University of Kent

Final Year Research Project

A Novel Variation of Classifier Chain for Multi-label Classification

Author: Maxwell Sime
mjs84@kent.ac.uk

Supervisor: Alex Freitas
AA.Freitas@kent.ac.uk

Word count: 7350

i. Abstract

Creation of a novel variant of the existing multi-label classification algorithm Classifier Chain (CC) originally proposed by J.Read in 2009. The Separated Classifier Chain (SCC) algorithm and it's variant Separated Classifier Chain - Binary Relevance (SCC-BR) uses a probability-based chain ordering function and separates CC into two classifiers which use classification confidences at runtime to choose the most probable value for the label. SCC and its variant SCC-BR have proved to have comparable results to CC and Binary Relevance (BR) on 7 multi-label datasets, out-performing BR in multiple performance metrics. Using research and experiments I will investigate the advantages and disadvantages of SCC and SCC-BR compared to CC and BR. Using SCC I will be able to investigate the significance of label-to-label relationships in multi-label problems and how manipulating these relationships could benefit algorithm performance. Later in the report, I discuss potential improvements that could be made to improve performance and continue this line of research.

The algorithm was programmed using Java, implementing the multi-label learning tool MEKA's API and experimentation program.

ii. Acknowledgments

This project was completed during the Corona-virus 2020/21 pandemic. I thank my supervisor Alex Freitas for being very understanding and empathetic of my degrading mental health state during this final year of my degree. Lockdown has been very difficult for us all and added new challenges to the completion of this project.

I also thank the University of Kent for being very understanding of this situation and giving students a lot of assistance during this difficult time.

iii. Contents

1. Introduction

Document Classification is a popular task applying machine learning techniques to classify documents into a specific category. It has been widely studied by the database, data mining, and information retrieval communities with it now being used for many challenging real-world applications[1-6]. An important subsection of Document Classification, and the focus of this project, is “Multi-Label Classification” a more difficult machine learning problem that focuses on categorizing documents into multiple categories, not just one. With the added difficulty of this task many researchers attempted to adapt Single-Label Classification algorithms to Multi-Label problems, one such adaption was Binary Relevance(BR) which was further improved upon into the main focus of this project, the Classifier Chain(CC) algorithm[7].

The objective of this project is to create a variant of the CC algorithm, dubbed “Separated Classifier Chain”, with the intention of it producing comparable results to the original CC and BR. In this report, I will use research and experiments to discuss the advantages and disadvantages of my CC variant, while discussing further possible improvements.

1.1. Vocabulary

Class = The group to which a document belongs to.
Classifier = An algorithm that learns to predict the class of a document.
Dataset = Collection of documents processed to be read by a classifier.
Training set = A portion of the dataset used to train the classifier.
Testing set = The remaining portion of the dataset used to test the classifier with unique instances.
Instance = A document inside the dataset.
Label = The classes related to a document.
Feature = An attribute of the document, for example, a word inside a text document.
CC = Classifier Chain
BR = Binary Relevance
SCC = Separated Classifier Chain
SCC-BR = Separated Classifier Chain, Binary Relevance version.
WEKA = The name of the University of Waikato’s open-source Java machine learning tool used by many researchers and professionals.
MEKA = A modification on the original WEKA architecture to enable it for multi-label machine learning purposes.

2. Background

In this section I will discuss the research relevant to my project, this will include information on text classification, the j48 classifier, and relevant multi-label classification techniques.

2.1. Text Classification

A subsection of Document Classification, Text Classification is the process of classifying text documents into specific data-related categories (labels). By implementing Natural Language Processing, Machine Learning, and Data Mining on a portion of a dataset, classifiers can learn to predict the values of labels for an instance with an accuracy of over 90% on certain datasets[8]. Classification has only risen in relevance because of the recent interest in machine learning, text mining, and the vast amount of available documents thanks to the internet.

Some applications for text classification include content tagging for websites, marketing, sentiment analysis, document summarization, and automatic document categorization (for use in businesses and academia). Some specific examples of applications include: university applications[1], business analytics[2], email categorization[3], research categorization[4], ICU mortality prediction[5] and cancer case management[6].

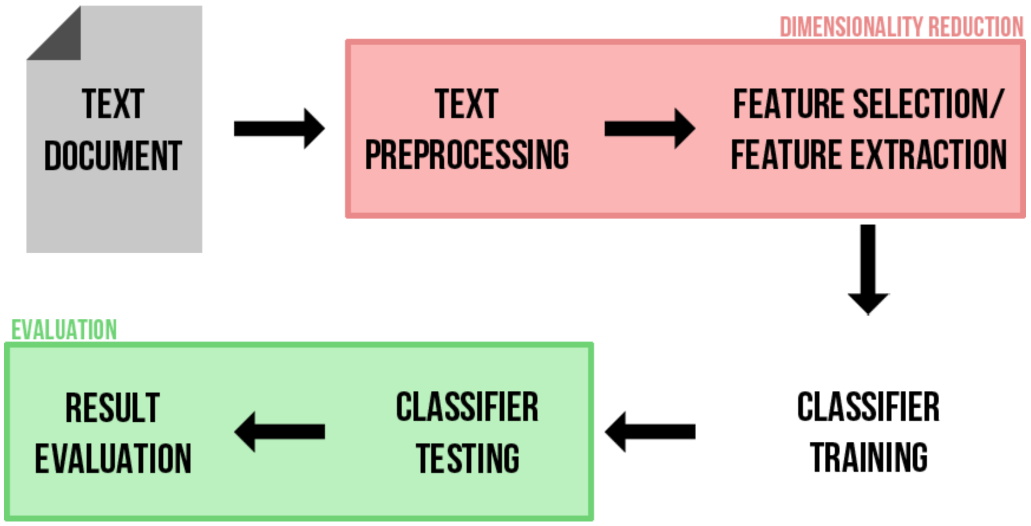

Fig 1. Text Classification process

Text classification comes naturally to a human whereas having a computer understand unstructured natural language documents is an impossible task. This is why documents have to be heavily formatted before text classification can be applied to them. Alongside formatting being required to make documents comprehensible for a computer it is used to reduce the “dimensionality” of the feature space in a document, this is because classification is very computationally expensive so reducing the complexity of the document lowers processing time and improves classification accuracy[9].

