DETERMINATION OF FUNCTIONAL DEPENDENCIES IN A RELATIONAL DATABASE USING CLUSTERING AND FUZZY RELATIONS

A Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Electronics and Telecommunication Engineering

Jadavpur University

June 2022

By

Arnab Jana

Examination Roll No.: M4ETC22013

Registration No.: 154082 of 2020-2021

Under the Guidance of

Prof. Amit Konar

Department of Electronics and Telecommunication Engineering

Jadavpur University

Kolkata-700032, India

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE

This is to certify that the dissertation entitled, **"Determination of Functional Dependencies in a Relational Database Using Clustering and Fuzzy Relations"** has been carried out by ARNAB JANA (University Registration No.: 154082 of 2020-2021) under my guidance and supervision and be accepted in partial fulfillment of the requirements for the degree of Master of Electronics and Telecommunication Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other university or institute.

Prof. Amit Konar
Thesis Supervisor
Dept. of Electronics and Telecommunication Engineering
Jadavpur University

Prof. Ananda Shankar Chowdhury
Head of the Department
Dept. of Electronics and Telecommunication Engineering
Jadavpur University

Prof. Chandan Mazumdar
Dean
Faculty of Engineering and Technology
Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The forgoing thesis is hereby approved as a creditable study of an engineering subject and presented in a manner satisfactory to warrant acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn there in but approves the thesis only for which it is submitted.

Committee on final examination for the evaluation of the thesis:

____

Signature of the Examiner

____

Signature of the Supervisor

*Only in the case the thesis is approved.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis entitled **“Determination of Functional Dependencies in a Relational Database Using Clustering and Fuzzy Relations”** contains references to the previous works and original research work by the undersigned candidate as part of his degree of Master of Electronics and Telecommunication Engineering. All information has been obtained and presented in accordance with academic rules and ethical conduct. It is hereby declared that, as required by these rules and conduct, all materials and results that are not original to this work have been properly cited and referenced.

Candidate Name: **Arnab Jana**

Examination Roll No.: M4ETC22013

Thesis Title: **Determination of Functional Dependencies in a Relational Database Using Clustering and Fuzzy Relations**

____

Signature of the Student

ACKNOWLEDGEMENT

I express my sincere gratitude to our college Jadavpur University for providing me an opportunity to undertake and complete such an interesting thesis entitled **"Determination of Functional Dependencies in a Relational Database Using Clustering and Fuzzy Relations"**.

I am very thankful to my guide Prof. Amit Konar for the confidence he has in me regarding the thesis and for providing me with valuable suggestions, motivation, and ideas during the whole thesis work. I am very much obliged to our respected Head of the Department (Electronics and Telecommunication Engineering) Prof. Ananda Shankar Chowdhury, for inspiring and motivating me to bring out a successful thesis.

I am also very thankful to the whole department of Electronics and Telecommunication Engineering, for providing me the technical support for carrying out the thesis work to let me utilize the necessary facilities of the institute and guidance at each step during the thesis work.

I would also like to take this opportunity to pay my thanks to all my classmates who motivated me to complete my research work successfully. I do wish to thank all our departmental support staff and all of those who were associated with this research work and contributed in some form or others.

Finally, I am immensely indebted to my parents, family, and friends for their continuous support and encouragement that made me believe in myself and provided me with the strength to work hard.

ARNAB JANA

(Dept. of Electronics and

Telecommunication Engineering)

Examination Roll Number: M4ETC22013

Registration Number: 154082 of 2020-2021

Place: Kolkata
Date:

PREFACE

I present my thesis on the topic **“Determination of Functional Dependencies in a Relational Database Using Clustering and Fuzzy Relations”**.

The report is based on fuzzy logic, its application in computer science and the real world, how it can be used to find the relationship between different features/characteristics in a relation, and its future scope. The document elicits the working and functions of a Fuzzy Logic based system lucidly and understandably. Relational database stores data in the form of tables/relations. Each table represents a relationship between different features present in the table and each row/tuple is a distinct entry in the relation. A relational database is a popular model in the scientific and research community to store information. The relations are used in Artificial Intelligence, Machine Learning applications, in the sector like banking, healthcare, education, manufacturing, etc. Finding functional dependencies present in the data helps in cleaning data, building constraints for data entry, removing redundancies, and helps in preserving the integrity and quality of data.

Functional dependency between two sets of attributes X and Y (denoted by $X \rightarrow Y$) supplies the information that sets of attribute X can uniquely identify sets of attribute Y. In the fundamental concept of functional dependency (FD) exploration, the constraint is too rigid, where a single tuple violating the rule completely discards the dependency between X and Y. In the real world the data collected is sometimes inconsistent, spurious and sometimes data is missing, these types of problems lead to lost in functional dependency and incorporation of false dependencies. Also, we must keep in mind that we find functional dependency only on the data present in the relation and may not convey holistic information. In these situations, the application of fuzzy logic-based relationship strength discovery coupled with the clustering technique can help find some interesting dependencies and exhaustive experimentation may lead to the establishment of a whole new tool in the scope of relational databases.

List of Figures

List of Tables

List of Abbreviations

Full Form	Short Form
FD	Functional Dependency
AI	Artificial Intelligence
ML	Machine Learning
DBMS	Database Management System
SQL	Structured Query Language
DBA	Database Administrators
DDL	Database Definition Language
DML	Data Manipulation Language
WCSS	Within Cluster Sum of Squares
RMSE	Root Mean Square Error
FD-CFR	Functional dependency by clustering and Fuzzy Relations

Contents

Chapter 1

Introduction

We are living in the ocean of data. With the development of modern-day electronics and sophisticated software and algorithms, devices are becoming smart. It is said, by the year 2025, 463 Exabytes of data will be created each day.

With so much data comes endless possibilities. Artificial intelligence, machine learning, and Deep Learning can take the help of high computationally effective hardware to bring meaningful insights into these data. The applications range from medical, sports, and entertainment, to scientific explorations, etc.

In this regard what is more important is to find the right relationship and correct meaning to data, as correct finding from corrupt data is as dangerous as wrong insights from good data.

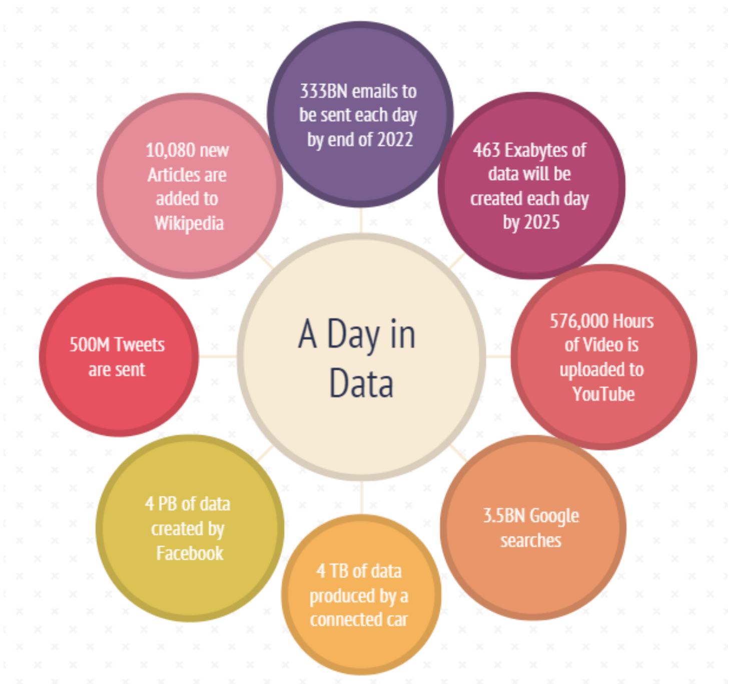

333BN emails to be sent each day by end of 2022
10,080 new Articles are added to Wikipedia
463 Exabytes of data will be created each day by 2025
500M Tweets are sent
A Day in Data
576,000 Hours of Video is uploaded to YouTube
4 PB of data created by Facebook
3.5BN Google searches
4 TB of data produced by a connected car

Figure 1.1: Amount of data generated each day

The last figure shows the amount of data generated on various platforms per day and the estimated increase in the amount of data produced daily as forecasted by various researchers.

These huge data are stored on various servers around the world in data centers and data warehouses. The databases contain these data in the form of tables/relations. The relations between different columns/attributes are called functional dependency (FD). Functional dependency was introduced by E. F. Codd, it helps in preventing data redundancy and gets to know about bad designs. Discovering FDs from a relation gives the database users, researchers, and engineers the opportunity to find attributes that can find other attributes from the relation, which are functionally dependent on them.

1.1 Motivation

Database management has become increasingly important in scientific and commercial applications including banking transactions, digital libraries, electronic commerce, space research, airline, telecommunication, supply chain management, and manufacturing. There are several data models used to store this information, and among all these data models, the relational data model is widely used.

The intent of relational database design is to ensure the relational schema should not include repeated entries, i.e., redundancy should be eliminated.

In today's digital age, data is the most precious asset that an association can hold. In a typical business association, data is generated from multiple streams, and it has become essential to extract meaningful information from the raw data collected from these multiple data sources to set the stage for business analysis and decision-making processes. When a database is modeled, all dependency constraints are not imposed on the database, as it is not possible to find the complete set of dependencies from the relations during the design phase. It is important to find dependency constraints from database tables for data management and database reverse engineering. Mining functional dependencies from relation is a tedious and complex task. Uncovering all functional dependencies from the data needs highly skilled domain experts as some dependencies are hidden in the data and not found intuitively. In practice all relations are not complete, they only include entries that are obtained from the users and during business operations, which may be incomplete or erroneous, or even missing entries can be there.

FD discovery is only done on the existing data and does not look at actual relationships between the attributes; it needs an explicit definition of dependencies by experts, which becomes unfeasible as the number of attributes and rows/tuples increases in a relation. The presence of a single exception in the equality of attribute values violates the dependency between the attributes. But indeed, if the number of exceptions is not very high, such functional dependencies with exceptions may represent some interesting patterns hidden in data. The application of fuzzy logic which works with multiple grades of truth values can help in finding those dependencies which otherwise would be overlooked if tested using the all or none concept. This thesis tries to discover this area of functional dependencies where the application of fuzzy logic using implication relations can find relationships/ functional dependencies with various levels of strengths.

1.2 Background Concept of Fuzzy Logic

Father of fuzzy logic Prof. Lotfi A. Zadeh published his famous research paper on fuzzy sets in 1965 and after that, the applications of fuzzy logic or the notion of biasness and uncertainty have been used in vast applications like weather forecasting systems, models for new product pricing or project risk assessment, medical diagnosis, washing machines, air conditioning systems, law enforcement, and the list goes on.

Conventional (Boolean) logic only works with two extreme values of truth, true (1) or false (0) whereas fuzzy logic has truth values ranging between 0 and 1. A fuzzy variable allows partial truth i.e. truth values between completely true and completely false.

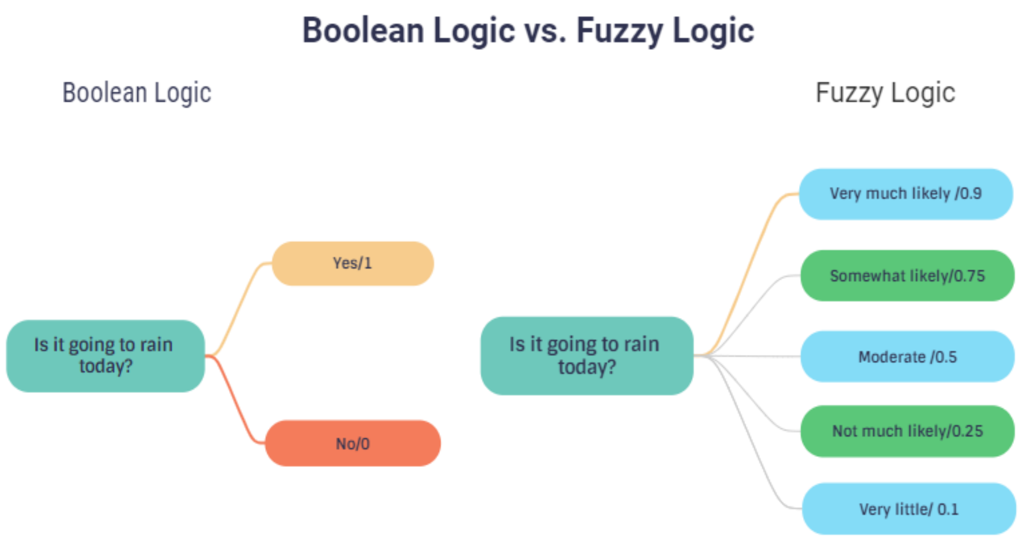


Figure 1.2: Boolean Logic vs. Fuzzy logic structure

The world does not work on just “yes” or “no”. The possibility of rain today may not always be given with 100% certainty, but one can say by looking at the sky that there is a 60% chance of rain today and a 40% chance of no rain.

Before we start with Fuzzy logic, we must understand the conventional logic /crisp logic.

1.2.1 Crisp Logic

Mathematicians define a **set** as a collection of objects having one more common characteristic. The objects that belong to a particular **set** are called members/elements of that set. The conditions or rules used to define a set are sufficient to identify its members.

For examples, if we want to find the set of students in class 9 having height greater than 5 feet may be defined as

$$students\ with\ height\ greater\ than\ 5\ feet, S = \{Amit, Anik, Ram, Jatin, Arjun\}$$

Here, the names given inside curly braces are the members of the set (students with height greater than 5 feet in a class of 9th standard).
Let A be a set and x be a member of set A, then this membership is denoted by

$$x \in A$$

Also let y be a member of set B, such that for y, y is also a member of set A. Then B is called a subset of A and it is denoted by

$$B \subseteq A$$

In a conventional set, the condition defining the set boundaries is very rigid. For example, consider we define a universal set TEMPERATURE, COLD, MODERATE, AND HOT are subsets of the universal set TEMPERATURE.

$$COLD = \{temperature \in TEMPERATURE: 0℃ \leq temperature < 25℃\}$$

$$MODERATE = \{temerature \in TEMPERATURE: 25℃ \leq temperature < 60℃\}$$

$$HOT = \{temperature \in TEMPERATURE : 60℃ \leq temperature \leq 100℃\}$$

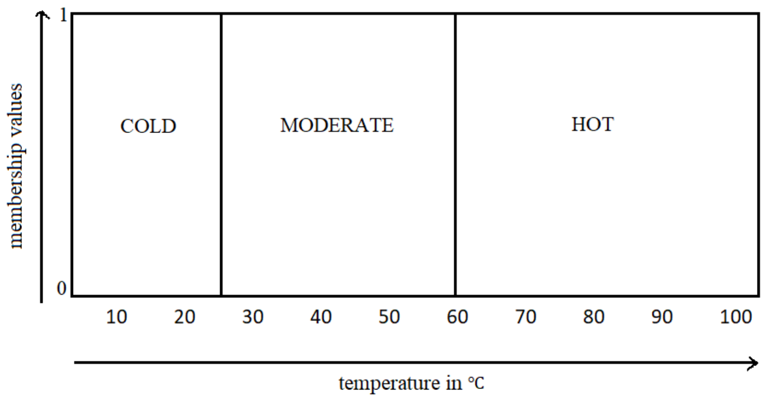


Figure 1.3: Crisp boundary demarcation in classical set

In the above definitions **temperature** is a variable that presents the temperature of the water and can take any value between [0,100]. As we can see from the above set definitions that the margins separating the sets are very rigid. A temperature of 24.9 falls under the **COLD** set but once the temperature goes to 25 it falls under the set **MODERATE**.

