Multi-Label Classification Using Binary Tree of Classifiers

Anwesha Law, *Student Member, IEEE*, and Ashish Ghosh, *Senior Member, IEEE*

***Abstract*—This article proposes a binary tree of classifiers for multi-label classification that preserves label dependencies and handles class imbalance. At each node, the input data is strategically split into two subsets for its subsequent child nodes, keeping the label correlations intact. A novel approach of partitioning the data based on label-set proximity has also been proposed. Various data appropriate classifiers are trained to learn the binary partition at every internal node. The tree of classifiers grows iteratively depending on two parameters – multi-label entropy and sample cardinality, computed on the data at the current node. During training, the decision at any node is based on these parameters and the branching out is restricted, if deemed unnecessary. Specific classifiers at the leaf nodes perform the final classification task and assign appropriate label-sets to the unlabelled data. The proposed system aims to appropriately split the data and build the hierarchical structure such that the training and classification tasks become simpler. Also, the problem of class imbalance leads to the irregular splitting of data and excessive branching out of the tree which is handled through the novel use of suitable classifiers and parameters at the intermediate and leaf nodes. The proposed method has shown significant performance improvement on fourteen datasets against fourteen existing multi-label classifiers. Two-tailed Wilcoxon signed rank test statistics show that for $T_{Wilcoxon}(14, 0.2) = 31$ the proposed method outperforms all the other comparison models.**

***Index Terms*—Multi-label classification, classifier tree, class imbalance, class correlation.**

I. Introduction

MULTI-LABEL classification (MLC) [1] is being worked on from the last two decades, and it is mostly used in the fields of text categorization, gene-based classification, learning from images, video, audio, etc. Unlike traditional machine learning techniques which vastly deal with data that belong to any one class (also known as single-label/multi-class data), multi-label learning aims to handle data which, as the name suggests, has multiple labels. However, unlike single-label data, simultaneous involvement of numerous classes makes multi-label data somewhat ambiguous. This ambiguity arises due to its association with multiple classes at the same time, but unlike fuzzy logic, data membership to all the classes for ML data is absolutely crisp. MLC is also quite different from multi-object recognition, since it is more focussed on identifying the underlying concepts in the data, instead of specific objects. Thus, it can be said, that ML data is quite complex and hence needs to be handled efficiently to deal with the inherent ambiguity in the data. Various types of approaches are used to handle multi-label data, namely, data transformation [2], [3], problem adaptation [4], [5], ensemble techniques [6], etc. Despite the various approaches to multi-label classification, the inherent ambiguous nature of the data results into several complexities that arise while dealing with this kind of data. Concerns like complex decision space and correlated classes need to be handled specifically. Researchers approach the issues from different perspectives to handle the shortcomings.

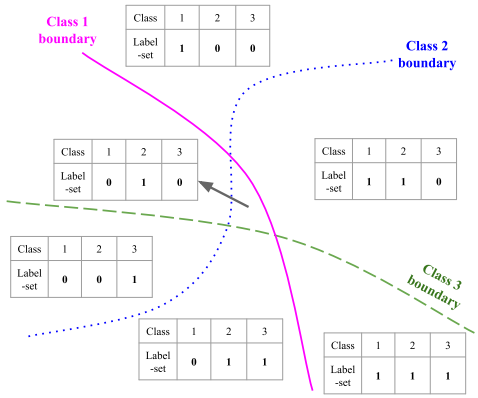

Fig. 1. An example of multi-label class boundaries.

When dealing with multi-label data, one of the most important issues to keep in mind is its complex decision space [1]. The classifiers need to learn decision boundaries to separate the various classes that are more complex than the ones we come across while performing single-label classification. Ideally, it can be said that there is one decision boundary for each of the classes, where it partitions the data depending on the classification outcome for that particular class. Therefore, on the whole, there are multiple overlapping class boundaries that help segregate different classes of multi-label data. For example, in Fig. 1 a toy scenario has been shown with 3 classes. Each class has a separate class boundary and the data which lie within the boundaries have the specific label-sets. Among various ML classifiers that exist, a common approach of learning these boundaries is by training the classifier one decision boundary at a time [2], i.e., it works like

Manuscript received October 2, 2020; revised January 9, 2021 and March 26, 2021; accepted April 8, 2021. This work was supported by Indian Statistical Institute, Kolkata and Technology Innovation Hub on Data Science, Big Data Analytics and Data Curation under Grant NMICPS/006/MD/2020-21 dt 16.10.2020. Prof. Xuelong Li *(Corresponding author: A. Ghosh.)*

The authors are with Machine Intelligence Unit, Indian Statistical Institute, Kolkata, West Bengal 700108, India (e-mail: anweshalaw_r@isical.ac.in; ash@isical.ac.in).

Digital Object Identifier 10.1109/TETCI.2021.3075717

a one-vs-all classifier [1] which handles each class individually. This essentially ignores the inter-relations between the data of various classes. Each single decision boundary is made for a single class at a time, disregarding the fact that it might co-occur and be closely related to any of the other classes. Previously, most of the multi-label classification techniques either used one complex classifier to classify the multi-label data [7], or they used an ensemble of classifiers [6]. In most cases, the stand-alone classifier needs to learn all the boundaries by itself, thus making the single classifier quite complex. On the other hand, when the model constitutes one-vs-all classifiers that learn individual decision boundaries for each class, it considers the classes to be independent and ends up ignoring any inter-relations between them. In the past few years, multi-label classifiers [8] are being developed which attempt to preserve the underlying correlations among labels along with other issues in the field of multi-label classification. This motivated us to build a ML classification model that will utilize the existing class correlations in the data to improve the classification performance.

In this regard, a tree-based hierarchical multi-label classifier has been proposed in this article, that aims to attack the problem from a different perspective. Instead of assuming the classes to be independent and learning one label at a time, it aims to utilize and preserve the label associations as much as possible. This will be done by localizing the data according to the class labels and then using multiple small classifiers for training instead of employing a massive complex classifier. The basic hierarchical model of the proposed classifier is inspired by decision trees, which contains the entire training data at the root node. In the proposed Multi-Label Binary Tree of Classifiers (ML-BTC), this data is split into two discrete chunks every time as the levels in the hierarchy increase, eventually to form a binary tree-like structure. Unlike other models in literature [8], [9], here, specifically a binary tree structure has been adapted instead of an n-ary tree with a variable number of children. It essentially means that at each intermediate node, only one split is necessary; it focuses on the major partition in the data. Subsequent child nodes look for further divisions in the partitioned sub-space. Moreover, contrary to the decision tree approach of exhaustively searching for the "best" split at each node, the proposed model seeks an approximate split which strives to keep label dependencies unchanged. Thus, the proposed model takes the data at a node and approximately splits it into two sets aiming to keep the correlation among classes intact. This consistent correlation is achieved with a novel approach to compute label-set proximity among the data. It forms subsets that contain data instances whose label-sets are similar to each other. This helps in the preservation of label correlations without computing class-wise correlations explicitly.

The motivation to opt for a decision tree based approach is to be able to manipulate, combine or split the ML datasets conveniently to help deal with the class imbalance issue [10], [11]. At the intermediate nodes of the tree, similar label-sets are grouped together to form two comparatively larger subsets which have increased sample size as compared to the individual label-sets. This makes the class imbalance issue less prominent. However, sometimes when the data is highly imbalanced, the class-based splitting at these nodes might lead to uneven subsets. This issue is also handled in the proposed technique by utilizing different types of classifiers and parameters for appropriate scenarios. Preliminary classification is performed at the intermediate nodes while the final classification is done at the leaf nodes. Unlike other tree-based models, the proposed tree is not built continuously until each node consists of an individual class or label-set. Since in the worst case, imbalanced classes might end up with a singleton training sample in a leaf node. To avoid unnecessary partitioning of imbalanced classes, the expansion of the tree is restricted with the help of parameters at a much earlier stage. Thus, suitable parameters and classifiers have been used to orchestrate a strategic yet efficient formation of the hierarchical tree of classifiers, sometimes also referred as "classifier tree" in this article. This name is apt, since it is an adaptation of the traditional decision tree approach with the "decision" at most of the nodes being made by some classifier. The overall goal of this work is to build a decision tree based model that utilizes the underlying label correlations in the data to perform efficient ML classification while handling the class imbalance issue. The main contributions of this article can be highlighted as follows.

- Build a tree of classifiers for multi-label classification that utilizes suitable classifiers at the intermediate and leaf nodes to handle various bottlenecks.
- While building the tree, a novel label-space partitioning technique is applied on the data to implicitly handle the underlying class correlations which otherwise might get ignored.
- Approximate splitting of data is done to achieve faster convergence instead of following the traditional exhaustive best split approach. ML data often have large input and output dimensions which is overwhelming for an exhaustive search. Also, the broad partition allocates more data with a group than the imbalanced classes and label-sets, thus making the problem class imbalance less prominent.
- Explicitly handling imbalanced classes that cause uneven splitting of data at the intermediate nodes. The use of appropriate classifiers at the intermediate and leaf nodes based on the data at hand ensures proper attention to the imbalanced data. Also, building of the tree is based on parameters which facilitate restrictions to prevent its unnecessary branching for smaller imbalanced classes.

The proposed model has been tested on fourteen standard ML datasets, with nine performance metrics, compared with fourteen relevant state-of-the-art techniques and it shows significant improvement with respect to most of the existing methods.

The organisation of the remainder of the article is as follows. Section II discusses several existing works in the field of multi-label classification with a focus on tree-based methods. Section III contains a detailed description of the proposed work along with the architecture and methodology. Experimental analysis has been provided in Section IV, and finally, Section V concludes the article.

