Research Article

Ephrem Admasu Yekun* and Abrahaley Teklay Haile

Student Performance Prediction with Optimum Multilabel Ensemble Model

<https://doi.org/10.1515/jisys-2021-0016>

Received Apr 27, 2020; accepted Dec 04, 2020

Abstract: One of the important measures of quality of education is the performance of students in academic settings. Nowadays, abundant data is stored in educational institutions about students which can help to discover insight on how students are learning and to improve their performance ahead of time using data mining techniques. In this paper, we developed a student performance prediction model that predicts the performance of high school students for the next semester for five courses. We modeled our prediction system as a multi-label classification task and used support vector machine (SVM), Random Forest (RF), K-nearest Neighbors (KNN), and Multi-layer perceptron (MLP) as base-classifiers to train our model. We further improved the performance of the prediction model using a state-of-the-art partitioning scheme to divide the label space into smaller spaces and used Label Powerset (LP) transformation method to transform each labelset into a multi-class classification task. The proposed model achieved better performance in terms of different evaluation metrics when compared to other multi-label learning tasks such as binary relevance and classifier chains.

Keywords: EDM, Student performance prediction, Ensemble model, Multi-label classification

2020 Mathematics Subject Classification: 62P15, 62P30, 68T10, 68T30

1 Introduction

The field of machine learning enjoys applications in a variety of disciplines such as image and speech recognition, product recommendation, traffic prediction, and fraud detection [1] to mention a few. In recent years, educational data mining (EDM) has been of great research interest due to the abundance of data about student's information mainly being stored in state databases as well as the increased use of instrumental educational software providing insight on how students learn [2]. The main objective of EDM is to understand and gain knowledge from these educational data using statistical, machine learning, and data mining algorithms and take corrective measures ahead of time to improve student's performance in the educational settings [3].

The EDM process follows the same procedure as other application areas in business, medicine, genetics etc. where raw data collected from educational systems is first preprocessed into useful information that could produce insight into the educational system and create awareness of the teaching-learning process [4]. Particularly, by analyzing the students' data accurate and efficient student performance prediction models can be designed and developed. Further, this can help teachers, school administrators, and legal guardians to assist failing students in improving their learning style, organize their resources, effectively manage time, and even address some hindering environmental or psychological factors that their students may face. It also encourages students to take remedial and appropriate actions ahead of time and focus on activities that require high priorities.

*Corresponding Author: Ephrem Admasu Yekun: School of Electrical and Computer Engineering, Ethiopian Institute of Technology – Mekelle, Ethiopia; Email: ephrem.admasu@mu.edu.et

Abrahaley Teklay Haile: School of Computing, Ethiopian Institute of Technology – Mekelle, Ethiopia

In this paper, we developed a multi-label ensemble model to predict high school students' performance based on five courses: English, Math, Physics, Chemistry, and Biology. The dataset for training and testing was collected from three public high schools located in Mekelle, Tigray, Ethiopia. To the best of our knowledge this is the first time a student prediction model was created in Ethiopia for high school students. The prediction model evaluates the result of each subject for the next semester as fail or pass, making each label a binary class. The task of prediction is first performed by partitioning the label space L (where $|L| = 5$ for the five courses), into smaller labelsets using a randomized partitioning algorithm called Randomized k labelset (RAkEL). Then the training data with each labelset is transformed into a single-label multi-class training set. Each of the single-label classification tasks are trained using a base-level classifier. The base-level classifiers we have considered in this work are Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), and Multi-layer Perceptron (MLP).

2 Literature Review

In recent years, educational data mining for student performance prediction has gained widespread popularity. Using different techniques and methods, EDM can mine important information regarding the performance of the student and the educational settings to which the students are exposed. Classification, regression, and association rules are commonly used methods in EDM: classification being the most widely implemented method. Several algorithms are used for classification including Decision tree, Artificial Neural Networks, Naive Bayes, K-Nearest Neighbor and Support Vector Machines [5].

