

**UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION
INSTITUTE OF SCIENCE AND TECHNOLOGY**

**MODIFIED k -NEAREST NEIGHBOR CLASSIFIERS FOR DEALING WITH
SECURE ENCRYPTED DATA**



MASTER THESIS

Ali Abbas Younis Al-Arbo

THE DEPARTMENT OF INFORMATION TECHNOLOGY

THE PROGRAM OF INFORMATION TECHNOLOGY

July, 2017

**UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION INSTITUTE
OF SCIENCE AND TECHNOLOGY**



**MODIFIED k -NEAREST NEIGHBOR CLASSIFIERS FOR DEALING WITH
SECURE ENCRYPTED DATA**

MASTER THESIS

Ali Abbas Younis Al-Arbo

1406050030

THE DEPARTMENT OF INFORMATION TECHNOLOGY

THE PROGRAM OF INFORMATION TECHNOLOGY

Thesis Supervisor: Assist. Prof. Dr. Shadi Al-Shehabi

Türk Hava Kurumu Üniversitesi Fen Bilimleri Enstitüsü'nün 1406050030 numaralı Yüksek Lisans öğrencisi “**Ali Abbas Younis Al ARBO**” ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “**MODIFIED k -NEAREST NEIGHBOUR CLASSIFIERS FOR DEALING WITH SECURE ENCRYPTED DATA**” başlıklı tezini aşağıda imzaları bulunan jüri önünde başarı ile sunmuştur.

Supervisor

Assist. Prof. Dr. Shadi AL SHEHABI

Türk Hava Kurumu Üniversitesi



Jury Members

Assist. Prof. Dr. Abdül Kadir GÖRÜR

Çankaya Üniversitesi



Assist. Prof. Dr. Yuriy ALYKSYEYENKOV

Türk Hava Kurumu Üniversitesi



Assist. Prof. Dr. Shadi AL SHEHABI


Türk Hava Kurumu Üniversitesi



Thesis Defense Date: 31.07. 2017

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.



Ali Abbas Younis Al-Arbo

July, 2017

ACKNOWLEDGEMENTS

This is a welcome opportunity to show profound gratitude to all those who supported me in completing this piece of work.

First of all, I would like to thank the wonderful teaching staff at the Department of Informational Technology at THKU.

Special thanks and respect go to my supervisor, Assist. Prof. Dr. Shadi Al-Shehabi for his patience in guiding, revising and giving me his valuable academic opinions and suggestions. Thanks to my family, specially to my father who sustained me to maintain my morale whenever I lost self-confidence and aided me to persevere at hard working and overcome all the spiritual obstacles that faced me during the course of my study.

Special thanks to my colloquies and my family.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF FIGURES.....	ix
LIST OF TABLES	x
LIST OF ABBREVIATIONS	xii
ABSTRACT	xiii
ÖZET.....	xv
CHAPTER ONE	1
INTRODUCTION.....	1
1.1. Data Types.....	2
1.2. Data Classifiers.....	2
1.3. Classifiers Performance Evaluation	3
1.4. Data Encryption.....	4
1.5. Problem Definition	5
1.6. Aim Of The Thesis	5
Chapter Two.....	6
Literature Review.....	6
Chapter three	16
k-Nearest Neighbor Classifier.....	16
3.1. Paillier Cryptosystem	16
3.2. Classifier Requirements	17
3.3. Attribute Weighting Methods.....	18
3.3.1. Global Entropy	18

3.3.2. Local Entropy	19
3.3.3. Global Gini Diversity Index	19
3.3.4. Local Gini Diversity Index	20
3.4. Methodology	20
3.4.1. Secure Multiplication (Sm)	20
3.4.2. Secure Squared Euclidean Distance (SSED).....	21
3.4.3. Basic Classifier	22
3.4.4. Weighted Distance.....	23
3.4.5. Nearest Neighbors Selection	23
3.4.6. Class Prediction	24
Chapter four	25
Experimental Results	25
4.1. Benchmark Datasets	25
4.1.1. Car Evaluation Dataset	25
4.1.2. Pima Indian Diabetes Dataset.....	25
4.1.3. Statlog (German Credit Data) Dataset.....	26
4.1.4. Credit Approval Dataset	26
4.2. Performance Measures	26
4.2.1. Overall Accuracy.....	27
4.2.2. F-Score	27
4.3. Weighting Methods	28
4.3.1. Global Entropy	28
4.3.2. Local Entropy	30
4.3.3. Global Gini Diversity Index	32
4.3.4. Local Gini Diversity Index.....	35

4.4. Basic Method.....	37
4.5. Plain Pima Indian Diabetes Dataset	39
Chapter five.....	42
Discussion	42
Chapter six	47
Conclusion.....	47
References	49
CURRICULUM VITAE.....	52

LIST OF FIGURES

Figure 2.1: k -NN classification	12
Figure 4.1: Illustration of classifier's performance using global entropy.....	30
Figure 4.2: Illustration of classifier's performance using local entropy.....	32
Figure 4.3: Illustration of classifier's performance using global Gini diversity index.	34
Figure 4.4: Illustration of classifier's performance using local Gini diversity index.	37
Figure 4.5: Illustration of basic classifier's performance.	39
Figure 5.1: Performance summary of the classifiers.....	43
Figure 5.2: F-score of each classifier per dataset.	45
Figure 5.3: Execution time for experiments in seconds.....	45

LIST OF TABLES

Table 4.1: Cars classification confusion matrix using global entropy.	28
Table 4.2: Pima Indian Diabetes classification confusion matrix using global entropy.	29
Table 4.3: German Credit Card classification confusion matrix using global entropy.	29
Table 4.4: Credit Card Approval dataset classification confusion matrix using global entropy.	29
Table 4.5: Classifier's performance summary using global entropy.	30
Table 4.6: Cars classification confusion matrix using local entropy.	30
Table 4.7: Pima Indian Diabetes classification confusion matrix using local entropy.	31
Table 4.8: German Credit Card classification confusion matrix using local entropy.	31
Table 4.9: Credit Card Approval dataset classification confusion matrix using local entropy.	31
Table 4.10: Classifier's performance summary using local entropy.	32
Table 4.11: Cars classification confusion matrix using global Gini diversity index.	33
Table 4.12: Pima Indian Diabetes classification confusion matrix using global Gini diversity index.	33
Table 4.13: German Credit Card classification confusion matrix using global Gini diversity index.	33
Table 4.14: Credit Card Approval classification confusion matrix using global Gini diversity index.	34
Table 4.15: Classifier's performance summary using global Gini diversity index. ..	34
Table 4.16: Cars classification confusion matrix using local Gini diversity index. .	35
Table 4.17: Pima Indian Diabetes classification confusion matrix using local Gini diversity index.	35
Table 4.18: German Credit Card classification confusion matrix using local Gini diversity index.	36

Table 4.19: Credit Card Approval classification confusion matrix using local Gini diversity index.	36
Table 4.20: Classifier's performance summary using local Gini diversity index.	36
Table 4.21: Cars classification confusion matrix using the basic classifier.....	37
Table 4.22: Pima Indian Diabetes classification confusion matrix using the basic classifier.....	38
Table 4.23: German Credit Card classification confusion matrix using the basic classifier.....	38
Table 4.24: Credit Card Approval classification confusion matrix using the basic classifier.....	38
Table 4.25: Basic classifier's performance summary.....	39
Table 4.26: Plain Pima Indian Diabetes classification results using global entropy.	40
Table 4.27: Plain Pima Indian Diabetes classification results using local entropy...	40
Table 4.28: Plain Pima Indian Diabetes classification results using global Gini diversity index.	40
Table 4.29: Plain Pima Indian Diabetes classification results using local Gini diversity index.	41
Table 4.30: Plain Pima Indian Diabetes classification results using the basic classifier.....	41
Table 5.1: Performance summary of the classifiers.	43
Table 5.2: Execution time of experiments in seconds.....	45

LIST OF ABBREVIATIONS

k -NN	k -Nearest Neighbors
ED	Euclidean Distance
SED	Squared Euclidean Distance
SMC	Simple Matching Coefficient
SM	Secure Multiplication
SSED	Secure Squared Euclidean Distance
SMIN _{n}	Secure Minimum out of n Numbers

ABSTRACT

MODIFIED k -NEAREST NEIGHBOR CLASSIFIERS FOR DEALING WITH SECURE ENCRYPTED DATA

Al-Arbo, Ali Abbas Younis

Master, Department of Information Technology

Thesis Supervisor: Assist. Prof. Dr. Shadi Al-Shehabi

July 2017, 68 pages

The recent trend of using shared servers for data storage and management poses many challenges to the data owners to protect the information being outsourced, especially if these data contain personal or sensitive information. To overcome these problems, the data are encrypted so that only authorized clients, who have the encryption key, are able to decrypt the contents of these data. Data encryption makes it impossible to apply traditional data mining techniques to these data, as these techniques must access the values stored in the data and the data type of each attribute. Data mining is very important for any corporation to make the best use of their data. One of the most important data mining tasks is the data classification, which enables predicting a class for a new unclassified tuple depending on the existing classified data. Thus, it has become mandatory to classify encrypted data without the need to reveal the encryption key to the data management server or expose any stored data to any other authorized client who is unrelated to those data. k -Nearest Neighbors classifier is widely used to classify data. Thus, many Secure k -NN classifiers are proposed to enable classifying a dataset without having access to these data using homomorphic cryptosystems, which are cryptosystems where an encrypted result of a mathematical operation on encrypted data may be calculated without decrypting these data. The existing Secure k -NN

classifiers rely on homomorphic calculations to find the distances among tuples depending on the encrypted values stored in the data.

In this study, modified Secure k -NN classifiers are proposed that use homomorphic calculations to compute distances among tuples depending on the encrypted values of the data and the weight of each attribute, according to its contribution to the actual classification. Two weighting methods are tested in this study, which are information entropy and Gini diversity index. Each method is tested in two different schemes that are global and local. Global weighting calculates one weight per attributes while local weighting calculates a weight per each class for each attribute. The experimental results show significant improvement in classification results, compared to the basic classifier.

Keywords: Data Mining; Classification; Encryption; Homomorphic; k -Nearest Neighbors.

ÖZET

KRIPTOLANMIŞ VERİLER İLE GÜVENLİ İŞLEM İÇİN K-EN YAKIN KOMŞU SINIFLANDIRICILARI DEĞİŞİKLİĞİ

Al-Arbo, Ali Abbas Younis

Yüksek Lisans, Bilişim Teknolojileri Anabilim Dalı

Tez Danışmanı: Doç. Dr. Shadi Al-Shehabi

Temmuz 2017, 68 sayfa

Son zamanlarda veri depolama ve veri yönetme amaçlı olarak ortaya çıkan paylaşımlı serverların kullanımı, özellikle bu veriler kişisel ve hassas bilgiler içermekteyse, veri sahipleri için çok sayıda güvenlik açığı ortaya koymaktadır. Bu sorunların üstesinden gelmek amacıyla veriler yalnızca kripto şifresine sahip yetkili kişilerin erişimine izin vermek amacıyla kriptolanmaktadır. Veri kriptolama işlemi bu verilere geleneksel veri madenciliğinin uygulanmasını imkânsız hale getirmektedir, çünkü bu tekniklerin bu verilerdeki değerlere ve içerikteki her özelliğin veri türüne erişiminin olması gerekmektedir.

Veri madenciliği her şirket için verilerinden en yüksek verimi alabilmek amacıyla çok önemlidir. En önemli veri madenciliği uygulamalarından biri sınıfı henüz belli olmayan bir veri grubunun (*tuple*) sınıfının mevcut sınıflandırılmış verilere dayanarak tahmin edilmesini mümkün kılan *Veri Sınıflandırması*dır. Bu bağlamda, kriptolama şifresini veri yönetim serverına göstermeden veya depolanmış herhangi bir veriyi bu veriyle ilgisi olmayan başka bir yetkili kullanıcının görmesine gerek kalmayacak şekilde kriptolanmış verilerin sınıflandırılması elzem hale gelmiştir. Verilerin sınıflandırılmasında yaygın olarak *k-En Yakın Komşu* sınıflandırma sistemi kullanılmaktadır. Verilere erişim olmaksızın bir verisetinin sınıflandırılması için

homomorfik kripto sistemleri kullanarak uygulanan güvenli birçok k -EYK sınıflandırıcıları önerilmektedir. Homomorfik kripto sistemlerinde kriptolanmış veriler üzerinde uygulanan bir matematiksel işlemin kriptolanmış sonucu bu veriler deşifre edilmeksizin hesaplanabilmektedir. Mevcut güvenli k -EYK sınıflandırıcıları, verilerde depolanmış olan kriptolanmış değerlere bağlı olan veri grupları (*tuples*) arasındaki mesafeleri bulmak amacıyla homomorfik hesaplamalara dayanmaktadır.

