People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of 8 May 1945-Guelma-

Faculty of Mathematics, Computer Science and Science of Matter
Department of Computer Science

Master Thesis

Specialty : Computer Science

Option : Information Systems

Theme :

A method for detecting spam emails based on fuzzy logic

Presented by : Amoura Mohcene

Jury Members :

N	Full Name	Quality
1	Dr. Hannousse Abdelhakim	Chairman
2	Dr. Farek Lazhar	Supervisor
3	Dr. Benhamida Nadjette	Examiner

June 2024

Acknowledgements

First and foremost, I thank God for everything he has given me. Then, I would like to thank every person who helped me, even in the slightest, be it a family member, friend or a teacher, may God bless them all.

I would like to express my deepest gratitude to my supervisor, **Dr. Farek Lazhar**, for his invaluable guidance, support, and encouragement throughout this research. His expertise and insights were instrumental in shaping this thesis and achieving its objectives.

I would also like to thank the faculty members of the Department of Computer Science at the **University of 8 May 1945-Guelma** for their academic support and for providing a conducive environment for research. Additionally, I extend my appreciation to my family and friends for their unwavering support and understanding during this challenging journey.

Thank you to everyone who contributed to this work and helped make it possible.

Abstract

Spam emails are an ever-increasing issue in the digital communication world, posing significant threats globally, ranging from privacy breaches to financial losses and the spread of malicious software. The sheer volume and evolving sophistication of spam necessitate robust detection systems to protect users and organizations. In this study, we introduce an advanced approach to spam email detection using the Improved Fuzzy K-Nearest Neighbors (IFKNN) algorithm. The primary objective is to enhance the accuracy and robustness of email classification systems in distinguishing spam from legitimate messages. Our method integrates fuzzy logic principles to account for the uncertainty and vagueness inherent in spam classification. We thoroughly evaluate the performance of IFKNN by comparing it with traditional algorithms such as KNN and Support Vector Machines (SVM) on real-world spam datasets. The experimental results demonstrate that IFKNN significantly outperforms existing methods in terms of accuracy, precision, recall, and F1-score, thus showcasing its potential in advancing cybersecurity measures.

Keywords: Spam Detection, Fuzzy Logic, Text Classification, KNN, Improved Fuzzy K-Nearest Neighbors (IFKNN), Machine Learning, Email Classification.

Résumé

Les emails indésirables représentent un problème en constante augmentation dans le monde des communications numériques, posant des menaces significatives à l'échelle mondiale, allant des violations de la vie privée aux pertes financières et à la propagation de logiciels malveillants. Le volume considérable et la sophistication croissante des spams nécessitent des systèmes de détection robustes pour protéger les utilisateurs et les organisations. Dans cette étude, nous présentons une approche avancée pour la détection des emails indésirables en utilisant l'algorithme des K-plus-proches-voisins flous amélioré (IFKNN). L'objectif principal est d'améliorer la précision et la robustesse des systèmes de classification des emails pour distinguer les spams des messages légitimes. Notre méthode intègre les principes de la logique floue pour prendre en compte l'incertitude et le flou inhérents à la classification des spams. Nous évaluons minutieusement les performances de l'IFKNN en le comparant à des algorithmes traditionnels tels que KNN et les machines à vecteurs de support (SVM) sur des ensembles de données de spam réelles. Les résultats expérimentaux montrent que l'IFKNN surpasse significativement les méthodes existantes en termes de précision, de rappel et de score F1, démontrant ainsi son potentiel dans l'avancement des mesures de cybersécurité.

Mots-clés : Détection de Spam, Logique Floue, Classification de Texte, KNN, K-plus-proches-voisins flous amélioré (IFKNN), Apprentissage Automatique, Classification des Emails.

Contents

List of Figures

List of Tables

List of Abbreviations

BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
BOW	**B**ag **O**f **W**ords
ELMO	**E**embeddings from **L**anguage **MO**del
GLoVe	**Glo**bal **Ve**ctors for Words Representations
KNN	**K** **N**earest **N**eighbor
FKNN	**F**uzzy **KNN**
IFKNN	**I**mproved **F**uzzy **KNN**
RFC	**R**andom **F**orest **C**lassifier
SVM	**S**upport **V**ector **M**achines
TF	**T**erm **F**requency
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency

General Introduction

The widespread presence of spam emails is a significant challenge in today's digital communication landscape. These unsolicited messages clutter inboxes and pose risks such as phishing attacks, malware distribution, and privacy breaches. With the increasing volume and sophistication of spam, there is a growing need for effective detection systems to filter out these unwanted messages reliably. Traditional spam detection methods, which often rely on rule-based systems and basic machine learning algorithms, may not be sufficient to address the complexity of modern spam emails.

This research explores the use of the Improved Fuzzy K-Nearest Neighbors (IFKNN) algorithm, which integrates fuzzy logic to improve classification accuracy. Fuzzy logic helps model the uncertainty and imprecision in spam classification, allowing for more nuanced decision-making. By combining the strengths of fuzzy logic with the established K-Nearest Neighbors (KNN) algorithm, IFKNN aims to provide better performance in identifying spam. The goal of this research is to contribute to the field of spam detection by offering an innovative and effective solution that enhances the security and efficiency of email communication systems.

This master's thesis is organized as follows:

— *Chapter 1. Fuzzy Logic: Fundamental Concepts*: This chapter introduces the main concepts and definitions of fuzzy logic.

— *Chapter 2. Text Classification*: This chapter covers the fundamentals of text classification, including definitions, key challenges, and various text representation techniques such as Bag-of-Words, TF-IDF, and word embeddings. It also discusses traditional text classification algorithms, machine learning techniques, and deep learning models.

— *Chapter 3. Enhancing Spam Email Detection through Fuzzy KNN Algorithm*: This chapter presents the architecture of our system with detailed explanations.

— *Chapter 4. Implementation*: This chapter details the main components of our code and presents the results of our research.

The thesis also includes an abstract, a general introduction, and a general conclusion.

Chapter 1

Fuzzy Logic: Fundamental Concepts

1.1 Introduction

Fuzzy Logic is a mathematical framework developed by Lotfi A. Zadeh in the 1960s to address the limitations of classical, or binary, logic in dealing with uncertainty and imprecision. Classical logic is based on a strict true/false dichotomy, where statements are either true (1) or false (0). However, in many real-world scenarios, information is often vague, incomplete, or uncertain, and classical logic struggles to capture this complexity. The motivation behind the development and adoption of Fuzzy Logic stems from the need to model and mimic human decision-making processes. In many real-world situations, human judgments are not black and white but involve shades of gray. Fuzzy Logic provides a way to capture and formalize this inherent uncertainty, making it a valuable tool in various fields. This chapter aims to introduce the key components and principles of fuzzy logic, offering clarity on its origins, core elements, and its role in addressing the challenges of uncertainty and imprecision.

1.2 Definitions and Fundamentals

1.2.1 Fuzzy Sets

1.2.1.1 Definition

Fuzzy logic is based on the theory of fuzzy sets, which is a generalization of the classical set theory. Saying that the theory of fuzzy sets is a generalization of the classical set theory means that the latter is a special case of fuzzy sets theory [1]. To make a metaphor in set theory speaking, the classical set theory is a subset of the theory of fuzzy sets, as Figure 1.1 illustrates [1].

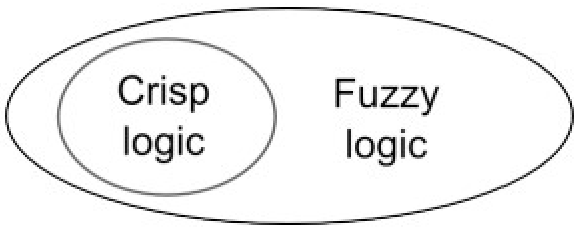


FIGURE 1.1 – Classical Set Theory as a Subset of Fuzzy Set Theory.

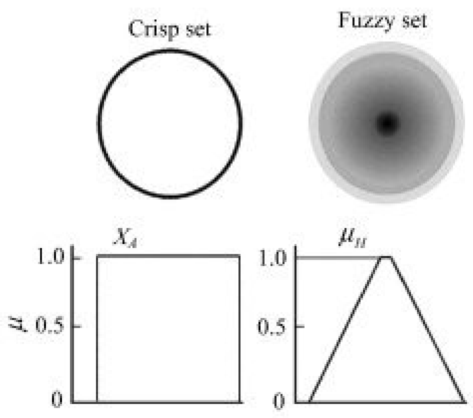


FIGURE 1.2 – Crisp Set vs Fuzzy Set [6].

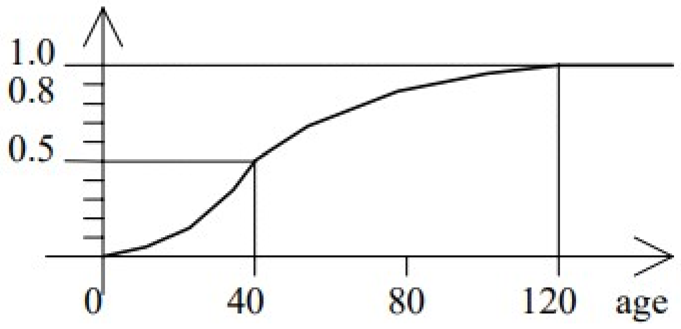


FIGURE 1.3 – An Example of Membership Functions [4].

The classical set theory is built on the fundamental concept of "set" of which an individual is either a member or not a member [4]. A sharp, crisp, and unambiguous distinction exists between a member and a nonmember for any well-defined "set" of entities in this theory, and there is a very precise and clear boundary to indicate if an entity belongs to the set [4], fuzzy set theory accepts partial memberships, and, therefore, in a sense generalizes the classical set theory to some extent [4].

A fuzzy set is a concept in fuzzy logic that extends the idea of classical sets by allowing elements to have degrees of membership rather than a binary, all-or-nothing membership. In a fuzzy set, each element belongs to the set to a certain degree between 0 and 1, indicating the strength of its membership. This degree of membership reflects the extent to which an element possesses the characteristic described by the fuzzy set [1].

Figure 1.2 shows the difference between Crisp set and Fuzzy set.

1.2.1.2 Membership Functions

The Membership Functions (MF) are the fundamental units of fuzzy set theory, and they are what determine how fuzzy a set is. Because they have an impact on a fuzzy inference system, MF shapes are crucial for a specific task. They could be triangular, trapezoidal, Gaussian, or any number of other shapes. Really, the sole prerequisite for an MF is that it must range between 0 and 1 [2].

Example: Figure 1.3 shows an example of membership functions.

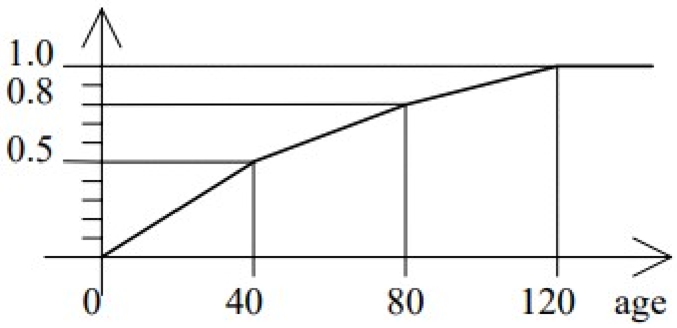


FIGURE 1.4 – Another Example of Membership Functions [4].

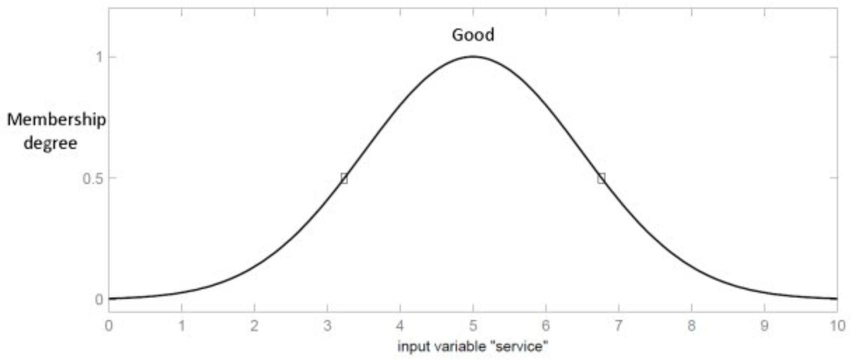


FIGURE 1.5 – Membership Function Characterizing the Subset of 'Good' Quality of Service [1].

An other example is illustrated in Figure 1.4 .

— The concept of "old" is not well-defined and cannot be precisely measured, so the set of "old" people (denoted as Sf) is not a classical, well-defined set [4].

— To make the set Sf well-defined, the concept of "old" needs to be quantified and characterized using a membership function [4].

— The membership function shown in Figure 1.4 is a mathematical measure for the "oldness" of a person, where a person of 40 years old is considered "old" with a degree of 0.5 and "young" with a degree of 0.5 [4].

— This membership function is a generalization of the classical characteristic function (which only assigns 0 or 1 to indicate non-membership or membership, respectively), allowing for partial membership [4].

— An alternative piecewise linear membership function, can also be used to describe the same concept of "oldness" for the set Sf [4].

— The selection of a membership function is subjective and depends on the user's knowledge, experience, and the specific application, similar to assuming a Gaussian noise distribution in classical probability theory [4].

A fuzzy set consists of two components: a subset and a membership function associated with it, which is different from classical set theory where all sets share the same two-valued characteristic function [4].

Example: Figure 1.5 shows the membership function chosen to characterize the subset of 'good' quality of service. In our tip example, we will redefine membership functions for each fuzzy set of each of our three variables [1]:

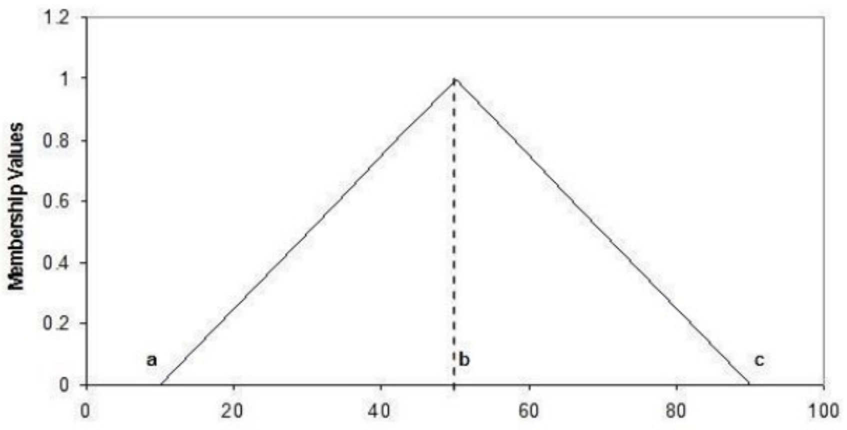


FIGURE 1.6 – Triangular Fuzzy MF [6].

— Input 1: quality of service. Subsets: poor, good and excellent.

— Input 2: quality of food. Subsets: awful and delicious.

— Output: tip amount. Subsets: low, medium and high.

The shape of the membership function is chosen arbitrarily by following the advice of the expert or by statistical studies: sigmoid, hyperbolic, tangent, exponential, Gaussian or any other form can be used [1].

The membership function is a graphical representation of the magnitude of participation of each input. The function is used to associate a degree of membership of each of the elements of the domain of the corresponding fuzzy set. Membership functions for fuzzy sets can be of any shape or type as determined by experts in the domain over which the sets are defined [3]. Those functions must satisfy the following constraints:

— A membership function has a lower limit of 0 and upper limit of 1 that is a range of [0, 1] [3].

— For each $x \in X$, $\mu_A(x)$ have to be sole [3].

The graphical representation may include different shapes such as:

1.2.1.3 Types of membership functions

1. **Triangular Membership Function:**
   Triangular MF (Figure 1.6) one of the simplest and most commonly used MFs for designing FLS [6]. The fundamental attributes of triangular MFs which differentiate it from other MFs are:

   — Here the boundary varies linearly from highest to lowest membership grade [6].

   — There's only 1 discrete element having the highest membership grade [6].
   A triangular MF is characterized by Equation 1.1.

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a < x \le b \\ (x-c)/(b-c), & b < x < c \\ 0, & x \ge c \end{cases} \tag{1.1}$$

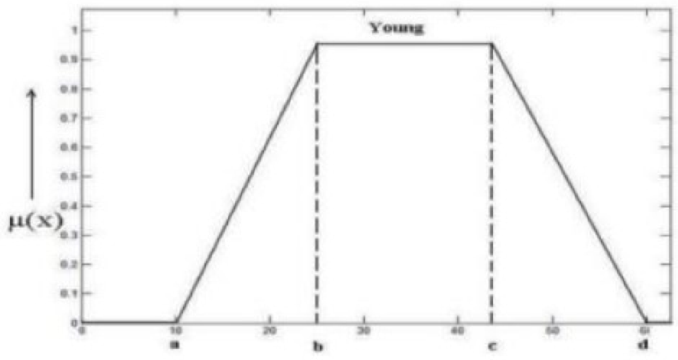


FIGURE 1.7 – Trapezoidal MF [6].

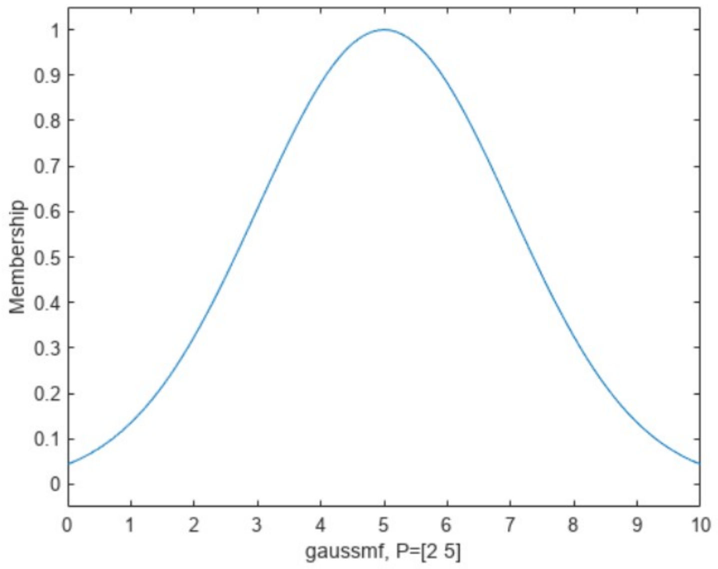


FIGURE 1.8 – Gaussian MF [6].

2. **Trapezoidal Membership Function:**

   Trapezoidal MF Like triangular, the trapezoidal also has a linear boundary for FS; however, a trapezoidal MF is characterized by a range of elements having maximum membership grade [6].
   Trapezoidal MF is represented by Equation 1.2 and illustrated by Figure 1.7.

$$\mu_A(x) = \left\{ \begin{array}{cr} 0, & x < a \\ (x-a)/(b-a), & a < x \leq b \\ 1 & b < x < c \\ (d-x)/(d-c), & c \leq x \leq d \\ 0, & x \geq d \end{array} \right\} \tag{1.2}$$

3. **Gaussian MF:**

   Gaussian MF (Figure) characterized by non-linear boundaries for MF variation and are defined as shown in Equation 1.3.

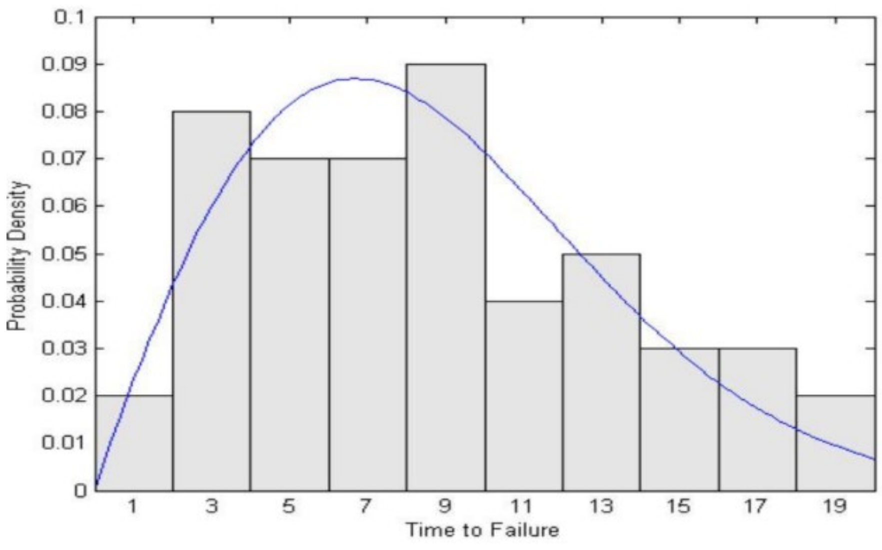


FIGURE 1.9 – Probability density function-based MF [6].

$$\mu_A(x) = e^{-(x-c)^{2/}\sqrt{(2\sigma)}} \tag{1.3}$$

Here, σ denotes the standard deviation, and is considered as a curve-fitting constant, governing the shape of the MF [6].
Probability Density Function (Figure 1.9) – This FS is obtained from the probability distribution of a variable [6]. The MF function is given by Equation 1.4.

$$\mu(x) = 1 - \frac{x}{a}; x \in [0, a] \tag{1.4}$$

For a nonlinear function, the linearized form is represented as shown in Equation 1.5.

$$\mu(x) = \alpha\left(1 - e^{(b-x)(b-a)}\right), x \in [a, b] \tag{1.5}$$

1.2.1.4 Features of Membership Function

There are three standard regions defined in the membership functions. They are Core, Support and Boundary. The membership functions can be symmetrical or asymmetrical with membership value between 0 and 1.
Core: The elements, which have the membership function as 1, are the elements of the core (Equation 1.6).

$$core(A) = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{1.6}$$

Support: The support has the elements whose membership is greater than 0 (Equation 1.7).

$$\sup\ port\ (A) = \{x \in X \mid \mu_A(x) > 0\} \tag{1.7}$$

Boundary: The boundary has the elements whose membership are between 0 and 1 (Equation 1.8).

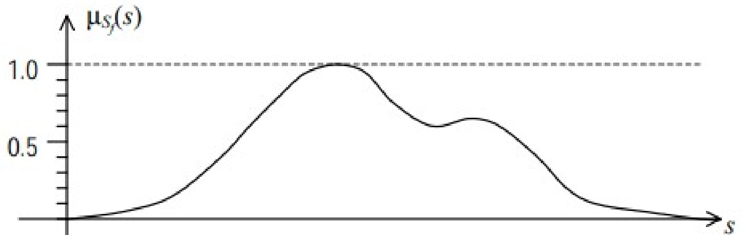


FIGURE 1.10 – A membership function associated with a fuzzy subset [4].

$$boundary(A) = \{x \in X \mid 0 < \mu_A(x) < 1\} \tag{1.8}$$

Height: The height of the fuzzy set A is the maximum value of the membership function as given by Equation 1.9.

$$hight(A) = \{x \in X \mid \max(\mu_A(x))\} \tag{1.9}$$

Center: there are four different situations that define the center of a fuzzy set. When the membership function of a fuzzy set reaches:

— Its maximum at only one element of the universe of discourse, the element is called the center of the fuzzy set.

— Attains it is greatest at more than one element of the universe of discourse and all these elements are bound, the middle point of the element is the center.

— Attains its maximum at more than one element of the universe of discourse and not all the elements are bound, the largest element is the center if it bounded otherwise, the smallest element is the center [3].

1.2.1.5 Operations on Fuzzy sets

Notions like equality and inclusion of two fuzzy sets are directly derived from ordinary set theory. In contrast to this, different definitions are possible for connectives like intersection, union, or complement, which only coincide for the special case of ordinary sets [18]. This may be an advantage, as it allows for a flexible and context-dependent choice of them, but it is also a challenge for the user to choose the appropriate ones for an application at hand [18].

1. **Fuzzy Subsets:**
   Let Sf be a fuzzy subset defined in a universe set S, together with a membership function $\mu_{Sf}(s)$ [4], as shown in Figure 1.10 [4].

$$S_\alpha = \{s \in Sf \mid \mu_{Sf}(s) > \alpha\}\alpha \in [0,1) \tag{1.10}$$

$$S_{\overline{\alpha}} = \{s \in Sf \mid \mu_{Sf}(s) \geq \alpha\}\alpha \in [0,1) \tag{1.11}$$

2. **Fuzzy Numbers and their Arithmetic:** For a normal fuzzy subset, where the membership function is convex and achieves the maximum number 1 with a convex fuzzy subset, if its weak α-cut (α level-set) is a closed interval, then it is called a fuzzy number. Hence, a fuzzy number is a convex fuzzy subset, which has a normalized membership function and represents an interval of confidence [4].

Note: the fuzzy subset, with subset $[0, \infty)$ and the associate fuzzy membership function $\mu(x) = 1 - e^{-x}$, is considered to be normal. Although this membership function does not attain the maximum value 1, it approaches 1 as a limit [4].

1.3 Fuzzy Logic

1.3.1 Fuzzy Propositions

1.3.1.1 Definition

The fundamental difference between classical propositions and fuzzy propositions is in the range of their truth values. While each classical proposition is required to be either true or false, the truth or falsity of fuzzy propositions is a matter of degree. Assuming that truth and falsity are expressed by values 1 and 0, respectively, the degree of truth of each fuzzy proposition is expressed by a number in the unit interval [0, 1] [8].

Example 1: The following sentences are propositions:
Smith hits 30 home runs in one season. (*true*)
2 + 4 = 7. (*false*)
For every x, if f(x) = sin x, then fc(x) = cos x. (*true*)
It rains now. (*true*)

Example 2: The followings are not propositions:
Why are you interested in the fuzzy theory?
He hits 5 home runs in one season.
x + 5 = 0
x + y = z
In the second example, we do not know who is "He" and thus cannot determine whether the sentence is true (1) or false (0). If "He"is replaced by "Tom", we have Tom hits 5 home runs in one season [8].

1.3.1.2 Types of Fuzzy propositions

Fuzzy propositions are classified into the following four types:

— Unconditional and unqualified fuzzy propositions.

— Unconditional and qualified propositions.

— Conditional and unqualified propositions.

— Conditional and qualified propositions.

a) Unconditional and unqualified fuzzy propositions

Definition: The canonical form of fuzzy propositions of this type, p, is expressed by the sentence: $p : VisF$, where V is a variable that takes values v from some universal set V, and F is a fuzzy set on V that represents a fuzzy predicate, such as tall, expensive, low, normal, and so on.

Given a particular value of V (say, v), this value belongs to F with membership grade F(v). This membership grade is then interpreted as the degree of truth, T(p), of proposition p. That is: $T(p) = F(v)$ for each given particular value v of variable V

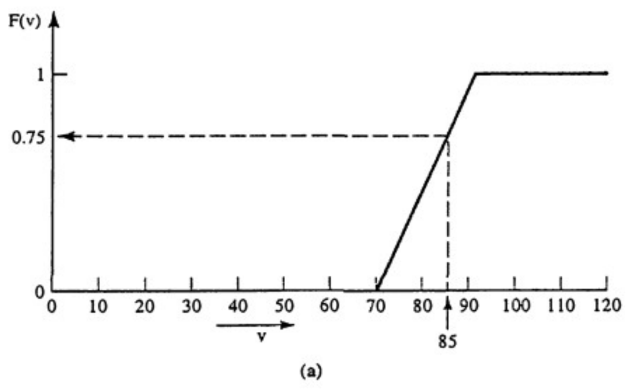


FIGURE 1.11 – Components of the Fuzzy Proposition p: Temperature (V) is High (F) [8].

in proposition p. Thus, truth value of the proposition depends on a fuzzy set on [0, 1], which assigns the membership grade F(v) to each value v of variable V [8].

Example : To illustrate the introduced concepts, let variable V be the air temperature at some particular place on the Earth (measured in F) and let the membership function shown in Figure 1.11 a represent, in a given context, the predicate high. Then, assuming that all relevant measurement specifications regarding the temperature are given, the corresponding fuzzy proposition, p, is expressed by the sentence [8].

p: Temperature (V) is High (F).

b) Unconditional and Qualified Fuzzy Propositions

Definition: Propositions p of this type is characterized by either the canonical form p: V is F is S, or the canonical form: p: Pro V is F is P, where V is a variable that takes values v from some universal set V, and F is a fuzzy set on V that represents a fuzzy predicate, Pro V is F is the probability of fuzzy event ”V is F”, S is a fuzzy truth qualifier, and P is a fuzzy probability qualifier [8].

Example: Truth-qualified proposition is the proposition “Geeta is young is very true”, where the predicate young and the truth qualifier very true are represented by the respective fuzzy sets shown in Figure 1.12 [8].

— Assuming that the age of Geeta is 26, she belongs to the set representing the predicate young with the membership grade 0.87.

— Proposition belongs to the set of propositions that are very true with membership grade 0.76.

— The degree of truth of our truth-qualified proposition is also 0.76.

— The proposition was modified by changing the predicate (e.g., to very young) or the truth qualifier (e.g., to fairly true, very false, etc.).

— In general, the degree of truth, T(p), of any truth-qualified proposition p is given for each $v \in V$ by the equation $T(p) = S(F(v))$.

— Viewing the membership function $G(v) = S(F(v))$, where $v \in V$, as a simple predicate, we can interpret any truth-qualified proposition of the form p: V is F is S, as the unqualified proposition “V is G” [8].

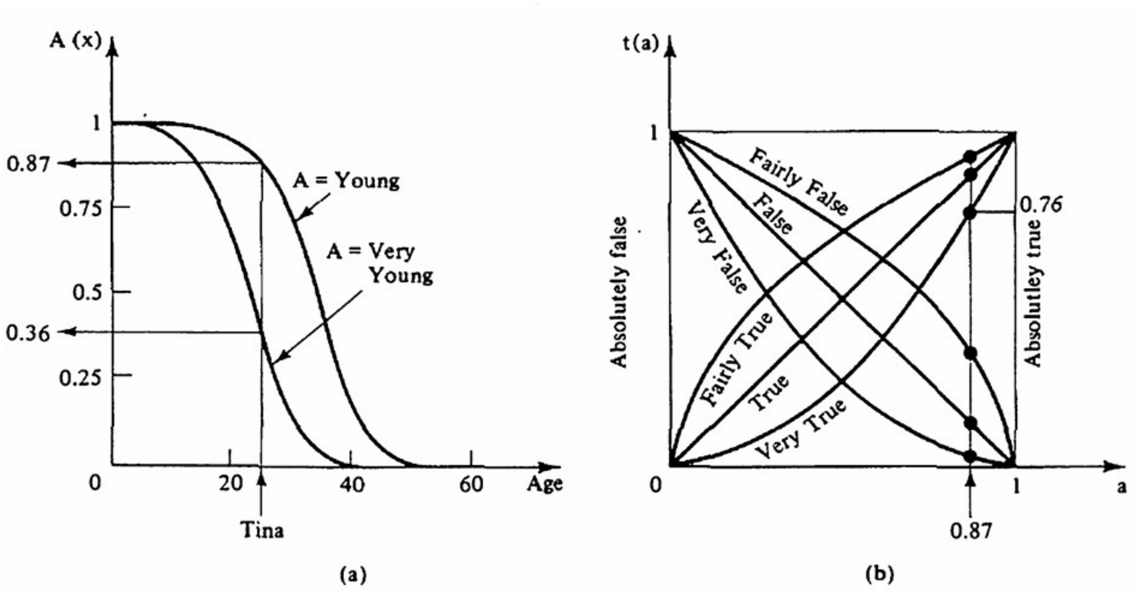


FIGURE 1.12 – Truth values of a fuzzy proposition [8].