Many works have been done for student performance prediction using EDM. Pandey and Pal [6] used Bayesian classification method to predict performance of students based on data of 600 students collected from colleges in Awadh University, Faizabad, India. They considered category, language and background qualification of students as input features to predict high and low performing students and take remedial actions for the low performing ones. With a sample of 300 students, Hijazi and Naqvi [7] predicted student performance using linear regression. In their work, they used attendance, hours spent studying, family income, mother's age and mother's education as attributes and showed that mother's education and student's family income are good indicators of student's academic performance.

Shovon M. et al. [8] used k-means to predict student performance by grouping them into "Good", "Medium", and "Low" categories. To help low-performing undergraduate university students cope with bright students, Raheela Asif et al [9] developed a student performance prediction system using Decision Tree, k-nearest algorithms, Rule Induction, Naïve Bayes and Artificial Neural Networks that takes only high-school, first year and second year student results without considering other factors such as demographic or socio-economic attributes. Their work shows that k-Nearest Neighbor and Naïve Bayes achieved the best results.

Havan Agrawal and Harshil Mavani [10] used a Neural Network model to predict performance of students mainly those with poor academic performance. They also identified three attributes student's: grade in secondary education, living location, and medium of teaching as the most impactful on students' performance. Paulo Cortez and Alice Silva [11] developed a student results prediction system for secondary education using Decision Trees, Random Forest, Neural Networks and Support Vector Machines modeled as binary, five-level classification, and regression tasks. They showed that first and/or second terms results has the most influence on students results followed by other factors such as number of absences, parent's job and education, and alcohol consumption.

Recent works have also been done using ensemble models. Mrinal Pandey and S. Taruna [12] compared five ensemble techniques i.e Adaboost, Bagging, Random Forest and Rotation Forest for predicting performance of four year engineering graduate program students under ten base classifiers. They found Rotation forest to have the best prediction performance compared to the rest. Ashwin Satya Narayana, Marinsz Nuckowski [13] used Decision Trees-J48, Naïve Bayes and Random Forest to improve prediction accuracy by removing noisy example in the student's data. They also used a combination of rule based techniques such as A priori, Filtered Associator, and Tertius to identify association rules that affect student outcomes. Natthakan Iam-On et al. [14]

presented a student dropout prediction model at Mae Fah Fuan University, Thailand using link-based cluster ensemble as a data transformation framework to improve prediction accuracy. Pooja Kumari et al. [15] used Bagging, Boosting, and Voting Algorithm ensemble methods on Decision Tree (ID3), Nave Bayes, K-Nearest Neighbor, Support vector machines to improve accuracy of student performance prediction. They also showed including student's behavioral (SB) features results in improving the accuracy of the prediction model.

All the works presented above are single-label classification or regression tasks with some that incorporate ensemble models. To the best of our knowledge, our paper is the first work to present student performance prediction modeled as a multi-label classification task.

3 Dataset

The dataset used in this work was gathered from three public high schools located in the city of Mekelle, Ethiopia. The process of data collection was divided into two separate tasks. First, basic information such as student name, ID, sex and their scores on five courses i.e. English, Mathematics, Physics, Biology and Chemistry, of three consecutive semesters was collected from the school administrators. Second, a questionnaire was distributed to all students that contains eight closed-ended questions such as students perception towards the importance of education, family's educational background, family average income, student's grade 10 GPA score and others. The dataset is freely available for researchers and practitioners working in the area of educational machine learning [16]. Table 1 summarizes all the variables of dataset.

