Victor Freitas Rocha

Exploring label correlations in multi-label ensemble classifiers using decision templates

Vitória, ES

Agosto, 2022

Victor Freitas Rocha

Exploring label correlations in multi-label ensemble classifiers using decision templates

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Supervisor: Prof. Dr. Flávio Miguel Varejão

Vitória, ES

Agosto, 2022

Ficha catalográfica disponibilizada pelo Sistema Integrado de
Bibliotecas - SIBI/UFES e elaborada pelo autor

Freitas Rocha, Victor, 1992-

F862e Exploring label correlations in multi-label ensemble classifiers using decision templates / Victor Freitas Rocha. - 2022.

65 f. : il.

Orientador: Flávio Miguel Varejão.

Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Inteligência artificial. 2. Classificação. 3. Multirrótulo. 4. Comitês. 5. Perfis de decisão. I. Miguel Varejão, Flávio. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Ensemble of classifier chains and decision templates for multi-label classification

Victor Freitas Rocha

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 25 de Agosto de 2022.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória/ES, 25 de Agosto de 2022

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
FLAVIO MIGUEL VAREJAO - SIAPE 297887
Departamento de Informática - DI/CT
Em 25/08/2022 às 16:34

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/548552?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
THIAGO OLIVEIRA DOS SANTOS - SIAPE 2023810
Departamento de Informática - DI/CT
Em 25/08/2022 às 18:19

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/548721?tipoArquivo=O

To my family and especially to my mother Rosangela. May she rest in peace.

Acknowledgements

The work presented in this dissertation would not be possible without the support and assistance I received.

I would like to thank my long-time supervisor, Professor Flávio Miguel Varejão, whose feedback and encouragement were fundamental to the development of not only this dissertation, but also the works on which this dissertation is based. There were major setbacks throughout the writing of this dissertation and even at times when I doubted myself, he encouraged me to continue. For that, I thank him sincerely.

In addition, I would like to thank my family for supporting me up to this point, especially my mother, who would undoubtedly be proud. Finally, I would not have completed this dissertation without the help of my friend Lucas Mello, who together with the NINFA lab team at UFES provided me with the necessary infrastructure to perform the experiments presented here.

“It’s not the voting that’s democracy, it’s the counting.“
(Tom Stoppard)

Resumo

No contexto de aprendizado de máquina, a classificação é a tarefa de identificar a que classe pertence uma instância de acordo com o conhecimento obtido atráves de um conjunto de treinamento, ou seja, um conjunto de instâncias cuja classificação é previamente conhecida. A classificação unirrótulo, uma das versões mais tradicionais do problema de classificação, permite que uma instância pertença a apenas uma classe considerando-as, portanto, mutualmente exclusivas. Entretanto, problemas do mundo real em que interseções entre classes ocorrem frequentemente podem ser melhor modelados como problemas de classificação multirrótulo, cuja tarefa permite que múltiplos rótulos sejam atribuidos à mesma instancia. Múltiplos classificadores, tanto unirrótulo quanto multirrótulo, podem ser treinados para o mesmo problema de classificação e gerar um resultado combinado. Essa técnica, conhecida como comitês de classificadores, é comumente utilizada para melhorar a performance de classificação. Várias abordagens já foram propostas para realizar a combinação dos resultados individuais dos classificadores. Neste trabalho, é apresentada uma abordagem de combinação de conjuntos de classificação múltirrótulo baseado na técnica Modelos de Decisão para Comitês de Cadeias de Classificadores que incorpora a exploração de correlações entre os rótulos no processo de fusão dos classificadores.
Na técnica de Perfis de Decisão, originalmente proposta para a fusão de classificadores de rótulo único, estima-se um perfil de decisão por classe usando o mesmo conjunto de treinamento que é usado para o conjunto de classificadores. A classificação para cada instância invisível é obtida medindo a similaridade entre seu perfil de decisão e os perfis de decisão das classes. O método proposto estima dois perfis de decisão por classe, um representando a presença da classe e o outro representando sua ausência. Para cada nova instância, é criado um novo perfil de decisão e a similaridade entre os perfis de decisão das classes e o perfil de decisão da instância determina o conjunto de rótulos resultante. Para cada rótulo analisado, informações sobre rótulos correlacionados é incorporada. O método de fusão proposto é utilizado em um algoritmo tradicional e de eficiência comprovada de comitê de classificadores múltirrótulo: Comitês de Cadeias de Classificadores. As evidências empíricas indicam que o uso da adaptação dos Perfis de Decisão proposto pode melhorar o desempenho em relação aos esquemas de combinação tradicionalmente usados na maioria das métricas avaliadas melhorando o desempenho de um método de comitê de classificadores já conhecido na literatura multirrótulo.

Palavras-chaves: Multirrótulo. Classificação. Comitês. Perfis de decisão.

Abstract

In the context of machine learning, classification is the task of identifying which class an instance belongs according to the knowledge obtained through a training set, that is, a set of instances whose classification is previously known. Single-label classification, one of the most traditional versions of the classification problem, allows an instance to belong to only one class, thus making them mutually exclusive. However, real-world problems where intersections between classes often occur can be better modeled as multi-label classification problems, whose task is to allow multiple labels to be assigned to the same instance. Multiple classifiers, both single-label and multi-label, can be trained on the same classification problem and generate a combined result. This technique, known as classifier ensembles, is commonly used to improve classification performance. Several approaches have already been proposed to perform the combination of the individual classifiers' results. In this work, an approach for combining multiple-label classification sets based on the Decision Templates for Ensemble of Classifier Chains technique is presented that incorporates the exploration of correlations between the labels in the classifiers' fusion process. In the Decision Templates technique, originally proposed for merging single-label classifiers, a per-class decision model is estimated using the same training set that is used for the set of classifiers. The classification for each unseen instance is obtained by measuring the similarity between its decision profile and the decision templates. The proposed method estimates two decision templates per class, one representing the presence of the class and the other representing its absence. For each new instance, a new decision profile is created and the similarity between the decision templates and the decision profile determines the resulting set of labels. For each label analyzed, information about correlated labels is incorporated. The proposed fusion method is used in a traditional and proven algorithm of multiple-label classifier committee: Ensemble of Classifier Chains. Empirical evidence indicates that the use of the proposed Decision Templates adaptation can improve performance over traditionally used fusion schemes on most of the evaluated metrics.

Keywords: Multi-label. Classification. Ensembles. Decision Templates.

List of Figures

List of Tables

List of abbreviations and acronyms

CC	Classifier Chains
CD	Critical Diagram
CV	Cross Validation
DT	Decision Template
DP	Decision Profile
DTECC	Decision Templates for Ensemble of Classifier Chains
UDDTECC	Unconditionally Dependent Decision Templates for Ensemble of Classifier Chains
ECC	Ensemble of Classifier Chains
ME	Mean Ensemble
MEECC	Mean Ensemble for Ensemble of Classifier Chains
ML	Multi-label
MLC	Multi-label Classification
MV	Majority Vote
MVECC	Majority Vote for Ensemble of Classifier Chains

Contents

1 Introduction

The single-label classification task associates a feature vector $\boldsymbol{x}$ from domain $\mathcal{X}$ with exactly one out of m possible labels ℓ_j, $j = 1, \ldots, m$. A training set $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ consists of feature-label pairs $(\boldsymbol{x}_k, y_k)$ where the desired output y_k is one of the possible m labels. A common practice is to code the class membership of the features as a vector $\boldsymbol{y}_k$ from domain $\mathcal{Y}$ of length m where the components are binary values. This is called *one-out-of-m* coding. Only one of the binary values has value one, i.e. $y_{k,j} = 1$, the remaining $(m-1)$ values are zero, hence the associated class information of the feature vector $\boldsymbol{x}_k$ becomes $\boldsymbol{y}_k = (y_{k,1} \ldots y_{k,j-1}, y_{j,k}, y_{k,j+1} \ldots y_{k,m})$. The single-label classification problem may be binary (two possible label values) or multiclass (three or more label values).

Multi-label classification is a generalization of the traditional single-label classification problem where instances are associated with a label set instead of a single label. Several real world domains are better represented as multi-label problems. In the movie genre classification, for instance, a movie can belong to any combination of genres, such as drama, action and romance at the same time. In the multi-label classification task let $\mathcal{X} \subseteq \Re^n$ be the feature space and $L = \{\ell_1, \ell_2, \ell_3, \ldots, \ell_m\}$ be the set of possible labels. A *classifier* can be defined as a function that maps the feature space $\mathcal{X}$ to the label space $\mathcal{Y} = \{0, 1\}^m$ or $h : \mathcal{X} \to \mathcal{Y}$, $\boldsymbol{x} \mapsto \boldsymbol{y}$. Many real-world problems such as image recognition (CHEN et al., 2019), subcellular locations of proteins (CHOU, 2019), and text classification (DU et al., 2019) can best be represented as multi-label problems.

Many of the proposed multi-label classification algorithms, such as Ensemble of Classifier Chains (ECC) (READ et al., 2011) are based upon the concept of ensembles of classifiers. In those methods, an ensemble is composed of multiple multi-label classifiers. In order to classify an unseen instance, the output of each member of the ensemble for that instance is collected and then a predefined *fusion function* determines the final classification result. The choice of this function has fundamental importance for the operation of the ensemble and directly influences its performance.

This work focuses on an adaptation of the Decision Templates (DT) classifier fusion method, which organizes the results of the ensemble members into matrices called Decision Profiles (DP). These DPs are then used to calculate the DT of each class. Given a label j, a DT can be understood as the expected DP for the instances that belong to the j-th class. During the classification process of an unknown instance, the class with the most similar DT is chosen. Since this method was created to work with single-label ensembles, DT cannot be applied directly in multi-label problems.

In a review study evaluating state-of-the-art multi-label classifiers based on ensembles, the ECC method stood out as the one with the best overall performance and, therefore, was chosen to be used in this work (MOYANO et al., 2018). In order to adapt the DT method to the ECC multi-label classifier, the Decision Templates for Ensemble of Classifier Chains (DTECC) method was proposed (ROCHA; VAREJÃO; SEGATTO, 2022). Two DP matrices are defined for each label, one representing the instances that are assigned to that label and a complementary DP representing the instances that are not assigned to that label. For an unseen instance, the task of classification for each label becomes the task of selecting the most suitable of the two DP matrices computed for that label. This procedure enables the assignment of multiple labels to each instance, and thus, adapts the DT method to multi-label problems. We also expand the ideia of the DTECC and propose a novel fusion method called UDDTECC (Unconditionally Dependent Decision Templates for Ensemble of Classifier Chains) where the DP matrices of a class j can use classification information from other labels in the problem if any correlation between those labels is identified.

An adaptation of the DT method for working with the *Recursive Dependent Binary Relevance* (RDBR) multi-label classifier was presented in (RAUBER et al., 2016). Despite not being an ensemble method, the intermediate values generated by the RDBR classification process can be used to compose DP matrices. This procedure, however, is not ideal since the RDBR method offers small diversity when compared to a real ensemble method. To the best of our knowledge, unless of the DTECC method, this is the unique application of Decision Templates in a multi-label context.

In (ROCHA; VAREJÃO; SEGATTO, 2022) the performance of the DTECC is empirically evaluated and compared to two known fusion methods often present in the literature: Majority Vote and Mean Ensemble. The empirical results show that replacing the fusion scheme of the ECC method with the DT can improve the performance for Accuracy and F-measure in most data sets, while remaining competitive in terms of other metrics.

1.1 Objectives

This work adapts Decision Templates fusion function to teh multi label classification problem and also includes the exploration of inter-label correlations in the fusion process. This is done specifically by proposing two new methods called DTECC (ROCHA; VAREJÃO; SEGATTO, 2022) and UDDTECC which extends the DTECC method so as to take label correlations into account. To accomplish this task, the specific objectives are:

- Literature review of the theoretical foundation necessary for the understanding of the proposed methods.

- Presentation of the proposed DTECC and UDDTECC methods.
- Performing experiments that compare the proposed method to the most widely used in the literature.
- Analysis of the experiments.

1.2 Justification and Contributions

We propose a novel approach, which incorporates the exploitation of label correlations in the multi-label ensemble fusion process in order to improve its performance. Another trainable method based on the stacking scheme (called STACKECC) is also proposed to compare the UDDTECC method due to the similarity of the DDTECC scheme to a stacking strategy. Therefore, in this dissertation two fusion methods are proposed and compared with the most widely used fusion methods in the multi-label classification literature. Furthermore, experimental results are presented in order to prove the effectiveness of using the proposed method.

1.3 Structure

This document is structured as follows:

Chapter 2 presents the theoretical foundation necessary for the understanding of the proposed method. Chapter 2.1 defines the problem of multi-label classification including the difference between problem adaptation methods and problem transformation methods. The Classifier Chains method, the base classifier of the proposed method, is also defined in detail. In Chapter 2.2 classifier ensemble methods are defined and explained. An explanation of the inner workings of the ECC method in its traditional configuration is also provided. Chapter 2.3 discusses fusion functions. The difference between fixed fusion rules and trainable functions is explained and this difference is exemplified through its main representatives. Chapter 3 proposes and explains how the proposed DTECC and UDDTECC methods work. Chapter 4 details the methodology used in the experiments including the metrics and datasets used. Chapter 5 presents and discusses the results of the experiments performed comparing the methods according to their results on each metric. Finally, Chapter 6 presents the work conclusions and indicates future works.

2 Theoretical Background

This section presents the theoretical concepts necessary to understand the proposed methodology. First, the concept of a multi-label classifier is defined and its different approaches are briefly explained. Being a fundamental part of the UDDTECC method, a more detailed explanation of the multi-label Classifier Chains classification method is provided. Then the concept of Ensemble of Classifiers is explained and the Ensemble of Classifier Chains method is detailed. Fusion functions, responsible for combining the results of the various classifiers in the ensemble, are discussed in section 2.3 and the differentiation between trainable and fixed methods is explained. The untrainable MajorityVote and MeanEssemble and trainable Decision Templates and Stacking fusion methods used in the experiments are detailed.

2.1 Multi-label Classification MLC

Multi-label classification, a generalization of the single-label classification problem, allows an observation (or instance) to be associated with multiple labels. This generalization allows us to more accurately model many real-world problems.

Multi-label classification generalizes the classification problem by removing the restriction in single-label classification that an observation (or instance) can only be associated with one label. This generalization is key to accurately model many real-world problems where instances may belong to intersections of the classes. Let L be the set of possible labels of a multi-label classification problem, all $2^{|L|}$ unique label combinations are therefore acceptable answers. In this context, it becomes critical to understand the correlations between the labels in order to extract the best possible result (LIN et al., 2017).

A multi-label classifier can be defined as a mapping $\boldsymbol{h} : \mathcal{X} \rightarrow \mathcal{Y}$, $\boldsymbol{x} \mapsto \boldsymbol{y}$ of the feature space $\mathcal{X}$ for the class domain $\mathcal{Y} = \{0, 1\}^m$. A multi-label dataset is defined as n pairs of feature vectors and binary vectors of dimension m as $S = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^n, \boldsymbol{y}^n)\}$, with $y_j^k \in \{0, 1\}$. Where $y_j^k = 1$ represents the presence of the label ℓ_j in example k and $y_j^k = 0$ represents its absence.

Multi-label methods can be divided into two groups: problem transformation methods and problem adaptation methods. The difference between these strategies for approaching the multi-label problem are explained next.

2.1.1 Problem adaptation

The problem adaptation methods strategy consists of modifying existing single-label methods to work directly with multi-label instances. Examples of methods that follow this approach include: ML-kNN (ZHANG; ZHOU, 2007) which adapts the single-label kNN method, Adaboost.MH and Adaboost.MR (SCHAPIRE; SINGER, 2000) both adapting the AdaBoost (FREUND; SCHAPIRE, 1997) method for multi-label classification, and BPMLL (ZHANG; ZHOU, 2006) which adapts a back-propagation neural network to work directly with multi-label classification.

2.1.2 Problem transformation

In the problem transformation approach, the multi-label problem is transformed into one or several single-label problems. This transformation allows the use of traditional single-label classification methods on this modified input space.

Binary relevance is widely considered as the baseline method of multi-label problem transformation, it consists of forming a binary classifier that independently classifies each label of a given multi-label problem. This strategy, although straightforward, does not take into account the correlations between the labels and can lead to suboptimal classification performance (TANAKA; BARANAUSKAS, 2012). Several problem transformation methods that seek to explore the correlation between labels have been proposed among which are Label Powerset (LP), Pruned Sets (PS), ChipDep and Classifier Chains (CC).

The Label Powerset (LP) (TSOUMAKAS; VLAHAVAS, 2007) transformation method incorporates correlations between labels by considering each of the different subsets of labels as a single label. It then then learns a single-label multiclass classifier to make the predictions. The main disadvantage of the LP method is the large number of label subsets presented in the datasets, most of them associated with very few examples.

In order to overcome the sparseness problem of LP, the Pruned Sets (PS) (READ; PFAHRINGER; HOLMES, 2008) method was proposed. Pruning removes the sets of labels that occur less frequently in the multi-label training set D. Therefore reducing the computational complexity of the problem by prioritizing the relations of more important label sets in D.

Following another transformation approach, the ChipDep (TENENBOIM-CHEKINA; ROKACH; SHAPIRA, 2010) method seeks to identify the dependencies between labels and clusters all labels in independent subsets, building an LP classifier for each subset. The LP classifiers are combined similarly to the Binary Relevance method, i.e, transforming the multi-label problem into several binary problems one for each label.