Bu araştırmada veri grupları arasındaki mesafelerin hesaplanması için verilerin kriptolanmış değerine ve asıl sınıflandırmadaki katkısına göre her özelliğin ağırlığına dayanan ve homomorfik metotları kullanan güvenli-modifiye k -EYK sınıflandırıcıları önerilmektedir. Bu araştırmada iki ağırlıklandırma metodu test edilmektedir; Bilgi Entropisi ve Gini Çeşitlilik Endeksi. Her metot iki değişik şekilde test edilmiştir, lokal ve global. Global ağırlıklandırma her özellik için bir ağırlık hesaplamaktadır; diğer yandan, lokal ağırlıklandırma her özellik için her sınıfın ağırlığını hesaplamaktadır. Deneysel sonuçlar sınıflandırma sonuçlarında Temel Sınıflandırma Sistemine göre önemli düzeyde gelişme göstermektedir.

Anahtar Kelimeler: Veri Madenciliği; Sınıflandırma; Kriptolama; Homomorfik; k -En Yakın Komşu (k -EYK).

CHAPTER ONE

INTRODUCTION

In recent years, providing computerized services over a network has become mandatory for every service provider. This leads to an enormous amount of data being stored on servers or transferred through the networks. These data may include some sensitive or personal information that requires maximum security measures to protect them during storage and communication. Moreover, the use of shared servers raises many additional security challenges, as the honesty of the server owner and the security of the communications between the service provision server and the data management server are being questioned. Thus, these data must be stored, processed and transferred in an encrypted form, without declaring the encryption key or exposing any data access pattern to the data management server. This ensures providing the required service without compromising the security of any valuable data. The encryption key is provided to the end user who has the authority to view that specific piece of data sent from the data management server after executing the required query.

The framework of the remainder of this study is organized as follows. Chapter two provides reviews of the literature related to this study. Chapter three explains the basic requirements that a secure classifier must fulfill and the methodology used to modify the basic k -NN classifier. Chapter four demonstrates the datasets used in experiments, the measures used to evaluate the performance of the classifiers, the conducted experiments and the results acquired from these experiments using all the datasets to evaluate the performance of every classifier. Chapter five discusses and compares these results and the performance of the classifiers. Finally, chapter six presents the conclusion of this study.

1.1. Data Types

Most of the data stored in a database table can be classified into one of two types that are numerical data and categorical data. As the name suggests, numerical data hold numerical values. These values may also be categorized into two categories that are discrete and continuous. Discrete numerical data are usually used with countable variables, while continuous numerical data are usually used to represent a variable that may have any value within some range. For example, the number of people is a discrete number while their weight is continuous.

Categorical data holds one of a predefined set of categories. This type of data may also be classified into two types that are ordinal and nominal. In ordinal categorical data, the categories have meaningful order. This means that it is possible to say that there is one category closer to another than the remaining categories, such as a student's grades. While the nominal categorical data have no meaningful order, or in other words, the distances among the categories are the same, such as gender. In categorical data, categories may be represented using numbers, and in ordinal data, these numbers may be used to calculate the distance between categories. But in nominal categorical data, these numbers represent only category number and all categories have the same distance from each other. It is important to mention that some data tables may include both numerical and categorical data simultaneously.

1.2. Data Classifiers

Data mining is the process of extracting knowledge and relations from a huge database, which contains data that may look unrelated to each other. The extracted knowledge assists predicting future behaviors of new entries according to their characteristics. Classifiers are one of the widely used data mining tools, which are used to suggest a matching class for a new entry depending on the patterns and relations learned from an existing pre-classified database, called training database. These classifiers use different techniques for classification; some of them rely on mathematical and statistical techniques, while some are based on artificial intelligence and machine

learning. One of the most used data classifiers is the k -Nearest Neighbors (k -NN), which depends on finding a predefined number (k) of neighbors for the new entry by comparing its data to the existing training data, then, the dominant class among these neighbors is used as a predicted class for the new entry. By knowing the predicted class, it becomes easy to predict the behavior of the new entry depending on the behavior of the entries that are already in this class.

The k -Nearest Neighbors of a record are found by calculating the distances between this record and all the records exist in the database, then, the (k) records with the least distances are chosen. To find these distances, all records must have the same structure (same attributes) to calculate the distances between the values of each attribute in order to calculate the overall distance. For numerical data, this is achieved using equations that calculate the distance depending on the values in the records. While, for nominal categorical data, the distance between two identical categories in the same attribute is zero, but there are many techniques proposed to calculate the distance between two non-identical categories in the same attribute.

1.3. Classifiers Performance Evaluation

As classifiers are used for prediction, it is difficult to evaluate these predictions as they have not happened yet. But, it is also very important to know how good the performance of a classifier is. Thus, to evaluate a classifier, it is used to classify the same training data that are used by the classifier to extract the knowledge. As these data are already classified, the prediction of the classifier may be compared to the actual class of the record. These results are distributed in a matrix called confusion matrix. Using this matrix, it is possible to calculate many performance measures that are used to describe and evaluate the performance of a classifier such as purity, accuracy and F-measure.

1.4. Data Encryption

One efficient way of storing data on a third party server, without concerns that these data may be compromised by the server owner, is to store this in an encrypted form. So that, even if there is any unauthorized access to that data, the retrieved data is still ciphered making it impossible for the attacker to decrypt this data without the encryption key. This shows the importance of protecting the encryption key, especially when the honesty of the server owner is in question, and as a result, it is preferred not to share that encryption key with any parties other than the authorized users who are making use of the provided service.

On the other hand, it is still important to execute the required queries, on the database, in the data management server, rather than sending the entire encrypted tables to the service provision server to do the required tasks, especially if the service provision server is a shared server too. In this case, it is critical to keep the data on a server and the encryption key in another server without any encryption key information interchange between these servers. For categorical data, encrypting the values in the database affects only the category name, which means that from the server's point of view, values are still either equal or not equal. But for numerical data, these encrypted values are required to be processed by the computer. This leads to the need of an encryption system that encrypts the data in a way that it is still possible to process those data without the need to share any information about the encryption key used for encryption. This kind of encryption is called Homomorphic Encryption.

Using this kind of encryption, the data management server executes any query requested from an authorized client and returns the required results to the client without the need to decrypt any of the database contents. At this point, the client receives the encrypted results from the data management server, but these results are useless unless the client is authorized by the service provision server and provided with the encryption key to decrypt these results into a more suitable form. This procedure ensures that no data is leaked from the data management server without the authorization of the service provision server, or in other words, the service provision server must have a role in the data leakage and it is impossible for the data management server to leak any decrypted

data to any unauthorized client. Moreover, there is still a problem of using a database that consists of both categorical and numerical data simultaneously on a third party server, where it is insecure to reveal the structure of the database as it may compromise the security of the stored information.

1.5. Problem Definition

The benefits of using shared servers, or third party servers, encouraged many service providers to use these servers in order to deliver their services to their clients. These benefits come with challenges faced by the service providers, which is the importance of protecting the clients' information. To protect these data, they must be encrypted before storing them into those servers in a way that it is still possible to process these data without the need to decrypt them. For the k -NN classifier, it is possible to use the methods used to classify categorical data no matter if the data is encrypted or not as the encryption will only affect the category name, which maintains the same relevance where attribute values are to be equal or not. On the other hand, the use of homomorphic encryption methods enables the data server to apply the k -NN classifier without the need to know the encryption key, which preserves the security of the stored data. Moreover, in the real world, most of the databases include both categorical and numerical values, making it impossible to use any of these techniques solely.

1.6. Aim Of The Thesis

In this thesis, a method is proposed to enable a database server of applying the k -NN classifier on an encrypted data that contains both categorical and numerical data. This method must be able to classify the data and return the results without the need of the encryption key. For maximum security, this method must also be able to classify the data without the need to know the structure of the database, which means that the type of each attribute, whether to be categorical or numerical, is not revealed.

CHAPTER TWO

LITERATURE REVIEW

Knowledge discovery from databases is a very important topic that is attracting many researchers for a wide variety of real world applications. According to **Fayyad et al.** [1], this attraction is caused by the obvious success of the early scientific applications of these methods, which led to a huge interest in using them in business applications. For example, the system proposed by **Agrawal et al.** [2] is designed to support marketing decisions by analyzing huge database, not only to detect if a pattern exists, but to recognize that pattern too. Another application, by using knowledge discovered from databases, is proposed by **Sabhnani and Serpen** [3] for intrusion detection by using an existing database, which covers four major intrusion categories, to extract knowledge in order to use this knowledge to predict if any of the new connections is an intrusion attempt.

The study conducted by **Sinha and Zhao** [4] suggests that classification is one of the popular problems that are solved using data mining techniques, where a classifier may use a pre-classified database, called the training database, to recognize a pattern, or relations, between the class of the entry and the attributes of that entry. If such pattern is recognized, the classifier will be able to predict a matching class for a new entry. The training data provided to the classifier may be actual data collected from real life or may represent the experience of an expert in that field who may give decisive possibilities for each class.

Another type of classifiers classifies the new data without the need to find patterns in the training data. This is done by attempting to find similarity between the new data and the data existing in the training set. Later, the classes of the most similar records are used to predict a class for the new data. This kind of classifiers is found, by **Aha et al.** [5], to be faster and more flexible than the pattern-recognition classifiers when learning for training data is required to be incremental. The results of the experiments conducted by **Sabhnani and Serpen** [3] shows that a classifier's evaluation, regarding

classification performance, may vary from one database to another. In other words, a specific classifier may result in better classification than another on a specific database, but may be outperformed by the other classifier when both are applied to a different database.

One of the widely used and simplest classifiers, as described by **Peterson** [6], is the k -Nearest Neighbor classifier, which is based on calculating the distance between the new data and each tuple in the training data to predict a class for the new data depending on the classes of the (k) tuples, in the pre-classified data, with the least distance from the new tuple. Thus, this classifier has the ability to work with databases where no or little prior knowledge, about how the data are distributed in the classes, exists.

To find the nearest neighbors to a specific tuple, the distance between this tuple and every tuple in the database must be calculated. The best method, to calculate these distances, is by considering each attribute as a dimension, then use the Euclidean Distance (ED) equation to measure that distance as shown by **Deza and Deza** [7]. For example, the data tuples (r, s), which contain (n) attributes, have a Euclidean distance $D_e(r, s)$ that is calculated using the equation,

$$D_e(r, s) = \sqrt{(r_1 - s_1)^2 + (r_2 - s_2)^2 + \dots + (r_n - s_n)^2}$$

which can be written as,

$$D_e(r, s) = \sqrt{\sum_{x=1}^n (r_x - s_x)^2}$$

The Euclidean distance is proven, by **Uhlmann** [8], to be a metric measure that satisfies the triangle inequality, which means that the summation of the Euclidean distances from one point to two different points must be equal to or larger than the distance between these two points. This feature is not mandatory in the comparisons made in the k -NN algorithm, as it is only required to know which point is closer to a specific point. These comparisons can be achieved using the Squared Euclidean Distance (SED), as $x > y$ when $x^2 > y^2$ and vice versa, as also, $x = y$ when $x^2 = y^2$ and

vice versa. Thus, it is possible to use the SED in the k -NN algorithm to simplify the calculations. The equation of the SED is,

$$D_e^2(r, s) = \sum_{x=1}^n (r_x - s_x)^2$$

The application of this equation on numerical data is very straight forward, and the results are meaningful. It may also be applied to the ordinal categorical data when the categories are represented using numbers, considering that the chosen numbers represent the order of the categories. However, as shown by **Boriah et al.** [9], this equation cannot be applied to nominal data, where categories are described using text, or described by numbers that do not represent the order of the categories. This raises the problem of measuring distances among tuples that contain categorical data. To overcome this problem, the Hamming distance is used with categorical data, instead of the Euclidean distance.

The Hamming distance is proposed by **Hamming** [10] for the purpose of error detection and correction. It is originally used to compare messages' bits to find the distance among these messages. It simply calculates the number of bits that share the same position but has different values. Mathematically, a Hamming distance between two bits, x and y , with the same position in the message position is,

$$D_h(x, y, z) = \begin{cases} = 0, & x = y \\ = z, & x \neq y \end{cases}$$

This formula is widely used to calculate the Hamming distance between two tuples that are consisted of categorical attributes. One of the simplest methods that use this formula is the Simple Matching Coefficient (SMC) that calculates the distance between two tuples r and s , where each is consisted of (d) categorical attributed, using the following equation,

$$SMC(r, s) = \sum_{x=1}^d D_h(r_x, s_x, 1)$$

The distances measured using methods like SMC are calculated by assuming that all attributes have the same effect on class prediction. This assumption is described by **Li et al.** [11] as rarely true in real life databases, especially in high-dimensional data where many attributes are found to be noisy, subsequently, have no effect on the data classification. Thus, many methods are proposed to eliminate the effect of such attributes on the classifier. In other words, the attributes are weighted, in which the attributes, that have high influence in the classification, are given more weight than the attributes that are found to be noisy or have no effect on the data classification.