II. Related Work

Among the multi-label approaches mentioned earlier, data transformation was one of the early approaches, where instead of

developing new algorithms, the multi-label data was converted to single-label data and existing classifiers were used to predict the class labels. Binary relevance (BR) [3] is a popular method in the domain of ML classification. There are many variations of the basic algorithm that transforms the multi-label data to a binary classification task where each label is learnt by a different classifier and the final predictions are combined as an ensemble. Another set of approaches, known as algorithm adaptation, mainly perform modifications based on traditional methods like decision trees [4], [12], neural networks [5], [13], support vector machines [14], [15], etc. Various single-label classification techniques based on popular models like convolutional neural networks (CNN) [16]–[18] etc. have been adapted for ML classification. Among these adaptations, multi-label k-nearest neighbour (ML-kNN) [7] is a well-known algorithm. It is an adaptation of the traditional k-NN algorithm which further uses maximum a posteriori (MAP) probabilities to compute the final label-set for the unknown sample. Simplified constraints RankSVM (SC-RankSVM) [15] is a modification of the traditional SVM with milder constraints to minimize a multi-label metric (namely, ranking loss) and have a large margin. In [19], ML classification using Laplacian Eigen Map (MLLEM) has been proposed as a non-linear embedding technique which incorporates instance-instance, label-instance and label-label relationship into the same space. This method attempts to provide a simultaneous visualization for the patterns as well as the labels. Multi-Label Tomek Link (MLTL) [20] is a recent work using Tomek link approach that handles class imbalance of ML data. This approach has been used with existing algorithms to improve their performance. Along with the evolution of stand-alone classifiers, the concept of ensemble of classifiers have also proven to be quite efficient in the field of ML classification. Classifier Chains (CC) [21] is one of the best-known ensembles of binary classifiers for multi-label classification where each chain gives a label-set as output. The ensemble of classifier chains (ECC) thus can be thought of as an ensemble of ML classifiers. The label specific features based classifier chain for multi-label classification (LSF-CC) [22] is a recent modification of CC to handle some drawbacks like random label ordering and noises in the original and additional features. Random k-labelsets (RAKEL) [23] is another ensemble technique which employs multiple single-label classifiers on subset of classes considering the existing label correlations. In [24], a stacked ensemble model has been developed that exploits label correlations as well as learns the weights of the members in the ensemble. Apart from ensembles, in [25], authors performed multi-label learning by exploiting the label correlations locally (ML-LOC), where a local correlation (LOC) code is determined and used as additional features. ML learning with label-specific features (LIFT) [26] conducts clustering analysis on positive and negative instances and constructs features specific to the individual classes. It then queries the clustering results to perform training and testing of the classifier. The authors in [27] propose a multi-label naïve Bayes classifier that considers correlations between pairs of classes.

Among all the existing types of classifiers, the focus of the proposed work is directed towards tree-based classifiers for general ML data. Tree based techniques have been widely used for single-label classification [28], but have not been extensively explored in the ML scenario. Decision based building of the tree structure is quite simple and robust, which makes it an apt candidate for ML data. Multi-label C4.5 [4] is one of the first adaptations of decision trees for multi-label classification. Here, the splitting of the tree is based on a modified entropy measure. In general, it is seen that decision tree-based techniques mostly vary on the way the data is split in each intermediate node. Hierarchy of multi-label classifiers (HOMER) [12] splits the classes into several groups such that at the end each leaf node corresponds to a particular class. Multi-label classifiers are used at intermediate stages to learn the subset of classes. Recently, [20] has developed MLTL-HOMER while handling class imbalance. ML-TREE [9] is another decision tree-based algorithm that splits the data into multiple branches using one-vs-all SVM for each class at intermediate nodes. ML-FOREST [8] is an ensemble of ensembles where multiple ML-TREEs are combined. In [29], a tree-shaped ensemble of ML classifiers has been built which is based on grammar-guided genetic programming (G3P-kEMLC). There are also some tree-based models in literature specifically for extreme ML data and hierarchical ML data, but it has not been included in our study.

III. Binary Tree of Classifiers for Multi-Label Classification

In this article, we develop a novel binary tree of classifiers for multi-label classification (ML-BTC). This method attempts to overcome a few limitations that are faced while handling multi-label data. In the literature, researchers are seen to approach the problem of multi-label classification from various perspectives. Among the existing obstacles, utilizing and preserving the dependencies among classes are quite important. Correlated classes that frequently occur together can be mapped simultaneously to validate their occurrences and reduce misclassification. Many of the existing methods may not explicitly handle this problem. In the proposed method, we have tried to utilize the class dependencies that exist within the data by grouping the data that belong to a similar set of classes. This helps the classifier to identify the related classes and guide an unknown data sample through the ML-BTC to its closest label-set. Along with this, the problem of class imbalance that commonly occurs in multi-label data has also been handled in the proposed method. Grouping of related label-sets at the intermediate steps help to increase the sample size in comparison to that of individual label-sets, thus diminishing the magnitude of the imbalance problem. However, the presence of highly imbalanced classes can still cause uneven partitioning of data and unnecessary branching of the tree which have been handled effectively through appropriate use of certain classifiers and restricting growth of the tree with the help of some parameters. Details of the proposed approach have been discussed in the following sections.

A. Building of the Tree

Starting from the root node, the tree is formed by successively splitting the available data at every internal node. Initially, the root node contains all the training instances N_{tr}, and the initial split at the root node separates the data into two subsets, say, N_{tr1} and N_{tr2}, where $N_{tr} = N_{tr1} \bigcup N_{tr2}$ and $N_{tr1} \bigcap N_{tr2} = \phi$.

This splitting of the data leads to the creation of two child nodes which contain one subset of N_{tr} each. The data present at any node is used as the training set of that particular node. In this way, the data at each internal node is iteratively split into smaller partitions and the tree eventually grows until the data cannot be split any further. The division at each internal node represents a prominent split in that data, keeping the correlation between labels intact. A novel proximity-based label-space partitioning scheme has been employed to preserve the label correlations. Once an approximate split has been determined at the current node, a classifier is trained to learn the decision boundary between the two subsets of data. In the subsequent steps, data chunks at the nodes are further split and specific classifiers are trained to learn the broad classification. The choice of the classifier is based on the type of partition that is made at an internal node. Researchers who developed similar tree-based multi-label classifiers [8], [9] in the past, have preferred to use a single type of classifier for the entire data. However, while dealing with ML data, the class imbalance problem leads to irregular partitioning in the intermediate steps. This, in turn, makes larger classifiers quite impractical to be trained on a smaller quantity of data. Keeping this in mind, we chose to incorporate different classifiers to suit the data at hand and improve effectiveness.

Every decision at any internal node is made depending on two parameters: ML entropy and sample cardinality – that determine the status of the current data chunk. If the parameters suffice, the data at a node is split further, otherwise, the node is converted to a leaf node. It is at these leaf nodes that the final multi-label class prediction is performed. The data that is at a leaf node might belong to a single set of labels or a varied set of labels. If there are different combinations of labels among the data in a leaf node, a suitable multi-label classifier is used in that node. If there is an unique label-set in a leaf node, no further classifier is needed. The intuition behind developing this model is that multi-label data is inherently imbalanced. It becomes expensive and increases the number of nodes in a decision tree if the tree grows until each leaf node contains an unique label-set. In this case, instances whose label-sets are quite similar to each other may be found residing in a leaf node if the data size is not large enough to be partitioned. Hence, a multi-label classifier, that can be trained on smaller data, at that leaf node performs efficient classification instead of the repetitive splitting of the data. The actions associated with the building of the classifier tree are discussed below in detail.

1) Splitting Strategy: Most decision tree-based algorithms split the data depending on a single feature at a time [4] or the entire feature space [9], [12]. Although this seems efficient for single-label classification, for multi-label classification with the increase in the input and output dimensions, the problem becomes more complex to handle. In the proposed method, instead of using the feature space for partitioning the data, we have utilized the label information in a novel way. For example, while dealing with multi-label images, a broad classification would mean images with one or more labels like "sea," "sand," "beach," etc. would belong together whereas images with labels like "grass," "mountains," etc. should be a part of another large group. To enable this type of broad classification, we propose to

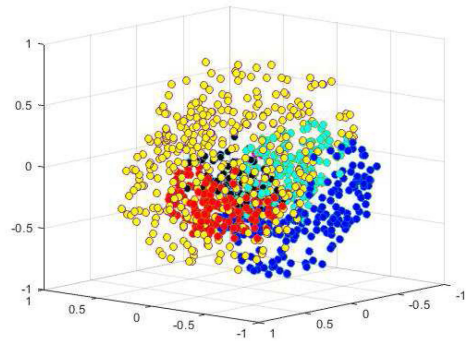

Fig. 2. Representation of data in the feature space.

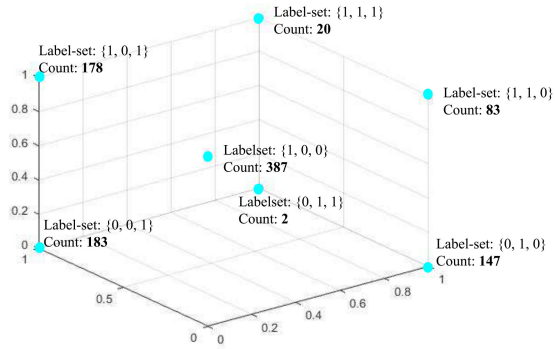

Fig. 3. Representation of data in the label space.

group the data in the label space instead of the feature space. This means that data whose class labels are similar will belong to one group and the classifier will train itself based on this partition.

Fig. 2 shows an example of a synthetic multi-label dataset with three features and three classes. Each of the label-sets is depicted in different colours to visualize the complexity of the data. For real multi-label data, with higher input and output dimensions, the data is far more complex. To deal with this data more simply, the proposed method visualizes the data in the label space and obtain a split which preserves label correlations approximately.

Fig. 3 represents the same dataset in the 3-dimensional output space, which appears to be far simpler and organized. Thus, the data to be split at any node is partitioned into two groups in the output dimension. Data with the same label-sets are represented with the same point in the label space; hence, data having the same label-set always belong to the same group. This reduces the scope of misrepresentation of data. This approach ensures that data of similar class labels will be closer in the decision space and thus make the task of partitioning simpler.

Fig. 4 depicts the splitting strategy developed here, with some three-class label-sets spread out in the label space (similar to Fig. 1). The entire data (represented by the label-sets) available at a node is split into two subsets based on the proximity of their label vectors. This proximity among any two data points is measured by computing the Hamming distance among their respective label-sets. Hamming distance is one of the easiest techniques to compute the difference between two label-sets which are same-length vectors of 1 s and 0 s. Difference of bit values indicate dissimilarity. Lower Hamming distance between

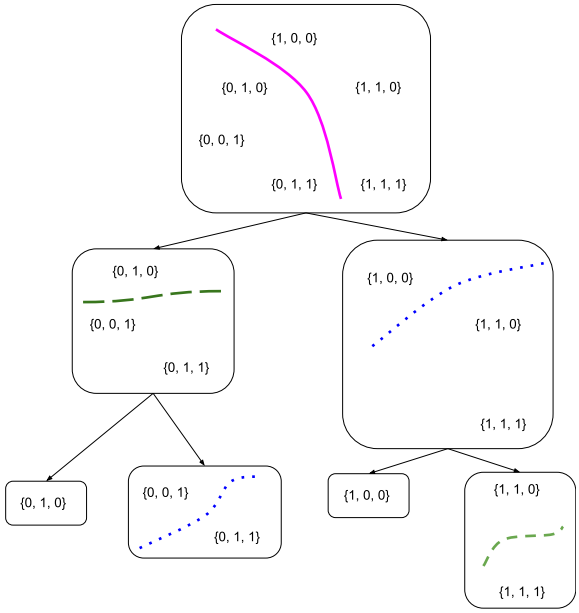

Fig. 4. Splitting strategy of ML-BTC.