Derived from BR, a chaining methodology was proposed as a way to explore possible correlations between labels in the classification process. This method, called Classifier

Chains (CC), is an integral part of the Ensemble Of Classifier Chains (ECC) methodology. Since the UDDTECC method is applied on top of ECC, a more detailed explanation is provided next.

The multi-label classification method Classifier Chain (CC) is categorized as a problem transformation method, i.e., it transforms the multi-label problem into n single-label problems to allow the use of single-label methods.

To explore possible correlations between the labels the CC method first defines the order in which the problem labels will be evaluated and for each evaluated label it incrementally adds the classification results using the evaluated labels. The first classifier of the string $h_1^{\mathrm{CC}} : \mathcal{X} \to \{0,1\}$, $\boldsymbol{x} \mapsto y_1$ works as one of m single-label binary classifiers. The prediction value of the first label is used in conjunction with the feature vector $\boldsymbol{x}$ to perform the prediction of the second label. $h_2^{\mathrm{CC}} : \mathcal{X} \times \{0,1\} \to \{0,1\}$, $(\boldsymbol{x}, y_1) \mapsto y_2$. The classifiers that compose the CC are trained using the real labels y_1, whereas in the prediction phase the estimated label values $\hat{y}_1$ are used in the mapping, such that $(\boldsymbol{x}, \hat{y_1}) \mapsto \hat{y_2}$. Therefore, CC is composed of m classifiers that can use from zero to $(m-1)$ labels as additional attributes, such that in the training phase $h_j^{\mathrm{CC}} : \mathcal{X} \times \{0,1\}^{j-1} \to \{0,1\}$, $(\boldsymbol{x}, y_1, \ldots, y_{j-1}) \mapsto y_j, j = 1, \ldots, m$ and in the prediction phase $h_j^{\mathrm{CC}} : \mathcal{X} \times \{0,1\}^{j-1} \to \{0,1\}$, $(\boldsymbol{x}, \hat{y}_1, \ldots, \hat{y}_{j-1}) \mapsto \hat{y}_j, j = 1, \ldots, m$. Although it directly influences the classification performance, the CC method does not define the order in which the labels are chained. The ordering used is defined randomly so that two CC classifiers trained on the same training set can generate different classifications for the same test instances. In order to create results less dependent on the random ordering defined by the CC method, the Ensemble of Classifier Chains (ECC) method was proposed. By training multiple CC classifiers each with random label ordering, the ECC method achieves a more stable classification result. The ECC method is explained in more detail below.

2.2 Ensembles of Multi-label Classifiers

The technique of combining the predictions of multiple classifiers to produce a single classifier is known as ensemble of classifiers (OPITZ; MACLIN, 1999; ROKACH, 2010). Typically, an classifier ensemble achieves better performance than any of the individual classifiers of the ensemble. Good ensembles are generally composed by individual classifiers which are both accurate and make their errors in different parts of the input space. Ensemble methods can be divided into two categories: classifier selection and classifier fusion (WOODS; KEGELMEYER; BOWYER, 1997; KUNCHEVA; BEZDEK; DUIN, 2001; KUNCHEVA, 2002; HO, 2000).

Classifier selection methods train classifiers to be local experts in some area of the feature space. The importance given to the prediction of a specific classifier is inversely proportional to the distance of the data used to train this classifier. One or more classifiers can be selected to give the final decision (WOODS; KEGELMEYER; BOWYER, 1997; JACOBS et al., 1991; ALPAYDIN; JORDAN, 1996; GIACINTO; ROLI, 2001).

In classifier fusion, classifiers are trained on the entire feature space. In those techniques the output of the weaker classifiers must be merged to create the stronger output. Traditional methods such as Bagging (BREIMAN, 1996) and Boosting (SCHAPIRE, 1990; FREUND; SCHAPIRE; ABE, 1999) are based on this approach. These methods are based on resampling techniques for obtaining different training sets for each of the individual classifiers. This work focuses on classifier fusion methods. Thus, a more in dept explanation on the operation of classifier fusion methods is presented in section 2.3.

Many of the proposed multi-label classification algorithms, such as Ensemble of Classifier Chains (ECC) (READ et al., 2011), Ensemble of Pruned Sets (EPS) (READ; PFAHRINGER; HOLMES, 2008), Ensemble of Subset Learners (ESL) (TENENBOIM-CHEKINA; ROKACH; SHAPIRA, 2010), and RAndom k-labELsets (RAkEL) (TSOUMAKAS; VLAHAVAS, 2007) are based upon the concept of ensembles of classifiers. In those methods, an ensemble is composed of multiple multi-label classifiers. In order to classify an unseen instance, the output of each member of the ensemble for that instance is collected and then a predefined *fusion function* determines the final classification result. The choice of this function has fundamental importance for the operation of the ensemble and directly influences its performance.

The RAndom k-labELsets (RAkEL) (TSOUMAKAS; VLAHAVAS, 2007) overcomes this problem by creating an ensemble composed of LP classifiers. Each ensemble member is trained based on k randomly chosen label subsets (denominated labelsets). Although this approach significantly reduces the number of label subsets, depending on the value of k, there is still the possibility of having large number of label subsets associated with very few examples.

Incorporating the idea of ensembles, EPS combines several PS models. During the training phase of the EPS method, c iterations are performed in which a subset of the examples is selected to train a PS classifier. At the end of the procedure, c different classifiers are trained.

The ESL method combines distinct ChiDep classifiers to improve classification performance. Each one of the c ChiDep classifiers that compose the ensemble is defined by a distinct label set partition. The c set partitions are selected from a large sample of randomly generated partitions. The score for each generated set partition is computed according to a normalized χ^2 score for of all its label pairs and only the best c partitions according to the highest scores are selected as members of the ensemble.

As explained in the section 2.1.2, the Classifier Chains method is a problem transformation method that chains the problem labels and performs their classification, the method, however, does not define a predetermined optimal chaining order. The order in which the chaining of results is done has an impact on the final result, and is therefore an important issue when using the CC method. With this in mind, the method Ensemble of Classifier Chains was proposed. In the ECC method n CC classifiers $C_1, C_2, \ldots, C_n$ are trained. In order to create a diverse set of classifiers, each of these classifiers is trained using a random ordering of labels and a random subspace of the training dataset, allowing each CC model to be unique and capable of making distinct predictions. (READ et al., 2011). In a review study evaluating state-of-the-art multi-label classifiers based on ensembles, the ECC method stood out as the one with the best overall performance and, therefore, was chosen to be used in this work (MOYANO et al., 2018). Thus, the UDDTECC fusion method proposed in this paper focuses on the Ensemble of Classifier Chains for its experimental evaluation.

2.3 Fusion Functions

Let $\mathcal{H} = \{h_1, h_2, \ldots, h_c\}$ be the set of classifiers of a classifier ensemble. The output of each classifier can be scaled to the [0,1] interval without loss of generality. For a feature vector $\boldsymbol{x} \in \mathcal{X}$, the output of the ith classifier is represented as $h_i(\boldsymbol{x}) = [d_{i,\ell_1}(\boldsymbol{x}), \ldots, d_{i,\ell_m}(\boldsymbol{x})]$, $d_{i,\ell_j}(\boldsymbol{x}) \in [0,1], 1 \leq j \leq m$, where $d_{i,\ell_j}(\boldsymbol{x})$ can be generally interpreted as the degree of *support* outputted by the h_i classifier to the hypothesis that $\boldsymbol{x}$ is labeled as ℓ_j. The output of all classifiers that compose an ensemble composed by c classifiers and m labels can be organized as a matrix known as the Decision Profile (DP) of $\boldsymbol{x}$:

$$\mathrm{DP}(\boldsymbol{x}) = \begin{pmatrix} d_{1,\ell_1}(\boldsymbol{x}) & \ldots & d_{1,\ell_j}(\boldsymbol{x}) & \ldots & d_{1,\ell_m}(\boldsymbol{x}) \\ \ldots & & & & \\ d_{i,\ell_1}(\boldsymbol{x}) & \ldots & d_{i,\ell_j}(\boldsymbol{x}) & \ldots & d_{i,\ell_m}(\boldsymbol{x}) \\ \ldots & & & & \\ d_{c,\ell_1}(\boldsymbol{x}) & \ldots & d_{c,\ell_j}(\boldsymbol{x}) & \ldots & d_{c,\ell_m}(\boldsymbol{x}) \end{pmatrix}.$$

The output of the resulting classifier $\widehat{h}$ is defined by a *fusion function* $\mathcal{F}$ that uses the classifiers outputs to create a combined output:

$$\widehat{h}(\boldsymbol{x}) = \mathcal{F}(h_1(\boldsymbol{x}), \ldots, h_c(\boldsymbol{x})) = \mathcal{F}(\mathrm{DP}(\boldsymbol{x})). \tag{2.1}$$

Fusion functions can be divided into two groups based on whether they require additional training once the classifiers of the ensemble are trained (KUNCHEVA, 2014). The explanation of both strategies is presented in the following subchapters.

2.3.1 Fixed Fusion Functions

One of the simplest and most direct strategies for combining individual classifications generated in a classifier committee is to use non trainable functions directly on the support values for that label (KUNCHEVA, 2014). Thus, given a DP matrix, the support value $d_j(\boldsymbol{x})$ for label ℓ_j is defined using a function f such that $\ell_j = \mathcal{F}(d_{1,\ell_j}(\boldsymbol{x}), \ldots, d_{c,\ell_j}(\boldsymbol{x}))$ and a threshold value t, typically defined as $t = 0.5$, is used to define the final labels of the classification.

Except for the majority vote, which uses DP($\boldsymbol{x}$) for computing the label that is most voted by the classifiers, all these functions compute the combined support value for each label as

$$d_j(\boldsymbol{x}) = f(d_{1,\ell_j}(\boldsymbol{x}), \ldots, d_{c,\ell_j}(\boldsymbol{x})), \tag{2.2}$$

where f represents the corresponding mathematical function (maximum, minimum, average, or product). For single-label classification, the resulting label is calculated using equation 2.3: In single-label problems, only one label can be ultimately assigned to $\boldsymbol{x}$. This is usually done by selecting the label with the greater support value:

$$\widehat{h}(\boldsymbol{x}) = \ell \Leftrightarrow d(\boldsymbol{x}) = \max_{j=1,\ldots,m} \{d_j(\boldsymbol{x})\}. \tag{2.3}$$

2.3.1.1 Majority Vote

This fusion function, named MajorityVote (MV), sums the positive and negative votes of all classifiers for each problem class. The fraction of positive votes relative to the total number of classifiers is calculated and only labels with a value above a defined threshold t are assigned to the example. Traditionally only classes with a majority of positive votes (i.e. $t \geq 0.5$) are assigned to the example and ties are broken randomly. This approach is exemplified with an example from the Image data set in table 1.

The fusion scheme originaly proposed by the authors of the ECC method was majority voting (MV). MV is one of the most straightforward and widely used fusion schemes in problems involving ensembles of multi-label classifiers. In this scheme, the label ℓ_j is only assigned to a test example $\boldsymbol{x}$ if a certain percentage of the classifier ensemble members (a predefined threshold value) have predicted for this label. The individual results of each classifier $h_i = (d_{i,\ell_1}, \ldots, d_{i,\ell_m})$ are summed and stored in a sum vector $W = (\lambda_1, \ldots, \lambda_m)$ such that $\lambda_j = \sum_{i=1}^{c} d_{i,\ell_j}$, which is then normalized into a vector W^{norm}, representing a distribution in $[0, 1]$. The final classification result Y is defined by a threshold value t so that $\ell_j \in Y$ only if $\lambda_j \geq t$. The threshold value is usually set to 50%, meaning that for the label ℓ_j to be in the final prediction at least half of the classifiers must have predicted this label.

$\mathcal{L} =$	{	Desert	Mountains	Sea	Sunset	Trees	}
$\boldsymbol{h_1}$	(	1	0	0	1	1	)
$\boldsymbol{h_2}$	(	1	1	0	1	0	)
$\boldsymbol{h_3}$	(	0	0	0	1	0	)
$\boldsymbol{h_4}$	(	1	0	0	1	0	)
$\boldsymbol{h_5}$	(	0	1	0	1	1	)
W	(	3	2	0	5	2	)
W_{avg}	(	.60	.40	.00	1.00	.40	)
Y	(	1	0	0	1	0	)

Table 1 – Example of MajorityVoteEnsemble fusion function for classifying an instance of the Image data set with an ensemble consisting of 5 multi-label classifiers. Each instance of the data set represents an image that can be classified according to five possible labels: Desert, Mountains, Sea, Sunset and Trees. If the average of the binary decisions for each label is larger than the threshold, the label is chosen. In this example, a user defined value of threshold $t = 0.5$ defines the labels present in the final classification result. The label set ultimately assigned to the image is Desert + Sunset.

2.3.1.2 Minimum/Maximum

One possibility when using fixed combination rules is the selection of maximum or minimum values. Such rules find almost no practical application as they ignore much of the information generated by the classifiers, focusing on extreme values that are very susceptible to outliers. Those rules are exemplified in table 2.

$\mathcal{L} =$	{	Desert	Mountains	Sea	Sunset	Trees	}
$\boldsymbol{h_1}$	(	0.8	0.3	0.1	0.6	0.7	)
$\boldsymbol{h_2}$	(	0.7	0.8	0.1	0.8	0.3	)
$\boldsymbol{h_3}$	(	0.2	0.4	0.2	0.8	0.2	)
$\boldsymbol{h_4}$	(	0.9	0.2	0.1	0.9	0.4	)
$\boldsymbol{h_5}$	(	0.3	0.8	0.2	0.8	0.8	)
min	(	0.2	0.2	0.1	**0.6**	0.2	)
max	(	**0.9**	**0.8**	0.2	**0.9**	**0.8**	)

Table 2 – Example of *min* and *max* as fusion functions for classifying an instance of the Image data set with an ensemble consisting of 5 multi-label classifiers. A user defined value of threshold $t = 0.5$ defines the labels present in the final classification result.

2.3.1.3 Average

This fusion function, named MeanEnsemble (ME), computes the average support value for each class and only includes in the final result classes with an average support value greater than a threshold t. The ith classifier $\boldsymbol{h}_i \in \mathcal{H}$ predicts a vector $\boldsymbol{h}_i(\boldsymbol{x}) = [d_{i,1}(\boldsymbol{x}), \dots, d_{i,m}(\boldsymbol{x})] \in [0,1]^m$. The MeanEnsemble method stores the sums of the predictions of each individual classifier in a vector $W = (\lambda_1, \dots, \lambda_m) \in \mathbb{I}_m$ such that

$\lambda_j = \sum_{i=1}^{c} d_{i,j}$. W is then averaged to $W_{avg} \in \mathbb{R}_m$, which represents a distribution of votes for each label in the $[0, 1]$ interval. Using a score-based method (GHARROUDI; ELGHAZEL; AUSSEM, 2015) a threshold t is used to choose the final multi-label set Y such that $\ell_j \in Y$ when $\frac{\lambda_j}{c} \geq t$. Hence the relevant labels in Y represent the final multi-label prediction. This scheme is represented in algorithm 1 and exemplified in table 3 for the Image data set. The sum of support values of all classifiers are stored in a vector and then averaged according to the number of classifiers. If the average support value for the label is greater than the threshold value, the label is assigned to the instance.

Algorithm 1: The MeanEnsemble multi-label fusion function.

```
Function $Y \leftarrow$ ClassifyME($\boldsymbol{x}$, $\mathcal{H}$, $\mathcal{L}$, $t$)
    Input: Test instance x, Ensemble of classifiers $\mathcal{H}$, set of possible labels $\mathcal{L}$,
           threshold $t$
    Output: label set $Y$
    for $j = 1$ to $|\mathcal{L}|$ do
        $W[j] \leftarrow 0$;
    forall $h \in \mathcal{H}$ do
        for $j = 1$ to $|\mathcal{L}|$ do
            $W[j] \leftarrow W[j] + h(\boldsymbol{x})[j]$
    for $j = 1$ to $|\mathcal{L}|$ do
        $W_{avg}[j] \leftarrow \frac{W[j]}{|\mathcal{H}|}$;
        if $W_{avg}[j] > t$ then
            $Y[j] \leftarrow 1$;
        else
            $Y[j] \leftarrow 0$;
    return $Y$
```

$\mathcal{L} =$	{	Desert	Mountains	Sea	Sunset	Trees	}
$\boldsymbol{h_1}$	(	0.8	0.3	0.1	0.6	0.7	)
$\boldsymbol{h_2}$	(	0.7	0.8	0.1	0.8	0.3	)
$\boldsymbol{h_3}$	(	0.2	0.4	0.2	0.8	0.2	)
$\boldsymbol{h_4}$	(	0.9	0.2	0.1	0.9	0.4	)
$\boldsymbol{h_5}$	(	0.3	0.8	0.2	0.8	0.8	)
W	(	2.9	2.5	0.7	3.9	2.4	)
W_{avg}	(	0.58	0.5	0.14	0.78	0.48	)
Y	(	1	1	0	1	0	)

Table 3 – Example of MeanEnsemble fusion function for classifying an instance of the Image data set with an ensemble consisting of 5 multi-label classifiers. The support values for each label are averaged and the final decision is taken. In this example, a user defined value of threshold $t = 0.5$ defines the labels present in the final classification result. The resulting label set for this instance is Desert + Mountains + Sunset.

ECC frequently is also used with the MeanEnsemble (ME) as its fusion function (GUO et al., 2016).

