



UNIVERSITY OF MACEDONIA

SCHOOL OF INFORMATION SCIENCES, DEPARTMENT OF APPLIED INFORMATICS
GRADUATE PROGRAM IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS (AIDA)

CLASSIFICATION ON MULTI-LABEL DATA WITH ORDERED LABELS

MASTER THESIS

of

ANTONIOS KAGIAS



Supervisor: Georgios Evangelidis
Professor

Thessaloniki, August 2024



UNIVERSITY OF MACEDONIA

SCHOOL OF INFORMATION SCIENCES, DEPARTMENT OF APPLIED INFORMATICS

GRADUATE PROGRAM IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS (AIDA)

CLASSIFICATION ON MULTI-LABEL DATA WITH ORDERED LABELS

MASTER THESIS

of

ANTONIOS KAGIAS

Supervisor: Georgios Evangelidis

Professor

Approved by the examination committee on dd/mm/2024.

(Signature)

(Signature)

(Signature)

.....

Georgios Evangelidis

Professor

.....

Georgia Koloniari

Assistant Professor

.....

Eftychios Protopapadakis

Assistant Professor

Thessaloniki, August 2024



UNIVERSITY OF MACEDONIA
SCHOOL OF INFORMATION SCIENCES, DEPARTMENT OF APPLIED INFORMATICS
GRADUATE PROGRAM IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS (AIDA)

Copyright © – All rights reserved.

Antonios Kagias, 2024.

The copying, storage and distribution of this master thesis, all or part of it, is prohibited for commercial purposes. Reprinting, storage and distribution for non - profit, educational or of a research nature is allowed, provided that the source is indicated and that this message is retained.

The content of this thesis does not necessarily reflect the views of the Department, the Supervisor, or the committee that approved it.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....

Antonios Kagias

dd/mm/2024

Abstract

Abstract.

Keywords

multi-label data, data mining, data classification, multi-label classification

to my family

Acknowledgements

This thesis is a result of research conducted within the "MSc in Artificial Intelligence and Data Analytics" of the Department of Applied Informatics of the University of Macedonia.

In the context of this thesis, the paper titled **[paper title]** was published at **[conference name]**. We thank University of Macedonia Research Committee for the funding.

Thessaloniki, August 2024

Antonios Kagias

Table of Contents

Abstract	1
Acknowledgements	5
Preface	13
1 Introduction	15
1.1 Problem - Importance of the topic	15
1.2 Aim - Objectives	15
1.3 Structure of the study	15
2 Literature review	17
3 BRkNN: a review	19
4 The proposed algorithms	21
4.1 Majority vote per position	21
4.2 Influenced string	22
4.3 Right-shifting	23
4.4 Labels like medals	25
5 Performance evaluation	27
5.1 Datasets	27
5.2 Experimental setup	28
5.2.1 Jaro-Winkler distance	28
5.2.2 Hamming loss	30
5.2.3 Ranked Hamming loss	30
5.2.4 Precision, recall, accuracy, F1 score	31
5.3 Results	33
6 Conclusion	41
6.1 Summary	41
6.2 Limitations	41
6.3 Future work	42
Bibliography	45

List of Figures

5.1	Metrics for 'emotions' dataset	37
5.2	Metrics for 'scene' dataset	37
5.3	Metrics for 'yeast' dataset	38
5.4	Metrics for 'birds' dataset	38
5.5	Metrics for 'image' dataset	38
5.6	Metrics for 'water-quality' dataset	38
5.7	Metrics for 'CHD49' dataset	38
5.8	Metrics for 'CAL500' dataset	38
5.9	Metrics for 'mediamill' dataset	39

List of Tables

4.1	Example instances for <i>clf1</i>	22
4.2	Occurrences of each label in every position	22
4.3	Example instances for <i>clf2</i>	23
4.4	Instances examined from <i>clf2</i> after 1st label is chosen	23
4.5	Instances examined from <i>clf2</i> after 2nd label is chosen	23
4.6	Initial state of example instances before the start of <i>clf3</i>	24
4.7	State of example instances for <i>clf3</i> after the 1st label is chosen	24
4.8	State of example instances for <i>clf3</i> after the 2nd label is chosen	25
4.9	Occurrences of each label in every position	25
4.10	Classification medal table for <i>clf4</i> algorithm	26
5.1	Datasets characteristics	27
5.2	The two strings and their common characters	29
5.3	Order of the common characters	30
5.4	Ranked Hamming loss: first array	30
5.5	Ranked Hamming loss: second array	31
5.6	Example arrays for ranked Hamming loss	31
5.7	Example confusion matrix	33
5.8	Precision, recall, accuracy and F1 score values	35
5.9	Jaro-Winkler distance, Hamming loss and ranked hamming Loss values	36

Preface

This thesis, entitled "Classification on multi-label data with ordered labels", is the product of my dedication over the last two years. My interest and love for data processing, analysis and visualization led me to choose the master's degree on Artificial Intelligence and Data Analytics to develop any existing knowledge as well as to learn more about the field. While the journey up to this point has not been easy, I am proud to have reached the end of it filled with new skills, experiences and resources for the future. Seeing that difficulties and challenges are part of the process taught me invaluable life and career lessons.

I would like to thank my supervisor, Georgios Evangelidis, for his guidance and support throughout the process of this thesis and the postgraduate programme as a whole. His suggestion to work on a particularly interesting topic as well as his comments, guidance and motivation were among the main reasons why this research took shape.

I would also like to thank all the other professors in the master's program for their interest in helping me learn new concepts on data analysis and for providing all kinds of resources during these two years.

Finally, I would like to thank my family for their endless support and motivation to become the best version of myself. Also, my friends for their support and company whenever I needed them the most.

I deeply hope that the reader will appreciate my work and find it interesting and enjoyable.

Sincerely,

Antonios Kagias, 2024

Chapter 1

Introduction

1.1 Problem - Importance of the topic

Multi-class classification is one of the most popular classification tasks where, given a set of unique labels, an instance can be assigned only one label. For example, if we have a set of images of animals where the possible classes are dog, cat and frog, an unlabeled image can only be classified as image of a dog, a cat or a frog. Multi-label classification and multi-label ranking are concepts closely related to the above. According to the analysis of [1], multi-label classification involves associating an instance with a subset of the possible labels instead of a single one. It is frequently used in order to classify diverse forms of data, such as songs or images [2]. For example, a song that is assigned “jazz, blues” belongs to both genres at the same time. Moreover, multi-label ranking not only labels each sample with a number of labels but these are also ranked in an order based on their relevance to the sample. An example to this is the previous song which in the context of multi-label ranking is primarily a “jazz” song and secondly a “blues” song.