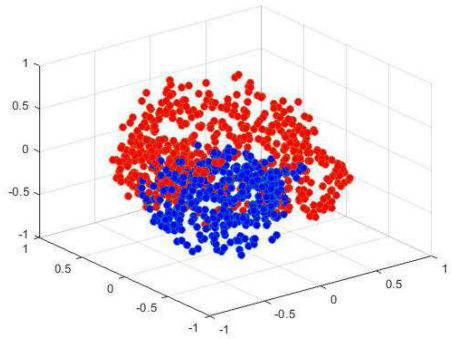

Fig. 5. Approximate partition of the data as per the output space.

two points indicates greater similarity among the label-sets. In the proposed ML-BTC, to find co-occuring classes, the label-sets with low Hamming distance are considered. Depending on these distances, the entire data is consecutively split into two groups at each intermediate node. The consecutive splitting of data at every node that leads to the formation of a tree-like structure calls for a divisive hierarchical clustering algorithm that can partition the available data into two chunks at every step. For this purpose, a hierarchical clustering algorithm, namely, bisecting k-medoids, is employed step-wise on the proposed tree structure.

Fig. 5 represents the synthetic dataset after application of bisecting k-medoids with the approximate split generated in the output space of the data mapped to the input space. After the splitting is done, data with similar label-sets are seen to belong to the same subset. The intuition behind this approach is that, while dealing with multi-label data we may come across instances whose features are similar but the label-sets are very different. Similarly, instances having similar label-sets may not have identical feature values. Hence, basing the decision at a node on feature values, may not be quite appropriate while dealing with multi-label data. Using the label information in this way we attempt to preserve and utilize the underlying correlation between classes in a particular dataset. If a set of labels were inter-related, i.e., they occur together frequently, they remain intact in the same subset. On the other hand, using a method to split the data based on each label or feature loses any label correlation information that was present in the data. The proposed method aims to utilize this label correlation information and assign the closely related labels to the relevant sample in turn improving the final classification performance. Additionally, this approximate splitting strategy helps to handle the class imbalance issue to some extent. When two broad groups are created, it contains larger number of data points than the individual imbalanced classes or label-sets. This helps to make the imbalance problem less prominent at this stage. However, if still the data splitting is uneven due to imbalanced classes, it is handled separately.

In this novel splitting strategy, we do not explicitly compute the correlation among labels, but it serves a similar purpose. Identical or similar label-sets end up in the same group, whereas dissimilar label-sets will have a higher Hamming distance, thus, they are most likely to be well separated. In case of equidistant label-sets, depending on the parameters, initially, one approximate split is made in the current node. Later, in the children nodes, further splits are made if necessary, eventually leading to few equidistant label-sets in the same group or as individual leaf nodes. However, once a split has been determined, it is the job of the binary classifier to train itself to be able to classify future samples in the same way.

2) Classifiers: In the proposed model, different classifiers have been used to handle specific scenarios. Unlike existing tree-based multi-label classification models, the proposed hierarchical ML-BTC uses different classifiers depending on the data at a node that it needs to handle. The intuition behind using different classifiers is that each splitting of data at an intermediate node may not result in similar partitions. Existing techniques [8], [9] use a single classifier in the entire model. Since multi-label data inherently is class imbalanced, few label-sets have insufficient data and splitting up of this set may not result in equal-sized partitions. If the split consists of a tiny and a large subset, using a generic classifier like an SVM might not prove to be quite useful. The uneven partition might cause the selection of insufficient support vectors thus leading to biased results. In this scenario, we opt to use a simpler classifier which can work with irregular as well as lesser quantity of data. A classifier like k-nearest neighbour (k-NN) generates a piece-wise decision boundary based on a small number of data points. Thus, the uneven sizes of the partitions do not affect the final classification boundary [30]. Thus, if the data at hand is split into two uneven partitions, a k-NN classifier suffices. On the other hand, if the split results into two large-sized groups, an SVM classifier is used. SVM has been chosen since is a strong classifier which is able to efficiently perform classification for ML data [8], [9].

Apart from the intermediate nodes, the proposed model attempts to employ classifiers in some leaf nodes when needed. Due to the class imbalance problem, the subsequent splitting of data might be quite irregular thus resulting in a node containing a smaller number of data, but from various label-sets. At such a node, where the data cannot be split any further, yet its entropy

is on the higher side, a multi-label classifier is employed at that node. This ensures that the classifier tree need not be expanded further, however, the data at that leaf node gets correctly classified. Since this job is of a simpler classifier that can work well with a smaller number of data, we use a multi-label k-NN classifier. In a similar scenario with a larger amount of data that cannot be split further, a shallow multi-layer perceptron (MLP) is used. In this way, to determine which classifier is suitable at the current node, few parameters have been used in the proposed model.

3) Parameters: The proposed binary tree of classifiers model for multi-label classification proceeds to split the available data at an internal node into two subsets. The data available at each node may not be sufficient or suitable for further splitting. The decision to be taken at any node is thus quite crucial and is based on the following two parameters.

1) **Multi-label entropy**: Entropy determines the degree of variation in the label-sets of the data available at a node. If entropy $= 0$, the chunk of data in the node belongs to a single label-set, i.e., all the instances at that node have the same set of labels. With the increase in entropy, it can be said that the data available at a node has increased variation. In other words, there are multiple instances whose label-sets are quite different. Entropy is defined as

$$H = -\sum_{i}(p_i \log p_i) \tag{1}$$

where p_i is the probability of an instance belonging to the i^{th} class. Multi-label entropy can be described [4] as

$$H = -\sum_{i}(p_i \log p_i + q_i \log q_i) \tag{2}$$

where $q_i = 1 - p_i$. At every child node, the entropy should be lower than its parent node. An entropy threshold, $H_{threshold}$, is provided in the model, which is used to determine the decision to be taken at any node. At the point where entropy cannot be reduced further, the tree stops growing.

2) **Sample Cardinality**: This parameter, named C, is used along with entropy to make decisions at any node. It represents the number of data points present at the current node. This is essential since decisions cannot be made based only on the variation in the data, the size of the available data also needs to be assessed. A cardinality threshold, $C_{threshold}$, determines if the amount of data at any node is sufficient to be partitioned further or the process should be stopped to form a leaf node. The importance of this parameter lies in the simple notion that we do not aim to unnecessarily keep on expanding the tree even if the data size becomes trivial.

A combination of the above two parameters helps to make a decision at any given node in the training phase. Splitting of the data at every node follows a simple strategy; if there is sufficient data at the node and the variation is high, the data needs to be split and the tree can be expanded. Sometimes, even if the entropy is high, the amount of data at a node may not be enough for splitting, i.e., since the sample cardinality is already low, splitting it into further small groups serves no purpose and simply increases the number of nodes for traversal in the testing phase. The aim of building this tree is not to have unique label-sets at each leaf node, but to keep building the tree as long as it is necessary to perform efficient multi-label classification. If a situation arises where every label-set has one training sample each, it is pointless to keep on splitting to end up with leaf nodes having singleton label-sets. Instead, the tree would first recognize prominent splits and employ a smaller classifier at a much earlier stage to do the same job and reduce unnecessary branching out. Thus, the actions within the tree, at any internal node have high importance.

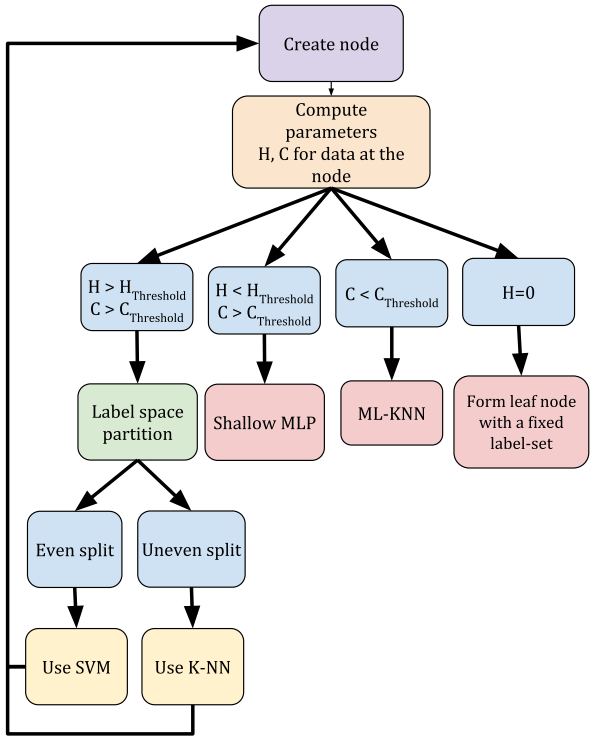

Fig. 6. Flow diagram for the creation of ML-BTC.

B. Methodology

The main purpose of this contribution is to construct a tree-like structure step-wise capable of performing multi-label classification. When the test data is fed to the root node of the trained tree, based on some decisions at the intermediate nodes, the data traverses smoothly from the root node to a leaf node which eventually ends up classifying the data sample to a specific label-set.

1) Training Phase: In the training phase, the hierarchical binary tree of classifiers is created in a step-wise fashion (Fig. 6). Initially, the empty root node contains the entire training dataset. The action-determining parameters (namely entropy and cardinality) are computed at the current node ($node_C$). Depending on these parameters, H and C, the next action is selected. The parameters (H & C) are evaluated based on threshold values,

$H_{threshold}$ and $C_{threshold}$. There are a few conditions that may arise depending on the parameters:

- $H > H_{threshold}$ and $C > C_{threshold}$: This indicates the presence of a large number of diverse data which should be split in order to expand the tree. Thus, a label space partition is performed using bisecting k-medoids at $node_C$ thereby creating a strategic split in the data.
  - If this splitting of data is uneven leading to one of the groups being too small, a k-NN classifier is used for this partition.
  - If the data has been split into even proportions, an SVM is used to learn a decision boundary between the two parts.
- $C < C_{threshold}$: If a node has a lesser amount of data and splitting it is not possible, the node is converted to a leaf node and an MLKNN classifier is used. This ensures that any data that land up in this node while testing gets properly classified. Even though the amount of training data might be small, the label-sets are not completely ignored. This ensures the handling of imbalanced classes that have a fewer number of samples.
- $H < H_{threshold}$ and $C > C_{threshold}$: If there is enough data with lower variation, it indicates a group of closely related label-sets. Splitting it further results in the separation of correlated label-sets. The low entropy is further reduced as a result of splitting, eventually leading to individual leaf nodes for each label-set. This elongated procedure is stopped early by employing a shallow MLP on the sufficient data available, to learn different classes at that stage.
- $H = 0$, this indicates that the dataset present at the current node belongs to the same label-set, hence no further action is required. This node also becomes a leaf node with a fixed label-set.

The depth of the tree is not fixed beforehand. It grows till there is sufficient data to be split at any intermediate node. If the conditions determine the data unfit to be split any further, that node is converted to a leaf node. In this way, the entire tree grows to some extent depending on the two parameters.