1.2.2 Fuzzy Logic

For a conventional/classical set, the membership value of an element is either 0 or 1, i.e., either the element fully belongs to that set or does not belong at all. The following connotation is used to describe that the membership of an element x in a set A is 1, and the membership of non-element y in set A is 0.

$$\mu_A(x) = 1$$

$$\mu_A(y) = 0$$

A fuzzy set extends the binary membership: $\{0,1\}$ of a conventional set to a spectrum in the interval of [0,1]. Further, unlike a conventional set, all elements of the universal set U are members of a given set A. Thus, for each element $x \in U$,

$$0 \leq \mu_A(x) \leq 1$$

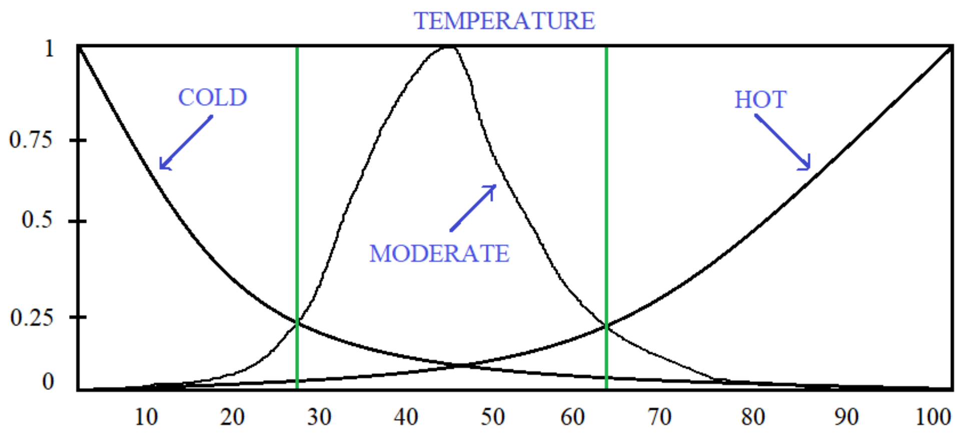


Figure 1.4: Smooth boundary demarcation in fuzzy set

A **fuzzy set** A is a set of ordered pairs, given by

$$A = \{(\, x, \mu_A(x)\,) : x \in X\}$$

Where X is a universal set of objects (also known as the universe of discourse) and $\mu_A(x)$ is the grade of membership of the object x in A. Usually, $\mu_A(x)$ lies in the closed interval of [0, 1].

1.2.3 Membership Function

The grade of membership maps the object or its attribute x to positive real numbers in the interval [0,1]. Because of its mapping characteristics like a function, it is called a **membership function**. A membership function is characterized by the following mapping:

$$\mu_A\colon\ x \to [0, 1], x\ \in X$$

Where x is a real number describing an object or its attribute and X is the universe of discourse and A is a subset of X. This membership function can take various forms to describe the grade of belongingness or membership of a particular member of the fuzzy set. Various membership functions used are L-function, S-function, γ-function, triangular, gaussian membership function, etc.

Fuzzy logic has been successfully used in many fields such as control systems engineering, image processing, robotics, power engineering, consumer electronics, industrial automation, and optimization.

1.3 Applications of Fuzzy Logic

In conventional logic or conventional sets, the truth value is either **0** or **1.**

$$x \in X \mid \mu(x) = \{0,1\}, where\ \mu(x) = membership\ values\ of\ x\ in\ set\ X$$

Fuzzy logic regards 0 and 1 as two extreme values of truth and includes all intermediate values between 0 and 1 as different degrees of truth. As a result of the incorporation of uncertainty and imprecision, it is applicable in various fields as:

1. Complex decision making where the boundary between different resolutions is not concrete such as natural language processing.
2. Controlling and regulating the output of control systems and machines where the number of variables is very high and random. Example: air conditioning systems.

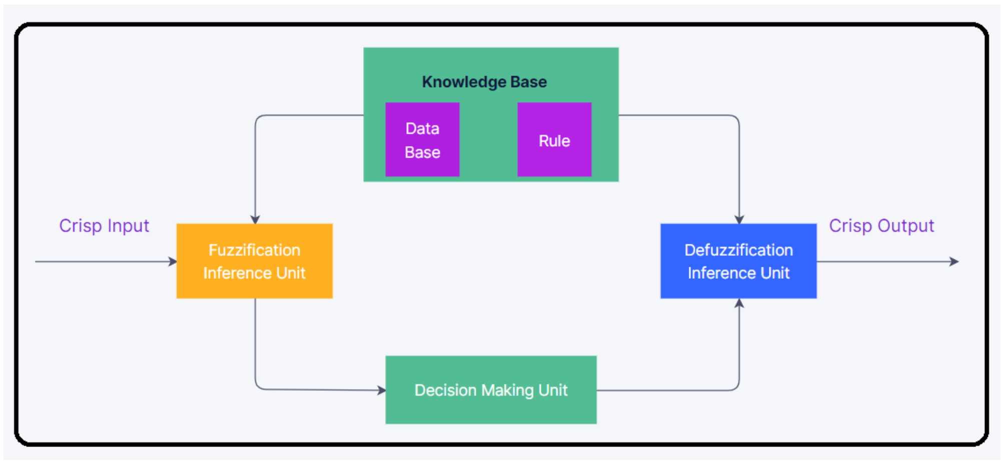


Figure 1.5: Schematic framework of a fuzzy inference system

Applications of Fuzzy logic can be seen in problems of software optimization, hardware fault/error detection, aerospace, civil, and electronics.

Here are some examples of Fuzzy-Logic based systems in a real-world scenario:

1. **Washing Machine:** Washing machines are an excellent example of a fuzzy system. There are so many constraints for a perfect wash or a good wash that a washing machine without a fuzzy inference system is kind of paralyzed. The load i.e., the amount of clothes determines the correct amount of water and detergent, speed of rotation, and duration of the wash cycles. There is no single

single standard for a good wash. Fuzzy logic enables the machine's computer to make "in-between" decisions. At the beginning of the wash load, the water is clear. As it gets dirtier, the onboard computer senses the discoloration that doesn't allow light to pass through as easily. The sensors use fuzzy logic to control settings, so you get a clean load.

2. **Law Enforcement:** The world is not black and white. There are degrees of truth lying between 0 and 1. Law enforcement software leverages the power of fuzzy logic to narrow down possible suspects in a database. To detect a possible suspect, we need high precision as the cost of false positives is very high here, we cannot let an innocent man receive punishment. This precision comes from a series of complex decisions.
3. **Air Conditioning:** Modern air conditioning systems use fuzzy logic and as a result, they are more efficient and highly effective. The old air conditioners used to implement binary logic of two extremes i.e., maximum and minimum, whenever the temperature reaches maximum turn on the AC, and whenever it reaches minimum turn off the AC. Modern air conditioners take into account temperatures, humidity, and other factors and sense slight fluctuations, and takes necessary adjustments.
4. **Pattern Recognition:** Fuzzy logic gives the human-like decision-making system and it has the advantage of low computational overhead. For different tasks in pattern recognition like image processing, audio pattern recognition, and facial pattern recognition fuzzy logic is used.
5. **Electric Vehicles:** In electric vehicles, the fuzzy logic controller is used to utilize the regenerative braking energy of the vehicle to control speed.
6. **Natural Language Processing:** Fuzzy logic is used in natural language processing to model the semantics of certain linguistic expressions as their inherent obscurity can be captured by fuzzy logic.
7. **Other applications involve:**

- In dishwashers, fuzzy logic is used to determine the washing strategy and power needed, which is based on factors such as the number of dishes and the level of food residue on the dishes.
- In copy machines, fuzzy logic is used to adjust drum voltage based on factors such as humidity, picture density, and temperature.

- In aerospace, fuzzy logic is used to manage altitude control for satellites and spacecrafts, based on environmental factors.
- In medicine, fuzzy logic is used for computer-aided diagnoses, based on factors such as symptoms and medical history.
- In chemical distillation, fuzzy logic is used to control pH and temperature variables.
- In a business rules engine, fuzzy logic may be used to streamline decision-making according to predetermined criteria.

1.4 Objective

Functional dependencies (FDs) between different attributes in a database relation are the relationships between these attributes. It states that the value of an attribute can be uniquely determined by another attribute (or attributes). Finding FDs from a relation has many applications from database design to knowledge discovery. For example, in an e-commerce database, the phone number can determine the customer's name, address, email, and other attributes.

Formally, a functional dependency over relational schema R, with X and Y as subsets of R is given by $X \rightarrow Y$ if, for any two tuples t1 and t2 in R, whenever t1[X] = t2[X] then t1[Y] = t2[Y]. In the case of relations where a large number of attributes are numerical in nature it is seen that there exists some meaningful FDs, which represent invaluable information, but if checked for validation of the above rule on all pairs of tuples, remains undetected, and results in loss of information. The discovery of these meaningful or logical and hidden dependencies is an interesting task in the world of data mining. These are approximate dependencies that almost hold perfectly in the relation. Such dependencies are inherent in the relation, but some tuples represent exceptions to the rule or contain errors. Finding these close or approximate FDs have huge application in research and database design.

Our objective is to apply fuzzy logic with the help of the clustering technique to not only find the FDs that hold according to the formal definition of FD but also those approximate FDs that can bring meaningful insights. We will also try to reduce the search space for FDs in relations to restricting the area of discovery by eliminating searching for trivial and non-normalized FDs. A trivial dependency is one where the $X \cap Y \neq \Phi$ (X= set of determinant attributes and Y = set of dependent attributes) and a normalized dependency is where the dependent set consists of a single attribute. This helps us to skip a lot of FD checks as they can easily be found from these trivial and normalized FDs by applying Armstrong’s axioms and secondary rules derived from them.

1.5 Organization of the Thesis

The thesis tries to summarize all the necessary information related to the project in a fluid and organized manner. The necessary concepts required to understand the motivation behind the work, tools, techniques, and concepts required for the experiments, are all incorporated for the ease of the reader. The rest of the thesis is organized as follows:

Chapter 1 introduces the basic concepts of Fuzzy Logic-based application along with the objective and motivation. Also, the work plan of our decision-making system has been described in this section. In chapter 2, the standard tools and techniques are discussed. This includes the statistical measures used in this work, the concept of data clustering, and fuzzy reasoning. Database Management System is studied in chapter 3. Chapter 4 is dedicated to the logic and implementation procedure of the proposed approach (FD-CFR). The work done using the proposed algorithm and the results obtained are given in chapter 5. It includes the different tests done and a comparative study. In chapter 6 the conclusions are drawn, and the future scopes are enlisted. Finally all the references to the works studied for the concept and inspiration for this work is given in the References. The previous works in the fields of fuzzy logic, relational databases, and functional dependencies are properly cited. The appendix contains the pseudocode of the proposed algorithm, and some outcomes from the code executions to show the hands-on experiments done.

Chapter 2

Standard Tools and Techniques

This chapter provides detailed descriptions of the standard tools and techniques required for carrying out the work described in the thesis. This section also gives the required concept for going through this thesis. We, first start with the statistical measures like mean, and standard deviation and then venture into the data clustering techniques and fuzzy logic-based relationship-building approach. Section 2.1 gives the statistical concepts of data aggregation and transformation operations. Section 2.2 provides the knowledge required for clustering data in relation to the efficient finding of the relationship present in the data and the techniques required for more efficient and simple clustering. Finally, section 2.3 presents the fuzzy reasoning mechanism for functional dependency discovery, which includes relational matrix building, fuzzy composition rules, and various fuzzy implication relations present in the literature.

2.1 Statistical Measures

In statistics the words **sample** and **population** have different meanings, sample is a small portion of the complete data (population) chosen to show the kind of data present in the original data, i.e., a sample is a representative of the population. Though sometimes population and sample are used interchangeably, measures like standard deviation and variance have different formulas for sample and population.

2.1.1 Mean

In statistics, the mean of a population is one of the three central tendencies, used to find one aggregate information or representative value to describe the data. It is sometimes also written as the average of the data. The mean or average is found by calculating the sum of all values of a particular representative variable and then dividing that sum by the number of observations. If we want to find the mean of a data represented by X, then its mean is written as $\bar{X}$.

Let, for a sample of data there are n observations present and they are $(X_1, X_2, X_3, X_4, \dots, X_n)$ observations, then the mean can be defined as,

$$\bar{X} = (X_1, X_2, X_3, X_4, \ldots, X_n)/n$$

$$= \frac{1}{n} \sum_{i=1}^{n} X_i \tag{2.1}$$

2.1.2 Standard Deviation

Standard deviation is a measure that is used to find the distribution of data points in a population. It gives the dispersion of data points present in the population. It shows how much the data is distributed around the mean, and gives the spread or deviation from the population mean. A small deviation means most of the data points are close to the **mean** and a high deviation means the data points are more spread out from the mean.

The standard deviation of the population is calculated using the population mean (µ). For a population of X if there are N data points present, then the standard deviation of the population denoted by σ is given by

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (X_i - \mu)^2} \tag{2.2}$$

Where, σ = Population standard deviation

N = Number of data points in the population

μ = Mean of the population

X_i= ith data point in the population

2.1.3 Min-Max Normalization

The term normalization means to bound the data values for a population in a regular interval like (0,1), (-1,1).

Normalization of features/attributes of a dataset or table is extensively used in the domain of AI/ML. As the features can be representative of different categories, a dataset of **houses with their price** can contain features like distance from the nearby train station (in KM), the number of bedrooms (1,2, etc.), floor area (in square feet), price (in INR); it is clearly visible that the values for these attributes range on different scales. If normalization is not used, then a feature with a large scale can highly dominate over other attributes with a small scale. This problem is overcome using normalization, which brings all data in different attributes on a similar scale giving all features equal opportunity to influence the required result.

Min-max normalization is one of the many normalization techniques used in the field of research and application. In the min-max normalization operation, the minimum value of the feature gets assigned to the value 0 and the maximum value of the feature gets assigned to 1. All other values lying in between the minimum and maximum values of that feature get assigned a value between 0 and 1; the more the value is near to minimum the more it will be near zero and the more it is near to maximum the more it will be near to 1 after transformation.