— Unqualified propositions are special truth-qualified propositions, in which the truth qualifier S is assumed to be true.

— As in Figure 1.11.b and Figure 1.12.b the membership function representing this qualifier is the identity function. That is, $S(F(v)) = F(v)$ for unqualified propositions; Hence, S may be ignored for the sake of simplicity [8].

c) Conditional and Unqualified Propositions Propositions p of this type is expressed by the canonical form:

p: If X is A, then Y is B.

Where X, d are variables whose values are in sets X, Y, respectively, and A, B are fuzzy sets on X, Y, respectively. These propositions may also be viewed as propositions of the form: X, Y is R.

R is a fuzzy set on X * Y that is determined for each x E X and each y E Y by the formula: $R(x, y) = \alpha[A(x), B(y)]$, where a denotes a binary operation on [0, 1] representing a suitable fuzzy implication [8].

d) Conditional and Qualified Propositions Propositions of this type can be characterized by either the canonical form:

p: If X is A, then Y is B is S,

or the canonical form:

p: Pro X is A|Y is B is P, where Pro X is A|Y is B is a conditional probability.

Since methods introduced for the other types of propositions can be combined to deal with propositions of this type, we do not deem it necessary to discuss them further [8].

1.3.2 Truth values

We now argue, after a brief discussion of consistency in both classical logic and multiple-valued logics, that certain subintervals of truth values in the real unit interval, analogous to Zadeh's linguistic truth-values, can be differentiated from others. We then use our classification of the unit interval of truth values to define

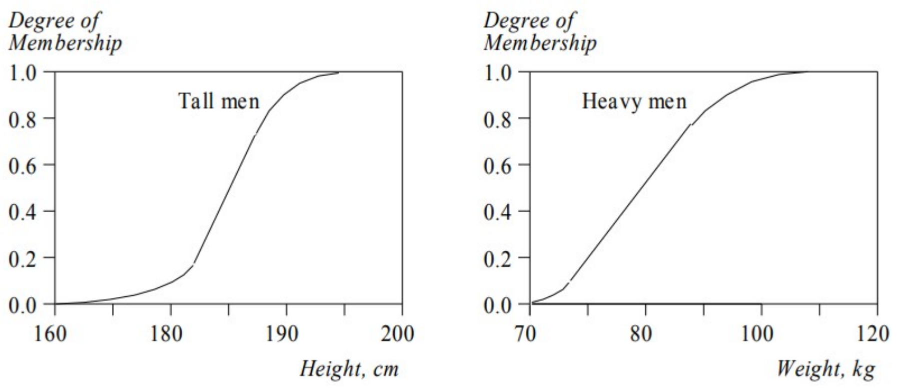


Figure 1.13 – Example Fuzzy Rules [5].

consistency for a fuzzy logic and, finally, prove that the Kenevan Truth Interval Fuzzy Propositional Logic is consistent under this definition.

The concept of "truth value" in fuzzy logic refers to the degree of truth of a proposition, which can be any real number within the inclusive unit interval [0.0, 1.0]. This range allows for varying degrees of truth rather than a binary true/false valuation found in classical logic [9].

Kenevan Truth Interval Fuzzy Logic: This variant represents truth values as subintervals within the unit interval [0.0, 1.0] rather than single values. For instance, a proposition A with an interval truth value is written as $A : [\alpha_0, \alpha_1]$, where $[\alpha_0, \alpha_1]$ contains the actual truth value α [9].

1.3.3 Fuzzy Rules

A fuzzy rule can be defined as a conditional statement in the form:

IF x is A THEN y is B

where x and y are linguistic variables; and A and B are linguistic values determined by fuzzy sets on the universe of discourses X and Y, respectively [5].

Firing Fuzzy Rules: These fuzzy sets provide the basis for a weight estimation model. As shown in Figure 1.13 , the model is based on a relationship between a man's height and his weight:

IF height is tall THEN weight is heavy.

A fuzzy rule can have multiple antecedents, for example [5]:

IF project_duration is long **AND** project_staffing is large **AND** project_funding is inadequate **THEN** risk is high

IF service is excellent **OR** food is delicious **THEN** tip is generous

The consequent of a fuzzy rule can also include multiple parts, for instance [5]:

IF temperature is hot **THEN** hot_water is reduced; cold_water is increased.

1.4 Fuzzy Inference Systems

1.4.1 Definitions

1.4.1.1 Definition 1

The fuzzy inference system is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. It has found successful applications in a wide variety of fields, such as automatic control, data classification, decision analysis, expert systems, time series prediction, robotics, and

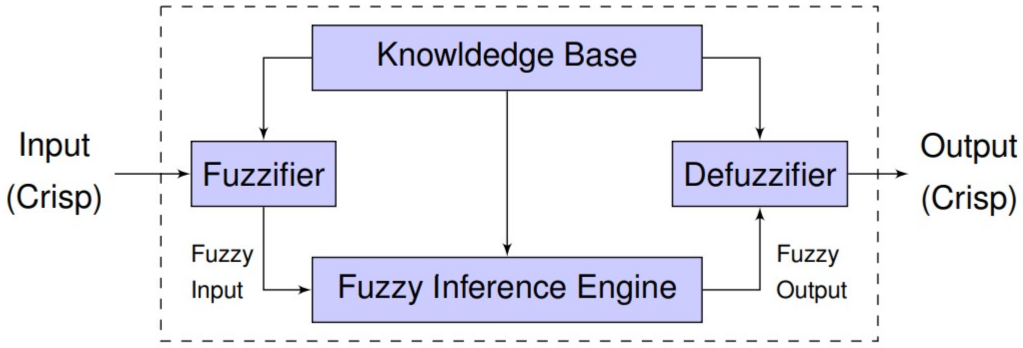


FIGURE 1.14 – A Diagram of Fuzzy Inference System [7].

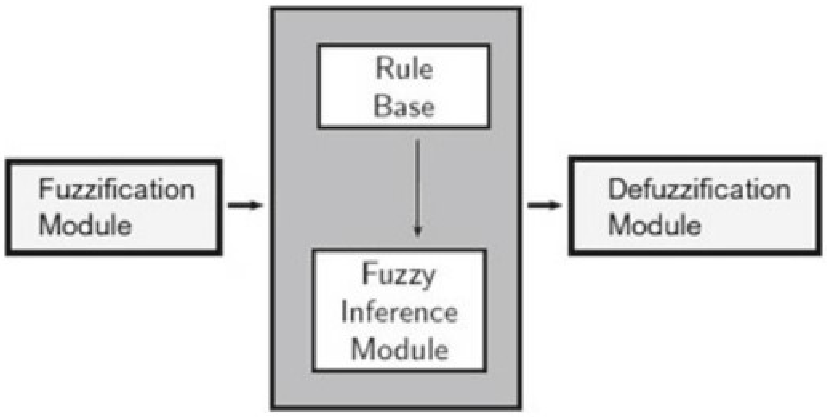


FIGURE 1.15 – The general FIS scheme [16].

pattern recognition. Because of its multi-disciplinary nature, the fuzzy inference system is known by numerous other names, such as fuzzy-rule-based system, fuzzy expert system, fuzzy model, fuzzy associative memory, fuzzy logic controller, and simply (and ambiguously) fuzzy system [19]. An FIS is a computing framework based on the concepts of fuzzy set theory, fuzzy (If-Then) rules and fuzzy reasoning [7]. An FIS consists of 4 components: fuzzifier, Knowledge base (rule base or database), fuzzy inference engine and defuzzifier [7].

Figure 1.14 is a block diagram of fuzzy interference system.

1.4.1.2 Definition 2

The terms "fuzzy inference system", "fuzzy rule-based system", "fuzzy controller" are used in the literature without distinction to mean a computing framework based on concepts of fuzzy set theory, fuzzy if-then rules and fuzzy reasoning [16].

A fuzzy inference system (FIS) can take either fuzzy or non-fuzzy inputs. In general, the outputs may be fuzzy sets. Then a method of defuzzification is needed in order to extract a numerical value that best represents a fuzzy set. Figure 1.15 shows a General FIS scheme [16].

1.4.2 Functional Blocks of FIS

The following four functional blocks will help you understand the construction of FIS:

— **Fuzzifiers:** It maps the crisp (real-valued) input into a fuzzy set defined in the universe of discourse (the domain of the fuzzy set) X characterized by membership functions. This process is called fuzzification. Note: The input can also be a fuzzy set.

— **Knowledge Base:** It is a database consisting of linguistic rules in If-Then format.

— Fuzzy Inference Engine: Using the If-Then rules in Knowledge base, it performs reasoning by producing a fuzzy output according to the fuzzy input given by the fuzzifier.

— **Defuzzifiers:** It converts the fuzzy output given by the fuzzy inference engine to produce a crisp (real-valued) output. This process is called defuzzification [7].

1.4.3 Structure of Fuzzy Inference Systems

1.4.3.1 Fuzzification module

The inputs of the system are modelled by means of fuzzy sets. The membership functions are usually chosen by consulting the experts in the domain. For each input numerical value, one obtains a vector whose elements are the membership degrees in the considered fuzzy set [16].

1.4.3.2 Rule-Base module

It consists of some fuzzy propositions, such as:

$$\text{Rule } i: \text{IF } \underbrace{x_1 \text{ is } A_{i1} \text{ and/or } x_2 \text{ is } A_{i2} \text{ and/or } \cdots}_{\text{antecedent}} \text{ THEN } \underbrace{y \text{ is } B_i}_{\text{consequent}}\ , i = 1, 2, \ldots, r$$

where:

— $x_1, x_2, \cdots$ are the fuzzy/linguistic variables.

— y is the output of the fuzzy inference system.

— "and" and "or" are fuzzy operators.

— $A_{i1}, A_{i2}, \cdots$ are the fuzzy sets (associated with a linguistic variable) representing the it h antecedent pairs.

— B_i is the fuzzy set (associated with a linguistic variable) representing the it h consequent.

— r is number of rules [7].

1.4.4 Fuzzy inference module

It translates fuzzy propositions into fuzzy relations. In this module, the kind of t-norms, t-conorms and fuzzy implications, to obtain the fuzzy relation that models the rule base, are fixed [16].

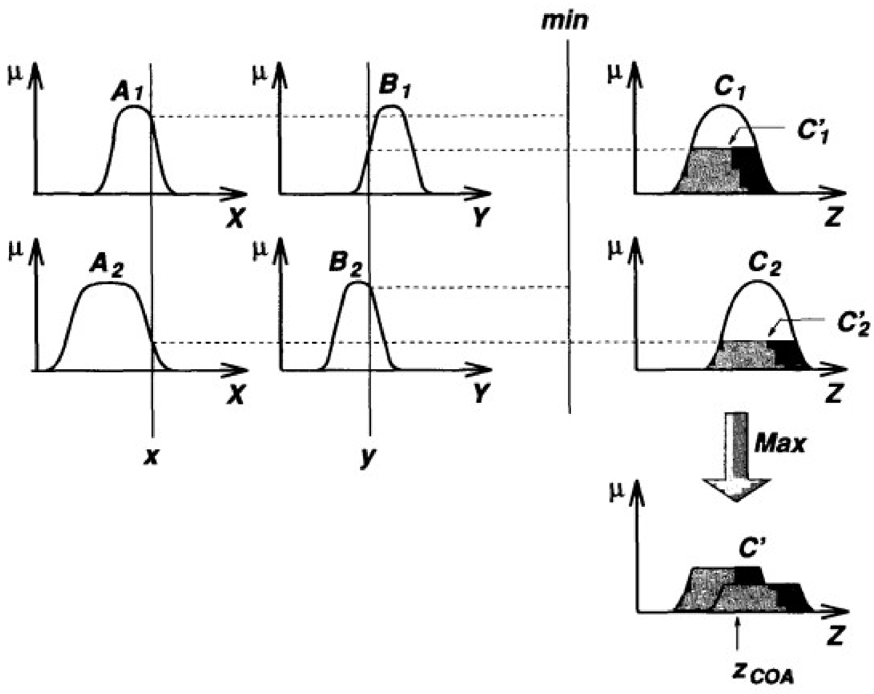


FIGURE 1.16 – The Mamdani fuzzy inference system [20].

1.4.4.1 Defuzzification module

It allows to represent a fuzzy set by a numerical value (real number) [16], it is also defined as a process to convert the fuzzy output (an inferred membership function) to a crisp value [7].

There are a number of methods available for defuzzification:

— max membership principle.

— centroid method.

— weighted average method.

— mean max membership.

— center of sums.

— center of largest area.

— first (or last) of maxima.

1.4.5 Types of Fuzzy Inference Systems

1.4.5.1 Mamdani Fuzzy Inference

a) Definition: The Mamdani fuzzy inference system was proposed as the first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators. Figure 1.16 is an illustration of how a two-rule Mamdani fuzzy inference system derives the overall output z when subjected to two crisp inputs x and y [19].

We have to use a defuzzifier to convert a fuzzy set to a crisp value [19].

The Mamdani-style fuzzy inference process is performed in four steps [5].

— Fuzzification of the input variables.

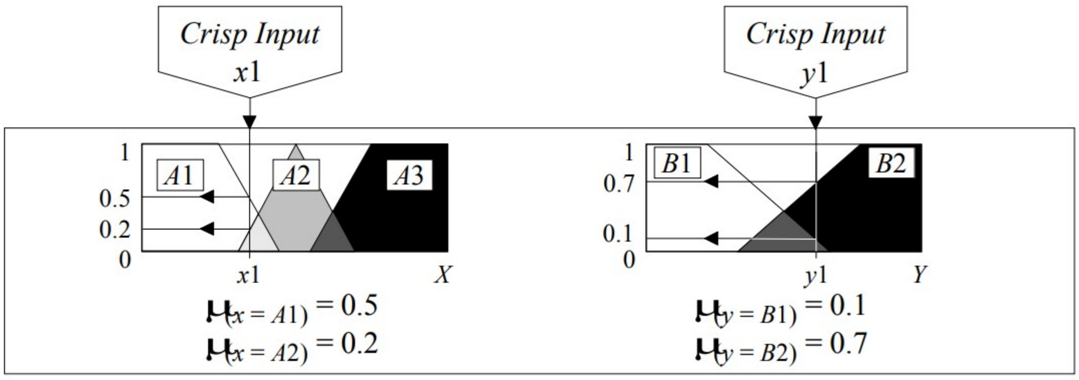


FIGURE 1.17 – The first step (Fuzzification) [5].

— Rule evaluation (inference).

— Aggregation of the rule outputs (composition).

— Defuzzification.

b) Steps

1. **Fuzzification**: The first step is to take the crisp inputs, x_1 and y_1 (project funding and project staffing), and determine the degree to which these inputs belong to each of the appropriate fuzzy sets [5].
2. **Rule Evaluation:**
   - The second step is to take the fuzzified inputs, μ(x=A1) = 0.5, μ(x=A2) = 0.2, μ(y=B1) = 0.1 and μ(y=B2) = 0.7, and apply them to the antecedents of the fuzzy rules.
   - If a given fuzzy rule has multiple antecedents, the fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation.
   - This number (the truth value) is then applied to the consequent membership function [5].

   Example:

   Figure 1.18 shows an example of rule evaluation.

   Now the result of the antecedent evaluation can be applied to the membership function of the consequent. There are two main methods for doing so:

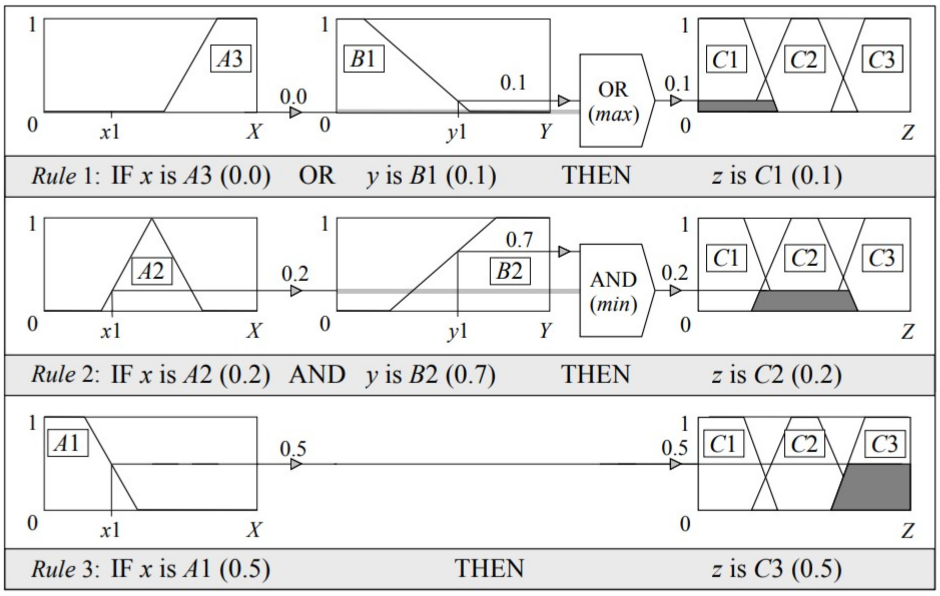
   - Clipping

   - Scaling

   The most common method of correlating the rule consequent with the truth value of the rule antecedent is to cut the consequent membership function at the level of the antecedent truth. This method is called clipping (alpha-cut) [5].

   Since the top of the membership function is sliced, the clipped fuzzy set loses some information.

   However, clipping is still often preferred because it involves less complex and faster mathematics, and generates an aggregated output surface that is easier to defuzzify [5].



A3
0.0
B1
0.1
OR (max)
0.1
C1
C2
C3
x1
X
y1
Y
Z
Rule 1: IF x is A3 (0.0) OR y is B1 (0.1) THEN z is C1 (0.1)
A2
0.2
0.7
B2
AND (min)
0.2
Rule 2: IF x is A2 (0.2) AND y is B2 (0.7) THEN z is C2 (0.2)
A1
0.5
0.5
Rule 3: IF x is A1 (0.5) THEN z is C3 (0.5)

FIGURE 1.18 – Example - Rule Evaluation [5].

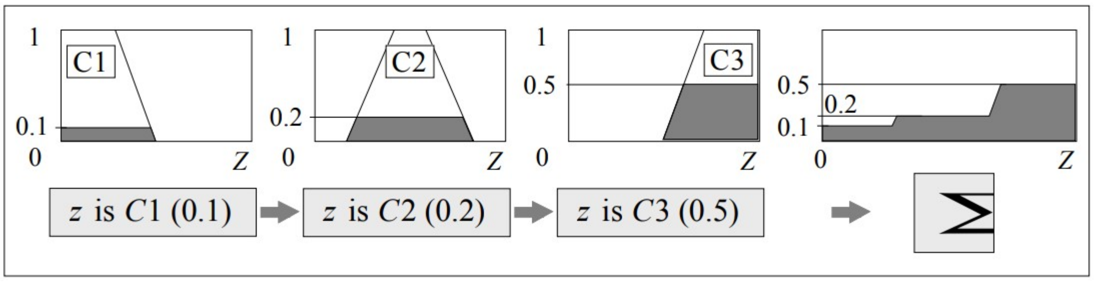
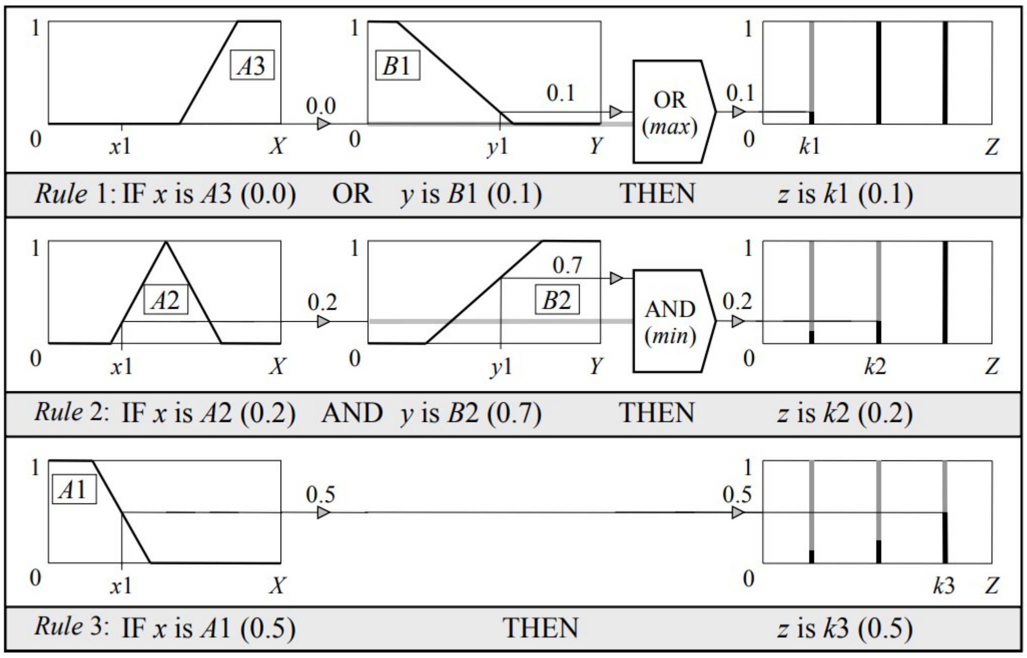


FIGURE 1.19 – Example - Aggregation of the Rule Outputs[5].

3. **Aggregation of the Rule outputs**:
   Aggregation is the process of unification of the outputs of all rules. We take the membership functions of all rule consequents previously clipped or scaled and combine them into a single fuzzy set.
   The input of the aggregation process is the list of clipped or scaled consequent membership functions, and the output is one fuzzy set for each output variable [5].
   Example: Figure 1.19 shows an example about aggregation of the rule outputs.
4. **Defuzzification**: The last step in the fuzzy inference process is defuzzification. Fuzziness helps us to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. The input for the defuzzification process is the aggregate output fuzzy set and the output is a single number [5].

1.4.5.2 Sugeno Fuzzy Inference

a) Definition: The Sugeno fuzzy model (also known as the TSK fuzzy model) was proposed by Takagi, Sugeno, and Kang in an effort to develop a systematic approach to generating fuzzy rules from a given input-output data set. A typical fuzzy rule in a Sugeno fuzzy model has the form:
if x is A and y is B then z = f (x, y).



Rule 1: IF x is A3 (0.0) OR y is B1 (0.1) THEN z is k1 (0.1)
OR (max)
Rule 2: IF x is A2 (0.2) AND y is B2 (0.7) THEN z is k2 (0.2)
AND (min)
Rule 3: IF x is A1 (0.5) THEN z is k3 (0.5)

FIGURE 1.20 – Example - Sugeno Rule Evaluation [5].

where A and B are fuzzy sets in the antecedent, while z = f (x, y) is a crisp function in the consequent. Usually f (x, y) is a polynomial in the inputs x and y [5].

Michio Sugeno suggested to use a single spike, a singleton, as the membership function of the rule consequent. A singleton, or more precisely a fuzzy singleton, is a fuzzy set with a membership function that is unity at a single particular point on the universe of discourse and zero everywhere else.

Sugeno-style fuzzy inference is very similar to the Mamdani method. Sugeno changed only a rule consequent. Instead of a fuzzy set, he used a mathematical function of the input variable [5]. The format of the Sugeno-style Fuzzy Rule is:

IF x is A AND y is B THEN z is $f(x, y)$

Where x, y and z are linguistic variables; A and B are fuzzy sets on universe of discourses X and Y, respectively; and f (x, y) is a mathematical function.

The most commonly used zero-order Sugeno fuzzy model applies fuzzy rules in the following form:

IF x is A AND y is B THEN z is k

where k is a constant.

In this case, the output of each fuzzy rule is constant. All consequent membership functions are represented by singleton spikes [5].

b) Sugeno Rule Evaluation :

Figure 1.20 shows an example about Sugeno Rule Evaluation.

c) Sugeno Aggregation of the Rule Outputs :

Figure 1.21 shows an example about Sugeno Aggregation of Rule Outputs

d) Sugeno Defuzzification :

Weighted Average (WA):

The Weighted Average (WA) can be computed by Equation 1.12.

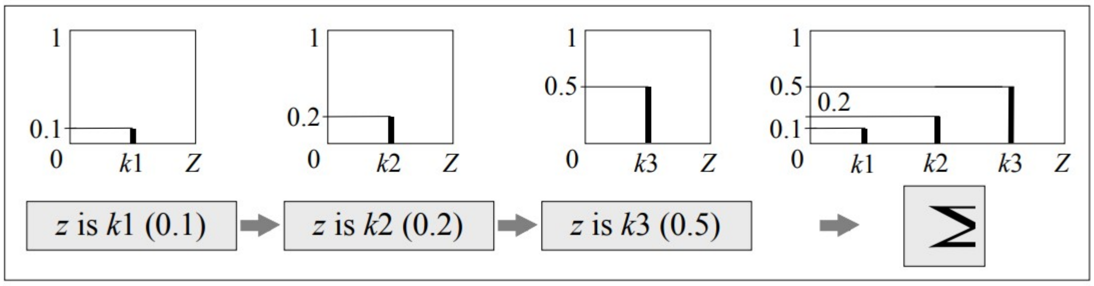


FIGURE 1.21 – Example - Aggregation of the Rule Outputs [5].

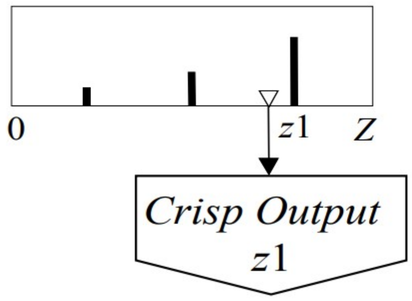


FIGURE 1.22 – Sugeno Defuzzification [5].

$$WA = \frac{\mu(k1)\times k1 + \mu(k2)\times k2 + \mu(k3)\times k3}{\mu(kl) + \mu(k2) + \mu(k3)} = \frac{0.1\times 20 + 0.2\times 50 + 0.5\times 80}{0.1 + 0.2 + 0.5} = 65 \tag{1.12}$$

Figure 1.22 shows an example about Sugeno Defuzzification.

1.4.5.3 Tsukamoto fuzzy inference system

a) Definition: In the Tsukamoto fuzzy models (Figure 1.23) , the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonical membership function. As a result, the inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's output.

Since each rule infers a crisp output, the Tsukamoto fuzzy model aggregate each rule's output by the method of weighted average and thus avoids the time-consuming process of defuzzification. However, the Tsukamoto fuzzy model is not used often since it is not as transparent as either the Mamdani or Sugeno fuzzy models[12].

Example single-input Tsukamoto fuzzy model:

An example of single-input single-output Tsukamoto fuzzy model with 3 rules [7].

Rule 1: IF X is Small THEN Y is C_1

Rule 2: IF X is Medium THEN Y is C_2

Rule 3: IF X is Large THEN Y is C_3

Inferred Output:

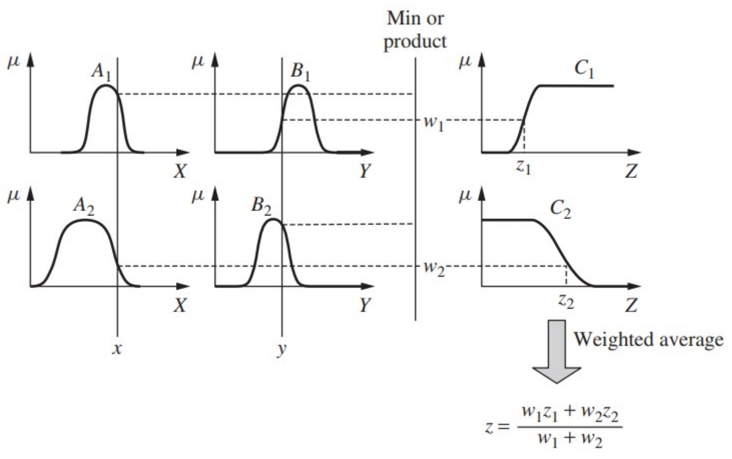


FIGURE 1.23 – The Tsukamoto Fuzzy model [7].

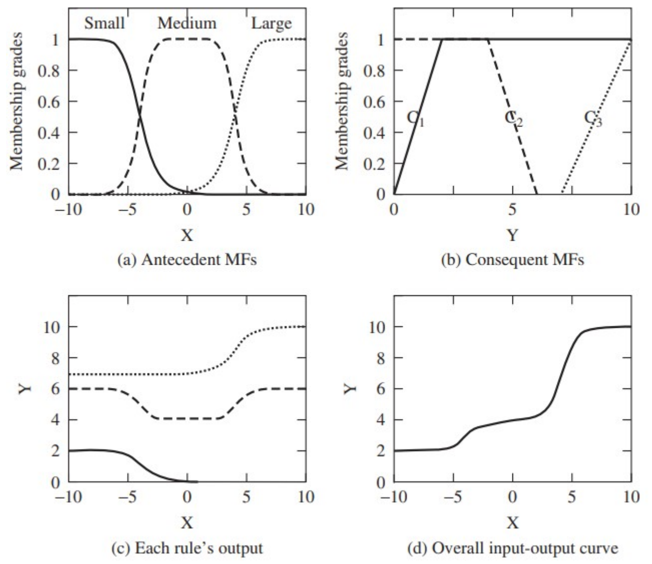


FIGURE 1.24 – Single-Input, Single-Output Tsukamoto Fuzzy Model with 3 Rules [7].

$$Y = \frac{\mu_{\text{Smal}}(X) C_1(X) + \mu_{\text{Modium}}(X) C_2(X) + \mu_{\text{Large}}(X) C_3(X)}{\mu_{\text{Smal}}(X) + \mu_{\text{Medium}}(X) + \mu_{\text{Large}}(X)} \tag{1.13}$$

1.4.5.4 Mamdani or Sugeno

— Mamdani method is widely accepted for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human-like manner. However, Mamdani-type fuzzy inference entails a substantial computational burden.

— On the other hand, Sugeno method is computationally effective and works well with optimization and adaptive techniques, which makes it very attractive in control problems, particularly for dynamic nonlinear systems [5].

1.5 Fuzzy Control Systems

1.5.1 Basics of Fuzzy Control

1.5.1.1 Definition

Fuzzy control is a type of control system that uses fuzzy logic to handle uncertainties and imprecise information in a decision-making process. The basic components of a fuzzy control system include fuzzification, rule evaluation, and defuzzification.