2) Testing Phase: After the hierarchical tree structure has been built in the training phase, the validation/testing phase begins. There are a few varied scenarios that may arise for the test data (Fig. 7). The testing process begins with an unknown sample at the root node. The binary classifier at the root classifies the test pattern to either the left child or the right child of the current node. In this way, the data moves downwards through the binary tree structure getting classified by the designated classifier at every intermediate node. Finally, at some point, the test pattern reaches a leaf node which performs the final multi-label classification. There are 3 scenarios that a test pattern may face at a leaf node:

- Fixed label-set: The leaf node has a fixed label-set and the test pattern is assigned that particular set of labels without any further investigation.
- MLP: The test pattern is fed to the trained network to output its final label-set.
- ML-KNN: The ML-KNN classifier finalizes the label-set to be assigned to the test pattern that reaches this leaf node.

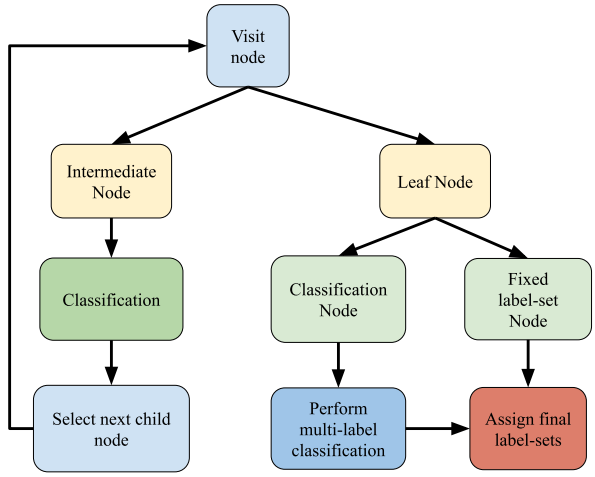

Fig. 7. Flow diagram for the testing phase of ML-BTC.

In this way, any unknown sample gets gradually classified as it traverses through the ML-BTC and receives its final label-set at the leaf nodes. In case of a test data that belongs to a label-set unknown to the classifier tree, it moves step-wise through the intermediate nodes to reach a leaf node that assigns a label-set nearest to its actual, thus minimizing the classification error. However, in the case of a stark domain shift, i.e., addition of data from new domains/classes, the classifier tree might need to be retrained and restructured with the new information. This is especially because in the multi-label scenario, addition of one new class indicates the possibility of twice the number of existing label-sets. If there were c_1 number of classes with the possibility of 2^{c1} label-sets, adding one more class would change the number of possible label-sets to 2.2^{c1} or 2^{c1+1}. Hence, this new class definitely needs to be included in the classifier tree.

C. ML-BTC v/s Few Tree-Based Multi-Label Classifiers

Comparing the existing tree-based multi-label classifiers with the proposed ML-BTC, the methodologies are seen to differ at a few elements. Firstly, the concept of label-space partition of the data at each intermediate node is unique which implicitly preserves the label correlations. Due to this practical approach, the proposed method employs a maximum of one classifier at every node (both, intermediate and leaf nodes) and results in the generation of approximate splits. This is a major improvement when compared to existing tree-based techniques, which looks for the best split, i.e., best feature or best class partition, by exhaustively searching the entire space. For example, ML-TREE [9] and ML-FOREST [8] use one-vs-all SVMs for every class at each intermediate node resulting in the use of numerous classifiers. G3P-kEMLC [29] uses a pool of label-powerset classifiers, each predicting a set of k labels. A tree structure is built utilizing context-free grammar and evolutionary algorithm. Another multi-label decision tree model, MLC4.5 [4] finds the best attribute that optimizes information gain. Whereas, the HOMER algorithm [12] creates distinct subsets of classes at every level of the tree and the final leaf nodes contain one class each. For a dataset with a huge number of classes, this would lead

to the formation of a bulky tree with atleast that many number of branches.

Unlike these methods, the proposed ML-BTC prefers to reduce the burden of extensively searching for the best, when it can produce similar results with a good approximation. Unlike methods like ML-TREE, ML-FOREST, HOMER which use one classifier per class, ML-BTC chooses to handle one of the drawbacks of multi-label data, i.e., the possibility of a large number of classes and a larger number of unique label-sets (sometimes in the order of hundreds). In such a case, employing even a small classifier for every class/label-set would increase the size of the overall system. Another novelty in ML-BTC is catering to the class imbalance problem which may lead to uneven class partitions while building the tree. This mainly handled by using appropriate classifiers at different nodes of the tree depending on the amount of available data. Existing tree-based ML classifiers [8], [9] use a single type of classifier for the entire model, say SVM, which may not be customisable for the data at hand. Also, since all the label-sets are not equally well-represented, the ML-BTC tree is not expanded till all the leaf nodes correspond to an unique label-set. Instead, the splitting is strategically stopped at an earlier stage to reduce the unnecessary expansion and complexity of the system. Also, we incorporate the class correlations into the ML-BTC algorithm which might otherwise get overlooked, to assemble the data that have similar label-sets. Keeping these in mind, the proposed technique has drawn inspiration from the existing algorithms and has aimed to handle the deficiencies related to the problem of multi-label classification.

IV. Experiment and Analysis

Various experiments have been performed on the proposed hierarchical model and have been compared with fourteen state-of-the-art algorithms in the domain of multi-label classification. The details of the experiments performed have been discussed in the following sections.

A. Experimental Setup

Fourteen publicly available datasets (http://www.uco.es/kdis/mllresources/) have been used for the experimental analysis of the proposed algorithm, namely, Emotions (music sentiments), Flags (image), CAL500 (audio tagging), CHD49 (medicine), Scene (image), Yeast (biology), Enron (text), Image (image), Water Quality (chemistry), Corel (image), Delicious (web text), Bibtex (text), EUR-Lex (text) and Yelp (web text) datasets. Description of the fourteen benchmark datasets used for experimental analysis has been given in Table I.Emotions, Flags, CAL500 and CHD49 are small sized datasets, Scene, Yeast, Enron, Image and Water quality are medium sized and Corel, Delicious, Bibtex, EUR-Lex and Yelp are large sized datasets. They are available in pre-processed forms, which has been used after normalization for the experiments. Multi-label datasets inherently have the issue of imbalanced classes which is being handled in the proposed work. To provide a better understanding of the class imbalance present in the data, Table I contains maximum and mean imbalance ratios as MaxIR and MeanIR respectively. The proposed work has been executed with 6 runs of 5-fold cross-validation on the fourteen datasets for nine performance measures [1], namely, Hamming loss (HL), ranking loss (RL), one error (OE), subset accuracy (SA), F-measure (FM), macro F1 (MacF1), micro F1 (MicF1), accuracy (Acc), G-mean (GM). Among these, HL, SA, FM, Acc and GM are example-based metrics evaluated instance-wise. They need the crisp output label-set to compute correctness. FM and GM metrics are used to assess algorithms based on the class imbalance problem. RL and OE are ranking-based metrics which only require a ranking of classes and not the actual labels for assessment. Finally, MacF1 and MicF1 are two label-based metrics that compute class-wise performance instead of sample-wise.

TABLE I
Dataset Details

Dataset	*Instances*	*Features*	*Classes*	*Label Card*	*Diversity*	*MaxIR*	*MeanIR*
CAL500	*502*	68	174	26.044	1	88.80	20.58
CHD49	*555*	49	6	2.58	0.531	26.38	5.77
Flags	194	19	7	3.392	0.422	5.88	2.25
Emotions	593	72	6	1.869	0.422	1.78	1.48
Enron	1702	1001	53	3.378	0.442	913.00	73.95
Yeast	2417	103	14	4.237	0.082	53.41	7.20
Water Quality	1060	16	14	5.073	0.778	2.84	1.77
Scene	2407	294	6	1.074	0.234	1.46	1.25
Image	2000	294	5	1.236	0.625	1.42	1.19
Eurlex	19350	5000	201	2.213	0.129	4290.00	536.98
Corel	5000	499	374	3.522	0.635	1120.00	189.57
Delicious	16110	500	983	19.02	0.981	309.29	71.13
Bibtex	7395	1836	159	2.402	0.386	20.43	12.50
Yelp	10810	671	5	1.638	1	7.57	2.88

Results for fourteen well-known multi-label classification techniques have been included in the comparative analysis. Out of these fourteen, four are tree-based algorithms, namely, ML-TREE (2015) [9], ML-FOREST (2016) [8], MLTL-HOMER (2020) [20] and G3P-kEMLC (2020) [29] which have been used for comparing the proposed tree-based ML-BTC. The MLTL-HOMER technique also handles class imbalance problem of ML data, making it apt for comparison. Next four methods are binary or ML ensemble classifiers, CC (2011) [21], ECC (2011) [21], LSF-CC (2020) [22] and RAKEL (2010) [23] which use multiple base classifiers (specifically SVMs) which seem fit for comparison with ML-BTC. Two methods utilize label correlations, LIFT (2014) [26] and ML-LOC (2012) [25]. One is a data transformation technique, BR (2018) [3] which also uses SVM as its base classifier, and the rest are well-known problem adaptation based ML classifiers, namely, ML-KNN (2007) [7], SC-RankSVM (2014) [15] and MLLEM (2016) [19]. Since ML-KNN and SVM are a part of our proposed method, we have focussed on comparing with existing works based on these algorithms. Like ML-BTC, MLLEM and RAKEL also aim to preserve label correlations, hence, has been considered for comparison.

MATLAB 2015b has been used for experiments on a Windows 7 OS with Intel Core i7 processor and 16 GB RAM. The code is available at https://www.researchgate.net/publication/349988491_MLBTC_code. Details of the experimental results are as follows.