First, documents are preprocessed to make them comprehensible for a computer. This pertains to separating sentences into individual features in a process called tokenization[10] then applying algorithms on them to remove and reduce the insignificant features. At this stage, unnecessary noise is also removed.

Second, documents are parsed through algorithms to further reduce dimensionality. This includes removing more insignificant features, on the basis of mathematical calculations, selecting only a subset of labels via feature selection, or generating new features to remove others via feature extraction.

2.1.1. Multi-label Classification

The earliest stages of Text Classification used datasets with instances being assigned a single binary choice, the instance was part of a single class in the dataset. This project focuses on multi-label classification where a single instance can have any amount of labels associated with it. By nature of allowing instances to have multiple label associations, the classification problem becomes exponentially harder.

To combat this more challenging classification problem binary classifiers were adapted for multi-label datasets[11,12]. While certain feature selection and extraction techniques were directly applicable to multi-label datasets[13] others required modifications to be applicable[14].

2.2. Decision Trees

The two multi-label classification techniques I implement as part of my algorithm still require a base-classifier to do classification, both Binary Relevance and Classifier Chain can implement any single-label classifier to process data. For this project, I will be generating all of my results using the decision tree classifier J48[15], an open-source Java implementation of the C4.5 classifier[16]. It is important to use the same classifier for each classification task otherwise the results are not comparable.

Decision trees are decision-support tools that implement tree-inspired models to map a decision process, they are often implemented as algorithms in machine learning. Decision trees map “test cases” onto decision nodes which have a “branch” for each possible answer, these branches link to other nodes which include further tests. The algorithm tests the input to answer each node until it has reached a “leaf node” which would be the output of the algorithm.

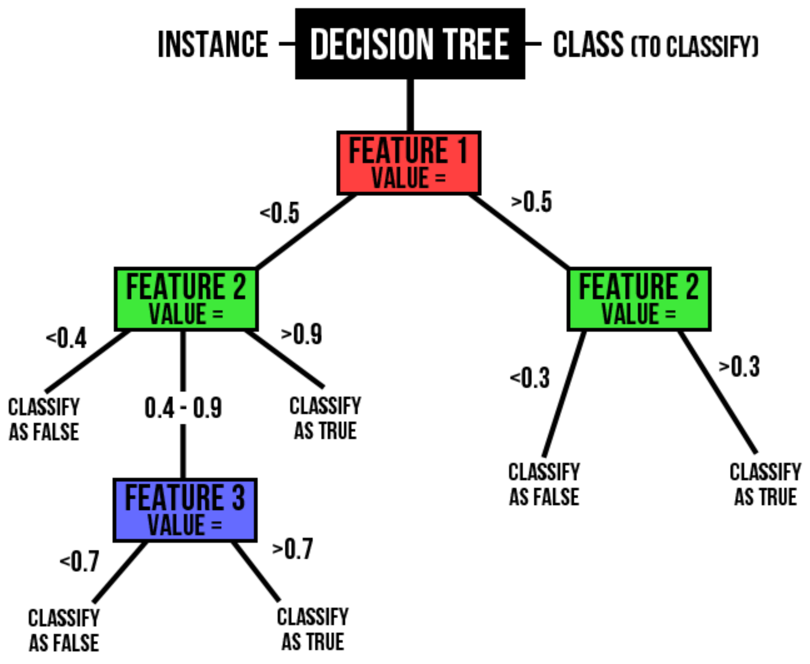

Fig 2. Decision Tree Example

Figure 2 shows a basic example of a decision tree algorithm. Given an instance from the dataset and a class to classify it will test the instance's features with each decision node until it reaches a conclusion on which value to output the class as.

2.2.1. J48 Classifier

As discussed above, the J48 classifier is the WEKA java implementation of the C4.5 decision tree algorithm[15]. The C4.5 algorithm was developed by J. Quinlan in 1993 as a modification of his earlier ID3 algorithm from 1986[17]. Described by the WEKA developer Eibe Frank as a:

"landmark decision tree program that is probably the machine learning workhorse most widely used in practice to date"[18]

```
Let $T$ be training set
FOR each feature $f$
    Calculate normalised information gain ratio of $f$
    Save the feature $f$ with the highest gain ratio as $f^*$
FOR each possible value $n$ for $f^*$ inside $T$
    Split $T$ into sublists of $T$ where all instances of $f^* = n$
CREATE a decision node for $f^*$

REPEAT previous FOR using $T_1, \ldots, T_n$
    Add the feature $f$ from $T_1, \ldots, T_n$ with the highest information gain ratios as child nodes
    Split $T_1, \ldots, T_n$ again using the same method as before
Continue until the tree is complete
```

Using a training set (a portion of the dataset), each C4.5 decision node is associated with 1 or more features. The algorithm starts as just the root, associated with the entire training set. The first node is generated by calculating the most relevant feature *f*, the training set is then split into sublists of instances that share the same value for *f*, the generated node holds a test case for *f*. A child node is generated for each sublist recursively by searching both sublists for their most relevant feature and doing the same process as before, with the most relevant feature being used to separate the training set sublists again. Once chosen to represent a node binary features are never used again however categorical/nominal features are sometimes used in multiple nodes.

To calculate feature relevancy C4.5 uses "Normalised Information Gain"[19] discussed below. Let: *T* be the training set; $T_i,\ldots,T_l$ be the subsets of the training set; *l* be the number of values of the class attribute being used; *freq*(C_j, *T*) be the number of examples of class labels C_j in *T*; |*T*| be the cardinality (number of instances) of *T*.

$$info(T) = -\sum_{j=1}^{l} \frac{freq(C_j,T)}{|T|} \times log_2\left(\frac{freq(C_j,T)}{|T|}\right)$$

info(T) calculates the entropy of *C* in *T* and is used in the gain equation:

$$gain(T) = info(T) - \sum_{i=1}^{l} \frac{|T_i|}{|T|} \times info(T_i)$$

gain(T) calculates the difference between the entropy (*info(T)*) and conditional entropy of *C* in *T*.

$$split(T) = -\sum_{i=1}^{l} \frac{|T_i|}{|T|} \times log_2\frac{|T_i|}{|T|}$$

split(T) calculates the entropy of *T*. It is used alongside *gain* to get the *gain ratio*:

$$gainratio(T) = \frac{gain(T)}{split(T)}$$

gainratio(T) normalises information gain by calculating the gain ratio, used to determine the most relevant feature in a training set. When a feature has been chosen for a previous node then the gain ratio returns with 0.