2.3.2 Trainable Fusion Functions

Trainable fusion methods, unlike untrainable methods, follow a strategy of learning and adapting to each problem specifically from previously observed examples. The idea behind this strategy is that the fusion methods can learn, through the training process, an efficient way to combine the outputs and generate a better output. This strategy introduces an additional computation cost in the classification process performed by the ensemble, but this trade-off is often small compared to the total cost for training and classification performed by the ensemble member classifiers. Decision Templates, a method fundamental to understanding UDDTECC, falls into this category of fusion methods and is explained in more detail below. Another trainable fusion method is based on the stacking strategy. An explanation of this generic method is also provided.

2.3.2.1 Decision Templates

The Decision Templates method (KUNCHEVA; BEZDEK; DUIN, 2001) is a single-label classifier fusion function. The main idea consists of using the DP matrix in its entirety thus avoiding missing information that can improve the overall performance of the ensemble. Let $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ be the single-label training set. An ensemble of classifiers is trained using $\mathcal{S}$. For each label $\ell_j \in L$ a DT matrix is computed:

$$\mathrm{DT}_j = \frac{\sum_{k=1}^{n} f(y_k, \ell_j) \cdot \mathrm{DP}(\boldsymbol{x}_k)}{\sum_{k=1}^{n} f(y_k, \ell_j)}, \tag{2.4}$$

where $f(y_k, \ell_j)$ is a function that assumes the value 1 if the label of y_k is ℓ_j. Otherwise, the function is 0.

DT_j can be defined as the expected value of $\mathrm{DP}(\boldsymbol{x})$ for the ℓ_j label. The final support value μ_j to ℓ_j is directly proportional to the level of *similarity* between the matrices $\mathrm{DP}(\boldsymbol{x})$ and DT_j.

The similarity function based on the Euclidean distance recommended in (ROGOVA, 1994) is calculated as:

$$\mu_j(\boldsymbol{x}) = \frac{1 - ||\mathrm{DP}(\boldsymbol{x}) - \mathrm{DT}_j||_2^2}{\sum_{i=1}^{m}(1 - ||\mathrm{DP}(\boldsymbol{x}) - \mathrm{DT}_i||_2^2)}. \tag{2.5}$$

where $||.||_2$ is the entry-wise matrix norm, or equivalently the Euclidean distance between the linearized DP and DT (all lines have been sequentially assembled into a $c \times m$ vector). After the support value ϕ_j has been obtained for each class, a further evaluation takes the final decision. Usually, in a single-label crisp classification task, the highest

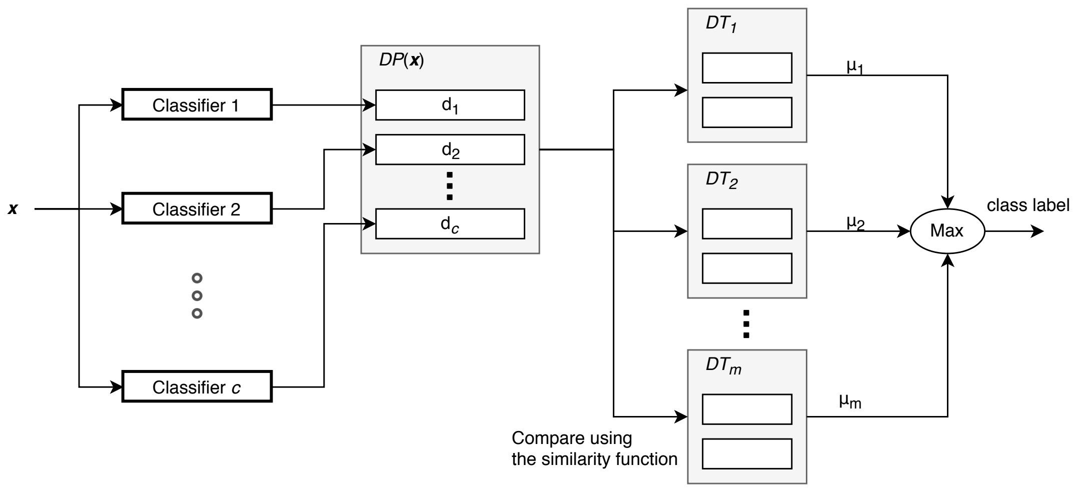


Figure 1 – *Decision Templates* fusion process for evaluating feature vector $\boldsymbol{x}$. An ensemble of c classifiers outputs the values that compose the *Decision Profile* (DP($\mathbf{x}$)). A similarity function computes the similarity values between DP($\mathbf{x}$) and DT$_j$ for all m labels. The resulting label is selected by the largest similarity value.

score defines the assigned class. In a binary classification task, the first class is chosen, if $\phi_1(\boldsymbol{x}) \geq \phi_2(\boldsymbol{x})$. In a multi-label problem, this strategy is not applicable. Figure 1 illustrates the application of the DT method for defining the classification of feature vector $\boldsymbol{x}$.

Algorithm 2 describes the process of composing a Decision Profile matrix. *Predict* is a function that has the unseen instance and a classifier as inputs, and outputs a $|\mathcal{L}|$-dimensional vector containing the support value for each label $j \in \mathcal{L}$. For a given instance, the support vector of each classifier is collected and its values are used to compose the DP matrix.

Algorithm 2: Decision Profile.

```
Function $DP \leftarrow ComputeDP(\boldsymbol{x}, \mathcal{H}, \mathcal{L})$
    Input: feature vector of an unseen instance $\boldsymbol{x}$, trained set of classifiers $\mathcal{H}$, set
           of possible labels $\mathcal{L}$
    Output: the resulting Decision Profile $DP$
    $DP$ is an $|\mathcal{H}| \times |\mathcal{L}|$ matrix;
    for $i = 1$ to $|\mathcal{H}|$ do  // all classifiers
        $s \leftarrow Predict(\boldsymbol{x}, \mathcal{H}_i)$ ;
        for $j = 1$ to $|\mathcal{L}|$ do  //all labels
            $DP_{i,j} \leftarrow s_j$ ;
        end
    end
    return DP
end
```

Algorithm 3 shows how to compute single label Decision Templates (DT) matrices. *ComputeDT* computes a Decision Profile (DP) matrix for each training instance. A DT matrix for a given label is obtained by averaging the values of all the DP matrices for the training instances associated with that label. Each DP is added to the DT matrix of the label to which the instance belongs. Then, the DT matrix is divided by the number of training instances belonging to the respective label (counting is performed by function *countByLabel*).

Algorithm 3: Single-label Decision Templates.

```
Function $\mathcal{DT} \leftarrow$ ComputeDT($\mathcal{S}$,$\mathcal{H}$,$\mathcal{L}$)
    Input: Single-label training data set $\mathcal{S}$, set of trained classifiers $\mathcal{H}$, set of
           possible labels $\mathcal{L}$
    Output: set of Decision Templates (one for each label) $\boldsymbol{DT}$
    for $(\boldsymbol{x}, y) \in \mathcal{S}$ do
        $DP \leftarrow ComputeDP(\boldsymbol{x}, \mathcal{H}, \mathcal{L})$ ;
        $DT_y \leftarrow DT_y + DP$ ;
    for $j = 1$ to $|\mathcal{L}|$ do  // all labels
        $cl \leftarrow countByLabel(\mathcal{S}, j)$ ;
        $DT_j \leftarrow \frac{DT_j}{cl}$ ;
    return $DT$
```

Algorithm 4 combines the classifier outputs in order to assign a label to an instance. The *ClassifyDT* function computes a DP for the test instance. According to the similarity function, the label ℓ associated with the most similar DT of the test instance DP is chosen as output.

Algorithm 4: Single-label Decision Templates Classification.

```
Function $l \leftarrow$ ClassifyDT($\boldsymbol{x}$,$\mathcal{H}$,$\boldsymbol{DT}$,$\mathcal{F}$,$\mathcal{L}$)
    Input: feature vector of an unseen instance $\boldsymbol{x}$, trained set of classifiers $\mathcal{H}$, set
           of Decision Templates - one for each possible label $\boldsymbol{DT}$, similarity
           function $\mathcal{F}$, set of possible labels $\mathcal{L}$
    Output: label $l$ assigned to instance $\boldsymbol{x}$
    $DP \leftarrow ComputeDP(\boldsymbol{x}, \mathcal{H}, \mathcal{L})$ ;
    $similarity \leftarrow -\infty$ ;
    $l \leftarrow 0$ ;
    for $j = 1$ to $|\mathcal{L}|$ do  // all labels
        $s \leftarrow \mathcal{F}(DP, \boldsymbol{DT}_j)$ ;
        if $s > similarity$ then
            $similarity \leftarrow s$
            $l \leftarrow j$
        end
    end
    return $l$
end
```

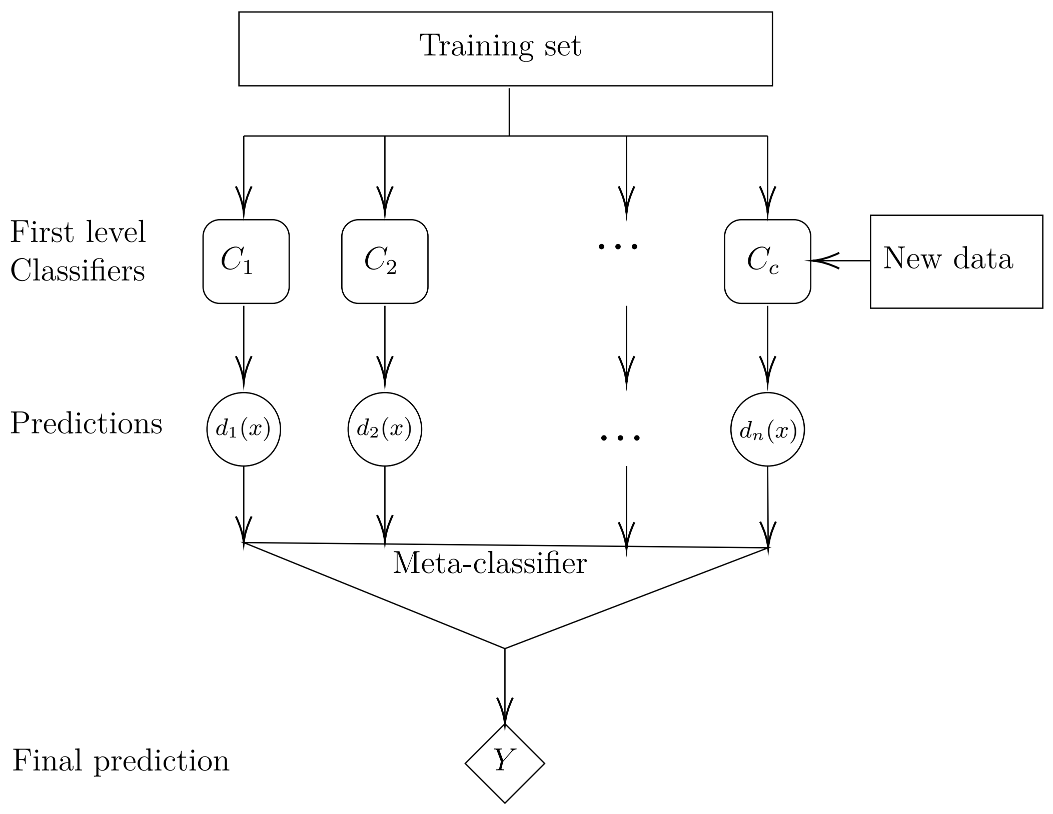


Figure 2 – The Stacking Classifier Ensemble Process Diagram.

2.3.2.2 Stacking

Stacking consists of the technique of combining ensembles in which the final classification result is obtained through a meta classifier. The ensemble member classifiers, also called first-level classifiers, are trained using the training set and the meta classifier is trained using the results of these methods.

The classification results generated by all first-level classifiers are combined into a single vector that is interpreted by the second-level classifier (or meta-classifier) as the problem features. The meta-classifier is trained using these meta-features and assigned the job of generating the final classification result for the problem. This fusion process is summarized in algorithm 5 and illustrated in fig. 2.

The stacking strategy relies on the strength of each individual estimator using its output as input to a final estimator. The outputs of the classifiers form instances that will be used as input data for a meta-classifier that generates the final classification result.

Any multi-label learner can be used as a meta-classifier for the Stack method, but in this work the chosen method was the Classifier Chains.

Algorithm 5: Stacking. Adapted from (TANG; ALELYANI; LIU, 2014).

```
Function $\boldsymbol{Y} \leftarrow Stack(D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n})$
    Input: Training data $D$
    Output: Ensemble of classifiers $\mathcal{H}$
    $c \leftarrow |\mathcal{H}|$;
    Step 1: Learn first level classifiers
    for $i = 1$ to $c$ do
        Learn a base classifier $h_i$ based on $D$;
    Step 2: Construct new data sets from $D$
    for $i = 1$ to $n$ do
        Construct a new data set that contains $\{\boldsymbol{x}'_i, \boldsymbol{y}_i\}$, where
          $\boldsymbol{x}'_i = \{h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), \ldots, h_c(\boldsymbol{x}_i)\}$;
    Step 3: Learn a second-level classifier
    Learn a new classifier $h'$ based on the newly constructed data set
    return $\boldsymbol{Y} = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_c(\boldsymbol{x}))$
```

3 FUSION OF MULTI-LABEL CLASSIFIER ENSEMBLES USING DECISION TEMPLATES

This section details the operation of the DTECC method that adapts the DT method for the multi-label classifier fusion problem. This method has been shown to be superior to MV and ME strategies mainly on datasets with large number of instances (ROCHA; VAREJÃO; SEGATTO, 2022). Following the explanation of the original DTECC method, the proposed UDDTECC method is explained, which extends the operation of the DTECC incorporating information about correlated labels.

3.1 DTECC: Decision Templates for Ensemble of Classifier Chains

In the DTECC method, given an ensemble composed by c multi-label classifiers, the problem is transformed into m binary classification problems. The DTECC method condenses the information of the original $c \times m$ into $c \times 2$ DP matrices. Using the previously labeled multi-label training set $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, the DP for each label $\ell_j \in L, j = 1, \dots, m$ becomes:

$$\mathrm{DP}_j(\boldsymbol{x}) = \begin{pmatrix} d_{1,\ell_j}(\boldsymbol{x}) \\ \dots \\ d_{c,\ell_j}(\boldsymbol{x}) \end{pmatrix}, j = 1, \dots, m. \tag{3.1}$$

Using the previously labeled multi-label training set $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, two corresponding matrices are computed for each label $\ell_j \in L, j = 1, \dots, m$:

$$\mathrm{DT}_j = \frac{\sum_{i=1}^{n} f(\boldsymbol{y}_i, \ell_j)\mathrm{DP}_j(\boldsymbol{x}_i)}{\sum_{i=1}^{n} f(\boldsymbol{y}_i, \ell_j)}, \tag{3.2}$$

$$\mathrm{DT}_{\bar{j}} = \frac{|\sum_{i=1}^{n}(1 - f(\boldsymbol{y}_i, \ell_j))\mathrm{DP}_j(\boldsymbol{x}_i)|}{\sum_{i=1}^{n} |1 - f(\boldsymbol{y}_i, \ell_j)|}, \tag{3.3}$$

where $f(\boldsymbol{y}_i, \ell_j)$ is a function that takes value 1 if $\ell_j \in \boldsymbol{y}_i$. Otherwise, its value is 0.

DT_j represents the average DP matrices of the training instances associated with the label ℓ_j, whilst $\mathrm{DT}_{\bar{j}}$ represents the average DP of the instances with the complementary labels. The classification of each label is done separately according to the level of similarity.

This work adopts a similarity function in the fusion process based on equation 2.5 but removes the normalization:

$$\mu_j(\boldsymbol{x}) = 1 - ||\mathrm{DP}_j(\boldsymbol{x}) - \mathrm{DT}_j||_2^2 =$$
$$1 - \sum_{i=1}^{c}\sum_{\ell=1}^{m} [d_{i,\ell}(\boldsymbol{x}) - \mathrm{DT}_j(i,\ell)]^2 . \tag{3.4}$$

Normalization is removed only to reduce computational cost and does not change the final result, as the ordering of labels with respect to their similarities remains unchanged, regardless of normalization.

Thus, the level of similarity measured between $\mathrm{DP}(\boldsymbol{x})$ and both DT_j and $\mathrm{DT}_{\bar{j}}$ is computed using equation 3.4. Finally, an unseen instance $(\boldsymbol{x}, \boldsymbol{y})$ is labeled as ℓ_j if $\mu_j(\boldsymbol{x}) > \mu_{\bar{j}}(\boldsymbol{x})$. The DTECC architecture is represented in fig. 3.