B. Analysis of Results

To analyse the performance of the proposed algorithm, six runs of five-fold cross-validation results for all the fifteen methods on fourteen datasets with nine performance measures have been shown in Table II to Table XV for the small (Emotions,

TABLE II
RESULTS FOR EMOTIONS (SMALL)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.3083	0.3323	0.5189	0.2428	0.3322	0.3479	0.4958	0.2763	0.4392
ML-FOREST	0.2755	0.3216	0.3129	0.3532	0.5011	0.4981	0.5124	0.3761	0.4892
MLTL-HOMER	0.2208	0.3425	0.3033	0.2834	0.6181	0.6598	0.6506	0.5217	0.6228
CC	0.2206	0.3852	0.2956	0.2464	0.5616	0.5964	0.5346	0.462	0.5072
ECC	0.2141	0.4029	0.2713	0.2651	0.5618	0.6092	0.5734	0.4819	0.5198
LSF-CC	0.2130	0.3542	0.2721	0.4176	0.6211	0.6398	0.5982	0.4982	0.5821
SC-RankSVM	0.2345	0.2181	0.2873	0.1944	0.4291	0.4342	0.4184	0.3125	0.4019
MLLEM	0.2712	0.2752	0.3679	0.1839	0.4319	0.4421	0.4377	0.3736	0.4229
MLKNN	0.2618	0.3009	0.3355	0.1816	0.4105	0.5257	0.4804	0.4017	0.4382
BR	0.2199	0.3421	0.2715	0.2867	0.6303	0.6579	0.6289	0.5438	0.5967
RAKEL	0.2369	0.3645	0.3052	0.2548	0.5686	0.6078	0.562	0.4837	0.2321
LIFT	0.2161	0.3776	**0.2443**	0.6034	0.6126	0.6331	0.6622	0.5099	0.6869
ML-LOC	0.2743	**0.1895**	0.3285	0.6241	0.6011	0.6180	0.6574	0.5264	0.1170
G3P-kEMLC	0.2291	0.3081	0.2902	0.3093	0.6261	0.6347	0.5831	0.4999	0.2566
ML-BTC	**0.2126**	0.2164	0.2639	**0.6384**	**0.6352**	**0.6601**	**0.6704**	**0.5518**	**0.7018**

TABLE III
RESULTS FOR CAL500 (SMALL)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.1424	0.3545	0.1508	0.0000	0.0465	0.0614	0.2181	0.0333	0.4506
ML-FOREST	0.1891	0.3412	0.1322	0.0639	0.1271	**0.4305**	0.2412	0.0521	0.4761
MLTL-HOMER	0.2166	0.5084	0.6832	0.0000	0.0641	0.3306	0.3255	0.2157	0.3281
CC	0.1581	0.3783	0.3721	0.0000	0.0638	0.329	0.3321	0.2032	0.4391
ECC	0.1583	0.5697	0.3256	0.0000	0.0821	0.192	0.3308	0.2033	0.4253
LSF-CC	0.1411	0.3872	0.3239	0.0000	0.0901	0.2011	0.3421	0.2098	0.4112
SC-RankSVM	0.1552	0.3619	0.2901	0.0000	0.2259	0.2879	0.3282	0.1023	0.3901
MLLEM	0.6253	0.5964	0.5583	0.0000	0.0675	0.3371	0.2862	0.1819	0.4182
MLKNN	0.1523	0.5796	**0.1245**	0.0000	0.0823	0.3092	0.3373	0.2052	0.4711
BR	0.1396	0.4995	0.3693	0.0000	0.0595	0.324	0.3483	0.2161	0.3901
RAKEL	**0.1372**	0.3549	0.2006	0.0000	0.0381	0.0662	0.3261	0.1991	0.2312
LIFT	0.1453	0.4706	0.1982	0.1997	0.1026	0.3248	0.3618	0.2022	0.4051
ML-LOC	0.1443	0.3509	0.2114	0.1988	0.0831	0.3043	0.3631	0.1861	0.4264
G3P-kEMLC	0.1992	0.3562	0.4236	0.0000	0.0982	0.1082	0.3372	0.2018	0.2516
ML-BTC	0.1931	**0.3445**	0.4195	**0.2052**	**0.1343**	0.2458	**0.3639**	**0.2241**	**0.5089**

TABLE IV
RESULTS FOR FLAGS (SMALL)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.4542	0.2524	0.2564	0.3165	0.3459	0.5187	0.6636	0.2043	0.5208
ML-FOREST	0.4051	**0.2415**	0.2411	0.3241	0.3928	0.5721	0.6518	0.2341	0.5312
MLTL-HOMER	0.3751	0.6164	0.2498	0.1341	0.6031	0.6136	0.6517	0.5035	0.4632
CC	0.3532	0.5181	0.2363	0.0927	0.4834	**0.6909**	0.6649	0.5409	0.4372
ECC	0.3683	0.2997	0.2414	0.1093	0.1638	0.6291	0.3308	0.2025	0.4891
LSF-CC	0.3527	0.3512	0.2301	0.1287	0.3213	0.6122	0.4322	0.3544	0.4992
SC-RankSVM	0.5542	0.4482	0.2259	0.1011	0.4882	0.5401	0.5199	0.3732	0.4561
MLLEM	0.4285	0.6104	0.3133	0.1125	0.4095	0.4448	0.4461	0.3211	0.3893
MLKNN	0.3592	0.6562	0.2263	0.0912	0.4795	0.6763	0.6619	0.5182	0.5981
BR	0.3529	0.5769	0.2185	0.1339	0.5051	0.6318	0.6017	0.5061	0.6015
RAKEL	**0.3645**	0.4632	0.2281	0.1236	0.4551	0.6739	0.6513	0.5104	0.5801
LIFT	0.3533	0.5551	0.2256	0.5291	0.4897	0.6513	0.6543	0.5068	0.6031
ML-LOC	0.3497	0.3709	0.3846	0.5221	0.5011	0.6457	0.6231	0.4965	0.6092
G3P-kEMLC	0.3731	0.3982	0.2315	0.2091	0.5009	0.6019	0.6511	0.4572	0.5982
ML-BTC	**0.3438**	0.3135	**0.2158**	**0.5922**	**0.5085**	0.6114	**0.6697**	**0.5194**	**0.6115**

TABLE V
RESULTS FOR CHD49 (SMALL)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.3562	0.2516	0.3831	0.5821	0.3627	0.6157	0.5941	0.4655	0.6156
ML-FOREST	0.3501	0.2981	0.3998	0.5901	0.3782	0.6102	0.5899	0.4561	0.5782
MLTL-HOMER	0.3245	0.3414	0.2924	0.1441	0.4214	0.6276	0.6042	0.4893	0.5688
CC	0.3325	0.4711	**0.0831**	0.1712	0.4102	0.6309	0.6117	0.4713	0.5794
ECC	0.3322	0.3928	0.0992	0.1712	0.3990	0.6333	0.6087	0.4773	0.5893
LSF-CC	0.3521	0.3092	0.3981	0.2091	0.4281	0.6009	0.5983	0.4664	0.5899
SC-RankSVM	0.4182	0.3708	0.4476	0.0306	0.4186	0.4912	0.4618	0.3498	0.5783
MLLEM	0.4301	0.2851	0.4152	0.0054	0.4092	0.5637	0.5763	0.3981	0.5711
MLKNN	0.3421	**0.2308**	0.2274	0.1459	0.4044	0.6391	0.6107	0.4819	0.6002
BR	0.3262	0.2563	0.2725	0.1748	0.4205	0.6288	0.6138	0.4803	0.6032
RAKEL	0.3484	0.3739	0.1606	0.1525	0.4161	0.6336	0.6058	0.4847	0.6109
LIFT	0.3303	0.5421	0.1412	0.5834	0.4066	0.6259	0.6006	0.4814	0.6152
ML-LOC	0.3438	0.2451	0.3194	0.5661	0.3152	0.5918	0.6102	0.4339	0.6167
G3P-kEMLC	0.3492	0.3569	0.3078	0.1892	0.4075	0.4933	0.5714	0.4557	0.6005
ML-BTC	**0.3244**	0.2659	0.3463	**0.6054**	**0.4338**	**0.6398**	**0.6139**	**0.4901**	**0.6288**

TABLE VI
RESULTS FOR SCENE (MEDIUM)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.1982	0.1377	0.2871	0.6618	0.5681	0.5784	0.7071	0.6067	0.6573
ML-FOREST	0.1872	0.1102	0.2701	0.6682	0.5743	0.5889	0.6632	0.6033	0.6392
MLTL-HOMER	0.1378	0.3036	0.3012	0.5933	0.6893	0.6797	0.6873	0.6634	0.6473
CC	0.1372	0.3537	0.3575	0.5526	0.6974	0.6898	0.6468	0.6231	0.6281
ECC	0.1452	0.3555	0.2712	0.5929	0.7029	0.7015	0.6941	0.6182	0.6709
LSF-CC	0.1324	0.2743	0.2572	0.5821	0.6352	0.6401	0.6478	0.6092	0.6382
SC-RankSVM	0.5629	0.3994	0.2846	0.1099	0.4107	0.4095	0.4186	0.3025	0.4089
MLLEM	0.1384	0.3535	0.2918	0.6143	0.7099	0.6762	0.6901	0.6102	0.6875
MLKNN	0.1452	0.2191	0.2411	0.6512	0.7217	0.7122	0.7219	0.6981	0.6901
BR	0.1343	0.3796	0.2671	0.6436	0.7161	0.7091	0.7133	0.6958	0.6877
RAKEL	0.1191	0.3707	0.2443	0.6568	0.7246	0.7126	0.7156	0.6808	0.6893
LIFT	0.1122	0.3198	**0.1247**	0.7166	0.7209	0.7116	0.7256	0.6796	0.6904
ML-LOC	0.2035	**0.0576**	0.1659	0.7163	0.6078	0.6088	0.7245	0.6878	0.6901
G3P-kEMLC	0.1342	0.2091	0.2691	0.6471	0.7082	0.7062	0.7162	0.6811	0.6782
ML-BTC	**0.1291**	0.1336	0.2658	**0.7291**	**0.7248**	**0.7153**	**0.7268**	**0.7093**	**0.7175**

TABLE VII
RESULTS FOR YEAST (MEDIUM)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.2994	0.2145	0.5021	0.2067	0.1912	0.1945	0.5062	0.1346	0.4341
ML-FOREST	0.2901	0.2119	0.4952	0.2198	0.1091	0.2098	0.5165	0.1472	0.5262
MLTL-HOMER	0.2817	0.4687	0.3128	0.1643	0.3303	**0.6515**	0.6407	0.4273	0.6397
CC	0.2611	0.3538	0.4034	0.1622	0.3364	0.6389	0.6145	0.4073	0.6045
ECC	0.2718	0.3637	0.3452	0.1849	0.3383	0.6376	0.6137	0.4152	0.6022
LSF-CC	0.3368	0.3526	0.3902	0.1772	0.3362	0.5261	0.6092	0.4242	0.5902
SC-RankSVM	0.3027	0.3334	0.2989	0.0008	0.3432	0.5264	0.5141	0.3728	0.5982
MLLEM	0.2452	0.3618	0.5006	0.0095	0.1716	0.1909	0.2031	0.1783	0.2381
MLKNN	0.2593	0.2882	0.2939	0.1899	0.3371	0.6432	0.6167	0.4182	0.6002
BR	0.2496	0.3827	0.2928	0.1506	0.3405	0.6363	0.6113	0.4027	0.5993
RAKEL	0.2981	0.4543	0.2889	0.1593	0.3446	0.6404	0.6189	0.4106	0.5912
LIFT	0.2322	0.4657	0.2974	0.5113	0.3177	0.6261	0.7182	0.3925	0.6445
ML-LOC	0.2493	**0.1911**	0.2989	0.5157	0.3149	0.5852	0.7125	0.4261	0.6374
G3P-kEMLC	0.2672	0.3051	0.3082	0.5083	0.3021	0.5463	0.6281	0.4096	0.6082
ML-BTC	**0.2342**	0.2671	**0.2891**	**0.5275**	**0.3443**	0.5605	**0.6339**	**0.4319**	**0.6425**

