Ml | k-medoids clustering with solved example. https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/. Accessed: 2021-11-30.

[11] Package 'dicer'. https://cran.r-project.org/web/packages/diceR/diceR.pdf. Accessed: 2021-12-02.

[12] Rocauc. https://www.scikit-yb.org/en/latest/api/classifier/rocauc.html. Accessed: 2021-12-10.

[13] Silhouette index – cluster validity index. https://www.geeksforgeeks.org/silhouette-index-cluster-validity-index-set-2/. Accessed: 2021-12-02.

[14] Tour of evaluation metrics for imbalanced classification. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/. Accessed: 2021-12-10.

[15] tsfresh feature extraction package modules. https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html. Accessed: 2021-11-25.

[16] Abbas Abbasi-Ghahramanloo, Ramin Heshmat, Amir-Masood Rafiemanzelat, Kimia Ghaderi, Mohammad Esmaeil Motlagh, Zeinab Ahadi, Gita Shafiee, Armita Mahdavi-Gorabi, Mostafa Qorbani, and Roya Kelishadi. Subgrouping of iranian children and adolescents based on cardiometabolic risk factors using latent class analysis: The caspian-v study. *Caspian Journal of Internal Medicine*, 11(4):370, 2020.

[17] Shahabeddin Abhari, Sharareh R Niakan Kalhori, Mehdi Ebrahimi, Hajar Hasannejadasl, and Ali Garavand. Artificial intelligence applications in type 2 diabetes mellitus care: focus on machine learning methods. *Healthcare informatics research*, 25(4):248–261, 2019.

[18] Vasundhara Acharya and Preetham Kumar. Identification and red blood cell automated counting from blood smear images using computer-aided system. *Medical & biological engineering & computing*, 56(3):483–489, 2018.

[19] Emma Ahlqvist, Petter Storm, Annemari Käräjämäki, Mats Martinell, Mozhgan Dorkhan, Annelie Carlsson, Petter Vikman, Rashmi B Prasad, Dina Mansour Aly, Peter Almgren, et al. Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables. *The lancet Diabetes & endocrinology*, 6(5):361–369, 2018.

[20] Tahani Alqurashi and Wenjia Wang. Clustering ensemble method. *International Journal of Machine Learning and Cybernetics*, 10(6):1227–1246, 2019.

[21] Hadeel Alzoubi, Raid Alzubi, Naeem Ramzan, Daune West, Tawfik Al-Hadhrami, and Mamoun Alazab. A review of automatic phenotyping approaches using electronic health records. *Electronics*, 8(11):1235, 2019.

[22] Ranjit Mohan Anjana, Viswanathan Baskar, Anand Thakarakkattil Narayanan Nair, Saravanan Jebarani, Moneeza Kalhan Siddiqui, Rajendra Pradeepa, Ranjit Unnikrishnan, Colin Palmer, Ewan Pearson, and Viswanathan Mohan. Novel subgroups of type 2 diabetes and their association with microvascular outcomes in an asian indian population: a data-driven cluster analysis: the inspired study. *BMJ Open Diabetes Research and Care*, 8(1):e001506, 2020.

[23] American Diabetes Association. Complications. `https://www.diabetes.org/diabetes/complications`.

[24] Hanan G Ayad and Mohamed S Kamel. On voting-based consensus of cluster ensembles. *Pattern Recognition*, 43(5):1943–1953, 2010.

[25] Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, 2011.

[26] Juan M Banda, Martin Seneviratne, Tina Hernandez-Boussard, and Nigam H Shah. Advances in electronic phenotyping: from rule-based definitions to machine learning models. *Annual review of biomedical data science*, 1:53–68, 2018.

[27] Aruna Bhat. K-medoids clustering using partitioning around medoids for performing face recognition. *International Journal of Soft Computing, Mathematics and Control*, 3(3):1–12, 2014.

[28] Amol Prataprao Bhatkar and G.U. Kharat. Detection of diabetic retinopathy in retinal images using mlp classifier. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 331–335, 2015.