Many methods are proposed to calculate the weight of each attribute depending on its effect on the data classification. These methods can be classified into two categories, global and local. The methods in the global category calculate one weight per attribute depending on the data pattern with respect to the classes that these data classified into it. The local methods calculate an attribute weight per attribute per class, which means that each attribute will have a number of weights equal to the number of classes in that database. These weights are calculated depending on the data pattern in that attribute through each class. Four attribute-weighting methods are used by **Chen and Guo** [12] to improve the performance of the k -NN algorithm when applied to categorical data. These methods are based on two measures, the information entropy and the Gini diversity index.

The information entropy is proposed by **Shannon** [13] as a measure of information and how unpredictable is a random variable. This measure is widely used in data mining techniques to measure the contribution of an attribute in a certain dataset. If the category (s_d) is in a dataset (tr) that contain a total of (N) tuples (X), which consist of (d) attributes and a class (y), distributed in (M) classes, each class is denoted by (c_m), then the global entropy of that category is calculated as,

$$H_G(s_d) = - \sum_{m=1}^M p(m|s_d) \log_2 p(m|s_d)$$

where,

$$p(m|s_d) = \frac{\sum_{(x,y) \in c_m} D_h(x_d, s_d, 1)}{\sum_{(x,y) \in tr} D_h(x_d, s_d, 1)}$$

and the weight of the attribute (d) using global entropy is,

$$\omega_d^{(H_G)} = e^{-\frac{1}{\log_2 M} \sum_{s_d \in S_d} p(s_d) \times H_G(s_d)}$$

while, for local entropy,

$$H_L(m, d) = - \sum_{s_d \in S_d} p(m|s_d) \log_2 p(m|s_d)$$

where,

$$p(m|s_d) = \frac{1}{|c_m|} \sum_{(x,y) \in c_m} D_h(x_d, s_d, 1)$$

and the weight of the attribute (d) with respect to class (m) using local entropy is,

$$\omega_{md}^{(H_L)} = e^{-\frac{1}{\log_2 |S_d|} \times H_L(m, d)}$$

The Gini diversity index is proposed by **Gini** [14] as a measure of concentration deficiency or diversity. This measure is also widely used in combination with the Hamming distance to measure the contribution of an attribute in the classification of a dataset. Again, **Chen and Guo** [12] used this index to calculate two different weights, global and local, that each of them is used to improve the performance of the k -NN classifier. The global Gini diversity factor is calculated using the following equation,

$$G_G(s_d) = 1 - \sum_{m=1}^M [p(m|s_d)]^2$$

where $p(m|s_d)$ is calculated using the same equation used to calculate the global entropy. The weight calculated using the global Gini diversity index is,

$$\omega_d^{(G_G)} = e^{-\frac{M}{M-1} \sum_{s_d \in S_d} p(s_d) \times G_G(s_d)}$$

while the formula used to calculate the local Gini diversity index is,

$$G_L(m, d) = 1 - \sum_{s_d \in S_d} [p(m|s_d)]^2$$

using the same equation of $p(m|s_d)$ as used to calculate the local entropy. The weights are calculated, depending on the local Gini diversity index, using the formula

$$\omega_{md}^{(G_L)} = e^{-\frac{|S_d|}{|S_d|-1} \times G_L(m,d)}$$

Eventually, the calculated global weight vector (W) is used to calculate the weighted distance between the tuples r and s is,

$$D_{Gw}(r, s, W) = \sum_{x=1}^d D_h(r_x, s_x, \omega_d)$$

and the weighted distance using the local weight matrix W_m , which is a two dimensional ($D \times M$) matrix, is calculated using the formula,

$$D_{LW}(r, s, W_m) = \sum_{x=1}^d D_h(r_x, s_x, \omega_{md})$$

By calculating the distances between the new tuple and every tuple in the pre-classified dataset, the k -NN uses these distances to find the tuples with the least distances. Then, the class that has the highest frequency among the selected tuples is selected as a class for the new unclassified tuple. The confidence of that prediction is measured by dividing the number of the selected tuples in the predicted class to the total number of selected tuples. For example, the point X_1 , in figure 2.1, is predicted by the k -NN classifier to be in class C, with a prediction confidence of 100% as the total number of the selected neighbors that are in class C is equal to the total number of the selected neighbors.

When the highest number of class frequency of the selected neighbors is achieved by more than one class, it is recommended by **Daelemans et al.** [15] to look for the next neighbor in order to break the tie. Thus, the point X_2 , in figure 2.1, is predicted to be in class A after breaking the first tie between classes A and B by selecting the next nearest neighbor, which happens to be in class A. The confidence of this prediction is 50% as there are three points of class A in the total six selected neighbors. It is also recommended to select all neighbors that share the exact same distance of any of the

included distance. For example, although the (k) , set to classify the point X_3 , is five, two more neighbors are found to be sharing the exact same distance as the fifth neighbor. Thus, the point X_3 , in figure 2.1, is predicted to be in class B with prediction confidence of 71% as the number of neighbors in class B is five out of the seven neighbors selected. These recommendations guarantee unbiased class prediction for the new data.

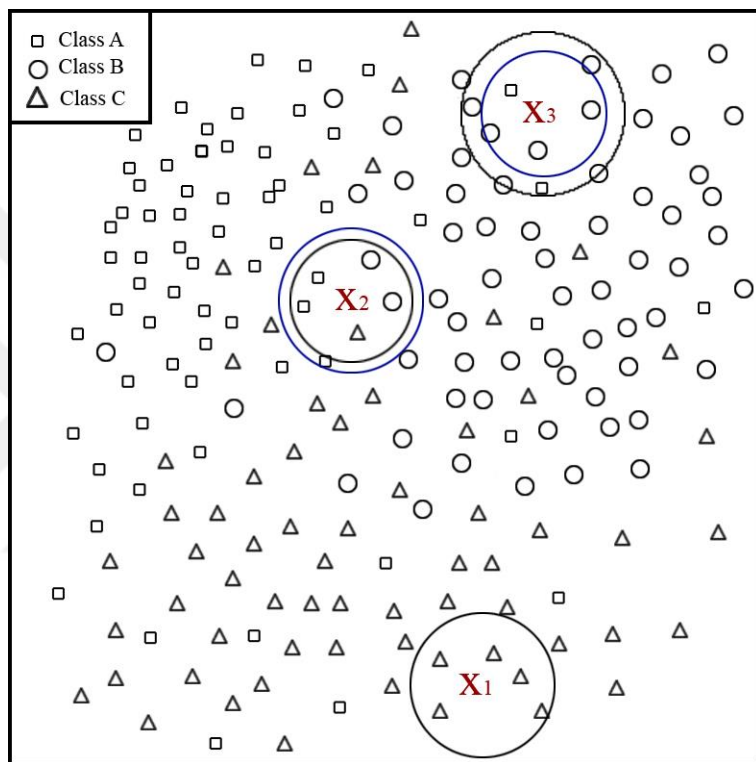


Figure 2.1: k -NN classification

Data are considered, by **Dong et al.** [16], as the heart of the applications and operations of business because of the critical role they have, in optimizing the workflow and customer's satisfaction improvement, for accelerated business growth. This importance leads to an exceptional interest in collecting and storing these data, which raises the problem of storage space and the cost of storage servers. Thus, more interest is shown recently in outsourcing these data to shared servers, **Gibson et al.** [17]. Moreover, these servers, and to ensure maximum data protection, are considered as untrusted servers. Thus, to outsource data with sensitive information, such as users' personal data or medical information, it is mandatory to encrypt these data so that only

authorized clients, who know the encryption key, are able to decrypt the data and view its contents, even if an unauthorized access occurs to the encrypted data, **Li et al.** [18].

By encrypting the data, access to those data can be restricted using the encryption key, instead of restricting access to these data, as the data access no longer controlled by the data owner. Thus, by **not** providing the encryption key to the servers owners, their ability to access the data is neutralized as they cannot decrypt the data, as long as they do not have the encryption key. On the other hand, it is still mandatory to process these data and execute authorized users' queries directly on the data server. These situations raise the importance of using Homomorphic encryption, **Rivest et al.** [19].

Homomorphic encryption is defined, by **Fontaine and Galand** [20], as an encryption system where mathematical operations are still applicable on the encrypted data without the need to know the encryption key. There are many types of homomorphic data, depending on the applicable operation on the encrypted results. For example, a multiplicatively homomorphic system is an encryption system where there is a certain way to get the encrypted product of multiplying two encrypted values without the need to know the encryption key. There are many homomorphic encryption systems, but this study focuses on homomorphic encryption systems that are qualified to enable the application the k -NN classifier on encrypted data without revealing the encryption key to the data server owner.

Another security concern is described by **Williams et al.** [21] that require attention, although it poses less risk than revealing the whole, or a part of, the secret key. This security is the data access pattern, where a data server owner may monitor that pattern in an attempt to gain information about the encrypted data. For example, if a dataset that is stored alphabetically in the server, the server owner may monitor the change in the index when new records are inserted in order to predict the encrypted information stored in the database. Although such methods leak a tiny amount of information about the information stored in the database, the sensitivity of this information makes it very important not to compromise its security under any circumstances. Thus, a secure k -NN classifier must not reveal any access pattern to the data server and perform the required query without the need of the encryption key.

Many secure k -NN methods are proposed for the encrypted numerical data. **Zhu et al.** [22] propose a secure k -NN method that does not reveal the encryption key to the user, but the data owner, in this method, is involved in the query encryption, which conflicts with the purpose behind using shared servers. Another secure k -NN method is proposed by **Yao et al.** [23] based on partitioning the data and sending the relevant partition that is assured to include the queried nearest neighbors to the user. These operations require no involvement from the data owner, but they leak the data access patterns to the data server owner, which may compromise the security of the stored data. Recently, a more secure k -NN method is proposed by **Elmehdwi et al.** [24] that requires no involvement from the data owner, and returns the exact required number of nearest neighbors without leaking any data access patterns. This method is denoted as $SkNN$.

The $SkNN$ method is designed to be applied on databases that are encrypted using Paillier cryptosystem, which is an additive homomorphic encryption scheme that is proposed by **Paillier** [25]. This cryptosystem is an asymmetric cryptosystem, which means that the system has two encryption keys, one public and one secret key. The public key is used only to encrypt data and execute homomorphic calculations and can never be used to decrypt the data, while the secret key is the only key that can be used to decrypt the data, **Fontaine and Galand** [20]. To demonstrate the properties of the Paillier cryptosystem, suppose there is a public key (pk) described as (N, g) , where N is a two large prime numbers' product and g is in \mathbb{Z}_N^* , is used by the Paillier encryption function (E_{pk}). The homomorphic addition of the $x, y \in \mathbb{Z}_N$ is calculated as

$$E_{pk}(x + y) \leftarrow E_{pk}(x) * E_{pk}(y) \bmod N^2$$

while the homomorphic multiplication is calculated as

$$E_{pk}(x * y) \leftarrow E_{pk}(x)^y \bmod N^2$$

Notice that (y) in the last equation is not encrypted. Thus, it is not possible to calculate the product of two encrypted numbers directly. On the other hand, it is not allowed to share any information about the data with the authorized user who has the secret key that enables the user from decrypting the data. To find the product of two

encrypted number without revealing details about these numbers to any of the participants, which are the data management server that has the public key and the authorized user who has the secret key, Secure Multiplication (SM) is used for this purpose. The SM is achieved by adding two random numbers, to the encrypted number that are required to be multiplied, using homomorphic addition. This assures the security of the data from being revealed to the authorized user as it may not be related to the executed query. As these numbers are known to the data server, as well as the public key, it may send the new numbers to the client with the secret key to decrypt the numbers, multiply them, then, send the encrypted result. The data server is now able to solve for the encrypted product of the two encrypted using the values of the random numbers it generated. This scheme ensures that no information about the decrypted numbers being multiplied is revealed to any of the participants.

As discussed earlier in this chapter, it is not necessary to calculate the Euclidean distance in order to compare distance. It is possible to achieve that by comparing the squared Euclidean distance. Thus, the *SkNN* method uses the Secure Squared Euclidean Distance (SSED) to find the nearest neighbors. In the earlier example, the squared Euclidean distance is calculated using the equation, $D_e^2(r, s) = \sum_{x=1}^n (r_x - s_x)^2$, where r and s are two tuples with n attributes. To calculate the SSED for r and s , the $E_{pk}(r_x - s_x)$ for each attribute is calculated using the homomorphic addition. Then, this value is squared using the SM. Finally, using the homomorphic addition equation,

$$E_{pk}(D_e^2(r, s)) = \prod_{x=1}^n E_{pk}((r_x - s_x)^2)$$

The secured squared Euclidean distances between the new tuple and every tuple in the dataset are calculated and compared to each other using Secure Minimum out of n Numbers (SMIN _{n}) to find the nearest neighbors to the requested query and return these encrypted tuples to the authorized user who has the secret key to decrypt these data.