2.2.2. J48 Example

```
Given Training Set = $T$
$T$ list of binary features = [$f_1, f_2, f_3$]                 (All either 0 or 1)
Build Tree:
        Calculate normalised information gain ratio for $f_1$
                $f_1$ ratio = 0.9
        Calculate normalised information gain ratio for $f_2$
                $f_2$ ratio = 1.5
        Calculate normalised information gain ratio for $f_3$
            $f_3$ ratio = 0.7
Build decision node for $f_2$
Split $T$ by $f_2$ into $T_1$ and $T_2$
        $T_1$ = All instances of $T$ where $f_2 = 0$
        $T_2$ = All instances of $T$ where $f_2 = 1$

For sublist $T_1$:
        Calculate normalised information gain ratio for $f_1$
            $f_1$ ratio = 0.9
        Calculate normalised information gain ratio for $f_2$
            $f_2$ ratio = 0                              (Already used)
        Calculate normalised information gain ratio for $f_3$
            $f_3$ ratio = 0.7
Build decision node $f_1$ underneath $f_2$ node for when $f_2 = 0$
Split $T_1$ by $f_1$ into $T_{1a}$ and $T_{2a}$
        $T_{1a}$ = All instances of $T$ where $f_1 = 0$
        $T_{2a}$ = All instances of $T$ where $f_1 = 1$

For sublist $T_2$:
        Calculate normalised information gain ratio for $f_1$
            $f_1$ ratio = 0.6
        Calculate normalised information gain ratio for $f_2$
            $f_2$ ratio = 0                              (Already used)
        Calculate normalised information gain ratio for $f_3$
            $f_3$ ratio = 0.9
Build decision node $f_3$ underneath $f_2$ node for when $f_2 = 1$
Split $T_1$ by $f_3$ into $T_{1b}$ and $T_{2b}$
        $T_{1b}$ = All instances of $T$ where $f_3 = 0$
        $T_{2b}$ = All instances of $T$ where $f_3 = 1$

For sublist $T_{1a}$ and $T_{2a}$ only $f_3$ returns a ratio
Build 2 decision nodes for $f_3$ underneath decision node $f_1$

For sublist $T_{1b}$ and $T_{2b}$ only $f_1$ returns a ratio
Build 2 decision nodes for $f_1$ underneath decision node $f_3$
```

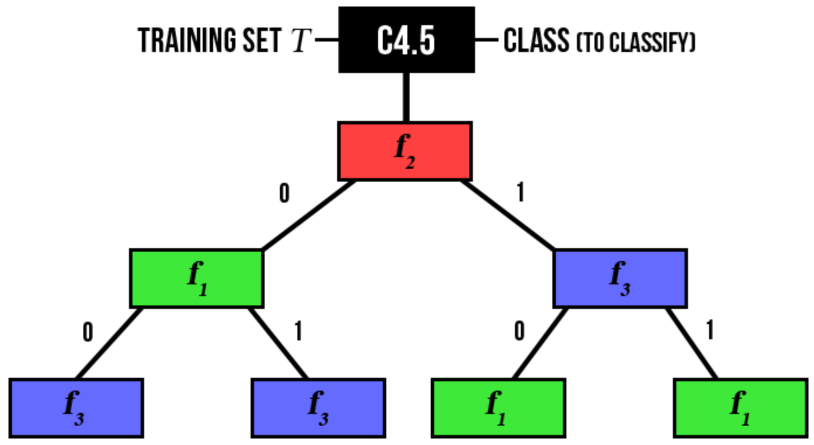

Fig 2. Partially-constructed C4.5 Decision Tree example without leaf-nodes

2.3. Relevant Multi-Label Classification Techniques

The two most relevant classification techniques for this project are Binary Relevance (BR) and Classifier Chain (CC).

2.3.1. Binary Relevance

Another method of approaching the multi-label classification problem is to use a "Problem Transformation" approach which consists of treating a multi-label dataset as multiple single-label classification problems[20]. A well-researched example of this is Binary Relevance (BR) which applies a base-classifier on each label separately.

The advantage of using BR is its simplicity, allowing any binary classifier to be applied to any multi-label classification problem. Although an intuitive solution BR has one clear major disadvantage to more suitable algorithms, BR doesn't exploit any correlation between labels[21]. Many modifications on BR have been proposed to exploit correlations between labels[22,23], including the focus of this project Classifier Chains(CC)[7].

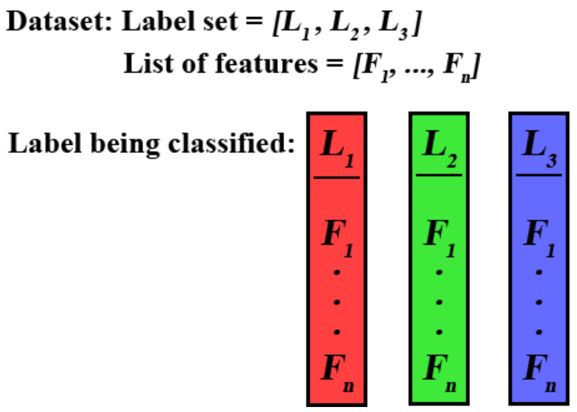
Dataset: Label set = $[L_1, L_2, L_3]$
List of features = $[F_1, ..., F_n]$
Label being classified:
L_1
F_1
F_n
L_2
F_1
F_n
L_3
F_1
F_n

Fig 3. Binary Relevance diagram

2.3.2. Classifier Chain

Originally proposed in 2009 by Jesse Read et al.[7] CC is a modification of BR, removing its biggest disadvantage by exploiting the correlation between labels. CC applies a random "chain" of labels which it uses as the order of classification, once a label is classified the predicted value is then appended onto the instance to improve the accuracy of the next classification. Compared to BR and other state-of-the-art algorithms (of the time) CC performs much stronger in Accuracy[7].

CC is a clear improvement on BR however it isn't fully taking advantage of label correlations. Labels typically have different degrees of importance[24,25] however CC does not take advantage of this because it generates a random chain. Alongside this, some label correlations can be negative and CC's method of bundling all available labels into the chain can reduce classification accuracy for some labels.