For a feature having values $x \in X$, the normalized values of x after the min-max transformation is given by,

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{2.3}$$

Where, x_{norm} = transformed value of x in [0,1]

min (x) = minimum value of x for the attribute

max (x) = maximum value of x for the attribute

2.1.4 Root Mean Square Error

Root-mean-square error (RMSE) also referred to as root-mean-square deviation (RMSD) is a statistical metric often used in experiments to find how well an estimated value is near to the observed or actual value. RMSE is always a non-negative value and the smaller the value is, the better the outcome of our experiment. The ideal value of RMSE is 0 which is obtained seldom. It gives the quadratic mean of the differences between the estimated values and already given data i.e., it gives the square root of the second sample moment of these differences. These errors between estimated values and observed values are called as **residuals** and RMSE represents the standard deviation of these residuals.

The RMSE helps us to aggregate the magnitudes of the errors in our estimation for various data points and presents us with a single measure of predictive power. Thus, we get a single value to analyze the result. RMSE is a measure of accuracy, to compare prediction errors of different models for a particular dataset and should be used for comparison within a dataset and not between datasets. Though it has many advantages as a metric for error calculation, the downside is that RMSE is **proportional to the size of squared error**, as a result, a larger error has a disproportionate effect on RMSE calculation.

If we have sample of N data points, then the formula for RMSE is given by

$$RMSE = \sqrt{\frac{\sum_{i=1}^{1}(x_i - \widehat{x_\iota})^2}{N}} \tag{2.4}$$

Where, N = number of datapoints in sample

x_i= ith actual data point

$\widehat{x_\iota}$= ith estimated sample

2.2 Data Clustering

Clustering data or objects means grouping similar data or objects together that bear a similar characteristic. It is a popular term in the world of classification problems where we want to group similar data by exploring different attributes of those data. The application lies to classify an unknown data/object correctly into one of the classes/clusters so that appropriate actions can be taken for that data/object. An example can be to classify a tumor as benign or malignant in a patient.

2.2.1 Feature Selection

Feature selection is a method of selecting a subset of features from the dataset for different decision-making tasks which may not include all the features present in the dataset. In the field of AI/ML, this method is commonly used to reduce the dimension of input variables. Feature selection is an important part of any model training as reducing the number of input variables makes the model training phase fast and simple and sometimes saves us from model overfitting.

Sometimes **feature selection** is confused with **feature extraction**. Though the objective of both feature engineering techniques is the same i.e., to select the important and relevant features for model building, feature selection takes a subset of features from the original feature set and feature extraction creates new features from the original feature set. So, it is important to keep in mind that feature selection takes a subset of features from the original feature set by either including the important features or excluding the irrelevant features in the dataset without changing them.

A common method for feature selection is to use the correlation coefficient when the dependent and determinant features are numerical. **Pearson's correlation coefficient** is one of those correlation coefficients. It is used to find highly correlated columns and then take only necessary columns for model building, thereby reducing the number of features from the input side.

2.2.2 K-Means Clustering

K-Means clustering is an unsupervised learning algorithm used to group similar data into different groups or clusters. This divides the whole dataset into K clusters. It clusters the data into different groups and helps us to find the characteristics of these groups easily thereby revealing some patterns in the data. K-Means clustering is an iterative algorithm that divides the whole dataset into k-groups in such a way that one data only belongs to a single cluster and share identical properties with data falling into the same cluster.

As the name suggests k-means work with the mean or the centroid of the k-clusters. The target of this algorithm is to minimize the distance of each data point from its cluster centroid, thereby creating groups where points are tightly bound together. It starts with randomly dividing the dataset into k-clusters, then recomputing the cluster centroids and reassigning the points to their nearby clusters.

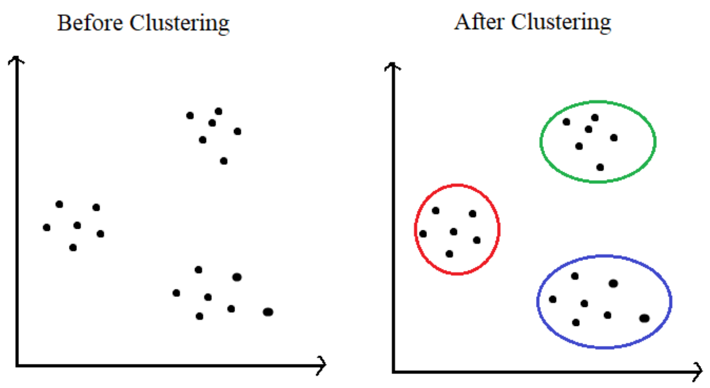


Figure 2.1: K-Means Clustering

The algorithm can be described in two steps:

1. Determine the best value for the number of clusters i.e., the value of K, and assign each data point to its nearest centroid. The k-centroids in the first step are chosen randomly or by following some rules.
2. After the first round of clustering, calculate the cluster centroids again from these newly assigned clusters and recalculate the distance of each data from k-centroids and reassign them to their nearby clusters. This process repeats until no better clusters are found i.e., there is no change in cluster assignment for all data.

2.2.3 Optimal Value of K for K-Means

K-Means clustering algorithm is used to group similar data together and thus find patterns in the data provided in a dataset. It is an unsupervised learning algorithm as we do not have any target variable for calculation. We try to cluster data into K-groups and hence find hidden information or characteristics in the data. It is used in classification problems like customer segmentation in a bank, classifying an email to be spam or not spam, and insurance fraud detection.

The important task in K-Means clustering is to select the best value of K for the dataset so that the squared distances of data points from its cluster centroids can be minimized. There are a few methods to select the number of clusters, two such methods are given here.

1. Elbow Method:

The Elbow method works with the metric **Within-Cluster-Sum-of-Squares (WCSS)** Error. We calculate WCSS error for a range of K (e.g., 1 to 10) on the dataset and then check for which value of K this error makes a significant drop, and then the fall becomes negligible for higher values of K.

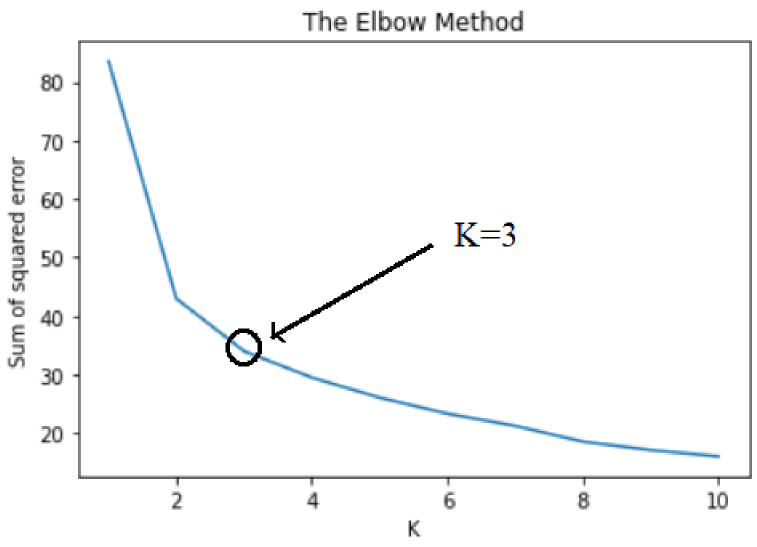


Figure 2.2: Sum of squared error vs. K plot

When the plot of WCSS Error vs. K is shown we find an elbow-like shape in the plot, the point where the graph makes this elbow gives the optimal value of K. WCSS error is the sum of the square of the distance of all data points from its cluster centroid. The distance can be calculated using many distance calculation metrics like Euclidean distance or Manhattan distance and the choice of distance measuring parameter effect in the selection of K.

Sometimes the elbow in the plot is not clearly visible, and we face confusions about which K-value to select, to resolve those cases we use another method known as the **Silhouette method** as a validation for the optimal value of K.

2. The Silhouette Method:

The **Silhouette coefficient** value gives the measure of how close a data point is to its own cluster (cohesion) compared to other clusters (separation). It means if data points are correctly clustered then the distance of a point from all other points in the same cluster will be a lot smaller than the distance of the point from all other points not in the same cluster.

The range of **the Silhouette coefficient** is [-1,1], a higher value of the Silhouette coefficient means better clustering of data and a negative value indicates the clustering is not done properly. Thus, **a higher value is preferable**, ideally close to 1. When we see the plot of the Silhouette coefficient vs. K, the maximum value in the plot gives an optimal value of K.

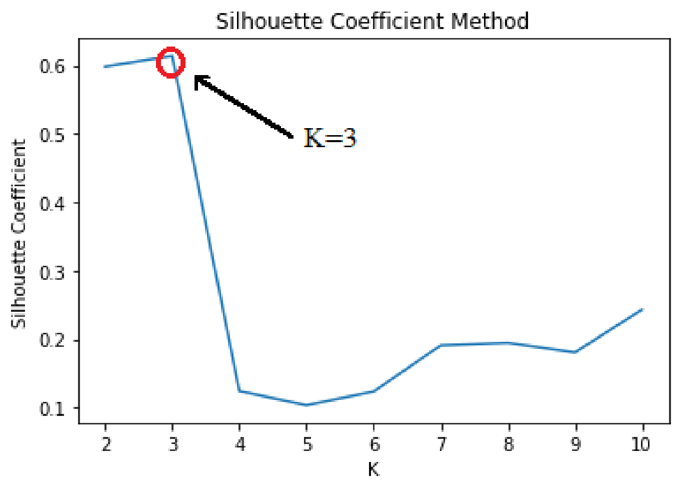


Figure 2.3: Silhouette coefficient vs. K plot

Though a global maximum Silhouette score is desirable for a clustering problem, it should not be used alone for determining the K-value. Silhouette coefficient should be used as a metric for validation technique and elbow method should be used as a resolution technique. Thus, the use of both methods together helps us in finding the optimal value of K.

2.3 Fuzzy Reasoning

Informally, the reasoning is the process of drawing conclusions from a collection of facts and principles. The subject logic deals with the formalization of methodology and reasoning concepts. In traditional logic, production rules are referred to as implication rules. Traditional logic's implication principles are extended to include fuzzy linguistic variables.

We can write one rule in classical predicate logic by

$$IF\ sky\ is\ cloudy\ and\ clouds\ are\ black\ THEN\ it\ is\ going\ to\ rain.$$

Three predicates for the above formalization can be given as Cloudy (x), Black (y), and Rain(z). Each predicate has just two truth values: true or false. The above rule can be written using the IF-THEN (implication) operator:

$$Cloudy(x), Black(y) \rightarrow Rain(z)$$

The comma in the left side of the implication sign (→) indicates logical conjunction of the antecedent predicates.

The implication rules in classical (propositional/ predicate) logic have their extension in the fuzzy domain and they are called fuzzy implication relations. They are discussed in the next section.

2.3.1 Fuzzy Implication Relations

A fuzzy implication relation for a given rule: IF x is $A_{i,}$ THEN y is B_i is represented as

$$R_i(x, y)= \{\mu_{Ri}(x, y) / (x, y)\} \quad (2.5)$$

The membership function R_i (x, y) can be formed in a variety of ways. Now, we will see some of the standard fuzzy implication relations.

1. **Zadeh Implication:** It is a fuzzy extension of classical propositional logic. The propositional implication $x \rightarrow y$ can also be phrased as '**either x and y are true, or x is false**' in propositional logic. As a result, $x \rightarrow y$ is equivalent to $(x \wedge y) \vee (\neg x)$. Where $\wedge$, $\vee$ and $\neg$ denote the logical AND, logical OR, and logical negation respectively.

We may state the membership function of the fuzzy implication rule IF x is Ai, THEN y is Bi by representing logical AND by min, logical OR by max, and negation by one's complement:

$$\mu_{Ri}\,(\mathrm{x},\mathrm{y}) \;= \mathrm{Max}[\;\mathrm{Min}\left(\mu_{Ai}\,(\mathrm{x}), \mu_{Bi}\,(\mathrm{y})\right), 1 - \mu_{Ai}\,(\mathrm{x})\;] \tag{2.6}$$

2. **Dienes-Rescher Implication:** In propositional logic if two propositions are given by **x** and **y**, then the implication $x \rightarrow y$ can be demonstrated to be equivalent to $(\neg x \;\vee y)$. The implication IF x then THEN y means that x is true, but y is false is impossible. Logically it means $(x \wedge \neg y\;)$ is false. Again $\neg x \;\vee y$ is true can be derived from $\neg\,(x \wedge \neg y\;)$ using De Morgan’s law.

   The membership function for Dienes-Rescher implication can be given for the implication rule IF x is Ai, THEN y is Bi by replacing negation with one’s complement and logical OR operator by max operator:

$$\mu_{Ri}\,(\mathrm{x},\mathrm{y}) \;= \mathrm{Max}[1 - \mu_{Ai}\,(\mathrm{x}),\; \mu_{Bi}\,(\mathrm{y})] \tag{2.7}$$

3. **Lukasiewicz Implication:** In fuzzy logic, the propositional implication function can be given by replacing negation with one's complement and logical OR with the sum (+) operator.

   The membership function for the fuzzy implication rule IF x is Ai, THEN y is Bi can be expressed as $1 - \mu_{Ai}\,(\mathrm{x}) + \;\mu_{Bi}\,(\mathrm{y})$ and as it is clearly visible the result may exceed 1, thus Lukasiewicz implication is given by

$$\mu_{Ri}\,(\mathrm{x},\mathrm{y}) \;= \mathrm{Min}[1, 1 - \mu_{Ai}\,(\mathrm{x}) + \;\mu_{Bi}\,(\mathrm{y})] \tag{2.8}$$

4. **Mamdani Implication:** It is one of the most widely is fuzzy implication. Mamdani implication assumes that fuzzy IF-THEN rules are local. Mamdani proposed two implication functions and they are known as Mamdani-Min and Mamdani-product.

   a) Mamdani-min: $\mu_{Ri}\,(\mathrm{x},\mathrm{y}) \;= \mathrm{Min}[\mu_{Ai}\,(\mathrm{x}),\; \mu_{Bi}\,(\mathrm{y})]$ (2.9)

   b) Mamdani-product: $\mu_{Ri}\,(\mathrm{x},\mathrm{y}) \;= \mu_{Ai}\,(\mathrm{x})\,\mu_{Bi}\,(\mathrm{y})$ (2.10)

2.3.2 Approximate Reasoning with Multiple Antecedent Clauses

Let us take a fuzzy IF-THEN rule, which has two antecedent clauses:

$$IF\ x\ is\ A\ and\ y\ is\ B\ THEN\ z\ is\ C$$

where x, y, and z are three linguistic variables in the universes X, Y, and Z, and A, B, and C are three fuzzy sets in the corresponding worlds.

If the membership functions for linguistic variables x, y, z belonging to fuzzy sets A, B, and C respectively is given by $\mu_A(x), \mu_B(y), \mu_C(z)$ then the membership distribution of the clause “if x is A and y is B” is written as t $(\mu_A(x), \mu_B(y))$.