Table 1 depicts the features of the dataset which can be numerical or categorical. Numerical features represent real numbers whereas categorical features are further divided into two: nominal and ordinal. Ordinal features are categorical values that can be ordered or sorted whereas in nominal features order is not inherent. Hence, family income, family education background, grade 10 scores are ordinal features and gender, student's perception on the quality of education, legal guardians, tutorial, family occupation, and student's perception

Table 1: Dataset features with possible values

Feature	Type	Possible values
Gender	Categorical (nominal)	Male or Female
Age	Numerical	Range: [1, 100]
Courses	Numerical	[0, 100]
Quality of education	Categorical (nominal)	{Excellent, Very good, Good, Satisfactory, Bad, Very bad}
Legal guardians	Categorical (nominal)	{mother and father, father only, mother only, siblings, other, live alone}
Family income	Categorical (ordinal)	{<5000, 5000-10000, 10000-20000, >20000}
Family educational background	Categorical (ordinal)	{diploma, degree, masters, PhD, high school, high school dropout, no education}
Tutorial	Categorical (nominal)	{Yes, No}
Grade 10 GPA	Categorical (ordinal)	{2-2.5, 2.5-3, 3-3.5, 3.5-4}
Parent occupation	Categorical (nominal)	{Civil servant, Artisan, Trading/merchant, Military}
Student's perception towards education	Categorical (nominal)	{Yes, No}

on the importance of education are nominal features. After preparing the dataset, the next step is to separate the input features and the target (output) values. In this study, we are building a machine learning model that predicts the performance by predicting a student's score in the next semester. We have collected scores of five courses within three consecutive semesters. Then, we used the scores of the first two semesters as input features, along with the other features, to predict the results of the third semester. Since we are building a classification model, we discretized the scores into two classes, 0 and 1, using two interval values greater than or equal to 50 or less than 50, respectively. After preprocessing, the total size of the dataset was 714.

4 Proposed Work

4.1 Objective

Let $\mathcal{X}$ be an example space which consists of tuples of input values, discrete and continuous, such that $\forall x_i \in \mathcal{X}, x_i = (x_{i_1}, x_{i_2}, \dots, x_{i_m})$ where m is the number of features and let $\mathcal{L}$ be a label space such that $\mathcal{L} = \{\lambda_1, \lambda_2, \dots, \lambda_L\}$ which is a tuple of L discrete variables of either 0 or 1. L is the number of labels in the dataset. Our training set, D_{train} , can be represented as a pair of tuples from the example space $\mathcal{X}$ and label

Figure 1: Scheme of the proposed system

space $\mathcal{L}$ where $D_{train} = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{L}, 1 \leq i \leq n\}$ and n is the number of examples in the training set i.e. $n = |D_{train}|$. The goal of multi-label learning model is to find a function $h : \mathcal{X} \rightarrow 2^{\mathcal{L}}$ that maximizes some predictive accuracy or minimizes some loss measure.

After training and validation our model predicts some output $y_i \in \mathbb{R}^{1 \times L}$ given a sample input $x_i \in \mathbb{R}^{1 \times m}$ for some student $student_i$. More precisely, our model predicts a student's results for the next semester in terms of one of the binary classes i.e. Fail or Pass (encoded as 0 and 1, respectively).

4.2 System Architecture

Figure 1 shows the overall architecture of our prediction system. First, the dataset was preprocessed using different techniques including data cleansing, scaling, feature selection. Then, a space partition algorithm, known as RAKEL is used to partition the labels into smaller label spaces. Each of the partitioned data is then transformed into single-label multi-class classification task using a transformation method known as Label Powerset (LP). The transformed multi-class data set of each partition was fed into our learning algorithm to train our model. We train our model using different learning algorithms such as SVM, random forest classifier, K-nearest neighbor, and feed forward neural networks.

After training, the trained model was tested using the testing set which also went through the same partitioning and transformation procedures. The output of each label was predicted using majority voting rule since one label can exist in more than one partition. The majority voting rule is an ensemble method which takes the majority value to predict the output of a given label. Finally, the performance of the algorithm was evaluated using different evaluation measures. A detailed discussion of each component of the system is presented in the following sections.

4.3 Preprocessing and encoding

Preprocessing and features selection are important steps in most machine learning models. The dataset contains a total of 20 features/columns; 11 numerical features out of which 10 are scores of five courses from previous two semesters and one feature is age of a student.