CHAPTER THREE

K-NEAREST NEIGHBOR CLASSIFIER

In order to predict a class for a new tuple using the k -Nearest Neighbors classifier, the classifier first queries the (k) nearest neighbors to that tuple in the classified dataset, then predicts a class for the tuple depending on the queried neighbors. When the data are encrypted, this query must be achieved without revealing any of the information stored in the database, the database structure, or data access patterns to any of the participants in distances calculations.

3.1. Paillier Cryptosystem

The cryptosystem proposed by Pascal Paillier, the French researcher, is an asymmetric cryptosystem, which means that the key used for data encryption is different from the key used to decrypt it. The key used for data encryption is called the public key, as it is possible to distribute this key to other parties, which will enable them of encrypting the data but does not enable them of decrypting them. The decryption key is also classed the secret key, because it is only distributed among authorized parties who have the authority to decrypt the encrypted data. These keys are generated using the following algorithm.

Algorithm: Paillier keys generation

1: Randomly select two prime number p and q , where the general common divisor (gcd) of the product of these numbers and $(p-1)(q-1)$ is equal to one.

2: Compute RSA modulus $n = pq$ and the Carmichael's function λ , which equals the least common multiple of $(p-1)$ and $(q-1)$ which may be calculated using

$$\lambda = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}$$

3: select random generator g from $\mathbb{Z}_{n^2}^*$, where $gcd\left(\frac{g^\lambda \bmod n^2 - 1}{n}, n\right) = 1$.

4: Calculate the modular multiplicative inverse $\mu = \frac{\bmod n}{L(g^\lambda \bmod n^2)}$, where $L(u) = \frac{u-1}{n}$

5: The encryption (public) key is (n, g) .

The decryption (secret) key is (λ, μ) .

3.2. Classifier Requirements

The k -NN classifier is required to predict a class for a new encrypted tuple. The classified database, which is supposed to be used by the classifier to query the nearest neighbors to the new tuple in order to predict a class for the new tuple, is encrypted and stored in a shared server C_1 , which has the public key pk of the encrypted data. A user C_2 , who has the secret key sk for the encrypted data, needs to use the k -NN classifier to predict a class for an encrypted new tuple. The classifier must achieve the required task while maintaining the following conditions:

- The secret key sk is never revealed to C_1 .
- Never disclose decrypted data to C_1 .
- None of the stored information is sent to C_2 .
- Never expose any information about the data structure or access patterns to C_1 .

Finally, the classifier predicts a class, in an encrypted form, for the new tuple in C_1 and sends it to C_2 , which has the secret key that can be used to decrypt the predicted class.

3.3. Attribute Weighting Methods

In this study, four attribute weighting methods are tested. These methods are global entropy, local entropy, global Gini diversity index and local Gini diversity index. The calculations of these weights require no information about the values stored in the database. These calculations are based on the number of distinct values per attribute in different ways. The number of distinct values is the same whether the data are encrypted or not, because when two equal values are encrypted using the same cryptosystem and encryption key, the resulting encrypted values are equal too. Thus, these weights are calculated in the data management server without the need of any encryption details or a participant who has the secret key.

3.3.1. Global Entropy

In this method, a weights vector is generated, which contains a weight for each attribute. To calculate the weight for an attribute, the weight of each category is calculated first. If the category (s_d) is in a dataset (tr) that contain a total of (N) tuples (X), which consist of (d) attributes and a class (y), distributed in (M) classes, each class is denoted by (c_m), then the global entropy of that category is calculated as,

$$H_G(s_d) = - \sum_{m=1}^M p(m|s_d) \log_2 p(m|s_d)$$

where,

$$p(m|s_d) = \frac{\sum_{(X,y) \in c_m} D_h(x_d, s_d, 1)}{\sum_{(X,y) \in tr} D_h(x_d, s_d, 1)}$$

and the weight of the attribute (d) using global entropy is,

$$\omega_d^{(H_G)} = e^{-\frac{1}{\log_2 M} \sum_{s_d \in S_d} p(s_d) \times H_G(s_d)}$$

where

$$p(s_d) = \frac{1}{N} \sum_{(X,y) \in t_r} D_h(x_d, s_d, 1)$$

3.3.2. Local Entropy

In this weighting method, a matrix with dimensions equal to the number of attributes by the number of classes is generated, where each attribute has a weight per each class. Each local entropy, of an attribute in a specific class is calculated using the equation,

$$H_L(m, d) = - \sum_{s_d \in S_d} p(m|s_d) \log_2 p(m|s_d)$$

where,

$$p(m|s_d) = \frac{1}{|c_m|} \sum_{(x,y) \in c_m} D_h(x_d, s_d, 1)$$

and the weight of the attribute (d) with respect to class (m) using local entropy is,

$$\omega_{md}^{(H_L)} = e^{-\frac{1}{\log_2 |S_d|} \times H_L(m,d)}$$

3.3.3. Global Gini Diversity Index

A weight vector is computed using the global Gini diversity index, where one weight is calculated per an attribute. The global Gini diversity factor is calculated using the following equation,

$$G_G(s_d) = 1 - \sum_{m=1}^M [p(m|s_d)]^2$$

where $p(m|s_d)$ is calculated using the same equation used to calculate the global entropy. The weight calculated using the global Gini diversity index for an attribute (d) is,

$$\omega_d^{(G_G)} = e^{-\frac{M}{M-1} \sum_{s_d \in S_d} p(s_d) \times G_G(s_d)}$$

3.3.4. Local Gini Diversity Index

This weighting method is used to calculate a weight for each attribute with respect to every class in the dataset. The weights, in the local Gini diversity index matrix, are computed using the equation,

$$G_L(m, d) = 1 - \sum_{s_d \in S_d} [p(m|s_d)]^2$$

using the same equation of $p(m|s_d)$ as used to calculate the local entropy. The weights are calculated, depending on the local Gini diversity index, using the formula

$$\omega_{md}^{(G_L)} = e^{-\frac{|S_d|}{|S_d|-1} \times G_L(m,d)}$$

3.4. Methodology

The proposed method is designed to work with databases that are encrypted using Paillier cryptosystems. This cryptosystem is an additive homomorphic cryptosystem, which means that the encrypted result of adding two encrypted number is obtained using only the public encryption key, without the need to decrypt these numbers using the secret key. The homomorphic addition equation is,

$$E_{pk}(x + y) \leftarrow E_{pk}(x) * E_{pk}(y) \bmod N^2 \quad (1)$$

where x and y are two numbers, and N is known from the public key.

Using equation (1), the result of multiplying any encrypted number to a known, non-encrypted, number can be calculated using the equation,

$$E_{pk}(x * y) \leftarrow E_{pk}(x)^y \bmod N^2 \quad (2)$$

3.4.1. Secure Multiplication (Sm)

The aim of the secure multiplication is to calculate the encrypted product of two encrypted numbers without revealing any of these numbers to any of the participants in the calculations. The participants are C_1 , which is the data management server, and C_2 ,

which is the client computer that had the secret key sk . The product of two encrypted numbers, $pk(x)$ and $pk(y)$, is required by the classifier in C_1 . As these numbers are supposed to be exposed to neither C_1 nor C_2 , two random numbers, r_x and r_y , are generated at C_1 and added to the encrypted numbers, $pk(x)$ and $pk(y)$, using equation (1), then, these new numbers are sent to C_2 to calculate their product. By adding random numbers that are only known to C_1 , the data information sent to C_2 does not expose the original values retrieved from the database. When the two new numbers are decrypted and multiplied at C_2 , the result is encrypted and sent back to C_1 .

The value received by C_1 is equal to $pk\left((x + r_x) * (y + r_y)\right)$, which can be used to calculate the required value $pk(x * y)$. Knowing that

$$x * y = (x + r_x) * (y + r_y) - x * r_y - y * r_x - r_x * r_y \quad (3)$$

where $(x * r_y)$ and $(y * r_x)$ can be calculated using equation (2), as the value of r_x and r_y are known to C_1 .

3.4.2. Secure Squared Euclidean Distance (SSED)

The aim of the SSED is to calculate the squared Euclidean distance between two tuples without exposing any information about the data in these tuples to any of the participants in the calculation process. This is achieved by calculating the squared distance between each attribute in the tuples using equation (1), then, the result is squared, or multiplied by itself, using the SM method. Eventually, the summation of these values is calculated using the equation (1), which results,

$$E_{pk}(D_e^2(r, s)) = \prod_{x=1}^n E_{pk}((r_x - s_x)^2) \quad (4)$$

where r and s are two tuples that contain n attributes.

Suppose an authorized client computer S_2 , which has the secret key to decrypt the data, sends an encrypted tuple (X) , which consists of (m) attributes, to the classifier in the shared data management server S_1 . The pseudo code of the algorithm that runs in S_1

and computes the squared Euclidean distances, with the corporation of S_2 , between this tuple and every tuple (Y) in the encrypted dataset, is shown below

Algorithm: SSED ($E_{pk}(X), E_{pk}(Y)$) \rightarrow ($E_{pk}(|X - Y|^2)$)

Require: S_1 has $E_{pk}(X)$ and $E_{pk}(Y)$; S_2 has sk .

1: S_1 , For $1 \leq i \leq m$ do:

$$E_{pk}(x_i - y_i) \leftarrow E_{pk}(x_i) * E_{pk}(y_i)^{N-1}$$

Generate two random numbers r_1 and r_2 .

Calculate $E1 = E_{pk}(x_i - y_i + r_1)$ and $E2 = E_{pk}(x_i - y_i + r_2)$ using homomorphic addition.

Calculate $r_1 * r_2$.

Calculate $E_{pk}(x_i - y_i) * r_1$ and $E_{pk}(x_i - y_i) * r_2$ using homomorphic multiplication.

S_1 , Send $E1$ and $E2$ to S_2 .

2: S_2 , Decrypts $E1$ and $E2$ into E_{d1} and E_{d2} using the secret key.

$$\text{Compute } E_{pk}(ME) = E_{pk}(E_{d1} * E_{d2})$$

Send $E_{pk}(ME)$ to S_1 .

3: S_1 , $E_{pk}(|X - Y|^2) = E_{pk}(ME) - r_1 * r_2 - E_{pk}(x_i - y_i) * r_1 - E_{pk}(x_i - y_i) * r_2$

3.4.3. Basic Classifier

The basic classifier, which is modified in this study, is proposed by **Samanthula et al.** [29]. This classifier is based on querying the (k) nearest neighbors depending on the distances depending on the Euclidean distance. As the classifier is designed to deal with encrypted data, the Euclidean distances are calculated using the homomorphic operations as well as the secure multiplication. Later, the secure minimum is used to query the nearest neighbors depending on the calculated distances.

3.4.4. Weighted Distance

The distances used in the proposed method are weighted distances, which are the products of the attribute weight and the SSED. Such distances are shown by **Hechenbichler and Schliep** [26] to produce better classification with the ordinal categorical attributes, which are categorical attributes where categories can be ordered. In other words, these attributes have categorical data that have distances from each other. This use of the weighted distances is very similar to the requirement of classifying an encrypted data where data types of the attributes are not disclosed to maximize the security, thus, the attributes are treated as numerical and categorical simultaneously.

As these weights are calculated in the data management server, it is possible to directly use equation (2) to multiply the squared value returned from C_2 before summing them. Thus, the weighted distance equation becomes,

$$E_{pk}(D_{\omega}^2(r, s)) = \prod_{x=1}^n E_{pk}((r_x - s_x)^2)^{\omega_{nm}} \quad (5)$$

where ω_{nm} is the weight of that attribute, or that attribute in the corresponding class, according to the used weighting method.

3.4.5. Nearest Neighbors Selection

After calculating the weighted distances, the tuples with minimum distances, from the new tuple being classified, are chosen using the Secure Minimum out of n Numbers (SMINn). As the method proposed by **Elmehdwi et al.** [24] is efficient and secure, and as the investigation of existing SMINn methods is not the aim of this study. This method is used to select the tuples with the least distances.

To ensure unbiased neighbors selection, when a neighbor is selected, all tuples that share the exact same distances as this neighbor are also selected regardless of the set number of nearest neighbors (k).

3.4.6. Class Prediction

After selecting the (k) nearest neighbors to the new tuple under investigation, the dominant class among these tuples is selected as a predicted class for the new tuple. Prediction confidence is a measure of how confident is the algorithm about the prediction it made. It is the ratio of the number of tuples in the selected neighbors, which are in the predicted class, to the total number of neighbors selected. Thus, a prediction of 100% (1) means that all selected neighbors fall in the same class.