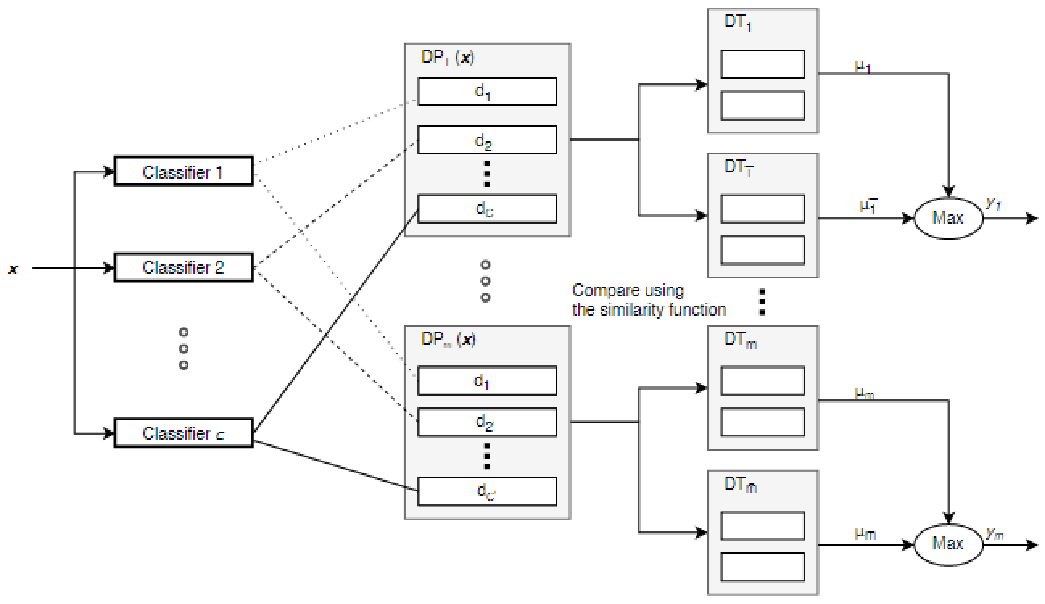
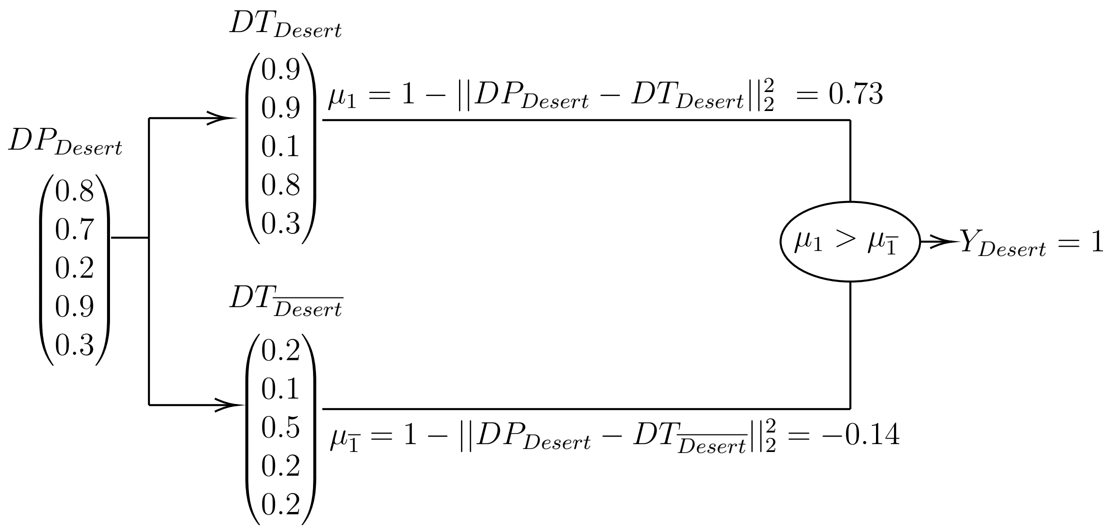


Figure 3 – (DTECC) *Decision Templates for Ensemble of Classifier Chains* fusion architecture. The output of c CC classifiers are combined into one final classification output.

Algorithm 6 shows how to compute the Ensemble of Classifier Chains Decision Templates. For each training instance, the results of the classifiers are compiled into a DP matrix, which is added to one of the two possible DT matrices for each label, according to the actual label of the instance. At the end of the process, the DT matrices are averaged. Function *ComputeDTECC* outputs two Decision Templates for each label instead of just one: a DT matrix for the positive training instances and another for the negative training instances. Algorithm 7 shows how to assign a label set $\boldsymbol{y}$ to an instance. The function *ClassifyDTECC* checks for each label whether the DP of instance x is more similar to DT of positive training instances or the DT of negative training instances. Fig. 4 shows an example of using the DTECC method in an instance of the Image data set for evaluating if the label Desert should be assigned to the multi-label set of the instance. The similarity



DT_{Desert}
$(0.9, 0.9, 0.1, 0.8, 0.3)$
$\mu_1 = 1 - ||DP_{Desert} - DT_{Desert}||_2^2 = 0.73$
DP_{Desert}
$(0.8, 0.7, 0.2, 0.9, 0.3)$
$\mu_1 > \mu_{\overline{1}}$
$Y_{Desert} = 1$
$DT_{\overline{Desert}}$
$(0.2, 0.1, 0.5, 0.2, 0.2)$
$\mu_{\overline{1}} = 1 - ||DP_{Desert} - DT_{\overline{Desert}}||_2^2 = -0.14$

Figure 4 – Example of classifier fusion using the DTECC method on an instance of the Image data set for the label Desert.

level of the DP_{Desert} matrix with DT_{Desert} and $DT_{\overline{Desert}}$ is measured, as $\mu_1 > \mu_{\overline{1}}$ the label is assigned to the instance. This procedure is performed for all labels in the problem: Desert, Mountains, Sea, Sunset and Trees. At the end of this procedure the Label set associated with the instance is determined.

Algorithm 6: Decision Templates for Ensemble of Classifier Chains.

```
Function $\mathcal{DT} \leftarrow$ ComputeDTECC($\mathcal{D}$,$\mathcal{H}$,$\mathcal{L}$)
    Input: Multi-label training data set $\mathcal{D}$, set of trained classifiers $\mathcal{H}$, set of
           possible labels $\mathcal{L}$
    Output: set of Decision Templates $\boldsymbol{DT}$, two for each possible label
    for $(\boldsymbol{x},\boldsymbol{y}) \leftarrow D$ do  // all training instances
        $DP \leftarrow ComputeDP(\boldsymbol{x}, \mathcal{H}, \mathcal{L})$ ;
        for $j = 1$ to $|\mathcal{L}|$ do  // all labels
            if $j \in \boldsymbol{y}$ then
                $\boldsymbol{DT}_j \leftarrow \boldsymbol{DT}_j + DP_j$
            else
                $\boldsymbol{DT}_{\overline{j}} \leftarrow \boldsymbol{DT}_{\overline{j}} + DP_j$
    for $j = 1$ to $|\mathcal{L}|$ do  // all labels
        $cl \leftarrow countByLabel(\mathcal{D}, j)$ ;
        $\boldsymbol{DT}_j \leftarrow \frac{\boldsymbol{DT}_j}{cl}$ ;
        $\boldsymbol{DT}_{\overline{j}} \leftarrow \frac{\boldsymbol{DT}_{\overline{j}}}{(|\mathcal{D}|-cl)}$
    return $\boldsymbol{DT}$
```

Algorithm 7: Decision Templates for Ensemble of Classifier Chains Classification.

```
Function $\boldsymbol{y} \leftarrow$ ClassifyDTECC($\boldsymbol{x}$,$\mathcal{H}$,$\boldsymbol{DT}$,$\mathcal{F}$,$\mathcal{L}$)
    Input: feature vector from an unseen instance $\boldsymbol{x}$, trained set of classifiers $\mathcal{H}$,
           set of Decision Templates $\boldsymbol{DT}$, similarity function $\mathcal{F}$, set of possible
           labels $\mathcal{L}$
    Output: Label set $\boldsymbol{y}$ assigned to the instance $\boldsymbol{x}$
    $DP \leftarrow ComputeDP(\boldsymbol{x}, \mathcal{H}, \mathcal{L})$ ;
    $\boldsymbol{y} \leftarrow \{\}$ ;
    for $j = 1$ to $|\mathcal{L}|$ do  // all labels
        $s_j \leftarrow \mathcal{F}(DP_j, \boldsymbol{DT}_j)$ ;
        $s_{\bar{j}} \leftarrow \mathcal{F}(DP_j, \boldsymbol{DT}_{\bar{j}})$ ;
        if $s_j > s_{\bar{j}}$ then
            $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \{\ell_j\}$
        end
    end
    return $\boldsymbol{y}$
end
```

3.2 UDDTECC: Unconditionally Dependent Decision Templates for Ensemble of Classifier Chains

For the proposed UDDTECC version, additional information regarding labels related to the label that is currently being classified is included. The classification process for a label l_j includes in its matrices DP_j and DT_j all labels correlated to l_j. Therefore, the only information known *a priori* is that the number of rows of the generated DP matrices will be equal to the number of classifiers that make up the set c and that the number of columns is in the range $[1, m]$. In the extreme cases, this method becomes equivalent to DTECC: If no correlation is identified among the labels the matrix becomes the same used in the DTECC method.

In order to identify unconditionally related label pairs Phi (ϕ) (COHEN et al., 2013) coefficients are used.

Given two labels ℓ_a and $\ell_b \in L$, and a contingency table for both labels as in Table 4, ϕ coefficient is calculated as:

	ℓ_b	$\neg\ell_b$
ℓ_a	A	B
$\neg\ell_a$	C	D

Table 4 – Contingency table for labels ℓ_a and ℓ_b

$$\phi(\ell_a, \ell_b) = \frac{AD - BC}{\sqrt{(A+B)(C+D)(A+C)(B+D)}} \tag{3.5}$$

Similarly, the Chi (χ^2) coefficient can also be used to determine unconditional relationships between pairs of labels. If $\chi^2 > 6.635$ values for a label pair, they are considered dependent at 99% confidence (GREENWOOD; NIKULIN, 1996).

$$\chi^2(\ell_a, \ell_b) = \frac{(AD - BC)^2(A + B + C + D)}{(A + B)(C + D)(A + C)(B + D)} \tag{3.6}$$

χ and ϕ coefficients follow the mathematical relation $\chi^2(\ell_a, \ell_b) = n \cdot \sqrt{\phi(\ell_a, \ell_b)}$, where n is the number of instances. The advantage of using the ϕ coefficient instead of χ as the threshold parameter for defining the correlations between the labels lies in the range defined by ϕ. As we are working with a 2×2 contingency matrix, the ϕ coefficient is within the -1 and 1 range while the χ coefficient can vary indefinitely according to the number of evaluated instances.

Like the DTECC, in the UDDTECC method, the DT_j and $\mathrm{DT}_{\bar{j}}$ matrices for each label are calculated using equation 3.2 and equation 3.3 respectively.

The difference between the proposed methods lies on the DP matrix of each label used for defining both DT_j and $\mathrm{DT}_{\bar{j}}$ as well the $\mathrm{DP}_j(\boldsymbol{x})$ matrix of the instance $\boldsymbol{x}$ to be classified. A user-defined threshold ϕ_t value defines whether the ϕ value is enough for the label pair to be considered correlated. Using the previously labeled multi-label training set $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, and $\boldsymbol{P}_j = \{p | p \in L \wedge |\phi(\ell_i, \ell_j)| > \phi_t\}$ representing the labels unconditionally dependent to ℓ_j, therefore the DP for each label $\ell_j \in L, j = 1, \ldots, m$ becomes:

$$\mathrm{DP}_j(\boldsymbol{x}) = \begin{pmatrix} d_{1,\ell_j}(\boldsymbol{x}) & d_{1,\ell_{p_1}}(\boldsymbol{x}) & \ldots \\ \ldots & \ldots & \ldots \\ d_{n,\ell_j}(\boldsymbol{x}) & d_{n,\ell_{p_1}}(\boldsymbol{x}) & \ldots \end{pmatrix}, j = 1, \ldots, m \;\; and \;\; \{p_1, \ldots\} = \boldsymbol{P}_j \tag{3.7}$$

Let's take the correlation matrix for the Scene dataset shown in fig. 5 as an example. Intuitively, the correlation matrix is symmetric because each pair of variables has to have the same relationship (correlation) whether its correlation is in the upper right triangle or the lower left triangle. If a value of $\phi_t = 0.20$ is set, the Mountain class would be considered unconditionally correlated with the Sunset and Urban classes in the fusion process. In this case, the vector of unconditionally related labels to Mountain would be $P_{Mountain} = \{Mountain, Sunset, Urban\}$ and the DP matrix for the label Mountain considered would become:

$$\mathrm{DP}_{Mountain}(\boldsymbol{x}) = \begin{pmatrix} d_{1,Mountain}(\boldsymbol{x}) & d_{1,Sunset}(\boldsymbol{x}) & d_{1,Urban}(\boldsymbol{x}) \\ \ldots & \ldots & \ldots \\ d_{c,Mountain}(\boldsymbol{x}) & d_{c,Sunset}(\boldsymbol{x}) & d_{c,Urban}(\boldsymbol{x}) \end{pmatrix} \tag{3.8}$$

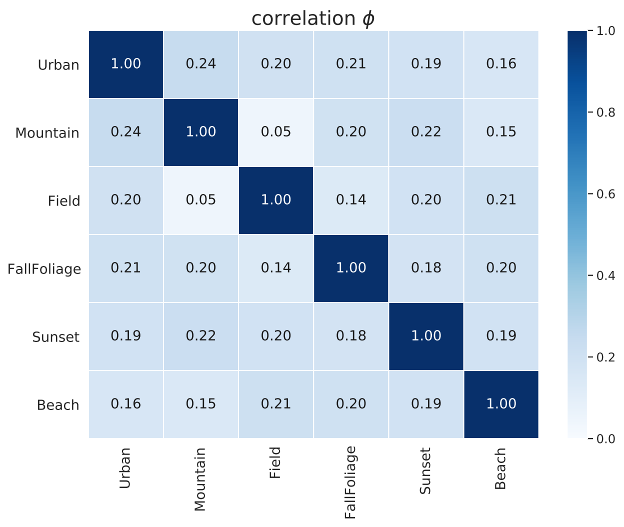


Figure 5 – Matrix of the ϕ coefficients correlation absolute values of the Scene dataset labels.

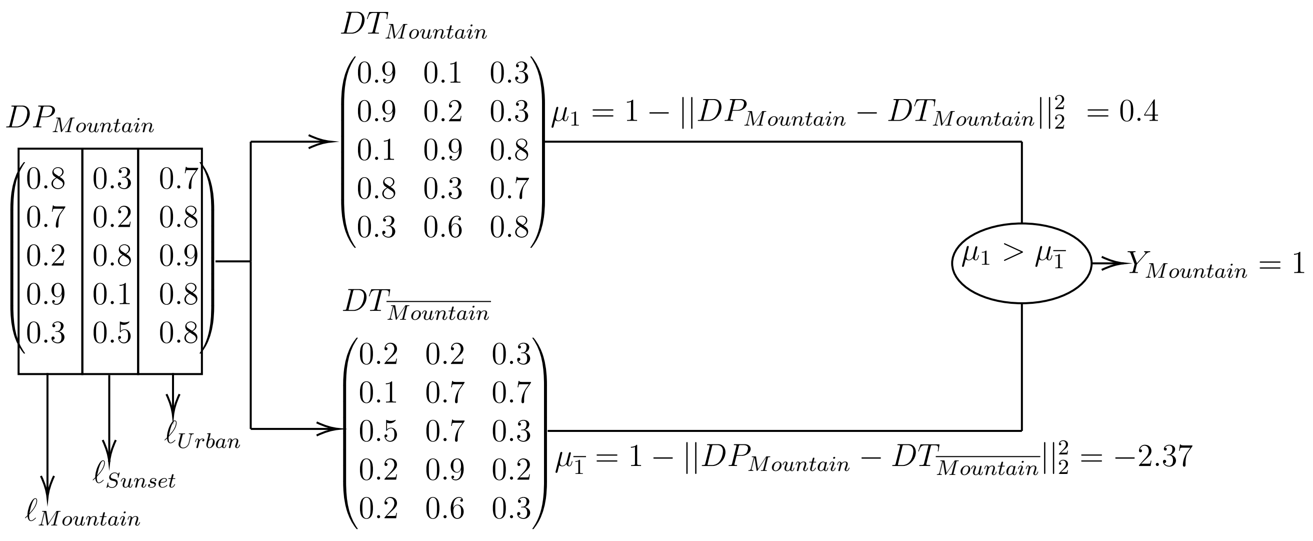

$DT_{Mountain}$
$\begin{pmatrix}0.9 & 0.1 & 0.3\\0.9 & 0.2 & 0.3\\0.1 & 0.9 & 0.8\\0.8 & 0.3 & 0.7\\0.3 & 0.6 & 0.8\end{pmatrix}$
$\mu_1 = 1 - ||DP_{Mountain} - DT_{Mountain}||_2^2 = 0.4$
$DP_{Mountain}$
$\begin{pmatrix}0.8 & 0.3 & 0.7\\0.7 & 0.2 & 0.8\\0.2 & 0.8 & 0.9\\0.9 & 0.1 & 0.8\\0.3 & 0.5 & 0.8\end{pmatrix}$
$\mu_1 > \mu_{\overline{1}}$
$Y_{Mountain} = 1$
$DT_{\overline{Mountain}}$
$\begin{pmatrix}0.2 & 0.2 & 0.3\\0.1 & 0.7 & 0.7\\0.5 & 0.7 & 0.3\\0.2 & 0.9 & 0.2\\0.2 & 0.6 & 0.3\end{pmatrix}$
ℓ_{Urban}
ℓ_{Sunset}
$\ell_{Mountain}$
$\mu_{\overline{1}} = 1 - ||DP_{Mountain} - DT_{\overline{Mountain}}||_2^2 = -2.37$

Figure 6 – Example of classifier fusion using the UDDTECC method on an instance of the Scene data set for the label Mountain.

and, given these conditions, the classification of an unseen instance is exemplified in fig. 6.