1.2 Aim - Objectives

This thesis revolves around the classification on multi-label datasets where the labels of the instances are ranked in an order based on their importance. The main goal is to significantly enhance the classification accuracy of this type of data. Additionally, by achieving better classification results, this thesis attempts to advance the state-of-the-art in multi-label data classification and provide useful insights for academics and practitioners working with such datasets. Moreover, an ambition of this work is also to close the gap between the practical needs of applications in real life and the complexity of multi-label data.

1.3 Structure of the study

The rest of the thesis is structured as follows: Studies related to the topic are presented in Section 2. Section 3 provides a quick review of the BRkNN algorithm (Binary Relevance kNN). In Section 4 we present the algorithms we created for our experiments. These experiments and their results are described in Section 5. Ultimately, Section 6 provides

a summary of our work, outlines limitations there are and discusses future research opportunities.

Chapter 2

Literature review

Discussing multi-label ranking, [3] described it as the task of building a model that associates to an instance a bipartite split of the label set into relevant and irrelevant labels, as well as a ranking of the entire label set. Consequently, it is possible to think of multi-label ranking as a generalization of multi-label classification and ranking. Their suggested model includes a calibrated scale with a built-in zero-point, 10. This zero-point offers a way to separate the labels that are relevant from those that are irrelevant. Positive labels are rated above the zero-point, while negative labels are ranked below it.

Moreover, a case-based approach is presented by [4]. According to them, model-based multi-label ranking techniques have a high level of computational complexity even for a moderate number of class labels because of the increased complexity of the target space in multi-label ranking compared to binary classification. They present an alternative multi-label ranking framework using a case-based methodology (by adapting the kNN algorithm) that is conceptually and computationally simpler. They also demonstrate that their approach compares favorably with model-based alternatives in terms of both complexity and predictive accuracy.

Multi-label ranking has also had applications on the field of computer vision. Given a test image, [5] propose an efficient algorithm for multi-label ranking aiming to order the object classes in a way that relevant classes are ranked higher than irrelevant classes. They are also able to record relationships between class labels without assuming anything regarding them. Compared to methods based on binary classification, their methodology appears to be more efficient both in terms of object recognition and computationally.

Rather than examining labels equally, [6] concentrate on the special problem of multi-label ranking in cases when the real label set's order is known. They utilize the ranking information in the input and offer a unique dedicated loss function to optimize models by adding penalties for pairings that are rated wrongly. Operating with both ranked and unranked datasets, a method that only works with unranked datasets is to maximize the cross entropy between the predicted probability and the ground truth. The Log-Sum-Exp Pairwise (LSEP) loss [7] is an additional solution to the label ranking issue. LSEP attempts to train a scoring function for an unranked dataset such that a positive label should have a greater score than a negative one. One issue is that while the scores exist, it is unclear which are labeled as positive and which as negative. To solve this issue, they employ the dynamic multi-threshold approach [7], which predicts thresholds for each label in order

to determine if it is positive or negative depending on its score. After that, they learn a scoring function f that yields a label-score vector in order to perform multi-label ranking. The relationship of the respective labels is represented by the relationship between the items in this vector. They propose the Ranked Log-Sum-Exp Pairwise (RLSEP) loss to learn how to sort the labels and they use the threshold technique to generate the ranking results.

Finally, [8] have provided an overall overview of multi-label classification. According to them, multi-label classification methods are split into two categories: a) problem transformation methods which transform the multi-label classification problem into one or more single-label classification or regression problems and b) algorithm adaptation methods which extend certain learning algorithms to directly handle multi-label data. Furthermore, they discuss these two classification categories by giving examples for each one, they explain basic terminology surrounding multi-label datasets characteristics and multi-label classification metrics and lastly they compare some of the aforementioned methods by running experiments.

Chapter 3

BRkNN: a review

This thesis was based on the idea of the BRkNN algorithm to develop the classifiers that were used in the multi-label data classification task. Therefore, it is deemed appropriate to provide a brief presentation of this algorithm.

BRkNN (Binary Relevance kNN) is essentially a multilabel classification adaption of the kNN algorithm (k -nearest neighbor) that is theoretically similar to utilizing the Binary Relevance (BR) problem transformation approach in combination with the kNN algorithm. It was introduced by [9] who in their research developed two versions of the classifier, namely BRkNN-a and BRkNN-b.

Using BR, a multi-label classification problem with L labels is transformed into L single-label independent binary classification problems. Instead of implementing BR together with kNN, [9] developed BRkNN in order to extend the kNN algorithm so that separate predictions are produced for every label after a single search of the k nearest neighbors rather than calculating the k nearest neighbors $|L|$ times, saving important computation time. The first version, BRkNN-a, assigns the labels that are assigned to at least half of the neighbors. For the second version, BRkNN-b, assuming that s is the average number of labels assigned to the object's neighbors, it assigns the s labels with the highest confidence (nearest integer to s).

Based on these two versions of the BRkNN algorithm, we came up with four different classifiers to use in multi-label classification tasks. Our methodology involved executing the kNN algorithm on each instance and obtaining its n neighbors. Next, we deploy our classifiers for these neighbors and a classification is made according to the approach of each classifier.

Chapter 4

The proposed algorithms

Before we present the algorithms we created, a few clarifications have to be made. The most common form of multi-label data, as well as those we used in our experiments, is the representation of the presence of a label with 1 and the non-existence of a label with 0. For example, if for a set of songs the possible labels are “rock”, “pop”, “jazz” and “blues”, then a representation in the form of 0011 means that the song has the labels “jazz” and “blues”. Similarly, a representation in the form of 0101 means that it has the labels “pop” and “blues”. For reasons that will be explained in section 5.2, the data we worked with was processed in such a way that, instead of having representations with 0 and 1, we have representations with characters (A, B, C, etc.). Thus, in the two previous examples the available labels will be A, B, C, D and the representations will be CD (for the 1st song) and BD (for the 2nd song).

4.1 Majority vote per position

The first classifier (*clf1*) we developed was based on the core idea of BRkNN-a. Given a set of instances, *clf1* will examine the labels in each position and assign the label that is in majority. To give an example, suppose we have the following instances:

- ABD
- BCA
- ACD

Based on the principle explained above, *clf1* will examine the first position of all instances where we see 2 occurrences of ‘A’ and 1 occurrence of ‘B’. Therefore, ‘A’ will be assigned as the first position label. Continuing at the second position, there are 2 occurrences of ‘C’ and 1 occurrence of ‘B’ so *clf1* will assign label ‘C’. Lastly, we observe that there are 2 occurrences of ‘D’ and 1 occurrence of ‘A’ at the third position so *clf1* will assign label ‘D’. The final classification of *clf1* in this case will be ‘ACD’.