Fuzzy control is a form of intelligent control characterized by the use of expert knowledge on the control strategy and/or the behavior of the controlled plant. This expert knowledge is represented by means of IF-THEN rules and linguistic variables. Attributes or values of these linguistic variables are linguistic terms associated with fuzzy sets, a generalization of ordinary ("crisp") sets. Fuzzy set theory is the theoretical basis underlying information processing in fuzzy control systems. From the systems theory's view, a fuzzy controller is a static nonlinear transfer element incorporated into a control loop. This gives rise to methods for analysis and systematic design. Fuzzy systems may perform different tasks within an automatic control system leading to different structural schemes. Nowadays, fuzzy control systems are successfully applied in many technical and non-technical fields. The application of fuzzy control systems is supported by numerous hardware and software solutions [18].

1.5.1.2 Components of Fuzzy Control

1. **Fuzzification:** This is the process of converting a crisp input value to a fuzzy value. The input variables in a fuzzy control system are generally mapped by sets of membership functions known as "fuzzy sets". The fuzzifier transforms the physical values as well as the error signals to a normalized fuzzy subset [11].
2. **Rule Evaluation:**
   - Fuzzy Rules: Fuzzy control systems operate based on a set of fuzzy rules. These rules are typically expressed in an "if-then" format and describe the relationship between fuzzy input values and fuzzy output values. For example, "if temperature is cold and humidity is high, then increase the heating" [11].
   - Rule Aggregation: The fuzzy rules are combined to determine the overall output of the system. This process involves aggregating the individual rule outputs to create a comprehensive fuzzy output.
3. **Defuzzification**: The defuzzifier converts the fuzzy quantities into crisp quantities from an inferred fuzzy control action. This is necessary because, for control purposes, a crisp control signal is required [11].

1.5.1.3 Fuzzy Control - A Simple Example

In the following section, a simple and illustrative example will be used to explain information processing in fuzzy systems:

Example: Control of room temperature: The temperature of a room equipped with a hot water heating should be controlled by adjusting the position of the valve at the radiator (see Figure 1.25). A human being would use meta-rules, such as If things are not OK but change in the right direction then maintain present settings or more specifically If the temperature is too warm but decreases, then leave valve

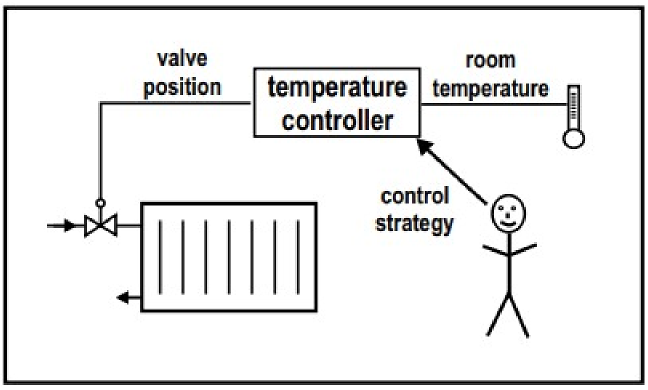


FIGURE 1.25 – Schematic Representation of the Control for Example 1 [18].

position unchanged or if the temperature is too cold and decreases, then increase the valve opening significantly.

Starting from these meta-rules, an experienced user would develop a set of control rules which are more specific regarding the linguistic description of the values of temperature, temperature change, and change of valve position [18].

1.5.2 Applications and Advantages of Fuzzy Control

Fuzzy control has become an essential tool in the field of automation, providing a more efficient and precise control system compared to traditional control methods. The application of fuzzy control in automation has revolutionized how industries operate, especially in the manufacturing and production sectors. The use of fuzzy logic algorithms in control systems has enabled automation engineers to create systems that are more adaptive to varying conditions, leading to a reduction in production costs and an increase in productivity. From a different perspective, the use of fuzzy control in automation has led to a reduction in human error, leading to increased safety in the workplace [10].

1.5.2.1 Applications of Fuzzy Control

In this section, we will explore the different applications of fuzzy control in automation and how they have transformed various industries (see Figure 1.26).

Here are some of the major applications of fuzzy control in automation:

— **a. Temperature control**: Fuzzy control has been applied in various temperature control systems, such as heating and cooling systems in buildings and industrial processes. Fuzzy logic algorithms have enabled these systems to adjust temperature settings according to environmental conditions, leading to significant energy savings [10].

— **b. Robotics**: Fuzzy control has been applied in robotics to create more adaptive and intelligent robots. The use of fuzzy logic algorithms has enabled robots to make decisions based on varying conditions, leading to more efficient and precise operations. For example, fuzzy control can be applied in the control system of a robotic arm to enable it to grasp objects of varying shapes and sizes [10].

— **c. Automotive industry**: Fuzzy control has been applied in the automotive industry to enhance the performance of vehicles, especially in the engine and

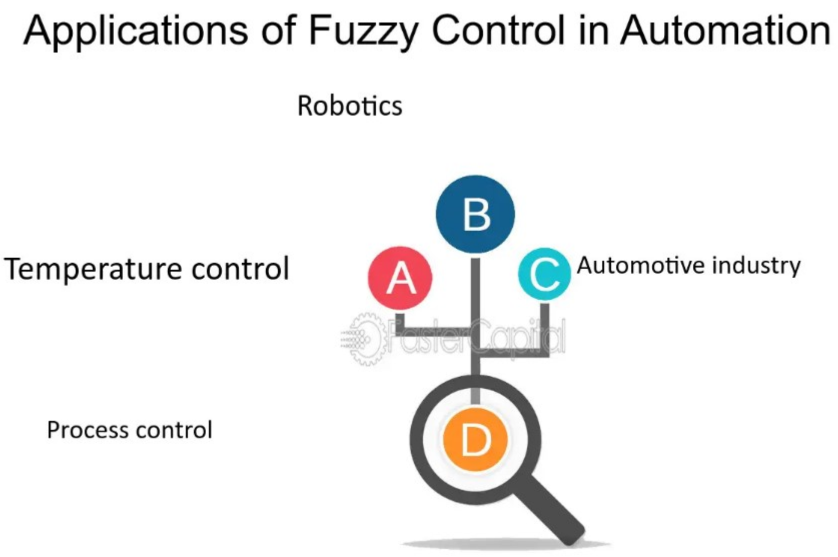


FIGURE 1.26 – Applications of Fuzzy Control [10].

transmission control systems. Fuzzy logic algorithms have enabled these systems to adjust to varying driving conditions, leading to improved fuel efficiency and reduced emissions [10].

— **d. Process control**: Fuzzy control has been applied in various industrial processes, such as chemical and pharmaceutical manufacturing, to improve the accuracy and efficiency of control systems. Fuzzy logic algorithms have enabled these systems to adjust to varying process conditions, leading to improved product quality and reduced production costs [10].

1.5.2.2 Advantages of Fuzzy Control

Fuzzy Control is an innovative technology that has revolutionized precision in automation. It is a control system that uses fuzzy logic to make decisions based on degrees of truth rather than binary (true or false) decisions. This technology has brought about several advantages in various industries, including manufacturing, robotics, and transportation. In this section, we will discuss the advantages of fuzzy control from different perspectives [10].

— Increased Efficiency: One significant advantage of fuzzy control is the increase in efficiency it provides. The system can adjust to changing conditions and make decisions based on these changes. For example, in the manufacturing industry, fuzzy control can adjust the production process to minimize waste, reduce energy consumption, and optimize product quality. In the transportation industry, it can optimize traffic flow, reduce fuel consumption, and improve safety [10].

— Reduced Costs: Fuzzy control can also help reduce costs in many industries. For example, in the manufacturing industry, it can reduce the need for human intervention, resulting in cost savings in labor. In the transportation industry, it can optimize the use of resources such as fuel, resulting in cost savings in fuel consumption [10].

— Improved Safety: Fuzzy control can improve safety in many industries. In the robotics industry, fuzzy control can make robots safer by ensuring they operate

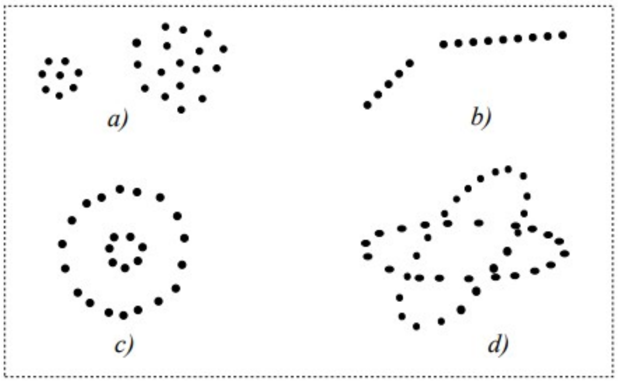


Figure 1.27 – Clusters of Different Shapes and Dimensions in $\mathbb{R}^2$ [12].

within safe parameters. In the transportation industry, it can optimize traffic flow, reducing the likelihood of accidents [10].

— Increased Flexibility: Fuzzy control can also increase flexibility in many industries. For example, in the manufacturing industry, it can adjust to changes in production requirements and adapt the production process accordingly. In the transportation industry, it can adapt to changes in traffic flow and adjust the route accordingly [10].

1.6 Advanced Topics in Fuzzy Logic

1.6.1 Fuzzy Clustering

1.6.1.1 Definition

Fuzzy Clustering is a type of clustering algorithm in machine learning that allows a data point to belong to more than one cluster with different degrees of membership. Unlike traditional clustering algorithms, such as k-means or hierarchical clustering, which assign each data point to a single cluster, fuzzy clustering assigns a membership degree between 0 and 1 for each data point for each cluster [13].

1.6.1.2 Overview of Clustering Methods

Many clustering algorithms have been introduced in the literature. Since clusters can formally be seen as subsets of the data set, one possible classification of clustering methods can be according to whether the subsets are fuzzy or crisp (hard) [12]. Figure 1.27 shows clusters of different shapes and dimensions in $\mathbb{R}^2$.

Hard clustering methods are based on classical set theory, and require that an object either does or does not belong to a cluster. Hard clustering means partitioning the data into a specified number of mutually exclusive subsets [12].

Fuzzy clustering methods, however, allow the objects to belong to several clusters simultaneously, with different degrees of membership. In many situations, fuzzy clustering is more natural than hard clustering. Objects on the boundaries between several classes are not forced to fully belong to one of the classes, but rather are assigned membership degrees between 0 and 1 indicating their partial membership. The discrete nature of the hard partitioning also causes difficulties with algorithms based on analytic functionals, since these functionals are not differentiable [12].

1.6.1.3 Applications in several fields of Fuzzy clustering

Fuzzy clustering is used in a wide range of fields due to its ability to handle complexity and uncertainty in data [13]. Here are some applications in various domains:

1. **Image segmentation:** Fuzzy clustering can be used to segment images by grouping pixels with similar properties together, such as color or texture.
2. **Pattern recognition:** Fuzzy clustering can be used to identify patterns in large datasets by grouping similar data points together.
3. **Marketing:** Fuzzy clustering can be used to segment customers based on their preferences and purchasing behavior, allowing for more targeted marketing campaigns.
4. **Medical diagnosis:** Fuzzy clustering can be used to diagnose diseases by grouping patients with similar symptoms together.
5. E**Environmental monitoring:** Fuzzy clustering can be used to identify areas of environmental concern by grouping together areas with similar pollution levels or other environmental indicators.
6. **Traffic flow analysis:** Fuzzy clustering can be used to analyze traffic flow patterns by grouping similar traffic patterns together, allowing for better traffic management and planning.
7. **Risk assessment:** Fuzzy clustering can be used to identify and quantify risks in various fields, such as finance, insurance, and engineering [13].

1.6.2 Fuzzy Reasoning

1.6.2.1 Definition

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts. Before introducing fuzzy reasoning, we shall discuss the compositional rule of inference, which plays a key role in fuzzy reasoning [14].

Another definition, fuzzy reasoning we mean the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise premises. Such reasoning is, for the most part, qualitative rather than quantitative in nature and almost all of it falls outside of the domain of applicability of classical logic (Zadeh, 1977a).

1.6.2.2 Compositionel Rule of inference

The compositional rule of inference is a generalization of the following notion. Suppose a curve y = f(x) that regulates the relation between x and y. When we are given x = a, then from y = f(x) we can infer that y = b = f(a); A generalization of the aforementioned process would allow a to be an interval and f(x) to be an interval-valued function. To find the resulting interval y = b corresponding to the interval x = a, we first construct a cylindrical extension of a and then find its intersection I with the interval-valued curve. The projection of I onto the y-axis yields the interval y = b.

Going one step further in our generalization, we assume that F is a fuzzy relation on X * Y and A is a fuzzy set of X. To find the resulting fuzzy set B, again we construct a cylindrical extension c(A) with base A. The intersection of c(A) and F forms the

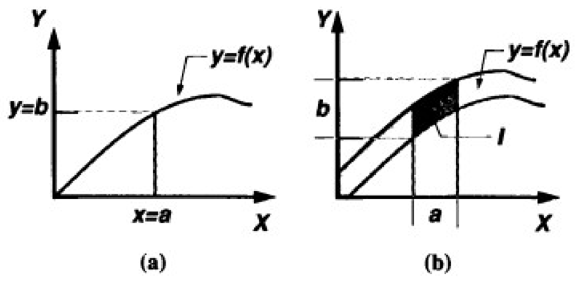


FIGURE 1.28 – Example of Compositional Rule of inference [14].

analog of the region of intersection I. By projecting c(A) ∩ F onto the y-axis, we infer y as a fuzzy set B on the y-axis [14].

Specifically, let $\mu_{(A)}$, μ_{cA} , μ_B, and μ_F be the MFs of A, c(A), B, and F, respectively, where $\mu_{c(A)}$) is related to μ_A as given by Equation 1.14 and illustrated by Figure 1.28.

$$\mu_{c(A)}(x, y) = \mu_A(x) \tag{1.14}$$

Equation 1.14 can be rewritten as shown by Equation 1.15.

$$\begin{aligned} \mu_{c(A)\cap F}(x, y) &= \min\left[\mu_{c(A)}(x, y), \mu_F(x, y)\right] \\ &= \min\left[\mu_A(x), \mu_F(x, y)\right]. \end{aligned} \tag{1.15}$$

By projecting $c(A) \cap F$ onto the y-axis, we get the formula shown in Equation 1.16.

$$\begin{aligned} \mu_B(y) &= \max_x \min\left[\mu_A(x), \mu_F(x, y)\right] \\ &= V_x\left[\mu_A(x) \wedge \mu_F(x, y)\right] \end{aligned} \tag{1.16}$$

This formula reduces to the max-min composition of two relation matrices if both A (a unary fuzzy relation) and F (a binary fuzzy relation) have finite universes of discourse. Conventionally, B is represented as given by Equation 1.17.

$$B = A \circ F \tag{1.17}$$

where ∘ denotes the composition operator.

It is interesting to note that the extension principle is in fact a special case of the compositional rule of inference. Specifically, if y = f(x) in Figure 1.28 is common crisp one-to-one or many-to-one function, then the derivation of the induced fuzzy set B on Y is exactly what is accomplished by the extension principle [14].

Using the compositional rule of inference, we can formalize an inference procedure upon a set of fuzzy if-then rules. This inference procedure, generally called approximate reasoning or fuzzy reasoning, is the topic of the next subsection [14].

1.6.2.3 Fuzzy Reasoning

The basic rule of inference in traditional two-value topic is modus ponens, according to which we can infer the truth of a proposition B from the truth of A and the implication $A \rightarrow B$. For instance, if A is identified with "the tomato is red" and B with "the tomato is ripe", then if it is true that "the tomato is red", it is also true that "the tomato is ripe" [14]. This concept is illustrated as follows:

premise 1 (fact): x is A

premise 2 (rule): if x is A then y is B

consequence (conclusion): y is B

However, in much of human reasoning, modus ponens is employed in an approximate manner. For example, if we have the same implication rule "if the tomato is red, then it is ripe" and we know that "the tomato is more or less red", then we may infer that "the tomato is more or less ripe" [14]. This is written as:

premise 1 (fact): x is A'

premise 2 (rule): if x is A then y is B

consequence (conclusion): y is B′

where A′ is close to A and B′ is close to B. When A, B, A′, and B′ are fuzzy sets of approximate universes, the foregoing inference procedure is called approximate reasoning or fuzzy reasoning; it is also called generalized modus ponens (GMP for short), since it has modus ponens as special case. Using the composition rule of inference introduced in the previous subsection, we can formulate the inference procedure of fuzzy reasoning as the following definition [14].

1.7 Conclusion

This chapter introduced Fuzzy Logic, renowned for its adaptability in handling uncertainty across various domains like control systems and artificial intelligence. Its capability to manage vague information has led to widespread adoption in real-world applications. Reflecting on its journey reveals fuzzy logic as a powerful tool for researchers and decision-makers, evolving alongside technological advancements. Looking ahead to text classification in the next chapter, leveraging fuzzy logic principles can enhance classification by accommodating the ambiguity of textual data, thus offering more robust models for addressing contemporary challenges.

Chapter 2

Text Classification

2.1 Introduction

In an era dominated by information overload, the ability to efficiently categorize and understand textual data has become increasingly crucial. Text classification, a branch of natural language processing (NLP), emerges as a powerful tool to automatically assign predefined categories or labels to textual content. This classification process enables effective organization, retrieval, and analysis of vast amounts of text, ranging from social media posts and news articles to academic papers.

The importance of text classification transcends numerous domains, impacting both individual users and businesses. It plays a pivotal role in sentiment analysis, spam detection, and document categorization. Businesses leverage text classification to automate customer support, extract valuable insights from unstructured data, and enhance decision-making processes. In the realm of social media, it aids in understanding user sentiments, detecting trends, and improving content recommendation systems.

This chapter aims to delve into the world of text classification, addressing both its theoretical foundations and practical applications.

2.2 Fundamentals of Text Classification

2.2.1 Definition and Scope

2.2.1.1 Definition

Text Classification: also known as text categorization, is a classical problem in natural language processing (NLP), which aims to assign labels or tags to textual units such as sentences, queries, paragraphs, and documents. It has a wide range of applications including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation, and so on. Text data can come from different sources, including web data, emails, chats, social media, tickets, insurance claims, user reviews, and questions and answers from customer services, to name a few. Text is an extremely rich source of information. But extracting insights from text can be challenging and time-consuming, due to its unstructured nature [20].

Text classification involves analyzing raw text data to identify its features and predict its corresponding categories. Over the past few decades, various models have been proposed for this task. Initially, traditional models like Naive Bayes (NB) were utilized, followed by more generalized classifiers such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Random Forest (RF), which are commonly applied in text classification. Recently, methods like eXtreme Gradient

Boosting (XGBoost) and Light Gradient Boosting Machine (LightGBM) have shown promise for achieving high performance. In the realm of deep learning, TextCNN has garnered significant attention, employing Convolutional Neural Networks (CNNs) for text classification. Additionally, although not originally intended for text classification, Bidirectional Encoder Representation from Transformers (BERT) has become widely adopted due to its effectiveness across various text classification datasets [34].

Text classification can be performed either through manual annotation or by automatic labeling. With the growing scale of text data in industrial applications, automatic text classification is becoming increasingly important [20]. Approaches to automatic text classification can be grouped into two categories:

— Rule-based methods

— Machine learning (data-driven)–based methods.

2.2.1.2 Text Classification Tasks

Text Classification (TC) is the process of categorizing texts (e.g., tweets, news articles, customer reviews) into organized groups. Typical TC tasks include sentiment analysis, news categorization and topic classification. Recently, researchers show that it is effective to cast many natural languages understanding (NLU) tasks (e.g., extractive question answering, natural language inference) as TC by allowing deep learning (DL)-based text classifiers to take a pair of texts as input. This section introduces five TC tasks discussed in this article, including three typical TC tasks and two NLU tasks that are commonly cast as TC in many recent DL studies [20].

— **a. Sentiment Analysis**: This is the task of analyzing people's opinions in textual data (e.g., product reviews, movie reviews, or tweets), and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities [20].

— **b. News Categorization** : News contents are among the most important information sources. A news classification system helps users obtain information of interest in real-time by, e.g., identifying emerging news topics or recommending relevant news based on user interests [20].

— **c. Topic Analysis**: The task, also known as topic classification, aims to identify the theme or topics of a text (e.g., whether a product review is about "customer support" or "ease of use") [20].

— **d. Question Answering (QA)** : There are two types of QA tasks: extractive and generative. Extractive QA is a TC task: Given a question and a set of candidate answers, a system classifies each candidate answer as correct or not. Generative QA is a text generation task, since it requires generating answers on the fly. This article only discusses extractive QA [20].

— **e. Natural language inference (NLI)** : NLI, also known as recognizing textual entailment (RTE), predicts whether the meaning of one text can be inferred from another. An NLI system needs to assign to a pair of text units a label such as entailment, contradiction, and neutral [20]. Paraphrasing is a generalized form of NLI, also known as text pair comparison, the task of measuring the semantic similarity of a sentence pair indicating how likely one sentence is a paraphrase of the other [20].

2.2.2 Key Challenges

2.2.2.1 Handling Large Datasets

Handling large datasets in text classification is a crucial aspect of building robust and scalable models. Dealing with vast amounts of textual information comes with its own set of challenges, but effective strategies can optimize processing efficiency and model performance. Strategies for Handling Large Datasets in Text Classification

1. **Data Sampling and Subset Creation**: Given the size of large datasets, consider creating representative subsets for initial model development and testing. This helps in faster prototyping and debugging [29].
2. **Batch Processing**: Implement batch processing techniques to handle the data in smaller chunks, facilitating parallel processing and reducing memory requirements during model training [29].
3. **Feature Engineering**: Focus on extracting relevant features from the text data to reduce dimensionality and computational load. Techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings can be beneficial [29].
4. **Distributed Computing**: Leverage distributed computing frameworks like Apache Spark to distribute the workload across multiple machines, enabling efficient processing of large datasets [29].
5. **Data Preprocessing and Cleaning**: Prioritize data cleaning and preprocessing steps to remove noise and irrelevant information. This not only enhances model accuracy but also speeds up the training process [29].
6. **Incremental Learning**: Consider implementing incremental learning techniques where the model is updated gradually with new data, avoiding the need to retrain the entire model from scratch [29].

2.2.2.2 Dealing with Multiclass and Imbalanced Classification

Dealing with multiclass and imbalanced classification can be challenging, but there are several approaches and resources available to help you navigate these issues [31]. Here are some strategies and resources:

— a. Imbalanced Classification Strategies

1. Resampling Techniques
    - — Over-sampling: Augment the instances in the minority class.
    - — Under-sampling: Reduce instances in the majority class.
    - — SMOTE (Synthetic Minority Over-sampling Technique): Generate synthetic samples for the minority class [31].
2. Cost-sensitive Learning
    - — Adjust misclassification costs to give higher penalties for misclassifying the minority class [31].
3. Ensemble Methods
    - — Employ ensemble techniques such as Random Forests or Gradient Boosting, known for their effectiveness with imbalanced datasets.
4. Algorithm-Specific Approaches

    — Leverage built-in parameters or techniques tailored for handling imbalanced data, such as class weights in scikit-learn.

— b. Multiclass Classification Approaches

1. One-vs-Rest (OvR) and One-vs-One (OvO)
   — Break down the multiclass problem into several binary classification tasks.
2. Multi-class Loss Functions
   — Utilize loss functions specifically crafted for multiclass scenarios, like cross-entropy loss.
3. Decision Trees and Random Forests
   — Take advantage of the inherent support for multiclass classification in decision trees and random forests.
4. Neural Networks
   — Apply neural networks with SoftMax activation in the output layer to facilitate multiclass classification.

Approaching these challenges with a combination of these techniques can enhance your ability to tackle both imbalanced and multiclass classification scenarios effectively [31].

2.2.2.3 Consideration of Context and Ambiguity

In text classification, the consideration of context and ambiguity is pivotal for building accurate and nuanced models. Textual data often contains subtle nuances, context-dependent meanings, and potential ambiguities that challenge traditional classification approaches. Here are some key considerations and resources for addressing context and ambiguity in text classification [30].

1. Context-Aware Feature Engineering
   — Develop features that capture the context around words or phrases. Consider n-grams, embeddings, or contextual embeddings (e.g., BERT) to preserve context in the input data.
2. Understanding Polysemy
   — Address words with multiple meanings (polysemy) by leveraging context information. Contextual embeddings and word sense disambiguation techniques can be valuable in such cases.
3. Ambiguity Handling Strategies
   — Implement techniques for dealing with ambiguity, such as probabilistic models, ensemble methods, or incorporating uncertainty estimates in predictions.
4. Contextual Preprocessing
   — Preprocess the text data by considering the context in which words appear. This may involve lemmatization, stemming, or removing stop words based on the specific context of the classification task.
5. Domain-specific Context Consideration
   — Understand the domain-specific context in which the text data exists. Customizing models for specific domains helps capture context more accurately.
6. Handling Negation and Modifiers
   — Recognize the impact of negations and modifiers on the overall meaning of sentences. Adjust classification models to account for changes in sentiment or intent due to such linguistic constructs.

2.3 Text Representation Techniques

2.3.1 Bag-of-Words Model

The Bag of Words model is perhaps one of the simplest yet most powerful techniques to extract features from text documents. The essence of this model is to convert text documents into vectors such that each document is converted into a vector that represents the frequency of all the distinct words that are present in the document vector space for that specific document. Thus, considering our sample vector from the previous mathematical notation for D, the weight for each word is equal to its frequency of occurrence in that document [21].

The bag-of-words model stands out as a widely adopted method for object categorization. Its core principle involves assigning each identified key point to a visual word and subsequently representing images through histograms of these visual words. Typically, clustering algorithms like K-means are employed to generate these visual words. While studies have showcased promising outcomes using the bag-of-words approach, theoretical examinations of its properties have been scarce, likely due to the complexities introduced by heuristic clustering processes. In this study, we introduce a statistical framework that extends the bag-of-words representation. Within this framework, visual words are derived through a statistical process rather than traditional clustering algorithms, yet the empirical results remain competitive. We provide a theoretical analysis based on statistical consistency for this proposed framework. Furthermore, leveraging this framework, we devise two algorithms that bypass the need for clustering while achieving comparable performance in object categorization to traditional bag-of-words representations based on clustering [29].

2.3.2 TF-IDF (Term Frequency-Inverse Document Frequency)

The Bag of Words model is good, but the vectors are completely based on absolute frequencies of word occurrences. This has some potential problems where words that may tend to occur a lot across all documents in the corpus will have higher frequencies and will tend to overshadow other words that may not occur as frequently but may be more interesting and effective as features to identify specific categories for the documents. This is where TF-IDF comes into the picture. TF-IDF stands for Term Frequency-Inverse Document Frequency, a combination of two metrics: term frequency and inverse document frequency. This technique was originally developed as a metric for ranking functions for showing search engine results based on user queries and has come to be a part of information retrieval and text feature extraction now [21].

Let us formally define TF-IDF now and look at the mathematical representations for it before diving into its implementation. Mathematically, TF-IDF is the product of two metrics and can be represented as $tfidf = tf \times idf$, where term frequency (tf) and inverse-document frequency (idf) represent the two metrics.

— **a. TF** Term frequency denoted by tf is what we had computed in the Bag of Words model. Term frequency in any document vector is denoted by the raw frequency value of that term in a particular document. Mathematically it can be represented as tf (w, D) = f wD, where f wD denotes frequency for word w in document D, which becomes the term frequency (tf). There are various other representations and computations for term frequency, such as converting frequency to a binary feature where 1 means the term has occurred in the document and 0 means it has not. Sometimes you can also normalize the absolute raw frequency using logarithms or averaging the frequency. We will be using the raw frequency in our computations [21].

— **b. IDF** Inverse document frequency denoted by idf is the inverse of the document frequency for each term. It is computed by dividing the total number of documents in our corpus by the document frequency for each term and then applying logarithmic scaling on the result. In our implementation we will be adding 1 to the document frequency for each term just to indicate that we also have one more document in our corpus that essentially has every term in the vocabulary. This is to prevent potential division-by-zero errors and smoothen the inverse document frequencies [21]. We also add 1 to the result of our idf computation to avoid ignoring terms completely that might have zero idf. Mathematically our implementation for idf can be represented by Equation 2.1.

$$idf(t) = 1 + log[C/(1 + df(t))] \tag{2.1}$$

where idf(t) represents the idf for the term t, C represents the count of the total number of documents in our corpus, and df(t) represents the frequency of the number of documents in which the term t is present.

Thus, the term frequency-inverse document frequency can be computed by multiplying the above two measures together. The final TF-IDF metric we will be using is a normalized version of the tfidf matrix we get from the product of tf and idf. We will normalize the tfidf matrix by dividing it with the L2 norm of the matrix, also known as the Euclidean norm, which is the square root of the sum of the square of each term's tfidf weight. Mathematically we can represent the final tfidf feature Vector as shown in Equation 2.2.

$$tfidf = tfid/||tfidf|| \tag{2.2}$$

where $||tfidf||$ represents the Euclidean L2 norm for the tfidf matrix [21].

2.3.3 Word Embeddings

2.3.3.1 A Review on Word Embedding Techniques for Text Classification

Word embedding serves as a technique to transform words from a vocabulary into vectors of real numbers, establishing correlations between words based on their contextual similarity. Unlike one-hot encoding, which represents words as binary values in a high-dimensional space, word embedding employs a dense representation in a low-dimensional vector space. For instance, words like "Boy," "grapes," and "man" are distributed in an n-dimensional vector space, where the proximity between words reflects their semantic similarity. This approach captures hidden relationships among words, providing a nuanced understanding beyond mere presence or absence. Word embeddings find success in unsupervised learning applications, eliminating the need for costly annotations. They play a crucial role in various Natural Language Processing tasks such as text classification, document clustering, and sentiment analysis. As show in Figure 2.1, the three main categories of word embedding techniques are Traditional or Frequency-based, Static, and Contextualized embeddings [23].

Traditional or Frequency-based Embeddings: Count vector, TF-IDF, and co-occurrence fall under this category.