The similarity value between the DT$_j$ matrix of the label ℓ_j being evaluated and the DT$_j$ and DT$_{\bar{j}}$ matrices are calculated using equation 3.9. As in the original DTECC method, an instance is labeled as ℓ_j only if $\mu_j(\boldsymbol{x}) > \mu_{\bar{j}}(\boldsymbol{x})$.

$$\mu_j(\boldsymbol{x}) = 1 - \sum_{i=1}^{n} \sum_{k \in P_j} \left[d_{i,k}(\boldsymbol{x}) - \mathrm{DT}_j(i,k)\right]^2 . \tag{3.9}$$

3.2.1 UDDTECC Workflow

Eu acho que o workflow deveria incluir os seguintes passos: 1. dividir os dados disponiveis em folds; 2. realizar validação cruzada aninhada para definir os valores de phit candidatos (aqueles que ganham em cada rodada de teste) - workflow da figura 7 está incluído neste passo e deve ser apresentado no fluxograma; 3. realizar validação cruzada simples para cada um dos metodos usando os phit candidatos; 4. Escolher como phit aquele que obtiver o melhor desempenho; 5. treinar o sistema usando todos os dados disponiveis (base completa inclusive com exemplos usados para teste) com o phit escolhido; 6. aplicar o classificador para novos casos (de acordo com uso pretendido).

To use the UDDTECC method, one must follow the workflow presented at fig. 7. The first step consists of calculating the correlation matrix of ϕ values between the labels. The calculation of the correlation matrix can be performed independently of the training of the ensemble base classifiers.

The second stage is the tuning of the ϕ_t value. It is necessary that the training of the base classifiers are complete.

After training the classifiers and calculating the classification ou correlation??? correlation matrix, several phi_t values should be tested for selecting the value which achieves the best result of a given metric que métrica é essa? Vc está falando da metrica usada para avaliar o desempenho de classificação? a métrica usada para selecionar o phit com melhor desempenho.

The ϕ_t value can be tuned as a method hyperparameter using the following steps:

1. Partition the available training data into folds;

2. Perform nested cross-validation to define candidate ϕ_t values (those that win in each test round);

3. Perform simple cross-validation for each of the methods using the candidate ϕ_t;

4. Choose the best performing ϕ as the ϕ_t;

Once the value of ϕ_t is selected, the next step consists of training the system using all available data (complete data set including examples used for testing) with the chosen ϕ_t; In the last step we apply the classifier to unseen instances.

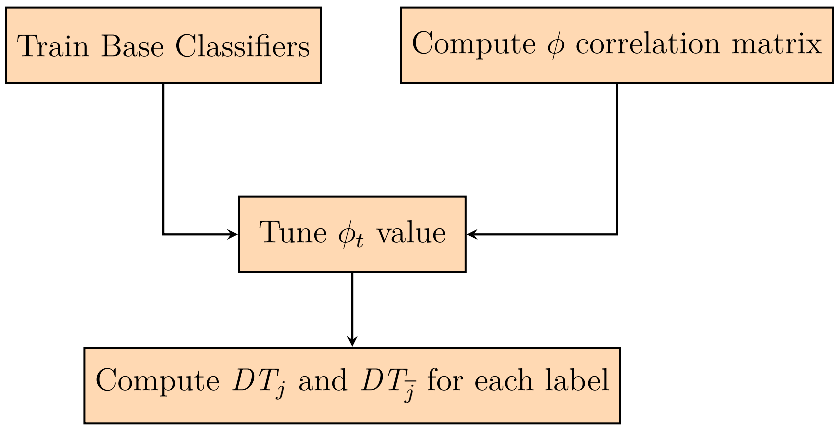

Train Base Classifiers
Compute ϕ correlation matrix
Tune ϕ_t value
Compute DT_j and $DT_{\bar{j}}$ for each label

Figure 7 – UDDTECC training workflow.

4 EXPERIMENTAL METHODOLOGY

This chapter details the methodology used in the experiments, describes the used datasets and also the metrics employed for evaluating the proposal.

4.1 Experimental Setup

All classifiers present in the experiments are evaluated using a 10-fold cross-validation technique. To ensure an unbiased evaluation of the methods, all the experiments use $c = 50$ classifiers in the ensemble, a threshold value of $t = 0.5$ and Naive Bayes as a single-label base classifier. While working on the multi-label feature ranking based on ensembles, (PETKOVIĆ; DŽEROSKI; KOCEV, 2020) showed that ensembles of 50 classifiers are enough to maximize most metrics. It has been also shown that ensemble learning works well when Naive Bayes is used as the base classifier (ANTONUCCI et al., 2013). Additional ECC parameters were set as the default values defined by the authors, e.g., size of each bag sample as a percentage of the training size is set to 100, and sampling with replacement is used.

Mulan (TSOUMAKAS et al., 2011) provides implementations for the Ensemble of Classifier Chains and for all evaluated metrics, those implementations were used to perform the experiments. Reproducibility and verifiability are very important issues when it comes to experimental results (RAUBER et al., 2020), in order to make the experiments in this work reproducible, the code and the results of the experiments are made available in the public repository <https://github.com/vfrocha/ddtecc>.

4.2 Datasets

Data sets from several domains were used in the experiments, some of their properties can be found on Table 5. Cardinality (Card) is defined as the average of the number of labels present in all examples in the dataset, and density (Dens) is the cardinality value divided by the number of labels. The number of labelsets present in the dataset divided by the maximum possible number of labelsets in the dataset is called label diversity (Div).

The datasets used were taken from the multilabel classification repository compiled by the KDIS (Knowledge Discovery and Intelligent Systems) research group at the Universidad de Córdoba available at <https://www.uco.es/kdis/mllresources/>. Due to the relatively high computational cost of training the classifier ensembles not all databases

Table 5 – Properties of the multi-label data sets used in the experiments: number of instances (n), number of features (d), number of labels (m), cardinality, label density and label diversity.

Dataset	Domain	n	d	m	Card	Dens	Div
3s-bbc1000	Text	352	1000	6	1.125	0.188	0.234
3s-guardian1000	Text	302	1000	6	1.126	0.188	0.219
3s-inter3000	Text	169	3000	6	1.142	0.190	0.172
CAL500	Music	502	68	174	26.044	0.150	1.000
Emotions	Music	593	72	6	1.868	0.311	0.422
Enron	Text	1702	1001	53	3.378	0.064	0.442
Genbase	Biology	662	1186	27	1.252	0.046	0.048
GnegativeGO	Biology	1392	1717	8	1.046	0.131	0.074
GpositiveGO	Biology	519	912	4	1.008	0.252	0.438
Image	Image	2000	294	5	1.236	0.247	0.625
Medical	Text	978	1449	45	1.245	0.028	0.096
Scene	Image	2407	294	6	1.074	0.179	0.234
VirusGO	Biology	207	749	6	1.217	0.203	0.266
Water-quality	Chemistry	1060	16	14	5.073	0.362	0.778
Yeast	Biology	2417	103	14	4.237	0.303	0.082

could be used in the experiments. Execution time and the amount of available memory were limiting factors regarding the number of databases used in the experiments.

4.3 Evaluation Metrics

This section presents the measures used to compare the algorithms in the experiments. Multi-label evaluation metrics are usually divided into two groups according to the way they are calculated: example-based metrics which are calculated for each test example separately, and then returning the mean value across the test set, and label-based which are calculated with respect to each class label separately, and then returning the macro/micro-averaged value across all class labels (ZHANG; ZHOU, 2013). The metrics used in this work are all example-based.

The Example-Based Accuracy measure is defined as the proportion of correctly classified labels on the total number of labels of each test sample.

$$\text{Accuracy}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{|\boldsymbol{y} \cap \boldsymbol{h}(\boldsymbol{x})|}{|\boldsymbol{y} \cup \boldsymbol{h}(\boldsymbol{x})|}. \tag{4.1}$$

Hamming Loss (SCHAPIRE; SINGER, 2000) computes the fraction of incorrectly predicted labels by the classifier $\boldsymbol{h}$. It is a loss metric, i.e, a classifier that only produces correct outputs would achieve a value of zero for this metric.

$$\text{HammingLoss}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m}\sum_{i=1}^{m}[y_i \neq h_i(\boldsymbol{x})] \tag{4.2}$$

Subset accuracy, or $\text{Subset}_{0/1}$ accuracy, (CHENG; HÜLLERMEIER; DEMBCZYNSKI, 2010) compares the label set predicted by $\boldsymbol{h}$ to the real label set $\boldsymbol{y}$ associated with the example. The set of labels predicted for the example must exactly match the corresponding set of labels in $\boldsymbol{y}$.

$$\text{Subset}_{0/1}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = [\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})] \tag{4.3}$$

It should be noted that Subset accuracy is a very strict metric since it penalizes nearly correct and completely wrong predictions in the same way. This metric may be useful, however, in certain applications where an exact classifier performance is of high priority.

Example-Based Precision is defined as the ratio of correct predicted labels, i.e., the ratio of labels predicted by $\boldsymbol{h}$ that actually occurs in $\boldsymbol{y}$:

$$Precision(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{|\boldsymbol{y} \cap \boldsymbol{h}(\boldsymbol{x})|}{|\boldsymbol{h}(\boldsymbol{x})|}. \tag{4.4}$$

Example-Based Recall represents the ratio of actual labels effectively predicted, i.e., the ratio of labels occurring in $\boldsymbol{y}$ that were actually predicted by $\boldsymbol{h}$:

$$Recall(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{|\boldsymbol{y} \cap \boldsymbol{h}(\boldsymbol{x})|}{|\boldsymbol{y}|}. \tag{4.5}$$

A classifier that has a high recall, but low precision values provides many labels, yet mostly incorrect. A classifier with high precision, but low recall values provides few correct labels, yet most actual labels are omitted.

Example-Based F-measure is a commonly applied multi-label metric that binds well with unbalanced data sets. F-measure is a function of two other multi-labels metrics: example-based precision and example-based recall.

The Example-Based F-measure, or F_1, is defined as the harmonic mean of the precision and recall values. The value of the F_1 measure metric is high only when recall and precision values are high.

$$F_1(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{4.6}$$

5 RESULTS AND DISCUSSION

This chapter presents the empirical study of the performance of the UDDTECC method and compares them to the MEECC (Mean Ensemble for ECC), MVECC (Majority Vote for ECC) and STACKECC (the proposed Stacking strategy for ECC) methods.

The evaluation and comparison of the classification methods is performed using a 10-fold cross-validation. The datasets are divided into 10 folds of equal or similar size, and in each of the 10 iterations of cross-validation, 9 of these folds are used for training and 1 is reserved for test, this procedure is performed in such a way that the validation fold is different in each iteration. At the end of the iterations, the 10 results collected for each metric are averaged and an aggregate result is obtained for that metric.

The use of the ECC method for all fusion schemes avoids the interference of other aspects than the fusion in the results. By keeping the same method and the same hyperparameters, one may be sure that only the classifier fusion method is responsible for the differences in the results.

5.1 DTECC

This section presents the empirical study of the performance of the DTECC method, and compares them to the MEECC and MVECC methods. The methods are evaluated against each other according to their average performance for all iterations of the 10-fold cross validation of each data set. All the experiments use $c = 50$ classifiers in the ensemble, a threshold value of $t = 0.5$ and Naive Bayes as a single-label base classifier.

Tables 6 to 11 show the results obtained by each method using Accuracy, Precision, Recall, F-measure, Subset Accuracy and Hamming Loss metrics, respectively. Altogether with the average performance, the rank achieved by the method is also presented between parenthesis. Best results are highlighted in bold face. At first glance, one may see that there is no method that is the best in all data sets when evaluated by one of the metrics. Indeed, every method is the best for at least two data sets in each metric.

name	DTECC	MEECC	MVECC
20NG	**0.4085(1.0)**	0.4066(2.0)	0.4056(3.0)
3sources_guardian1000	0.0778(2.0)	**0.0790(1.0)**	0.0773(3.0)
3sources_inter3000	0.0059(2.5)	**0.0088(1.0)**	0.0059(2.5)
CAL500	0.2186(3.0)	0.2210(2.0)	**0.2212(1.0)**
Emotions	**0.5329(1.0)**	0.5322(2.0)	0.5301(3.0)
Enron	**0.2484(1.0)**	0.2448(2.0)	0.2444(3.0)
Genbase	0.0982(3.0)	0.2980(2.0)	**0.3007(1.0)**
GnegativeGO	**0.8990(1.5)**	0.8983(3.0)	**0.8990(1.5)**
GpositiveGO	0.8950(2.0)	0.8931(3.0)	**0.8959(1.0)**
HumanGO	**0.6685(1.0)**	0.6633(3.0)	0.6639(2.0)
Image	**0.3579(1.0)**	0.3571(3.0)	0.3573(2.0)
Langlog	**0.1958(1.0)**	0.1948(2.0)	0.1946(3.0)
Medical	0.0253(3.0)	0.3714(2.0)	**0.3736(1.0)**
Reuters-K500	0.1463(2.0)	**0.1470(1.0)**	0.1453(3.0)
Scene	**0.4617(1.0)**	0.4600(2.0)	0.4589(3.0)
VirusGO	0.8121(2.0)	0.8008(3.0)	**0.8123(1.0)**
Water-quality	**0.3923(1.0)**	0.3870(2.0)	0.3869(3.0)
Yeast	0.4253(3.0)	0.4294(2.0)	**0.4309(1.0)**
Yelp	**0.5512(1.0)**	0.5453(3.0)	0.5466(2.0)

Table 6 – Results for Accuracy for the ECC using all the fusion schemes and data sets.

Table 6, table 7, table 9 and table 11 show that the DTECC method presents the largest number of wins for Accuracy, Precision, F-measure and Hamming Loss on the evaluated data sets. Table 8 shows that the MVECC method achieves the best results for the Recall metric, winning in 11 of the data sets, while the DTECC achieves almost the same performance, winning in 10 data sets. The only metric that the DTECC actually achieves an inferior performance is Subset Accuracy, shown in table 10, even though it was still able to win in 6 data sets.

name	DTECC	MEECC	MVECC
20NG	**0.4097(1.0)**	0.4077(2.0)	0.4067(3.0)
3sources_guardian1000	0.0856(2.0)	**0.0867(1.0)**	0.0851(3.0)
3sources_inter3000	0.0088(2.5)	**0.0118(1.0)**	0.0088(2.5)
CAL500	0.2802(3.0)	**0.2855(1.0)**	0.2834(2.0)
Emotions	0.5814(2.0)	**0.5818(1.0)**	0.5798(3.0)
Enron	**0.3633(1.0)**	0.3587(2.0)	0.3574(3.0)
Genbase	0.1044(3.0)	0.3323(2.0)	**0.3368(1.0)**
GnegativeGO	0.9123(2.0)	0.9120(3.0)	**0.9130(1.0)**
GpositiveGO	0.8988(2.0)	0.8969(3.0)	**0.8998(1.0)**
HumanGO	**0.6904(1.0)**	0.6880(2.0)	0.6872(3.0)
Image	**0.3850(1.0)**	0.3835(2.0)	0.3834(3.0)
Langlog	**0.1970(1.0)**	0.1959(2.0)	0.1956(3.0)
Medical	0.0254(3.0)	0.4217(2.0)	**0.4224(1.0)**
Reuters-K500	0.1522(2.0)	**0.1529(1.0)**	0.1511(3.0)
Scene	**0.4689(1.0)**	0.4672(2.0)	0.4661(3.0)
VirusGO	0.8298(2.0)	0.8210(3.0)	**0.8300(1.0)**
Water-quality	**0.4734(1.0)**	0.4600(2.0)	0.4583(3.0)
Yeast	0.5310(3.0)	**0.5419(1.0)**	0.5414(2.0)
Yelp	0.6514(3.0)	0.6564(2.0)	**0.6577(1.0)**

Table 7 – Results for Precision for the ECC using all the fusion schemes and data sets.

name	DTECC	MEECC	MVECC
20NG	**0.7496(1.0)**	0.7313(3.0)	0.7335(2.0)
3sources_guardian1000	**0.0961(2.0)**	**0.0961(2.0)**	**0.0961(2.0)**
3sources_inter3000	**0.0088(2.0)**	**0.0088(2.0)**	**0.0088(2.0)**
CAL500	0.5030(2.0)	0.5013(3.0)	**0.5077(1.0)**
Emotions	**0.7547(1.0)**	0.7494(3.0)	0.7516(2.0)
Enron	0.6460(2.0)	0.6449(3.0)	**0.6470(1.0)**
Genbase	**0.3967(1.0)**	0.2980(3.0)	0.3007(2.0)
GnegativeGO	**0.9407(1.0)**	0.9389(3.0)	0.9400(2.0)
GpositiveGO	0.9191(2.0)	0.9114(3.0)	**0.9210(1.0)**
HumanGO	**0.8388(1.0)**	0.8269(3.0)	0.8288(2.0)
Image	0.7963(3.0)	0.7993(2.0)	**0.8008(1.0)**
Langlog	0.8125(3.0)	0.8136(2.0)	**0.8158(1.0)**
Medical	**0.5338(1.0)**	0.4365(3.0)	0.4401(2.0)
Reuters-K500	0.8530(3.0)	0.8534(2.0)	**0.8543(1.0)**
Scene	0.8542(2.0)	0.8534(3.0)	**0.8551(1.0)**
VirusGO	0.8676(2.0)	0.8556(3.0)	**0.8679(1.0)**
Water-quality	0.7021(3.0)	0.7295(2.0)	**0.7303(1.0)**
Yeast	**0.6149(1.0)**	0.6097(3.0)	0.6139(2.0)
Yelp	**0.6712(1.0)**	0.6436(3.0)	0.6456(2.0)

Table 8 – Results for Recall for the ECC using all the fusion schemes and data sets.