The antecedent membership AM is denoted by the above t-norm, and $\mu_C(z)$ is consequent membership CM. The membership distribution $\mu_R\ (x, y; z)$ for the implication IF x is A and y is B THEN z is C is given by

$$\text{Mamdani implication} \begin{cases} AM\ CM \\ Min\ (AM, CM) \end{cases}$$

$$\text{Dienes-Rescher implication Max (1-AM, CM)}$$

For n number of antecedent clauses, the generalized version of membership distribution for fuzzy implication rule IF x_1 is A_1 and x_2 is A_2 and …. X_n is A_n THEN y is B is

$$AM\ =\ t\ (\mu_{A1}(x1), \mu_{A2}\ (x2), \mu_{A3}(x3), \dots, \mu_{An}\ (xn))\ and\ CM\ =\ \mu_B\ (y) \qquad (2.11)$$

Chapter 3

Database Management System

Database essentially means the computerized storage of data and a database management system is a software developed to keep those data efficiently.

3.1 Database Management System

A database management system (DBMS) is software that helps in the creation and management of databases. DBMS establishes an interface between the database and the applications and users. A DBMS manages to create, read, update and delete operations on databases.

3.1.1 Advantages of Database Management System

The need for database management system rises because of certain advantages over normal file system:

1. **Data Redundancy**: In a normal file system there is no logical way to remove redundancy. There may be a possibility that two users are maintaining the data of the same file for different applications. The changes made by one user do not reflect the data of the other user. This leads to inconsistency in data. DBMS maintains a single instance of data that can be accessed by many users. Whenever one user makes some changes to data, it is reflected everywhere the same data is being used.

2. **Concurrent Access**: It means the same data is being accessed by multiple users. The normal file system does not provide any security features for concurrent access, a problem arises when one user is trying to read the data and the other is trying to write onto the same data. DBMS provides a locking system to stop anomalies to occur. It ensures that multiple users are not trying to manipulate the same data simultaneously.

3. **Data Sharing**: Normal file system does not provide data sharing features, or this comes with too many constraints. DBMS provides an easy way to share data due to its centralized nature.
4. **Data Searching**: For different search operations performed on a file system different application program must be written because the file system contains unstructured and unrelated data. With DBMS the search is very efficient as with simple queries we can retrieve the needed data. This can be done using SQL (Structured Query Language).
5. **Data Integrity**: Data integrity means some constraints to be followed while inserting, deleting, or updating data, the file system does not provide such integrity rules. DBMS enforces data integrity by following user-defined constraints.
6. **Interfacing**: DBMS provides different interfacing techniques like **Graphical User Interface (GUI)**, **Application Programming Interface (API).**
7. **Data Security**: There may be situations where all users are not authorized to view all data. This type of permission-based data access can easily be applied to DBMS. Only authorized users can view data, it resolves issues like data theft and misuse. Data security keeps the integrity, availability, and confidentiality of data.

3.1.2 Components of DBMS

The components of a Database Management System (DBMS) can be divided into six major components, and they are:

1. Hardware
2. Software
3. Data
4. Procedures
5. Database Access Language
6. Users

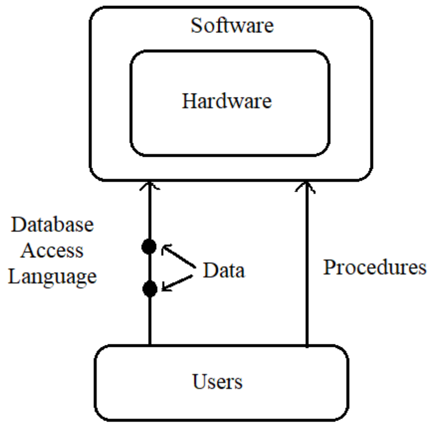


Figure 3.1: Components of DBMS

Hardware: It is the physical component of the DBMS. It is the computer that is responsible for storing the data and includes hard disks, I/O channels for data, and other physical components responsible for the successful storage of the data.

Software: It is the main component of a Database Management System. It is a set of programs/instructions that control everything from how data is stored, what operations are performed, and how data is queried from the database. It is capable of understanding Database Access Language and converting it into actual database commands to be run on the database.

Example: MySQL, PostgreSQL, Oracle, Sybase.

Data: Data is the main resource for which the DBMS was designed. It is the raw information stored in the database from which meaningful conclusions will be drawn. There are structured data, non-structured data, and logical data. The database also stores metadata into it; metadata is the data about data.

Procedures: Procedures are the instructions/guidelines about how to use the database and what rules to follow to efficiently maintain the database. Procedures include setup and installation of a DBMS, managing of database, login, and logout of DBMS, taking backups and generating reports, etc.

Database Access Language: It is the language used to write commands to retrieve, insert, delete, and update data stored in the database. Using this language users write commands to be submitted to the database for execution. Examples of database access languages are **SQL (structured query language), My Access, Oracle**, etc.

Database Access language is of two types:

a) Data Definition Language (DDL)
b) Data Manipulation Language (DML)

Users: Users are the people who manage and control the databases and perform different operations on the data stored in the DBMS. The users are categorized mainly into three people.

1. **Database Administrator:** Database Administrators (DBA) manage the complete database management system. They take care of the security of the DBMS, its availability, access protocols, user accounts, license keys, etc.

2. **Software Developer:** Software developers design the parts of DBMS. They can handle massive quantities of data, alter and edit databases, design and develop new databases, and troubleshoot database issues.

3. **End-User:** End users are the ones whose data is stored in the database of DBMS. They are the ones who create, read, update and delete data.

3.2 Relational Database Model

Database management has become increasingly important in scientific and commercial applications including banking transactions, digital libraries, electronic commerce, space research, airline, telecommunication and supply chain management in manufacturing. There are several data models used to store this information.

Among these data models the popular,

1. **Hierarchical data model**
2. **Relational data model**
3. **Network data model**
4. **Object-oriented data model**
5. **Logic program-based data model**

Among all these data models, the relational data model is widely used.

Relational database design intends to ensure the relational schema should not include repeated entries, i.e., redundancy should be eliminated. Two basic conditions should be satisfied for a relational database design,

1. The lossless join condition should be intact in decomposing a relation.
2. Functional dependency preservation criteria should be satisfied.

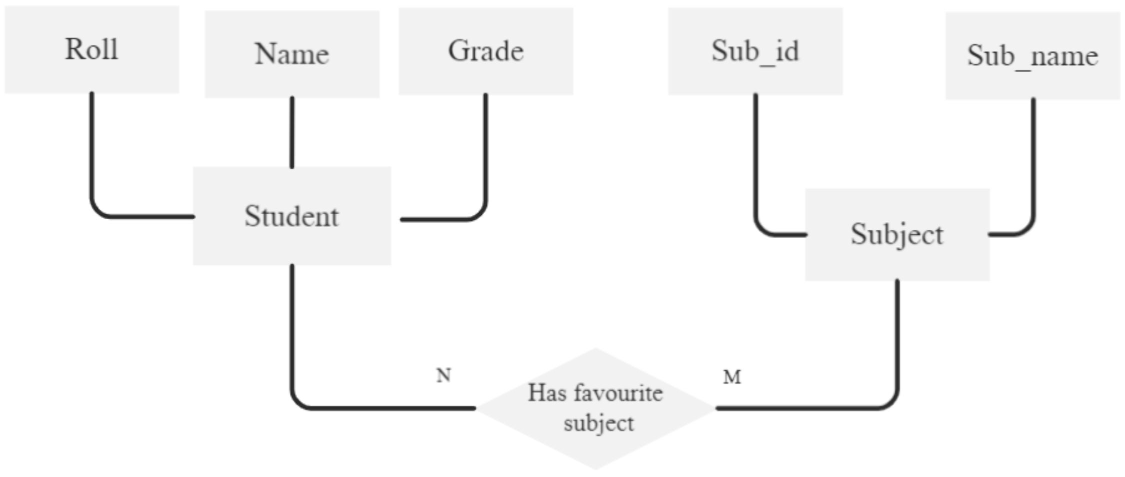
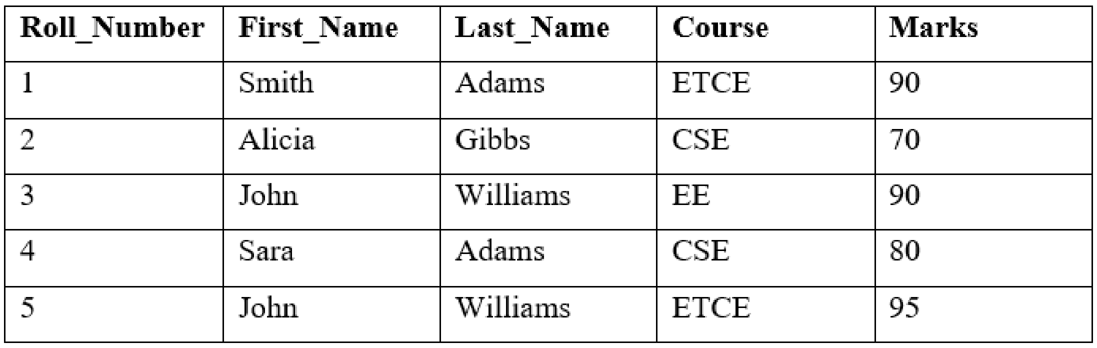


Figure 3.2: Student database where students have multiple favourite subjects

3.2.1 How to Identify Functional Dependency

If **B** is functionally dependent on **A** in a relation R, then it is denoted by $A \rightarrow B$. A is called the **determinant** and B is called the **dependent**.

Table 3.1: An example student database table



Roll_Number	First_Name	Last_Name	Course	Marks
1	Smith	Adams	ETCE	90
2	Alicia	Gibbs	CSE	70
3	John	Williams	EE	90
4	Sara	Adams	CSE	80
5	John	Williams	ETCE	95

From above table we can find some functional dependencies:

- {Roll_Number, First_Name, Last_Name} → Course
- {Roll_Number, First_Name, Last_Name} → {Course, Marks}
- Roll_Number → {First_Name, Last_Name, Course, Marks}
- Roll_Number → {First_Name, Last_Name}

There are a few more functional dependencies in this database table since **Roll_Number** can uniquely determine all four attributes {First_Name, Last_Name, Course, Marks} any subset of these four attributes can also be determined with roll number.

3.2.2 Conditions of Functional Dependency

FD on a relation R for two attributes A and B where A can uniquely identify B is written as $A \rightarrow B$ and it means from the relation R (also called a table) if we know the value of A for a tuple then we also know the value B for that tuple.

This means with help of FD we can look up the value of dependent attributes with the help of determinant attributes with reference to the relation R. It does not mean we are computing the value of B from the value of A with help of a function, like B=f(A).

Table 3.2: Example relation between work experience and salary of employees

Work experience(years)	Salary (in INR)
5	25000
10	60000
2	18000
15	100000

From the above relation given in figure 3.2, FD $\{Work\ experience\} \rightarrow \{Salary\}$ exists and gives the notion of looking up **Salary** value from **Work experience** value and not computing the value of **Salary** with the help of **Work experience**.

The conditions for the FD, $A \rightarrow B$ exists when at least one of the following conditions are met:

1. All values of **A** are unique in the relation.
2. All values of **B** are same in the relation.
3. When two tuples/rows in the relation have same value for attribute A, then the tuples also have same values for attribute B i.e., if t1[A] = t2[A] then there must be t1[B] = t2[B].

3.2.3 Armstrong's Axioms

William W. Armstrong gave a set of inference rules or axioms that can find the exhaustive set of functional dependencies in a relational database, these are popularly known as Armstrong's axioms. This exhaustive set of all functional dependencies in a relational database is called the **closure** of functional dependencies.

If F is a set of functional dependencies for a relation r, then $F^+$is called the closure of functional dependency F, which is a set of all regular functional dependencies derived from F using Armstrong's axioms on relation r.

Let A and B be two sets of attributes in a relation r.

1. **Reflexivity**: If B is subset of A i.e., B$\subseteq A$ then $A \rightarrow B$ is a functional dependency in r.

Example: {Roll_Number, First_Name} → First_Name is valid.

2. **Augmentation**: If $A \rightarrow B$ is a valid functional dependency in r then $AC \rightarrow BC$ is also a valid functional dependency in r by the rule of augmentation. Here, AC means $A \cup C$ and BC means $B \cup C$.
   Example: If Roll_Number → Course is valid then {Roll_Number, First_Name} → {First_Name, Course} is also valid.

3. **Transitivity**: If $A \rightarrow B$ and $B \rightarrow C$ are both valid functional dependencies in r then by the rule of transitivity $A \rightarrow C$ is also valid in r.
   Example: If {Roll_Number} → {First_Name} is valid and {First_Name} → {Last_Name} is also valid functional dependency in r then {Roll_Number} → {Last_Name} is also valid functional dependency in r.

The above three rules are primary rules of functional dependency using which we find other functional dependencies also valid in the database. Apart from these three rules, there are four secondary rules which can be derived from the above primary axioms.

Secondary rules:

I. **Union:** If A→ B and A → C are valid functional dependencies in relation r, then A → BC is also a valid functional dependency.
II. **Pseudo Transitivity:** If A → B and BC → D are valid functional dependencies, then AC → D is also a valid functional dependency.
III. **Decomposition:** If A → BC is a valid functional dependency in r, then A → B and A → C are both valid.
IV. **Composition:** If A → B and C → D are both valid functional dependencies in relation r, then AC → BD is also a valid functional dependency in r.

Armstrong's axioms are a set of inference rules used to find all functional dependencies (FD) from a set of functional dependency F. It means if we are given a set of FDs over a relational schema r, then with the help of Armstrong's axioms the complete set of FDs can be inferred using the set F, which is called the closure of FD over F and denoted by

F^+. All FDs that are valid in the relation r will be discovered for relation only if F contains all minimal, non-trivial, and normalized FDs.

Types of Functional Dependency:

1. **Trivial Functional Dependency**: A dependent is always a subset of the determinant in trivial functional dependency. If $X \rightarrow Y$ is a FD and Y is a subset of X then it is trivial functional dependency.
   For example, from table 3.1 {Roll_Number, First_Name} → {First_Name} is a trivial FD as dependent set {First_Name} is subset of determinant set {Roll_Number, First_Name}.

2. **Non-Trivial Functional Dependency**: The dependant in non-trivial functional dependency is strictly not a subset of the determinant. If $X \rightarrow Y$ is a FD and Y is not a subset of X then it is called non-trivial functional dependency.
   For example, from table 3.1 Roll_Number → First Name is a non-trivial FD as dependent set {First_Name} is not a subset of determinant set {Roll_Number}.

3. **Multivalued Functional Dependency**: Entities in the dependent set are not dependent on each other in multivalued functional dependency. A multivalued functional dependency exists when X → {Y, Z} exist but there is no functional dependency between Y and Z.
   For example, from table 3.1, Roll_Number → {First_Name, Course} is a multivalued FD since dependents First_Name and Course are not dependent on each other (i.e., First_Name → Course or Course → First_Name does not exist).