TABLE VIII
RESULTS FOR ENRON (MEDIUM)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.0891	0.3438	0.6029	0.1218	0.0491	0.1363	0.2971	0.1784	0.2877
ML-FOREST	0.0921	0.3718	0.4582	0.1324	0.0535	0.1372	0.3092	0.1891	0.2983
MLTL-HOMER	0.0671	0.4985	0.4156	0.1017	0.1926	0.3662	0.4752	0.2707	0.4872
CC	0.0585	0.5142	0.1152	0.1211	0.2066	0.3694	0.5147	0.3058	0.5019
ECC	0.0584	0.5407	0.1382	0.1198	**0.2099**	0.3523	0.5192	0.3097	0.5008
LSF-CC	0.0899	0.5012	0.1109	0.1213	0.1672	0.3281	0.3872	0.2562	0.3982
SC-RankSVM	0.0971	0.4691	0.6921	0.0059	0.1448	0.2587	0.2209	0.1499	0.2432
MLLEM	0.8526	0.7816	0.8011	0.2018	0.1019	0.1306	0.129	0.0699	0.1625
MLKNN	0.0627	0.7791	0.5081	0.1141	0.0545	0.2807	0.3057	0.2461	0.3241
BR	0.0661	0.8417	0.3532	0.1251	0.1892	0.3692	0.5043	0.2962	0.5092
RAKEL	0.0562	0.6089	0.1852	0.1281	0.1941	0.3245	0.5206	0.3098	0.5021
LIFT	**0.0461**	0.5064	**0.0924**	0.3516	0.2077	0.3666	**0.6952**	0.3101	0.5179
ML-LOC	0.0531	0.3276	0.2324	0.3403	0.1763	0.3722	0.6705	0.3068	0.5181
G3P-kEMLC	0.0673	0.3569	0.4081	0.2081	0.1572	0.3298	0.4381	0.3022	0.5096
ML-BTC	0.0889	**0.3257**	0.3892	**0.3622**	0.1679	**0.3724**	0.4557	**0.3119**	**0.5193**

TABLE IX
RESULTS FOR IMAGE (MEDIUM)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.4112	0.2007	0.3298	0.5423	0.4901	0.4843	0.5001	0.4595	0.4078
ML-FOREST	0.3701	0.3811	0.4421	0.4253	0.1899	0.4361	0.5002	0.2831	0.5435
MLTL-HOMER	0.3656	0.6228	0.3405	0.4545	0.5545	0.5526	0.6192	0.5071	0.5699
CC	0.2829	0.5362	0.2039	0.3979	0.5452	0.5436	0.5497	0.5108	0.4673
ECC	0.3501	0.6316	0.2705	0.4341	0.5552	0.5512	0.5937	0.5531	0.4891
LSF-CC	0.3527	0.3821	0.2531	0.0212	0.3982	0.4803	0.4577	0.3225	0.4903
SC-RankSVM	0.4044	0.8241	0.3491	0.1231	0.5457	0.5367	0.5393	0.4195	0.2549
MLLEM	0.4144	0.8043	0.3745	0.4555	0.5616	0.5595	0.5677	0.4993	0.1092
MLKNN	0.2848	0.7992	0.3297	0.4792	0.5483	0.5365	0.5951	0.5058	0.5226
BR	0.3031	0.8213	0.3281	0.4761	0.5406	0.5571	0.6226	0.5053	0.4539
RAKEL	0.3013	0.6842	0.3155	0.4951	0.5354	0.5613	0.6258	0.5124	0.4401
LIFT	**0.2075**	0.5077	**0.1154**	0.5583	0.5581	0.5474	0.6051	0.4817	0.5297
ML-LOC	0.3298	**0.1432**	0.2682	0.5657	0.5481	0.5494	**0.6319**	0.5035	0.6026
G3P-kEMLC	0.3271	0.4821	0.4102	0.5023	0.5493	0.5692	0.5825	0.5193	0.4983
ML-BTC	0.2189	0.2429	0.4008	**0.5729**	**0.5637**	**0.5713**	0.5979	**0.5241**	**0.6089**

TABLE X
RESULTS FOR WATER QUALITY (MEDIUM)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.3729	0.3929	0.4545	0.4101	0.1868	0.4262	0.4919	0.2781	0.5301
ML-FOREST	0.3701	0.3883	0.4342	0.4198	0.2012	0.4311	0.5002	0.3019	0.5291
MLTL-HOMER	0.3656	0.6215	0.4151	0.0028	0.4026	0.5533	0.5281	0.3397	0.5699
CC	0.3992	0.3486	0.1809	0.0162	0.3822	0.4939	0.4549	0.3264	0.4762
ECC	0.3501	0.4269	0.1726	0.0171	0.3767	0.4838	0.4466	0.3187	0.4553
LSF-CC	0.3499	0.4362	0.1766	0.0391	0.3361	0.3221	0.3872	0.3021	0.3992
SC-RankSVM	0.4044	0.7015	0.3953	0.0047	0.2018	0.2105	0.1981	0.1421	0.2029
MLLEM	0.4144	0.6751	0.4566	0.0047	0.1036	0.1058	0.0987	0.0701	0.1034
MLKNN	0.3448	0.7283	0.2406	0.0217	0.4247	0.4521	0.5181	0.3218	0.5224
BR	0.3531	0.7326	0.3198	0.0094	0.3594	0.4786	0.4351	0.3099	0.4555
RAKEL	0.3513	0.4955	0.2019	0.0151	0.3581	0.4599	0.4151	0.2964	0.4198
LIFT	0.3458	0.7343	**0.1141**	0.3699	0.2408	0.4143	0.5215	0.2562	0.5297
ML-LOC	0.3501	**0.2847**	0.3711	0.4671	0.4012	0.5065	0.5348	0.3429	0.6326
G3P-kEMLC	0.3581	0.4838	0.4192	0.0921	0.3762	0.4689	0.4266	0.3201	0.4226
ML-BTC	**0.3436**	0.3177	0.4039	**0.4796**	**0.4273**	**0.4955**	**0.5365**	**0.3431**	**0.5769**

TABLE XI
RESULTS FOR DELICIOUS (LARGE)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.0213	0.7871	0.6786	0.0537	0.0538	0.0559	0.2065	0.0284	0.3333
ML-FOREST	0.0201	0.7628	0.4532	0.0673	0.0598	0.0582	0.2383	0.0293	0.3621
MLTL-HOMER	0.0308	0.3297	0.7291	0.0007	0.1045	0.2904	0.2345	0.1577	0.3092
CC	0.0287	0.1781	0.1459	0.0025	0.1023	0.2257	0.2243	0.1467	0.2561
ECC	0.0269	0.1864	**0.1226**	0.0025	0.1123	0.2466	0.2241	0.1466	0.2583
LSF-CC	0.0208	0.2034	0.2387	0.0031	**0.1298**	0.2473	0.2322	0.1372	0.2313
SC-RankSVM	0.0563	0.3721	0.6873	0.0032	0.0653	0.1982	0.1654	0.1221	0.1782
MLLEM	0.0292	0.5961	0.9757	0.0009	0.0126	0.0321	0.1326	0.0436	0.1028
MLKNN	0.0278	0.7371	0.3573	0.0019	0.0414	0.1829	0.1651	0.1051	0.1322
BR	0.0189	0.8824	0.3414	0.0012	0.0471	0.1933	0.1691	0.1089	0.1547
RAKEL	**0.0182**	0.2002	0.1426	0.0026	0.1013	0.2082	0.2181	0.1429	0.2093
LIFT	0.0489	0.3265	0.6632	0.0209	0.0782	0.0835	0.1782	0.1282	0.1872
ML-LOC	0.0193	**0.1637**	0.6034	0.0441	0.0533	0.0465	0.1963	0.0253	0.3925
G3P-kEMLC	0.0452	0.3082	0.6832	0.0018	0.0732	0.2109	0.2271	0.1342	0.2481
ML-BTC	0.0381	0.2816	0.6549	**0.2152**	0.0824	**0.2488**	**0.2439**	**0.1609**	**0.4012**

TABLE XII
RESULTS FOR COREL (LARGE)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.0344	**0.2824**	0.7633	0.1505	0.1495	0.1475	0.1015	0.0897	0.2457
ML-FOREST	0.0332	0.3281	0.7601	0.1023	0.1499	0.1501	0.1036	0.0901	0.2342
MLTL-HOMER	0.0227	0.3187	0.7636	0.0061	0.0464	0.1709	0.1074	0.1055	0.2231
CC	0.0225	0.3335	0.5130	0.0076	0.0476	0.1571	0.1024	0.0986	0.2291
ECC	0.0223	0.3454	0.5590	0.0072	0.0471	0.1691	0.1114	0.0998	0.2353
LSF-CC	0.0243	0.4326	0.5722	0.0083	0.0542	0.1503	0.1126	0.0953	0.2245
SC-RankSVM	0.0256	0.4692	0.5328	0.0091	0.0782	0.1461	0.1092	0.0896	0.2135
MLLEM	0.0951	0.6971	0.9738	0.0012	0.0177	0.0194	0.0194	0.0098	0.0162
MLKNN	0.0309	0.6165	0.7291	0.0004	0.0051	0.1076	0.1014	0.0656	0.1132
BR	0.0292	0.7519	0.6412	0.0066	0.0227	0.1701	0.1116	0.1053	0.1642
RAKEL	0.0316	0.3651	0.6458	0.0092	0.0454	**0.1722**	0.1077	0.1035	0.1527
LIFT	0.0228	0.9893	**0.0153**	0.0214	0.1493	0.0244	0.0449	0.0146	0.0451
ML-LOC	0.0432	0.3542	0.8721	0.0127	0.0321	0.0453	0.1064	0.0873	0.2099
G3P-kEMLC	0.0312	0.3581	0.8089	0.0032	0.1093	0.1321	0.1019	0.1071	0.1823
ML-BTC	**0.0221**	0.3101	0.7592	**0.1566**	**0.1527**	0.1521	**0.1129**	**0.1096**	**0.2483**

TABLE XIII
RESULTS FOR BIBTEX (LARGE)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.0314	0.3784	0.4511	0.1567	0.1414	0.3257	0.2549	0.2367	0.3761
ML-FOREST	0.0301	0.3421	0.4982	0.2001	0.1231	0.3091	0.4021	0.2319	0.3533
MLTL-HOMER	0.0312	0.4037	0.5857	0.0949	0.1497	0.3186	0.3239	0.2139	0.3382
CC	0.0315	0.4131	0.2597	0.1532	0.1285	0.4331	0.4058	0.2247	0.3502
ECC	0.0309	0.4245	**0.2504**	0.1559	0.1259	0.4293	0.4065	0.2155	0.3609
LSF-CC	0.0308	0.4372	0.3092	0.1872	0.1099	0.4019	0.4122	0.2281	0.3521
SC-RankSVM	0.0391	0.5764	0.6972	0.1789	0.1088	0.1342	0.2002	0.1543	0.2091
MLLEM	0.0829	0.8182	0.7921	0.0871	0.0849	0.1223	0.1481	0.1292	0.1562
MLKNN	0.0762	0.5846	0.6531	0.0552	0.0562	0.1882	0.1837	0.1433	0.1892
BR	0.6823	0.3961	0.3602	0.2151	0.1504	0.4313	**0.4519**	0.2139	0.3728
RAKEL	0.0321	0.5261	0.3505	0.1813	0.1232	**0.4378**	0.4275	0.2283	0.3592
LIFT	0.0308	0.7466	0.3607	0.2734	0.1424	0.2885	0.4072	0.2295	0.3825
ML-LOC	0.0312	**0.2858**	0.7872	0.1201	0.0563	0.1264	0.2128	0.0961	0.2115
G3P-kEMLC	0.0304	0.5628	0.5879	0.2415	0.1035	0.2473	0.2891	0.2029	0.3741
ML-BTC	**0.0299**	0.5524	0.5736	**0.2919**	**0.1586**	0.2597	0.3014	**0.2391**	**0.3911**