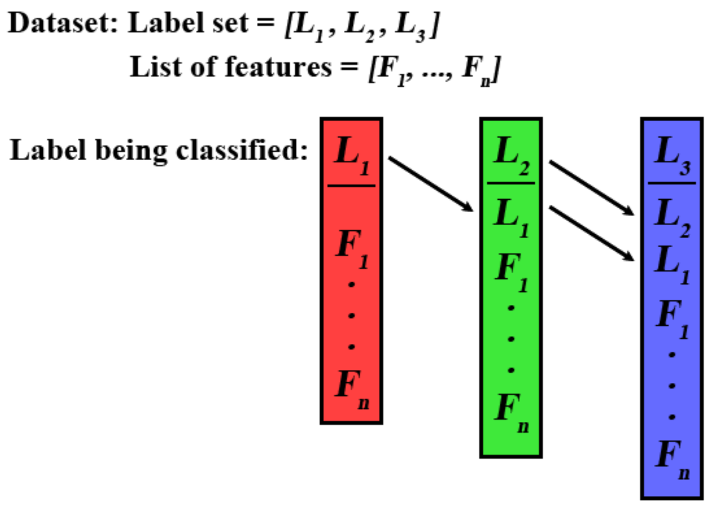
Dataset: Label set = $[L_1, L_2, L_3]$
List of features = $[F_1, ..., F_n]$
Label being classified:
L_1
F_1 . . . F_n
L_2
L_1 F_1 . . . F_n
L_3
L_2 L_1 F_1 . . . F_n

Fig 4. Classifier Chain diagram

2.3.3. Modifications on Classifier Chain

Modifications of CC applying chain generation techniques to further take advantage of label correlations have shown to improve performance[26,27,28]. A paper from K. Dembzyński et al.[26] generated chains by using the product rule of probability to calculate the effectiveness of each label combination. This method dubbed "Probabilistic Classifier Chain''(PCC) and its ensemble variant proved to outperform CC and even Ensemble Classifier Chain[29] however was only tested on 2 simple artificial datasets, further testing on public datasets would be necessary to further validate the algorithm. Similar to PCC, "Entropy-based Classifier Chains" (EbCC) applies equations to generate the optimal order. Proposed by X. Jun et al.[27] in 2019 there are 4 variants of EbCC each of which consistently outperformed CC in all metrics by a large amount, only 3/4 of the variants outperformed ECC however they all offer far lower computational complexity.

A paper from E. C. Gonçalves, et al. in 2015[28] implemented a genetic algorithm to calculate the most optimal chain order, the results showed both variants of the “Genetic Algorithm Classifier Chain”(GACC) performed higher than CC in all tested performance metrics. Alongside removing the disadvantage of a random chain, the variation of their proposed algorithm “Genetic Algorithm-Partial Classifier Chain”(GA-PARTCC) removed the disadvantage of bunching all dataset labels into a single chain by generating partial chains. This variation of GACC allowed the algorithm to avoid the negative label correlations which reduce the classification accuracy of the original CC, the GA-PARTCC algorithm out-performed CC and GACC on multiple datasets.

J. Read et al. developed “Ensemble Classifier Chain” (ECC) first in 2011[29], a variation of CC which builds CCs for each possible label order and uses a weighting scheme to calculate the correct classification. Although having a much higher computational complexity than many other classifiers ECC’s performance is very strong, outperforming EbCC, GACC, and GA-PARTCC on multiple datasets. Tests have also shown it to be comparative to other state-of-the-art ensemble classifiers[30] and to be effective in multiple applications[31].

3. Methodology

3.1. Separated Classifier Chain

Separated Classifier Chain(SCC) learns 2 classifier chains: the first classifier is a base Classifier Chain, it uses features and any previously classified labels to classify the instance; the second classifier doesn't use any features and only contains the previously used labels to classify an instance. The order of the labels (the chain), is generated like so:

```
FOR each class label $l$
    FOR each class label $i$
        FOR each class label $j \neq i$
            Learn classifier to predict label $j$, using only label $i$ and labels added to the chain
        Predictive power of label $i$ = average of predictive performance of classifiers in above FOR $j$ loop
    Select label $i$ with the highest predictive power, set as the next label in the chain
```

Cycling through the list of labels, the algorithm trains a classifier on each label using only a single label as a predictive feature. The average confidence of the classifiers, trained on 1 label, is equal to the predictive performance of the label. After classifiers have been trained for each label (using each other label) the label with the highest predictive performance is chosen as the first member of the chain. Afterward, cycling through the list of remaining labels, the algorithm trains a classifier on each label using the current member(s) of the chain and 1 of the remaining labels. Once again, the average confidence of the classifiers is equal to the predictive performance of the chain plus the single label. The label which produces the highest predictive performance is then appended onto the end of the chain. This cycle continues until there are no longer remaining labels and the chain is complete.

3.1.1. SCC Example

```
Given dataset labels = $\mathbf{L}^1$, $\mathbf{L}^2$, $\mathbf{L}^3$
            Train a classifier to classify label $\mathbf{L}^2$ using only label $\mathbf{L}^1$
            Train a classifier to classify label $\mathbf{L}^3$ using only label $\mathbf{L}^1$
    Calculated predictive performance for label $\mathbf{L}^1$ = **0.9**
            Train a classifier to classify label $\mathbf{L}^1$ using only label $\mathbf{L}^2$
            Train a classifier to classify label $\mathbf{L}^3$ using only label $\mathbf{L}^2$
    Calculated predictive performance for label $\mathbf{L}^2$ = **0.4**
            Train a classifier to classify label $\mathbf{L}^1$ using only label $\mathbf{L}^3$
            Train a classifier to classify label $\mathbf{L}^2$ using only label $\mathbf{L}^3$
    Calculated predictive performance for label $L^3$ = 0.5
First member of the chain = $L^1$                    (highest predictive performance)

        Train a classifier to classify label $L^3$ using label $L^1$ and $L^2$
    Calculated predictive performance for label $L^2$ = 0.2
        Train a classifier to classify label $L^2$ using label $L^1$ and $L^3$
    Calculated predictive performance for label $L^3$ = 0.7
Second member of the chain = $L^3$                   (highest predictive performance)

    Only 1 remaining label ($L^2$)
Full order of chain = $L^1$, $L^3$, $L^2$

Train a classifier to classify label $L^1$ using only features
Train a classifier to classify label $L^3$ using only features
    Train a classifier to classify label $L^3$ using only label $L^1$
Train a classifier to classify label $L^2$ using only features
    Train a classifier to classify label $L^2$ using label $L^1$, $L^3$
```

3.1.2. Classifying Test Instances

The algorithm is given a new instance from the test portion of the dataset. Classifying the first label in the chain: the output is given from classifier 1 (the base Classifier Chain) trained for the first label, since the first label doesn't have a second classifier no comparison is available. Classifying the second label in the chain: the output is from whichever classifier outputs the highest confidence, classifier 1 which is the standard CC (trained on the original features and previous class labels), or classifier 2 which is a separated classifier chain trained on only the labels (not using the original features). The process for the second label in the chain is continued for the remaining members in the chain until each label in the instance is classified.