[29] LV Bijuraj. Clustering and its applications. In *Proceedings of National Conference on New Horizons in IT-NCNHIT*, volume 1, pages 169–172, 2013.

[30] Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta. clvalid: An r package for cluster validation. *Journal of Statistical Software*, 25:1–22, 2008.

[31] Jason Brownlee. How to resample and interpolate your time series data with python. `https://machinelearningmastery.com/resample-interpolate-time-series-data-python/`, Feb 2020. Online accessed :23-November-2021.

[32] Derek S Chiu and Aline Talhouk. dicer: an r package for class discovery using an ensemble driven approach. *BMC bioinformatics*, 19(1):1–4, 2018.

[33] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[34] Bénédicte Clarisse, Christophe Demattei, Lydia Nikasinovic, Jocelyne Just, Jean-Pierre Daures, and Isabelle Momas. Bronchial obstructive phenotypes in the first year of life among paris birth cohort infants. *Pediatric allergy and immunology*, 20(2):126–133, 2009.

[35] Wikipedia contributors. Statistical classification, 2021. Online accessed :20-November-2021.

[36] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[37] Delbert Dueck, Brendan J Frey, Nebojsa Jojic, Vladimir Jojic, Guri Giaever, Andrew Emili, Gabe Musso, and Robert Hegele. Constructing treatment portfolios using affinity propagation. In *Annual International Conference on Research in Computational Molecular Biology*, pages 360–371. Springer, 2008.

[38] N Emami and A Pakzad. A new knowledge-based system for diagnosis of breast cancer by a combination of the affinity propagation and firefly algorithms. *Journal of AI and Data Mining*, 7(1):59–68, 2019.

[39] Laura Ernande, Etienne Audureau, Christine L Jellis, Cyrille Bergerot, Corneliu Henegar, Daigo Sawaki, Gabor Czibik, Chiara Volpi, Florence Canoui-Poitrine, Helene Thibault, et al. Clinical implications of echocardiographic phenotypes of patients with diabetes mellitus. *Journal of the American College of Cardiology*, 70(14):1704–1716, 2017.

[40] Javier Escudero, Emmanuel Ifeachor, John P Zajicek, Alzheimer's Disease Neuroimaging Initiative, et al. Bioprofile analysis: a new approach for the analysis of biomedical data in alzheimer's disease. *Journal of Alzheimer's Disease*, 32(4):997–1010, 2012.

[41] International Diabetes Federation. Idf diabetes atlas 9th edition 2019. `https://www.diabetesatlas.org/en/sections/worldwide-toll-of-diabetes.html`.

[42] Xiaoli Z Fern and Carla E Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 186–193, 2003.

[43] Xiaoli Z Fern and Wei Lin. Cluster ensemble selection. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 1(3):128–141, 2008.

[44] Caston Fernandes. Get rid of the dirt from your data - data cleaning techniques. `https://medium.com/@casternxavier/get-rid-of-the-dirt-from-your-data-data-cleaning-techniques-743253b1c3e5`, May 2018.

[45] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[46] Ana LN Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005.

[47] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[48] Ben D Fulcher. Feature-based time-series analysis. *arXiv preprint arXiv:1709.08055*, 2017.

[49] Hripcsak G and Albers DJ. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association : JAMIA*, 20:117–121, 2013.

[50] Vipul Gandhi. Gaussian mixture models clustering - explained. `https://www.kaggle.com/vipulgandhi/gaussian-mixture-models-clustering-explained`. Accessed: 2021-11-30.

[51] Reza Ghaemi, Md Nasir Sulaiman, Hamidah Ibrahim, Norwati Mustapha, et al. A survey: clustering ensembles techniques. *World Academy of Science, Engineering and Technology*, 50:636–645, 2009.

[52] Joydeep Ghosh and Ayan Acharya. Cluster ensembles. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 1(4):305–315, 2011.

[53] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. *Acm transactions on knowledge discovery from data (tkdd)*, 1(1):4–es, 2007.