TABLE XIV
RESULTS FOR EUR-LEX (LARGE)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.2478	0.1378	0.7369	0.0489	0.2817	0.0493	0.0257	0.0254	0.1405
ML-FOREST	0.2082	0.13471	0.7223	0.0557	0.2899	0.0532	0.0281	0.0277	0.1498
MLTL-HOMER	0.0353	0.4985	0.5064	0.1147	0.2091	0.1407	0.1443	**0.1787**	0.1402
CC	0.0445	0.7851	0.5373	0.1479	0.2908	0.1376	0.2631	0.1786	0.1601
ECC	0.0405	0.7861	0.4433	0.1401	0.2914	0.1415	**0.2634**	0.1783	0.1608
LSF-CC	0.0392	0.7212	0.3981	0.1321	0.2739	0.1548	0.1621	0.1534	0.1512
SC-RankSVM	0.1028	0.7328	0.7621	0.1092	0.1087	0.1101	0.0081	0.0721	0.1024
MLLEM	0.1986	0.7736	0.7759	0.0021	0.0208	0.0218	0.0217	0.0112	0.0342
MLKNN	0.0367	0.9355	0.5332	0.1316	0.2154	0.1509	0.1115	0.1427	0.1526
BR	0.0353	0.8478	0.3221	0.1201	0.1953	0.1417	0.1132	0.1515	0.1526
RAKEL	0.0346	0.7107	0.3543	0.0982	0.2914	0.1281	0.1775	0.1718	0.1631
LIFT	0.0411	0.5628	**0.3182**	0.1834	0.2823	0.1521	0.1315	0.1026	0.1609
ML-LOC	0.0337	**0.1312**	0.7621	0.1316	0.2899	0.1575	0.2374	0.0943	0.1606
G3P-kEMLC	0.0423	0.6619	0.7732	0.1082	0.2781	0.1462	0.1247	0.0932	0.1622
ML-BTC	**0.0315**	0.6015	0.7689	**0.1551**	**0.2968**	**0.1579**	0.1334	0.1083	**0.1699**

TABLE XV
RESULTS FOR YELP (LARGE)

	HL	RL	OE	SA	MacF1	MicF1	FM	Acc	GM
ML-TREE	0.2649	0.2028	0.2816	0.5364	0.3104	0.5198	0.7066	0.4639	0.6203
ML-FOREST	0.2561	0.2003	0.2781	0.5463	0.3201	0.5261	0.7098	0.4873	0.6571
MLTL-HOMER	0.1856	0.3269	0.2432	0.3789	0.6513	0.7076	0.658	0.5842	0.6753
CC	0.1805	0.3847	0.1432	0.3905	0.6511	0.7063	0.652	0.5821	0.6657
ECC	0.1804	0.3524	0.1583	0.3882	0.6513	0.7061	0.6503	0.5803	0.6601
LSF-CC	0.1782	0.3658	0.1408	0.3801	0.6572	0.6741	0.6577	0.5899	0.6709
SC-RankSVM	0.2584	0.2064	0.3254	0.2291	0.4526	0.4539	0.3919	0.3402	0.3892
MLLEM	0.3275	0.3236	0.5555	0.0745	0.4638	0.4766	0.4189	0.3902	0.4093
MLKNN	0.274	0.2626	0.3351	0.1904	0.1581	0.3299	0.2664	0.2281	0.2761
BR	0.1785	0.1556	0.2476	0.3899	0.6348	0.7035	0.6444	0.5757	0.6579
RAKEL	0.1663	0.2271	0.1872	0.4555	0.6855	0.7266	0.6908	0.6271	0.6866
LIFT	0.1951	0.3981	0.1641	0.6336	0.6203	0.6961	0.7037	0.5846	0.6749
ML-LOC	**0.1095**	0.1579	0.1425	0.6345	**0.6735**	**0.7162**	0.6988	**0.6513**	0.6844
G3P-kEMLC	0.2231	0.1629	0.1572	0.4781	0.5893	0.6521	0.6858	0.5538	0.6583
ML-BTC	0.2167	**0.1461**	**0.1387**	**0.6436**	0.5944	0.6602	**0.7104**	0.5619	**0.6872**

CAL500, Flags, CHD49), medium (Scene, Yeast, Enron, Image, Water quality) and large (Delicious, Corel, Bibtex, EUR-Lex, Yelp) datasets respectively. The values of the threshold have been fixed for all datasets, $H_{threshold} = \frac{H_{initial}}{5}$ and $C_{threshold} = \frac{N_{tr}}{5}$, where, $H_{initial}$ is the ML entropy of the data at the root node and N_{tr} is the training sample size.

At the first glance, the proposed ML-BTC is seen to perform better than the other fourteen algorithms for all fourteen datasets in most of the cases. specifically the example-based and the label-based metrics. Only for the ranking-based metrics, RL and OE, ML-BTC does not score the highest in most cases, but lies very close to the best performing algorithm. Among all the performance metrics used for comparison, one, in particular, is quite difficult to achieve. While the others base their outcome on partial correctness, the "subset accuracy" metric determines the full correctness, i.e., the correct prediction of the entire label-set. For multi-label data, this is a difficult feat to achieve, even for a small number of test instances. However, from the results, it is seen that the proposed ML-BTC surpasses all the other methods by a significant margin for SA. Especially for the CAL500 dataset, where eleven of other methods have scored SA $= 0$. Similarly, metrics F-Measure and G-Mean are used to better assess the class imbalance issue in ML data. It is seen that for most of datasets, ML-BTC has surpassed the others in terms of FM and GM, indicating its ability to handle class imbalance.

From a dataset perspective, ML-BTC has performed equally well for datasets of small, medium and large sizes. Only for the ranking-based metrics, ML-LOC is seen to have a better RL, whereas, LIFT shows a better OE compared to the other methods. ML datasets in general have the drawback of class imbalance, and all the fourteen datasets used for experimentation are quite imbalanced (details given in the supplementary material). Out of the fourteen datasets, CAL500, Delicious and Yelp in particular have $1, 0.981$ and 1 diversity respectively, indicating a high ratio of distinct label-sets to number of instances. This makes the train set and test set completely different and thus are likely to get misclassified. Yet, the proposed ML-BTC performs substantially well compared to the other algorithms, especially for SA, FM and GM metrics.

For datasets like CAL500, Delicious and Yelp, where the training set and test set will have mostly distinct label-sets, the propagation of error from the root node to the leaf nodes in a tree structure can be a major concern. Any misclassification at an intermediate node might propagate to the leaf nodes, affecting the final classification results. However, through the performance achieved by ML-BTC, we can say that it has not succumbed to the problem of error propagation. In the proposed ML-BTC, since there is a separate binary classifier at every intermediate node which learns different decision boundaries, any data point that is misclassified at one level can be rectified in the later steps. At every node, since an approximate partition is being estimated, we assume some amount of misclassification to occur. However, the novel approach of label-space partitioning ensures that data from similar classes will belong to the same side of the partition. Since the final classification does not take place until the sample reaches some leaf node, even if some data have been misclassified at the intermediate nodes, it still has chance to be correctly classified at the leaf nodes. Either an entire leaf node is allocated for that label-set, or the ML classifiers at the leaf nodes definitely handle these anomalous points, thus preventing the error from affecting the results. To evaluate the sensitivity of ML-BTC to the two parameters, namely multi-label entropy, H, and sample cardinality, C, that orchestrate the building of the tree-like structure, experiments have been performed for different combinations of $H_{threshold}$ and $C_{threshold}$. Fig. 8 depicts the G-Mean values (Y-axis) obtained for the fourteen

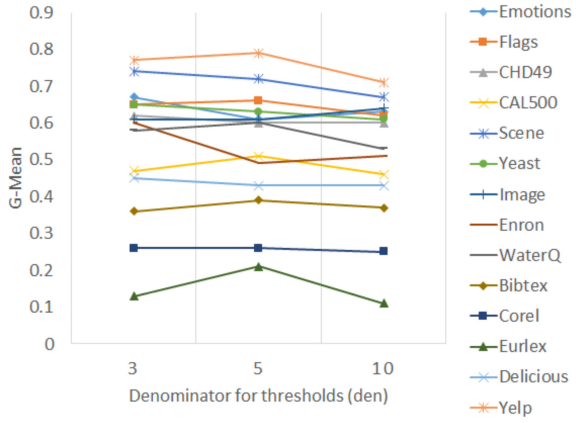

Fig. 8. Effect of the two parameters $H_{threshold}$ and $C_{threshold}$ on G-Mean metric.

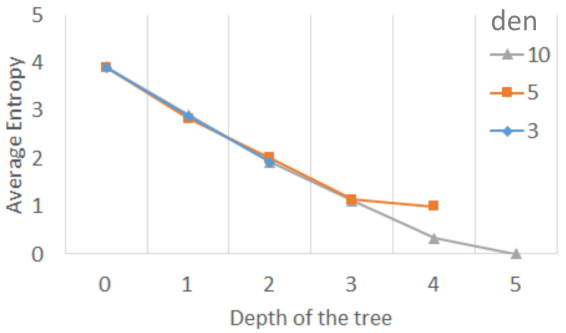

Fig. 9. Average entropy at every depth of the ML-BTC tree for different $(H_{threshold}, C_{threshold})$ combination for scene dataset.

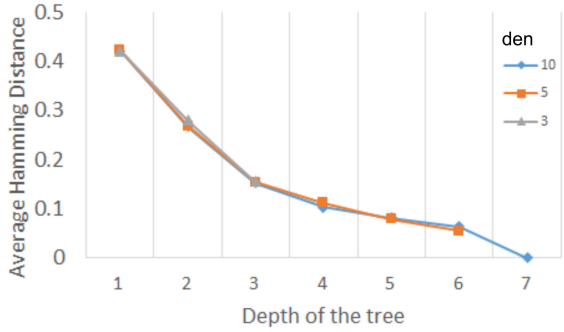

Fig. 10. Average Hamming distance at every depth of the ML-BTC tree for different $(H_{threshold}, C_{threshold})$ combination for Emotions dataset.