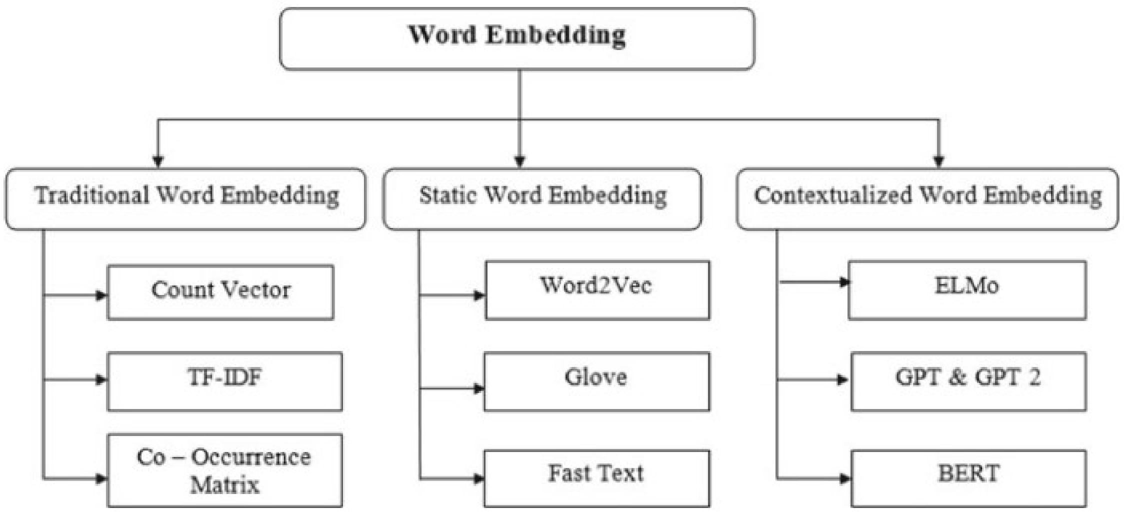


FIGURE 2.1 – Types of Word Embedding Techniques [23].

Static Word Embeddings: Word2Vec, Glove, and Fast Text are examples of static word embeddings.

Contextualized Word Embeddings: Elmo, GPT-2, and BERT are part of the contextualized word embedding category.

The following subsection delves into the exploration of word embedding techniques for text classification, shedding light on their applications and effectiveness in this domain.

2.3.3.2 Types of Word Embedding techniques

A. Traditional Word Embedding: Traditional word embedding is based on the frequency that considers the whole document and discovers the significance of rare words in the document, count occurrence of each word, and co-occurrence of words [23].

TABLE 2.1 – Traditional Word Embedding [23].

Traditional	**Description**
Count vector	Count vector is a method of counting the number of times each word occurs in the document.
TF-IDF (term frequency-inverse document frequency).	TF-IDF is calculated by the product of the frequency of the word in a particular sample and frequency of the word in the whole document.
Co-occurrence vector.	The co-occurrence matrix is based on the concept of similar words in the same context that tend to occur. together.

B. Static Word Embedding: Static word embedding is a prediction-based that provides probabilities to the words and maps each word into a vector. Static embeddings learn by training the lookup tables which converts words into dense vectors. This embedding is static in the way that they do not alter the context once been learned and also the embedding tables do not change among different sentences [23].

TABLE 2.2 – Static Word Embedding [23].

Static Word Embedding	Description
Word2Vec	Word2Vec method can generate word embedding by using dense representation. It is a predictive model and it provides probabilities to the words which excel in word similarity tasks.
Glove	The Glove is a count-based model. It is an unsupervised model for generating word vectors. Glove combines two features namely, local context window method and global matrix factorization.
Fast text	Fast text is an extension of Word2Vec. It identifies the words as n-gram of characters. It provides an efficient vector representation of rare words

TABLE 2.3 – Contextualized Word Embedding [23].

Contextualized Word Embedding	Description
ELMo	Elmo embedding is context-dependent and is character-based. A word may have dissimilar meanings depending upon the context where it is being used.
GPT-2	GPT-2 denotes Generative Pre-trained Transformer 2. It is a decoder only transformer.
BERT	BERT is the first unsupervised deep bidirectional system with a multi-layer bidirectional Transformer encoder.

C. Contextualized Word Embedding:

2.4 Text Classification Algorithms

2.4.1 Traditional Algorithms

2.4.1.1 Classification Algorithms

Classification algorithms are supervised ML algorithms that are used to classify, categorize, or label data points based on what it has observed in the past. Each classification algorithm, being a supervised learning algorithm, requires training data. This training data consists of a set of training observations where each observation is a pair consisting of an input data point, usually a feature vector like we observed earlier, and a corresponding output outcome for that input observation [21].

I will touch upon a couple of algorithms that are quite effective for text classification and try to explain them, keeping the mathematical formulae to the base essentials.

These algorithms are the following:

— Naïve Bayes

— Support Vector machines

— Decision Tree

— Random Forest

— K-Nearest Neighbor (KNN)

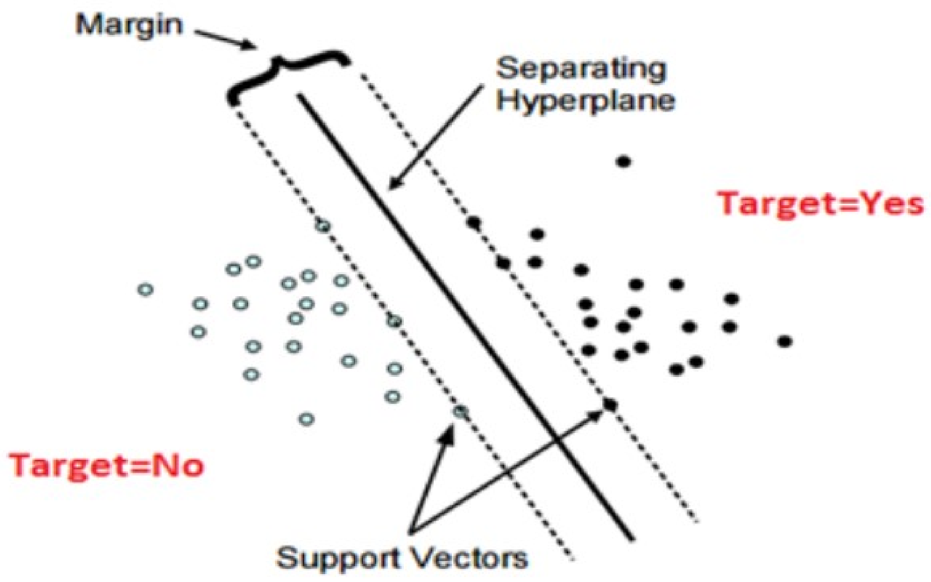


FIGURE 2.2 – Representation of Support Vector Machine (SVM) [22].

2.4.2 Machine Learning Techniques

2.4.2.1 Naïve Bayes Classification

Naïve Bayes is a simple probabilistic classifier which works on the assumption of conditional independence between the features of a text document. Given a text document, the Naïve Bayes classifier find out the class with maximum posterior probability [22]. It is based on the Bayes Rule shown in Equation 2.3.

$$p(c/d) = \frac{p(d/c) \times p(c)}{p(d)} \tag{2.3}$$

Where$p(c/d)$, is the probability of document d to belong to class c called the posterior probability, $p(d/c)$ is the likelihood and $p(c)$ is the prior.

Naïve Bayes classifies the document to the class which maximizes the posterior probability.

Naïve Bayes classification, they are Multivariate Bernoulli Model and the Multinomial Model. The Multivariate model has binary representation of features while the later represents the features with term frequency. Naïve Bayes classifier seem to work well even with the conditional independence assumption. Authors uses Naïve Bayes to perform spam email detection and reveals the necessity of increased training samples for an accurate classification [22].

2.4.2.2 SVM Classification

Support Vector Machines are linear classifiers suitable for classifying high dimensional data. SVM performs well in text classification scenario as it involves a high dimensional feature space. SVM tries to find out the maximum separating hyper plane between different classes [22].

In Figure 2.2 among the three lines the bold line best separates the two classes and it is the maximum marginal hyper plane. The linear equation describing this hyper plane will be of the form $Y = AX + b$. Here X is the feature vector representation, A

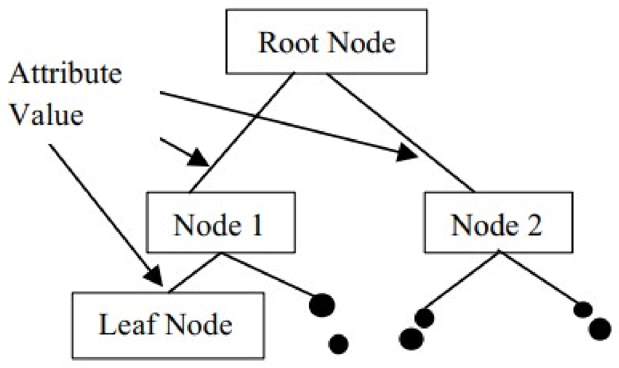


FIGURE 2.3 – The basic structure of a decision tree [23].

is the coefficient vector and b are a constant. This predictor hyper plane will classify the documents to different classes. The points on the dotted lines are the support vectors and they are the deciding factors in categorization. SVM can also model nonlinear decision boundaries in base feature space by applying kernel trick [22].

2.4.2.3 Decision Trees Algorithm

The decision tree algorithm is a part of the supervised learning algorithm. This algorithm is used for solving the regression and classification problems. The decision tree's structure is similar to a flowchart as shown in Figure 2.3 [23].
Root node: It comprises of the entire population which is further divided into multiple homogeneous sets.
Leaf /Terminal node: Nodes that do not divide further are called leaf/terminal nodes.
Decision tree begins with a single node (root node) that represents all the records of the training dataset. If the records are in the same category, then that node becomes leaf node and is assigned a category to which it belongs. If not, then this algorithm will use a method based on entropy, which is also called information gain and chooses the attribute which gives the best classification. This attribute becomes the test attribute of that node. For every test attribute's known value, a branch is created, and the dataset is divided into every branch in the same way. To create a sample decision tree for every division, this algorithm follows the same procedure recursively [23]. This recursive division stops only when the below given conditions are true:

— All the sample data of the given node belong to the same class.

— After the split, certain branches do not have any sample records.

— The major limitation of the decision tree is the overfitting problem.

2.4.2.4 Random Forests Algorithm

Random forest algorithm handles regression and classification problems. A forest is a group of trees whereas random forest is a group of classification trees. Random forest is constructed by merging the predictions of various trees; each one of them gets treated in isolation. Regression trees, also called decision trees, are a tree that consists of the members of the decision/predictor variable on its leaf nodes and the entities of other dependent variables live on the intermediate nodes. For example,

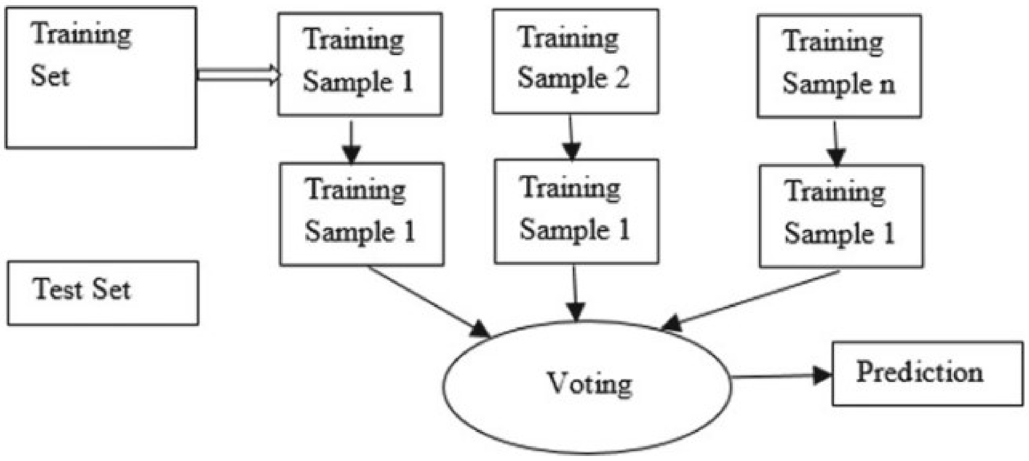


FIGURE 2.4 – Random Forest algorithms steps [23].

spam/no spam for emails, 0–9 for handwritten digits in pattern recognition, etc. Each tree gives its decision as output, and that output is its vote/solution for that problem. The last class is that class which has received most votes from the trees present in the forest. Working on this algorithm is in the image shown in Figure 2.4 [23].

— Start by selecting random samples from the given dataset.

— This algorithm will build a decision tree for every sample and will predict the output from every tree.

— Voting takes place for every predicted output.

— The output with most votes is the last output of the algorithm.

Constructing a random forest is very difficult and less intuitive when having a large accumulation of decision trees [23].

2.4.2.5 K-Nearest Neighbors Algorithm

K-NN algorithm assorts the objects based on the closest training examples for pattern recognition inside the feature space. Instead of learning from the data, this algorithm memorizes it [17]. In this algorithm, each sample is given a class depending on its surrounding samples. This algorithm will predict the class in which the unclassified sample will belong by considering the class of its nearest samples [23].

Figure 2.5.**a** shows the K-NN for k = 1; here, the new sample is classified with the help of only one known sample. Similarly, in Figure 2.5.**b**, the value of k is 4, which means the new sample is classified considering four of its closest/nearest neighbors. In this case, three neighbors are in the same class and one in a different class. Now, the class of unknown sample is predicted depending on the class in which its majority of the neighbors lie. The classification performance of the algorithm is dependent on the selection of K [23].

If K is very small, then the data is mislabeled, and if K is very large, then many outliers from its neighborhood are involved that creates a problem. In the case of regression problems, the average of the K-NN output is the final prediction as shown in Equation 2.4:

$$y = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{2.4}$$

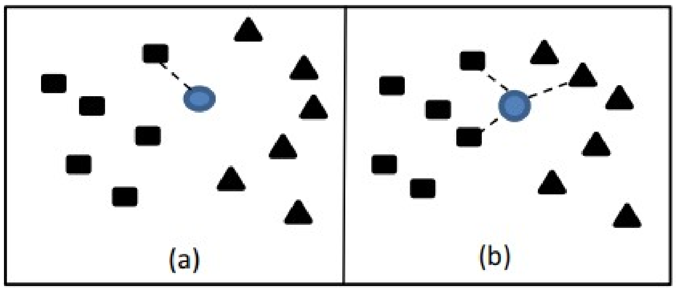


FIGURE 2.5 – KNN (a): $k = 1$, (b): k-NN for $k = 4$ [4].

where y_i is the ith case of the samples, and y is the outcome of the query point [23].

2.4.3 Deep Learning Models

Deep learning is an artificial neural network based on machine learning to extract higher level features from data through multiple layers of processing. Today text classification in natural language processing (NLP) is one of the most used methods to classify text documents which is based on predefined classes [30]. Deep learning models utilize hierarchical architectures to process low-level word vector features into high-level abstract feature vectors. Unlike traditional machine learning approaches, their performance relies heavily on datasets rather than user knowledge. Consequently, deep learning techniques have significantly improved performance in computer vision and other fields [32].

Convolutional Neural Networks (CNNs) (LeCun, Bottou, Bengio, & Haffner, 1998) and Recurrent Neural Networks (RNNs) (Rumelhart, Hinton, & Williams, 1986) are the two most popular approaches for text classification based on deep learning [36].

2.4.3.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) play a pivotal role in capturing long-range dependencies in text, making them widely employed for tasks such as text classification. In our proposed RNN-based text classification model, each input word undergoes representation through word embedding technology. These embedding word vectors are sequentially fed into RNN cells, where the output of each cell matches the input vector dimension. Parameter sharing across different segments of the model ensures consistent weights for each input word. The final prediction for the input text's label is derived from the last output of the hidden layer [24].
Example: As shown in Figure 2.6, firstly, each input word is represented by a specific vector using a word embedding technology. Then, the embedding word vectors are fed into RNN cells one by one. The output of RNN cells is the same dimension with the input vector and are fed into the next hidden layer. The RNN shares parameters across different parts of the model and has the same weights of each input word. Finally, the label of input text can be predicted by the last output of the hidden layer [24].

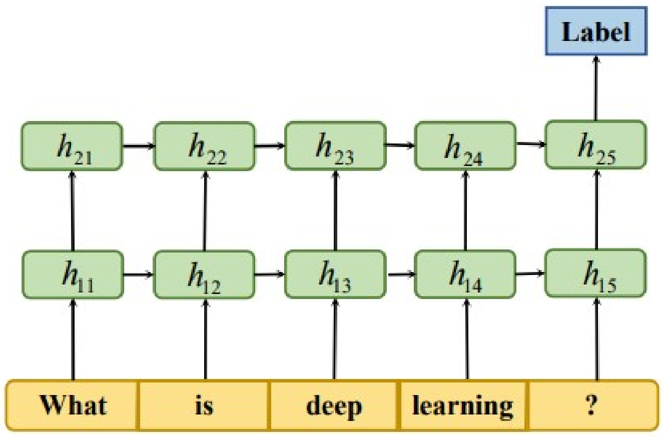


FIGURE 2.6 – The RNN based model [24].

2.4.3.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have been proposed for image classification, leveraging convolving filters to extract features from pictures. In contrast to Recurrent Neural Networks (RNNs), CNNs can concurrently apply convolutions using different kernels to multiple segments of a sequence. This versatility makes CNNs applicable to various Natural Language Processing (NLP) tasks, including text classification. In the context of text classification, the input text is transformed into a vector representation akin to image processing. The process involves splicing word vectors into a matrix, which is then inputted into a convolutional layer containing filters of various dimensions. Subsequently, the output of the convolutional layer undergoes pooling and concatenation in the pooling layer to yield the final vector representation of the text, facilitating category prediction. An exemplary model for text classification using CNNs is Text CNN, introduced by Kim, which excels in identifying discriminative phrases through one convolutional layer and static word vectors [24].
Example: In Figure 2.6, firstly, the word vectors of the input text are spliced into a matrix. The matrix is then fed into the convolutional layer, which contains several filters with different dimensions. Finally, the result of the convolutional layer goes through the pooling layer and concatenates the pooling result to obtain the final vector representation of the text. The category is predicted by the final vector [24].

2.5 Supervised Learning and Training Data in Text Classification

Supervised learning techniques are essential in text classification, relying on externally supplied sample data or instances to produce general patterns and hypotheses.

2.5.1 Supervised Machine Learning

Supervised Machine Learning techniques are those in which the sample data or instances are supplied externally. Using those instances, it produces general patterns and hypothesis. In general, Supervised Machine Learning algorithm works on prior information. Moreover, Supervised Machine Learning algorithms are further

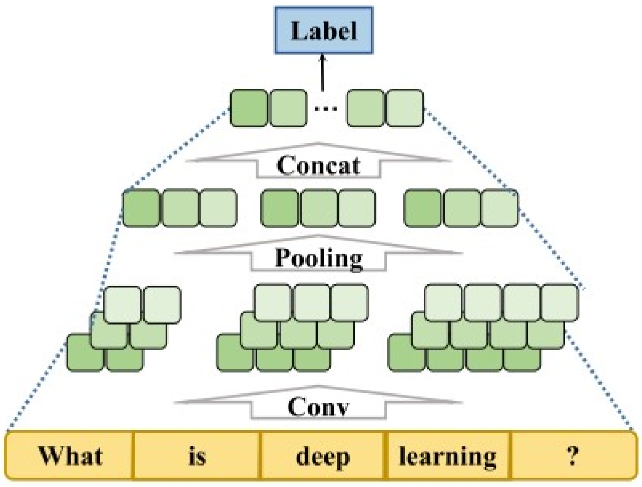


FIGURE 2.7 – The CNN based model [24].

divided into single label and multi-label. Single label algorithms are used to assign a document to a single category and multi-label algorithms work on assigning a document to more than one category [25].

2.5.2 Supervised Learning in Text Classification

We present Supervised Machine Learning algorithms tailored for text classification, with a focus on Feature Selection. This involves identifying and retaining crucial data features to streamline text classification by removing redundancies, ultimately reducing data dimensionality. Text categorization in Supervised Machine Learning falls into two main categories: Single and Hybrid approaches. Single approaches employ one algorithm, such as Bayesian or K-Nearest Neighbor (KNN), organizing training data into subsets with similar class labels. Meanwhile, Hybrid approaches combine multiple methods to enhance performance and noise tolerance, like the two-phase SVM and KNN classifier or integrating Random Forest with the Rocchio algorithm [25]. Figure 2.8 illustrates the Machine Learning Algorithms categorized into Single and Hybrid approaches, further dividing them based on existing classification algorithms.

2.6 Text Pre-processing Techniques

2.6.1 Definition

Text pre-processing is an essential part of any NLP system, since the characters, words, and sentences identified at this stage are the fundamental units passed to all further processing stages, from analysis and tagging components, such as morphological analyzers and part-of speech taggers, through applications, such as information retrieval and machine translation systems. It is a Collection of activities in which Text Documents are pre-processed [35]. another definition, pre-processing a text simply means to bring the document to a format that is easily understandable, predictable and analyzable by the machine through the various machine learning algorithms [26].

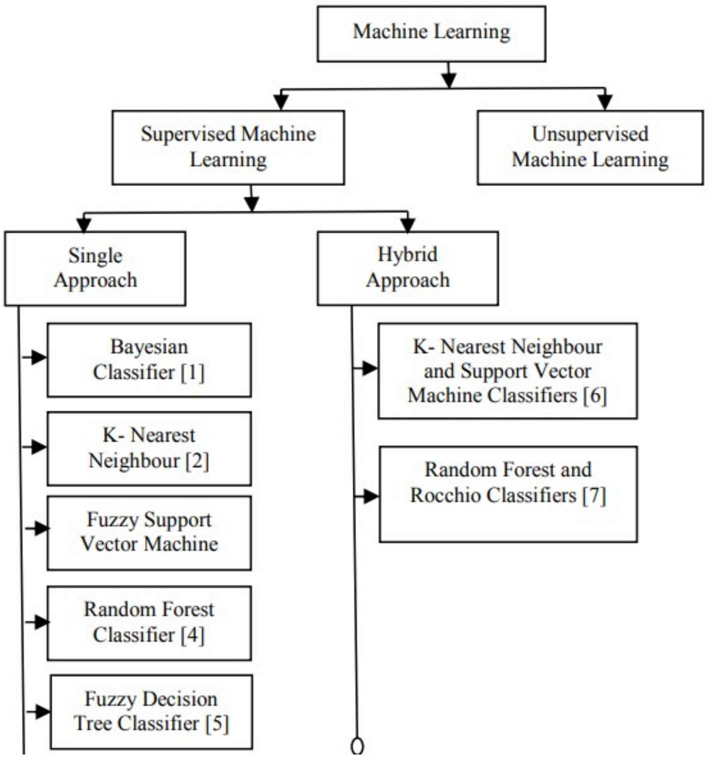

Machine Learning
Supervised Machine Learning
Unsupervised Machine Learning
Single Approach
Hybrid Approach
Bayesian Classifier [1]
K- Nearest Neighbour [2]
Fuzzy Support Vector Machine
Random Forest Classifier [4]
Fuzzy Decision Tree Classifier [5]
K- Nearest Neighbour and Support Vector Machine Classifiers [6]
Random Forest and Rocchio Classifiers [7]

FIGURE 2.8 – Machine Learning Text Classification Approaches [25].

2.6.2 Tokenization

Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. The aim of the tokenization is the exploration of the words in a sentence [35].

2.6.2.1 Example

“NLP is the future of Speech Recognition Systems!” This sentence will be tokenized as: “NLP”, “is”, “the”, “future”, “of”, “Speech”, “Recognition” “Systems”, “!” [7].

2.6.2.2 Challenges in Tokenization

The complexity of tokenization varies depending on the language’s structure. Languages like English and French, where words are separated by spaces (space-delimited), present a simpler challenge. In contrast, unsegmented languages like Chinese and Thai lack clear word boundaries, requiring additional information about word formation (morphology) and vocabulary (lexicon) to break down sentences into tokens [35].

2.6.3 Stop-Words Removal

Text mining often encounters a challenge: frequent yet uninformative words. These "stop words," like "and," "are," and "this," contribute little to the overall meaning or content of documents. Their high frequency can hinder the process of understanding the document’s true message. Stop words are not useful for classifying documents and should ideally be removed.

However, creating a definitive stop word list is difficult. It can vary based on the specific text source and domain. While removing stop words reduces data size and potentially improves system performance, it's important to acknowledge that this process can also discard potentially relevant information depending on the context. It's crucial to consider the trade-off between efficiency and potential loss of meaning when deciding on stop word removal for a specific text mining application [35].

2.6.4 Stemming

Stemming is a technique in text processing that aims to group words with similar meanings into a single form, called the stem. This simplifies the representation of words. Imagine the words "presentation," "presented," and "presenting" - stemming would reduce them all to the common stem "present.". The underlying principle of stemming in information retrieval (IR) is that searching for a term like "presenting" implies interest in documents containing words like "presentation" and "presented" as well. However, stemming is not perfect and can introduce two types of errors [35].

2.6.4.1 Type of errors

Over-stemming: This occurs when two words with distinct meanings are mistakenly reduced to the same stem. For instance, stemming "running" and "runner" might both result in "run", which can be misleading. (This is also known as a false positive). Under-stemming: Here, two words that share a common meaning are not recognized as such. Stemming "agrees" and "agreed" might leave them as separate entities, even though they convey the same concept. (This is also known as a false negative) [35].

2.6.4.2 Example

For Example: 'Studying' changes to 'Study' wherein the "Ing" is chopped off while retaining the root word. But stemming applied to words like 'Coding' will result in 'Cod' which doesn't make sense. Thus, the need for stemming depends again on the problem statement [26].

2.7 Evaluation Metrics in Text Classification

In order to assess the performance of our classifier and compare it with existing methods, evaluation becomes crucial. Researchers consistently prioritize evaluating their methods to identify any inconsistencies or errors and implement improvements. However, this task is challenging due to the absence of standardized data collection methods. Various performance metrics, such as recall, F-measure, accuracy, and precision, are utilized to evaluate the algorithms [33].

2.7.1 Accuracy

Accuracy (Equation 2.5) refers to the ratio of sum of true positive (number of items correctly predicted belonging to the positive class) and true negative (number of items correctly predicted belonging to the negative class) to the total number of elements predicted belonging to both the positive and negative class.

TABLE 2.4 – The Notations Used in Evaluation Metrics [24].

Notations	Descriptions
TP	True positive
FP	False positive
TN	True negative
FP	false positive of the t^{th} label on a text

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

2.7.2 Precision

Precision (Equation 2.6) refers to the ratio of correctly predicted positive items to all the positive predicted items.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

2.7.3 Recall

Recall (Equation 2.7) refers to the ratio of correctly predicted positive items to all the actual positive items [33].

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

2.7.4 F1

F1 (Equation 2.8) is the harmonic average of Precision and Recall [24].

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.8}$$

2.7.5 Macro-F1

The Macro-F1 calculates the average F1 of all labels. Unlike Micro-F1, which sets even weight to every example, Macro-F1 sets the same weight to all labels in the average process [24]. Formally, Macro-F1 is defined as given by Equation 2.9.

$$Macro - F1 = \frac{1}{S} \sum_{t \in S} \frac{2 \times P_t \times R_t}{P_t + R_t} \tag{2.9}$$

Where $P_t = TP_t/(TP_t + FP_t)$, $R_t = TP_t/(TP_t + FN_t)$.

2.8 Challenges and Considerations

2.8.1 Handling Noisy Text and Irrelevant Features

In text classification, noisy text and irrelevant features pose significant challenges, hindering algorithm performance. Noisy text includes irrelevant or erroneous data, while irrelevant features mislead classification models. Robust preprocessing techniques like normalization and spell checking are crucial for enhancing data quality and filtering out noise. Advanced machine learning algorithms, such as ensembles or deep learning, can mitigate the impact of noise on classification accuracy [37] [39].

— **Challenges:** Noisy text can lead to misclassifications and hinder model performance. Irrelevant features can increase training time and reduce model accuracy.

— **Solutions:** Pre-processing techniques like text cleaning, normalization, and stemming/lemmatization can address noise. Feature selection methods can identify and remove irrelevant features.

2.8.2 Addressing Bias in Text Classification

Bias in text classification poses a significant concern as it can perpetuate discrimination and reinforce existing societal inequalities. Bias can manifest in various forms, including but not limited to gender bias, racial bias, and cultural bias, leading to unfair treatment or misrepresentation of certain groups within the text data. Addressing bias requires a multifaceted approach involving data collection, preprocessing, algorithmic design, and model evaluation. It is crucial to scrutinize the training data for biased patterns and take corrective measures such as data augmentation, bias-aware feature engineering, or algorithmic adjustments to mitigate bias during the classification process. Additionally, incorporating fairness metrics and conducting thorough bias analysis can help assess and mitigate bias in text classification systems, promoting equitable outcomes across diverse demographic groups [38].

2.8.3 Ethical Considerations in Text Classification

Ethical considerations play a pivotal role in the development and deployment of text classification systems, particularly concerning privacy, transparency, and accountability. Text classification algorithms often deal with sensitive information, raising concerns regarding user privacy and data protection. It is imperative to adhere to established ethical guidelines and regulatory frameworks, such as GDPR or HIPAA, to ensure lawful and ethical handling of textual data. Moreover, ensuring transparency in the classification process by providing clear explanations of model predictions and decision-making criteria fosters trust and accountability among stakeholders. Ethical considerations also encompass issues of algorithmic fairness, cultural sensitivity, and potential societal impacts, emphasizing the need for interdisciplinary collaboration and ethical oversight throughout the lifecycle of text classification projects. By addressing these ethical considerations proactively, practitioners can uphold ethical standards and promote responsible innovation in text classification technology [40].

Text classification, while powerful, raises important ethical concerns that require careful attention. These considerations are crucial for ensuring the responsible and fair use of these models in real-world applications.

1. **Key Ethical Concerns:**

    — **Fairness and non-discrimination:** Text classification systems shouldn't discriminate against any specific group or individual based on protected characteristics like race, gender, or religion [28].

— **Transparency and explainability:** It's essential to understand how the model makes decisions and be able to explain those decisions in human-understandable terms. This allows for identifying and addressing potential biases and ensuring fair treatment of individuals [28].

— **Privacy and security:** The privacy of individuals whose data is used to train and test the model needs to be protected. This includes anonymizing data, obtaining informed consent, and implementing robust security measures [28].

— **Accountability and responsibility:** Developers and users of text classification systems hold responsibility for the model's outcomes. This involves addressing unintended consequences, mitigating potential harm, and ensuring accountability for biased or unfair results [28].

2. **Additional Considerations:**

   — **Potential for misuse:** Text classification models can be misused for malicious purposes, such as hate speech detection or profiling individuals based on their online activity. It's critical to consider potential misuse cases and implement safeguards to prevent them [28].

   — **Impact on society:** The widespread use of text classification models can have significant societal impact. It's important to consider potential implications and ensure that the technology is used for beneficial purposes aligned with ethical values [28].

2.9 Applications

Text classification and categorization is used in several real-world scenarios and applications, including the following [21].

1. **Spam Detection:** Text classification is commonly used in email filtering to identify and categorize messages as spam or non-spam.
2. **Sentiment Analysis:** Businesses often use sentiment analysis to gauge customer opinions and feedback. This can be applied to social media comments, product reviews, or survey responses to understand customer sentiment towards a product, service, or brand.
3. **Music or movie genre categorization:** Text classification can be employed to categorize music or movie titles into specific genres. This is particularly useful for online platforms, streaming services, or media libraries, allowing users to explore and discover content based on their preferred genres.
4. **News articles categorization:** Text classification is applied to automatically categorize news articles into different topics or genres, such as politics, sports, technology, or entertainment. This helps in organizing large volumes of news content and facilitates efficient content retrieval for users interested in specific subjects.
5. **Language detection:** Text classification is used to identify the language of a given text, which is useful for multilingual applications and content localization. The possibilities with text data are indeed endless, and with a little effort you can apply classification to solve various problems and automate otherwise time-consuming operations and scenarios [21].