[54] Keyvan Golalipour, Ebrahim Akbari, Seyed Saeed Hamidi, Malrey Lee, and Rasul Enayatifar. From clustering to clustering ensemble selection: A review. *Engineering Applications of Artificial Intelligence*, 104:104388, 2021.

[55] Jacques A Hagenaars and Allan L McCutcheon. *Applied latent class analysis*. Cambridge University Press, 2002.

[56] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[57] Marwan Hassani and Thomas Seidl. Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science*, 4(3):171–183, 2017.

[58] Jette Henderson, Ryan Bridges, Joyce C Ho, Byron C Wallace, and Joydeep Ghosh. Pheknow–cloud: A tool for evaluating high-throughput phenotype candidates using online medical literature. *AMIA Summits on Translational Science Proceedings*, 2017:149, 2017.

[59] Yue Huang, Paul McCullagh, Norman Black, and Roy Harper. Feature selection and classification model construction on type 2 diabetic patients' data. *Artificial intelligence in medicine*, 41(3):251–262, 2007.

[60] Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. *DMKD*, 3(8):34–39, 1997.

[61] Natthakan Iam-On, Tossapon Boongeon, Simon Garrett, and Chris Price. A link-based cluster ensemble approach for categorical data clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):413–425, 2012.

[62] Natthakan Iam-on and Simon Garrett. Linkclue: A matlab package for link-based cluster ensembles. *Journal of Statistical Software*, 36:1–36, 2010.

[63] Francisco Hernansanz Iglesias, Joan Carles Martori Cañas, Esther Limón Ramírez, Clara Alavedra Celada, and Carles Blay Pueyo. Clustering complex chronic patients: A cross-sectional community study from the general practitioner's perspective. *International Journal of Integrated Care*, 21(2), 2021.

[64] Edy Irwansyah, Ebiet Salim Pratama, and Margaretha Ohyver. Clustering of cardiovascular disease patients using data mining techniques with principal component analysis and k-medoids. 2020.

[65] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, 33(10):913–933, 2019.

[66] Jon Kleinberg. An impossibility theorem for clustering. *Advances in neural information processing systems*, pages 463–470, 2003.

[67] Pang-Ning Tan; Michael Steinbach; Anuj Karpatne; Vipin Kumar. *Introduction to Data Mining*. Pearson, 2005.

[68] Ludmila I Kuncheva and Stefan Todorov Hadjitodorov. Using diversity in cluster ensembles. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 2, pages 1214–1219. IEEE, 2004.

[69] Gayaneh Kyureghian, Oral Capps, and Rodolfo M Nayga. A missing variable imputation methodology with an empirical application. In *Missing data methods: Cross-sectional methods and applications*. Emerald Group Publishing Limited, 2011.

[70] Dingcheng Li, Gyorgy Simon, Christopher G Chute, and Jyotishman Pathak. Using association rule mining for phenotype extraction from electronic health records. *AMIA Summits on Translational Science Proceedings*, 2013:142, 2013.

[71] Tao Li, Mitsunori Ogihara, and Sheng Ma. On combining multiple clusterings. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 294–303, 2004.

[72] Minlei Liao, Yunfeng Li, Farid Kianifard, Engels Obi, and Stephen Arcona. Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis. *BMC nephrology*, 17(1):1–14, 2016.

[73] Roderick JA Little and Donald B Rubin. The analysis of social science data with missing values. *Sociological Methods & Research*, 18(2-3):292–326, 1989.

[74] Alam Mahbubul. Clustering:concepts, algorithms and applications, Dec 2020.

[75] M Malarvizhi and A Thanamani. K-nn classifier performs better than k-means clustering in missing value imputation. *IOSR Journal of Computer Engineering*, 6(5):12–15, 2012.

[76] Md Maniruzzaman, Md Jahanur Rahman, Benojir Ahammed, and Md Menhazul Abedin. Classification and prediction of diabetes disease using machine learning paradigm. *Health information science and systems*, 8(1):1–14, 2020.

[77] Farrokh Sohrabi Marie Suszynski. 8 important diabetes tests to get regularly. `https://www.everydayhealth.com/hs/type-2-diabetes-live-better-guide/important-diabetes-tests/`, urldate = 2013-05-16, 2015.