4.3.1 Data cleansing:

Some of the fields in our dataset are filled with inconsistent data, outliers, and missing values. Since the percentage of these fields is very small, we replaced them with the mean of the feature input for numerical features and with the most frequent category in the case of categorical features.

4.3.2 Scaling:

The numerical features are measured on different scales (age and course score, for example). Therefore, it is reasonable to use some normalization technique for the prediction model to work properly. We normalized the numerical data by transforming it into a unit scale (mean=0, variance=1), a technique called standardization.

4.3.3 Label encoding and mapping:

From the 9 categorical features 3 are ordinal and 6 are nominal as shown in Table 1. Since ordinal features have order among their values, we mapped each ordinal feature f_i with n unique values into a set of integers $i \in \{1, \dots, n\}$ where each feature value is assigned a number based on its order in the feature. For the nominal

features, since there is no inherent order, we used label binarizer which assigns a unique binary value to each label in the feature. For a feature with $\{c_1, \dots, c_n\}$ unique categories, the label binarizer generates n new features of binary values all filled with 0 except the position at which the category belongs in the set of unique features. After implementing feature encoding in our dataset, we will have a total of 26 features due to label binarizer.

4.4 Feature Selection

Not all features in a dataset are useful since some are redundant or irrelevant. Using the right feature selection algorithm, we can remove all the redundant features and obtain the best ones for our learning algorithm. This again improves the storage requirement and reduces the running time of the learning algorithm; and sometimes improves the predictive performance of the classifier.

Several methods are known for feature selection [17], [18], [19], [20]. The choice of the best feature selection algorithm depends on the dataset and the model used for training. Our dataset contains numerical and categorical features. For each we used one type of feature selection technique: Pearson's correlation for the numerical and Chi-squared for the categorical. These methods are selected because they either provided the best results among other candidate feature selection algorithms or have equal predictive performance but chosen for their simplicity and efficiency.

4.4.1 Pearson correlation:

One of the most common ways to select features is to find the correlation between the numerical input features and the output values using Pearson's correlation coefficient r given as:

$$r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}} \quad (1)$$

where X and Y are random variables and $\bar{X}$ and $\bar{Y}$ mean of X and Y respectively. The value of r is always between -1 and 1. The higher the magnitude of r , is a good indicator that one of the two variables can be used as a feature input to the other variable [17]. We selected 8 features with the highest Pearson's correlations.

4.4.2 Chi-Square

Another method for feature selection between categorical values is the Chi-Squared test which measures how much two features are dependent to each other. Given two variables, Chi-Square calculates the squared difference between observed (O) and expected (E) frequencies divided by the expected frequency for all the cells given as [17].

$$X^2 = \sum \frac{(O - E)^2}{E} \quad (2)$$

Using the Chi-Square with *null* hypothesis, 10 features out of 15 categorical features are selected. This reduces the original 26 features into 18 input features. Hence, the final preprocessed dataset becomes a 2D with input of size 714×18, 8 of which are numerical values and the remaining 10 one-hot encoded categorical values. Also, the output (dependent variables) are represented in matrix of size 714×5 consisting of binary values representing 0 for fail and 1 for pass for each student in every five courses.

5 Multi-label Classification

Our dataset contains five output labels which makes it a multi-label classification task. When input instances are allocated only one category, the task becomes single-label classification task. The field of single-label classification is more matured than multi-label classification for solving classification problems. Therefore, in most cases the multi-label classification is transformed into a single-label classification problem through the so-called *problem transformation* methods. The two commonly used transformation methods are Label powerset (LP) [21–23] and Binary relevance (BR) [24, 25].