2.10 Conclusion

The chapter covered a range of text representation techniques, from Bag-of-Words to advanced embeddings like BERT and GPT, vital for effective feature extraction. Various classification algorithms were discussed, underscoring the importance of labeled training data and preprocessing for model performance. Evaluation metrics and considerations for noisy text and ethical concerns were addressed. Real-world applications highlighted text classification's practicality in sentiment analysis and spam detection. Text classification serves as a cornerstone in NLP, aiding decision-making and information organization. This understanding, alongside fuzzy logic exploration, sets the stage for implementing ML methods in spam email detection in the next chapter.

Chapter 3

Enhancing Spam Email Detection through Fuzzy KNN Algorithm

3.1 Introduction

As communication technology continues to advance, so too does the sophistication of malicious activities, particularly in the realm of email-based threats such as spam. The battle against spam email has been ongoing for decades, with traditional methods often struggling to keep pace with the evolving tactics of spammers. In response to this challenge, the integration of fuzzy logic with the K-Nearest Neighbors (KNN) algorithm has emerged as a promising approach to enhance spam email detection. This chapter delves into the theoretical foundations, related work, and proposed algorithm aimed at improving spam email detection through the fusion of fuzzy logic with the KNN algorithm. By leveraging the principles of fuzzy logic, which enable the representation of uncertainty and ambiguity, and the robustness of the KNN algorithm in pattern recognition tasks, we aim to develop a more effective and adaptable solution for identifying spam emails. We present enhanced algorithm for spam email detection, which builds upon the foundation of Fuzzy KNN. Our approach involves refining and optimizing the Fuzzy KNN algorithm specifically to address the challenges inherent in identifying spam emails. Through a meticulous examination of the algorithmic intricacies, we introduce novel enhancements aimed at bolstering its effectiveness and adaptability in real-world scenarios. In our thorough exploration, we detail the algorithmic steps, highlighting the nuances that differentiate our proposed approach. We also discuss innovative preprocessing techniques tailored to email data, aiming to strengthen the algorithm's ability to discern meaningful patterns amidst the noise present in spam email datasets. Additionally, we present strategies for meticulous distance metric selection and parameter tuning, crucial components for optimizing algorithm performance and achieving robust spam detection capabilities.

3.2 Theoretical Background

3.2.1 K-Nearest Neighbors (KNN) Algorithm

3.2.1.1 Definition

KNN stands out as one of the most straightforward algorithms to put into practice. It operates by retaining all available data and then determining the classification of a new data point based on its likeness to other data points. While its primary use is for classification, it can also serve in regression analysis. This algorithm is termed non-parametric because it does not rely on assumptions about the underlying data

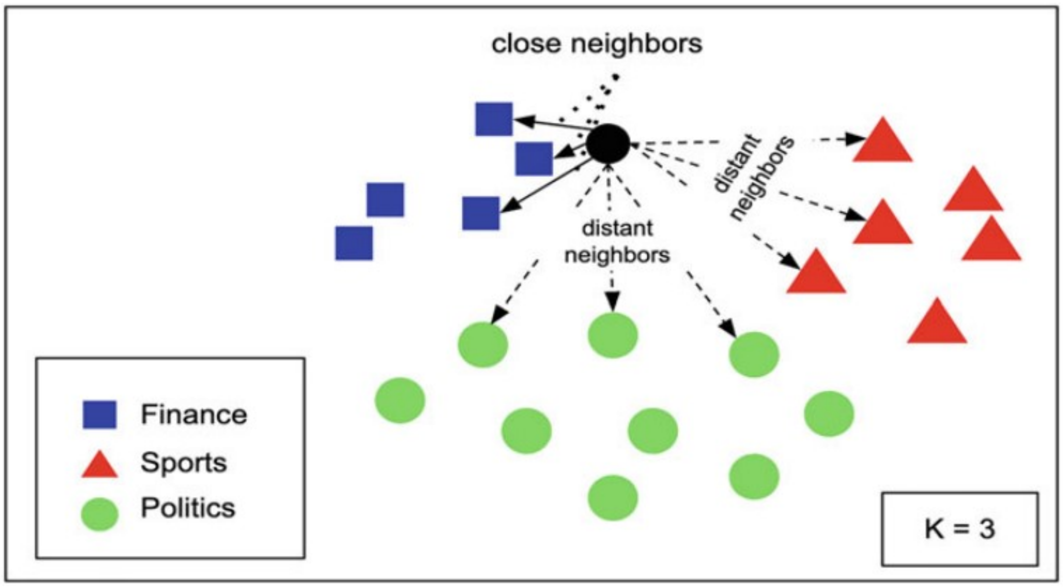


FIGURE 3.1 – K-Nearest Neighbor Classification [41].

[41]. In essence, KNN endeavors to classify a data point by examining its nearby instances, presuming that it belongs to the same category as the majority of its neighbors. For instance, one might predict an individual's voting preference by observing the voting trends among her neighbors. Naturally, if additional details about the person and her neighbors are available—such as age, income, and education level—making predictions based on neighbors with similar attributes leads to more accurate results. Throughout the subsequent discussions, all mentions of KNN pertain to its supervised application [43]. The diagram shows the result of KNN classification. Given a set of data points (news articles), it classifies the documents into three categories, sports, politics, and finance. This is depicted in Figure 3.1 [41].

Note that we must convert the text documents into vectors before they can be input into the algorithm. After we classify the training dataset into different categories, the category of an unseen data point can be determined by finding its closeness to the neighboring data points. For example, the data point in question (black rectangle) has three close blue points and two red points, followed by one green point in that order. As the number of blue points is more than compared to the other two, the data point would be classified as belonging to the blue group, that is, the finance category [41].

3.2.1.2 KNN Algorithm

The Algorithm:

a) Distance Function The effectiveness of the KNN algorithm hinges on the utilization of a distance metric. Numerous distance functions are available to measure proximity or similarities, including Euclidean distance, Hamming distance, Manhattan distance, Minkowski distance, cosine similarity, Tanimoto distance, Chebyshev distance, and Jaccard distance. These distance measures play a crucial role in determining the nearest neighbors and ultimately influence the prediction accuracy of the algorithm [43].

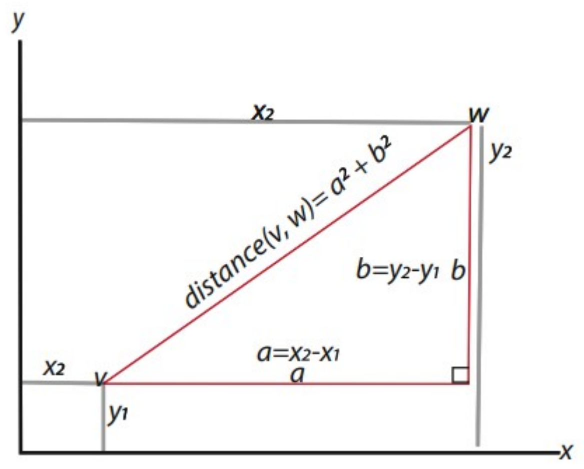


FIGURE 3.2 – Euclidean Distance based on the Pythagorean Theorem [43].

Euclidean Distance: The Euclidean distance between two vectors $v(v_1, v_2, \ldots, v_n)$ and $w(w_1, w_2, \ldots, w_n)$ is given by Equation 3.1.

$$\text{Euclidian Distance}(v, w) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} \tag{3.1}$$

In a machine learning dataset, v and we are two vectors with attributes v_i and w_i, i = 1 to n, where n is the number of available attributes. In the case of a two-dimensional space, the Euclidean distance (Equation 3.2) between two points $v(x_1, y_1)$ and $w(x_2, y_2)$ is computed by the Pythagorean theorem (Figure 3.2) [43].

$$\text{Euclidian Distance}(v, w) = \sqrt{\sum_{i=1}^{n} (x_i - x_i)^2 + (y_i - y_i)^2} \tag{3.2}$$

When input variables are similar in type, Euclidean distance is usually used.
Manhattan Distance: Manhattan distance (Equation 3.3) calculates the sum of the absolute difference between two vectors $v(v_1, v_2, \ldots, v_n)$ and $w(w_1, w_2, \ldots, w_n)$ [43].

$$\text{Manhattan Distance}(v, w) = \sqrt{\sum_{i=1}^{n} |v_i - w_i|} \tag{3.3}$$

Manhattan distance can be computed as shown in Equation 3.4.

$$\text{Manhattan Distance}(v, w) = \sqrt{|x_2 - x_1| + |y_2 - y_1|} \tag{3.4}$$

When input variables are not similar in type (e.g., age, gender, weight), Manhattan distance is usually used.

Equations 3.3 and 3.4 illustrate two ways to express the Manhattan distance between two vectors v and w.

Equation **??** gives a general formulation with a sum of the absolute differences between the respective components of vectors v and w. This form is suitable for vectors of any dimension.

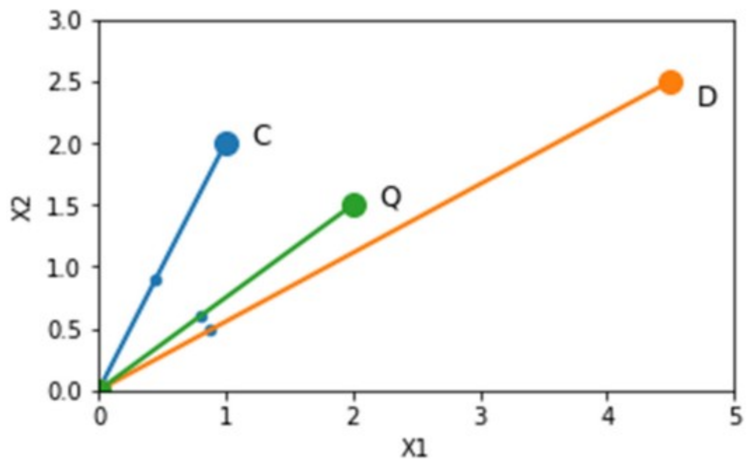


FIGURE 3.3 – Cosine Similarity [42].

Equation **??** is a simplified specific form for the case of 2-dimensional vectors, with only two components x and y. It is equivalent to the general Equation 3.3 but explicit for 2D vectors.

So, Equation 3.3 is the general formulation for vectors of any dimension, while Equation 3.4 is a particular case in 2 dimensions, simpler to write for this specific situation.

Minkowski Distance: The Minkowski distance i given by Equation 3.5.

$$\text{Minkowski Distance}(v, w) = \sum_{i=1}^{n} (|v_i - w_i|^p)^{1/p} \tag{3.5}$$

Where p represents the order of the distance. When p = 2, the Minkowski distance corresponds to the Euclidean distance, and when p = 1, it corresponds to the Manhattan distance. When p tends towards infinity, the Minkowski distance corresponds to the Chebyshev distance [43].

Cosine Similarity:

Similar to the Minkowski distance, Cosine Similarity operates on feature vector data. However, it assesses similarity based on the angles between the feature vectors, as depicted in Figure 3.3. Although C might appear closer to Q according to the Euclidean distance, D emerges as closer to Q when accounting for the angles between the feature vectors [42].

The Cosine similarity between a query q and x_i is as follows:

$$\text{Cos}(\mathbf{q}, \mathbf{x}_i) = \frac{\sum_{f \in F} \mathbf{q}_f \cdot \mathbf{x}_{if}}{\sqrt{\sum_{f \in F} \mathbf{q}_f^2} \sqrt{\sum_{f \in F} \mathbf{x}_{if}^2}}. \tag{3.6}$$

— q and x are the two vectors for which we want to calculate the similarity.

— f represents the dimensions/features of the vectors.

— q_f and x_{if} are the components of vectors q and x for dimension f.

This is the dot product of the feature values normalized by the lengths of the feature vectors. Cosine similarity is a popular metric in text analytics. When text is processed as a bag of words the features are word counts and Cosine similarity has the advantage that it is independent of the magnitude of the feature vectors. Thus, it is insensitive to document size. Cosine similarity requires that all feature values are positive real numbers [42].

If feature values are positive, then the Cosine similarity will be in the interval [0, 1]. So, we can define a Cosine distance measure as shown in Equation 3.7.

$$\mathrm{CosD}(\mathbf{q}, \mathbf{x}_i) = 1 - \mathrm{Cos}(\mathbf{q}, \mathbf{x}_i) \tag{3.7}$$

Hamming Distance: Hamming distance quantifies the similarity between two strings of identical length by counting the positions where their characters differ. For instance, consider strings S1 as "Health Informatics" and S2 as "Health Information." The Hamming distance between them is 2, indicating two differing positions. The greater the distance, the more dissimilar the strings. A Hamming distance of 0 signifies identical strings. It's important to note that Hamming distance is applicable only when comparing strings of equal length [43].

b) KNN for Classification KNN for Classification: Typically, when employing the K-Nearest Neighbors (KNN) algorithm for classification tasks, you'll work with a dataset comprising various features or attributes and their corresponding class labels. These class labels often represent binary outcomes, such as 'readmitted' or 'not readmitted' in a medical context, 'satisfied' or 'not satisfied' in a customer feedback scenario, or 'true' and 'false' in a binary decision problem like spam detection. Alternatively, they might denote categories, such as different movie ratings. The KNN algorithm works by identifying the K nearest data points to a given test point based on a chosen distance metric (e.g., Euclidean distance), and then assigning the majority class among these neighbors to the test point. This approach makes KNN intuitive and easy to understand, as it essentially relies on the idea that similar data points tend to belong to the same class or category [43]. The algorithm is as follows [43].

Algorithm 1 K-Nearest Neighbors (KNN) Algorithm.

```
Require: New instance $\mathbf{v}$, Dataset $\mathcal{D} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$, Number of neighbors $k$
 1: Initialize: Distance list $\mathcal{L} \leftarrow []$
 2: for each point $\mathbf{w}_i$ in the dataset $\mathcal{D}$ do
 3:     Compute the distance between $\mathbf{v}$ and $\mathbf{w}_i$: $d(\mathbf{v}, \mathbf{w}_i)$
 4:     Store the distance $d(\mathbf{v}, \mathbf{w}_i)$ in the list $\mathcal{L}$
 5: end for
 6: Sort the list $\mathcal{L}$ by distance in ascending order
 7: Select the top $k$ instances from $\mathcal{L}$
 8: Count the votes for each class in the top $k$ instances
 9: Assign the class with the highest votes to $\mathbf{v}$
10: return The predicted class for $\mathbf{v}$
```

c) KNN for Regression Regression involves utilizing available data to forecast forthcoming numerical values [43]. For instance, one might utilize the weekly study hours of students to predict their future GPA, leverage various attributes of a house (such as room count, dimensions, and location) to estimate its potential sale price [43], or predict the expected energy consumption of a household based on factors like the number of occupants, square footage, and climate. In the context of regression problems, the K-Nearest Neighbors (KNN) algorithm aims to predict a continuous value. It achieves this by identifying the k closest instances (i.e., those most similar) to the new data point under consideration and calculating the mean or median of their outcomes. This mean or median value then serves as the predicted outcome for the new data point. Thus, while the fundamental KNN algorithm remains unchanged, it adapts for regression by determining the average or median of the k-nearest neighbors' outcomes, rather than employing a voting mechanism to determine a winning class [43].

Advantages and Disadvantages of KNN

Advantages	Disadvantages
1. Training is very fast. 2. Simple and easy to learn 3. Robust to noisy training data 4. Effective if training data is large	1. Biased by value of k 2. Computational complexity 3. Memory limitation 4. Being a supervised learning lazy algorithm (i.e., runs slowly) 5. Easily fooled by irrelevant attributes

TABLE 3.1 – Advantages and Disadvantages of KNN [45].

3.2.2 Fuzzy K-Nearest Neighbors (Fuzzy KNN) Algorithm

3.2.2.1 Introduction

Fuzzy logic provides a framework to model and reason with vague or imprecise information, making it suitable for various real-world applications where uncertainty or imprecision is present.

In the context of the K-Nearest Neighbors (KNN) algorithm, fuzzy logic can be employed to address the limitations of the traditional KNN approach. The traditional KNN algorithm treats all neighbors within the defined neighborhood equally, regardless of their proximity to the new data point being classified. This equal weighting of neighbors can lead to suboptimal performance, especially when decision boundaries are complex or when the data contains noise. By incorporating fuzzy logic into the KNN algorithm, the resulting Fuzzy KNN algorithm assigns varying membership degrees or weights to the neighbors based on their proximity to the new data point [47]. Neighbors that are closer to the new data point are assigned higher membership degrees, while those that are farther away are given lower membership degrees. This approach allows the algorithm to handle uncertainty and imprecision in the data more effectively, potentially improving classification performance by considering the relative closeness of each neighbor to the new data point.

3.2.2.2 Explanation of how Fuzzy KNN extends traditional KNN by incorporating fuzzy logic

K-nearest neighbors (KNN) and fuzzy K-nearest neighbors (FKNN) are classification algorithms used in machine learning for pattern recognition. While KNN relies on majority class voting among nearest neighbors, FKNN introduces fuzzy membership functions to consider data point uncertainties. FKNN tends to outperform KNN in datasets with overlapping classes or noise due to its nuanced decision-making process. However, FKNN's computational complexity is higher due to fuzzy membership calculations. Mining with FKNN involves applying the algorithm alongside data mining techniques for robust results, especially in uncertain data scenarios, although computational overhead may be a concern in large-scale tasks [66].

Fuzzy KNN extends traditional KNN (K-Nearest Neighbors) by incorporating fuzzy logic, which allows for more flexible and nuanced classification in situations where data points may belong to multiple classes to varying degrees. Here's how it works:

1. Traditional KNN (K-Nearest Neighbors):

— Classifies a new data point by finding the k nearest neighbors (data points) in the training set based on a distance metric (e.g., Euclidean distance).

— Assigns the class label that is most frequent among those k neighbors.

— Assumes crisp (binary) class memberships, meaning a data point belongs definitively to one class or another [65].

2. Fuzzy Logic:

— Deals with degrees of truth, allowing for data points to have partial membership in multiple classes.

— Represents uncertainty using membership functions, which map a data point's features to a value between 0 (no membership) and 1 (full membership) [5].

3. Fuzzy KNN (Fuzzy K-Nearest Neighbors):

— Extends traditional KNN by incorporating fuzzy logic to handle data points that may lie on class boundaries or exhibit some degree of uncertainty.

— Employs membership functions to determine the degree of membership of a new data point in each class based on its similarity to the k nearest neighbors.

— Weights the votes of neighbors according to their membership values in the predicted class.

3.3 Related Work

3.3.1 Overview of traditional spam email detection techniques

a) Standard Spam Filtering Method: It is a system that employs a predefined set of rules and protocols to classify emails. Figure 3.4 depicts a standard approach to spam filtering. Initially, content filters are utilized, leveraging artificial intelligence techniques to identify spam messages. Following this, the email header filter extracts header information from emails. Subsequently, blacklist filters are applied to block emails originating from blacklisted sources and prevent spam. Rule-based filters are then employed to identify senders based on subject lines and user-defined parameters. Finally, permission and task filters allow account holders to manage outgoing emails effectively [52].

b) The Client-Side Spam Filtering: A client is a person who can use the Internet or email network to send or receive an email. Spam detection at the client point offers different rules and mechanisms to ensure secure communications transmission between people and organizations. For transmission of data, a client should deploy multiple existing frameworks on his/her system. Such systems connect with client mail agents and filter the client's mailbox by compositing, accepting, and managing the incoming emails [52].

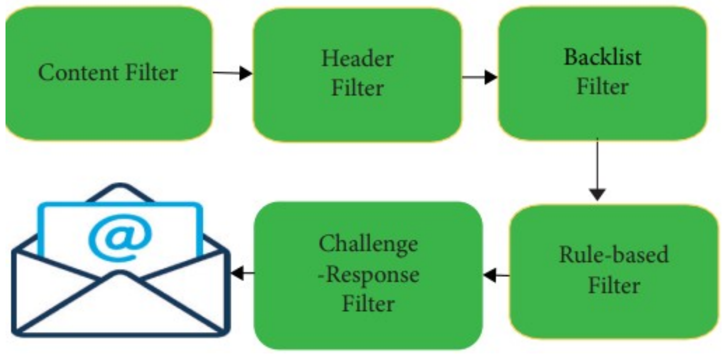


FIGURE 3.4 – Standard Spam Filtering [52].

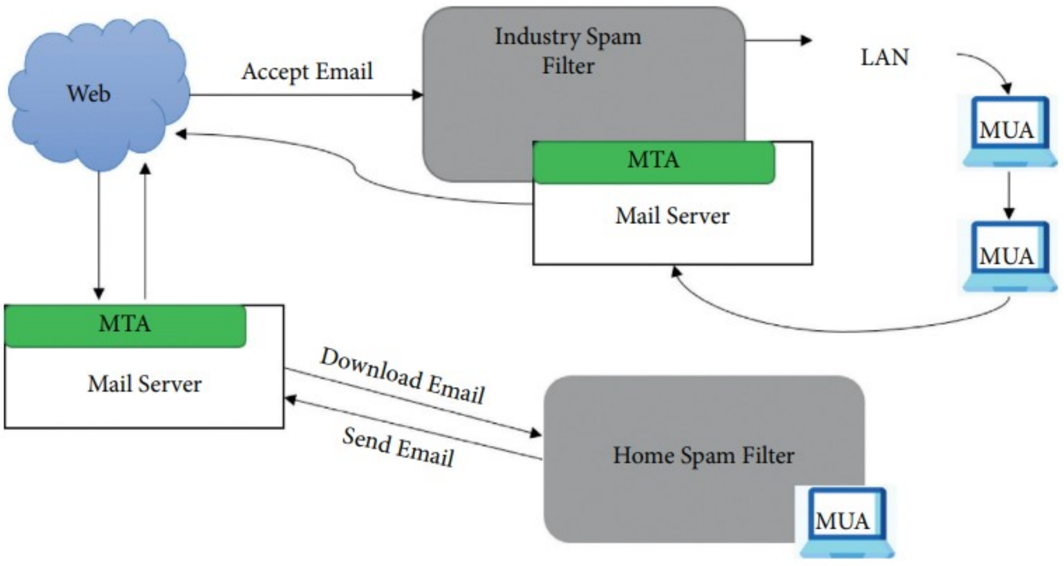


FIGURE 3.5 – Client-Based and Enterprise-Level Spam Filtering [52].

c) Enterprise Level Spam Filtering: involves implementing various filtering frameworks on the server, which interact with the mail transfer agent to categorize incoming emails as either spam or legitimate (ham). This system is utilized consistently and effectively across a network by clients to filter emails. Traditional spam detection methods typically rely on ranking emails based on predefined rules. Each email is assigned a score or rank using a specified ranking function, allowing for the classification of messages as either junk mail or legitimate. To combat evolving spamming techniques, these rules are regularly updated, often employing list-based techniques to automatically block suspicious messages. The architecture of client and enterprise-level spam filtering processes is depicted in Figure 3.5, adapted from Bhuiyan et al. [52].

Figure 3.5 illustrates how emails are accepted, transferred, filtered for spam, and finally delivered to users through various agents and filters on a local network or via the internet. Below is an explanation of the different acronyms used in the figure:

— MTA (Mail Transfer Agent) The MTA is software that transfers email messages from one computer to another. It uses protocols such as SMTP (Simple Mail Transfer Protocol) to send and receive emails between mail servers.

— MUA (Mail User Agent) The MUA is a client software used to read, compose,

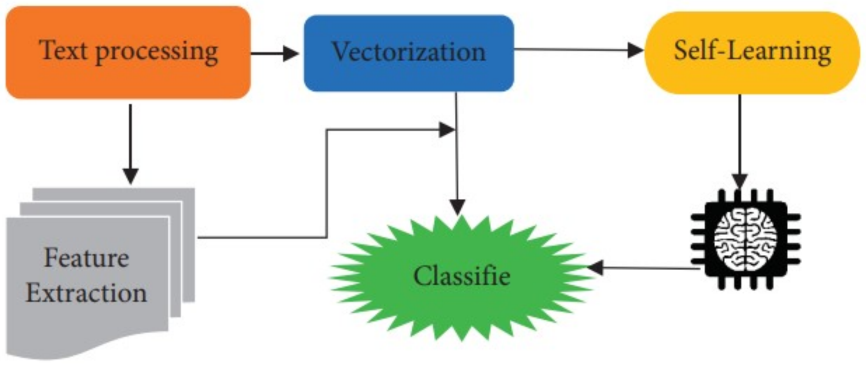


FIGURE 3.6 – Case-Based Spam Filtering [52].

and send email messages. Examples of MUAs include Outlook, Thunderbird, and webmail applications like Gmail.

— LAN (Local Area Network) The LAN is a network that connects computers within a limited area, such as a building or campus. In this figure, it shows the connection between client computers (MUAs) and the mail server on a local network.

— Industry Spam Filter This is a spam filter used by organizations to filter out unwanted emails before they reach the end users. It is placed before the MTA of the industry's mail server to block spam at the server level.

— Home Spam Filter

This is a spam filter used at home to filter out unwanted emails before they reach the user's MUA. It is placed after downloading emails from the mail server. Richard Segal, Jason Crawford, and their IBM colleagues present a comprehensive overview of enterprise-level spam email detection techniques, encompassing tokenization, DNS analysis, and machine learning classifiers. Their research emphasizes the need for adaptable pipelines to counter evolving spam tactics. Integration of classifiers like naive Bayes and Chung-Kwei aims to bolster accuracy and robustness. Their solution, "Spam Guru," combines machine learning with intelligent design, offering a scalable anti-spam solution for enterprises. Enterprise anti-spam filtering is crucial in today's digital landscape, enhancing security and efficiency. Machine learning techniques and dynamic classification mechanisms adapt to evolving spam tactics, while user feedback and configurability ensure effective management. Overall, enterprise anti-spam filtering is vital for cybersecurity, safeguarding digital assets and maintaining productivity.

d) Case-Based Spam Filtering: One of the well-known and conventional machine learning methods for spam detection is the case-based or sample-based spam filtering system. A typical case base filtering structure is illustrated in Figure 3.6. There are many phases to this type of filtering with the aid of the collection method; it collects data (mails) during the first step. After that, the major transition continues with the preprocessing steps through the client graphical user interface, outlining abstraction, and choice of email data classification, testing the entire process using vector expression and classifying the data into two classes: spam and legitimate email [52].

However, these traditional rule-based techniques have several limitations [48] [51] [49]:

— Spammers can easily change their email addresses or use techniques like phishing to bypass blacklists and whitelists.

— Rule-based methods require frequent updates and maintenance, which can be time-consuming and resource-intensive.

— They may mistakenly flag legitimate emails as spam, leading to a loss of important messages or business opportunities.

3.3.2 Review of Literature on the Application of Fuzzy Logic and KNN in Various Domains, Including Spam Email Detection

3.3.2.1 KNN in Spam Email Detection

The research project, "Email Spam Detection Using KNN Algorithm," by Dr. M. Senthil et al., introduces K-Nearest Neighbor (KNN) for spam detection, emphasizing the need for improved email filtering. It outlines the system architecture involving data preprocessing, feature extraction, and KNN implementation, followed by performance analysis. The study demonstrates KNN's superior performance over other algorithms like SVM and Naive Bayes in spam detection through an activity diagram. While recognizing KNN's effectiveness, the conclusion calls for additional techniques to address emerging spam types. Rigorous validation ensures accurate classification of spam and non-spam emails. Implemented in Python using relevant datasets, the model leverages KNN's principles for effective detection [50].

3.3.2.2 Fuzzy Logic in Spam Email Detection

The research, "Fuzzy Logic Approach for Email SPAM Detection System," by Damian Prihadi and Vivin Trisyanti, introduces fuzzy logic combined with association rules for spam detection. The system comprises email preprocessing, feature selection, and cluster construction with a Naive Bayes classifier. Testing on diverse datasets showcases the approach's high accuracy and effectiveness, nearing 1 in metrics like precision and recall. Overall, the study underscores the potential of integrating fuzzy logic and association rules in spam detection [46].

3.4 Proposed Algorithm

Existing methods, such as Fuzzy KNN, face limitations in detecting spam emails. For instance, using a single value of k for all messages can lead to similar membership degrees for both spam and non-spam categories. For example, the membership degrees could be No Spam: 0.47, Spam: 0.53. These membership degrees are close, so the classification may not be correct. This leads to less precise classification outcomes, undermining the overall effectiveness of the model. To address these limitations, I've developed an enhanced approach known as Improved Fuzzy KNN (IFKNN). This method identifies messages with membership degrees close, indicating classification ambiguity. IFKNN subsequently retrains the model with varying k values for these ambiguous messages. This adaptive strategy refines the model based on message-specific characteristics, ensuring more accurate and dependable classification results. By dynamically adjusting k values as necessary, IFKNN can better grasp the intricacies of the data, thus enhancing the overall performance of spam email classification.

3.4.1 Choosing Distance Metrics in Improved Fuzzy KNN (IFKNN) for Spam Email Detection

In Improved Fuzzy KNN (IFKNN), the choice between Euclidean distance and cosine similarity is made during the initialization of the Improved Fuzzy KNN class. In the realm of spam email detection, Euclidean distance and cosine similarity emerge as preferred metrics for their adeptness in capturing diverse facets of textual data. Euclidean distance discerns variations in word frequencies or distributions between spam and legitimate emails, despite its broad applicability. Conversely, cosine similarity excels in assessing the semantic similarity of documents, irrespective of their length or overall frequency. These metrics are favored over alternatives like Jaccard or Manhattan distances owing to their simplicity, interpretability, and established efficacy in text classification tasks. Their extensive validation and widespread adoption render them dependable choices for identifying spam emails based on distinct characteristics and similarities to known spam messages.

3.4.2 Organizational chart

This Organizational chart describes a spam detection process using two IFKNN (Improved Fuzzy K-Nearest Neighbors) classification models. Here is a brief explanation of the different steps:

— **Dataset:** The process starts with the collection of data, including emails classified as spam and non-spam.

— **Preprocessing:** The raw data is preprocessed to extract relevant features necessary for analysis.

— **Extracted Features:** The extracted features are prepared for use in the classification models.

— **IFKNN Model 1:** The first IFKNN model uses these features to analyze a new email and assign membership degrees to either spam or non-spam.

— New email: A new email is evaluated by the IFKNN Model 1.

— **(Spam/ No Spam) membership Degrees:** Membership degrees (spam or non-spam) are assigned by the model IFKNN.

— **Membership Degree Difference $\leq$ 0.2:** If the difference between the membership degrees is less than or equal to 0.2, additional messages are extracted for further analysis.