[78] Allan L McCutcheon. *Latent class analysis*. Number 64. Sage, 1987.

[79] Behrouz Minaei-Bidgoli, Alexander Topchy, and William F Punch. Ensembles of partitions via data resampling. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, volume 2, pages 188–192. IEEE, 2004.

[80] Geert Molenberghs, Herbert Thijs, Ivy Jansen, Caroline Beunckens, Michael G. Kenward, Craig Mallinckrodt, and Raymond J. Carroll. Analyzing incomplete longitudinal clinical trial data. *Biostatistics*, 5(3):445–464, 07 2004.

[81] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118, 2003.

[82] Abu Saleh Mohammad Mosa, Chalermpon Thongmotai, Humayera Islam, Tanmoy Paul, K. S. M. Tozammel Hossain, and Vasanthi Mandhadi. Evaluation of machine learning applications using real-world ehr data for predicting diabetes-related long-term complications. *Journal of Business Analytics*, 0(0):1–11, 2021.

[83] Boland MR, Hripcsak G, Shen Y, Chung WK, and Weng C. Defining a comprehensive verotype using electronic health records for personalized medicine. *Journal of the American Medical Informatics Association : JAMIA*, 20(e2):e232–e238, 2013.

[84] Nam Nguyen and Rich Caruana. Consensus clusterings. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 607–612. IEEE, 2007.

[85] Caterina Penone, Ana D Davidson, Kevin T Shoemaker, Moreno Di Marco, Carlo Rondinini, Thomas M Brooks, Bruce E Young, Catherine H Graham, and Gabriel C Costa. Imputation of missing data in life-history trait datasets: Which approach performs the best? *Methods in Ecology and Evolution*, 5(9):961–970, 2014.

[86] Sajida Perveen, Muhammad Shahbaz, Aziz Guergachi, and Karim Keshavjee. Performance analysis of data mining classification techniques to predict diabetes. *Procedia Computer Science*, 82:115–121, 2016. 4th Symposium on Data Mining Applications, SDMA2016, 30 March 2016, Riyadh, Saudi Arabia.

[87] G. A. Pethunachiyar. Classification of diabetes patients using kernel based support vector machines. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–4, 2020.

[88] Anindya Apriliyanti Pravitasari, Nur Iriawan, Siti Azizah Nurul Solichah, Kartika Fithriasari, Santi Wulan Purnami, Widiana Ferriastuti, et al. Gaussian mixture model for mri image segmentation to build a three-dimensional image on brain tumor area. *Matematika*, 36, 2020.

[89] LKPJ Rdusseeun and P Kaufman. Clustering by means of medoids. In *Proceedings of the Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, Switzerland*, pages 405–416, 1987.

[90] Gold S Rasmussen L Richesson R, Wiley LK. Electronic health records–based phenotyping, Dec 2021.

[91] David Robinsoni. K-means clustering is not a free lunch. `http://varianceexplained.org/r/kmeans-free-lunch/`, urldate = 2021-10-11, January 2015.

[92] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[93] Zimi Sawacha, Gabriella Guarneri, Angelo Avogaro, and Claudio Cobelli. A new classification of diabetic gait pattern based on cluster analysis of biomechanical data, 2010.

[94] Joseph L Schafer and John W Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[95] James B Schreiber and Andrew J Pekarik. Using latent class analysis versus k-means or hierarchical clustering to understand museum visitors. *Curator: The Museum Journal*, 57(1):45–59, 2014.

[96] Yasin Senbabaoğlu, George Michailidis, and Jun Z Li. Critical limitations of consensus clustering in class discovery. *Scientific reports*, 4(1):1–13, 2014.

[97] Jess J Shen, Phil Hyoun Lee, Jeanette JA Holden, and Hagit Shatkay. Using cluster ensemble and validation to identify subtypes of pervasive developmental disorders. In *AMIA Annual Symposium Proceedings*, volume 2007, page 666. American Medical Informatics Association, 2007.