4. **Transitive Functional Dependency**: Dependent is indirectly dependent on determinant in transitive functional dependency. i.e., if X → Y and Y → Z, then according to the axiom of transitivity X → Z.
   For example, in an ecommerce database, Customer_Id → Order_Id and Order_Id → Address, then according to axiom of transitivity Customer_Id → Address is also a valid FD.

Now, let us discuss two more types of FD over a relation r, and they are,

1. **Minimal Functional Dependency**: If the dependent is not functionally dependent on any subset of determinant, then it is a minimal functional dependency. i.e., X → Y is said to be minimal if no subset of X determines Y. For example, if $\{X, Y\} \rightarrow Z$ is valid in a relation r but $X \rightarrow Z$ and $Y \rightarrow Z$ are not valid, then $\{X, Y\} \rightarrow Z$ is said to be minimal FD in relation r.
2. **Normalized Functional Dependency:** A functional dependency is said to be normalized if the dependent set is composed of a single attribute. i.e., the FD X → Y is to be normalized if Y is a single attribute in the relation.

From, relation r, it is sufficient to find all non-trivial, minimal, and normalized FDs to discover the complete set of FDs existing in the relation

Chapter 4

Proposed Approach

This chapter discusses the concepts and logic used for the proposed approach in this thesis. It helps us to understand how the algorithm/approach is working and what it is trying to achieve. The theoretical concepts studied, literature surveys done, and studies of various implementations and algorithms conceptualized by several researchers helped to propose this new approach. The different problems solved by previous research in the field of discovering approximate dependencies and functional dependencies help us to understand the motivation and scope of this approach.

It talks about the logic and how it can be incorporated into the real use case for real-world applications. It starts with the intention and core concept of the approach. The algorithm developed for the approach is named **FD-CFR** (Functional Dependency by Clustering and Fuzzy Relations). Then we will see the work plan of the project.

4.1 Main Idea Behind the Logic

The functional dependency in a relational database is given by, if r is a relation in a relational schema R and X, Y are any two subsets of R then the relation satisfies FD $X \rightarrow Y$ if for any two tuples t_1, t_2 whenever $t_1[X]=t_2[X]$ then $t_1[Y]=t_2[Y]$. This method of finding FDs from a relation is too rigid and does not actually try to find the existence of the meaningful relationship between attributes.

In the real world, the data is often full of errors and ambiguities thus checking with absolute equality may lead to a loss in interesting findings. The problem becomes even worse when the attributes in a relation are mostly numeric or all attributes are numeric. The collected data may show some dependencies between sets of attributes, where one set of attributes is determining the other set but may deviate by small margins in some tuples. It is expected in real-life scenarios. Examples could be while measuring the weight of a person on different machines, the machines can show weights with slight variations in values. This does not mean, it is not giving meaningful values as 68.54 kg, 68.50 kg, and 68.4kg are more or less the same. The variations are small enough to ignore. Thus, to capture this information fuzzy logic is used in this approach.

The typical propositional inference rules are:

1. Modus Ponens
2. Modus Tollens
3. Hypothetical Syllogism

These rules are used in classical propositional logic for logical inferencing, i.e., to derive logical conclusions. The fuzzy extensions of these three classical propositional logics are:

1. Generalized Modus Ponens
2. Generalized Modus Tollens
3. Generalized Hypothetical Syllogism

The fuzzy compositional rule of inference uses the above fuzzy propositional logics. This gives us the following desired advantages to solve the problem of finding approximate dependencies:

1. Where classical propositional logic works on binary valuation space of true/false, the fuzzy logic uses a multivalued truth space [0,1], giving different degrees of truth values.

2. Classical propositional logic generates conclusions based on a complete match of the antecedent clauses with the available data, whereas fuzzy logic can generate inferences even when the antecedent clauses are partially matched to the data components in WM.

Using the conditions of FD, we find the dependencies in a relation. But, with the number of attributes growing, the number of possible FDs present in a relation can become exponentially high. We can bound the number of FDs to be checked by restricting the search space to only **minimal, normalized, and non-trivial FDs**. The TANE algorithm [11] finds only **minimal and non-trivial** approximate dependencies using an error threshold $\varepsilon\,(0 \leq \varepsilon \leq 1)$. The measure of the error in the TANE algorithm is represented as a fraction of tuples with exceptions or errors affecting the dependency. On the other hand, the FD_MINE algorithm [23] uses equivalences to reduce the size of the dataset which in turn reduces search space for FDs.

For our approach, we are using fuzzy logic and clustering techniques to efficiently find the FDs and approximate dependencies present in the relation. We try to focus on finding dependencies between numerical attributes, that's why the datasets used contain only numerical columns and the datasets where categorical columns present with insignificant importance. Clustering can find some interesting patterns in the data, that's why we first cluster the complete dataset into K-clusters using K-Means with an optimal value of K. Then we define the membership functions of each attribute/column with Gaussian membership functions within clusters. After that, approximate reasoning is done using fuzzy inferences. The RMSE between fuzzy inference generated and actual membership values gives the weakness in the relationship. We call the error (RMSE) relationship weakness parameter (α). The less the value of α the stronger the dependency exists between two sets of attributes. In the proposed approach we only check the **normalized** and **non-trivial** dependencies.

The complete workplan of the approach is given in the next section.

4.2 Workplan

The Design and structure of a Fuzzy logic-based system not only just include the concept of uncertainty and imprecision but interfaces different sets of ideas and functions all together to implement the whole system. The main idea is to provide a more accurate finding of the relationship between different attributes, also find meaningful approximate dependencies, and measure the relationship strengths. Fuzzy implication relations using the membership values between sets of attributes are built using the relational matrices with the data falling under various clusters differently and comparing them with different implication relations give a holistic view and uncovers sensible data dependencies if present. It helps database designers, data scientists, and AI and ML Engineers more sophistication to handle data, avoid redundancy, and provide important insights.

The complete Functional dependency finding system consists of the following steps:

1. Collect the raw data from sensors or use already collected data.
2. Remove unwanted formatting discrepancies and perform the necessary transformations required.
3. Select the attributes for which the FD discovery algorithm proposed in this thesis be applied.

4. Normalize the data and find the attributes which will be used in the clustering of data.
5. Then use the K-Means clustering algorithm for clustering and use validation techniques to find an optimal value for the number of clusters K.
6. Find the membership value of each entry in a column as a measure of the deviation of the data point from the attribute means using statistical measures. The Gaussian membership function is used in this thesis which gives how dispersed the data is from the attribute centroid.
7. The next step is to build relational matrices between two sets of attributes A, and B using fuzzy implication relations.
8. The fuzzy composition techniques and error measuring tools find how good a dependency holds between a set of attributes.
9. A comparative study between implication relations reveals a relatively satisfactory implication function to make sense of relationship strengths.
10. Finally, varying the threshold for the relationship weakness parameter (α) we find a different number of dependencies from the relation.

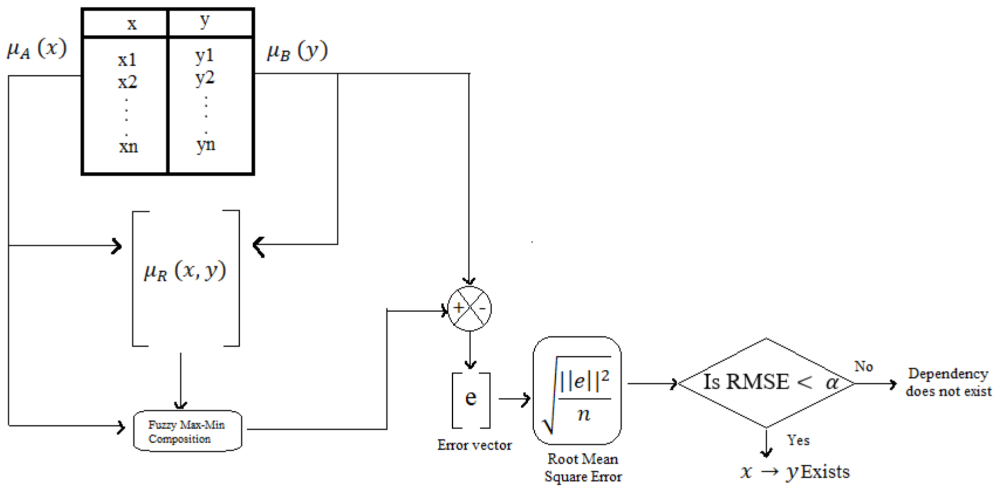


Figure 4.1: Calculation procedure to determine the existence of dependency between a set of attributes

In the next section complete work of the proposed approach is given in the form of flowcharts.

4.3 Flowchart

The complete flow of the proposed approach is divided into four main steps.

1. Preparing the data
2. Clustering the data
3. Defining membership values for the crisp dataset
4. Finding functional dependency using the proposed algorithm

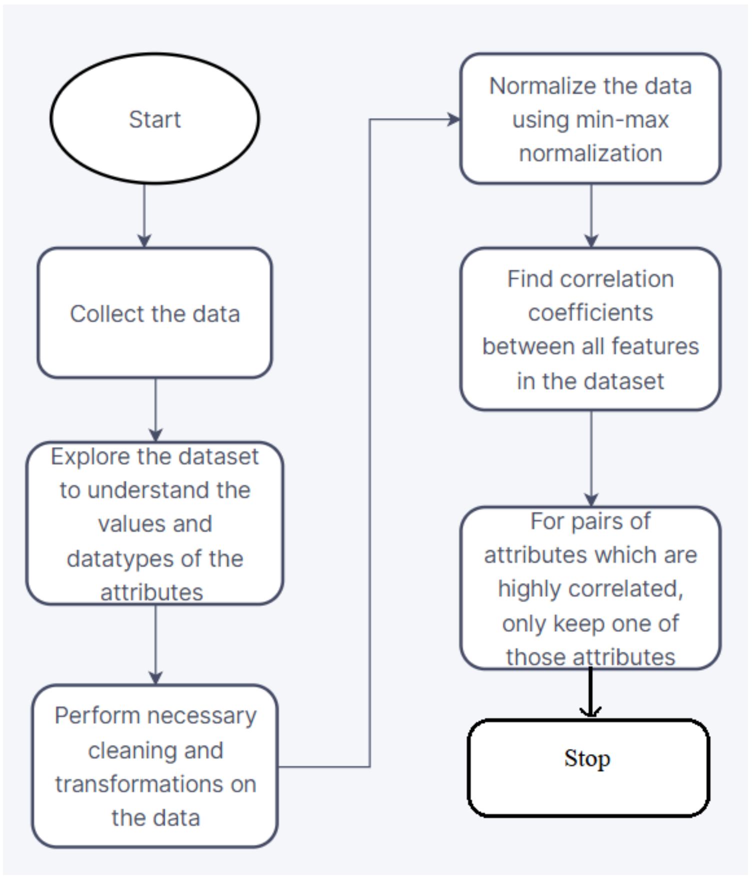

Start
Collect the data
Explore the dataset to understand the values and datatypes of the attributes
Perform necessary cleaning and transformations on the data
Normalize the data using min-max normalization
Find correlation coefficients between all features in the dataset
For pairs of attributes which are highly correlated, only keep one of those attributes
Stop

Figure 4.2: Data preparation phase

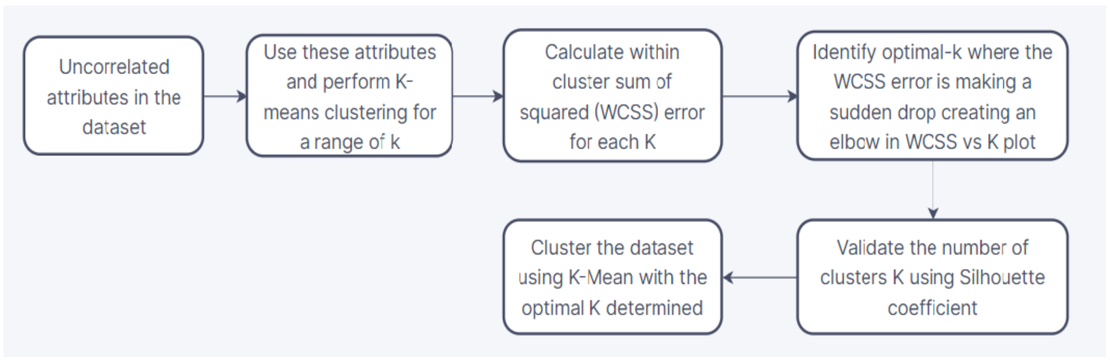

Uncorrelated attributes in the dataset
Use these attributes and perform K-means clustering for a range of k
Calculate within cluster sum of squared (WCSS) error for each K
Identify optimal-k where the WCSS error is making a sudden drop creating an elbow in WCSS vs K plot
Validate the number of clusters K using Silhouette coefficient
Cluster the dataset using K-Mean with the optimal K determined

Figure 4.3: Clustering the dataset with K-Means clustering

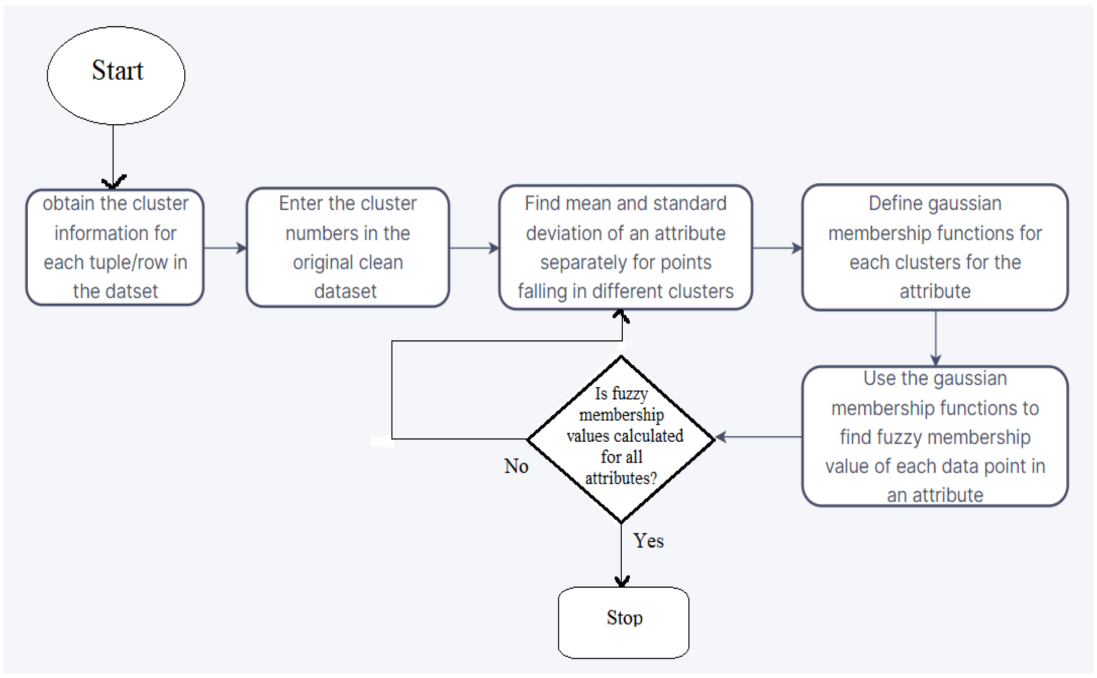

Start
obtain the cluster information for each tuple/row in the datset
Enter the cluster numbers in the original clean dataset
Find mean and standard deviation of an attribute separately for points falling in different clusters
Define gaussian membership functions for each clusters for the attribute
Use the gaussian membership functions to find fuzzy membership value of each data point in an attribute
Is fuzzy membership values calculated for all attributes?
No
Yes
Stop