Of course, the example provided was a simple one with small sized instances, instances with the same number of labels and labels that were not the majority in any position more than one time. Most of the time, these requirements won’t be met. In

order to handle more complex cases as well as tiebreakers, there are some rules that were followed during the development of all classifiers:

1. The size of the final string that will be created will be equal to the label cardinality of the instances that are examined. Label cardinality is the average number of labels the instances have, rounded to the closest integer. In our example, label cardinality is 3.
2. If two or more labels are in the majority at the same position, the label which is present in instances with the smallest total size will be the one selected. This rule was implemented with the thought of “rewarding” labels that are more influential to the instances they are present.

To better understand these rules let’s take a look in a more complex example. Suppose we have the following instances:

ABC	ACDE	CBA	C	BA
-----	------	-----	---	----

Table 4.1. Example instances for *clf1*

		Labels				
		A	B	C	D	E
Occurrences in position	1	2	1	2	0	0
	2	1	2	1	0	0
	3	1	0	1	1	0
	4	0	0	0	0	1

Table 4.2. Occurrences of each label in every position

Based on table 4.2, for the 1st position *clf1* will choose between ‘A’ and ‘C’ which both have 2 occurrences. To break this tie, we examine the length of the strings each of these labels is present in. ‘A’ is present in ‘ABC’, ‘ACDE’, ‘CBA’ and ‘BA’, meaning a total length of 12. ‘C’ is present in ‘ABC’, ‘ACDE’, ‘CBA’ and ‘C’, meaning a total length of 11. Therefore, label ‘C’ is chosen.

For the 2nd position we see that there is no tie so label ‘B’ is chosen’.

For the 3rd position there is a tie between labels ‘A’, ‘C’ and ‘D’, all having 1 occurrence. Label ‘C’ is excluded because it is already chosen. The total length of ‘A’ is 12 and the total length of ‘D’ is 4 (only present in ‘ACDE’), so label ‘D’ is chosen.

The final classification of *clf1* for the instances of table 4.1 is ‘CBD’ because the label cardinality of the instances is:

$$\frac{\text{sum of instances' lengths}}{\text{num of instances}} = \frac{13}{5} = 2.6 \approx 3$$

4.2 Influenced string

Our second classifier’s (*clf2*) methodology is similar to *clf1*. At first, it will examine the labels in each position and again assign the label that is in majority. At the next step,

however, it will examine the labels of the instances that at the previous position had the label that was assigned.

We show how *clf2* works given the example instances from table 4.3:

ABC	ACDE	CBA	CE	BA	EDB	ABD	CEBD	BEA	ECA	DB
-----	------	-----	----	----	-----	-----	------	-----	-----	----

Table 4.3. Example instances for *clf2*

For the 1st position, *clf2* will again choose between ‘A’ and ‘C’ (3 occurrences each), ultimately choosing label ‘C’ (total length of ‘A’ = 21, total length of ‘C’ = 19).

ABC	ACDE	<u>CBA</u>	<u>CE</u>	BA	EDB	ABD	<u>CEBD</u>	BEA	ECA	DB
-----	------	------------	-----------	----	-----	-----	-------------	-----	-----	----

Table 4.4. Instances examined from *clf2* after 1st label is chosen

For the 2nd position, not all instances are taken into account. Instead, as seen in table 4.4, *clf2* will examine instances ‘CBA’, ‘CE’, and ‘CEBD’ because these are the instances that have label ‘C’ in the 1st position. Between their 2nd positions, label ‘B’ has 1 occurrence and label ‘E’ has 2 occurrences, therefore *clf2* will choose ‘E’.

ABC	ACDE	CBA	<u>CE</u>	BA	EDB	ABD	<u>CEBD</u>	BEA	ECA	DB
-----	------	-----	-----------	----	-----	-----	-------------	-----	-----	----

Table 4.5. Instances examined from *clf2* after 2nd label is chosen

For the 3rd position, *clf2* will examine instances ‘CE’ and ‘CEBD’ because they have label ‘E’ in the 2nd position (table 4.5). In this occasion, only ‘CEBD’ has label in the 3rd position so *clf2* will assign label ‘B’.

The final classification of *clf2* is ‘CEB’ (label cardinality = 3).

4.3 Right-shifting

At its first step, the third classifier (*clf3*) will also choose the label that is in the majority in the first position. Moving on to the second position, instances that did not have this label in their previous position will right-shift and then *clf3* will again choose the label with the most occurrences. We will continue the same process for all positions, with the exception of that for the remainder of the algorithm’s execution, the instances with the label selected in the position we looked at will not right-shift. An example is provided below with the instances of table 3 but differently presented to be easily for readers to understand the procedure of *clf3*.

Table 4.6 above presents the labels each instance has in every position. As explained before for *clf1* and *clf2*, for the 1st position *clf3* will choose the label that has the most occurrences and the tiebreaker will be the label that is present in instances with the smallest total length. Since we are using the same example instances from *clf2*, then *clf3* will too choose label ‘C’ which has the most occurrences in the first position together with label ‘A’ (3 occurrences each shown underlined) but smallest total length comparing to it (19 vs 21). The situation of the table then changes in the way shown below:

		Positions					
		1	2	3	4	5	6
Instances	<u>A</u>	B	C				
	<u>A</u>	C	D	E			
	<u>C</u>	B	A				
	<u>C</u>	E					
	B	A					
	E	D	B				
	<u>A</u>	B	D				
	<u>C</u>	E	B	D			
	B	E	A				
	E	C	A				
	D	B					

Table 4.6. Initial state of example instances before the start of *clf3*

		Positions					
		1	2	3	4	5	6
Instances		A	B	C			
		A	C	D	E		
	C	B	A				
	C	<u>E</u>					
		B	A				
		<u>E</u>	D	B			
		A	B	D			
	C	<u>E</u>	B	D			
		B	E	A			
		<u>E</u>	C	A			
		D	B				

Table 4.7. State of example instances for *clf3* after the 1st label is chosen

We can observe now that all instances that did not have label ‘C’ in the first position have shifted one position to the right while the rest remained “frozen” (grey filled cells). After identifying which label is in the majority for the 2nd position, our *clf3* algorithm selects label ‘E’ which has the most occurrences – four – among all labels (underlined).

Lastly, this process will be repeated to select the label for the 3rd place. This is how our table will look now:

		Positions					
		1	2	3	4	5	6
Instances				<u>A</u>	B	C	
				<u>A</u>	C	D	E
	C	B	<u>A</u>				
	C	E					
			B	A			
		E	D	B			
			<u>A</u>	B	D		
	C	E	B	D			
			B	E	A		
		E	C	A			
			D	B			

Table 4.8. State of example instances for *clf3* after the 2nd label is chosen

Instances ‘EDB’ and ‘ECA’ from the previous stage will “freeze” (grey filled cells). After right-shifting instances ‘ABC’, ‘ACDE’, ‘BA’, ‘ABD’, ‘BEA’ and ‘DB’ it is evident by looking at table 4.8 and examining the labels in the 3rd position that *clf3* will choose label ‘A’ which is in the majority with 4 occurrences (shown underlined).