3.1.3. Advantages and Disadvantages

The second, label-only classifier, enables labels to have a stronger influence on the classification than CC. The extra influence from the labels could have a positive or negative effect on the algorithm's performance based on the dataset being used.

As shown by multiple papers[24,25] labels have varying correlations between each other and can often hold negative correlations. Applying an inadequate ordered chain to CC can actually decrease performance[28]. The chain generation technique applied by SCC does not take negative label-to-label correlations into consideration and doesn't attempt to explore more than a single order meaning it is likely to build a sub-optimal chain.

Without the algorithm comparing chain performance, it won't be able to produce the most optimal chain order.

SCC shares the same disadvantage with CC in which all labels are added to the chain, this allows negative label-to-label correlations to impact performance. As shown by Gonçalves, E et al.[28] generating multiple "partial chains" containing only positively correlated labels greatly increases performance compared to single-chain methods.

The large amount of classifiers required to generate a chain for SCC makes it very computationally expensive, exponentially more than CC. Other CC variants apply mathematic-based chain generation techniques which are much less computationally expensive[26,27] and more likely to generate an optimal chain.

3.2. Separated Classifier Chain - Binary Relevance

Separated Classifier Chain - Binary Relevance (SCC-BR) is a variant of SCC in which the first classifier CC is replaced with BR, the rest of the classifier is completely identical. I created this variant to discover if SCC-BR's chain generation was a direct improvement on BR. Its results in comparison to SCC have allowed for further discoveries relating to SCC and the effectiveness of the SCC chain generation technique.

3.2.1. Advantages and Disadvantages

SCC-BR enables the labels to have a greater influence on classification compared to BR however SCC-BR has the same disadvantageous chain generation method that SCC has which frequently generates sub-optimal chains. Alongside this SCC-BR is also as computationally expensive as SCC.

4. Results

4.1. MEKA

To generate results I used the open-source Java multi-label classification tool MEKA[32], based on the machine learning toolkit WEKA and developed by Jesse Read, Peter Reutemann, and Joerg Wicker. I used the MEKA implementation of CC, BR, and the WEKA classifier J4.8[15] (Java implementation of C4.5).

4.2. Performance Metrics

To evaluate the performance of my algorithm I used 4 specific performance metrics, Accuracy, Exact Match, F1 Measure, and Hamming Loss.

4.2.1. Accuracy

Applying the *Jaccard Index*[33] for multi-label classification tasks, the accuracy for each instance is calculated by dividing the number of correctly predicted labels by the sum of predicted and actual labels for that instance[34]. *Accuracy* outputs the average accuracy of each predicted instance.

$$Accuracy = \frac{1}{x} \sum_{i=1}^{x} \frac{|A_i \cap B_i|}{|A_i \cup B_i|}$$

Let x be the number of instances in the dataset; A_i be the instance i's labels from the dataset; B_i the instance i's labels predicted by the classifier.

4.2.2. Exact Match

The average amount of 100% correctly predicted instances.

$$Exact\ Match = \frac{1}{x} \sum_{i=1}^{x} I(A_i = B_i)$$

Where I is the function that returns 1 if $A_i = B_i$ and 0 otherwise.

4.2.3. F1 Macro

Calculated using 2 other metrics: *precision* and *recall*, F1 Macro is one of the strongest performance metrics for classification tasks.

Precision is the proportion of correctly predicted labels over the total of predicted labels, averaged.

$$Precision = \frac{1}{x} \sum_{i=1}^{x} \frac{|A_i \cap B_i|}{|B_i|}$$

Recall is the proportion of correctly predicted labels over the total amount of actual labels, averaged.

$$Recall = \frac{1}{x} \sum_{i=1}^{x} \frac{|A_i \cap B_i|}{|A_i|}$$

F1 Macro calculates using the precision and recall of a predicted instance and can be considered as the mean of precision and recall.

$$F1\ Macro = \frac{1}{x} \sum_{i=1}^{x} \frac{2|A_i \cap B_i|}{|A_i| + |B_i|}$$

4.2.4. Hamming Loss

Calculated as the fraction of wrong labels to the total number of labels[35].

$$Hamming\ Loss = \frac{1}{xl} \sum_{i=1}^{x} \sum_{j=1}^{l} A_{i,j} \oplus B_{i,j}$$

Let l be the number of labels in the dataset; $A_{i,j}$ be the instance i's label j; $B_{i,j}$ be the instance i's label j; $\oplus$ be XOR (Exclusive OR) operator, if $A_{i,j}$ and $B_{i,j}$ match it returns a 0 otherwise it returns a 1.

4.3. Experimentation

To generate results I ran CC, BR, SCC, and SCC-BR together on the same dataset in MEKA running each classifier 10 times, using 10-fold cross-validation. Running each classifier 10 times helps generate accurate statistics by averaging the performance of each run.

"K-fold cross-validation" is used to partition the dataset into multiple subsets, 1 is used for the "training set" to train the classifier and the remainder is used as "testing sets" to test the classifier to generate results.

4.3.1. Datasets

I tested each algorithm on 7 different text classification datasets, I planned on generating further results using the RCV1 and Bibtex datasets however the computational complexity was too high for my computer to handle. After 4 days of trying to generate results with both datasets, neither of them had finished a single classification task with BR (the lowest computationally complex classifier). All datasets used were downloaded from the University of Córdoba's Multi-Label Classification Dataset Repository[36].