TABLE XVI
TWO-TAILED WILCOXON SIGNED-RANK TEST STATISTICS FOR ALL THE METHODS V/S PROPOSED ML-BTC (BASED ON ACCURACY)

ML-BTC v/s method	*p* Value	Value of sign rank
ML-TREE	0.000122	105
ML-FOREST	0.000122	105
MLTL-HOMER	0.067627	82
CC	0.172607	75
ECC	0.135254	77
LSF-CC	0.019043	89
SC-RankSVM	0.000122	105
MLLEM	0.000122	105
MLKNN	0.003052	97
BR	0.040039	85
RAKEL	0.090576	80
LIFT	0.002319	98
ML-LOC	0.008545	93
G3P-kEMLC	0.000122	105

datasets by considering three combinations of $H_{threshold}$ and $C_{threshold}$. The thresholds have been set to a fraction of the initial entropy, $(H_{initial})$, and training sample size, N_{tr}. In each case, $H_{threshold} = \frac{H_{initial}}{den}$ and $C_{threshold} = \frac{N_{tr}}{den}$. The X-axis represents three values for the denominator of thresholds, $den = 3, 5, 10$. On the X-axis, when $den = 3$ it depicts results from ML-BTC experimented with parameters, $H_{threshold} = \frac{H_{initial}}{3}$ and $C_{threshold} = \frac{N_{tr}}{3}$, and so on. Analysis of the performance of the proposed algorithms for the various combinations of thresholds on the datasets shows that the G-Mean does not vary drastically with the variation of the thresholds. Within the range tested, the exact choice of thresholds do not seem crucial. This shows that the proposed algorithm is not too dependent on the thresholds and can perform efficiently with any approximate value chosen within the range.

To investigate the tree structure built by ML-BTC algorithm, in Fig. 9 and Fig. 10 plots of the average entropy and average Hamming distance at each depth of the trees have been shown respectively. These results have been obtained from different runs with different training sets. Three combinations of thresholds have been considered. $H_{threshold} = \frac{H_{initial}}{den}$ and $C_{threshold} = \frac{N_{tr}}{den}$, where den = 3, 5, 10. From the graphs, it is seen that the average entropy and Hamming distance at the nodes of a particular depth of the tree keeps systematically decreasing. Entropy determines the diversity of the data, while Hamming distance portrays the existing closeness and label dependencies. The tree building is effectively stopped before all the nodes attain entropy and Hamming distance 0.

To evaluate the performance of the proposed method even further, the non-parametric two-tailed Wilcoxon signed rank test [31] (Table XVI) has been performed on the accuracy metric for all datasets against all the other methods. It indicates that with alpha=0.20, $T_{Wilcoxon}(14)$=31, the proposed ML-BTC is statistically superior than all the other algorithms for the accuracy metric.

Additionally, to analyse the computational cost of the proposed method, comparison of all the run times has been made for all datasets and all algorithms (Table in supplementary section). ML-BTC is seen to have average computational time as compared to the other models. However, it is seen that the ML-BTC surpasses most of the similar performing methods like MLLOC, ML-FOREST, G3P-kEMLC, SC-RankSVM, LSF-CC, ECC etc with respect to computational time. Both train and test times for ML-BTC have been shown in the table, and it is seen that the test time is very low as compared to the train time. If the training of the model is done offline, the computational time for actual ML classification by ML-BTC is quite low.

Thus, analysing the overall performance of the proposed method, it can be said that the ML-BTC fares well above the other state-of-the-art algorithms for multi-label classification. Significant improvement in most cases when compared to the

tree-based models, ensemble classifiers, data transformation techniques and other standard ML algorithms can be observed.

V. Conclusion

A binary tree of classifiers had been proposed in this article which efficiently performs multi-label data classification. Few well-identified bottlenecks in the field of multi-label classification have been attempted to be handled by the proposed model. The unique binary tree structure builds itself in the training phase by approximately partitioning the data into two discrete chunks without performing an exhaustive search. This is achieved by a novel label-space partitioning approach that strives to preserve class dependencies implicitly. A suitable classifier is trained to learn the partitioned data. Moreover, the inherent problem of imbalanced classes in multi-label data has been looked after by broadly partitioning the data if possible, otherwise using different classifiers specifically suited for the unevenly split data at an internal node and some proposed parameters. Two appropriate parameters have been included to help the action-decision at every node of the tree. As the tree grows in the training phase, restrictions are imposed strategically to avoid redundant and unnecessary branches. Finally, the leaf nodes are capable of assigning the final labels to samples taking the class dependencies and imbalance into consideration. In this way, once the binary tree of classifiers is formed it can be successfully used for efficient multi-label classification. The experimental results for the proposed model in comparison to fourteen other state-of-the-art algorithms display improvement in the majority of the scenarios thus establishing the success of the proposed model. In future, this method can be extended by making the parameters adaptive to further reduce the algorithm's dependence on them.

Acknowledgment

We sincerely thank all the reviewers for their valuable comments, which has helped us immensely to improve the overall article.

References

[1] F. Herrera, F. Charte, A. J. Rivera, and M. J. Del Jesus, *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Berlin, Germany: Springer, pp. 13–63, 2016.

[2] T. Gonçalves and P. Quaresma, "A preliminary approach to the multilabel classification problem of portuguese juridical documents," in *Proc. Portuguese Conf. Artif. Intell.*, 2003, pp. 435–444.

[3] M. L. Zhang, Y. K. Li, X. Y. Liu, and X. Geng, "Binary relevance for multi-label learning: An overview," *Front. Comput. Sci.*, vol. 12, no. 2, pp. 191–202, 2018.

[4] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, 2001, pp. 42–53.

[5] A. Law, K. Chakraborty, and A. Ghosh, "Functional link artificial neural network for multi-label classification," in *Proc. Int. Conf. Mining Intell. Knowl. Exploration*, 2017, pp. 1–10.

[6] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: Models, experimental study and prospects," *Inf. Fusion*, vol. 44, pp. 33–45, 2018.

[7] M. L. Zhang and Z. H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, 2007.

[8] Q. Wu, M. Tan, H. Song, J. Chen, and M. K. Ng, "ML-FOREST: A multi-label tree ensemble method for multi-label classification," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2665–2680, Oct. 2016.

[9] Q. Wu, Y. Ye, H. Zhang, T. W. S. Chow, and S. S. Ho, "ML-TREE: A tree-structure-based approach to multilabel learning." *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 430–443, Mar. 2015.

[10] P. Kaur and A. Gosain, "Robust hybrid data-level sampling approach to handle imbalanced data during classification," *Soft Comput.*, vol. 24, no. 20, pp. 15715–15732, 2020.

[11] P. Kaur and A. Gosain, "GT2FS-SMOTE: An intelligent oversampling approach based upon general Type-2 fuzzy sets to detect web spam," *Arabian J. Sci. Eng.*, vol. 46, no. 4, pp. 3033–3050, 2021.

[12] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD Workshop Mining Multidimensional Data*, vol. 21, 2008, pp. 53–59.

[13] A. Law and A. Ghosh, "Multi-label classification using a cascade of stacked autoencoder and extreme learning machines," *Neurocomputing*, vol. 358, pp. 222–234, 2019.

[14] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2002, pp. 681–687.

[15] J. Wang, J. Feng, X. Sun, S.-S. Chen, and B. Chen, "Simplified constraints Rank-SVM for multi-label classification," in *Proc. Chin. Conf. Pattern Recognit.*, 2014, pp. 229–236.

[16] Y. Yuan, J. Fang, X. Lu, and Y. Feng, "Remote sensing image scene classification using rearranged local features," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1779–1792, Mar. 2019.

[17] X. Zheng, Y. Yuan, and X. Lu, "A deep scene representation for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4799–4809, Jul. 2019.

[18] Q. Wang, J. Gao, and Y. Yuan, "A joint convolutional neural networks and context transfer for street scenes labeling," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1457–1470, May 2018.

[19] K. Kimura, M. Kudo, and L. Sun, "Simultaneous nonlinear label-instance embedding for multi-label classification," in *Joint IAPR Int. Workshops Stat. Techn. Pattern Recognit. Struct. Syntactic Pattern Recognit.*. 2016, pp. 15–25.

[20] R. M. Pereira, Y. M. Costa, and C. N. Silla Jr, ": A multi-label approach for the tomek link undersampling algorithm," *Neurocomputing*, vol. 383, pp. 95–105, 2020.

[21] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, 2011.

[22] W. Weng, D. H. Wang, C. L. Chen, J. Wen, and S. X. Wu, "Label specific features-based classifier chains for multi-label classification," *IEEE Access*, vol. 8, pp. 51265–51275, 2020.

[23] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random K-labelsets for multilabel classification," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 1079–1089, Jul. 2011.

[24] Y. Xia, K. Chen, and Y. Yang, "Multi-label classification with weighted classifier selection and stacked ensemble," *Inf. Sci.*, vol. 557, pp. 421–442, 2021.

[25] S. J. Huang and Z. H. Zhou, "Multi-label learning by exploiting label correlations locally," in *Proc. AAAI Conf. Artif. Intell.*, 2012, pp. 949–955.

[26] M. L. Zhang and L. Wu, "LIFT: Multi-label learning with label-specific features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 107–120, Jan. 2015.

[27] H. C. Kim, J. H. Park, D. W. Kim, and J. Lee, "Multilabel naïve bayes classification considering label dependence," *Pattern Recognit. Lett.*, vol. 136, pp. 279–285, 2020.

[28] R. C. Barros, M. P. Basgalupp, A. C. De Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Trans. Syst., Man, Cybern., Part C. Appl. Rev.*, vol. 42, no. 3, pp. 291–312, May 2012.

[29] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Tree-shaped ensemble of multi-label classifiers using grammar-guided genetic programming," in *Proc. IEEE Congr. Evol. Comput.*, 2020, pp. 1–8.

[30] B. Zang, R. Huang, L. Wang, J. Chen, F. Tian, and X. Wei, "An improved KNN algorithm based on minority class distribution for imbalanced dataset," in *Proc. Int. Comput. Symp.*, 2016, pp. 696–700.

[31] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2653–2688, 2017.

Anwesha Law (Student Member, IEEE) completed the B. Tech. degree in computer science and engineering from Sikkim Manipal Institute of Technology, Sikkim, India, in 2011 and the M. Tech. degree in computer science & engineering from Jadavpur University, Kolkata, India, in 2013. She is working toward the Ph.D. degree in computer science as a Senior Research Fellow with the Machine Intelligence Unit, Indian Statistical Institute. Her research interests include multi-label learning, artificial neural networks, pattern recognition, and machine learning.

Ashish Ghosh (Senior Member, IEEE) is a Professor with the Machine Intelligence Unit, Indian Statistical Institute. He has authored or coauthred around 250 research papers in internationally reputed journals and refereed conferences, and has edited eight books. His research interests include pattern recognition, machine learning, data mining, image and video analysis, soft computing, neural networks, evolutionary computation, and bioinformatics. He was the recipient of the the Young Scientists Award from the Indian Science Congress Association in 1992 and the Indian National Science Academy in 1995. He is acting as a Member of the Editorial boards of various international journals.