[98] Chaitanya Shivade, Preethi Raghavan, Eric Fosler-Lussier, Peter J Embi, Noemie Elhadad, Stephen B Johnson, and Albert M Lai. A review of approaches to identifying patient phenotype cohorts using electronic health records. *Journal of the American Medical Informatics Association*, 21(2):221–230, 2014.

[99] Valérie Siroux, Xavier Basagaña, Anne Boudier, Isabelle Pin, Judith Garcia-Aymerich, Aurélien Vesin, Rémy Slama, Déborah Jarvis, Josep M Anto, Francine Kauffmann, et al. Identifying adult asthma phenotypes using a clustering approach. *European Respiratory Journal*, 38(2):310–317, 2011.

[100] Deepti Sisodia and Dilip Singh Sisodia. Prediction of diabetes using classification algorithms. *Procedia Computer Science*, 132:1578–1585, 2018. International Conference on Computational Intelligence and Data Science.

[101] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[102] Alexander Topchy, Anil K Jain, and William Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1866–1881, 2005.

[103] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[104] Shahadat Uddin, Arif Khan, Md Ekramul Hossain, and Mohammad Ali Moni. Comparing different supervised machine learning algorithms for disease prediction. *BMC medical informatics and decision making*, 19(1):1–16, 2019.

[105] Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

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

[107] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(03):337–372, 2011.

[108] S Vijayarani and S Sudha. An efficient clustering algorithm for predicting diseases from hemogram blood test samples. *Indian Journal of Science and Technology*, 8(17):1, 2015.

[109] Kushal Virupakshappa and Erdal Oruklu. Unsupervised machine learning for ultrasonic flaw detection using gaussian mixture modeling, k-means clustering and mean shift clustering. In *2019 IEEE International Ultrasonics Symposium (IUS)*, pages 647–649, 2019.

[110] Xi Wang, Chunyu Yang, and Jie Zhou. Clustering aggregation by probability accumulation. *Pattern Recognition*, 42(5):668–675, 2009.

[111] Zhiqiang Wang, Catherine Da Cunha, Mathieu Ritou, and Benoît Furet. Comparison of k-means and gmm methods for contextual clustering in hsm. *Procedia Manufacturing*, 28:154–159, 2019.

[112] Anthony Waruru, Thomas NO Achia, James L Tobias, et al. Finding hidden hiv clusters to support geographic-oriented hiv interventions in kenya. *Journal of acquired immune deficiency syndromes (1999)*, 78(2):144, 2018.

[113] Jong Chul Won, Yong-Jin Im, Ji-Hyun Lee, Chong Hwa Kim, Hyuk Sang Kwon, Bong-Yun Cha, and Tae Sun Park. Clinical phenotype of diabetic peripheral neuropathy and relation to symptom patterns: cluster and factor analysis in patients with type 2 diabetes in korea. *Journal of diabetes research*, 2017, 2017.

[114] Adam Wright, Allison B McCoy, Stanislav Henkin, Abhivyakti Kale, and Dean F Sittig. Use of a support vector machine for categorizing free-text notes: assessment of accuracy across two institutions. *Journal of the American Medical Informatics Association*, 20(5):887–890, 2013.

[115] Junjie Wu. Cluster analysis and k-means clustering: an introduction. In *Advances in K-means Clustering*, pages 1–16. Springer, 2012.

[116] Jiang Zhang, Dahuan Li, Huafu Chen, and Fang Fang. Analysis of activity in fmri data using affinity propagation clustering. *Computer methods in biomechanics and biomedical engineering*, 14(03):271–281, 2011.

[117] Shang-Ming Zhou, Fabiola Fernandez-Gutierrez, Jonathan Kennedy, Roxanne Cooksey, Mark Atkinson, Spiros Denaxas, Stefan Siebert, William G Dixon, Terence W O'Neill, Ernest Choy, et al. Defining disease phenotypes in primary care electronic health records by a machine learning approach: a case study in identifying rheumatoid arthritis. *PloS one*, 11(5):e0154515, 2016.