For dataset *d*: ***I*** is the number of instances; ***L*** is the number of labels; ***A*** is the number of attributes; ***Card.*** is the cardinality, the average number of labels assigned to each instance; ***Div.*** is the diversity, the percentage of labelsets present in the dataset divided by the number of possible labelsets. The *Yelp* dataset has 64 unique combinations of labels (labelsets) associated to instances, the maximum possible amount is 2^L (64) thus *Yelp* has a diversity of 1. On the other hand *3s-bbc1000* has 15 unique labelsets out of the maximum amount of 64 thus *3s-bbc1000* has a diversity of 0.234375.

Dataset	**I**	**L**	**A**	**Card.**	**Div.**
3s-bbc1000[37]	352	6	1000	1.125	0.234
3s-guardian1000[37]	302	6	1000	1.126	0.219
3s-inter3000[37]	169	6	3000	1.142	0.172
3s-reuters1000[37]	294	6	1000	1.126	0.219
20NG[38]	19300	20	1006	1.029	0.003
Yelp[39]	10810	5	671	1.638	1.000
Enron[40]	1702	53	1001	3.378	0.442

4.3.2. Results

I have displayed my results in tabular form for each performance metric: Accuracy, Exact Match, F1 Macro, and Hamming Loss. The entry in bold is the best performing algorithm on that dataset, each entry has the "rank" (1-4) of the algorithm's performance on that dataset for that performance metric. I have calculated the mean rank of each algorithm on each dataset then wrote the average rank order for that performance metric below each table.

Accuracy				
Dataset	**BR**	**CC**	**SCC**	**SCC-BR**
3sources_bbc	0.182 (2)	**0.211** (1)	0.167 (4)	0.171 (3)
3sources_guardian	0.176 (3)	**0.219** (1)	0.173 (4)	0.179 (2)
3sources_inter	**0.143** (1)	0.141 (2)	0.126 (3)	0.123 (4)
3sources_reuters	0.18 (3)	**0.198** (1)	0.191 (2)	0.176 (4)
20NG	0.581 (4)	**0.641** (1)	0.632 (2)	0.632 (2)

Yelp	0.684 (2)	**0.689** (1)	0.51 (4)	0.512 (3)
Enron	0.441 (2)	**0.451** (1)	0.406 (4)	0.408 (3)
Average Rank	2.285	**1.142**	3.285	2.857

CC > BR > SCC-BR > SCC

In Accuracy, CC clearly outperforms all classifiers, only being outperformed by BR on a single dataset. BR on average outperforms both SCC and SCC-BR while SCC-BR outperforms SCC slightly. In the less diverse datasets (*3sources & 20NG*) SCC and SCC-BR are very comparable to BR, frequently out-performing it, however in the higher diversity datasets (*Yelp & Enron*), BR outclasses SCC and SCC-BR. SCC and SCC-BR perform their best on *20NG* the dataset with the lowest diversity while their worst on *Yelp* the dataset with the highest diversity because the chain generation technique used by SCC and SCC-BR has difficulty producing an optimal chain on diverse datasets. Interestingly the least diverse *3sources* dataset *3sources_inter* was the only dataset in which all classifier chain variants were outperformed by BR because it has 3000 attributes so the labels have much lower influence over classification so the label chains negatively impacted classification.

Exact Match				
Dataset	**BR**	**CC**	**SCC**	**SCC-BR**
3sources_bbc	0.088 (4)	**0.173** (1)	0.134 (2)	0.125 (3)
3sources_guardian	0.075 (4)	**0.176** (1)	0.137 (2)	0.126 (3)
3sources_inter	0.072 (3)	**0.084** (1)	0.076 (2)	0.065 (4)
3sources_reuters	0.083 (4)	**0.151** (1)	0.15 (2)	0.127 (3)
20NG	0.514 (4)	**0.591** (1)	0.587 (2)	0.586 (3)
Yelp	0.479 (2)	**0.502** (1)	0.342 (3)	0.342 (3)
Enron	0.172 (2)	**0.198** (1)	0.157 (3)	0.157 (3)
Average Rank	3.285	**1**	2.285	3.285

CC > SCC > SCC-BR, BR

In Exact Match, similar to Accuracy, CC is clearly the best performer. SCC shows great performance, almost matching CC in many datasets however suffers significantly once again in the higher diversity datasets. SCC-BR is a clear improvement over BR in the lower diversity datasets where the added label influence helps with classification, however once again struggles in the higher diversity datasets due to generating a suboptimal chain. The difference between the label influence of low diversity and high diversity datasets is clear here, BR performs very poorly compared to CC in *3sources* and *20NG*, however performs comparably in both *Yelp* and *Enron*.

F1 Macro				
Dataset	**BR**	**CC**	**SCC**	**SCC-BR**
3sources_bbc	0.217 (2)	**0.225** (1)	0.179 (4)	0.188 (3)
3sources_guardian	0.214 (2)	**0.233** (1)	0.186 (4)	0.199 (3)
3sources_inter	**0.171** (1)	0.162 (2)	0.145 (3)	0.145 (3)
3sources_reuters	**0.218** (1)	0.214 (2)	0.205 (3)	0.194 (4)
20NG	**0.166** (1)	0.141 (2)	0.13 (4)	0.131 (3)
Yelp	**0.709** (1)	0.706 (2)	0.518 (4)	0.52 (3)
Enron	**0.417** (1)	0.41 (2)	0.371 (3)	0.371 (3)
Average Rank	**1.285**	1.714	3.571	3.142

BR > CC > SCC-BR > SCC

As found with other comparisons of BR and CC[27,28], BR frequently outperforms CC in F1 Macro. Both SCC and SCC-BR performed very poorly in this metric, below BR and CC, because SCC and SCC-BR produce rather low Precision scores.