In case there is more than one dominant class exist in the selected neighbors, which means that the max number of tuples per class is equal between two or more classes, the next nearest neighbor is also selected, until the tie is broken and only one dominant class exists among the selected neighbors.

CHAPTER FOUR

EXPERIMENTAL RESULTS

The modified classifiers are implemented for the test using C# programming language via visual studio development environment. Both client and server modules are implemented in the same project to minimize any network effect. All experiments are executed using a computer with Intel® Core™ i7-5200U @ 1.8GHz 2.7GHz and 8.00 GB of memory, which is running on Windows 7 operating system.

4.1. Benchmark Datasets

The proposed methods are evaluated using four UCI machine learning repository, **Lichman** [27], real life datasets. The selected datasets include one categorical, one numerical and two mixed datasets. Each dataset is encrypted using Paillier cryptosystem prior to any processing using the implemented test software.

4.1.1. Car Evaluation Dataset

The UCI's Car Evaluation dataset consists of 1728 tuples, which represent evaluated cars, described using six attributes. All these attributes are categorical; three of them have four categories while the other three have only three categories in each attribute with no missing values entire dataset. The tuples of this dataset are classified into four evaluations that are "unacc", "acc", "good" and "vgood". The "unacc" class includes the majority of the tuple, which are 1210 tuples, and the "acc" class has 384 tuples while the "good" class has 69 tuples, and the "vgood" class has 65 tuples in it.

4.1.2. Pima Indian Diabetes Dataset

The UCI's Pima Indian Diabetes dataset consists of 768 tuples, which represents diabetes patients, described using eight attributes. All these attributes are numerical; two attributes have continuous values while all other attributes have discrete values with no

missing values in the dataset. The patients in this dataset are classified into two classes, according to their test results for diabetes. The first class includes 500 patients who tested negative for diabetes while the remaining 268 patients tested positive.

4.1.3. Statlog (German Credit Data) Dataset

The UCI's Statlog (German Credit Data) dataset consists of 1000 tuples that represent evaluated bank customers, each contains 20 attributes. These attributes contain two types of data, which are numerical and categorical distributed as 13 categorical and 7 numerical attributes with no missing values anywhere in the dataset. The number of categories per categorical attribute differs from one attribute to another, while all numerical attributes are discrete. The tuples are classified into two classes, according to the credits evaluations, which are "Good" and "Bad". The "Good" class contains 700 customers while the "Bad" class contains 300.

4.1.4. Credit Approval Dataset

The UCI's Credit Approval dataset consists of 690 tuples that represent credit applicants, each tuple contains 15 attributes. There are six numerical attributes in this dataset while the remaining nine attributes are categorical. Sixty-seven missing values are found in 37 tuples. Tuples that have missing values are deleted from the dataset before any further processing, which means that only 653 tuples are remaining. The data is classified into two classes, according to the final decisions of the applications, which are positive or negative. After deleting the tuples with missing values, the remaining data is distributed as 296 tuples in the positive class and 357 tuples in the negative class.

4.2. Performance Measures

In order to evaluate the performance of the modified classifiers, two evaluation measures are calculated for each classification results. Every dataset is classified using the each of the modified classifiers, then, the classification results are compared to the actual classes that these tuples are in. This comparison is achieved using confusion matrices that are used to compute the performance measure, **Sokolova and Lapalme**

[28]. In this study, two classification performance measures are used, which are overall accuracy and F-score;

4.2.1. Overall Accuracy

The overall accuracy of a classifier is defined as the number of tuples, where the predicted classes match their actual classes, to the total number of tuples. Suppose a classifier is tested using a dataset that consists of (N) tuples. The classifier is able to classify (C) of these tuples correctly, while the remaining tuples are predicted to be in classes different than the classes they are actually in. The overall accuracy of the classifier is equal to (C/N). This measure describes the overall performance of the used method and is not affected by the actual distribution of the tuples in the classes.

4.2.2. F-Score

The F-score provides a better description to the distribution of the tuples with correctly predicted classes. To calculate the F-score of a class, the precision and recall for that class are calculated first. The precision of a class is the ratio of the correctly classified tuples in that class to the total number of tuples predicted to be in that class. Moreover, the recall of a class is the number of tuples predicted to be in that class to the total number of tuples that are actually in that class. For a dataset that has two classes, positive and negative, is classified using a classifier that predicted (TP) of the positive tuples to be positive, while (FN) of the positive tuples are predicted to be negative. For the negative tuples, (TN) tuples are predicted to be negative, while (FP) tuples are predicted to be positive. The recall for the positive class in this example is calculated using the equation,

$$recall = \frac{TP}{TP + FN}$$

while the precision for that class is

$$precision = \frac{TP}{TP + FP}$$

The F-score is then calculated for each class using the equation

$$F_{score} = \frac{2 * precision * recall}{(precision + recall)}$$

Finally, the average of the F-score of all classes is calculated as the classifier’s F-score.

4.3. Weighting Methods

The performance of the modified classifier is tested using the four attribute weighting methods discussed in chapter two. As the performance of a classifier may vary from one database to another, and in order to achieve unbiased evaluations, the classifier is tested using all the databases and the average performance of these databases is considered as the classifier’s measure using that method. Throughout all the experiments, the number of neighbors selected for classification (k) is set to five, as this value returned best results when tested against other values.

4.3.1. Global Entropy

Global entropy is used for attribute weighting in the classifier and evaluated using all the included datasets. The confusion matrix and performance measures of classifying the “Car Evaluation” are shown in Table 4.1.

Table 4.1: Cars classification confusion matrix using global entropy.

		Predicted				Recall	Precision	F-score
		unacc	acc	good	vgood			
Actual	unacc	1168	39	0	3	0.965289	0.986486	0.975773
	acc	16	348	8	12	0.90625	0.876574	0.891165
	good	0	4	55	6	0.846154	0.733333	0.785714
	vgood	0	6	12	51	0.73913	0.708333	0.723404
Overall F-score:								0.844014
Overall Accuracy:								0.938657

The Pima Indian Diabetes dataset is also used to test the performance of the classifier when global entropy is used for attribute weighting. The results of the classification are shown via the confusion matrix in Table 4.2.

Table 4.2: Pima Indian Diabetes classification confusion matrix using global entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	158	110	0.589552	0.655602	0.690378
	-	83	417	0.834	0.791271	0.770469
Overall F-score:						0.730423
Overall Accuracy:						0.748697

The confusion matrix for the Statlog (German Credit Data) dataset classification results, using the modified classifier with the global entropy for attribute weighting, is shown in Table 4.3.

Table 4.3: German Credit Card classification confusion matrix using global entropy.

		Predicted		Recall	Precision	F-score
		Good	Bad			
Actual	Good	619	81	0.884286	0.766089	0.711223
	Bad	189	111	0.37	0.578125	0.70503
Overall F-score:						0.708127
Overall Accuracy:						0.73

Finally, the Credit Card Approval dataset is used to test the method with the global entropy as attribute weight. The results of this evaluation are summarized in the confusion matrix shown in Table 4.4.

Table 4.4: Credit Card Approval dataset classification confusion matrix using global entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	255	41	0.861486	0.855705	0.852824
	-	43	314	0.879552	0.884507	0.886992
Overall F-score:						0.869908
Overall Accuracy:						0.871363

To demonstrate the overall performance of the classifier, using the global entropy as a weighting method, the results from each dataset are summarized in Table 4.5 to calculate the average F-score and average overall accuracy. These values are also illustrated in figure 4.1.

Table 4.5: Classifier's performance summary using global entropy.

Measure	Cars	Pima	German CC	Credit Approval	Average
F-score	0.844014	0.730423	0.708127	0.869908	0.788118
Accuracy	0.932657	0.744698	0.73	0.871363	0.81968

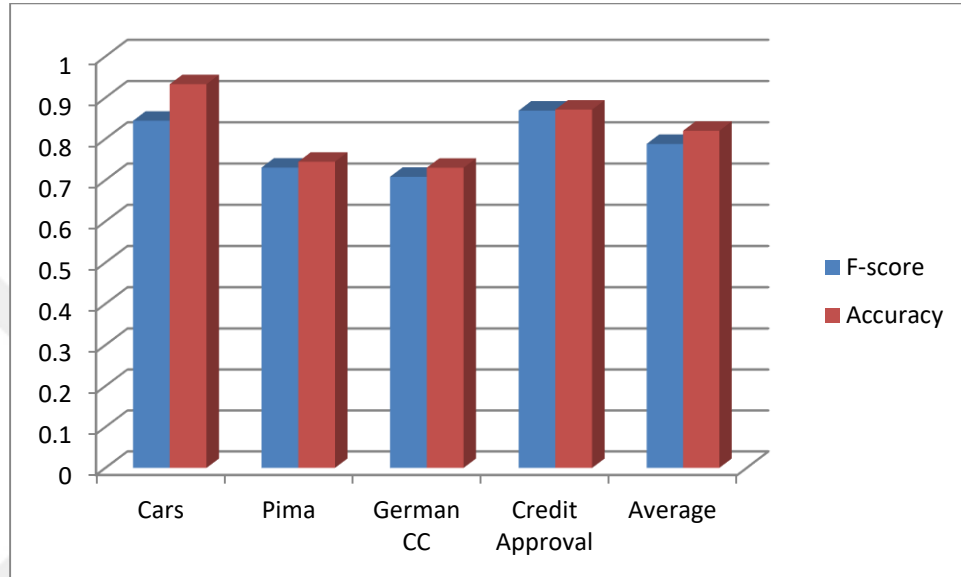


Figure 4.1: Illustration of classifier's performance using global entropy.

4.3.2. Local Entropy

In this section, the classifier is tested using all the included datasets while using the local entropy as attribute weighting method. The use of local entropy means that every attribute has different weight for each class. The first dataset used to test the classifier using the local entropy for attributes weighting is the Cars Evaluation dataset. The classification results are summarized in Table 4.6.

Table 4.6: Cars classification confusion matrix using local entropy.

		Predicted				Recall	Precision	F-score
		unacc	acc	good	vgood			
Actual	unacc	1199	11	0	0	0.990909	0.974005	0.982384
	acc	32	347	2	3	0.903646	0.930295	0.916777
	good	0	9	56	0	0.861538	0.8	0.82963
	vgood	0	6	12	51	0.73913	0.944444	0.829268
Overall F-score:								0.889515
Overall Accuracy:								0.956597

The Pima Indian Diabetes dataset is also used to test the performance of the classifier when the local entropy used to weight the distances. The results are shown in the confusion matrix in Table 4.7.

Table 4.7: Pima Indian Diabetes classification confusion matrix using local entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	147	121	0.548507	0.668182	0.733905
	-	73	427	0.854	0.779197	0.743509
Overall F-score:						0.738707
Overall Accuracy:						0.747396

Next, the Statlog (German Credit Data) dataset is classified using the modified classifier with the local entropy as attributes weighting method. Table 4.8 shows the confusion matrix for this test's classification results.

Table 4.8: German Credit Card classification confusion matrix using local entropy.

		Predicted		Recall	Precision	F-score
		Good	Bad			
Actual	Good	599	101	0.855714	0.750627	0.70152
	Bad	199	101	0.336667	0.5	0.59761
Overall F-score:						0.649565
Overall Accuracy:						0.7

The confusion matrix for the classification results of the Credit Card Approval dataset, using the local entropy as a weighting method for attributes distances, is shown in Table 4.9.

Table 4.9: Credit Card Approval dataset classification confusion matrix using local entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	260	36	0.878378	0.836013	0.815354
	-	51	306	0.857143	0.894737	0.913937
Overall F-score:						0.864645
Overall Accuracy:						0.866769

To evaluate the overall performance of the classifier using the local entropy as an attribute weighting method, the F-score and overall accuracy of the conducted tests using the included datasets are grouped in Table 4.10 and the average measures are calculated. These measures are illustrated in figure 4.2 as well.

Table 4.10: Classifier's performance summary using local entropy.