Binary relevance (BR) transforms the multilabel classification task by learning $|L|$ binary classifiers. BR creates $|L|$ separate datasets by combining the input feature with every label in the label set and L classifiers are used to train these dataset. When a new instance is to be classified, BR reports the final result by combining the output of each classifiers. The problem with BR is that it fails to maintain the correlation between labels in the training set and results in low predictive performance. To overcome this problem an extension of BR is introduced in [26] known as Chain classifier (CC). Like BR, CC also creates $|L|$ binary classifier but every binary classifier $C_j, j \in \{1, \dots, |L|\}$, uses all the previous labels $\{1, \dots, j-1\}$ as input for training hence creating chains of labels to maintain correlation. With this characteristics, CC improves the prediction performance of BR but introduces a small amount of time and space complexity [26].

The second problem transformation scheme is known as label power-set. *Label powerset (LP)* maps each unique label combination into a unique class. One strength of LP is that it preserves the correlation between labels. The weakness of LP is that mapping a label space of size $|L|$ requires yielding a multi-class classification problem of $2^{|L|}$ classes which can be impractically large as $|L|$ gets bigger. One way to circumvent this problem is to limit the unique label combinations only to the ones that occur in training set but this also has a problem of over-fitting and only a small number of training examples are associated with most of the classes [27, 28].

We can avoid the problems of LP by partitioning the label space into smaller labelsets and apply LP in the labelsets. We consider three common ways to partition the labelspace: 1) Random k labelsets (RAkEL) 2) Data-driven portioning [29, 30] and 3) Stochastic Block Model (SBM) [31, 32]. In this work, we used RAkEL to partition the label space in our dataset.

5.1 Label Partitioning with RAkEL_o

As discussed previously, the resulting single-label classification in LP is not equally distributed among the class values thereby causing over-fitting. The partitioning of the original label space into smaller labels increases equal distribution of class values across the input examples. This can be achieved using RAkEL which divides the number of labels into smaller labels by randomly picking the label groups. Here, k denotes the number of the labelsets. RAkEL comes in two variants: RAkEL_d, which partitions the labels into k *disjoint* labelsets, and RAkEL_o, which also partitions the labels into k labelsets but allows *overlapping* of label subspaces. We used RAkEL_o to partition the labels in our dataset since it achieves more predictive performance than RAkEL_d [28].

Algorithms 1 shows how the training and testing process is implemented using the proposed LP + RAkEL_o algorithm. Let S be the set of all labelsets of size k , and hence, $S = \binom{L}{k}$, where L is the set of labels. RAkEL_o randomly chooses m k -labelsets from S without replacement. For each k -labelsets, it learns a multilabel classifier using LP, training a total of m models. This is done by first transforming the labelsets into a single-label multi-class task. Then the trained classifier C_i outputs predictions of the testing set $\mathbf{x}$ for each label l_j in the k -labelset R_i . Since RAkEL_o allows overlap of labels, the final prediction of the label is achieved using majority vote, that is for each label the value predicted more than 50% of the time in each of the m k -labelsets is decided to be the final predicted value.

Algorithm 1 Training and testing procedure using the proposed LP + RAKEL_o

```

1: procedure LP-RAKELoMODEL( $L, k, m, D_{train}, \mathbf{x}$ )
2:    $\triangleright L$  is the set of labels
3:    $\triangleright k$  is the size of labelsets
4:    $\triangleright m$  is the number of  $k$ -labelset
5:    $\triangleright D_{train}$  is the training set
6:    $\triangleright \mathbf{x}$  is an unseen instance for testing
7:    $S \leftarrow \binom{L}{k}$   $\triangleright$  set of all possible labelsets
8:   for  $i=1$  to  $m$  do
9:      $R_i \leftarrow$  a  $k$ -labelset randomly selected from  $S$ 
10:    train an LP classifier  $C_i$  on  $D_{train}$  and labelspace  $R_i$ 
11:     $S \leftarrow S \setminus R_i$   $\triangleright$  Remove  $R_i$  from  $S$ 
12:  end for
13:
14:   $\triangleright$  Prediction of instance  $\mathbf{x}$  using majority vote
15:  Initialize a list of  $sum$  and  $votes$  of size  $|L|$  to zero.
16:  for  $i=1$  to  $m$  do
17:    for all labels  $l_j \in R_i$  do
18:       $sum_j \leftarrow sum_j + C_i(\mathbf{x}, l_j)$ 
19:       $votes_j \leftarrow votes_j + 1$ 
20:    end for
21:  end for
22:  for  $j=1$  to  $|L|$  do
23:     $avg_j \leftarrow sum_j / votes_j$ 
24:    if  $avg_j > 0.5$  then
25:       $\tilde{y}_j \leftarrow 1$ 
26:    else
27:       $\tilde{y}_j \leftarrow 0$ 
28:    end if
29:  end for
30:  return  $\tilde{\mathbf{y}} \leftarrow \{\tilde{y}_1, \dots, \tilde{y}_{|L|}\}$   $\triangleright$  return the predicted mutliable target value
31: end procedure