Figure 4.4: Defining gaussian membership values for the crisp dataset

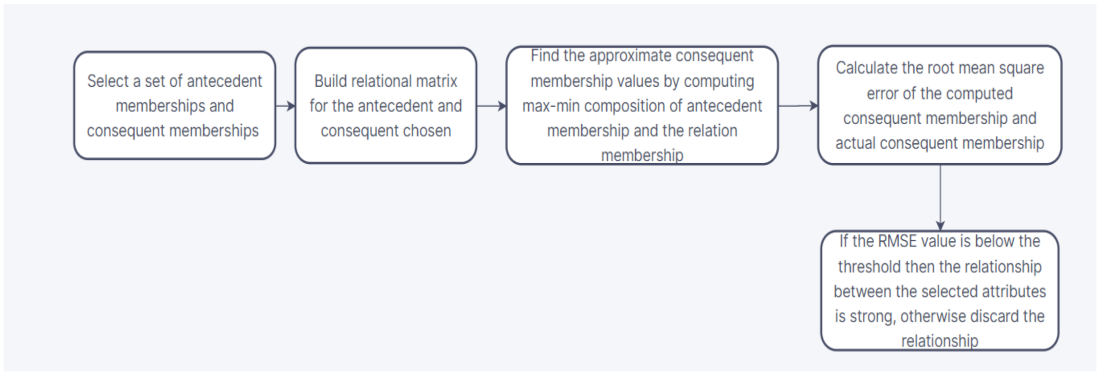

Select a set of antecedent memberships and consequent memberships
Build relational matrix for the antecedent and consequent chosen
Find the approximate consequent membership values by computing max-min composition of antecedent membership and the relation membership
Calculate the root mean square error of the computed consequent membership and actual consequent membership
If the RMSE value is below the threshold then the relationship between the selected attributes is strong, otherwise discard the relationship

Figure 4.5: Finding the existence of dependency between a set of attributes

Chapter 5

Work Done and Results

The work done and results talk about the actual work of the project, the results obtained, the analysis, and the performance of the proposed logic. The analysis of experiments performed on various datasets and their outcomes are given in this chapter.

5.1 Dataset 1

Let us first discover how the implication relations and gaussian membership functions defined over the crisp values of the table looks like. Let us take a simple table with only two attributes A and B and **three** variations of the table, one with if t1(A)=t2(A) then t1(B)=t2(B) holding good and other ones with one and two tuples violating the rule respectively.

5.1.1 Variation-1 of Dataset-1

Table 5.1: Example dataset for testing

A	B
1	1
2	2
3	3
4	4
1	1
6	7
9	8

The non-trivial functional dependencies holding in the above table is $A \rightarrow B$ & $B \rightarrow A$ i.e., the equivalence $A \leftrightarrow B$ exists.

Now, let us see the relationship weakness parameter α (RMSE) found for all trivial and non-trivial FDs for different fuzzy implication functions.

Table 5.2: Relationship strengths using different implication relations for table 5.1

Dependency	Mamdani Min	Mamdani product	Dienes-Rescher	Zadeh	Lukasiewicz	Gödel
$A \rightarrow A$	0	0.00362	0.05796	0.05796	0.16196	0
$A \rightarrow B$	**0**	**0.00348**	**0.01376**	**0.01376**	**0.17667**	**0**
$B \rightarrow A$	**0.00015**	**0.00393**	**0.11296**	**0.11296**	**0.17155**	**0.00015**
$B \rightarrow B$	0	0.00377	0.06876	0.06876	0.16315	0

From the above result and the known dependencies, the results from Mamdani-Min and Gödel relations are intuitive. Let us further test on the same dataset.

5.1.2 Variation-2 of Dataset-1

Now changing one tuple of the previous table 5.1, so that we get one tuple violating the FD constraint.

Table 5.3: Modified version of example table 5.1

A	B
1	1
2	2
3	3
4	4
1	**5**
6	7
9	8

As a result of the modification, now the $A \rightarrow B$ dependency does not hold but $B \rightarrow A$ dependency holds in the relation.

Table 5.4: Relationship strengths using different implication relations for table 5.3

Dependency	Mamdani Min	Mamdani product	Dienes-Rescher	Zadeh	Lukasiewicz	Gödel
$A \rightarrow A$	0	0.00362	0.05796	0.05796	0.16196	0
$A \rightarrow B$	0	0.00349	0	0	0.17437	0
$B \rightarrow A$	0.00054	0.00472	0.09216	0.09216	0.16215	0.00054
$B \rightarrow B$	0	0.00455	0.03392	0.03392	0.16344	0

The dependency $A \rightarrow B$ does not hold now in the relation because of one tuple violating the condition and which is 0.14 as a fraction of (error tuples)/(total tuples). Still the results from Mamdani-Min and Gödel relations are intuitive.

5.1.3 Variaton-3 of Dataset-1

Now changing one tuple of the previous table 5.3 so that we get **two** tuples violating the FD constraint,

Table 5.5: Modified version of example table 5.3

A	B
1	1
2	2
3	3
4	4
1	5
6	7
2	8

As a result of the modification, now the $A \rightarrow B$ dependency does not hold but $B \rightarrow A$ dependency still holds in the relation.

Table 5.6: Relationship strengths using different implication relations for table 5.5

Dependency	Mamdani Min	Mamdani product	Dienes-Rescher	Zadeh	Lukasiewicz	Gödel
$A \to A$	0	0.00962	0.06651	0.06651	0.15948	0
$A \to B$	0.00239	0.00921	0.00606	0.00606	0.16308	0.00239
$B \to A$	0	0.00475	0.09487	0.09487	0.16400	0
$B \to B$	0	0.00455	0.03392	0.03392	0.16344	0

From dataset-1 we found that Gödel and Mamdani-min fuzzy relations are giving intuitively good results and for the condition FD $A \to B$ dependency not holding, the relationship weakness parameter (α) is coming to be 0.00239. So if the cut off is less than this then we can ignore $A \to B$. This conclusion is not strong enough, for setting values of **α**, we need extensive testing. That will be performed on some more example datasets and after that on benchmark datasets.

5.2 Dataset-2

Table 5.7: Example relation 2

A	B	C
7	1	8
7	2	5
7	3	5
5	8	8

The trivial functional dependencies such as $A \to A$, $B \to B$ are not considered as they are absolute basic dependencies present in any database. The **non-trivial** and **normalized** functional dependencies present in the dataset are:

1. $B \to A$
2. $B \to C$
3. $AB \to C$
4. $BC \to A$

5.2.1 Application of Dienes-Rescher Implication Relation to Dataset-2

Without using clustering of this data as the number of tuples is very small (less than 10) the **Dienes-Rescher** implication found these dependencies with (**RMSE threshold set as 0.002**):

1. $A \rightarrow C$
2. $B \rightarrow C$
3. $AB \rightarrow C$

Table 5.8: Outcome of FDs using Dienes-Rescher implication relation

Correct Dependencies Found	**Incorrect Dependencies Found**	**Lost dependencies**
2	1	2

5.2.2 Application of Mamdani-Min Implication Relation to Dataset-2

Without using clustering of this data as the number of tuples is small (less than 10) the **Mamdani-min** implication found these dependencies with (RMSE threshold set as 0.002):

1. $A \rightarrow C$
2. $B \rightarrow A$
3. $B \rightarrow C$
4. $AB \rightarrow C$

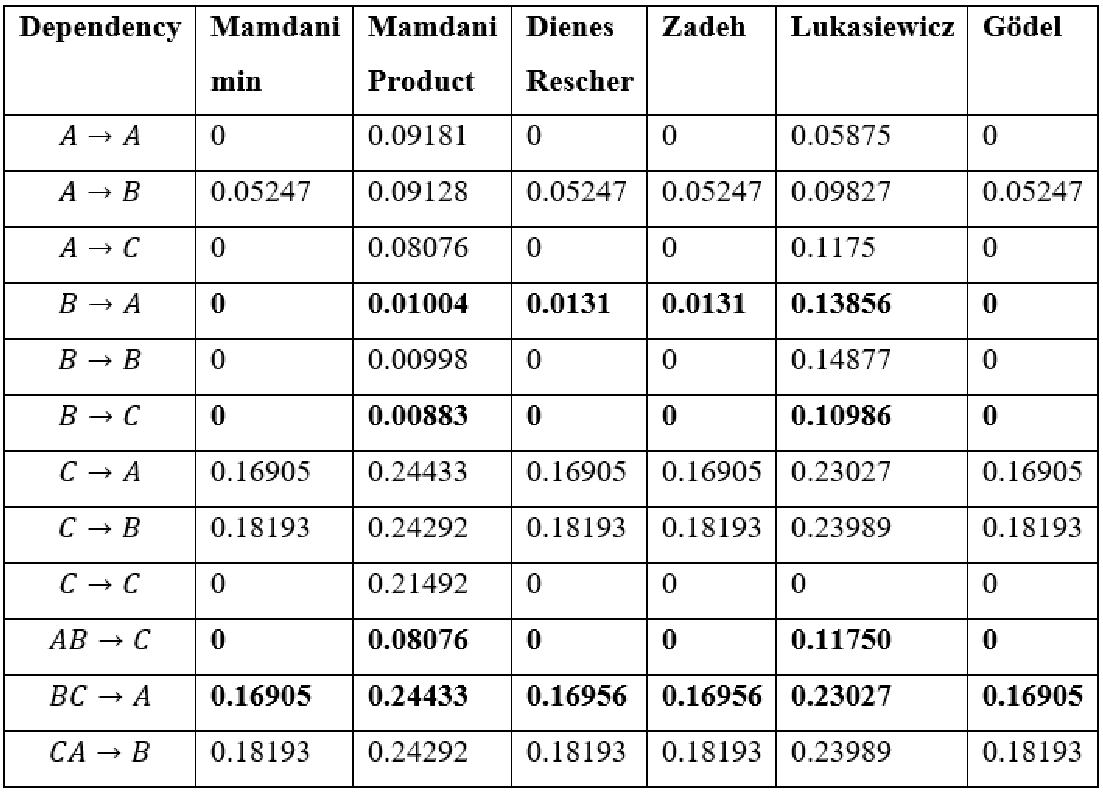
Table 5.9: Outcome of FDs using Mamdani-Min implication relation

Correct Dependencies Found	**Incorrect Dependencies Found**	**Lost dependencies**
3	1	1

5.2.3 Outcome of Functional Dependencies Found from Dataset-2

Non-trivial functional dependencies discovered are highlighted in bold. The below table contains all kinds of dependencies that can be drawn from table 5.7. The outputs are calculated even for trivial FDs to get the idea of our parameter (α) describing relationship weakness (RMSE) so that we can get an idea of (α) we should select for the discovery of FDs from various relations.

Table 5.10: value of α discovered using different implication relations



Dependency	**Mamdani min**	**Mamdani Product**	**Dienes Rescher**	**Zadeh**	**Lukasiewicz**	**Gödel**
$A \rightarrow A$	0	0.09181	0	0	0.05875	0
$A \rightarrow B$	0.05247	0.09128	0.05247	0.05247	0.09827	0.05247
$A \rightarrow C$	0	0.08076	0	0	0.1175	0
$B \rightarrow A$	**0**	**0.01004**	**0.0131**	**0.0131**	**0.13856**	**0**
$B \rightarrow B$	0	0.00998	0	0	0.14877	0
$B \rightarrow C$	**0**	**0.00883**	**0**	**0**	**0.10986**	**0**
$C \rightarrow A$	0.16905	0.24433	0.16905	0.16905	0.23027	0.16905
$C \rightarrow B$	0.18193	0.24292	0.18193	0.18193	0.23989	0.18193
$C \rightarrow C$	0	0.21492	0	0	0	0
$AB \rightarrow C$	**0**	**0.08076**	**0**	**0**	**0.11750**	**0**
$BC \rightarrow A$	**0.16905**	**0.24433**	**0.16956**	**0.16956**	**0.23027**	**0.16905**
$CA \rightarrow B$	0.18193	0.24292	0.18193	0.18193	0.23989	0.18193

5.3 Dataset-3

Table 5.11: Example Relation-3

A	**B**	**C**
1	6	8
2	7	8
5	6	8
5	7	8

The **non-trivial** and **normalized** functional dependencies present in the dataset are:

1. $A \rightarrow C$
2. $B \rightarrow C$
3. $AB \rightarrow C$

Table: 5.12: value of α discovered using different implication relations for table 5.11

Dependency	Mamdani min	Mamdani Product	Dienes Rescher	Zadeh	Lukasiewicz	Gödel
$A \to A$	0	0.11782	0	0	0.12692	0
$A \to B$	0	0.11541	0	0	0.14479	0
$A \to C$	**0.16792**	**0.16792**	**0.16792**	**016792**	**0.16792**	**0.16792**
$B \to A$	0.07275	0.21941	0.07275	0.07275	0.0996	0.07275
$B \to B$	0	0.21492	0	0	0	0
$B \to C$	**0.31271**	**0.31271**	**0.31271**	**0.31271**	**0.31271**	**0.31271**
$C \to A$	0	0	0	0	0	0
$C \to B$	0	0	0	0	0	0
$C \to C$	0	0	0	0	0	0
$AB \to C$	**0.31271**	**0.31271**	**0.31271**	**0.31271**	**0.31271**	**0.31271**
$BC \to A$	0.07275	0.21941	0.07275	0.07275	0.0996	0.07275
$CA \to B$	0	0.11541	0	0	0.14479	0

The results obtained show an odd behavior and this is since column 'C' has all identical entries making the Gaussian membership function to be described for this column difficult. As there is no variation in column 'C' the values are taken as 1 but this does not provide a good result. Though the result looks a little spurious, it does not hamper our current study as in relation to having all identical values for an attribute gives no special meaning to be derived from that column and it will be functionally dependent on all other features, so we can analyze the FDs discarding this column.