name	DTECC	MEECC	MVECC
20NG	**0.4916(1.0)**	0.4854(2.0)	0.4851(3.0)
3sources_guardian1000	0.0859(2.0)	**0.0866(1.0)**	0.0854(3.0)
3sources_inter3000	0.0078(2.5)	**0.0098(1.0)**	0.0078(2.5)
CAL500	0.3447(3.0)	0.3477(2.0)	**0.3483(1.0)**
Emotions	**0.6295(1.0)**	0.6278(2.0)	0.6270(3.0)
Enron	**0.3649(1.0)**	0.3601(2.0)	0.3596(3.0)
Genbase	0.1594(3.0)	0.3071(2.0)	**0.3104(1.0)**
GnegativeGO	**0.9173(1.5)**	0.9164(3.0)	**0.9173(1.5)**
GpositiveGO	0.9043(2.0)	0.9005(3.0)	**0.9056(1.0)**
HumanGO	**0.7284(1.0)**	0.7221(3.0)	0.7227(2.0)
Image	**0.4818(1.0)**	0.4809(3.0)	0.4812(2.0)
Langlog	**0.2401(1.0)**	0.2386(2.0)	0.2382(3.0)
Medical	0.0483(3.0)	0.4072(2.0)	**0.4094(1.0)**
Reuters-K500	0.2092(2.0)	**0.2101(1.0)**	0.2081(3.0)
Scene	**0.5735(1.0)**	0.5720(2.0)	0.5715(3.0)
VirusGO	0.8365(2.0)	0.8261(3.0)	**0.8368(1.0)**
Water-quality	**0.5369(1.0)**	0.5333(2.0)	0.5329(3.0)
Yeast	0.5416(3.0)	0.5452(2.0)	**0.5466(1.0)**
Yelp	**0.6262(1.0)**	0.6163(3.0)	0.6178(2.0)

Table 9 – Results for F-measure for the ECC using all the fusion schemes and data sets.

name	DTECC	MEECC	MVECC
20NG	0.2170(3.0)	**0.2250(1.0)**	0.2224(2.0)
3sources_guardian1000	0.0562(2.5)	**0.0596(1.0)**	0.0562(2.5)
3sources_inter3000	0.0000(2.5)	**0.0059(1.0)**	0.0000(2.5)
CAL500	**0.0000(2.0)**	**0.0000(2.0)**	**0.0000(2.0)**
Emotions	0.2327(2.0)	**0.2361(1.0)**	0.2310(3.0)
Enron	0.0047(2.5)	0.0047(2.5)	**0.0053(1.0)**
Genbase	0.0000(3.0)	0.2764(2.0)	**0.2779(1.0)**
GnegativeGO	**0.8441(2.0)**	**0.8441(2.0)**	0.8441(2.0)
GpositiveGO	0.8670(2.5)	**0.8709(1.0)**	0.8670(2.5)
HumanGO	**0.5009(1.0)**	0.4984(3.0)	0.4987(2.0)
Image	0.0740(3.0)	**0.0760(1.0)**	0.0755(2.0)
Langlog	**0.1418(2.0)**	0.1418(2.0)	**0.1418(2.0)**
Medical	0.0000(3.0)	0.2710(2.0)	**0.2730(1.0)**
Reuters-K500	0.0382(2.0)	**0.0383(1.0)**	0.0372(3.0)
Scene	**0.1824(1.0)**	0.1803(2.0)	0.1778(3.0)
VirusGO	0.7395(2.0)	0.7250(3.0)	**0.7398(1.0)**
Water-quality	**0.0019(2.0)**	**0.0019(2.0)**	**0.0019(2.0)**
Yeast	0.1047(3.0)	**0.1088(1.0)**	0.1084(2.0)
Yelp	0.3123(3.0)	0.3215(2.0)	**0.3222(1.0)**

Table 10 – Results for Subset Accuracy for the ECC using all the fusion schemes and data sets.

name	DTECC	MEECC	MVECC
20NG	0.0848(3.0)	**0.0825(1.0)**	0.0831(2.0)
3sources_guardian1000	**0.2037(1.5)**	**0.2037(1.5)**	0.2059(3.0)
3sources_inter3000	0.2011(2.5)	**0.2001(1.0)**	0.2011(2.5)
CAL500	0.2909(3.0)	**0.2855(1.0)**	0.2892(2.0)
Emotions	0.2459(2.0)	**0.2454(1.0)**	0.2468(3.0)
Enron	**0.1745(1.0)**	0.1797(2.0)	0.1812(3.0)
Genbase	0.1615(3.0)	0.0341(2.0)	**0.0339(1.0)**
GnegativeGO	**0.0234(2.0)**	**0.0234(2.0)**	**0.0234(2.0)**
GpositiveGO	0.0462(2.5)	0.0462(2.5)	**0.0453(1.0)**
HumanGO	0.0560(3.0)	**0.0556(1.0)**	0.0558(2.0)
Image	**0.4115(1.0)**	0.4162(2.0)	0.4169(3.0)
Langlog	**0.2066(1.0)**	0.2093(2.0)	0.2111(3.0)
Medical	0.5824(3.0)	**0.0248(1.5)**	**0.0248(1.5)**
Reuters-K500	**0.2697(1.0)**	0.2775(2.0)	0.2797(3.0)
Scene	**0.2324(1.0)**	0.2337(2.0)	0.2345(3.0)
VirusGO	**0.0665(1.5)**	0.0674(3.0)	**0.0665(1.5)**
Water-quality	**0.3998(1.0)**	0.4233(2.0)	0.4239(3.0)
Yeast	0.3012(3.0)	**0.2924(1.0)**	0.2928(2.0)
Yelp	0.2161(3.0)	0.2118(2.0)	**0.2115(1.0)**

Table 11 – Results for Hamming Loss for the ECC using all the fusion schemes and data sets.

Learner	DTECC	MEECC	MVECC
Accuracy	**1.736842**	2.157895	2.105263
Precision	1.921053	**1.842105**	2.236842
Recall	1.789474	2.684211	**1.526316**
Subset accuracy	2.315789	**1.710526**	1.973684
F-measure	**1.736842**	2.157895	2.105263
Hamming Loss	2.052632	**1.710526**	2.236842

Table 12 – Average rank of the ECC methods using the evaluated fusion schemes for all data sets.

Arguably the most important result obtained in the experiments comes from comparing the average ranking results for the metrics. Table 12 shows the average rank of all methods for each evaluated metric. Whilst the DTECC fusion method version was superior on average for the Accuracy and F-measure metrics, MEECC was the best for the Hamming Loss, Precision and Subset accuracy metrics. MVECC only gives the best Recall results. However, among the three fusion methods evaluated, the DTECC method was the most consistent in its evaluation metrics results, not being among the two best metrics only for Subset accuracy. The MEECC method, on the other hand, despite the good results for the Precision, Subset Accuracy and Hamming Loss presented the worst results for all other metrics. Similarly, MVECC has the best average rankings for Recall but the worst for Precision and Hamming Loss. In addition, while the MEECC method was ranked best in most metrics, one may argue that results obtained in Accuracy and F-measure are the most relevant since they are two of the three most commonly used example-based metrics for evaluating multi-label classifiers (PEREIRA et al., 2018).

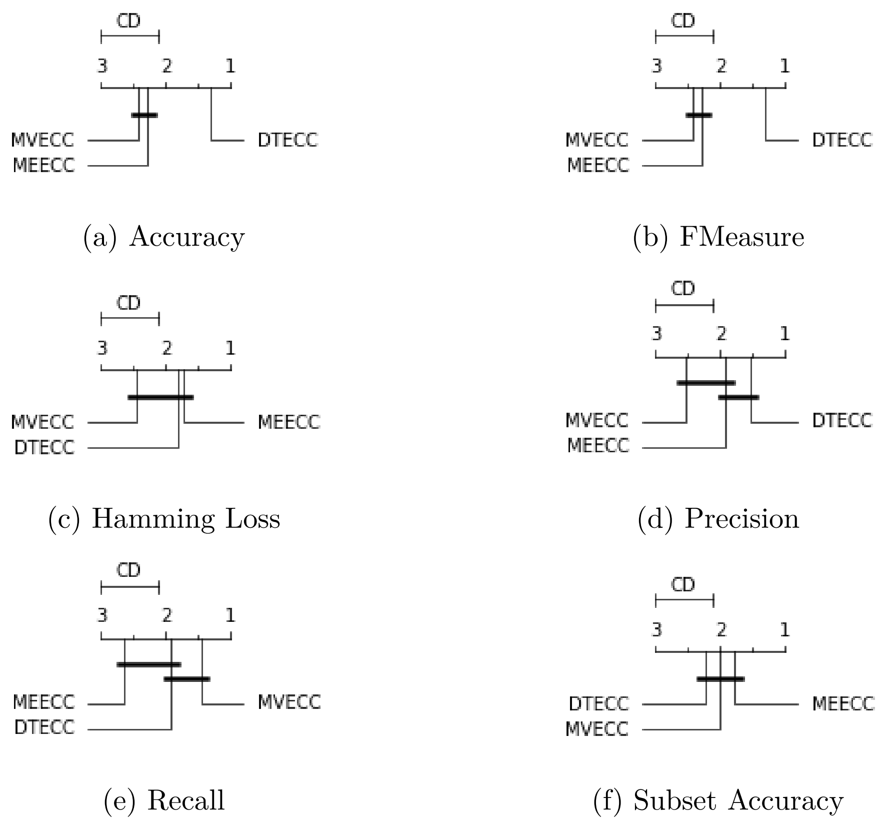
DTECC results are particularly positive in data sets with a reliably large number of instances. A possible explanation for the good results of the DTECC for large data sets is that a larger number of instances allows the creation of more accurate decision profiles compared with data sets with fewer instances. As the decision profiles are calculated through the training instances it is desirable that there is a reasonable number of instances for the creation of a reliable decision profile. Table 13 shows the average rank of all three methods considering only data sets with more than a thousand instances: 20NG, Enron, GnegativeGO, HumanGO, Image, Langlog, Reuters-K500, Scene, Water-quality, Yeast and Yelp. As one may see, DTECC now wins in three out of six metrics.

Learner	DTECC	MEECC	MVECC
Accuracy	**1.318182**	2.272727	2.409091
Precision	**1.545450**	1.909091	2.545455
Recall	1.909091	2.636364	**1.454545**
Subset accuracy	2.227273	**1.772727**	2.000000
F-measure	**1.318182**	2.272727	2.409091
Hamming Loss	1.818182	**1.727273**	2.454545

Table 13 – Average rank of the ECC methods using the evaluated fusion schemes for the largest data sets in terms of number of instances.

The positive results on data sets with a larger number of instances was confirmed by statistical tests. Whenever performed using all data sets they showed no statistically significant difference on the overall results. However, these differences were found when analyzing only the results on data sets with more than a thousand instances. The Friedman-Nemenyi test was applied. First the Friedman test is performed to reject the null hypothesis, and then proceed with a post-hoc analysis based on the Nemenyi method. For the Nemenyi test it was used $\alpha = 0.1$. Fig. 8 shows the critical difference diagrams. Wherever methods are connected by a horizontal line, it means they are not significantly different. It can be seen from the diagrams in fig. 8a and fig. 8b that the DTECC method is significantly better on the Accuracy and F-measure metrics in the data sets evaluated. The DTECC method is also the best on Precision, but it is not statistically better than the MEECC method, as shown in fig. 8d. For the metrics Hamming Loss, Recall and Subset Accuracy, respectively shown in fig. 8c, fig. 8e and fig. 8f, no statistically significant difference was found between the DTECC method and the other methods.

The fusion methods running times were also compared. Since the running times of



(a) Accuracy (b) FMeasure

(c) Hamming Loss (d) Precision

(e) Recall (f) Subset Accuracy

Figure 8 – Critical Difference (CD) diagrams obtained from the average ranking comparison of the methods over the 11 data sets with more than 1000 instances: 20NG, Enron, GnegativeGO, HumanGO, Image, Langlog, Reuters-K500, Scene, Water-quality, Yeast and Yelp.

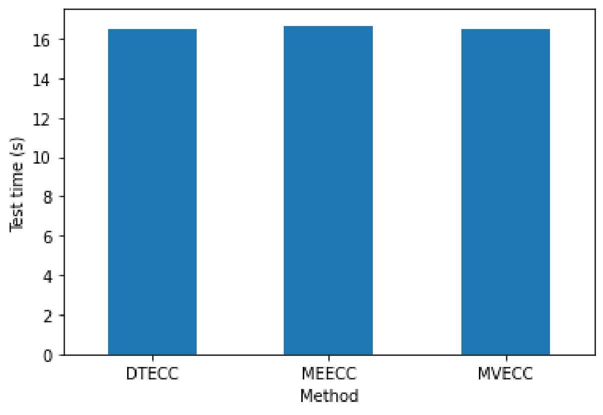


Figure 9 – Average of test runtime in seconds for each cross validation round on the CAL500 data set.

the fusion methods depends mainly on the number of labels, the CAL500 data set, which has the highest number of labels, was chosen to perform the runtime test. Fig. 9 reports the average running times for each cross-validation round in seconds. One may observe that there is virtually no difference in terms of execution time between the evaluated methods.

5.2 UDDTECC

For the UDDTECC method, in the experimental setup, the data set is divided into 10 folds, in each iteration of the cross-validation one of these folds is selected as the test set and the remaining folds are divided into validation and training sets in the ratio of 33% (3 folds) and 67% (6 folds) respectively as shown in fig. 10. Each UDDTECC model with $\phi_t \in [0.0, 0.25, 0.5, 0.75, 1]$ is then trained on the training set and validated on the validation set. The results for each model are then compared according to a pre-selected metric that is currently being evaluated (Accuracy, F-Measure, Subset Accuracy, and Hamming Loss). The best model is then selected and trained using both the inner training and the inner validation folds and finally applied to the test fold. After 10 iterations we have the results for all folds in the original data set for the selected metric. The final result of the experiment is the average of the metric over these 10 results.

As shown in the performance evaluation framework in algorithm 8 the total amount of n patterns is randomly split into K stratified folds. The intermediate layer trains the classifier with $(K-1)$ folds and tests the remaining fold in a usual K-fold cross validation (CV) procedure (outer CV loop). However, before the training of the classifier is performed, the best ϕ_t hyperparameter for the UDDTECC classifier is estimated in a tuning step. The tuning is achieved by an inner layer (inner CV loop). The inner CV loop further subdivides the training data of the outer CV loop into $K-1$ stratified folds. 3 of these folds are then used as internal validation and the remaining folds are used as internal training. Each evaluated ϕ_t value is trained on the internal training folds and validated on the internal validation fold. The ϕ_t value of the classifier with the best result on the evaluated metric is then selected and used in the outer CV loop.

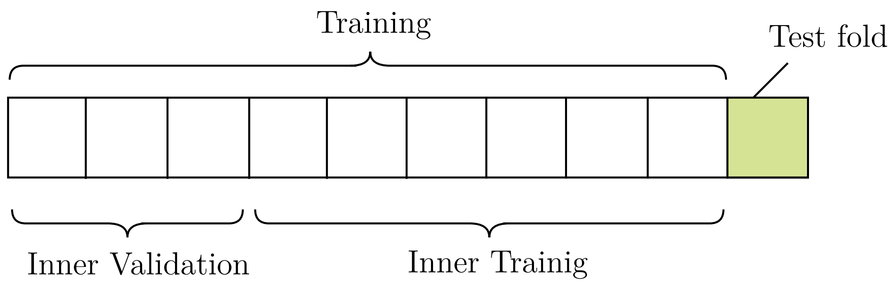


Figure 10 – Fold division used in the experiments.

To ensure an unbiased evaluation of the methods, all experiments use $n = 50$ classifiers in the ensemble, a threshold value of $t = 0.5$ and Naive Bayes as a single-label base classifier. Bayesian classifiers are often efficient when used as base classifiers of ensembles (ANTONUCCI et al., 2013). The STACKECC method uses Classifier Chains as its meta-classifer. Additional ECC parameters were defined as the default values defined by their authors, e.g., size of each bag sample as a percentage of the training size is set to 100, and the choice of sampling with replacement. By keeping the values of these parameters constant, one may get a better idea of the impact of the fusion scheme on the final classification result, as this is the only difference that exists between the methods used in the experiments.

Algorithm 8: The Performance Evaluation Framework.

```
Function $PM \leftarrow ModelPerformance(\mathcal{D},K)$
    Input: $n$ labelled patterns $\mathcal{D}$ from $c$ classes, number of folds $K$ (outer loop)
    Output: Make $K$ stratified folds. One fold is fixed as test, remaining folds are
            train and tune. Calculate performance criterion for each combination
            of round and fold and put into $R \times K$ performance matrix PM
    // generate $K$ stratified folds
    $f_k; k = 1, \ldots, K$;
    for $k = 1$ to $K$ do  // all folds
        // training set of $k$-th fold
        $\mathcal{T} \leftarrow \{f_1 \cup \ldots \cup f_K\} \setminus f_k$ ;
        // test set of $k$-th fold
        $\mathcal{V} \leftarrow f_k$ ;
        $\mathcal{P}^* \leftarrow$ Tuning($\mathcal{T}$);
        $\mathcal{C} \leftarrow$ trainclassifier( $\mathcal{T}, \mathcal{P}^*$ );
        $crit \leftarrow$ testclassifier($\mathcal{C}, \mathcal{V}$);
        PM(k)$\leftarrow crit$
    end
end
```

```
Function $\mathcal{P}^* \leftarrow Tuning(\mathcal{D})$
    Input: Data set $\mathcal{D}$
    Output: Optimal hyperparameter set $\mathcal{P}^*$ of classifier model
    initialize best criterion $maxcrit \leftarrow 0$;
    // three first folds are used as inner validation
    $\mathcal{V} \leftarrow \mathcal{D}$[:3];
    // remainder are used as inner training
    $\mathcal{T} \leftarrow \mathcal{D}$[3:];
    repeat // grid search for best hyperparameter set $\mathcal{P}^*$
        generate hyperparameter candidate set $\mathcal{P}$;
        $\mathcal{C} \leftarrow$ trainclassifier( $\mathcal{T}, \mathcal{P}$ );
        $crit \leftarrow$ testclassifier($\mathcal{C}, \mathcal{V}$);
        if $crit > maxcrit$ then
            $maxcrit \leftarrow crit$;
            $\mathcal{P}^* \leftarrow \mathcal{P}$
        end
    until $search\ completed$;
end
```

5.2.1 Overall Datasets Results

Tables 14, 15, 16 and 17 show the results obtained by each method using Accuracy, F-measure, Subset Accuracy and Hamming Loss metrics, respectively. Altogether with the average performance, the rank achieved by the method is also shown in parenthesis. The average ranking is presented in the last row of each table and the best results are highlighted in bold face. At first glance, one may see that there is no method that is the best in all data sets when evaluated by one of the metrics.