4.4 Labels like medals

The final algorithm with developed is *clf4*. It treats labels as Olympic medals, meaning that for each label *clf4* counts its occurrences in every position in a set of instances and then produces a classification medal table. In this table, ranking is decided by total number of 1st position occurrences first (total number of gold medals), then total number of 2nd position occurrences (total number of silver medals) and son on. If two or more labels have the same amount of total occurrences in any position, ranking is decided recursively by total number of occurrences in the remaining positions. In the case of labels having the same number of occurrences in every position the tiebreaker is, as stated before, the label that is present in instances with the smallest total length.

We present an example of how *clf4* works using the same example instances from *clf2* and *clf3* (table 4.3). The following table 4.9 shows how many times a label is present in every position:

		Labels				
		A	B	C	D	E
Occurrences in position	1	3	2	3	1	2
	2	1	4	2	1	3
	3	3	2	1	2	0
	4	0	0	0	1	1

Table 4.9. Occurrences of each label in every position

For the 1st position, *clf4* will decide between labels 'A' and 'C' that have 3 occurrences (gold medals). To break this tie, *clf4* will take a look at each label's 2nd positions occurrences. Label 'C' has 2 occurrences compared to the 1 occurrence of 'A' so the classifier's choice for the 1st position is label 'C'.

Given that it is the only label left with three gold medals (occurrences in the 1st positions), *clf4* will undoubtedly select label 'A' for the 2nd position.

For the 3rd position we notice that again two labels are tied in gold medals, namely label 'B' and label 'E', each one having 2. Comparing their silver medals (occurrences in 2nd position), *clf4* will eventually assign label 'B' because it has 4 silver medals while label 'E' has 3.

Thus, the final classification of *clf4* is '**CAB**'. The table below shows the classification medal table for the instances that were examined.

Label	Occurrences in position			
	1st	2nd	3rd	4th
C	3	2	1	0
A	3	1	3	0
B	2	4	2	0
E	2	3	0	1
D	1	1	2	1

Table 4.10. Classification medal table for *clf4* algorithm

With the help of table 4.10 it is now straightforward to understand why our classifier's classification was 'CAB'. Label 'C' is at the top of the table since it has 3 occurrences in 1st positions and 2 occurrences in 2nd positions, contrary to label 'A' which has also 3 occurrences in 1st positions but only 1 occurrence in 2nd positions. Similarly, label 'B' beats label 'E' because of more occurrences in 2nd positions (while both have 2 occurrences in 1st positions) and, lastly, label 'D' is at the bottom of the table having only 1 occurrence in 1st positions.

As a result, if *clf4* needed to assign one more label (meaning the label cardinality would be 4) it would choose label 'E' which is directly below 'B' since it has the same number of gold medals (1st position occurrences) but one less silver (2nd position occurrences).

Chapter 5

Performance evaluation

5.1 Datasets

The datasets that were used in this research can be found on the website of the KDIS (Knowledge Discovery and Intelligent Systems) research group of the University of Cordoba [10]. It includes a variety of datasets that can be used in multi-label classification projects along with their characteristics, description and source they were obtained from.

We ran our experiments in a total of 9 datasets from different fields. Details about the datasets are shown in the table below.

Dataset name	Instances	Labels	Cardinality	Density
Birds	645	19	1.014	0.053
CAL500	502	174	26.044	0.150
CHD49	555	6	2.580	0.430
Emotions	593	6	1.868	0.311
Image	2000	5	1.236	0.247
Mediamill	43,910	101	4.376	0.043
Scene	2407	6	1.074	0.179
Water-quality	1060	14	5.073	0.362
Yeast	2417	14	4.237	0.303

Table 5.1. *Datasets characteristics*

Apart from their names, table 11 includes for every dataset the number of instances it contains, the number of labels that are available, the cardinality of the instances - which was explained in section 4.1 - and the density. The density of a dataset is its cardinality divided by the number of labels.

All our datasets have the same format. In an .arff file, each instance is in a separate line. At first, all the attributes are numeric and separated by comma followed by the labels this instance has. These labels are represented with 1 while the rest of the labels - which the instance does not have - are represented with 0. The order of the labels is the same for all instances.

5.2 Experimental setup

Depending on the number of labels each dataset has, we need the same number of characters to represent each label. Most of our datasets (7 out of the 9) have fewer labels than the number of characters of the English alphabet so we can use the capital letters from A to Z without a problem. For the remaining 2 datasets that have more possible labels (101 and 174), we can use capital and lowercase letters (A-Z and a-z) since the Jaro-Winkler distance differentiates them from each other as well as special characters (+, -, @, * etc.) and Unicode characters used with their codes.

As explained in the start of section 4, we transformed our data from sequences of ones and zeros to sequences of characters. At a next step, however, we had to deal with another limitation which was that, as stated in section 5.1, the label order for all instances is the same. For example, four instances with labels 01011 means that the string equivalents would always be four instances of 'BDE' and a label set of 'DEB' would never be possible. As a result, we decided that during the transformation to strings, the order of the labels would be randomized to ensure better representation of all possible combinations of labels. In the previous example, our approach ensures that instances with 01011 as their labels could each time be transformed to one of the possible 'BDE', 'BED', 'DBE', 'DEB', 'EBD' and 'EDB'.

After the transformation and randomization of the labels, we run a simple kNN classifier for every instance in our data and retrieve the n neighbors. The value of n ranges from 1 to 30. We then deploy our four classifiers which take as input the neighbors we found and. Each classifier will return a classification for every instance based on the neighbors it examined and the process it follows to do so, as explained in section 4 for every classifier.

Our experiments were conducted in the environment of the server provided by the MSc in Artificial Intelligence and Data Analytics of the University of Macedonia. The hardware specifications are:

- CPU: Intel Core i9-10900K at 3.7-5.3GHz with 10 cores, 20 threads and 20MB cache
- RAM: 128GB DDR4
- Disk space: 4TB
- Graphics card: RTX3090

5.2.1 Jaro-Winkler distance

The reason we went with the approach of transforming our data from representations of ones and zeros to representations of characters is because of one of the metrics we decided to use in order to measure the performance of our classifiers. This metric is the **Jaro-Winkler distance** which is a string metric measuring an edit distance between two sequences. It was proposed by [11] and it extends the Jaro distance metric [12]. The degree of similarity between two strings is shown by their lower Jaro-Winkler distance.