Measure	Cars	Pima	German CC	Credit Approval	Average
F-score	0.889515	0.738707	0.649565	0.864645	0.785608
Accuracy	0.956597	0.747396	0.7	0.866769	0.81769

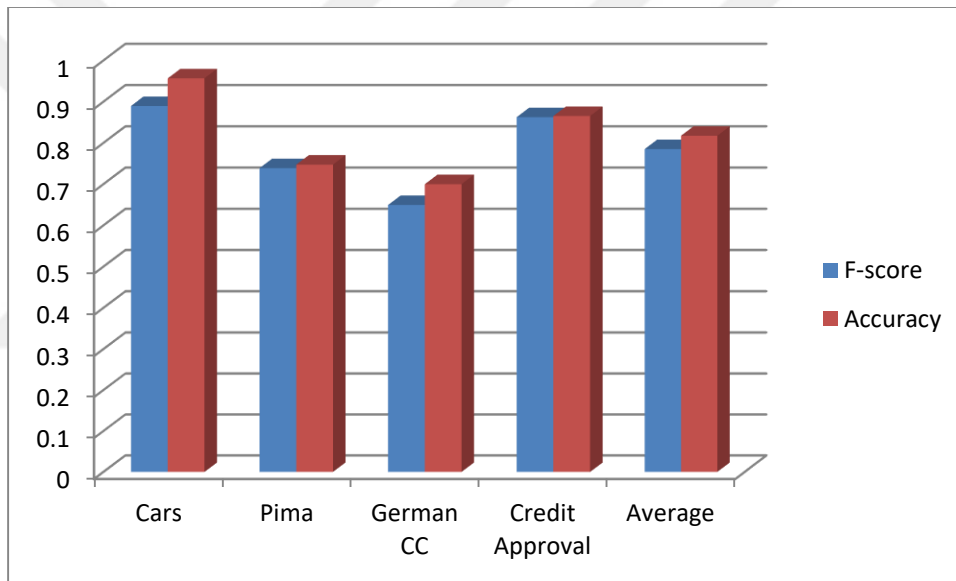


Figure 4.2: Illustration of classifier's performance using local entropy.

4.3.3. Global Gini Diversity Index

The performance of the classifier, when the global Gini diversity index is used as weighting method, is evaluated using all the datasets included in the experiments. The use of global Gini diversity index means that there is only one weight for each attribute throughout the entire dataset. First, the performance is evaluated using the Car Evaluation dataset. The results of this evaluation are summarized in the confusion matrix shown in Table 4.11.

Table 4.11: Cars classification confusion matrix using global Gini diversity index.

		Predicted				Recall	Precision	F-score
		unacc	acc	good	vgood			
Actual	unacc	1168	39	0	3	0.965289	0.986486	0.975773
	acc	16	348	8	12	0.90625	0.876574	0.891165
	good	0	4	55	6	0.846154	0.733333	0.785714
	vgood	0	6	12	51	0.73913	0.708333	0.723404
Overall F-score:								0.844014
Overall Accuracy:								0.938657

The Pima Indian Diabetes dataset is also used to test the performance of the classifier when global Gini diversity index is used for attribute weighting. The results of the classification are shown via the confusion matrix in Table 4.12.

Table 4.12: Pima Indian Diabetes classification confusion matrix using global Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	157	111	0.585821	0.654167	0.690223
	-	83	417	0.834	0.789773	0.768261
Overall F-score:						0.729242
Overall Accuracy:						0.747396

Next, the Statlog (German Credit Data) dataset is classified using the classifier and the global Gini diversity index as attributes weighting method. Table 4.13 shows the confusion matrix for this test's classification results.

Table 4.13: German Credit Card classification confusion matrix using global Gini diversity index.

		Predicted		Recall	Precision	F-score
		Good	Bad			
Actual	Good	614	86	0.877143	0.762733	0.709519
	Bad	191	109	0.363333	0.558974	0.677545
Overall F-score:						0.693532
Overall Accuracy:						0.723

The credit card approval dataset is also used to evaluate the performance of the classifier when weights are calculated using global Gini diversity index. The confusion matrix shown in Table 4.14 summarizes the classification results of this test.

Table 4.14: Credit Card Approval classification confusion matrix using global Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	251	45	0.847973	0.859589	0.865437
	-	41	316	0.885154	0.875346	0.87047
Overall F-score:						0.867953
Overall Accuracy:						0.8683

The overall performance of the classifier when global Gini diversity index is used for weighting is evaluated using F-score and accuracy by summarizing the performance of the classifier with every included dataset. The performance measures are summarized in Table 4.15, which is used to calculate the overall performance measure. These values are also illustrated in Figure 4.3.

Table 4.15: Classifier's performance summary using global Gini diversity index.

Measure	Cars	Pima	German CC	Credit Approval	Average
F-score	0.844014	0.729242	0.693532	0.87196	0.784687
Accuracy	0.938657	0.747396	0.723	0.872894	0.820487

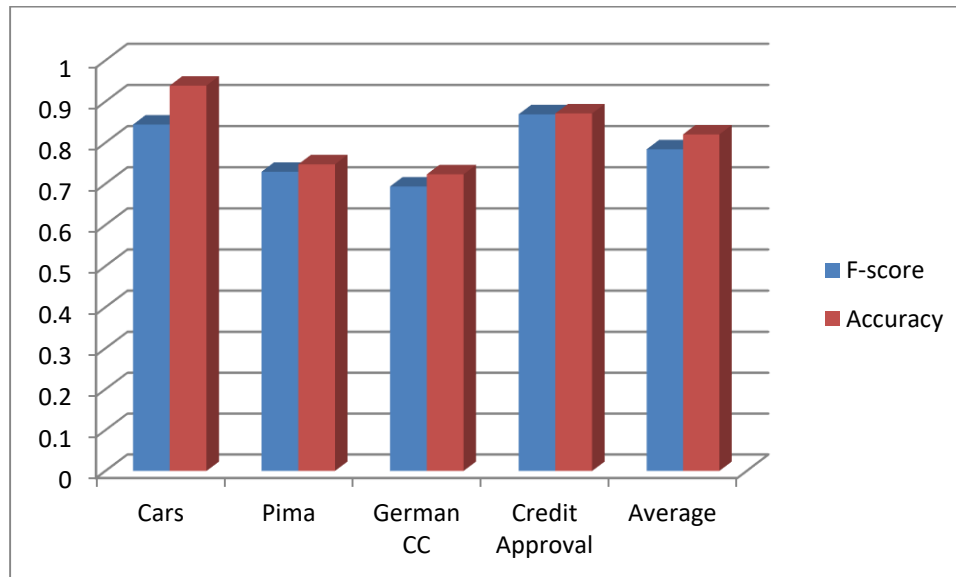


Figure 4.3: Illustration of classifier's performance using global Gini diversity index.

4.3.4. Local Gini Diversity Index

In this section, the performance of the classifier is evaluated when the local Gini diversity index is used as weighting method. The use of local Gini diversity factor requires a number of weights in each attribute equal to the number of classes so that the weight of that attribute in the corresponding class is used. The results of testing the performance of the classifier using the Car Evaluation dataset, while local Gini diversity index is used for weighting, are shown in Table 4.16.

Table 4.16: Cars classification confusion matrix using local Gini diversity index.

		Predicted				Recall	Precision	F-score
		unacc	acc	good	vgood			
Actual	unacc	1199	11	0	0	0.990909	0.974005	0.982384
	acc	32	347	2	3	0.903646	0.930295	0.916777
	good	0	9	56	0	0.861538	0.8	0.82963
	vgood	0	6	12	51	0.73913	0.944444	0.829268
Overall F-score:								0.889515
Overall Accuracy:								0.956597

The Pima Indian Diabetes dataset is also used to test the performance of the classifier when local Gini diversity index is used for attribute weighting. The results of the classification are shown via the confusion matrix in Table 4.17.

Table 4.17: Pima Indian Diabetes classification confusion matrix using local Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	154	114	0.574627	0.652542	0.693974
	-	82	418	0.836	0.785714	0.761351
Overall F-score:						0.727662
Overall Accuracy:						0.744792

The confusion matrix of the Statlog (German Credit Data) dataset classification results, when using local Gini diversity index for attributes weighting, is shown and evaluated in Table 4.18.

Table 4.18: German Credit Card classification confusion matrix using local Gini diversity index.

		Predicted		Recall	Precision	F-score
		Good	Bad			
Actual	Good	615	85	0.878571	0.754601	0.697321
	Bad	200	100	0.333333	0.540541	0.66871
Overall F-score:						0.683016
Overall Accuracy:						0.715

Finally, the performance of the classifier using the local Gini diversity index for attributes weighting is evaluated using the Credit Card Approval dataset. The confusion matrix of the classification results and the evaluation measures are shown in Table 4.19.

Table 4.19: Credit Card Approval classification confusion matrix using local Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	256	40	0.864865	0.842105	0.830877
	-	48	309	0.865546	0.885387	0.89542
Overall F-score:						0.863148
Overall Accuracy:						0.865237

The overall performance of the classifier when local Gini diversity index is used for attributes weighting is summarized and evaluated, using the performance measures F-score and accuracy, in Table 4.20. These measures are also illustrated graphically in figure 4.4.

Table 4.20: Classifier's performance summary using local Gini diversity index.

Measure	Cars	Pima	German CC	Credit Approval	Average
F-score	0.889515	0.727662	0.683016	0.863148	0.790835
Accuracy	0.956597	0.746792	0.715	0.865237	0.820907

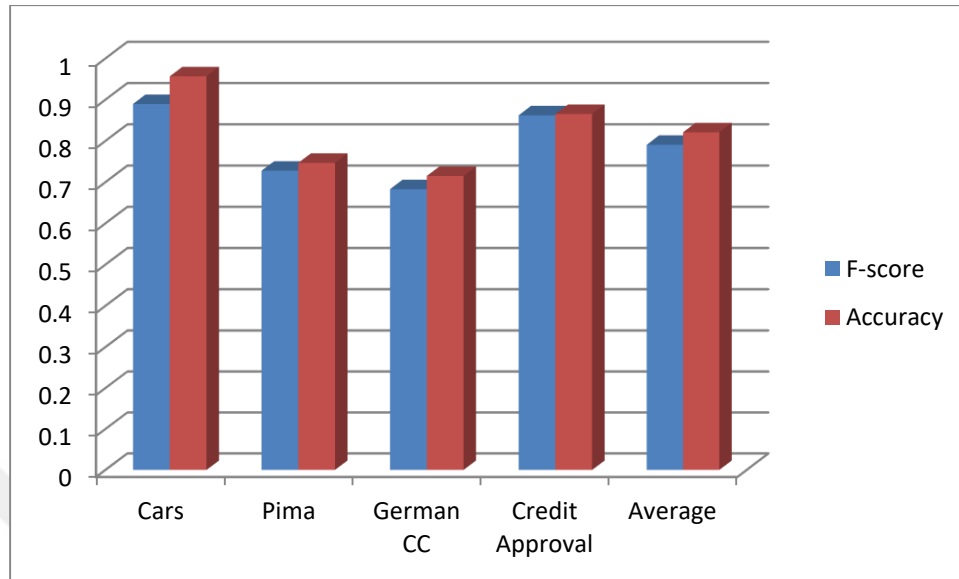


Figure 4.4: Illustration of classifier's performance using local Gini diversity index.

4.4. Basic Method

For comparison purposes, the datasets included in the experiments are classified using the basic classifier proposed by **Samanthula et al.** [29], which find the nearest neighbors depending on the squared Euclidean distance. The summary of the classification results for the Car Evaluation dataset is shown in the confusion matrix alongside with the performance measure calculated using this matrix in Table 4.21.

Table 4.21: Cars classification confusion matrix using the basic classifier.

		Predicted				Recall	Precision	F-score
		unacc	acc	good	vgood			
Actual	unacc	1123	87	0	0	0.928099	0.903459	0.915614
	acc	119	242	13	10	0.630208	0.52381	0.572104
	good	1	64	0	0	0	0	0
	vgood	0	69	0	0	0	0	0
Overall F-score:								0.371929
Overall Accuracy:								0.789931

The performance of the basic classifier is also evaluated using the Pima Indian Diabetes dataset. The confusion matrix of the classification results alongside with the performance measures is shown in Table 4.22.

Table 4.22: Pima Indian Diabetes classification confusion matrix using the basic classifier.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	0.570896	0.64557	0.685199	0.570896	0.64557
	-	0.832	0.783427	0.759871	0.832	0.783427
Overall F-score:						0.722535
Overall Accuracy:						0.740885

The Statlog (German Credit Data) dataset is also used to evaluate the performance of the basic method. Table 4.23 shows the confusion matrix of the classification results and the performance measure calculated for these results.

Table 4.23: German Credit Card classification confusion matrix using the basic classifier.

		Predicted		Recall	Precision	F-score
		Good	Bad			
Actual	Good	585	115	0.835714	0.72401	0.672158
	Bad	223	77	0.256667	0.401042	0.489075
Overall F-score:						0.580616
Overall Accuracy:						0.662

Finally, the Credit Approval dataset is classified using the basic classifier. The performance measure for the classifier and the confusion matrix used to calculate these measures from classification results are shown in Table 4.24.

Table 4.24: Credit Card Approval classification confusion matrix using the basic classifier.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	248	48	0.837838	0.846416	0.850728
	-	45	312	0.87395	0.866667	0.86304
Overall F-score:						0.856884
Overall Accuracy:						0.85758

The summary of overall performance measures for each dataset used to evaluate the basic performance of the basic classifier are shown in Table 4.25. These measures are also illustrated in figure 4.5.