— **Extracting Messages:** Relevant messages are extracted for a second evaluation.

— **IFKNN Model 2:** A second IFKNN model, utilizing a new K value, is employed for a more precise analysis of the extracted messages.

— **Result + Evaluation:** The final results are generated and evaluated, providing a definitive classification of the new email as spam or non-spam.

This process as depicted in Figure 3.7 combines two classification models to improve the accuracy of spam detection by using fuzzy membership degrees and successive analyses.

3.4.3 Explanation of Improved Fuzzy KNN(IFKNN) Algorithm for Spam Email Detection

We propose an improved fuzzy KNN to address the limitations of the traditional Fuzzy KNN and KNN in the domain of spam email detection. The Fuzzy KNN

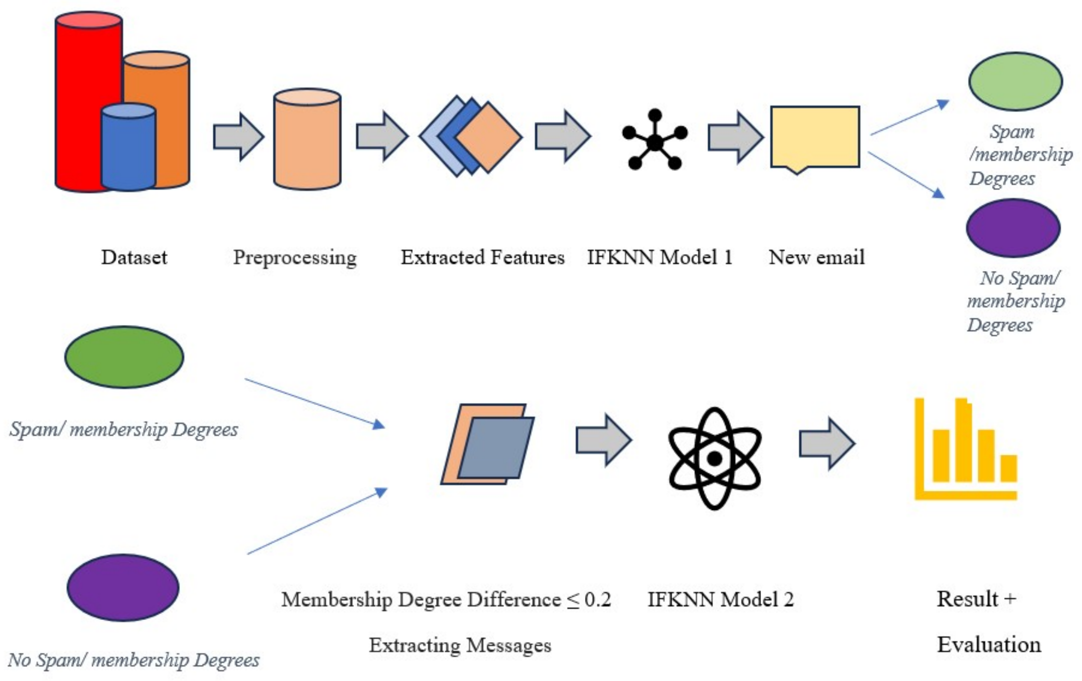


FIGURE 3.7 – Organizational Chart of IFKNN Model.

algorithm classifies messages as spam or non-spam with degrees of membership. However, there are instances where the membership degrees (Spam/Not-Spam) are close, for example, (Not-Spam: 0.45, Spam: 0.50). This results in uncertainty in the classification of messages. IFKNN resolves this issue by focusing on messages with close membership degrees, providing more precise and reliable classifications.

Here's a detailed description of how the Improved Fuzzy KNN (IFKNN) algorithm is applied and modified specifically for this purpose:

3.4.3.1 Preprocessing

Email text data undergoes preprocessing to ensure uniformity and eliminate noise. This involves steps like converting text to lowercase, removing non-alphanumeric characters, eliminating stop words, tokenizing, and stemming. we will provide a detailed explanation of the pre-processing steps in the following title: Preprocessing Steps for Email Data.

3.4.3.2 Feature Extraction Using TF-IDF

In the Feature Extraction method using TF-IDF (Term Frequency-Inverse Document Frequency), textual data is converted into numerical feature vectors. This transformation is performed by assigning each word a weight that captures its relevance in the context of the entire corpus. The TF-IDF is calculated by considering both the frequency of a term's occurrence in a specific document (TF) and its relative importance across the entire set of documents (IDF). Thus, words that are frequent in a specific document but uncommon in the entire corpus will receive a high weight, while words common across the corpus will be weighted lower. This technique allows textual features to be effectively represented as numerical vectors, thereby facilitating their use in text classification algorithms.

3.4.3.3 Training the Improved Fuzzy KNN Classifier

The IFKNN classifier is initialized with a default or user-defined value of k, representing the number of nearest neighbors to consider. It's then trained on the preprocessed features of the training data, learning the relationships between feature vectors and their corresponding labels (spam or non-spam).

3.4.3.4 Testing New Emails

This step, Testing New Emails, is designed to determine the final classification (spam or non-spam) for a new email message. Here's how it Works:

— Firstly, the email text is preprocessed using the preprocess text function to clean and prepare it for analysis. Then, the preprocessed text is transformed into numerical feature vectors using the vectorizer object, we use the vectorizer object to transform the email texts into numerical vectors. The vectorizer is a tool that converts the words in the emails into a numerical representation that machine learning models can process. It considers word frequency and other features to create vectors that quantitatively capture the essence of the emails.

— Next, the IFKNN model predicts whether the email is spam or not based on the extracted features. This prediction is stored in the prediction variable.

— Afterwards, the membership degrees of the email text to the spam and non-spam classes are calculated using the IFKNN model. These membership degrees represent the degree of similarity between the email and examples of spam and non-spam in the training dataset.

— Based on the prediction result, a result is generated to indicate whether the email is classified as spam or non-spam. If the prediction is 1, indicating spam, if the predication is 0, indicating no spam the result string includes a message stating that the email is spam; otherwise, it states that the email is not spam.

— Additionally, the membership degrees are appended to the result string to provide further insight into the classification decision.

— Finally, the function returns the result containing the classification result (spam or non-spam) and the membership degrees.

3.4.3.5 Extracting Messages

This "extract message" function is designed to extract email messages and evaluate whether they are spam or not using an IFKNN model. Here's how the function works:

— **Text Transformation into Feature Vectors:** The email text is preprocessed (using a "preprocess text" function) to clean it and prepare it for analysis. Subsequently, the preprocessed text is transformed into numerical feature vectors using a vectorizer (likely a TF-IDF Vectorizer).

— **Prediction:** Once the email text is transformed into a feature vector, the IFKNN model is used to predict whether the email is spam or not.

— **Calculation of Membership Degrees:** The membership degrees of the email text to the spam and non-spam classes are calculated using the IFKNN model. These membership degrees indicate how similar the email is to spam and non-spam examples in the training dataset.

— **Evaluation of Membership Degree Difference:** The difference between the spam and non-spam membership degrees is calculated. If this difference is less than or equal to 0.2, it indicates that the model is uncertain about the email's categorization. In this case, the message is considered "extracted".

We have two options for extracting messages:
Manually choose the value of *k*: Reclassification with a Different IFKNN Model: If the model is uncertain about the email's categorization, another IFKNN model is applied with different parameters for Example (k=18, distance metric='cosine'). The email is then reclassified as spam or non-spam, and the new membership degrees are calculated and displayed.
Automatically choose the value of *k*: The function to find the best value of k for use with the extracted messages works by selecting the k that maximizes the difference between the membership degrees for spam and non-spam. By iterating through these values, the function identifies the optimal k, ensuring the most accurate classification by increasing the distinction between the membership degrees for spam and non-spam emails. Once the best k is found, it is used to initialize the IFKNN classifier, significantly enhancing its ability to accurately classify the extracted messages.
Display of Results: The evaluation results, including the email text, the initial model prediction, the membership degrees, and, if applicable, the new prediction after reevaluation with another model, are formatted and displayed. This function thus enables spam detection by evaluating the membership degrees of emails to the spam and non-spam classes, while taking into account the uncertainty of the initial prediction. It provides a robust approach to filter out unwanted emails and ensure accurate classification.

3.4.3.6 Adjusting Fuzzy KNN Parameters

For the extracted messages, the classifier IFKNN is retrained with a different value of k, we choose a large value of k (k=100) for all messages tested. If a message requires extraction, we minimize the value of k to ensure accurate results and degrees of membership that are not close. This approach fine-tunes the classifier to adapt to the specific characteristics of each message, enhancing the precision of spam detection. By adjusting the Improved Fuzzy KNN algorithm to consider degrees of membership that are not closely aligned, this enhancement ensures more accurate and precise results. This guarantees that the classification of messages is correct, improving the overall effectiveness of the algorithm in distinguishing between spam and non-spam emails. Explaining the effect of changing the value of k on the general behavior of IFKNN and on the distance metric used helps better understand how adjustments impact the algorithm's ability to effectively discriminate between types of messages. Specifically, altering the value of k influences the level of granularity in the classification process. A smaller k value results in a more localized decision boundary, potentially leading to overfitting, while a larger k value might lead to over smoothing and loss of important details in the data. Additionally, the choice of distance metric, such as Euclidean or Cosine Similarity, can also influence the performance of the algorithm, as it determines how similarity between data points is measured.

3.4.3.7 Evaluation

The performance of the IFKNN classifier is evaluated using a separate test dataset, with metrics such as accuracy calculated to assess the model's effectiveness in distinguishing between spam and non-spam emails. By iteratively refining the IFKNN model and adapting its parameters based on the characteristics of ambiguous messages, the proposed algorithm aims to enhance the accuracy and reliability of email spam detection. In the subsequent chapter, we provide a comprehensive explanation of the evaluation of my IFKNN and analyze the results of various evaluation metrics.
Example Testing Message with IFKNN:

— Model IFKNN (K = 100, Distance metric='Euclidean')

— Testing New message: "Win an iPhone X by entering our giveaway!": Ce message "spam"
Membership Degree: No Spam: 0.47, Spam: 0.53
Membership Degree Difference = 0.06 $\leq$ 0.20 $\rightarrow$ Extracting message

— Model IFKNN2 (K Automatically or k manual = 18)
Automatically choose the value of k: The result of Improved FuzzyKNN (IFKNN) with k=28 and euclidean distance:
This email is: Spam
New membership degrees: 0: 0.25, 1: 0.75

Manually choose the value of k: The result of Improved FuzzyKNN (IFKNN) with k=18 and euclidean distance:
This email is: Win an iPhone X by entering our giveaway!: "Spam"
New membership degrees: 0: 0.05263157894736842, 1: 0.9473684210526315

3.4.4 Preprocessing Steps for Email Data

Email data preprocessing plays a crucial role in readying textual content for analytical endeavors. This involves a series of essential preprocessing stages aimed at ensuring the integrity and relevance of data for subsequent algorithmic processes.

In this section, we present Preprocessing Steps for Email Data:

— **Lowercasing and Stripping:** Initially, text undergoes lowercase conversion for casing uniformity, accompanied by the removal of any extraneous leading or trailing whitespace, ensuring consistency [53].

— **Removing Non-Alphanumeric Characters:** Following this, non-alphanumeric characters are purged from the text, leaving behind only letters, numbers, and spaces. This streamlines the text while retaining its pivotal content [54].

— **Stop word Elimination:** Through the use of a predefined set of English stop words, ubiquitous words lacking significant contextual meaning, such as "the" and "is," are sifted out from the tokenized text. This step enhances focus on content-laden words for subsequent analysis [53].

— **Tokenization:** Subsequently, the text undergoes tokenization, fragmenting it into discrete words to facilitate further processing. This segmentation ensures the text is digestible for subsequent analysis [53].

— **Stemming with Porter Algorithm:** Leveraging the Porter stemming algorithm, words are condensed to their root form, diminishing feature space and encapsulating core word meanings. This optimization heightens the efficiency of subsequent analytical processes [55].

— **Reassembly:** Finally, the processed words are reassembled into a unified string, meticulously formatted and purged of extraneous noise [54].

This refined output stands poised to serve as input data for the Fuzzy KNN algorithm. Such comprehensive preprocessing endeavors seek to standardize text data, attenuate noise, and elevate the quality of subsequent Fuzzy KNN analyses.

While these preprocessing endeavors undoubtedly enhance the suitability of email data for IFKNN analyses, it's imperative to recognize the necessity of experimentation and evaluation of alternative preprocessing methodologies. Such considerations should be tailored to the unique characteristics of the email dataset and the specific requisites of the algorithm at hand.

3.4.5 Selection of Distance Metrics and Parameter Tuning

Improved Fuzzy KNN (IFKNN) can be a powerful tool for spam email detection, but its effectiveness hinges on two crucial aspects:

— Distance Metric Selection: Choosing the right way to measure similarity between emails.

— Parameter Tuning: Optimizing the parameters of the IFKNN algorithm for your specific dataset.

Here's a breakdown of these two areas:

3.4.5.1 Distance Metrics

In traditional KNN, distance metrics like Euclidean or Manhattan measure the raw distance between data points. Fuzzy KNN incorporates fuzziness, allowing for partial membership in classes (spam/not spam). Here are some distance metrics suitable for IFKNN in spam detection [57]:

— **a) Euclidean Distance with Fuzzy Weighting:** This incorporates a weight based on the degree of membership in a class, emphasizing closer neighbors with higher spam likelihood [58].

— **b) Cosine Similarity:** This measures the directional similarity between email features, effective when the order of features is less important than their co-occurrence [60].

3.4.5.2 Parameter Tuning

FKNN introduces additional parameters compared to traditional KNN:

1. Fuzzifier: This controls the fuzziness level, with higher values allowing for more flexibility in class membership [61].

2. 'k' (number of neighbors): This determines the number of nearest neighbors considered for classification [62].

Here are some strategies for tuning these parameters:

— **Grid Search or Random Search:** Evaluate FKNN performance with different combinations of fuzzifier and 'k' values. Choose the combination yielding the best spam detection metrics (accuracy, precision, recall) [63].

— **Cross-Validation:** Divide your data into training and validation sets. Train models with various parameters on the training set and evaluate them on the validation set. Select the parameters that generalize best to unseen data [64].

3.5 Conclusion

This chapter explores enhancing spam email detection through the Fuzzy KNN algorithm. It outlines objectives and delves into theoretical backgrounds of KNN and Fuzzy KNN, emphasizing their importance in various domains. The proposed algorithm is systematically detailed, with emphasis on preprocessing steps, distance metric selection, and parameter tuning for optimization. Overall, the chapter provides a comprehensive framework for leveraging Fuzzy KNN in spam detection to improve accuracy and efficiency. The next chapter will implement the refined IFKNN algorithm, analyzing its performance on real-world datasets to assess its impact on cybersecurity and spam detection systems.

Chapter 4

Implementation

4.1 Introduction

In this chapter, we delve into the implementation of the Improved Fuzzy K Nearest Neighbors (IFKNN) algorithm for spam email detection. We begin by addressing the pervasive issue of spam emails and the various risks they pose in today's digital landscape. Recognizing the significance of machine learning in combating this problem, we introduce the IFKNN algorithm as a promising approach to enhancing classification accuracy and robustness. The chapter proceeds to provide a step-by-step guide to implementing IFKNN, starting with the preprocessing of raw email data and culminating in the development of a user-friendly interface using Tkinter. Key components such as text preprocessing, feature extraction using TF-IDF, and the IFKNN classifier are thoroughly explained, highlighting their roles in accurately categorizing and extracting spam emails. Finaly, we conduct a comprehensive evaluation of the Improved Fuzzy KNN algorithm, comparing its performance with other well-known algorithms such as K Nearest Neighbors (KNN) and Support Vector Machines (SVM). Through rigorous experimentation and analysis, we aim to demonstrate the effectiveness and superiority of IFKNN in spam email detection tasks.

4.2 Developpement environnement

4.2.1 Hardware

We utilized a computer with the following specifications for our project:

— RAM: 16 GB

— Processor: AMD Ryzen 7 5800H with Radeon Graphics.

— GPU: NVIDIA GeForce GTX 1650 4 GB

— These hardware components were employed to facilitate smooth execution and efficient processing of our application.

4.2.2 Software

4.2.2.1 Visual Studio Code Software

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET) [67].

4.2.2.2 Programming Language Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [68].

4.3 Libraries Used

4.3.1 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selection, I/O (Input/Output), discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, etc. [69].

4.3.2 Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way towards this goal [70].

4.3.3 Tkinter

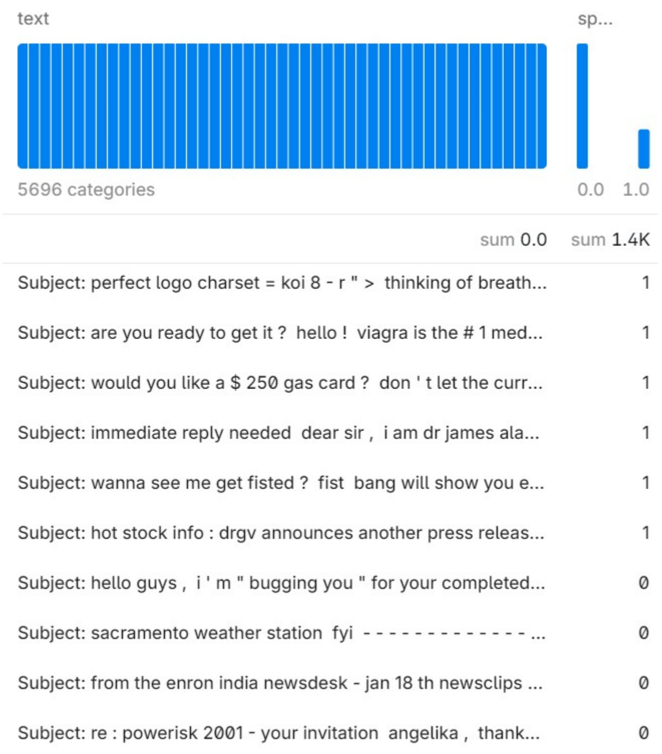
Tkinter is a standard Python library for creating graphical user interfaces (GUIs) for desktop applications. Tkinter is based on the Tk toolkit, which is a set of tools for creating graphical interfaces [71].

4.3.4 Datetime

The datetime module in Python provides classes for handling dates and times. While date and time calculations are supported, the primary focus during implementation has been on efficiently extracting attributes for manipulation and formatting for display purposes. I have utilized this library to display the date and time of messages classified as spam in the spam box [72].

4.3.5 PIL

Pillow is an image processing library, which is a fork and successor of the PIL (Python Imaging Library) project. It's designed to provide fast access to image data and supports various file formats including PPM, PNG, JPEG, GIF, TIFF, and BMP [73].



Subject: perfect logo charset = koi 8 - r " > thinking of breath...	1
Subject: are you ready to get it ? hello ! viagra is the # 1 med...	1
Subject: would you like a $ 250 gas card ? don ' t let the curr...	1
Subject: immediate reply needed dear sir , i am dr james ala...	1
Subject: wanna see me get fisted ? fist bang will show you e...	1
Subject: hot stock info : drgv announces another press releas...	1
Subject: hello guys , i ' m " bugging you " for your completed...	0
Subject: sacramento weather station fyi - - - - - - - - - - - - - ...	0
Subject: from the enron india newsdesk - jan 18 th newsclips ...	0
Subject: re : powerisk 2001 - your invitation angelika , thank...	0

FIGURE 4.1 – Spam Email Dataset [76].

4.3.6 Scikit-learn (sklearn)

Scikit-learn, often abbreviated as sklearn, is an open-source Python library that provides a wide range of tools for machine learning and statistical modeling. It is built on top of other popular Python libraries such as NumPy, SciPy, and matplotlib, and is designed to be simple and efficient, making it accessible to both beginners and experienced practitioners [74].

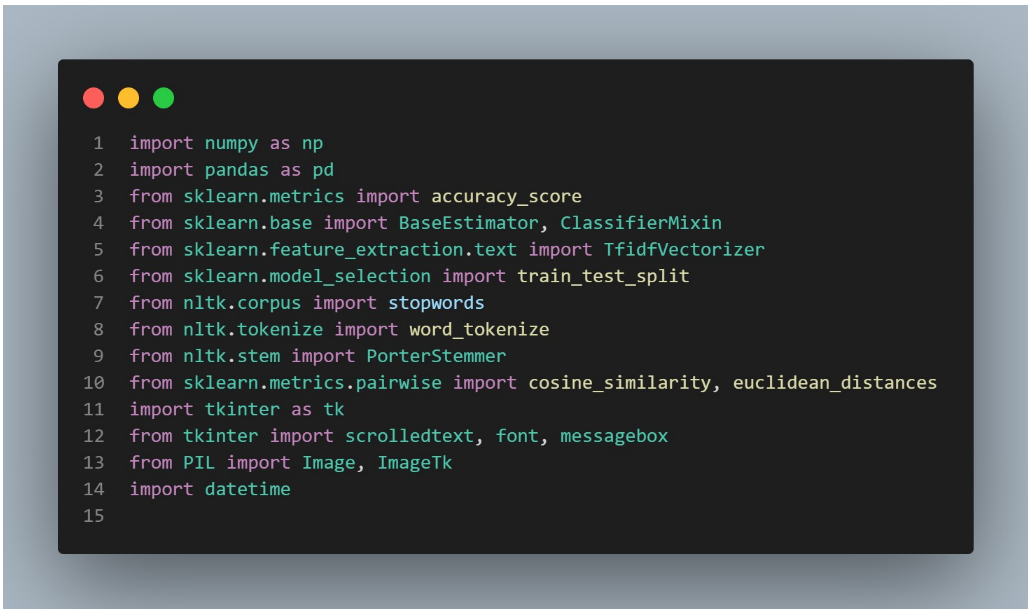
4.3.7 The Natural Language Toolkit (nltk)

The Natural Language Toolkit (NLTK) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning [75].

4.4 System overview

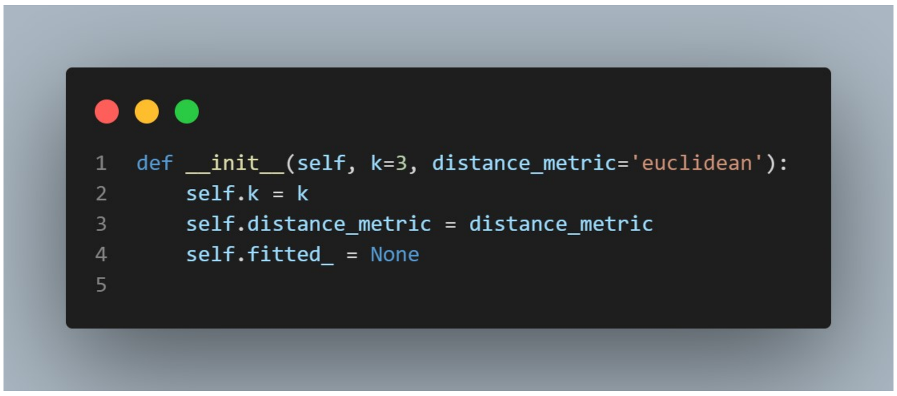
4.4.1 Dataset description

We have used dataset called "Spam Email Dataset" We can find it here: [Spam email Dataset (kaggle.com)].
Description: This dataset shown in Figure 4.1 contains a collection of email text messages, labeled as either spam or not spam. Each email message is associated with a binary label, where "1" indicates that the email is spam, and "0" indicates that it is not spam. The dataset is intended for use in training and evaluating spam email classification models [76].



```python
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.base import BaseEstimator, ClassifierMixin
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances
import tkinter as tk
from tkinter import scrolledtext, font, messagebox
from PIL import Image, ImageTk
import datetime

```

FIGURE 4.2 – Importing Libraries.



```python
def __init__(self, k=3, distance_metric='euclidean'):
    self.k = k
    self.distance_metric = distance_metric
    self.fitted_ = None

```

FIGURE 4.3 – Initialization Method for Improved Fuzzy KNN Class [77].

Now we will present the main functions of our system:

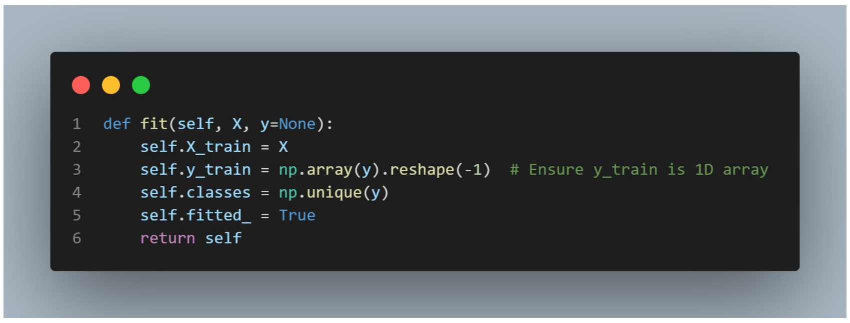
4.4.2 Importing libraries

Now we import the libraries that will be used in our code. Figure 4.2 illustrates this operation.

4.4.3 Improved Fuzzy KNN(IFKNN) model

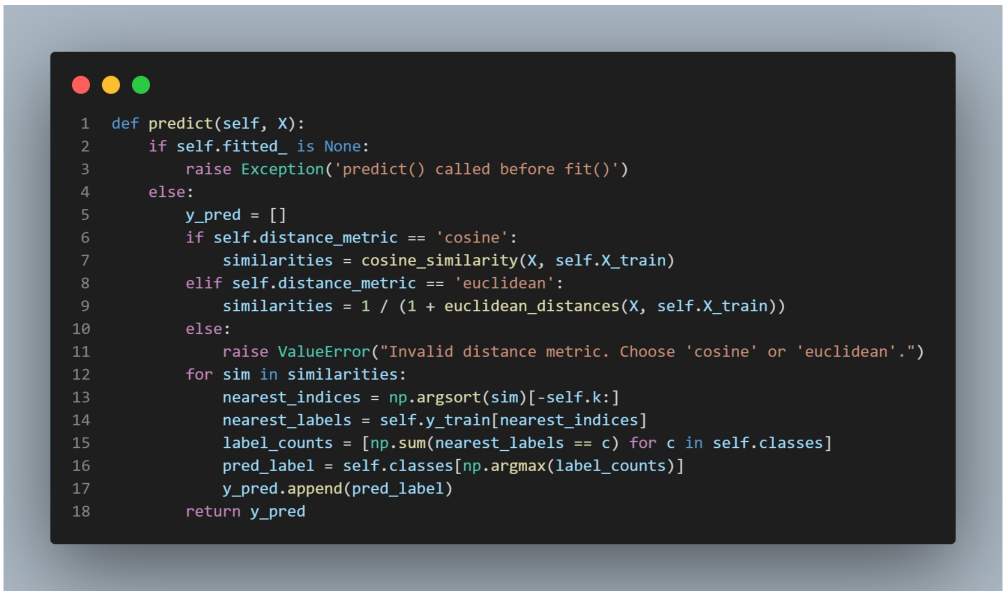
a) Initialization Method for Improved Fuzzy KNN Class: Figure 4.3 shows a snippet of Python code that defines the "init" method of a class. This method initializes the class with three attributes: k, distance metric, and fitted.

b) Training Method for Improved Fuzzy KNN Class: Figure 4.4 shows a code Python that defines a method named "fit" for a class. This method is typically used to train or fit the model with given training data.



```python
def fit(self, X, y=None):
    self.X_train = X
    self.y_train = np.array(y).reshape(-1)  # Ensure y_train is 1D array
    self.classes = np.unique(y)
    self.fitted_ = True
    return self
```

FIGURE 4.4 – Training Method for IFKNN Class [77].



```python
def predict(self, X):
    if self.fitted_ is None:
        raise Exception('predict() called before fit()')
    else:
        y_pred = []
        if self.distance_metric == 'cosine':
            similarities = cosine_similarity(X, self.X_train)
        elif self.distance_metric == 'euclidean':
            similarities = 1 / (1 + euclidean_distances(X, self.X_train))
        else:
            raise ValueError("Invalid distance metric. Choose 'cosine' or 'euclidean'.")
        for sim in similarities:
            nearest_indices = np.argsort(sim)[-self.k:]
            nearest_labels = self.y_train[nearest_indices]
            label_counts = [np.sum(nearest_labels == c) for c in self.classes]
            pred_label = self.classes[np.argmax(label_counts)]
            y_pred.append(pred_label)
        return y_pred
```

FIGURE 4.5 – Prediction Method for IFKNN Class [77].

c) Prediction Method for IFKNN Class: Figure 4.5 shows a code defines a predict method, designed to make predictions based on the training data and specified distance metric ('Cosine' or 'Euclidean').

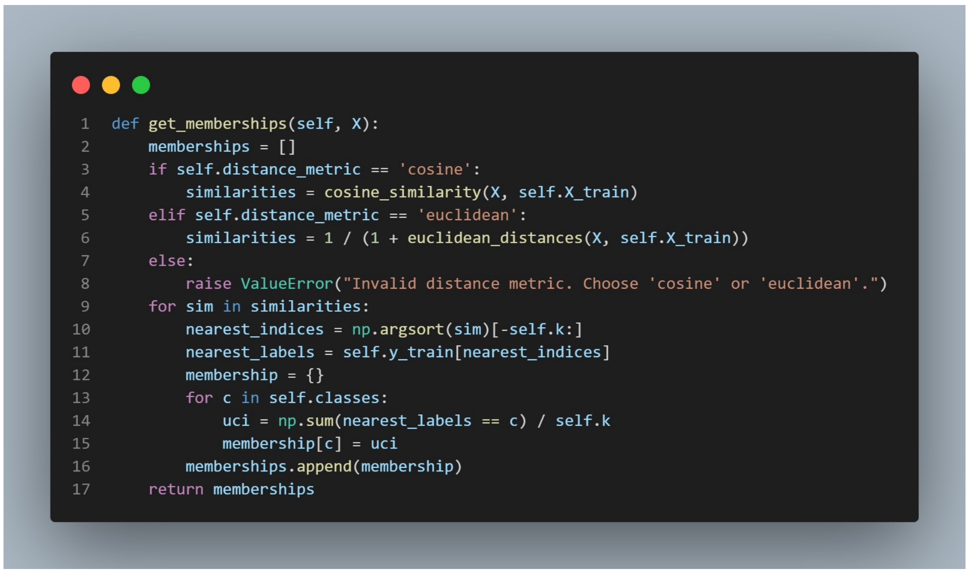
d) Membership Calculation Method for IFKNN Class Figure 4.6 shows a code defines a method get memberships, which calculates the membership proportions of each class for the input data based on the nearest neighbors.

4.4.4 Reading Dataset

The next figure (4.7) shows how to read Data set.