The score is standardized so that 1 denotes no similarity and 0 indicates an exact match. Another characteristic of this metric is that it rewards strings with common prefixes. In other words, using the Jaro-Winkler distance we can determine how similar two text strings are based on how similar their prefixes are.

In order to calculate the Jaro-Winkler distance, we first have to calculate the Jaro similarity and then the Jaro-Winkler similarity as depicted below:

Jaro similarity of two strings, s_1 and s_2 , is

$$sim_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3} \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) & \text{otherwise} \end{cases} \quad (5.1)$$

where:

- $|s_i|$ is the length of the string s_i
- m is the number of the common characters between the strings
- t is the half the number of the transpositions needed for all the common characters to be in the same order

Jaro-Winkler similarity of two strings, s_1 and s_2 , is

$$sim_{jw} = sim_j + lp(1 - sim_j) \quad (5.2)$$

where:

- sim_j is the Jaro similarity of strings s_1 and s_2
- l is the number of the common prefix at the start of the strings with the maximum value being 4 characters
- p is a constant scaling factor to give better scores for common prefixes and the standard value is $p = 0.1$

Obviously, the **Jaro-Winkler distance** of two strings, s_1 and s_2 , is

$$dist_{jw} = 1 - sim_{jw} \quad (5.3)$$

The Jaro-Winkler distance was calculated easily using a python package, however a simple example is provided to understand how this metric is calculated. Let 'TRATE' and 'TRACE' be the strings we will compare. First, we calculate the Jaro similarity.

T	R	A	T	E
T	R	A	C	E

Table 5.2. The two strings and their common characters

As seen in table 5.2, we can count a total of four common characters between the two strings, so $m = 4$. Continuing, to find the number of transpositions we look for the number of common characters that are not in the right order.

T	R	A	E
T	R	A	E

Table 5.3. Order of the common characters

Since all characters are in the same order, the number of transpositions that have to be made is zero, so half that number is $t = 0$. Using the Jaro similarity formula:

$$sim_j = \frac{1}{3} \times \left(\frac{4}{5} + \frac{4}{5} + \frac{4-0}{4} \right) = \frac{1}{3} \times \frac{13}{5} = 0.867 \quad (5.4)$$

Similarly, using the Jaro-Winkler similarity formula:

$$sim_{jw} = 0.867 + 3 \times 0.1 \times (1 - 0.867) = 0.867 + 0.0399 = 0.907 \quad (5.5)$$

Finally, the Jaro-Winkler distance of 'TRATE' and 'TRACE' is:

$$dist_{jw} = 1 - 0.907 = 0.093 \quad (5.6)$$

The metric confirms what was initially apparent: these two strings are really close to one another and very similar.

5.2.2 Hamming loss

Another metric we used in our experiments is the **Hamming loss** metric, a widely known and one of the most used metrics in classification problems. In multi-label classification, Hamming loss treats an entire predicted set of labels as a wrong classification if it does not exactly match the true set of labels. For example, if the true labels are 'ABCDE' and our classifier predicts 'ABCD', then Hamming loss is 1 (the worst possible value, the best being 0).

5.2.3 Ranked Hamming loss

The third metric we employed in our research is the **ranked Hamming loss**. In order to calculate it for two instances, we first need to create two arrays for each instance. The first array will have a size equal of the number of the possible labels. Each position in the array corresponds with a label of the possible label set in the order of the labels and represents the position that this label has in the instance examined. For example, if the possible labels are A, B, C, D, E and F and our instance is 'CAB':

A	B	C	D	E	F
2	3	1	-	-	-

Table 5.4. Ranked Hamming loss: first array

Number 2 in the start of the table means that label 'A' is in the 2nd position in 'CAB', number 3 means that label 'B' is in the 3rd position and number 1 means that label 'C' is in the 1st position of the instance. Dashes (-) are used for labels that are not present in an instance.

The second array (P) we create is based on the first one. Starting from the first position, we compare each label's rank with all the rest. We assign 1 if the rank i we compare is superior to the rank j (better rank implies lower numeric value); otherwise, we assign 0. In the case of comparing a rank with a dash, the rank value always wins and we assign 1 and if we compare two dashes we will assign a dash. All ranks will be compared with each other only one time. That means that the size of array P is equal to $\binom{n}{k}$:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

where n is the number of all possible labels and k is the number of labels we compare each time. In our example, $n=6$ and $k=2$ so the size of array P will be:

$$\frac{6!}{2!(6-2)!} = \frac{6!}{2! \times 4!} = 15$$

The array will be filled this way:

1	0	1	1	1	0	1	1	1	1	1	1	-	-	-
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Table 5.5. Ranked Hamming loss: second array

For the first position of array P we compare positions 1 and 2 from the first array, meaning values 2 and 3. Since $2 < 3$, 2 is a better rank from 3 and we assign 1. In other words, in 'CAB' label's 'A' position (2) is better than label's 'B' position (3).

For the second position of array P we compare positions 1 and 3 from the first array, meaning values 2 and 1. Since $2 > 1$, 2 is a worse rank from 1 and we assign 0. In other words, in 'CAB' label's 'A' position (2) is worse than label's 'C' position (1). We will then compare positions 1 and 4 from the first array and this process continues until array P is fully completed.

Ranked Hamming loss between two strings is calculated by comparing the created tables (like table 5.5) of each string and calculating the standard Hamming loss. For example, if we have the following arrays for two strings, one for the true instance (P) and one for the predicted (P'):

P	1	0	1	-	-
P'	1	0	0	1	-

Table 5.6. Example arrays for ranked Hamming loss

We see that the predicted instance has 2 misclassifications so the final ranked Hamming loss would be $\frac{2}{5} = 0.4$.

5.2.4 Precision, recall, accuracy, F1 score

Finally, we also calculate the following metrics: **precision, recall, accuracy and F1 score**. After creating tables P and P' , we calculate the confusion matrix for the assignments of 1, 0 and - we make. We compute the aforementioned metrics using this confusion matrix.

Let's demonstrate what was said with an example. Assume the following format for the confusion matrix:

		Predicted label		
		1	0	-
True label	1	6	2	2
	0	0	2	0
	-	2	0	1

Table 5.7. Example confusion matrix

For the assignments of 1:

- True positives (TP) are the instances that were predicted as 1 and were actually 1
- False positives (FP) are the instances that were predicted as 1 and were actually not 1
- True negatives (TN) are the instances that were not predicted as 1 and were actually not 1
- False negatives (FN) are the instances that were not predicted as 1 but were actually 1

The same applies for assignments of 0 and -. After finding the number of TP, FP, TN and FN for every assignment, we calculate precision, recall, accuracy and F1 score separately for each one based on the formulas below. We then compute an average final score for each metric.