5.4 Dataset-4

Table 5.13: Example relation used in the research paper of discovering functional dependencies with FD_Mine algorithm [23]

A	**B**	**C**	**D**	**E**
0	0	0	1	0
0	1	0	1	0
0	2	0	1	2
0	3	1	1	0
4	1	1	2	4
4	3	1	2	2
0	0	1	1	4

The **α** values are taken as [0.005, 0.004, 0.003, 0.002, 0.0015,0.001]

Number of FDs checked = 75

Table 5.14: Summary of the result for finding FDs on example dataset 4 for different fuzzy implication relations

Implication-relation	**Number of FDs Found**					
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005
Mamdani-Min	30	30	30	30	30	30
Mamdani-Product	0	0	0	0	0	0
Dienes-Rescher	21	21	21	21	21	21
Zadeh	21	21	21	21	21	21
Lukasiewicz	0	0	0	0	0	0
Gödel	30	30	30	30	30	30

5.5 Test on Benchmark Datasets

Now we will see results on some benchmark datasets obtained from the UCI Machine Learning Repository. The above example datasets do not need any clustering to be performed as the number of rows/tuples is very small (less than 10) and the number of attributes is also small. The Functional dependencies are performed only on numeric columns. We will see the result on 5 UCI repository datasets.

The Strength of the relationship is represented with the parameter **α** (which is the RMSE of the calculated consequent membership values and actual consequent membership values), by varying the values for α we can find different numbers of FDs from a dataset.

As discussed earlier, the use of fuzzy logic helps in finding the functional dependencies from the dataset, and the implication relations **Gödel** and **Mamdani-Min** work better for FD discovery as for most of the non-trivial FDs that holds absolutely, or the strength is strong and the trivial-FDs, it produces α=0 (the ideal value). The more the α is near zero the more it is better but setting only the threshold for α to 0 is too rigid, and misses many FDs existing in the relation. Though both Gödel and Mamdani-Min performed relatively well, we are selecting the Mamdani-Min implication for our next experiments as this implication relation is simple, intuitive, and widely used.

5.5.1 Iris Dataset from UCI Repository

The total number of columns/attributes present in the dataset is 6, one of which is “**id**” which is unique in all rows and is in the range (of 1,150). It is redundant to use the 'id' column for FD discovery as it is unique in all rows, it can uniquely determine any other features. The last attribute is “**Species**” which represents the species a sample/row belongs to, it is a categorical feature, we will use this column to create 3 clusters and will not use it in FD discovery. So, the below result is based on 4 features of the dataset = {'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'}.

The total number of FDs checked on the mentioned attributes is 28, which includes only the normalized and non-trivial FDs possible in the dataset. The runtime for the proposed algorithm on this dataset is 0 seconds and the number of FDs found with different threshold values of weakness parameter (α) and different implication relations is given below.

Table 5.15: Summary of the result for finding FDs on Iris dataset for different fuzzy implication relations

Implication-relation	**Number of FDs Found**					
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005
Mamdani-Min	7	8	8	12	13	15
Mamdani-Product	0	0	0	4	6	6
Dienes-Rescher	0	0	0	0	0	0
Zadeh	0	0	0	0	0	0
Lukasiewicz	0	0	0	0	0	0
Gödel	7	8	8	12	13	15

5.5.2 Glass Dataset from UCI Repository

This dataset contains 10 attributes and 214 rows. One of the attributes is '**id**' which is a unique number of the sample or data taken and ranges from 1 to 214. It is redundant to use this attribute in FD discovery because it is unique in all rows and can determine all other attributes uniquely, so Armstrong's axioms can be used to find all other FDs where the determinant set contains the 'id' feature, but they will not be minimal with respect to this column, so no need to find explicitly.

The summary of the result is given below with only Mamdani-Min implication relations, as it worked best for FD discovery in our tests.

Table 5.16: Summary of results obtained from glass dataset with Mamdani-min implication relation

Number of FDs checked	Number of FDs Found						Runtime (seconds)
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005	
2295	88	112	125	151	169	186	33

5.5.3 Breast-Cancer Wisconsin (Diagnostic) Dataset from UCI Repository

This dataset contains a total of 32 attributes and 569 rows. One of the attributes is 'id' which is a unique code number for the sample or data taken. It is redundant to use this attribute in FD discovery because it is unique in all rows and can determine all other attributes uniquely, so Armstrong's axioms will be used to find all other FDs where the determinant set contains the 'id' feature, but they will not be minimal with respect to this column, so no need to find explicitly. The attribute-2 is the 'diagnosis' column and currently, we will ignore it as it is a categorical column, and columns 13-32 are derived columns. Features 3-12 are ten real-valued features and we will only apply our FD finding algorithm to these features.

The summary of the result is given below with only Mamdani-Min implication relations, as it worked best for FD discovery in our tests.

Table 5.17: Summary of results obtained from Breast-Cancer Wisconsin (Diagnostic) dataset with Mamdani-min implication relation

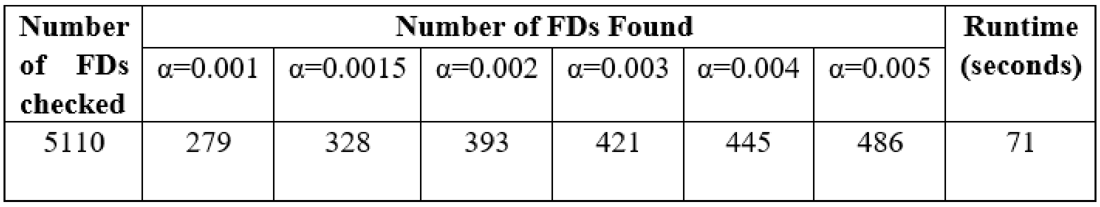
Number of FDs checked	Number of FDs Found						Runtime (seconds)
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005	
5110	553	656	747	1056	1315	1506	75

5.5.4 Breast-Cancer Wisconsin (Original) Dataset from UCI Repository

This dataset contains a total of 11 attributes and 699 rows. The first attribute is '**Sample code number**' which is a unique id number for the sample or data taken. It is redundant to use this attribute in FD discovery because it does not contain any meaningful information to convey. Features 2-10 are **nine real-valued features** that contain integer values in the range [1,10]. The attribute-11 is 'class' and it contains only two values {2,4} for '**benign**' and '**malignant**' respectively. After removing the '**Sample code number**' attribute we are left with 10 attributes, and we will only apply our FD finding algorithm on these features.

The summary of the result is given below with only Mamdani-Min implication relations, as it worked best for FD discovery in our tests.

Table 5.18: Summary of results obtained from Breast-Cancer Wisconsin (Original) dataset with Mamdani-min implication relation



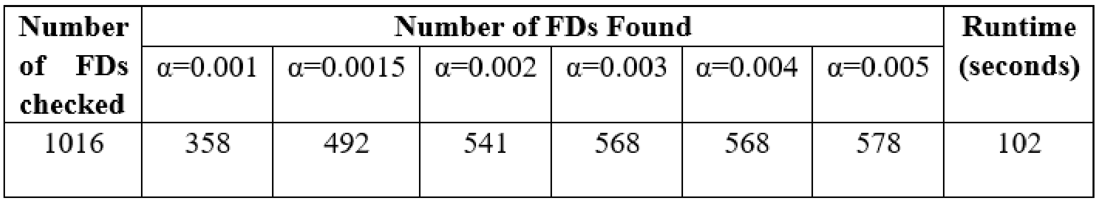
Number of FDs checked	**Number of FDs Found**						**Runtime (seconds)**
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005	
5110	279	328	393	421	445	486	71

5.5.5 Abalone Dataset from UCI Repository

This dataset contains 9 attributes and 4177 rows. One of the attributes is '**Sex**' which is a categorical column and only gives the gender of the abalone. The other 8 attributes are real-valued features, and we will apply our FD discovery algorithm only to these 8 attributes.

The summary of the result is given below with only Mamdani-Min implication relations, as it worked best for FD discovery in our tests.

Table 5.19: Summary of results obtained from Abalone dataset with Mamdani-min implication relation



Number of FDs checked	**Number of FDs Found**						**Runtime (seconds)**
	α=0.001	α=0.0015	α=0.002	α=0.003	α=0.004	α=0.005	
1016	358	492	541	568	568	578	102

5.5.6 Summary

Table 5.20: Summary of tests performed on benchmark datasets with FD-CFR

Dataset Name	**Number of Attributes**	**Number of Rows**	**Number of FD Checks**	**Runtime (seconds)**
Dataset 3	5	7	75	0
Iris	4	150	28	0
Glass	9	214	2295	33
Breast Cancer Wisconsin (Diagnostic)	10	569	5110	75
Breast Cancer Wisconsin (Original)	10	699	5110	71
Abalone	8	4177	1016	102

5.6 Comparison

Now, we will see the comparison among our proposed algorithm FD-CFR, FD_Mine [23], and the TANE [11] algorithm in terms of the number of FDs checked. The comparison is only for the number of FD checks these algorithms are performing and does not show the efficiency. The proposed algorithm is built so that it can find meaningful dependencies (approximate dependencies) between different attributes and FDs, especially for numeric attributes, and also reduces the search space by finding only the normalized non-trivial approximate dependencies. The scope of our algorithm is in real-world applications and in its simplicity. FD_MINE and TANE search in a smaller space as FD_MINE uses equivalences to reduce dataset and FD search space, on the other hand, TANE searches all minimal non-trivial dependencies.

Table 5.21: Comparison on UCI Datasets

Dataset Name	**Number of Attributes**	**Number of Rows**	**Number of FD Checks**		
			Proposed algorithm FD-CFR	FD_MINE	TANE
Abalone	8	4177	1016	594	594
Breast Cancer Wisconsin (Original)	10	699	5110	4562	4562

Chapter 6

Conclusions and Future Scopes

This chapter deals with the conclusions drawn from the approach given in this work, its current implementation, and its limitations. Also, recommendations for future enhancements, additions, and improvements to the present work are given.

6.1 Summary

The proposed algorithm is based on fuzzy relational logic and clustering techniques applied to the relational schemas. Some example tables are taken as the first level of testing where the number of attributes and tuples is small. This gives us the opportunity to view the working of the approach in an easy-to-understand manner. Due to this small number of attributes, the FDs can be found manually and checked with the outcome of the algorithm.

Also, the parameter values (relationship weakness α) we receive for different scenarios of the same dataset or different datasets can be seen across various implication relations. Results obtained for cases where FDs exist, or approximate and meaningful dependencies are present with relatively small exceptions can be studied in a simple way.

The benchmark datasets used by many researchers in the field of designing relational databases, the discovery of functional dependencies from relations and the use of fuzzy logic in database designs are taken for standardizing the results. A comparative study with other algorithms and tools gives us a notion of how well this approach works in terms of run time, search space, and most importantly in finding FDs and approximate dependencies.

In the next section, the conclusions from the complete study of our proposed approach and already existing approaches in the literature are given below for the understanding of this work and the limitations that are holding back the proposed logic to be applied widely.

6.2 Conclusions

The results derived from the application of our algorithm and the study of the foundational logic used as the backbone for this proposal provide us with the opportunity to finally give conclusions about the current scope and limitations binding this approach from reaching its maximum effectiveness.

1. A fuzzy approximate reasoning approach for single and multiple antecedent clauses is used for the determination of FDs and approximate dependencies.

2. The clustering technique is used to group similar data and gaussian membership values are calculated for each entry in an attribute within a cluster.

3. The relational matrices are built for dependent and determinant sets within-cluster and not for all tuples in one shot, this gives us the scope to explore the possibility of data coming from different streams representing a similarity within groups.

4. The relationship weakness parameter (RMSE) is used as a metric for the calculation of dependencies found in a relation. But the threshold used here is based on the limited study done on example datasets and benchmark datasets. The threshold or cut-off values for this parameter can be selected more prominently with more testing and domain-specific analysis.

5. Currently, the algorithm is finding dependencies only on the numeric attributes as the effectiveness of the logic on categorical features could not be formalized due to the limited scope of the study.

6. Only gaussian membership is used here where the values give the deviation from the attribute mean; though it is a widely used membership function and gives a good representational value in terms of dispersion from the mean, there may be possibilities for better membership function selection or building according to the type, domain, and nature of data.

There are some other limitations also present in this approach which can further be improved in future works. The next section gives some of the recommendations for the refinement of the approach discussed here.

6.3 Recommendations for Future Works

The thorough analysis and understanding of the present approach present us with some scopes of enhancement, which can be done in the future for further improvements. These are:

1. Due to the limited study of the topic and works done by various researchers a more accurate measure of the relationship weakness parameter (**α**) could not be proven, the threshold values are taken solely from the experiments done during the work. An approach or mathematical method can be formulized for setting the thresholds of the parameter (**α**).
2. The value of **α** is taken as a measure of relationship weakness, i.e., the more the value the weak the relationship but it is not giving any quantifiable measure of how much the values differ in determinant and dependent sets which reduce the strength of dependency. A new parameter can be formalized or a mathematical explanation with a quantifiable study of **α** can be given which can give us a measure of dependency between two sets of attributes.
3. In this approach, only the numerical columns are taken for the finding for FDs, a more rigorous study and experiments can bring methods to transform the categorical attributes into numerical values without hampering the simplicity of the algorithm.
4. Domain-specific studies and literary surveys can give ways to automatically build better membership functions for attributes for relations representing different industries/sectors and different optimal membership functions for attributes within a relation.
5. The proposed approach checks only the normalized and non-trivial FDs which reduces the search space by a large margin, but the search space can further be reduced by finding not only the normalized and non-trivial but also minimal FDs using a new algorithm or finding equivalences as given in FD_Mine algorithm [23].

In the future, an automatic functional dependency finding toolbox or algorithm can be developed which includes a fuzzy inference system and reasoning to find relationship strengths between different sets of features. The system can be made robust by incorporating AI techniques. As we know there are lots of fuzzy implication relations are available and different types of membership functions are also available, it can be made to derive a unique membership function to describe the data set and then use comparative analysis between implication relations and further optimize the logic to make the decision whether the relationship is strong enough or not.

References

[1] Zhu, X., Huang, Z., Yang, S., & Shen, G. (2007, August). Fuzzy implication methods in fuzzy logic. In Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) (Vol. 1, pp. 154-158). IEEE.

[2] Galindo, J. (2008). Introduction and trends to fuzzy logic and fuzzy databases. In Handbook of Research on Fuzzy Information Processing in Databases (pp. 1-33). IGI Global.

[3] Dubois, D., Lang, J., & Prade, H. (1991). Fuzzy sets in approximate reasoning, part 2: logical approaches. Fuzzy sets and systems, 40(1), 203-244.

[4] Dubois, D., & Prade, H. (1991). Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distributions. Fuzzy sets and systems, 40(1), 143-202.

[5] Baldwin, J. F., & Pilsworth, B. W. (1980). Axiomatic approach to implication for approximate reasoning with fuzzy logic. Fuzzy sets and systems, 3(2), 193-219.

[6] Konar, A. (2018). Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain. CRC press.

[7] Housel, B. C., Waddle, V., & Yao, S. B. (1979, October). The functional dependency model for logical database design. In Fifth International Conference on Very Large Data Bases, 1979. (pp. 194-208). IEEE.

[8] Konar, A. (2006). Computational intelligence: principles, techniques and applications. Springer Science & Business Media.