Dataset	UDDTECC	MEECC	MVECC	STACKECC
3sources_bbc1000	0.3219(1.0)	0.1526(4.0)	0.1547(3.0)	0.2852(2.0)
3sources_guardian1000	0.2840(1.0)	0.0790(3.0)	0.0773(4.0)	0.2243(2.0)
3sources_inter3000	0.1195(2.0)	0.0088(3.0)	0.0059(4.0)	0.1538(1.0)
CAL500	0.2187(3.0)	0.2210(2.0)	0.2212(1.0)	0.1991(4.0)
Emotions	0.5336(1.0)	0.5322(2.0)	0.5301(3.0)	0.5141(4.0)
Enron	0.2510(1.0)	0.2448(2.0)	0.2444(3.0)	0.0863(4.0)
Genbase	0.2002(4.0)	0.2980(3.0)	0.3007(2.0)	0.3819(1.0)
GnegativeGO	0.8990(1.5)	0.8983(3.0)	0.8990(1.5)	0.5649(4.0)
GpositiveGO	0.9113(1.0)	0.8931(4.0)	0.8959(3.0)	0.9085(2.0)
Image	0.3652(1.0)	0.3571(4.0)	0.3573(3.0)	0.3614(2.0)
Medical	0.0484(3.0)	0.3714(2.0)	0.3736(1.0)	0.0458(4.0)
Scene	0.4620(1.0)	0.4600(2.0)	0.4589(3.0)	0.4074(4.0)
VirusGO	0.8121(2.0)	0.8008(3.0)	0.8123(1.0)	0.6963(4.0)
Water-quality	0.3927(1.0)	0.3870(2.0)	0.3869(3.0)	0.3760(4.0)
Yeast	0.4261(3.0)	0.4294(2.0)	0.4309(1.0)	0.3821(4.0)
Avg. Rank	**1.766667**	2.733333	2.433333	3.066667

Table 14 – Results for Accuracy for the ECC using all the fusion schemes and data sets.

We note from Table 14 that the UDDTECC method obtained the best Accuracy result in 9 of the 15 datasets evaluated. In second place we have the MVECC method with 5 wins, which, together with consistent results in the other datasets, gave it the second best average ranking for Accuracy. The MEECC method did not obtain the best result in any of the evaluated datasets, but its average results in most datasets gave it a better average ranking than the STACKECC method, which, despite obtaining the best result in two of the evaluated datasets, was the worst overall.

Dataset	UDDTECC	MEECC	MVECC	STACKECC
3sources_bbc1000	0.3979(1.0)	0.1669(4.0)	0.1699(3.0)	0.3618(2.0)
3sources_guardian1000	0.3360(1.0)	0.0866(3.0)	0.0854(4.0)	0.3053(2.0)
3sources_inter3000	0.1648(2.0)	0.0098(3.0)	0.0078(4.0)	0.2238(1.0)
CAL500	0.3446(3.0)	0.3477(2.0)	0.3483(1.0)	0.3217(4.0)
Emotions	0.6260(3.0)	0.6278(1.0)	0.6270(2.0)	0.6175(4.0)
Enron	0.3671(1.0)	0.3601(2.0)	0.3596(3.0)	0.1554(4.0)
Genbase	0.2915(4.0)	0.3071(3.0)	0.3104(2.0)	0.4339(1.0)
GnegativeGO	0.9166(2.0)	0.9164(3.0)	0.9173(1.0)	0.6841(4.0)
GpositiveGO	0.9229(1.0)	0.9005(4.0)	0.9056(3.0)	0.9210(2.0)
Image	0.4930(1.0)	0.4809(3.0)	0.4812(2.0)	0.4801(4.0)
Medical	0.0911(3.0)	0.4072(2.0)	0.4094(1.0)	0.0838(4.0)
Scene	0.5737(1.0)	0.5720(2.0)	0.5715(3.0)	0.5416(4.0)
VirusGO	0.8334(2.0)	0.8261(3.0)	0.8368(1.0)	0.7708(4.0)
Water-quality	0.5371(1.0)	0.5333(2.0)	0.5329(3.0)	0.5208(4.0)
Yeast	0.5445(3.0)	0.5452(2.0)	0.5466(1.0)	0.5028(4.0)
Avg. Rank	**1.933333**	2.600000	2.266667	3.200000

Table 15 – Results for F-Measure for the ECC using all the fusion schemes and data sets.

The results of the F-Measure metric are presented in table 15. UDDTECC again proved to be the best both in terms of average ranking and number of best results, outperforming the others in 7 of the 15 bases evaluated. MVECC kept the second average ranking with 5 wins and consistent results in the other datasets. STACKECC was the best in 2 of the evaluated datasets but often showed the worst results, which put it in the last position behind MEECC method.

Dataset	UDDTECC	MEECC	MVECC	STACKECC
3sources_bbc1000	0.1786(1.0)	0.1162(2.5)	0.1162(2.5)	0.0996(4.0)
3sources_guardian1000	0.1423(1.0)	0.0596(2.0)	0.0562(3.0)	0.0528(4.0)
3sources_inter3000	0.0180(1.0)	0.0059(2.5)	0.0000(4.0)	0.0059(2.5)
CAL500	0.0000(2.5)	0.0000(2.5)	0.0000(2.5)	0.0000(2.5)
Emotions	0.2326(2.0)	0.2361(1.0)	0.2310(3.0)	0.1940(4.0)
Enron	0.0094(1.0)	0.0047(3.0)	0.0053(2.0)	0.0000(4.0)
Genbase	0.0167(4.0)	0.2764(2.0)	0.2779(1.0)	0.2388(3.0)
GnegativeGO	0.8441(2.0)	0.8441(2.0)	0.8441(2.0)	0.2443(4.0)
GpositiveGO	0.8805(1.0)	0.8709(2.5)	0.8670(4.0)	0.8709(2.5)
Image	0.0720(4.0)	0.0760(2.0)	0.0755(3.0)	0.0860(1.0)
Medical	0.0000(4.0)	0.2710(2.0)	0.2730(1.0)	0.0010(3.0)
Scene	0.1824(1.0)	0.1803(2.0)	0.1778(3.0)	0.0657(4.0)
VirusGO	0.7395(2.0)	0.7250(3.0)	0.7398(1.0)	0.4745(4.0)
Water-quality	0.0019(2.5)	0.0019(2.5)	0.0019(2.5)	0.0019(2.5)
Yeast	0.1088(1.5)	0.1088(1.5)	0.1084(3.0)	0.0732(4.0)
Avg. Rank	**2.033333**	2.200000	2.500000	3.266667

Table 16 – Results for Subset Accuracy for the ECC using all the fusion schemes and data sets.

The Subset Accuracy results presented in table 16 show more balanced results between the UDDTECC and MEECC methods. Still the UDDTECC method stood out as the best in terms of average result. MVECC obtained the third best average ranking, doing considerably better than the STACECC method in most of the datasets evaluated. Observing the relatively low results we note the difficulty of optimizing this metric on datasets with large numbers of labels for all the methods evaluated. The CAL500 dataset, which has 174 labels, makes this difficulty evident with all methods obtaining the same value of 0.0, i.e., no total hits.

Dataset	UDDTECC	MEECC	MVECC	STACKECC
3sources_bbc1000	0.2174(2.0)	0.2155(1.0)	0.2179(3.0)	0.3433(4.0)
3sources_guardian1000	0.2037(1.5)	0.2037(1.5)	0.2059(3.0)	0.4411(4.0)
3sources_inter3000	0.2011(2.5)	0.2001(1.0)	0.2011(2.5)	0.4643(4.0)
CAL500	0.2899(3.0)	0.2855(1.0)	0.2892(2.0)	0.3737(4.0)
Emotions	0.2468(2.5)	0.2454(1.0)	0.2468(2.5)	0.2684(4.0)
Enron	0.1737(1.0)	0.1797(2.0)	0.1812(3.0)	0.6141(4.0)
Genbase	0.1704(4.0)	0.0341(2.0)	0.0339(1.0)	0.1251(3.0)
GnegativeGO	0.0234(2.0)	0.0234(2.0)	0.0234(2.0)	0.1405(4.0)
GpositiveGO	0.0443(1.5)	0.0462(4.0)	0.0453(3.0)	0.0443(1.5)
Image	0.4005(2.0)	0.4162(3.0)	0.4169(4.0)	0.3929(1.0)
Medical	0.5488(4.0)	0.0248(1.5)	0.0248(1.5)	0.5162(3.0)
Scene	0.2324(1.0)	0.2337(2.0)	0.2345(3.0)	0.2619(4.0)
VirusGO	0.0665(1.5)	0.0674(3.0)	0.0665(1.5)	0.1172(4.0)
Water-quality	0.3998(1.0)	0.4233(3.0)	0.4239(4.0)	0.4111(2.0)
Yeast	0.3018(3.0)	0.2924(1.0)	0.2928(2.0)	0.3615(4.0)
Avg. Rank	2.166667	**1.933333**	2.533333	3.366667

Table 17 – Results for Hamming Loss for the ECC using all the fusion schemes and data sets.

The results presented in table 17 for the Hamming loss metric differ largely from the others. In this metric, unlike the other results, the UDDTECC method ranked second and the MEECC method achieved the best average result. Nine best results were obtained by MEECC versus seven obtained by the UDDTECC method. The MVECC and STACKECC methods ranked third and fourth on average, respectively.

Table 18 – Average rank of the ECC methods using the evaluated fusion schemes for all datasets.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
UDDTECC	**1.766667**	**1.933333**	2.166667	**2.033333**
MEECC	2.733333	2.600000	**1.933333**	2.200000
MVECC	2.433333	2.266667	2.533333	2.500000
STACKECC	3.066667	3.200000	3.366667	3.266667

Table 18 shows the average rank of all methods for each evaluated metric. The UDDTECC fusion method version was superior on average for the Accuracy, F-measure and Subset Accuracy metrics, and it ranks second in Hamming Loss. The only method other than UDDTECC that achieved the top rank in a metric was the MEECC method in the Hamming Loss metric. The MVECC method presented consistent results, ranking second or third depending on the metric evaluated. The STACKECC method stood out negatively, presenting the worst average result in all metrics evaluated. One possible explanation for these poor results is the use of all support values in the classification process performed by the meta-classifier. By using all the support values generated in the classification we end up passing to the meta-classifier values that are not correlated to the label being evaluated. These values do not contribute to the result of the method and end up working as noise in the classification process. This being true, it demonstrates the importance of the selection process of correlated labels in UDDTECC since the indiscriminate inclusion of all support values for all labels is actually detrimental to the final result.

5.2.2 Higher Diversity Datasets Results

Diversity indicates the fraction of label sets present in a dataset relative to the maximum possible number of sets. Thus, lower diversity values indicate that a dataset has more predictable label sets relative to datasets with high diversity values. When we look specifically at the experimental results for datasets with higher diversity ($Diversity > 0.1$) we see that the best UDDTECC results are concentrated exactly in datasets with this characteristic. The table 19 table shows the average results of the analyzed metrics when we focus on the 11 bases (3sources_bbc1000, 3sources_guardian1000, 3sources_inter3000, CAL500, Emotions, Enron, GpositiveGO, Image, Scene, VirusGO, Water-quality) whose Diversity > 0.1.

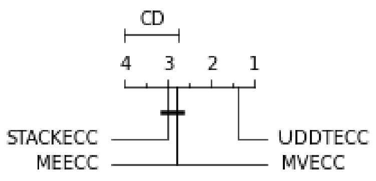
Table 19 – Average rank of the ECC methods using the evaluated fusion schemes for all datasets.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
UDDTECC	**1.363636**	**1.545455**	**1.772727**	**1.727273**
MEECC	2.818182	2.636364	2.045455	2.318182
MVECC	2.818182	2.636364	2.863636	2.772727
STACKECC	3.000000	3.181818	3.318182	3.181818

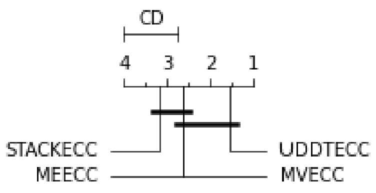
It can be seen that in the set of datasets with such a characteristic, the UDDTECC method is superior in all metrics evaluated. These results seem to show a tendency for UDDTECC to be better suited to datasets where labelsets results are less predictable.

Statistical tests were performed in order to prove such results in the two scenarios presented. Although no significant difference was identified when analyzing the total set of datasets, analyzing the datasets with higher diversity shows a slightly different scenario. The Friedman-Nemenyi statistical test was selected and applied. First the Friedman test is performed to reject the null hypothesis, and then proceed with a post-hoc analysis based on the Nemenyi method. For the Nemenyi test it was used $\alpha = 0.1$.

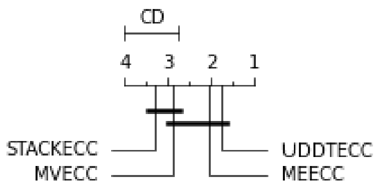
To better illustrate the test results, critical difference diagrams have been generated.



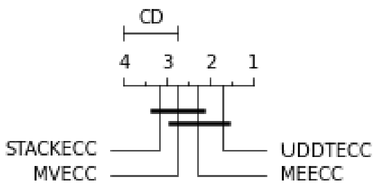
(a) Accuracy



(b) F-Measure



(c) Hamming Loss



(d) Subset Accuracy

Figure 11 – Critical difference (CD) diagrams for comparing the average of the methods evaluated in the 11 data sets with the highest diversity.

Wherever methods are connected by a horizontal line, it means they are not significantly different. It can be seen from the diagrams in fig. 11a that the UDDTECC method is significantly better on the Accuracy metric in the data sets evaluated.

The UDDTECC method is also the best on F-Measure, Hamming Loss and Subset Accuracy, respectively shown in fig. 11b, fig. 11c and fig. 11d, but no statistically significant difference was found between UDDTECC and the MEECC and MVECC methods.

5.2.3 DTECC x UDDTECC

This section compares the results of the two methods proposed in this work: DTECC and UDDTECC. Since the experiments were performed separately, the DTECC method was tested on a larger number of datasets compared to the UDDTECC method. The experiments performed with DTECC were performed before the UDTECC experiments and therefore had more time available for execution and consequently have results in a larger number of data sets. Thus, the experimental results presented so far cannot be compared directly. In order to provide a fairer comparison between the methods only results where both methods have been evaluated were considered.
Three comparison scenarios were performed: (a) the general scenario one uses all datasets where both methods were evaluated; (b) the scenario with datasets with higher number of instances where the DTECC method showed statistical difference compared to MVECC and MEECC for the Accuracy metric; and (c) the scenario with datasets with higher diversity where the UDDTECC method showed the best experimental results.

Table 20 compares the results of the two experiments performed using the 10 mutually occurring datasets: Water-quality, VirusGO, Medical, GpositiveGO, GnegativeGO, Genbase, Enron, Emotions, CAL500, 3sources_inter3000, Yeast, Image, Scene. With these results we see that the UDDTECC method outperforms the DTECC method in all the metrics evaluated. Contrasting to the result presented in section 5.2.1, the MEECC method ranked the best in Subset Accuracy while previously was the second behind the STACKECC method. The inclusion of the DTECC method and the use of a subset of datasets used there modified the results enough to change the ranking of the methods in this metric.

Table 20 – Average rank of the ECC methods using the evaluated fusion schemes for all datasets.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
DTECC	3.230769	2.923077	2.923077	3.423077
MEECC	3.230769	3.076923	**2.500000**	**2.500000**
MVECC	2.730769	2.538462	2.961538	2.807692
STACKECC	3.846154	4.076923	3.961538	3.692308
UDDTECC	**1.961538**	**2.384615**	2.653846	2.576923

5.2.3.1 Larger Datasets

Table 21 – Average rank of the ECC methods using the evaluated fusion schemes for the larger datasets.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
DTECC	2.500000	2.250000	2.250000	3.083333
MEECC	3.333333	3.166667	2.916667	2.583333
MVECC	3.166667	2.916667	3.750000	2.916667
STACKECC	4.500000	5.000000	4.000000	4.000000
UDDTECC	**1.500000**	**1.666667**	**2.083333**	**2.416667**

The average ranking results of the methods on data sets with more than 1000 instances are presented in table 21. Despite the good results presented by the DTECC method in these databases, the UDDTECC method presented the best result in all metrics evaluated. The results presented in section 5.1 show that the performance data sets with a larger number of instances are the main strength of the DTECC method but the UDDTECC method achieved results that exceeded them, showing that the use of the method that takes into account the dependencies between labels is preferable. The inclusion of the DTECC method and the use of a subset of the datasets modified the results enough to change the ranking of the methods in this metric.