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

Precision is the ratio of true positive predictions among all positive predictions made by the classifier, meaning the ability of the classifier not to label a negative sample as positive.

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

Recall (also known as *sensitivity*) measures the proportion of true positive predictions among all actual positive instances. In other words, it is the ability of the classifier to find all the positive samples.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

The accuracy of the classifier is determined by how many of its predictions are correct over the whole dataset.

$$F1score = \frac{2 \times precision \times recall}{precision + recall}$$

The harmonic mean of the precision and recall is used to compute the F1 score, which combines both of these metrics into a single metric.

5.3 Results

The results of the experiments we conducted on the nine datasets are shown in the summary tables below. The first table includes the metrics precision, recall, accuracy

and F1 score. The second table presents the results for Jaro-Winkler distance, Hamming loss and ranked Hamming loss. The split of the results into two tables is intended to improve table comprehension. The values in the first table should be as **high** as they can get, while the values in the second table should be as **low** as they can get. The calculated metrics values for each of the four classifiers are displayed individually for each dataset. For every metric, the best value across all classifiers is shown in bold. This is not the case if the results of all four classifiers for a given metric are identical.

Dataset	Classifier	Precision	Recall	Accuracy	F1 score
emotions	1	0,49	0,50	0,68	0,47
	2	0,50	0,50	0,69	0,48
	3	0,51	0,51	0,69	0,49
	4	0,50	0,51	0,69	0,48
scene	1	0,70	0,70	0,88	0,69
	2	0,70	0,70	0,88	0,69
	3	0,70	0,70	0,88	0,69
	4	0,70	0,70	0,88	0,70
yeast	1	0,58	0,59	0,74	0,56
	2	0,57	0,57	0,73	0,55
	3	0,59	0,59	0,74	0,56
	4	0,59	0,60	0,74	0,57
birds	1	0,37	0,37	0,91	0,36
	2	0,37	0,37	0,91	0,36
	3	0,36	0,37	0,91	0,36
	4	0,37	0,38	0,91	0,37
image	1	0,54	0,54	0,78	0,53
	2	0,54	0,54	0,78	0,53
	3	0,54	0,54	0,78	0,53
	4	0,55	0,55	0,78	0,54
water-quality	1	0,46	0,47	0,65	0,44
	2	0,46	0,46	0,65	0,43
	3	0,47	0,47	0,65	0,44
	4	0,48	0,49	0,66	0,46
CHD_49	1	0,49	0,49	0,66	0,45
	2	0,50	0,50	0,67	0,46
	3	0,50	0,49	0,66	0,46
	4	0,49	0,49	0,66	0,45
CAL500	1	0,43	0,44	0,75	0,43
	2	0,47	0,46	0,77	0,46
	3	0,41	0,42	0,74	0,41
	4	0,46	0,46	0,77	0,46
mediamill	1	0,66	0,69	0,95	0,66
	2	0,69	0,67	0,95	0,66
	3	0,67	0,70	0,95	0,66
	4	0,68	0,71	0,95	0,67

Table 5.8. Precision, recall, accuracy and F1 score values

We can see that all classifiers have very similar results with each other in every dataset across all metrics. Let's examine each dataset separately:

- **emotions:** *clf3* has the best performance across all metrics
- **scene:** all classifiers have the same performance, *clf4* has slightly better results only in F1 score
- **yeast:** *clf4* has the best overall performance
- **birds:** *clf4* has the best results in 3 out of the 4 metrics

- **image:** *clf4* has the best results in 3 out of the 4 metrics
- **water-quality:** *clf4* has the best performance across all metrics
- **CHD_49:** *clf2* has the best overall performance
- **CAL500:** *clf2* has the best overall performance with *clf4* being a very close second
- **mediamill:** there is no classifier that outperforms the others

Dataset	Classifier	Jaro-Winkler distance	Hamming loss	Ranked hamming loss
emotions	1	0,69	0,30	0,48
	2	0,70	0,30	0,47
	3	0,68	0,30	0,47
	4	0,68	0,30	0,47
scene	1	0,30	0,10	0,18
	2	0,30	0,10	0,18
	3	0,30	0,10	0,18
	4	0,30	0,10	0,18
yeast	1	0,49	0,25	0,39
	2	0,53	0,25	0,40
	3	0,49	0,24	0,39
	4	0,49	0,24	0,39
birds	1	0,50	0,07	0,13
	2	0,50	0,07	0,13
	3	0,50	0,07	0,13
	4	0,50	0,07	0,13
image	1	0,47	0,20	0,34
	2	0,47	0,20	0,34
	3	0,47	0,20	0,34
	4	0,46	0,20	0,33
water-quality	1	0,54	0,36	0,53
	2	0,57	0,35	0,53
	3	0,54	0,35	0,52
	4	0,53	0,34	0,51
CHD_49	1	0,65	0,34	0,51
	2	0,66	0,34	0,50
	3	0,66	0,34	0,51
	4	0,67	0,34	0,51
CAL500	1	0,63	0,21	0,37
	2	0,61	0,19	0,34
	3	0,63	0,22	0,38
	4	0,61	0,20	0,35
mediamill	1	0,52	0,04	0,08
	2	0,57	0,04	0,07
	3	0,52	0,04	0,08
	4	0,51	0,04	0,08

Table 5.9. Jaro-Winkler distance, Hamming loss and ranked hamming Loss values

Once more, we observe that the final results produced by each classifier are quite similar:

- **emotions:** *clf3* and *clf4* have the best overall performance in 2 out of the 3 metrics
- **scene:** all four classifiers have the same performance
- **yeast:** *clf3* and *clf4* have the best results, *clf1* very close to them
- **birds:** no classifier outperforms the others
- **image:** *clf4* has slightly better results than the rest of the classifiers
- **water-quality:** *clf4* has the best performance across all metrics
- **CHD_49:** overall same results from all classifiers
- **CAL500:** *clf2* has the best overall performance
- **mediamill:** generally same results across all classifiers

While it is clear that all four classifiers have very similar performances and not a single one of them stands out as clearly best or clearly worst of all, it is also important to mention that *clf4* slightly stands out in many cases, often delivering superior results compared to the other three classifiers. Similarly, *clf3* has shown instances where it either produces better outcomes than the others or is up there with *clf4*. These observations suggest that while the overall performance across classifiers is relatively balanced, *clf4* and *clf3* exhibit particular strengths in specific scenarios, highlighting their potential advantages in certain applications.

In order to better display the small difference in performance by the four classifiers, we present one graph for each dataset where Jaro-Winkler distance (in orange) and Hamming loss (blue) are illustrated for every *k* we ran our experiments for (1 to 30).