Hamming Loss				
Dataset	**BR**	**CC**	**SCC**	**SCC-BR**
3sources_bbc	0.308 (4)	0.284 (2)	**0.258** (1)	0.298 (3)
3sources_guardian	0.313 (4)	0.284 (2)	**0.265** (1)	0.296 (3)
3sources_inter	0.311 (3)	**0.305** (1)	0.307 (2)	0.314 (4)
3sources_reuters	0.315 (4)	0.285 (2)	**0.267** (1)	0.288 (3)
20NG	0.049 (4)	**0.04** (1)	**0.04** (1)	**0.04** (1)
Yelp	0.157 (2)	**0.153** (1)	0.209 (4)	0.208 (3)
Enron	0.091 (2)	**0.085** (1)	0.101 (4)	0.098 (3)
Average Rank	3.285	**1.428**	1.857	2.857

CC > SCC > SCC-BR > BR

While CC produced the highest average of results, SCC was not far behind. SCC performed the best on the lower diversity datasets however struggled again with the high diversity datasets. The large difference between BR and the classifier chains shows the importance of label correlations in minimizing Hamming Loss.

Overall SCC and SCC-BR performed comparatively to CC and BR. While SCC-BR performed worse than SCC in Exact Match and Hamming Loss, SCC-BR outperformed SCC in both Accuracy and F1 Macro. SCC enabling labels to have further influence negatively impacted Accuracy and F1 Macro however improved performance in Exact Match and Hamming Loss showing the relevancy label influence has on both metrics.

The disappointing performance on the datasets with high diversity of labelsets, *Yelp* and *Enron*, show clear problems with the chain generation techniques used by SCC and SCC-BR. Labels in higher diversity datasets naturally have a lower influence on classification and have harder to discern label-to-label correlations, shown by the smaller difference between the Accuracy, Exact Match, and Hamming Loss results for BR and CC on *Yelp* and *Enron* compared to the results on *3sources* and *20NG*.

The results show that for lower diversity datasets SCC performs better than BR because the label correlations are much easier to discern, with further improvements on SCC's chain generation it could easily outperform BR consistently in Accuracy and compete with CC. While SCC-BR outperformed SCC in Accuracy and F1, it is clearly an inferior algorithm producing much worse results in most situations due to its reliance on the chain generation technique, with improvements SCC-BR could definitely outperform BR because it already frequently outperforms it on lower diversity datasets.

5. Conclusion

In conclusion, SCC was relatively successful, this project proved the potential of a label-only classifier, and showed that with further improvements SCC could definitely outperform standard CC. That being said, building a chain using just the confidences of classifiers trained on only labels is an inefficient technique, other techniques are computationally cheaper and result in higher performance[26,27]. SCC generates suboptimal chains, does not evaluate more than a single full chain, cannot decipher more complex label-to-label relationships, doesn't consider the level of influence labels have on the dataset being processed, and is extremely computationally expensive. For these reasons, I cannot suggest to anyone to use SCC over CC or its other variants[26-29].

This paper shows that increasing the influence labels have on classification improves Hamming Loss and Exact Match for low diversity datasets and that increasing label influence without an adequate chain generation technique severely decreases performance in diverse datasets. My experiments show clearly that labelsets have a different level of influence for each dataset and not considering this when increasing label influence severely decreases performance. Diverse datasets and datasets with a large disparity between the number of labels and attributes have less significant label-to-label correlations and have much more complex label-to-label relationships making them harder to generate an adequate chain for.

6. Future Research

The most obvious and influential improvement to make to SCC would be improving the inadequate chain generation technique to find the optimal label order, as proven by Gonçalves, E et al.[28] a suboptimal chain order actually negatively impacts performance. The best choice would be to explore alternate chain orders and evaluate each individually to find the optimal solution, this is done using a genetic algorithm by Gonçalves, E et al.[28] however, this technique is computationally expensive since it still requires training the classifiers to evaluate each chain. Both PCC[26] and EbCC[27] offer mathematical methods of evaluating chains without having to train classifiers, PCC calculates the probability of a given chain while EbCC calculates the entropy of each label, similar to C4.5, and uses it to order the chain. Variants of EbCC are some of the highest performing CC variants to date, averaging a better performance than ECC[29] while boasting a much lower computational complexity EbCC still falls short, chaining all available labels together even those with negative label-to-label correlations. The GA-PARTCC algorithm[28] implements a partial chain technique which showed a vast improvement over the original GACC algorithm. Partial chains could be applied to EbCC for potentially even stronger performance albeit at a higher computational cost.

If increasing computational complexity is viable, ensemble techniques could be used to lower the impact of negative label-to-label correlations. ECC[29] creates CCs for each possible label order which results in higher computational costs than CC and EbCC but at a large performance improvement over CC. Since ECC requires the no. of labels^2 classifiers the computational costs are extremely high on datasets with a large number of labels. Ensemble techniques could be easily applied to SCC to completely replace the chain generation technique however would result in a much more computationally expensive algorithm (due to the coronavirus pandemic I have not been able to access a computer that could generate results for an ensemble SCC variant).

As discussed above, SCC and SCC-BR perform much stronger on lower-diversity datasets because the labels have a stronger influence over the classification whereas more diverse datasets have less influential labels meaning the second “label-only” chain negatively impacts performance. A method of improving performance on diverse datasets would be to calculate the influence labels have on the dataset, this calculation would have to consider the diversity of the dataset and the number of attributes and labels.

By making improvements on the SCC chain generation, computation complexity could be severely reduced. The optimal variant of SCC I could propose through my research would be a Partial-Chain Entropy-based SCC whereas an Ensemble SCC could have comparable results, both potentially higher performance than CC.