5.2.3.2 Higher Diversity

As shown in section 5.2.2, the UDDTECC method tends to be superior to the other methods especially on datasets with more diversity. As shown in table 22, this trend holds when we include the DTECC method in the comparison. The UDDTECC method outperforms the DTECC and all other methods in all metrics evaluated for the subset of data sets with this property.

Table 22 – Average rank of the ECC methods using the evaluated fusion schemes for higher diversity datasets.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
DTECC	2.888889	2.500000	2.611111	3.166667
MEECC	3.444444	3.222222	2.833333	2.722222
MVECC	3.277778	3.055556	3.500000	3.222222
STACKECC	3.888889	4.222222	3.944444	3.555556
UDDTECC	**1.500000**	**2.000000**	**2.111111**	**2.333333**

5.2.4 Comparison with Baseline Methods

The results shown so far directly compare the fusion schemes used in the ECC classifier. In order to give a broader perspective and determine if the use of ensemble of classifiers was indeed positive on the considered data sets, results are presented including two traditional multi-label classifiers that are not ensemble based: Binary Relevance (BR) and Classifier Chain (CC).

The results shown in table 23 demonstrate the advantage of using the ensemble-based method over more traditional classification schemes where only one classifier is used. ECC-based methods performed better in all metrics of the experiments, the UDDTECC method with 3 wins and with MEECC winning in Hamming Loss.

Table 23 – Comparison of ECC-based methods with Binary Relevance and Classifier Chain.

Learner	Accuracy	F-measure	Hamming Loss	Subset Accuracy
BR	4.266667	3.933333	4.400000	4.033333
CC	3.566667	3.466667	4.400000	3.400000
UDDTECC	**2.200000**	**2.466667**	2.433333	**2.566667**
MEECC	3.600000	3.600000	**2.133333**	2.966667
MVECC	3.166667	3.200000	2.733333	3.233333
STACKECC	4.200000	4.333333	4.900000	4.800000

6 CONCLUSIONS

As the complexity of classification systems continually increases over the years, multi-label classification has became an important factor in current classification systems (LUGHOFER, 2022). Many real-world problems can be best modeled as classification problems where multiple labels can be assigned to each example. And, because these problems are more complex compared to single-label classification problems, the development of robust and efficient classifiers is required. In this context, ensemble of classifiers allow complex multi-label classification problems to be solved elegantly by combining the results of simpler classifiers.

In this context, we propose the UDDTECC method. A trainable multi-label classifier fusion method based on the single-label classifier combination scheme called Decision Templates. The main features that distinguish the UDDTECC method from other fusion methods typically used in multi label classification are its ability to adapt to the problem it is being used on and the exploitation of unconditional dependencies between problem labels. Methods commonly used in Multi-Label ensembles, such as MV and ME, consist of applying fixed and invariant rules regardless of the problem to which they are being applied.

In this study, the UDDTECC method is evaluated using a cross-validation scheme in which it is directly compared to the most commonly used fusion schemes in multi-label classifier fusion: Majority Vote and MeanEnsemble. STACKECC, another trainable fusion model is also included in the tests to see how the UDDTECC would perform against a Stacking method given the similarity between the two strategies.

Of the four metrics evaluated, the UDDTECC method was the best on average in three of them: Accuracy, F-measure and Subset-Accuracy. In Hamming Loss, the method was in second place, losing to MEECC in terms of average results. The STACKECC method, on the other hand, despite obtaining competitive results in very few datasets, obtained the worst average result in all the metrics evaluated, showing that the strategy of using trainable functions is not necessarily superior to fixed combination rules in all cases. The use of all unfiltered information by the STACKECC method may have introduced noise (information about labels unrelated to what is being evaluated) which would explain the low performance of the method. Using label information that does not correlate to what is being evaluated can be detrimental in the classification process. In the way it was proposed, the meta-classifier is responsible for filtering the useful attributes and performing the classification process, a strategy that did not prove to be good enough in the experiments.

This work made some simplifying choices for making the experiments lighter. The Naive Bayes classifier was used for not requiring hyperparameter tuning. In addition, the threshold value of 0.5 was used by the MVECC and MEECC methods in all experiments with different data sets. Therefore, future work should expand the methodology of experiments including other classifiers beyond Naive Bayes and performing hyperparameter optimization using a nested cross-validation model selection procedure.

The incorporation of the proposed DT method in other multi-label classification methods, such as Ensemble of Pruned Sets (EPS) (READ; PFAHRINGER; HOLMES, 2008), Ensemble of Subset Learners (ESL) (TENENBOIM-CHEKINA; ROKACH; SHAPIRA, 2010), and RAndom k-labELsets (RAkEL) (TSOUMAKAS; VLAHAVAS, 2007) should also be tested, since its operation is not intrinsically dependent on the ECC method.

Execution time and computational cost were limiting factors in the choice of databases used in this study. If more time and computational resources are available, several multi-label datasets can be added to the experimental results.

A version of the STACKECC method in which only label values that have a correlation with what is being evaluated are used, similar to what is done with UDDTECC, can be proposed and tested. If the result is significantly better than that presented by STACKECC, the effectiveness of the proposed technique would be demonstrated.

Overall, it is best to tune the threshold value for each data set. New experiments adjusting the threshold parameters can be done for the MVECC and MEECC methods.

The Naive Bayes classifier was used in all experiments performed in order to ensure that all methods tested had a fair comparison. Besides the method used, it would be interesting to verify the behavior of the evaluated methods when using another base classifier such as kNN and Random Forest.

The evaluation of other statistical methods of correlation between nominal variables besides the ϕ coefficient, for example χ^2 and mutual information, employed in the experiments, is also a possibility for future work.

It can also be argued that the UDDTECC method by including confidence values for all related labels ends up taking away some of the importance of the label that is being classified. Based on this a new method can be developed where the values of confidence are weighted in order to preserve the importance of the confidences of the label being classified.

References

ALPAYDIN, E.; JORDAN, M. I. Local linear perceptrons for classification. *IEEE Transactions on Neural Networks*, IEEE, v. 7, n. 3, p. 788–794, 1996. Citado na página 30.

ANTONUCCI, A. et al. An ensemble of bayesian networks for multilabel classification. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2013. p. 1220–1225. Citado 2 vezes nas páginas 49 and 60.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 30.

CHEN, Z.-M. et al. Multi-label image recognition with graph convolutional networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 5177–5186. Citado na página 23.

CHENG, W.; HÜLLERMEIER, E.; DEMBCZYNSKI, K. J. Bayes optimal multilabel classification via probabilistic classifier chains. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. Haifa, Israel: [s.n.], 2010. p. 279–286. Citado na página 51.

CHOU, K.-C. Advances in predicting subcellular localization of multi-label proteins and its implication for developing multi-target drugs. *Current medicinal chemistry*, Bentham Science Publishers, v. 26, n. 26, p. 4918–4943, 2019. Citado na página 23.

COHEN, J. et al. *Applied multiple regression/correlation analysis for the behavioral sciences*. [S.l.]: Routledge, 2013. Citado na página 44.

DU, J. et al. ML-Net: multi-label classification of biomedical texts with deep neural networks. *Journal of the American Medical Informatics Association*, v. 26, n. 11, p. 1279–1285, 06 2019. ISSN 1527-974X. Disponível em: <https://doi.org/10.1093/jamia/ocz085>. Citado na página 23.

FREUND, Y.; SCHAPIRE, R.; ABE, N. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, JAPANESE SOC ARTIFICIAL INTELL, v. 14, n. 771-780, p. 1612, 1999. Citado na página 30.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, Elsevier, v. 55, n. 1, p. 119–139, 1997. Citado na página 28.

GHARROUDI, O.; ELGHAZEL, H.; AUSSEM, A. Ensemble multi-label classification: A comparative study on threshold selection and voting methods. *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, p. 377–384, 2015. Citado na página 34.

GIACINTO, G.; ROLI, F. An approach to the automatic design of multiple classifier systems. *Pattern recognition letters*, Elsevier, v. 22, n. 1, p. 25–33, 2001. Citado na página 30.

GREENWOOD, P. E.; NIKULIN, M. S. *A guide to chi-squared testing.* [S.l.]: John Wiley & Sons, 1996. v. 280. Citado na página 45.

GUO, X. et al. Human protein subcellular localization with integrated source and multi-label ensemble classifier. *Scientific Reports*, Nature Publishing Group, v. 6, p. 28087, 2016. Citado na página 35.

HO, T. Complexity of classification problems and comparative advantages of combined classifiers. *Multiple Classifier Systems*, Springer, volume 1857 of Lecture Notes in Computer Science, p. 97–106, 2000. Citado na página 29.

JACOBS, R. A. et al. Adaptive mixtures of local experts. *Neural computation*, MIT Press, v. 3, n. 1, p. 79–87, 1991. Citado na página 30.

KUNCHEVA, L. I. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 32, n. 2, p. 146–156, 2002. Citado na página 29.

KUNCHEVA, L. I. *Combining pattern classifiers: methods and algorithms.* [S.l.]: John Wiley & Sons, 2014. Citado 2 vezes nas páginas 31 and 32.

KUNCHEVA, L. I.; BEZDEK, J. C.; DUIN, R. P. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, Elsevier, v. 34, n. 2, p. 299–314, 2001. Citado 2 vezes nas páginas 29 and 35.

LIN, B. Y. et al. Multi-channel bilstm-crf model for emerging named entity recognition in social media. In: *Proceedings of the 3rd Workshop on Noisy User-generated Text.* [S.l.: s.n.], 2017. p. 160–165. Citado na página 27.

LUGHOFER, E. Evolving multi-label fuzzy classifier. *Information Sciences*, v. 597, p. 1–23, 2022. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522002560>. Citado na página 71.

MOYANO, J. M. et al. Review of ensembles of multi-label classifiers: models, experimental study and prospects. *Information Fusion*, Elsevier, v. 44, p. 33–45, 2018. Citado 2 vezes nas páginas 24 and 31.

OPITZ, D.; MACLIN, R. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, v. 11, p. 169–198, 1999. Citado na página 29.

PEREIRA, R. B. et al. Correlation analysis of performance measures for multi-label classification. *Information Processing and Management*, Elsevier, v. 54, n. 3, p. 359–369, 2018. Citado na página 58.

PETKOVIĆ, M.; DŽEROSKI, S.; KOCEV, D. Multi-label feature ranking with ensemble methods. *Machine Learning*, Springer, v. 109, p. 2141–2159, 2020. Citado na página 49.

RAUBER, T. W. et al. An experimental methodology to evaluate machine learning methods for fault diagnosis based on vibration signals. *Expert Systems with Applications*, Elsevier, p. 114022, 2020. Citado na página 49.

RAUBER, T. W. et al. Decision template multi-label classification based on recursive dependent binary relevance. In: IEEE. *Evolutionary Computation (CEC), 2016 IEEE Congress on.* [S.l.], 2016. p. 2402–2408. Citado na página 24.

READ, J.; PFAHRINGER, B.; HOLMES, G. Multi-label classification using ensembles of pruned sets. In: IEEE. *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. [S.l.], 2008. p. 995–1000. Citado 3 vezes nas páginas 28, 30, and 72.

READ, J. et al. Classifier chains for multi-label classification. *Machine learning*, Springer, v. 85, n. 3, p. 333–359, 2011. Citado 3 vezes nas páginas 23, 30, and 31.

ROCHA, V. F.; VAREJÃO, F. M.; SEGATTO, M. E. V. Ensemble of classifier chains and decision templates for multi-label classification. *Knowledge and Information Systems*, Springer, p. 1–21, 2022. Citado 2 vezes nas páginas 24 and 41.

ROGOVA, G. Combining the results of several neural network classifiers. *Neural networks*, Elsevier, v. 7, n. 5, p. 777–781, 1994. Citado na página 35.

ROKACH, L. Ensemble-based classifiers. *Artificial Intelligence Review*, Springer, v. 33, n. 1, p. 1–39, 2010. Citado na página 29.

SCHAPIRE, R. E. The strength of weak learnability. *Machine learning*, Springer, v. 5, n. 2, p. 197–227, 1990. Citado na página 30.

SCHAPIRE, R. E.; SINGER, Y. Boostexter: A boosting-based system for text categorization. *Machine learning*, Springer, v. 39, n. 2, p. 135–168, 2000. Citado 2 vezes nas páginas 28 and 50.

TANAKA, E. A.; BARANAUSKAS, J. A. An adaptation of binary relevance for multi-label classification applied to functional genomics. In: *Proceedings of the XXXII Congress of the Brazilian Computer Society, XII Workshop on Medical Informatics*. [S.l.: s.n.], 2012. Citado na página 28.

TANG, J.; ALELYANI, S.; LIU, H. Data classification: algorithms and applications. *Data Mining and Knowledge Discovery Series*, CRC press, p. 37–64, 2014. Citado na página 39.

TENENBOIM-CHEKINA, L.; ROKACH, L.; SHAPIRA, B. Identification of label dependencies for multi-label classification. In: *Working Notes of the Second International Workshop on Learning from Multi-Label Data*. [S.l.: s.n.], 2010. p. 53–60. Citado 3 vezes nas páginas 28, 30, and 72.

TSOUMAKAS, G. et al. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2411–2414, 2011. Citado na página 49.

TSOUMAKAS, G.; VLAHAVAS, I. Random k-labelsets: An ensemble method for multilabel classification. *Machine learning: ECML 2007*, Springer, p. 406–417, 2007. Citado 3 vezes nas páginas 28, 30, and 72.

WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 19, n. 4, p. 405–410, 1997. Citado 2 vezes nas páginas 29 and 30.

ZHANG, M.-L.; ZHOU, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, IEEE, v. 18, n. 10, p. 1338–1351, 2006. Citado na página 28.

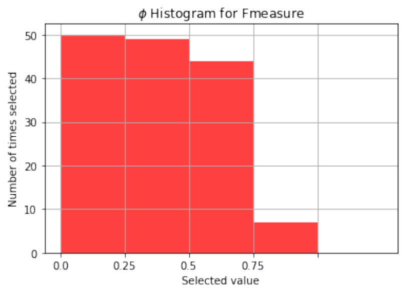
ZHANG, M.-L.; ZHOU, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, Elsevier, v. 40, n. 7, p. 2038–2048, 2007. Citado na página 28.

ZHANG, M.-L.; ZHOU, Z.-H. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, IEEE, v. 26, n. 8, p. 1819–1837, 2013. Citado na página 50.

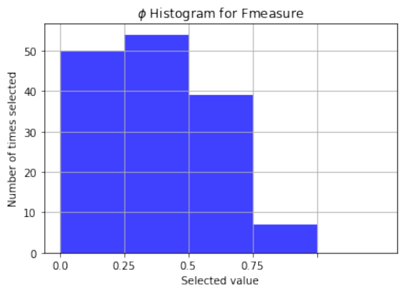
Appendix

Selected ϕ_c histograms

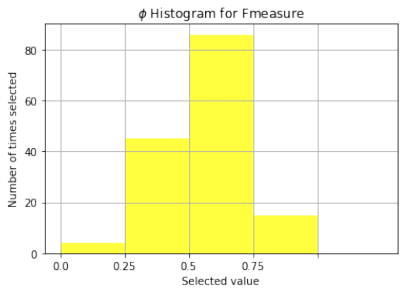
May 9, 2022



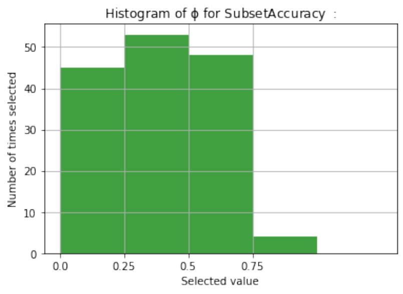
(a) Selected ϕ_c values for the Accuracy metric



(b) Selected ϕ_c values for the F-measure metric.



(c) Selected ϕ_c values for the Hamming Loss metric.



(d) Selected ϕ_c values for the Subset Accuracy metric.

Figure 1: Selected ϕ_c by the DTECCd method in the experiments by metric.

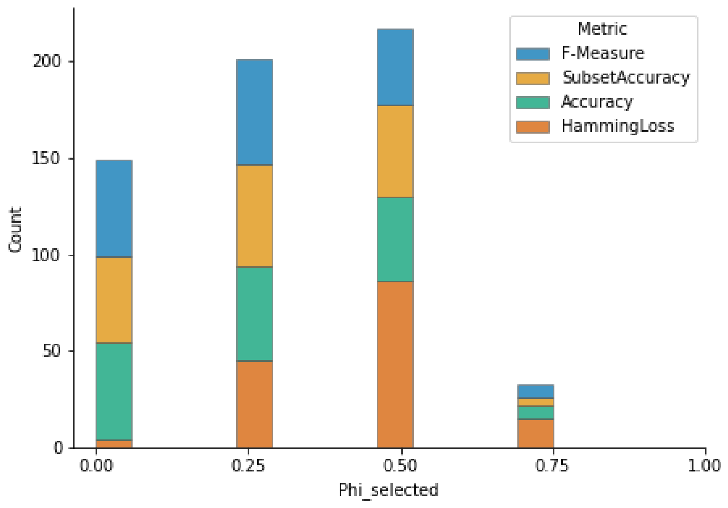


Figure 2: Selected ϕ_c values combined.