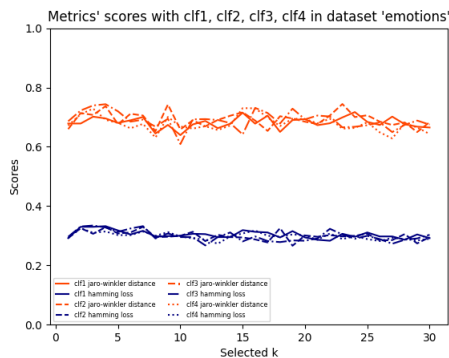


Figure 5.1. Metrics for 'emotions' dataset

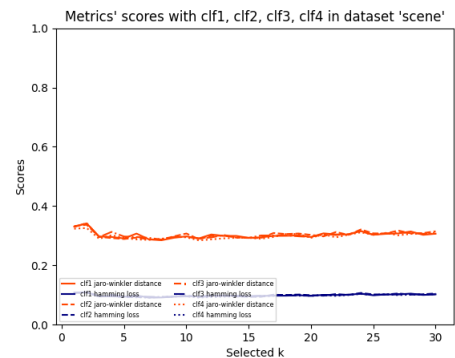


Figure 5.2. Metrics for 'scene' dataset

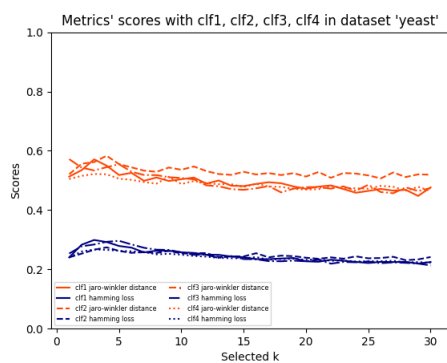


Figure 5.3. Metrics for 'yeast' dataset

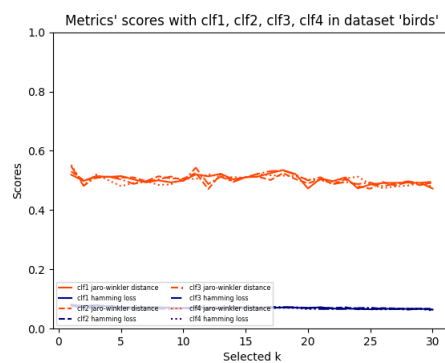


Figure 5.4. Metrics for 'birds' dataset

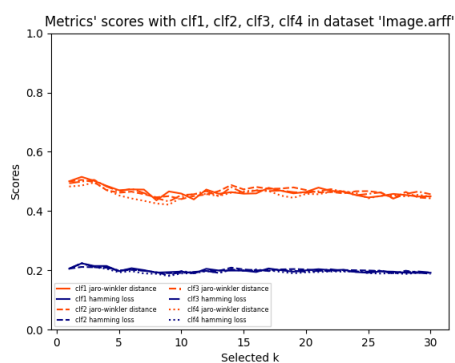


Figure 5.5. Metrics for 'image' dataset

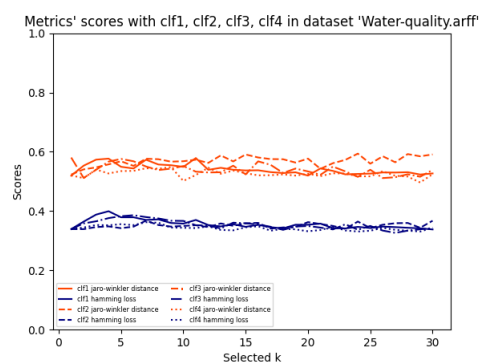


Figure 5.6. Metrics for 'water-quality' dataset

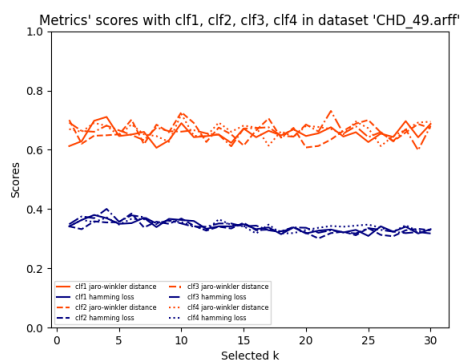


Figure 5.7. Metrics for 'CHD49' dataset

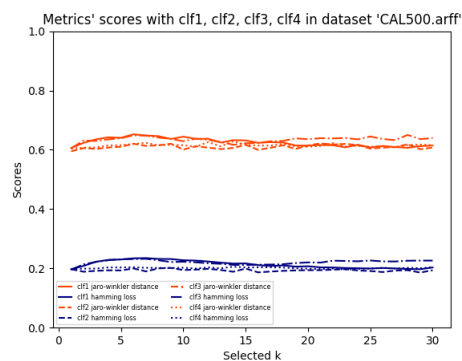


Figure 5.8. Metrics for 'CAL500' dataset

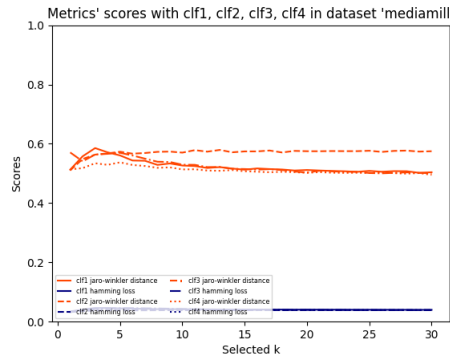


Figure 5.9. Metrics for 'mediamill' dataset

The results of the classifiers in the figures above demonstrate very clearly what was presented in table 5.9; that all four of them are very close in both metrics with little variations and spikes. In figure 5.1 we observe that all classifiers exhibit nearly identical behaviors for the 'emotions' dataset with minor variations in their overall performance. On the other hand, figure 5.2 shows that the four classifiers perform almost the same on the 'scene' dataset. For the 'yeast' dataset (figure 5.3), although Hamming loss has no visible differences between the classifiers, it is worth mentioning that *clf2* has slightly worse Jaro-Winkler distance than the other three classifiers, an observation we previously reported in table 5.9. We also verify the results of the same table for datasets 'birds' and 'image' as figures 5.4 and 5.5 respectively show no apparent differences for all classifiers in either metric.