7. Bibliography

[1] Burström, J., 2019. *A Multimodal Approach to Autonomous Document Categorization Using Convolutional Neural Networks*. Undergraduate. Umeå University.
[2] Halibas, A., Shaffi, A. and Mohamed, M., 2018. ‘Application of text classification and clustering of Twitter data for business analytics’. *2018 Majan International Conference (MIC)*, IEEE, pp. 1-7.
[3] Brutlag, J. and Meek, C., 2000. ‘Challenges of the Email Domain for Text Classification’. *Proceedings of the Seventeenth International Conference on Machine Learning,* Microsoft.
[4] Rios, A. and Kavuluru, R., 2015. ‘Convolutional neural networks for biomedical text classification: application in indexing biomedical articles’. *BCB '15: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics,* BCB, pp. 258-267.
[5] Marafino, B., John Boscardin, W. and Adams Dudley, R., 2015. Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to ICU risk stratification from nursing notes. *Journal of Biomedical Informatics*, 54, pp.114-120.
[6] Garla, V., Taylor, C. and Brandt, C., 2013. Semi-supervised clinical text classification with Laplacian SVMs: An application to cancer case management. *Journal of Biomedical Informatics*, 46(5), pp.869-875.
[7] Read, J., Pfahringer, B., Holmes, G. and Frank, E., 2009. Classifier Chains for Multi-label Classification. *Machine Learning and Knowledge Discovery in Databases*, pp.254-269.
[8] Moyano, J., Gibaja, E., Cios, K. and Ventura, S., 2018. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, 44, pp.33-45.
[9] Uysal, A. and Gunal, S., 2014. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1), pp.104-112.
[10] Verma, T., Renu, R. and Gaur, D., 2014. Tokenization and Filtering Process in RapidMiner. *International Journal of Applied Information Systems*, 7(2), pp.16-18.
[11] Clare, A. and King, R., 2001. Knowledge Discovery in Multi-label Phenotype Data. *Principles of Data Mining and Knowledge Discovery*, pp.42-53.
[12] Allwein, E.L., Schapire, R.E. and Singer, Y., 2000. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, *1*(Dec), pp.113-141.
[13] Kohavi, R. and John, G.H., 1997. Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), pp.273-324.
[14] Park, C. and Lee, M., 2008. On applying linear discriminant analysis for multi-labeled problems. *Pattern Recognition Letters*, 29(7), pp.878-887.
[15] Weka.sourceforge.io. 2021. *J48 (weka-dev 3.9.5 API)*. [online] Available at: <https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/J48.html> [Accessed 1 April 2021].

[16] Quinlan, J., 1993. *C 4.5: programs for machine learning*. San Mateo, Calif.: Morgan Kaufmann.
[17] Quinlan, J., 1986. Induction of decision trees. *Machine Learning*, 1(1), pp.81-106.
[18] Witten, I. and Frank, E., 2005. *Data mining*. San Francisco, Calif.: Morgan Kaufmann, p.181.
[19] Ruggieri, S., 2002. Efficient C4.5 [classification algorithm]. *IEEE Transactions on Knowledge and Data Engineering*, 14(2), pp.438-444.
[20] Cherman, E., Monard, M. and Metz, J., 2011. Multi-label Problem Transformation Methods: a Case Study. *CLEI Electronic Journal*, 14(1).
[21] Zhang, M., Li, Y., Liu, X. and Geng, X., 2018. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2), pp.191-202.
[22] Montañes, E., Senge, R., Barranquero, J., Ramón Quevedo, J., José del Coz, J. and Hüllermeier, E., 2014. Dependent binary relevance models for multi-label classification. *Pattern Recognition*, 47(3), pp.1494-1508.
[23] Zhang, M. and Zhang, K., 2010. Multi-label learning by exploiting label dependency. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*,.
[24] Zhang, M., Zhang, Q., Fang, J., Li, Y. and Geng, X., 2019. Leveraging Implicit Relative Labeling-Importance Information for Effective Multi-Label Learning. *IEEE Transactions on Knowledge and Data Engineering*, pp.1-1.
[25] Chang, W., Dembczynski, K. and Hüllermeier, E., 2010. Graded Multilabel Classification: The Ordinal Case. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Haifa, Israel: ICML 2010, pp.223-230.
[26] Dembczynski, K., Cheng, W. and Hüllermeier, E., 2010. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Haifa, Israel: DBLP.
[27] Jun, X., Lu, Y., Lei, Z. and Guolun, D., 2019. Conditional entropy based classifier chains for multi-label classification. *Neurocomputing*, 335, pp.185-194.
[28] Gonçalves, E., Plastino, A. and Freitas, A., 2015. Simpler is Better. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*,.
[29] Read, J., Pfahringer, B., Holmes, G. and Frank, E., 2011. Classifier chains for multi-label classification. *Machine learning*, *85*(3), p.333.
[30] Moyano, J., Gibaja, E., Cios, K. and Ventura, S., 2018. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, 44, pp.33-45.
[31] El Kafrawy, P., Mausad, A. and Esmail, H., 2015. Experimental comparison of methods for multi-label classification in different application domains. *International Journal of Computer Applications*, *114*(19), pp.1-9.
[32] Read, J., Reutemann, P., Pfahringer, B. and Holmes, G., 2021. *MEKA: A Multi-label/Multi-target Extension to WEKA*. [online] Jmlr.org. Available at: <http://jmlr.org/papers/v17/12-164.html> [Accessed 12 April 2021].
[33] Meka.sourceforge.net. 2021. *Metrics*. [online] Available at: <http://meka.sourceforge.net/api-1.7/meka/core/Metrics.html> [Accessed 14 April 2021].

[34] Godbole S., Sarawagi S. (2004) Discriminative Methods for Multi-labeled Classification. In: *Advances in Knowledge Discovery and Data Mining. PAKDD 2004. Lecture Notes in Computer Science, vol 3056*. Springer, Berlin, Heidelberg. pp. 22-30
[35] Modi, H. and Panchal, M., 2012. Experimental Comparison of Different Problem Transformation Methods for Multi-Label Classification using MEKA. *International Journal of Computer Applications*, 59(15), pp.10-15.
[36] Uco.es. 2021. *Multi-Label Classification Dataset Repository – Knowledge Discovery and Intelligent Systems – KDIS – University of Córdoba*. [online] Available at: <http://www.uco.es/kdis/mllresources/> [Accessed 14 April 2021].
[37] Greene, D. and Cunningham, P., 2009. A matrix factorization approach for integrating multiple data views. In: *ECMLPKDD'09: Proceedings of the 2009th European Conference on Machine Learning and Knowledge Discovery in Databases*. Berlin: Springer-Verlag, pp.423-438.
[38] Rennie, J., 2008. Home Page for 20 Newsgroups Data Set. [online] Qwone.com. Available at: <http://qwone.com/~jason/20Newsgroups/> [Accessed 14 April 2021].
[39] Sajnani, H., Saini, V., Kumar, K., Gabrielova, E., Choudary, P. and Lopes, C., 2018. *The Yelp dataset challenge - Multilabel Classification of Yelp reviews into relevant categories*. [online] Ics.uci.edu. Available at: <https://www.ics.uci.edu/~vpsaini/> [Accessed 14 April 2021].
[40] Read, J., Pfahringer, B. and Holmes, G., 2008. Multi-label Classification Using Ensembles of Pruned Sets. In: *ICDM’08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. United States: IEEE Computer Society, pp. 995-1000.