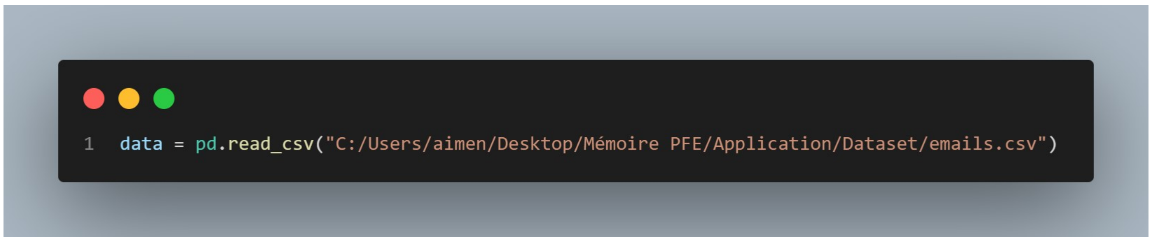
4.4.5 Preprocessing dataset

Figure 4.8 shows how to preprocess the textual data. The preprocessing steps include converting text to lowercase, removing non-alphanumeric characters, tokenizing the text into words, removing stop words, stemming, and finally rejoining the processed words into a string.



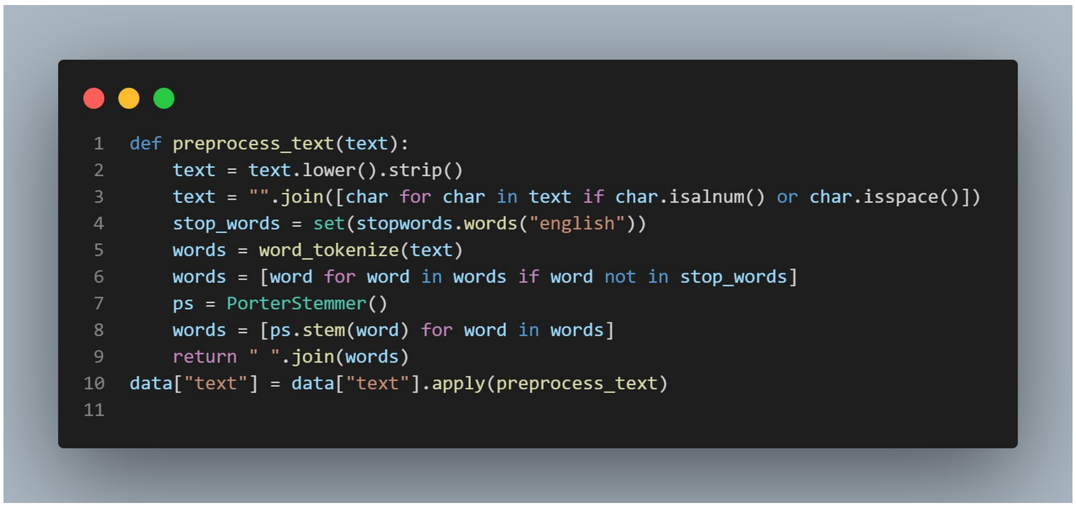
```python
def get_memberships(self, X):
    memberships = []
    if self.distance_metric == 'cosine':
        similarities = cosine_similarity(X, self.X_train)
    elif self.distance_metric == 'euclidean':
        similarities = 1 / (1 + euclidean_distances(X, self.X_train))
    else:
        raise ValueError("Invalid distance metric. Choose 'cosine' or 'euclidean'.")
    for sim in similarities:
        nearest_indices = np.argsort(sim)[-self.k:]
        nearest_labels = self.y_train[nearest_indices]
        membership = {}
        for c in self.classes:
            uci = np.sum(nearest_labels == c) / self.k
            membership[c] = uci
        memberships.append(membership)
    return memberships
```

FIGURE 4.6 – Membership Calculation Method for IFKNN Class [77].



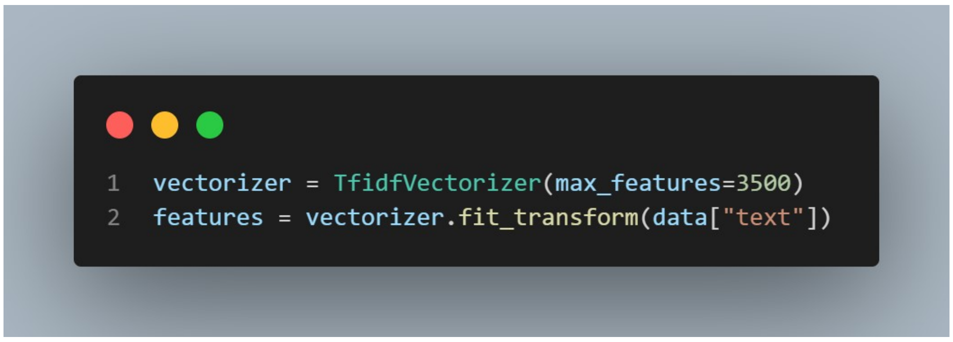
```python
data = pd.read_csv("C:/Users/aimen/Desktop/Mémoire PFE/Application/Dataset/emails.csv")
```

FIGURE 4.7 – Reading Dataset.



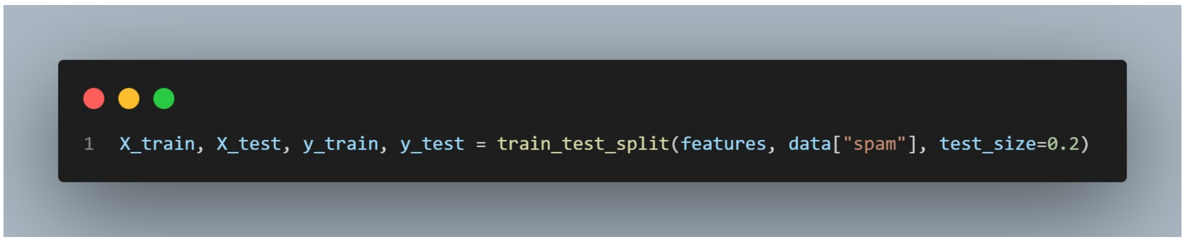
```python
def preprocess_text(text):
    text = text.lower().strip()
    text = "".join([char for char in text if char.isalnum() or char.isspace()])
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    ps = PorterStemmer()
    words = [ps.stem(word) for word in words]
    return " ".join(words)
data["text"] = data["text"].apply(preprocess_text)

```

FIGURE 4.8 – Preprocessing Dataset.



```python
vectorizer = TfidfVectorizer(max_features=3500)
features = vectorizer.fit_transform(data["text"])
```

FIGURE 4.9 – Text Vectorization.



```python
X_train, X_test, y_train, y_test = train_test_split(features, data["spam"], test_size=0.2)
```

FIGURE 4.10 – Train-Test Split.

4.4.6 Text Vectorization using TF-IDF Vectorizer

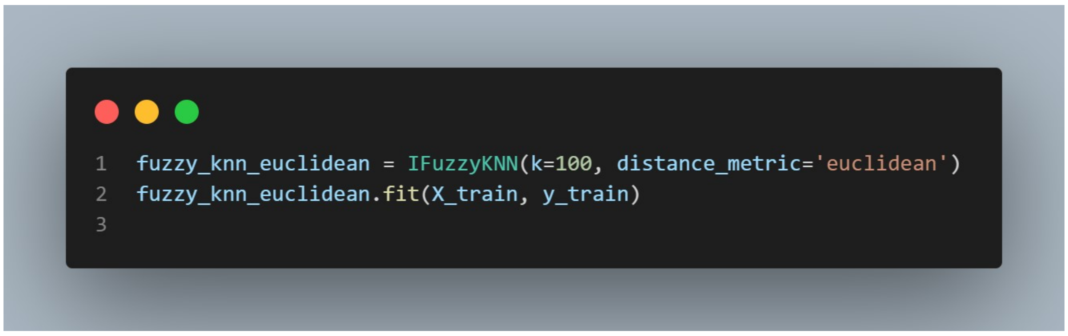
The code illustrated by Figure 4.9 show how to perform text vectorization using the TF-IDF vectorizer in Python. It initializes the vectorizer with a maximum of 3500 features and then applies it to the textual data in the "text" column of a Data Frame.

4.4.7 Train-Test Split

Figure 4.10 shows how to split the dataset into training and testing sets for supervised learning tasks, specifically for text classification.

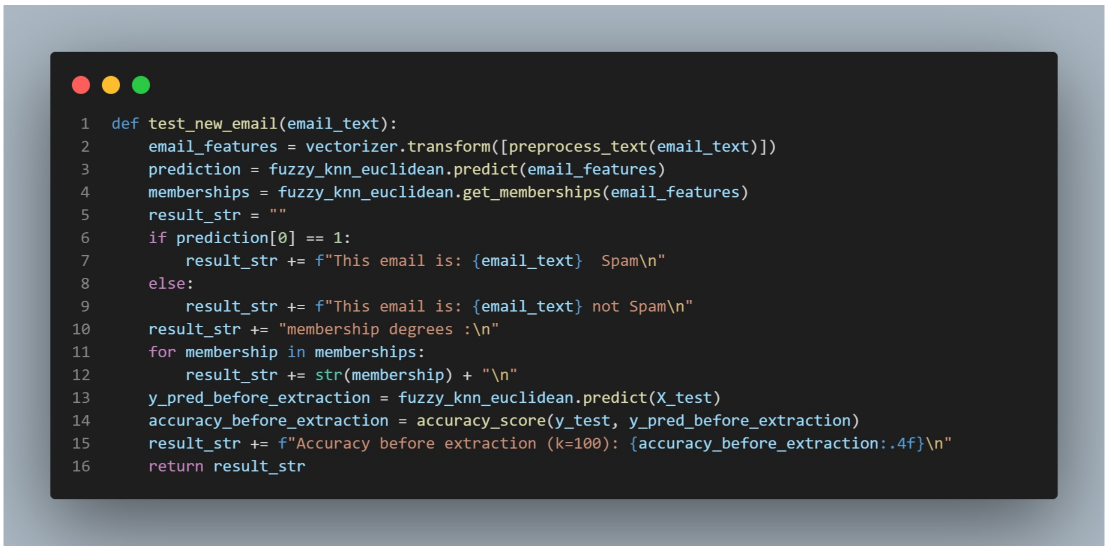
4.4.8 Initialization and Training of IFKNN Classifier with Euclidean Distance Metric

The initialization and training of IFKNN classifier is shown in Figure 4.11, setting the number of neighbors (*k*) to 100 and specifying the distance metric as 'Euclidean'.



```python
fuzzy_knn_euclidean = IFuzzyKNN(k=100, distance_metric='euclidean')
fuzzy_knn_euclidean.fit(X_train, y_train)

```

FIGURE 4.11 – Initialization and Training of IFKNN Classifier.



```python
def test_new_email(email_text):
    email_features = vectorizer.transform([preprocess_text(email_text)])
    prediction = fuzzy_knn_euclidean.predict(email_features)
    memberships = fuzzy_knn_euclidean.get_memberships(email_features)
    result_str = ""
    if prediction[0] == 1:
        result_str += f"This email is: {email_text}  Spam\n"
    else:
        result_str += f"This email is: {email_text} not Spam\n"
    result_str += "membership degrees :\n"
    for membership in memberships:
        result_str += str(membership) + "\n"
    y_pred_before_extraction = fuzzy_knn_euclidean.predict(X_test)
    accuracy_before_extraction = accuracy_score(y_test, y_pred_before_extraction)
    result_str += f"Accuracy before extraction (k=100): {accuracy_before_extraction:.4f}\n"
    return result_str
```

FIGURE 4.12 – Testing New Email and Evaluating IFKNN Performance.

4.4.9 Test New Email and Evaluate IFKNN Performance

The operation of testing method for new messages after vectorization and pre-processing of the data, with an evaluation of the IFKNN model, is illustrated by Figure 4.12.

4.4.10 Automatically choose the value of k

The find_best_k function (Figure 4.13) iterates over possible k values from 1 to 30, training the IFKNN classifier with each value and evaluating its performance. The function measures the difference between spam and non-spam membership values, selecting the k that minimizes this difference. By iterating through these values, the function identifies the optimal k.

4.4.11 Message Extraction and Evaluation IFKNN Performance

Figure 4.14 illustrates how to extract messages that have already been tested using the previous method, and the results of the membership degrees (Spam and Non-Spam) are close. Then, the message is extracted, and IFKNN is applied with a new manual value of K. Finally, the performance of IFKNN is evaluated.

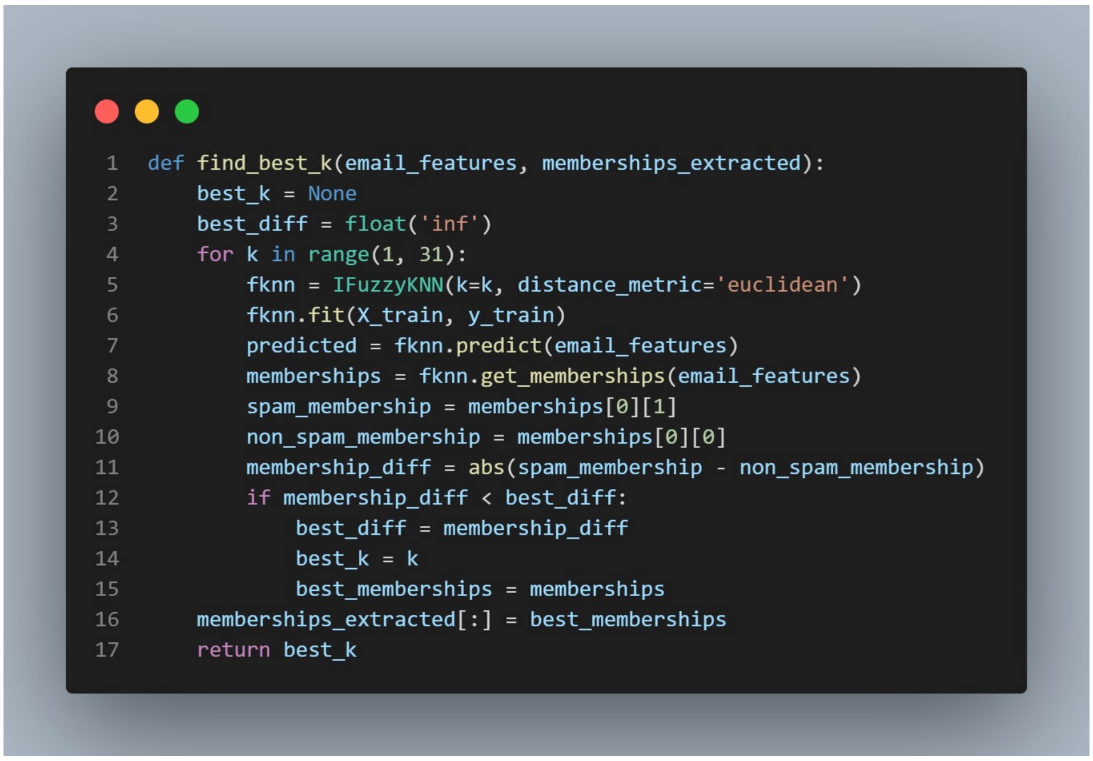
4.5 System interface

In this section, I will present several interfaces of my program. The first interface, "How to Use This Application", contains steps on how to use the application. Additionally, there are other interfaces for testing messages to determine if they are spam or not. Finally, there is a "Spam Box" interface, which contains spam messages along with their exportation date and time.

The main interface of our system is illustrated in Figure 4.15.

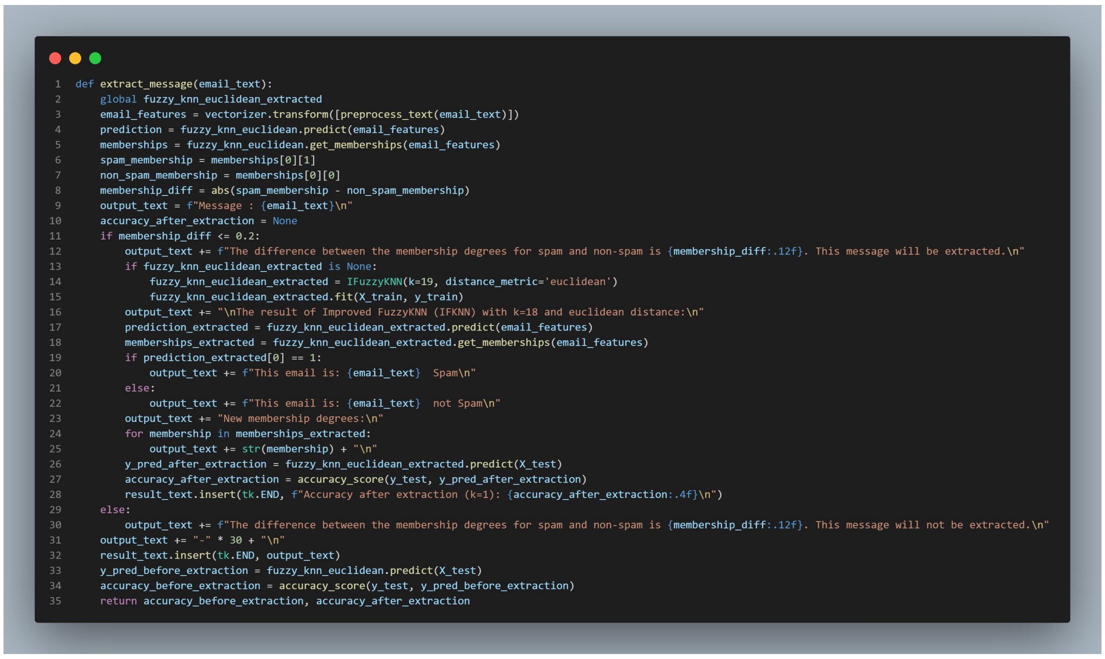
Description:

— Button "Test and Extract Email": When the user enters an email and clicks this button, display whether the message is spam or not in the result.



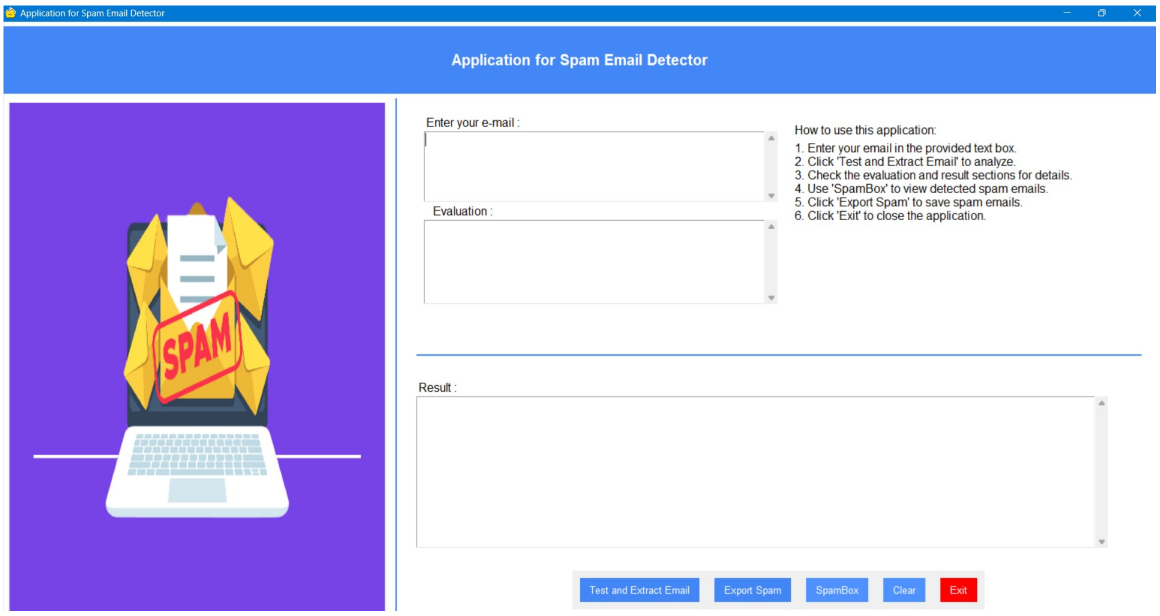
```python
def find_best_k(email_features, memberships_extracted):
    best_k = None
    best_diff = float('inf')
    for k in range(1, 31):
        fknn = IFuzzyKNN(k=k, distance_metric='euclidean')
        fknn.fit(X_train, y_train)
        predicted = fknn.predict(email_features)
        memberships = fknn.get_memberships(email_features)
        spam_membership = memberships[0][1]
        non_spam_membership = memberships[0][0]
        membership_diff = abs(spam_membership - non_spam_membership)
        if membership_diff < best_diff:
            best_diff = membership_diff
            best_k = k
            best_memberships = memberships
    memberships_extracted[:] = best_memberships
    return best_k
```

FIGURE 4.13 – Automatically Choosing the Value of *k*.



```python
def extract_message(email_text):
    global fuzzy_knn_euclidean_extracted
    email_features = vectorizer.transform([preprocess_text(email_text)])
    prediction = fuzzy_knn_euclidean.predict(email_features)
    memberships = fuzzy_knn_euclidean.get_memberships(email_features)
    spam_membership = memberships[0][1]
    non_spam_membership = memberships[0][0]
    membership_diff = abs(spam_membership - non_spam_membership)
    output_text = f"Message : {email_text}\n"
    accuracy_after_extraction = None
    if membership_diff <= 0.2:
        output_text += f"The difference between the membership degrees for spam and non-spam is {membership_diff:.12f}. This message will be extracted.\n"
        if fuzzy_knn_euclidean_extracted is None:
            fuzzy_knn_euclidean_extracted = IFuzzyKNN(k=19, distance_metric='euclidean')
            fuzzy_knn_euclidean_extracted.fit(X_train, y_train)
        output_text += "\nThe result of Improved FuzzyKNN (IFKNN) with k=18 and euclidean distance:\n"
        prediction_extracted = fuzzy_knn_euclidean_extracted.predict(email_features)
        memberships_extracted = fuzzy_knn_euclidean_extracted.get_memberships(email_features)
        if prediction_extracted[0] == 1:
            output_text += f"This email is: {email_text}  Spam\n"
        else:
            output_text += f"This email is: {email_text}  not Spam\n"
        output_text += "New membership degrees:\n"
        for membership in memberships_extracted:
            output_text += str(membership) + "\n"
        y_pred_after_extraction = fuzzy_knn_euclidean_extracted.predict(X_test)
        accuracy_after_extraction = accuracy_score(y_test, y_pred_after_extraction)
        result_text.insert(tk.END, f"Accuracy after extraction (k=1): {accuracy_after_extraction:.4f}\n")
    else:
        output_text += f"The difference between the membership degrees for spam and non-spam is {membership_diff:.12f}. This message will not be extracted.\n"
    output_text += "-" * 30 + "\n"
    result_text.insert(tk.END, output_text)
    y_pred_before_extraction = fuzzy_knn_euclidean.predict(X_test)
    accuracy_before_extraction = accuracy_score(y_test, y_pred_before_extraction)
    return accuracy_before_extraction, accuracy_after_extraction
```

FIGURE 4.14 – Message Extraction and Evaluation of IFKNN Performance.



How to use this application:
1. Enter your email in the provided text box.
2. Click 'Test and Extract Email' to analyze.
3. Check the evaluation and result sections for details.
4. Use 'SpamBox' to view detected spam emails.
5. Click 'Export Spam' to save spam emails.
6. Click 'Exit' to close the application.

FIGURE 4.15 – The Main Interface of Our System.

— Button "Export Spam": If the message is detected as spam and the user clicks on this button, then the message will be exported to the Spam Box file.

— Button "Spam Box": This button opens the Spam Box file, which stores spam messages exported with date and time. Figure 4.19 shows the Spam Box File.

— Button "Exit": This button closes or exits the application.

— Button "Clear": allows the user to clear the content of the text fields for email input, evaluation, and results when clicked.

Exception: If the user enters no message and clicks the 'Test and Extract Message' button, then display the message on the result: 'Please enter an email before testing.

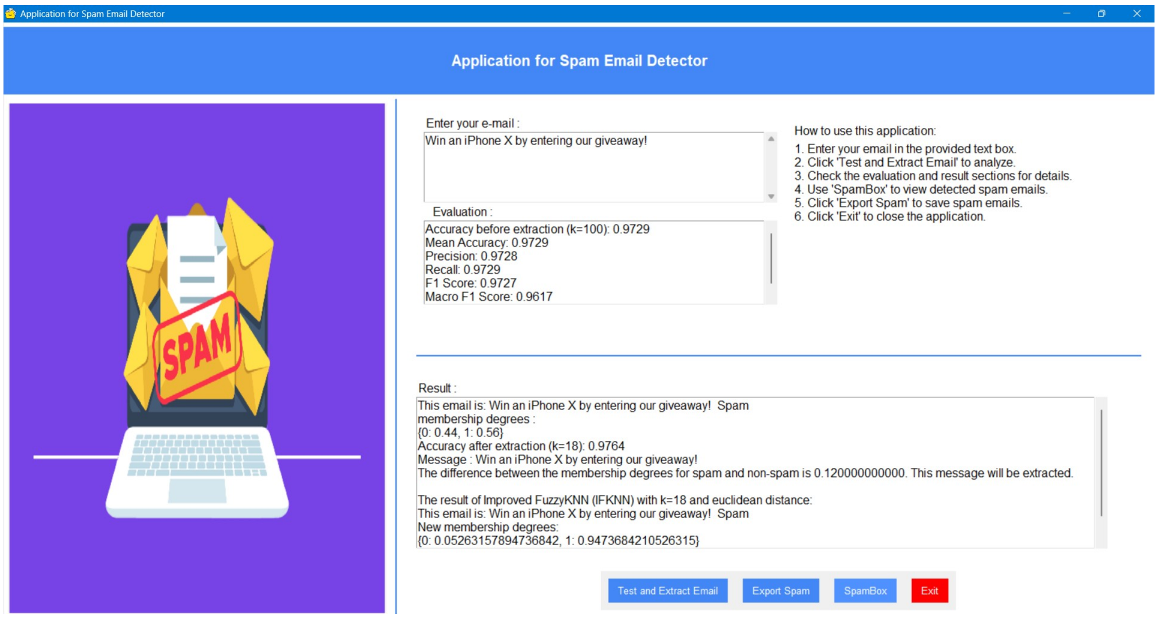
Example 1:

In Figure 4.16, I tested the message "Win an iPhone X by entering our giveaway!" When the "Test and Extract Email" button is clicked, the result shows that the message is classified as spam, and the message is extracted because the membership degrees are close. we applied IFKNN with a different value of k (k = 18), and the final result indicates that the message is spam with the membership degrees (Not Spam: 0.06, Spam: 0.94).

Example 2: In Figure 4.17, I tested the message "Reminder: Your appointment with Dr. Smith is tomorrow at 10 AM. Please confirm your attendance". When the "Test and Extract Email" button is clicked, the result shows that the message is classified as not spam with the membership degrees as shown in the figure.

Example 3: Figure 4.18 shows the operation of Testing with Automatically choose the value of k.

Spam Box: In Figure 4.19, the SpamBox interface opens when the "SpamBox" button is clicked. When a message is identified as spam and the "Export Spam" button is clicked, the spam message is exported to a .txt file along with the date and time of export. If the message is not spam and the "Export Spam" button is clicked, a message will be displayed stating, "This email is not spam and will not be exported".


Application for Spam Email Detector
Application for Spam Email Detector
SPAM
Test and Extract Email
Export Spam
SpamBox
Exit

Enter your e-mail :
Win an iPhone X by entering our giveaway!

Evaluation :

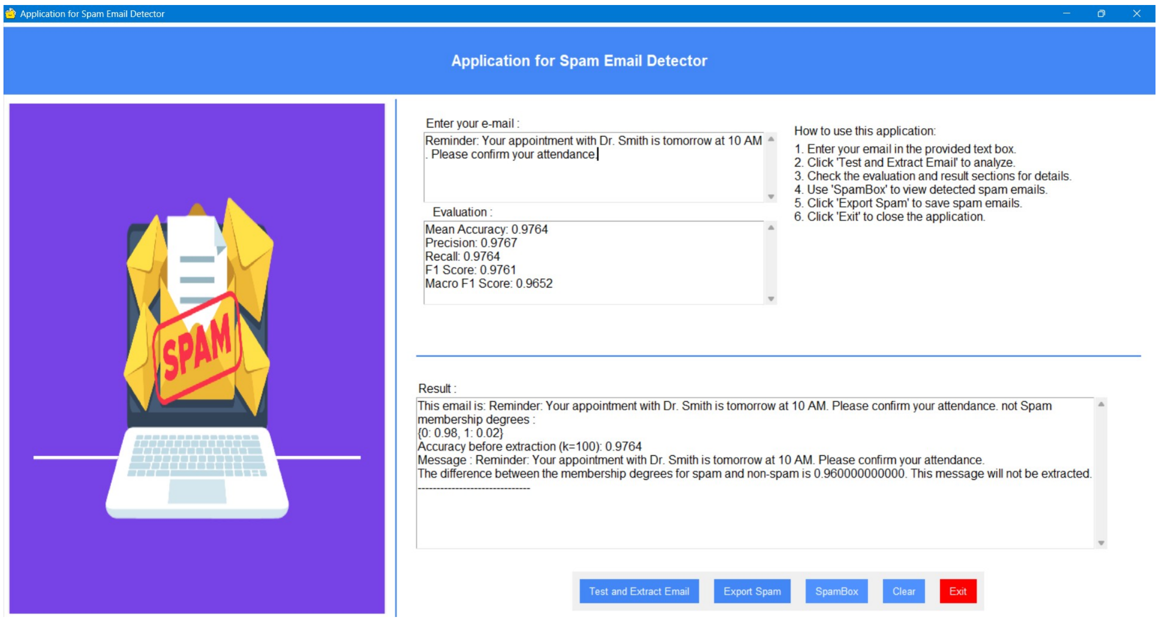
```
Accuracy before extraction (k=100): 0.9729
Mean Accuracy: 0.9729
Precision: 0.9728
Recall: 0.9729
F1 Score: 0.9727
Macro F1 Score: 0.9617
```

How to use this application:
1. Enter your email in the provided text box.
2. Click 'Test and Extract Email' to analyze.
3. Check the evaluation and result sections for details.
4. Use 'SpamBox' to view detected spam emails.
5. Click 'Export Spam' to save spam emails.
6. Click 'Exit' to close the application.

Result :

```
This email is: Win an iPhone X by entering our giveaway!  Spam
membership degrees :
{0: 0.44, 1: 0.56}
Accuracy after extraction (k=18): 0.9764
Message : Win an iPhone X by entering our giveaway!
The difference between the membership degrees for spam and non-spam is 0.120000000000. This message will be extracted.

The result of Improved FuzzyKNN (IFKNN) with k=18 and euclidean distance:
This email is: Win an iPhone X by entering our giveaway!  Spam
New membership degrees:
{0: 0.05263157894736842, 1: 0.9473684210526315}
```

FIGURE 4.16 – Example 1: Testing a Spam Message.