The graph (5.6) for the metrics' results of the 'water-quality' dataset shows again a quite equal output of the classifiers in terms of Hamming loss values, however *clf2* stands out once more as a slightly inferior performer regarding Jaro-Winkler distance. Dataset 'CHD49' results of figure 5.7 demonstrate a more balanced performance regarding Hamming loss. As for the Jaro-Winkler distance, there are a number of spikes visible along the range of *k*, but still no classifier shows clear advantage over the others. Moving to figure 5.8 and the 'CAL500' dataset, we note that for a little bit over the first half of the *k* values *clf1* produces slightly worse results than the other three classifiers in both metrics, *clf3* does the same for the remaining *k* values and *clf2* is by far the top classifier across all *k* values in both metrics. Finally, in figure 5.9 of the 'mediamill' dataset all classifiers have identical results regarding the Hamming loss metric while *clf2* stands out by having the worst results in terms of Jaro-Winkler distance.

Conclusion

6.1 Summary

The problem we intended to address with this research is the classification of multi-label data where the order of the labels is based on importance. After obtaining our datasets, those were processed in a way that both reflects the fact that the instances have more than one labels ranked in an order of significance and also allowed us to later make use of the Jaro-winkler distance metric. Our methodology involved deploying the kNN algorithm and retrieving a number of neighbors for each instance. We developed four algorithms with each one taking as input a list of neighbors of an instance and, based on its design, creates a string as a final classification for the corresponding instance.

Following the implementation of our classifiers, we recommended seven metrics to assess the performance of them: Jaro-Winkler distance, Hamming loss, ranked Hamming loss, precision, recall, accuracy and F1 score. It was clear that the four classifiers' performances were fairly comparable to one another. However, we would argue that classifier 4 (*labels like medals*) and classifier 3 (*right-shifting*) performed somewhat better in several cases, which makes them worthy of a higher ranking than the other two. These observations suggest that while the overall performance across classifiers is relatively balanced, *clf4* and *clf3* exhibit particular strengths in specific scenarios, highlighting their potential advantages in certain applications.

6.2 Limitations

One of the most important limitations of our work was the difficulty of finding real-world data. Multi-label datasets containing data with labels that are in a ranked order can be challenging to find and obtain. Thus, because our datasets were not in the format we wanted, we had to process them in an arbitrary way and deploy our algorithms on them. Following this process of converting our data to strings, the order of the labels was decided in a random way. Normally, instances that are similar in terms of attributes and belong in the same neighborhood likely also share comparable labels. Unfortunately, this connection could be ruined by the labels' randomness.

6.3 Future work

One of the most important extensions to our work is the development of further algorithms which will follow different methodologies to classify multi-label data where the order of the labels is based on significance. New algorithms could be proven to provide better results and perform overall better than the four classifiers we created. Another opportunity that can be used in future research is the improvement of the time it takes for the algorithms to run. While at no point in our research were we limited by the computing time of our classifiers, we feel that handling datasets that are bigger in size is an aspect of this thesis that deserves to be explored more thoroughly. Lastly, it would be beneficial for this research field to have more real-world datasets with ranked labels available in order to conduct better testing of algorithms and obtain more precise findings.

Epilogue

In this thesis, Data Reduction Techniques (DRTs) were examined to improve the performance of lazy classification algorithms like k-Nearest Neighbors (k-NN). This thesis was prompted by gaps in the literature on the effectiveness and use of various DRTs, especially those developed after 2012.

An extensive experimental analysis was conducted on both Prototype Selection (PS) and Prototype Generation (PG) methods. These methods were further divided into condensation, edition, and hybrid techniques. Twenty advanced DRTs were tested across 20 datasets, which included 13 small and 7 large datasets. Key metrics such as reduction rate, accuracy, and equilibrium were evaluated to understand each technique's performance and trade-offs.

For small datasets, it was found that Particle Swarm Optimization (PSO) was the most effective DRT, performing best in balance, reduction rate, and accuracy. For large datasets, Fuzzy Rough Prototype Selection (FRPS) had the highest accuracy, while Random Mutation Hill Climbing (RMHC) showed the best balance and reduction rate.

This research has provided significant contributions to data mining by offering updated insights and practical evaluations of modern DRTs. Future research will focus on the efficiency of DRTs in terms of execution time and include additional evaluation metrics. New DRTs expected to be published soon will also be included in future studies.

The process of conducting this research has been a great learning experience. It allowed for a deep understanding of data reduction techniques and their importance in improving the efficiency and accuracy of machine learning models. It is hoped that these findings will be a valuable resource for researchers and practitioners, leading to further advancements in data reduction methods.

In the context of this thesis, the paper titled "An Empirical Analysis of Data Reduction Techniques for k-NN Classification" was published at the conference "20th International Conference on Artificial Intelligence Applications and Innovations (AIAI2024)."

Bibliography

- [1] Yangming Zhou, Yangguang Liu, Jiangang Yang, Xiaoqi He και Liangliang Liu. *A Taxonomy of Label Ranking Algorithms*. *Journal of Computers*, 9:558, 2014.
- [2] Panagiotis Filippakis, Stefanos Ougiaroglou και Georgios Evangelidis. *Prototype Selection for Multilabel Instance-Based Learning*. *Information*, 14(10):1, 2023.
- [3] Klaus Brinker, Johannes Fürnkranz και Eyke Hüllermeier. *A Unified Model for Multilabel Classification and Ranking*. *Proceedings of the 17th European Conference on Artificial Intelligence*, ECAI 2006, σελίδες 489–493. IOS Press, 2006.
- [4] Klaus Brinker και Eyke Hüllermeier. *Case-based multilabel ranking*. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, σελίδα 702–707, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.
- [5] Serhat Bucak, Pavan Mallapragada, Rong Jin και A.K. Jain. *Efficient Multi-label Ranking for Multi-class Learning: Application to Object Recognition*. *Proceedings of the 12th IEEE International Conference on Computer Vision*, IEEE 2009, σελίδες 2098–2105, 2009.
- [6] Emine Dari, Yesilkaynak V. Bugra, Alican Mertan και Gozde Unal. *RLSEP: Learning Label Ranks for Multi-label Classification*. *arXiv (Cornell University)*, 2022.
- [7] Yuncheng Li, Yale Song και Jiebo Luo. *Improving Pairwise Ranking for Multi-label Image Classification*. *arXiv (Cornell University)*, 2017.
- [8] Grigorios Tsoumakas και Ioannis Katakis. *Multi-Label Classification: An Overview*. *International Journal of Data Warehousing and Mining*, 3:1–13, 2009.
- [9] Ioannis Vlahavas Eleftherios Spyromitros, Grigorios Tsoumakas. *An Empirical Study of Lazy Multilabel Classification Algorithms*. *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008)*, 2008.
- [10] KDIS. *Multi-Label Classification Dataset Repository*, 2023.
- [11] William Winkler. *String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*. *Proceedings of the Section on Survey Research Methods*, 1990.
- [12] Matthew A. Jaro. *Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida*. *Journal of the American Statistical Association*, 84:414–420, 1989.