Table 4.25: Basic classifier's performance summary.

Measure	Cars	Pima	German CC	Credit Approval	Average
F-score	0.371929	0.722535	0.580616	0.856884	0.632991
Accuracy	0.789931	0.740885	0.662	0.85758	0.762599

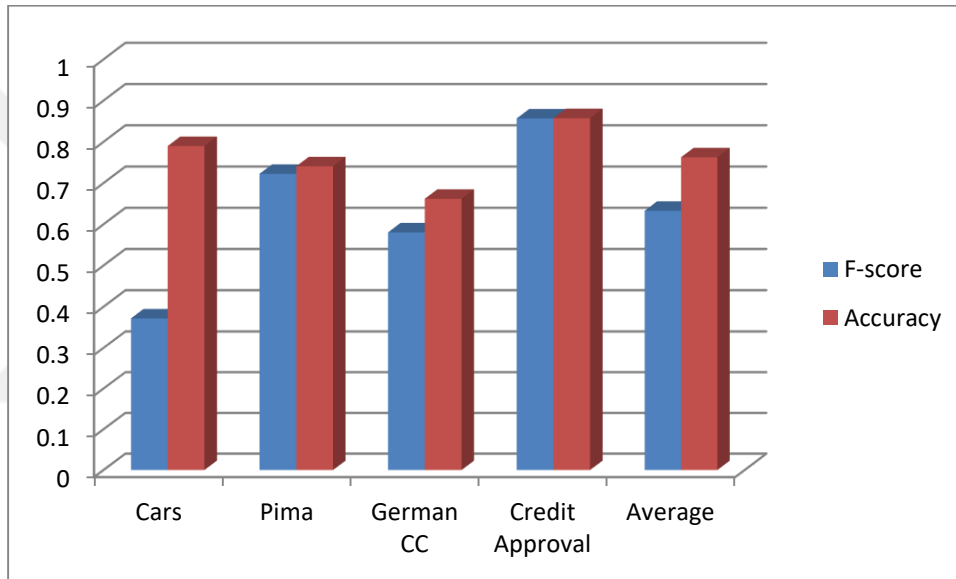


Figure 4.5: Illustration of basic classifier's performance.

4.5. Plain Pima Indian Diabetes Dataset

In order to verify that the investigated classifiers return the exact same results from an encrypted dataset as if these methods are used on the same dataset but in a plain text form, the Pima Indian Diabetes dataset is tested as is without any encryption using the same classification methods but without the need to the homomorphic or secure calculations. The weights and distances are calculated directly. The results of classifying the plain Pima Indian Diabetes dataset using the global entropy for attribute weighting are shown in the confusion matrix in Table 4.26.

Table 4.26: Plain Pima Indian Diabetes classification results using global entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	158	110	0.589552	0.655602	0.690378
	-	83	417	0.834	0.791271	0.770469
Overall F-score:						0.730423
Overall Accuracy:						0.748697

The plain Pima Indian Diabetes dataset is also classified using the local entropy as an attribute weighting method. The results of this classification are distributed in the confusion matrix shown in Table 4.27.

Table 4.27: Plain Pima Indian Diabetes classification results using local entropy.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	147	121	0.548507	0.668182	0.733905
	-	73	427	0.854	0.779197	0.743509
Overall F-score:						0.738707
Overall Accuracy:						0.747396

Table 4.28 shows the confusion matrix that resulted from classifying the plain Pima Indian Diabetes dataset using the global Gini diversity index as attribute weighting method.

Table 4.28: Plain Pima Indian Diabetes classification results using global Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	157	111	0.585821	0.654167	0.690223
	-	83	417	0.834	0.789773	0.768261
Overall F-score:						0.729242
Overall Accuracy:						0.747396

The classification results of classifying the plain Pima Indian Diabetes dataset, using the local Gini diversity index as a weighting method for the attributes, are shown in the confusion matrix shown in Table 4.29.

Table 4.29: Plain Pima Indian Diabetes classification results using local Gini diversity index.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	154	114	0.574627	0.652542	0.693974
	-	82	418	0.836	0.785714	0.761351
Overall F-score:						0.727662
Overall Accuracy:						0.744792

Finally, the plain Pima Indian Diabetes dataset is classified using the basic classifier, which does not compute the weights of the attributes when the distance between tuples is calculated. The classification results are summarized in the confusion matrix shown in Table 4.30.

Table 4.30: Plain Pima Indian Diabetes classification results using the basic classifier.

		Predicted		Recall	Precision	F-score
		+	-			
Actual	+	0.570896	0.64557	0.685199	0.570896	0.64557
	-	0.832	0.783427	0.759871	0.832	0.783427
Overall F-score:						0.722535
Overall Accuracy:						0.740885

CHAPTER FIVE

DISCUSSION

In digital systems, all information is stored as ones and zeros, which is known as the binary system. Thus, it is mandatory to have a description for that information in order to display that information appropriately. This is the reason, for example, that data type must be set to every attribute when a new data table is created. This specifies to which form the data is converted prior to any further processing. If the data type is not specified, then it is mostly treated the way the data are stored, which is numbers. When data is encrypted, and data type is not declared, it becomes impossible to know the original data type of that information. Thus, all encrypted data may be processed as numbers, regardless of the fact that this information is originally numbers or not.

The performance of the basic classifier, which is proposed by **Samantula et al.** [29] based on the secure nearest neighbor query proposed by **Elmehdwi et al.** [24], is demonstrated in section 4.4, where table 4.25 and figure 4.5 show the good performance of this classifier with the Pima Indian Diabetes dataset, which is a numerical dataset. The classifier's performance starts descending as data contain more categorical attributes until it reaches the minimum performance at the Cars Evaluation dataset, which is a categorical dataset. This behavior of the classifier enforces the data owner to accept the low performance of the classifier or reveal the data structure to the shared data management server, which may pose a threat to the data security.

Even in a plain text dataset, which is a non-encrypted dataset, most of the classifiers that are designed to work with numerical data, such as the k -NN classifier, are unable to process categorical data, or have very low performance when used with these data. Thus, many methods are proposed to enhance the performance of such classifiers when used with categorical data. One of the best methods used for that purpose is attribute weighting, which is based on many weighting techniques as shown by **Chen and Guo** [12]. These weighting techniques are combined with the basic squared Euclidean

distance used by the basic method and tested over different datasets. The performance of the modified classifiers, over those datasets, is summarized in Table 5.1.

Table 5.1: Performance summary of the classifiers.

	Global Entropy		Local Entropy		Global Gini		Local Gini		Basic Classifier	
	Fscore	Acc.	Fscore	Acc.	Fscore	Acc.	Fscore	Acc.	Fscore	Acc.
Cars	0.8440	0.9327	0.8895	0.9566	0.8440	0.9387	0.8895	0.9566	0.3719	0.7899
Pima	0.7304	0.7447	0.7387	0.7474	0.7292	0.7474	0.7277	0.7468	0.7225	0.7408
GCC	0.7081	0.7300	0.6496	0.7000	0.6935	0.7230	0.6830	0.7150	0.5806	0.662
CRX	0.8699	0.8714	0.8646	0.8668	0.8720	0.8729	0.8631	0.8652	0.8568	0.8575
Average	0.7881	0.8197	0.7856	0.8177	0.7847	0.8205	0.7908	0.8209	0.6330	0.7626

For a better illustration of classifiers' performance, the performance measures of each classifier per each data are visualized in figure 5.1.

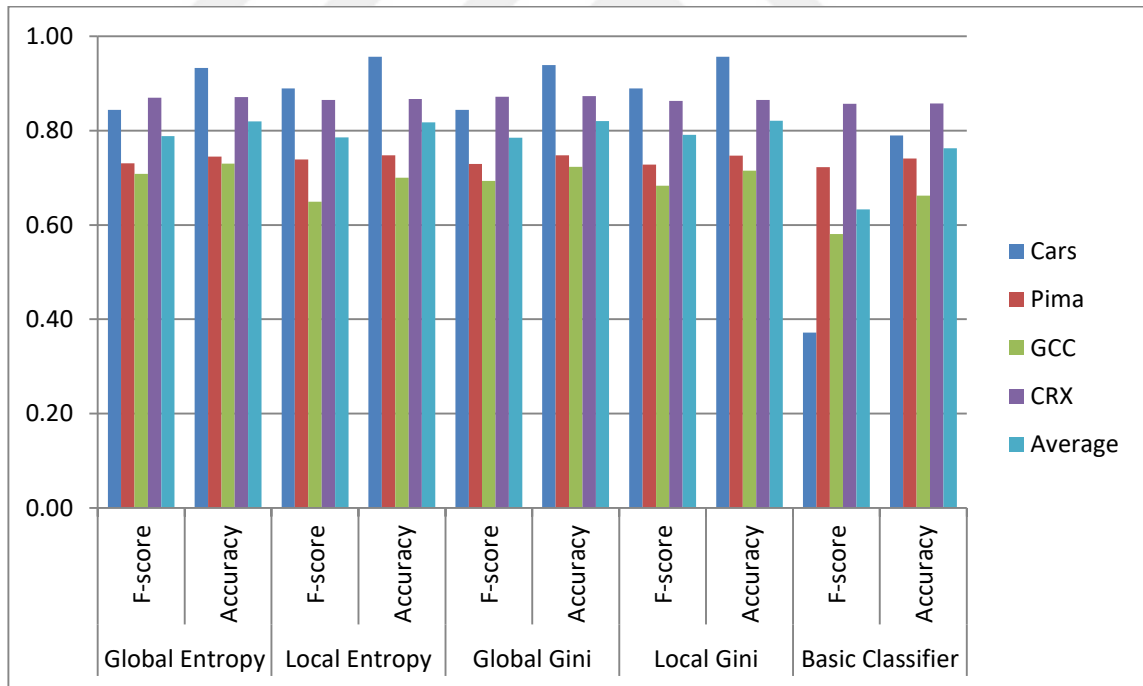


Figure 5.1: Performance summary of the classifiers.

The illustration in figure 5.1 shows the close performance of the modified classifiers when used for each dataset. The performance of the basic classifier, on the other hand, is affected by the number of categorical attributes in the dataset. The higher the number of categorical attributes, the less the classification performance. This situation is well illustrated by the difference in performance between the modified and basic classifiers tested using the Car Evaluation dataset, which contains only categorical dataset, and the Pima Indian Diabetes dataset, which contains numerical data only. The variation in classifiers' performance is minimal when all attributes are numerical, while the performance of the basic classifier is extremely low, compared to the modified classifiers, when all the attributes are categorical. This gap decreases as the ratio of categorical attributes to the total number of attributes decreases, when data with mixed data types are used for testing.

Another important illustration is shown in figure 5.2, where the F-score of all classifiers per a dataset is demonstrated. This illustration shows that the performance of the modified classifiers are very close to each other on all dataset and are always performing better than the basic classifier no matter what data types does the database contain. Even when the classifiers are tested using the numerical dataset, Pima Indian Diabetes, where the basic classifier is expected to have good performance, it is slightly outperformed by the modified classifiers.

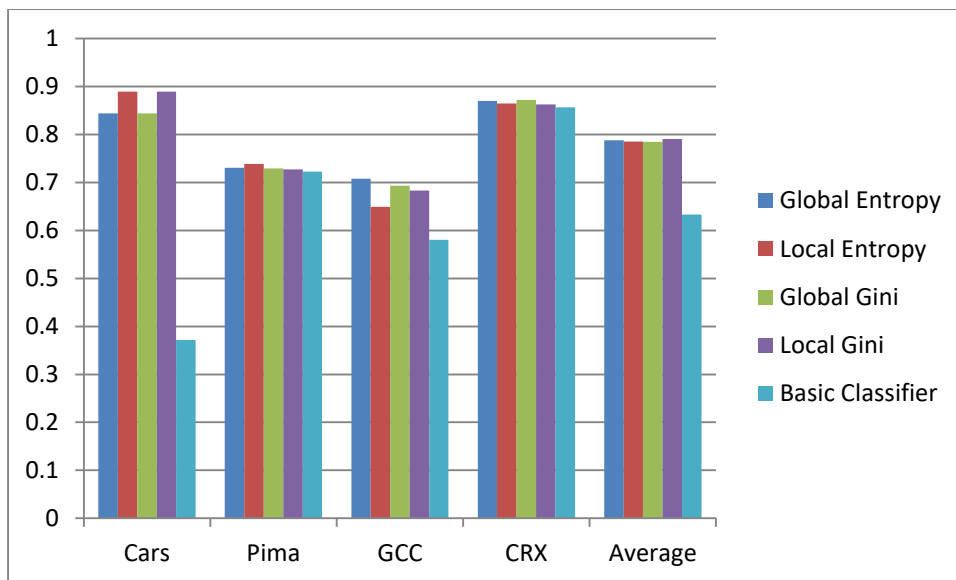


Figure 5.2: F-score of each classifier per dataset.