```

6 Environment

In this work, we used `scikit-multilearn` [33] – a `scikit-learn` API compatible library for multi-label classification in python which supports several classifiers and label partition models. We have also used `scikit-learn` [34] for data preprocessing and evaluation metrics. `scikit-learn` is widely used in the scientific Python community and supports many machine learning application areas.

6.1 Evaluation Metrics

The evaluation metrics used for single-label classification are different from multilabel classification. In single-label classification the training samples can be either correct or incorrect. In multi-label classification since labels introduce additional degrees of freedom it is important to consider multiple and contrasting measures [35]. In this study, we use three example-based measures: accuracy, Hamming loss, and Jaccard similarity as well as one label-based measure, F1, evaluated by two averaging schemes: micro and macro. We also use the following definitions as discussed in [29]:

- X is the set of objects used in the testing scenario for evaluation
- L is the set of labels that spans the output space Y
- $\bar{x}$ denotes an example object undergoing classification
- $h(\bar{x})$ denotes the label set assigned to object $h(\bar{x})$ by the evaluated classifier h
- y denotes the set of true labels for the observation $\bar{x}$
- tp_j, fp_j, fn_j, tn_j are respectively true positives, false positives, false negatives and true negatives of the of label L_j , counted per label over the output of classifier h on the set of testing objects $\bar{x} \in X$, i.e., $h(X)$
- the operator $\llbracket p \rrbracket$ converts logical value to a number, i.e. it yields 1 if p is true and 0 if p is false

The example-based metrics, Hamming loss, subset accuracy, and Jaccard similarity and the label-based metric, F1 measure, are defined as follows:

1. **Hamming Loss:** evaluates the number of times an example-label pair is misclassified, i.e., label not belonging to the example is predicted or a label belonging to the example is not predicted. $\otimes$ denotes the logical exclusive or.

$$HammingLoss(h) = \frac{1}{|X|} \sum_{\bar{x} \in X} \frac{1}{|L|} \llbracket (L_j \in h(\bar{x})) \otimes (L_j \in y) \rrbracket \quad (3)$$

2. **Accuracy score (Subset accuracy):** is instance-wise measure that evaluates the set of predicted labels for a sample that exactly match the corresponding set of true labels.

$$SubsetAccuracy(h) = \frac{1}{|X|} \sum_{\bar{x} \in X} \llbracket h(\bar{x}) = y \rrbracket \quad (4)$$

3. **Jaccard Similarity:** also simply called accuracy, is a measure of similarity between the predicted and true labels. It evaluates the coefficient of the size of the intersection between the predicted and true labels and size of their union.

$$Jaccard(h) = \frac{1}{|X|} \sum_{\bar{x} \in X} \frac{h(\bar{x}) \cap y}{h(\bar{x}) \cup y} \quad (5)$$