[9] Ali, O. A. M., Ali, A. Y., & Sumait, B. S. (2015). Comparison between the effects of different types of membership functions on fuzzy logic controller performance. International Journal, 76, 76-83.

[10] Yao, H., & Hamilton, H. J. (2008). Mining functional dependencies from data. Data Mining and Knowledge Discovery, 16(2), 197-219.

[11] Huhtala, Y., Kärkkäinen, J., Porkka, P., & Toivonen, H. (1999). TANE: An efficient algorithm for discovering functional and approximate dependencies. The computer journal, 42(2), 100-111.

[12] Hameed, I. A. (2011). Using Gaussian membership functions for improving the reliability and robustness of students' evaluation systems. Expert systems with Applications, 38(6), 7135-7142.

[13] Laha, M., Konar, A., Rakshit, P., & Nagar, A. K. (2019). Exploration of Subjective Color Perceptual-Ability by EEG-Induced Type-2 Fuzzy Classifiers. IEEE Transactions on Cognitive and Developmental Systems, 12(3), 618-635.

[14] Raju, K. V. S. V. N., & Majumdar, A. K. (1988). Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. ACM Transactions on Database Systems (TODS), 13(2), 129-166.

[15] Ilyas, I. F., Markl, V., Haas, P., Brown, P., & Aboulnaga, A. (2004, June). CORDS: Automatic discovery of correlations and soft functional dependencies. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (pp. 647-658).

[16] Guilly, M. L., Petit, J. M., & Scuturici, V. M. (2020). Evaluating Classification Feasibility Using Functional Dependencies. In Transactions on Large-Scale Data-and Knowledge-Centered Systems XLIV (pp. 132-159). Springer, Berlin, Heidelberg.

[17] Smink, R. (2021). Using functional dependency thresholding to discover functional dependencies for data cleaning (Bachelor's thesis, University of Twente).

[18] Chu, X., Ilyas, I. F., & Papotti, P. (2013, April). Holistic data cleaning: Putting violations into context. In 2013 IEEE 29th International Conference on Data Engineering (ICDE) (pp. 458-469). IEEE.

[19] Berti-Equille, L., Harmouch, H., Naumann, F., Novelli, N., & Saravanan, T. (2018, August). Discovery of genuine functional dependencies from relational data with missing values. In The 44th International Conference on Very Large Data Bases (VLDB) (Vol. 11, No. 8).

[20] Guo, Z., & Rekatsinas, T. (2019). Learning functional dependencies with sparse regression. arXiv preprint arXiv:1905.01425.

[21] Abraham, J., & Priya, R. (2017). Mining Approximate Functional Dependencies from Large Databases Based on Concept Similarities to Answer Imprecise Queries. International Journal of Pure and Applied Mathematics, 116(21), 651-660.

[22] Nambiar, U., & Kambhampati, S. (2004, June). Mining approximate functional dependencies and concept similarities to answer imprecise queries. In Proceedings of the 7th International Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004 (pp. 73-78).

[23] Yao, H., Hamilton, H. J., & Butz, C. J. (2002, December). FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences. In ICDM (pp. 729-732).

[24] Al-Hamouz, S., & Biswas, R. (2006). Fuzzy functional dependencies in relational databases. International Journal of computational cognition, 4(1), 39-43.

[25] UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[26] Wosiak, A., & Zakrzewska, D. (2018). Integrating correlation-based feature selection and clustering for improved cardiovascular disease diagnosis. Complexity, 2018.

Appendix A

ALGORITHM FD-CFR

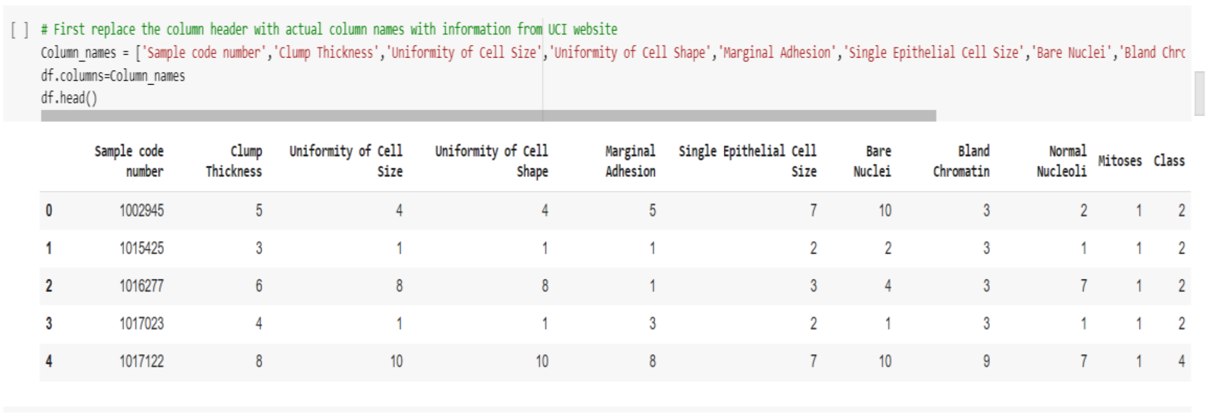
```
BEGIN
1. READ fuzzy membership values of each entry in the dataset into
   membershipData.
2. GET columns names of the dataset and store into a LIST called attributes.
     a. Using the LIST attributes COMPUTE all possible set of attributes possible
        and store into setAttributes.
     b. REMOVE the empty set and the set containing all attributes from
        setAttributes.
3. INIT fdCheck= 0.
4. INIT fdFound=0.
5. SET fdCheckSet = setAttributes.
6. FOR antecedentSet in fdCheckSet
     6.1 SET the LIST dependentSet ← attributes do not present in the antecedentSet.
     6.2 FOR dependent in dependentSet
          6.2.1  INIT error ← 0
          6.2.2  FOR cluster = 0 to (number of cluster) – 1
                   6.2.2.1 INIT the LIST listAntecedent ← attributes in
                       antecedentSet.
                   6.2.2.2 CALCULATE antecedent membership AM from
                       listAntecedent.
                   6.2.2.3 GET consequent membership CM = dependent
                       membership values       from membershipData.
                   6.2.2.4 COMPUTE the relational matrix with AM, CM.
                   6.2.2.5 COMPUTE $\acute{CM}$ = fuzzy composition of AM, CM
                   6.2.2.6 CALCULATE root mean square error of CM and $\acute{CM}$
                       and ADD to error.

                 ENDFOR

          6.2.3  INCREMENT fdCheck.
          6.2.4  IF error < $\alpha$
                   6.2.4.1 INCREMENT fdFound
          6.2.5 ENDIF
          6.2.6 PRINT error for antecedentSet and dependent.
     6.3 ENDFOR
7. ENDFOR
8. PRINT fdCheck
9. PRINT fdFound
END
```

Appendix B

Some outputs generated during the code run is given here to give an idea of how the actual output looks like when codes are executed.

B.1 Associated software Requirements:

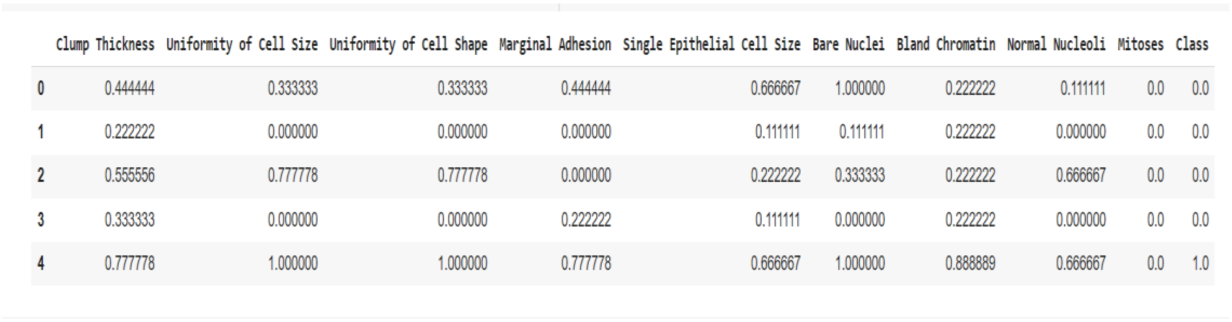
- Complete work is done using Python programming language.
- The platform used to write code for this thesis is Google Colab, which is based on Jupyter open source. It does not require any installation as Google Colab is a hosted Jupyter notebook service, just access to an internet connection and one google account is enough.
- While the experimentation was done, the python version in Google Colab was Python 3.7.13.
- Though we do not need any software installation to run the codes, we need to import the necessary libraries into the Colab environment and install fuzzy toolkits scikit-fuzzy and pyit2fls.



```
[ ] # First replace the column header with actual column names with information from UCI website
    Column_names = ['Sample code number','Clump Thickness','Uniformity of Cell Size','Uniformity of Cell Shape','Marginal Adhesion','Single Epithelial Cell Size','Bare Nuclei','Bland Chrc
    df.columns=Column_names
    df.head()
```

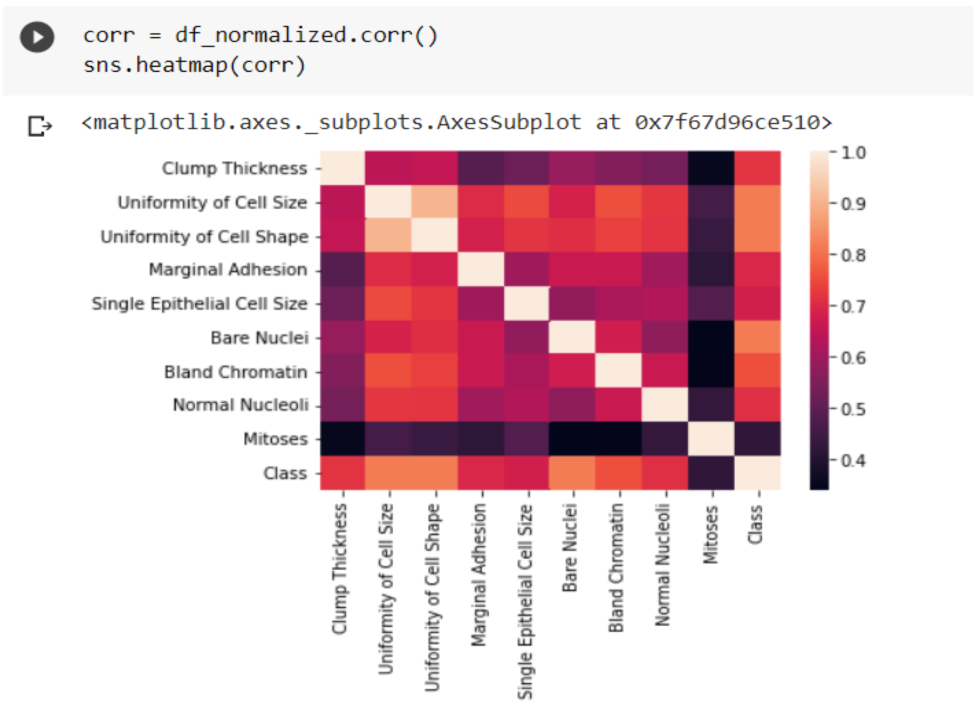
	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1002945	5	4	4	5	7	10	3	2	1	2
1	1015425	3	1	1	1	2	2	3	1	1	2
2	1016277	6	8	8	1	3	4	3	7	1	2
3	1017023	4	1	1	3	2	1	3	1	1	2
4	1017122	8	10	10	8	7	10	9	7	1	4

Fig B.1: Breast Cancer Wisconsin (Original) Dataset loaded into Google Colab Notebook



	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	0.444444	0.333333	0.333333	0.444444	0.666667	1.000000	0.222222	0.111111	0.0	0.0
1	0.222222	0.000000	0.000000	0.000000	0.111111	0.111111	0.222222	0.000000	0.0	0.0
2	0.555556	0.777778	0.777778	0.000000	0.222222	0.333333	0.222222	0.666667	0.0	0.0
3	0.333333	0.000000	0.000000	0.222222	0.111111	0.000000	0.222222	0.000000	0.0	0.0
4	0.777778	1.000000	1.000000	0.777778	0.666667	1.000000	0.888889	0.666667	0.0	1.0

Figure B.2: Breast Cancer Wisconsin (Original) Dataset normalized using min-max normalization method



```
corr = df_normalized.corr()
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f67d96ce510>
```

Figure B.3: Correlation coefficients between different features of Breast Cancer Wisconsin (Original) Dataset

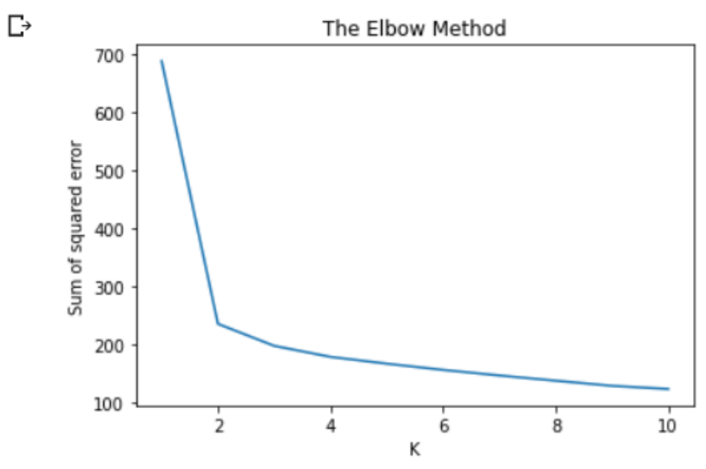


Figure B.4: Plot of sum of squares vs. K for K-Means clustering of Breast Cancer Wisconsin (Original) Dataset

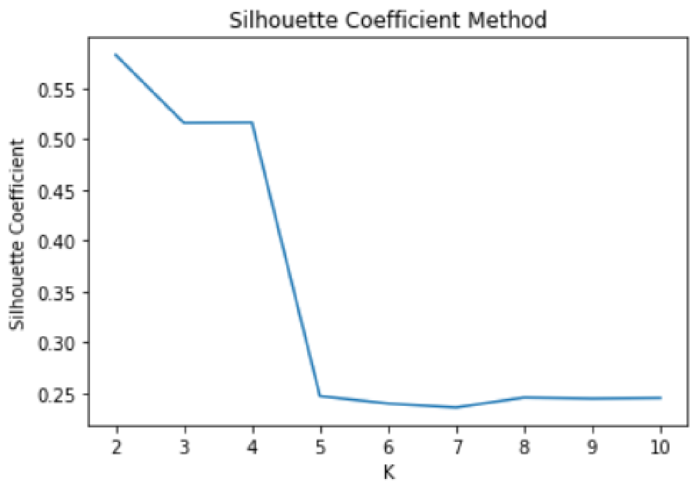


Figure B.5: Plot of sum of Silhouette coefficient vs. K for K-means clustering of Breast Cancer Wisconsin (Original) Dataset