The complexity of the modified classifiers is also compared to the basic classifier using the execution time as a complexity measure. Execution times for the conducted experiments are summarized in Table 5.2.

Table 5.2: Execution time of experiments in seconds.

Dataset	Global Entropy	Local Entropy	Global Gini	Local Gini	Basic Classifier
Cars	465.145	496.786	444.715	442.976	418.154
Pima	119.482	119.134	117.382	117.64	109.015
GCC	511.426	499.532	491.557	497.589	475.042
CRX	175.701	175.122	174.133	180	150.7
Average	317.9385	322.6435	306.9468	309.5513	288.2278

Figure 5.3 shows the visual illustration of time consumed by every classifier to classify the included datasets.

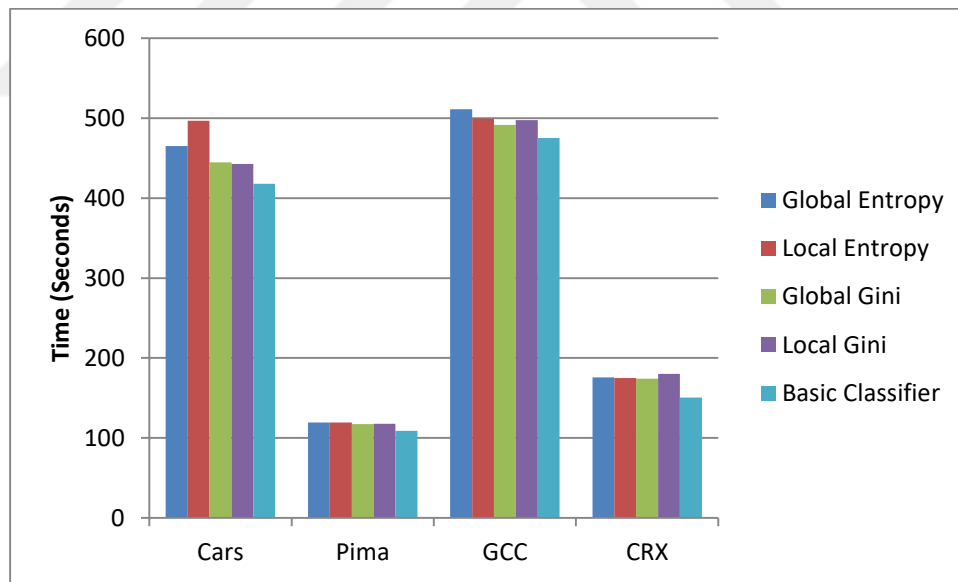


Figure 5.3: Execution time for experiments in seconds.

The average extra time consumed by the modified classifiers to calculate the attributes weights and use these weights with every Euclidean distance calculated, during the nearest neighbors seeking process, are only (9%) of the total time consumed by the basic classifier to complete the exact same task. Keeping in mind that the

experiments are conducted on the same computer, which eliminated network latency that delays the Secure Multiplication, as the data needs to travel to the client who has the secret key, processed, and then travel back to the data management server. This increases the execution time for the basic classifier as well as the modified ones, but the extra time required for weights calculation and distances weighting is not affected by the network latency, as these calculation are computed entirely in the data management server. Thus, the percentage of the extra time consumed by the modified classifiers is less in the real world than in experiments.

The results of the experiments conducted in section 4.5, which are the classification results of the Pima Indian Diabetes dataset using the same methods applied directly on the plain data before encryption, show that the use of homomorphic and secure calculation enabled executing the same methods and acquiring the same results without the need to reveal any information about the dataset or any values stored in that dataset to any of the participants of the calculations tasks.

CHAPTER SIX

CONCLUSION

Classification is one of the most important data mining techniques that is used in many fields to find patterns and relations between the attributes of a database from one side and the classification of the tuples in that database from another side. This extracted knowledge can be used later to classify any new tuples by looking into the values of their attributes, then a class is predicted from that tuple, which enables predicting the behavior of that tuple in the future depending on the behavior of the tuples, which are already in the predicted class.

The importance of a database for any corporation is the key to success for that corporation. Thus, storing and maintaining all available data is mandatory for these corporations. This leads to the need of data management servers with huge storage capacity, high availability and reliability, which require significant financial and technical support. Shared servers, on the other hand, provide the solutions for these concerns, which require attention on other concerns regarding outsourcing sensitive information to a server that is not run by the corporation. To protect these data, it is important to encrypt them before outsourcing to the shared data management server.

Encrypted data are still needed to be classified without revealing any information about these data to any of the participants, unless these data are targeted to them. In other words, the data management server must be able to classify an encrypted dataset without knowing any of their contents or access patterns. To achieve that, homomorphic encryption is used. A cryptosystem is said to be homomorphic when it is possible to perform a mathematical operation on the encrypted data and get the encrypted result without the need to decrypt the data or know the decryption key.

k -NN classifier is widely used because of its reliability and flexibility according to the fact that the neighbors are always extracted from the most recent dataset. Thus, it does not require any further training when new data are added to the database. The methods used to calculate the distances between the tuples in order to choose the nearest

neighbors based on these distances requires knowing the type of the data and their values. This conflicts with the principle of data encryption, especially when the honesty of the owner of the shared data management server is questionable. The currently existing k -NN classifiers securely compute the distances depending on the encrypted values stored in the database.

In this study, the basic k -NN classifier is modified to improve its performance over different databases that contain different data types. The modification is based on adjusting the distances calculated depending on the encrypted values, as in the basic classifier, by multiplying it by the weight of the attribute, which represents the effect of that attribute on the actual classification of the tuples in the database. These weights are calculated in the data management server and do not require any knowledge about the values stored in the database. There are many methods proposed for attribute weighting. In this study, four methods are used based on two weighting methods, which are information entropy and Gini diversity index, each is used in two ways which are local and global. The local weighting methods generate one weight for each attribute, while the local weighting methods generate one weight per class for each attribute. In other words, the global methods represent the effect of the attribute on the actual classification, while the local methods represent the effect of each attribute in each class.

To test the performance of the modified classifiers, four datasets are used to measure the performance, of every classifier, using two performance measures that are accuracy and F-score. The datasets used in the experiments are real life UCI datasets and the test results show good performance of the modified classifiers. Although the classifier that uses the local Gini diversity index for attribute weighting is the classifier with the maximum average performance, all modified classifiers show very close performances to each other. All modified classifiers outperformed the basic classifier in every experiment conducted during this study. The modified classifiers do not require much extra execution time, because the weights are calculated by the data management server, thus, homomorphic multiplication may be applied directly.

In future work, to simplify the complexity of the classifier, we suggest setting up a weight threshold so that attributes with weights less than the threshold are not measured.

REFERENCES

- [1] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, *From data mining to knowledge discovery in databases*. AI magazine, 1996. 17(3): p. 37.
- [2] Agrawal, R., M. Mehta, J.C. Shafer, R. Srikant, A. Arning, and T. Bollinger. *The Quest Data Mining System*. in *KDD*. 1996.
- [3] Sabhnani, M. and G. Serpen. *Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context*. in *MLMTA*. 2003.
- [4] Sinha, A.P. and H. Zhao, *Incorporating domain knowledge into data mining classifiers: An application in indirect lending*. Decision Support Systems, 2008. 46(1): p. 287-299.
- [5] Aha, D.W., D. Kibler, and M.K. Albert, *Instance-based learning algorithms*. Machine learning, 1991. 6(1): p. 37-66.
- [6] Peterson, L.E., *K-nearest neighbor*. Scholarpedia, 2009. 4(2): p. 1883.
- [7] Deza, M.M. and E. Deza, *Encyclopedia of distances*, in *Encyclopedia of Distances*. 2009, Springer. p. 1-583.
- [8] Uhlmann, J.K., *Satisfying general proximity/similarity queries with metric trees*. Information processing letters, 1991. 40(4): p. 175-179.
- [9] Boriah, S., V. Chandola, and V. Kumar. *Similarity measures for categorical data: A comparative evaluation*. in *Proceedings of the 2008 SIAM International Conference on Data Mining*. 2008. SIAM.
- [10] Hamming, R.W., *Error detecting and error correcting codes*. Bell Labs Technical Journal, 1950. 29(2): p. 147-160.
- [11] Li, H., G. Wen, Z. Yu, and T. Zhou, *Random subspace evidence classifier*. Neurocomputing, 2013. 110: p. 62-69.
- [12] Chen, L. and G. Guo, *Nearest neighbor classification of categorical data by attributes weighting*. Expert Systems with Applications, 2015. 42(6): p. 3142-3149.
- [13] Shannon, C.E., *A mathematical theory of communication*. ACM SIGMOBILE Mobile Computing and Communications Review, 1948. 5(1): p. 3-55.
- [14] Gini, C., *Variabilità e mutabilità*. Reprinted in *Memorie di metodologica statistica* (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi, 1912.

- [15] Daelemans, W., J. Zavrel, K. van der Sloot, and A. Van den Bosch, *Timbl: Tilburg memory-based learner*. Tilburg University, 2004.
- [16] Dong, C., G. Russello, and N. Dulay, *Shared and searchable encrypted data for untrusted servers*. *Journal of Computer Security*, 2011. 19(3): p. 367-397.
- [17] Gibson, G.A., D.F. Nagle, K. Amiri, J. Butler, F.W. Chang, H. Gobiuff, C. Hardin, E. Riedel, D. Rochberg, and J. Zelenka. *A cost-effective, high-bandwidth storage architecture*. in *ACM SIGOPS operating systems review*. 1998. ACM.
- [18] Li, M., S. Yu, W. Lou, and Y.T. Hou. *Toward privacy-assured cloud data services with flexible search functionalities*. in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. 2012. IEEE.
- [19] Rivest, R.L., L. Adleman, and M.L. Dertouzos, *On data banks and privacy homomorphisms*. *Foundations of secure computation*, 1978. 4(11): p. 169-180.
- [20] Fontaine, C. and F. Galand, *A survey of homomorphic encryption for nonspecialists*. *EURASIP Journal on Information Security*, 2007. 2007(1): p. 013801.
- [21] Williams, P., R. Sion, and B. Carbunar. *Building castles out of mud: practical access pattern privacy and correctness on untrusted storage*. in *Proceedings of the 15th ACM conference on Computer and communications security*. 2008. ACM.
- [22] Zhu, Y., R. Xu, and T. Takagi. *Secure k-NN computation on encrypted cloud data without sharing key with query users*. in *Proceedings of the 2013 international workshop on Security in cloud computing*. 2013. ACM.
- [23] Yao, B., F. Li, and X. Xiao. *Secure nearest neighbor revisited*. in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. 2013. IEEE.
- [24] Elmehdwi, Y., B.K. Samanthula, and W. Jiang. *Secure k-nearest neighbor query over encrypted data in outsourced environments*. in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. 2014. IEEE.
- [25] Paillier, P. *Public-key cryptosystems based on composite degree residuosity classes*. in *Eurocrypt*. 1999. Springer.
- [26] Hechenbichler, K. and K. Schliep, *Weighted k-nearest-neighbor techniques and ordinal classification*. 2004.
- [27] Lichman, M., *UCI machine learning repository*. 2013.
- [28] Sokolova, M. and G. Lapalme, *A systematic analysis of performance measures for classification tasks*. *Information Processing & Management*, 2009. 45(4): p. 427-437.

- [29] Samanthula, B.K., Y. Elmehdwi, and W. Jiang, *K-nearest neighbor classification over semantically secure encrypted relational data*. IEEE transactions on Knowledge and data engineering, 2015. 27(5): p. 1261-1273.



CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name : Ali Abbas Younis AL-ARBO
Date and Place of Birth : 04.05.1983 Iraq
Phone : +9647701613180 ; +905398888183
Email : ali_bayati83@yahoo.com

EDUCATION

<u>Degree</u>	<u>Institution</u>	<u>Year of Graduation</u>
High School	Abdul-Rahman Al Ghafiki	2001
Bachelor Degree	Informational Technology	2006
M.Sc.	Informational Technology	2017

WORK EXPERIENCE

<u>Year</u>	<u>Place</u>	<u>Enrollment</u>
2008-2014	Iraqi Ministry of Higher Education and Scientific Research	Programmer

LANGUAGES

<u>Language</u>	<u>Speaking</u>	<u>Reading</u>	<u>Writing</u>	<u>Listening</u>
<u>Arabic</u>	Excellent	Excellent	Excellent	Excellent
<u>English</u>	Good	Good	Good	Good
<u>Turkish</u>	Excellent	Excellent	Excellent	Excellent