Application for Spam Email Detector
Application for Spam Email Detector
SPAM
Test and Extract Email
Export Spam
SpamBox
Clear
Exit

Enter your e-mail :
Reminder: Your appointment with Dr. Smith is tomorrow at 10 AM. Please confirm your attendance.

Evaluation :

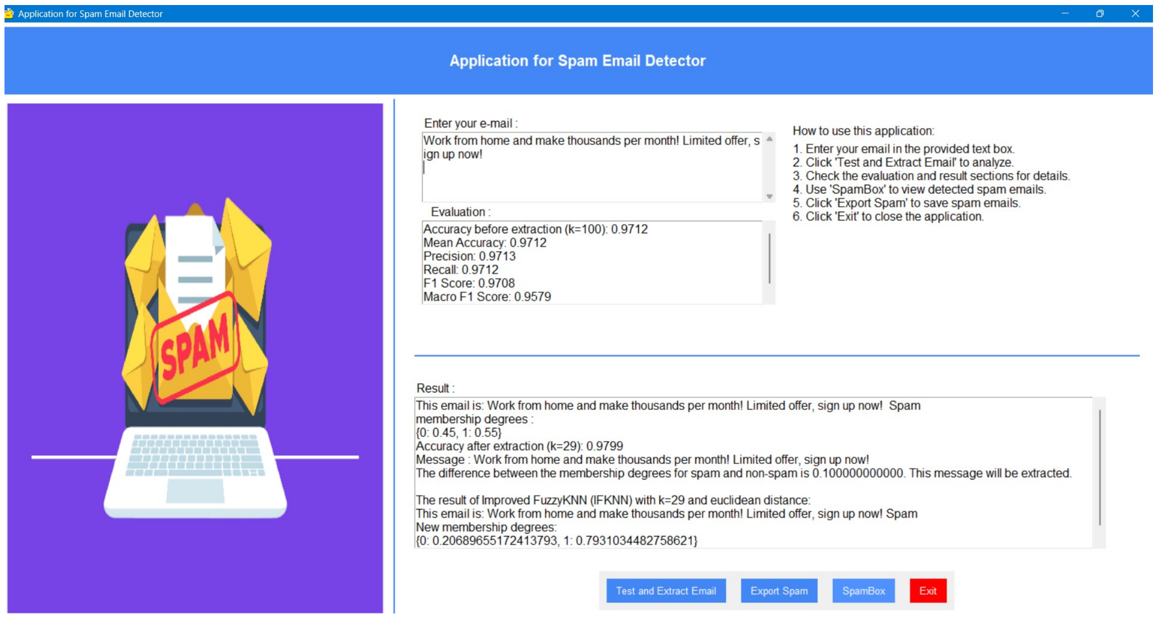
```
Mean Accuracy: 0.9764
Precision: 0.9767
Recall: 0.9764
F1 Score: 0.9761
Macro F1 Score: 0.9652
```

How to use this application:
1. Enter your email in the provided text box.
2. Click 'Test and Extract Email' to analyze.
3. Check the evaluation and result sections for details.
4. Use 'SpamBox' to view detected spam emails.
5. Click 'Export Spam' to save spam emails.
6. Click 'Exit' to close the application.

Result :

```
This email is: Reminder: Your appointment with Dr. Smith is tomorrow at 10 AM. Please confirm your attendance. not Spam
membership degrees :
{0: 0.98, 1: 0.02}
Accuracy before extraction (k=100): 0.9764
Message : Reminder: Your appointment with Dr. Smith is tomorrow at 10 AM. Please confirm your attendance.
The difference between the membership degrees for spam and non-spam is 0.960000000000. This message will not be extracted.
------------------------------
```

FIGURE 4.17 – Example 2: Testing a Non-Spam Message.


Application for Spam Email Detector
Application for Spam Email Detector
SPAM
Test and Extract Email
Export Spam
SpamBox
Exit

Enter your e-mail :

Work from home and make thousands per month! Limited offer, sign up now!

Evaluation :

```
Accuracy before extraction (k=100): 0.9712
Mean Accuracy: 0.9712
Precision: 0.9713
Recall: 0.9712
F1 Score: 0.9708
Macro F1 Score: 0.9579
```

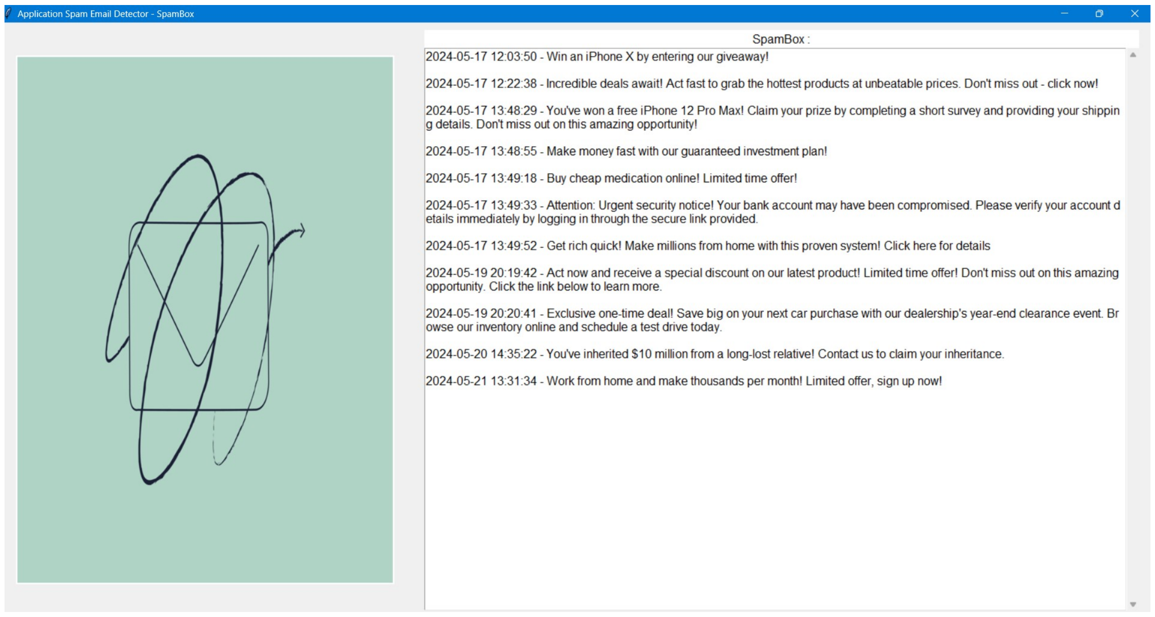
How to use this application:

1. Enter your email in the provided text box.
2. Click 'Test and Extract Email' to analyze.
3. Check the evaluation and result sections for details.
4. Use 'SpamBox' to view detected spam emails.
5. Click 'Export Spam' to save spam emails.
6. Click 'Exit' to close the application.

Result :

```
This email is: Work from home and make thousands per month! Limited offer, sign up now!  Spam
membership degrees :
{0: 0.45, 1: 0.55}
Accuracy after extraction (k=29): 0.9799
Message : Work from home and make thousands per month! Limited offer, sign up now!
The difference between the membership degrees for spam and non-spam is 0.100000000000. This message will be extracted.

The result of Improved FuzzyKNN (IFKNN) with k=29 and euclidean distance:
This email is: Work from home and make thousands per month! Limited offer, sign up now! Spam
New membership degrees:
{0: 0.20689655172413793, 1: 0.7931034482758621}
```

FIGURE 4.18 – Example 3: Testing with Automatically Chosen Value of k.



SpamBox :

```
2024-05-17 12:03:50 - Win an iPhone X by entering our giveaway!

2024-05-17 12:22:38 - Incredible deals await! Act fast to grab the hottest products at unbeatable prices. Don't miss out - click now!

2024-05-17 13:48:29 - You've won a free iPhone 12 Pro Max! Claim your prize by completing a short survey and providing your shipping details. Don't miss out on this amazing opportunity!

2024-05-17 13:48:55 - Make money fast with our guaranteed investment plan!

2024-05-17 13:49:18 - Buy cheap medication online! Limited time offer!

2024-05-17 13:49:33 - Attention: Urgent security notice! Your bank account may have been compromised. Please verify your account details immediately by logging in through the secure link provided.

2024-05-17 13:49:52 - Get rich quick! Make millions from home with this proven system! Click here for details

2024-05-19 20:19:42 - Act now and receive a special discount on our latest product! Limited time offer! Don't miss out on this amazing opportunity. Click the link below to learn more.

2024-05-19 20:20:41 - Exclusive one-time deal! Save big on your next car purchase with our dealership's year-end clearance event. Browse our inventory online and schedule a test drive today.

2024-05-20 14:35:22 - You've inherited $10 million from a long-lost relative! Contact us to claim your inheritance.

2024-05-21 13:31:34 - Work from home and make thousands per month! Limited offer, sign up now!
```

FIGURE 4.19 – Example of Spam Box window.

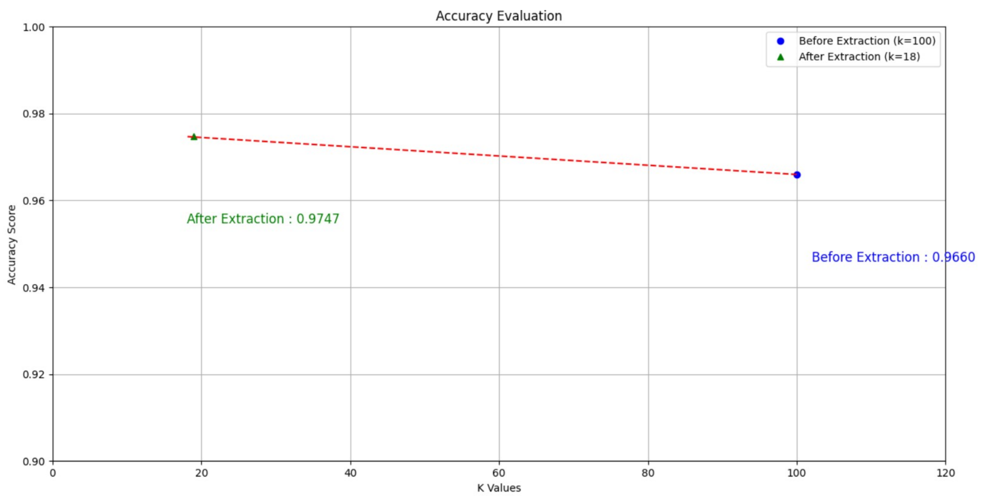


FIGURE 4.20 – Evaluation Before and After Extraction with Manual Value of *k*.

4.6 Evaluation

4.6.1 Evaluation metrics

In section, we present the evaluation of the IFKNN model when testing the same message with both manual and automatic values of k. In both cases, the message is extracted, and then we assess the IFKNN model before and after this extraction. We chose to utilize the extracted messages for this evaluation, as they represent the focal point of this study.

Example 1:

Figure 4.20 represents the accuracy evaluation of the IFKNN (Improved Fuzzy K-Nearest Neighbors) algorithm with Manual value of k. The graph shows the improvement in accuracy after feature extraction. Before extraction, with k=100, the accuracy before Extraction was 0.9660 (indicated by the blue dot). After extraction, with k=18, the accuracy improved to 0.9747 (indicated by the green triangle). This improvement suggests that extracted message helped optimize the algorithm, thereby enhancing its performance.

Example 2:

Figure 4.21 represents the accuracy evaluation of the IFKNN (Improved Fuzzy K-Nearest Neighbors) algorithm Manual value of k. The graph shows the improvement in accuracy after feature extraction. Before extraction, with k=100, the accuracy before Extraction was 0.9703 (indicated by the blue dot). After extraction, with auto value k, the accuracy improved to 0.9712 (indicated by the green triangle). This improvement suggests that extracted message helped optimize the algorithm, thereby enhancing its performance.

4.6.2 Comparison with other algorithms

In this section, we presented a comparison of the performance of IFKNN with other algorithms such as SVM, NB, and KNN. I applied the four algorithms (NB, SVM, KNN, and Improved Fuzzy KNN) on the same dataset, and the results are shown in Table 4.1.

Table 4.1 presents the evaluation metrics of four different algorithms—KNN, SVM, NB, and IFKNN—using accuracy, precision, recall, and F1 score as performance

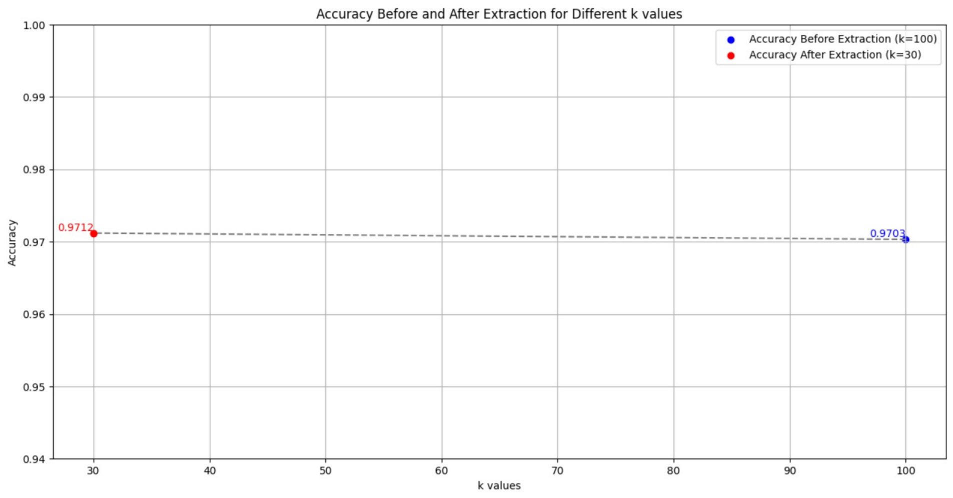


FIGURE 4.21 – Evaluation Before and After Extraction with Auto Value of k.

indicators. The IFKNN algorithm outperforms KNN across all evaluation metrics, showing higher precision (0.9781), recall (0.9782), and F1 score (0.9780). While SVM has the highest accuracy (0.9877), IFKNN exceeds both SVM and NB in terms of precision, recall, and F1 score, making it a superior choice in these metrics.

TABLE 4.1 – Comparison with Other Algorithms.

Algorithm	Accuracy	Precision	Recall	F1 Score
KNN	0.9642	0.9606	0.8872	0.9224
SVM	0.9877	0.9765	0.9689	0.9727
NB	0.9808	0.9699	0.9485	0.9591
IFKNN	0.9782	0.9781	0.9782	0.9780

4.7 Conclusion

In this chapter, we implemented the Improved Fuzzy K Nearest Neighbors (IFKNN) algorithm for spam email detection, covering data preprocessing, feature extraction, and interface development. Detailed steps for initializing, training, and evaluating IFKNN showcased its superiority over traditional algorithms. Comprehensive evaluation demonstrated IFKNN's enhanced accuracy and robustness, emphasizing the potential of machine learning, particularly fuzzy logic, in spam detection. We explained IFKNN's model components, preprocessing steps, and dataset preparation, including text vectorization and splitting. Overall, this chapter offers a comprehensive guide to IFKNN implementation for spam detection, spanning preprocessing, feature extraction, model training, evaluation, and user-friendly interface development. Overall, this chapter served as a comprehensive guide to implementing the IFKNN algorithm for spam email detection, encompassing data preprocessing, feature extraction, model training and evaluation, and a user-friendly interface for testing and managing spam emails.

General Conclusion

This thesis presented a comprehensive study on the application of the Improved Fuzzy K-Nearest Neighbors (IFKNN) algorithm for spam email detection, aiming to address the growing challenge of spam emails by enhancing the accuracy and robustness of email classification systems. Through an extensive literature review, we established the significance of fuzzy logic and machine learning in this domain, which formed the foundation for developing our proposed method. Our findings underscore the potential of integrating fuzzy logic into machine learning algorithms to manage the uncertainty and vagueness inherent in spam classification. This approach not only improves spam detection systems but also contributes to advancing cybersecurity measures. Future work could explore the application of IFKNN to other domains of text classification and further enhance the algorithm by incorporating more advanced machine learning techniques and larger datasets. Finally, this model will be improved to detect phishing emails and make this implementation suitable for professional use in enterprises.

Bibliography

Chapter 1:

[1]: Dernoncourt, F. (2013). Introduction to fuzzy logic. Massachusetts Institute of Technology, 21, 50-56

[2]: Ali Sadollah, Introductory Chapter: Which Membership Function Is Appropriate in Fuzzy System? in Fuzzy Logic Based in Optimization Methods and Control Systems and Its Applications (IntechOpen, 2018), accessed May 21, 2024, https://www.intechopen.com/chapters/62600.

[3]: Sabri, N., Aljunid, S. A., Salim, M. S., Badlishah, R. B., Kamaruddin, R., & Malek, M. A. (2013). Fuzzy inference system: Short review and design. Int. Rev.Autom. Control, 6(4), 441-449.

[4]: G. Chen and Trung Tat Pham, Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems (Boca Raton, FL: CRC Press, 2001). ISBN 0-8493-1658-8.

[5]: AGENTI CHE RAGIONANO LOGICAMENTE. LOGICA FUZZY E.Mumolo mumolo@units.it

[6]: Jain, A., & Sharma, A. (2020). Membership function formulation methods for fuzzy logic systems: A comprehensive review. Journal of Critical Reviews, 7(19), 8717-8733.

[7]: Dr H.K. Lam (KCL) Advanced Topics of Nature-Inspired Learning Algorithms NILAs 2020-21, University of london, Introduction to Fuzzy Inference System.

[8]: George J. Klir and Bo Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications (Upper Saddle River, N.J: Prentice Hall PTR, 1995), ISBN 0-13-101171-5.

[9]: Carl W. Entemann, A fuzzy logic with interval truth values, Fuzzy Sets and Systems 113 (2000) 161–183, DOI: https://doi.org/10.1016/S0165-0114(98)00101-8.

[10]: Site: https://fastercapital.com/startup-topic/Fuzzy-Control.html,FasterCapital Fuzzy Control, Number of Articles: 14, Date: 12/02/2024 hour: 12:15.

[11]: Site: https://klu.ai/glossary/fuzzy-control-system, Title What is a fuzzy control system? by Stephen M. Walker II, Co-Founder / CEO, 13/02/2024 hour: 12:13.

[12]: Université à Milan, Italie <https://valentini.di.unimi.it › Bioinformatica05> PDF Link: https://valentini.di.unimi.it/SlideCorsi/Bioinformatica05/Fuzzy-Clustering-lecture-Babuska.pdf.

[13]: https://www.geeksforgeeks.org/ml-fuzzy-clustering/, Title: ML | Fuzzy Clustering, date: 14/02/2024, Hour: 12:01

[14]: Jyh-Shing Roger Jang et al., Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, First Edition, Prentice Hall, 1997 (PDF)

[15]: Carl W. Entemann, A fuzzy logic with interval truth values, School of Computer Science and Telecommunications, College of Arts and Sciences, Roosevelt University, 430 South Michigan Avenue, Chicago, IL 60605-1394, USAReceived June 1996; received in revised form March 1998.

[16]: Tomasiello, S., Pedrycz, W., & Loia, V. (2022). Combining Artificial Neural Networks and Fuzzy Sets. In S. Tomasiello, W. Pedrycz, & V. Loia (Eds.), Contemporary Fuzzy Logic (Vol. 1, pp. 79–93). https://link.springer.com/book/10.1007/978-3-030-98974-3.

[18]: Jaekel, Jens, Ralf Mikut, et al. "Fuzzy Control Systems" Chapter, January 2004. ResearchGate publication with URL https://www.researchgate.net/publication/258484987.

[19]: Jang, Jyh-Shing Roger, Chuen-Tsai Sun, and Eiji Mizutani. "Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence." Prentice Hall, 1997.

Chapter2:

[20]: Shervin Minaee et al., Deep Learning–Based Text Classification: A Comprehensive Review, ACM Computing Surveys 54, no. 3 (April 30, 2022): 1–40, accessed May 21, 2024, https://dl.acm.org/doi/10.1145/3439726.

[21]: Dipanjan Sarkar, Text Analytics with Python (Berkeley, CA: Apress, 2016), accessed May 21, 2024, http://link.springer.com/10.1007/978-1-4842-2388-8.

[22]: International Conference on Advances in Computing, Communications and Informatics, 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI): 13-16 Sept. 2017. (Piscataway, NJ : IEEE, 2017).

[23]: Jennifer S. Raj et al., eds., Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020, vol. 59, Lecture Notes on Data Engineering and Communications Technologies (Singapore: Springer Singapore, 2021), accessed May 21, 2024, https://link.springer.com/10.1007/978-981-15-9651-3.

[24]: Qian Li et al., A Survey on Text Classification: From Traditional to Deep Learning, ACM Transactions on Intelligent Systems and Technology 13, no. 2 (April 30, 2022): 1–41, accessed May 21, 2024, https://dl.acm.org/doi/10.1145/3495162.

[25]: International Conference on Circuits and Systems in Digital Enterprise Technology, 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET). ([Piscataway, New Jersey]: IEEE, 2018), ISBN: 978-1-5386-0576-9.

[26]: IRJET- International Research Journal of Engineering and Technology, accessed May 21, 2024, https://www.irjet.net/, e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[27]: Julliano Trindade Pintas, Leandro A. F. Fernandes, and Ana Cristina Bicharra Garcia, “Feature Selection Methods for Text Classification: A Systematic Literature Review,” Artificial Intelligence Review 54, no. 8 (2021): 6149–6200, accessed May 21, 2024, https://link.springer.com/10.1007/s10462-021-09970-6.

[28]: Site: https://www.geeksforgeeks.org/ethical-considerations-in-natural-language-processing-bias-fairness-and-privacy/?ref=header_search/, Last Updated : 05 Dec, 2023, Title: Ethical Considerations in Natural Language Processing: Bias, Fairness, and Privacy, date: 04/03/2024, 23:15.

[29]: Viktor Mayer-Schönberger, Kenneth Cukier, Book: "Big Data: A Revolution that Will Transform how We Live, Work, and Think",Book design by : Melissa Lotfy.

[30]: Jurafsky, D., & Martin, J. H. (2009). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition (2nd ed.). Upper Saddle River, N.J: Pearson Prentice Hall. Prentice Hall series in artificial intelligence. ISBN 9780131873216.

[31]: Haibo He, Yunqian Ma,Imbalanced Learning: Foundations, Algorithms, and Applications, Date: 05/03/2024 17:39.

[32]: Muhammad Zulqarnain, Rozaida Ghazali, Yana Mazwin Mohmad Hassim, Muhammad Rehan, Title: A comparative review on deep learning models for text classification, Indonesian Journal of Electrical Engineering and Computer Science, Vol. 19, No. 1, July 2020, pp. 325 335. DOI: 10.11591/ijeecs.

[33]: Ritu Tiwari, Mario F. Pavone, and Ranjith Ravindranathan Nair, eds., Proceedings of International Conference on Computational Intelligence: ICCI 2021, Algorithms for Intelligent Systems (Singapore: Springer Nature Singapore, 2023), accessed May 21, 2024, https://link.springer.com/10.1007/978-981-19-2126-1.

[34]: Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. A Survey on Text Classification: From Traditional to Deep Learning. ACM Trans. Intell. Syst. Technol. 37, 4, Article 111 (April 2021), 39 pages. https://doi.org/10.1145/1122445.1122456

[35]: Dr.S. Kannan, Vairaprakash Gurusamy, Associate Professor, Research Scholar, Preprocessing Techniques for Text Mining, https://www.researchgate.net/publication/273127322_Prepro

[36]: Xin Zhao, Zhe Jiang, and Jeff Gray, Text Classification and Topic Modeling for Online Discussion Forums: An Empirical Study from the Systems Modeling Community, in Advances in Data Mining and Database Management, ed. Alessandro Fiori (IGI Global, 2020), 151–186, accessed May 21, 2024, http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-9373-7.ch006.

[37]: Wang, J., Zou, H., & Sun, H. (2019). Handling noisy data in text classification: A review. Information Fusion, 44, 145-157.

[38]: Caliskan, A., Bryson, J. J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334), 183-186.

[39]: Grigonyuk, V., & Ryabov, D. (2020, June). Big Text advantages and challenges: classification perspective. [Proceedings of International Conference on Data Stream Mining & Applications] (pp. 221-232). (this paper is from the proceedings of a conference named "International Conference on Data Stream Mining & Applications").

[40]: Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. Big Data & Society, 3(2), 2053951716679679.

Chapter 3:

[41]: Poornachandra Sarang, Thinking Data Science: A Data Science Practitioner's Guide, The Springer Series in Applied Machine Learning (Cham: Springer International Publishing, 2023), accessed May 21, 2024, https://link.springer.com/10.1007/978-3-031-02363-7.

[42]: Pádraig Cunningham and Sarah Jane Delany, "K-Nearest Neighbour Classifiers - A Tutorial," ACM Computing Surveys 54, no. 6 (July 31, 2022): 1–25, accessed May 21, 2024, https://dl.acm.org/doi/10.1145/3459665.

[43]: Christo El Morr et al., Machine Learning for Practical Decision Making: A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics, vol. 334, International Series in Operations Research & Management Science (Cham: Springer International Publishing, 2022), accessed May 21, 2024, https://link.springer.com/10.1007/978-3-031-16990-8.

[44]: B. S. Daya Sagar et al., eds., Encyclopedia of Mathematical Geosciences, Encyclopedia of Earth Sciences Series (Cham: Springer International Publishing, 2023), accessed May 21, 2024, Encyclopedia of Mathematical Geosciences | SpringerLink.

[45]: Bhatia, N. (Corresponding Author), & Vandana. (2010). Survey of Nearest Neighbor Techniques. International Journal of Computer Science and Information Security (IJCSIS), Vol. 8, No. 2 accessed May 21, 2024, https://sites.google.com/site/ijcsis.

[46]: Prihadi, D., & Trisyanti, V. (2020). Fuzzy Logic Approach for Email SPAM Detection System. JuTISI (Jurnal Teknik Informatika dan Sistem Informasi), Vol. 6, No. 1, April 2020.

[47]: Keller, J. M., Gray, M. R., & Givens Jr., J. A. (1985). A Fuzzy k-Nearest Neighbor Algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, No. 4, July/August 1985.

[48]: Moutafis, I., Andreatos, A., & Stefaneas, P. (Year). Spam Email Detection Using Machine Learning Techniques. Journal/Conference Name, Volume (Issue), Page numbers.

[49]: Kornraphop Kawintiranon, Lisa Singh, and Ceren Budak, "Traditional and Context-Specific Spam Detection in Low Resource Settings," Machine Learning 111, no. 7 (2022): 2515–2536, accessed May 21, 2024, https://link.springer.com/10.1007/s10994-022-06176-x.

[50]: Senthil, M., Sruthi, D., Pravallika, G., Ajay, C., & Kumar, A. (Year of publication not provided). Email Spam Detection Using KNN Algorithm. International Journal of Creative Research Thoughts (IJCRT). Retrieved from https://www.ijcrt.org/.

[51]: Mange, P., Lule, A., & Savant, R. (2024). Advanced Spam Email Detection using Machine Learning and Bio-Inspired Meta-Heuristics Algorithms. INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING, International Journal of Intelligent Systems and Applications in Engineering (IJISAE), 12(4s), 122–135. ISSN: 2147-6799. Retrieved from www.ijisae.org.

[52]: Naeem Ahmed et al., Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges, Published 3 February 2022 https://www.hindawi.com/journals/scn/2022/1862888/.

[53]: Christopher D.Manning, Prabhakar Raghavan,Hinrich Schütze. (2008). Introduction to Information Retrieval. Cambridge University Press. (Chapter 2: Text Preprocessing).

[54]: Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed. draft). (Chapter 2: Regular Expressions and Text Normalization).

[55]: Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

[56]: Segal, R., Crawford, J., Kephart, J., & Leiba, B. (2003). SpamGuru: An Enterprise Anti-Spam Filtering System. IBM Thomas J. Watson Research Center.

[57]: Bezdek, J. C. (1999). Pattern Recognition with Fuzzy Objective Function Algorithms. Springer Science & Business Media. Book, Chapter: Chapter 5 - "The Fuzzy k-Nearest Neighbor Algorithm (Fuzzy k-NN)", Page: 127-156, DOI: 10.1007/978-1-4615-4889-2.

[58]: Bezdek, J. C. (1981). Pattern Recognition with Fuzzy Objective Function Algorithms. Springer Science & Business Media, Book: Chapter 6 - "Fuzzy k-Nearest Neighbor Algorithm", ISBN: 978-1461250447.

[59]: Hamming, R. W. (1950). Error detecting and error correcting codes. The Bell System Technical Journal, 29(2), 147-160. DOI: 10.1002/j.1538-7305. 1950.tb00463. x.

[60]: Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620. Doi: 10.1145/361219.361220.

[61]: Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. Pattern Recognition, 26(9), 1277-1294. https://doi.org/10.1016/0031-3203(93)90135-J.

[62]: Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27. https://doi.org/10.1109/TIT.1967.1053964.

[63]: Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13(Feb), 281-305. DOI: 10.5555/2188385.2188395.

[64]: Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (Vol. 2, pp. 1137-1143). DOI : 10.5555/1643031.1643047.

[65] : Maillo, J., Luengo, J., García, S., Herrera, F., & Triguero, I. (2017). Exact Fuzzy k-Nearest Neighbor Classification for Big Datasets. IEEE Transactions on Fuzzy Systems, 25(6), 1585-1597.

[66]: Kumbure, M. M., Luukka, P., & Collan, M. An Enhancement of Fuzzy K-Nearest Neighbor Classifier Using Multi-Local Power Means. Atlantis Studies in Uncertainty Modelling, Volume 1. Presented at the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019). School of Business and Management, LUT University, Skinnarilankatu 34, 53850.

Chapter 4:

[67]: Documentation Visual Studio Code https://code.visualstudio.com/docs, Date: 15/05/2024, Hour: 12:41.

[68]: Python, https://www.python.org/doc/essays/blurb/ Date: 15/05/2024, Hour: 12:40.

[69]: NumPy. https ://numpy.org/doc/stable/user/whatisnumpy.html Date: 15/05/2024, Hour: 12:49.

[70]: Pandas, https://pypi.org/project/pandas/, Date: 15/05/2024, Hour: 12:40.

[71]: Tkinter. https ://www.tutorialspoint.com/python/pythonguiprogramming, Date: 15/05/2024, Hour: 18:50.

[72]: https://docs.python.org/fr/3/library/datetime.html, Date: 15/05/2024, Hour: 18:50.

[73]: https://he-arc.github.io/livre-python/pillow/index.html, Date: 15/05/2024, Hour: 18:59.

[74]: Scikit-learn, https://scikit-learn.org/stable, Date: 15/05/2024, Hour: 19:11.

[75]: The Natural Language Toolkit (NLTK), https://www.nltk.org, Date: 15/05/2024, Hour: 19:14.

[76]: https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset, Dataset: Spam Email Dataset.

[77]: https://github.com/sahilsehwag/FuzzyKNN, Fuzzy KNN.