4. **F1 Measure:** The label-based evaluation method we use in this work is the F1 measure. F1 measure is the harmonic mean of precision and recall and is often considered to be a good indicator of the relationship between precision and recall. Precision is the measure of how much negative cases are misclassified as positives and recall is the measure of how much positive cases are misclassified as negatives. The average of these two measures is computed using two different methods, micro- and macro-averaging, which can give different interpretations specially in multi-labels settings. Micro-averaging takes the aggregate contributions of all classes true/false positives/negatives and computes the average metric. This is given as:

$$\begin{aligned} \text{precision}_{micro}(h) &= \frac{\sum_{j=1}^{|L|} tp_j}{\sum_{j=1}^{|L|} tp_j + fp_j} \\ \text{recall}_{micro}(h) &= \frac{\sum_{j=1}^{|L|} tp_j}{\sum_{j=1}^{|L|} tp_j + fn_j} \\ F1_{micro}(h) &= 2 \cdot \frac{\text{precision}_{micro}(h) \cdot \text{recall}_{micro}(h)}{\text{precision}_{micro}(h) + \text{recall}_{micro}(h)} \end{aligned} \quad (6)$$

On the other hand, macro-averaging first evaluates the metric independently for each class and takes the average over the number of labels. Hence, macro-averaging treats all classes equally, while micro-averaging does not make it suitable for cases where there is a class imbalance.

$$\begin{aligned} \text{precision}_{macro}(h, j) &= \frac{tp_j}{tp_j + fp_j} \\ \text{recall}_{macro}(h, j) &= \frac{tp_j}{tp_j + fn_j} \\ F1_{macro}(h, j) &= 2 \cdot \frac{\text{precision}_{macro}(h, j) \cdot \text{recall}_{macro}(h, j)}{\text{precision}_{macro}(h, j) + \text{recall}_{macro}(h, j)} \end{aligned} \quad (7)$$

7 Results and Discussions

7.1 Performance results of different transformation methods

In this study, we used four different base-level classifiers and compared the results of the student performance using the evaluation metrics discussed in section 6.1. Note that for all evaluation metrics, except Hamming loss, the higher the value indicates a better performance. The four base-level classifiers are SVM, Random forest (RF), KNN, and Multilayer perceptron (MLP). We also compared three problem transformations: label power-set (LP), binary relevance (BR), and classifier chains (CC), and the improved LP (LP + RAKEL_o), all discussed in section 5. We used 10-Fold stratified cross validation to represent each target value equally across each fold as our dataset is imbalanced. The prediction results of the cross validation are given in terms of mean and respective 95% confidence intervals.

Table 2 shows the overall performance of multi-label classifiers. We used bold to indicate the best performance scored for a particular classifier in a given transformation method and used underline to show the best transformation method for a given evaluation metric. As we can see from the table, SVM has scored the best performance in terms of all evaluation metrics regardless of which transformation method is used, except RF scored better accuracy when BR is used as a transformation method. When we compare the transformation methods, we can see that LP has an overall slightly better performance than BR and CC only when MLP is used as a base classifier. BR performed poorly in terms of all prediction measures when SVM and KNN are the base classifiers, while LP and CC scored comparative performances with RF, SVM, and KNN. This overall average poor performance of LP in most base classifiers mainly due to the nature of our dataset. There are a total of 30 unique labelets (when converted into a single-label multi-class task) in our dataset, slightly less than the maximum number $2^5 = 32$. The total number of examples used for training is 714 and most of the class values are associated with very few examples. This results in poor performance for the classifier since it fails from learning adequately for all classes.

To boost the performance of LP, we partitioned the original label space using RAKEL_o. By reducing the label space now more examples can be associated with the new fewer labelsets and the models will avoid overfitting. We selected the number of labelsets k to be 3, and the cluster size m to be 5 using a few trial and error rounds to achieve optimum value. As we can see in Table 2, the LP partitioned by RAKEL_o scores higher performance in terms almost all evaluation metrics using all base classifiers. The average evaluation measurement of the transformation schemes is also compared as shown in Figure 2. We can see that LP+RAKEL_o has significant performance superiority (mainly in terms of Jaccard similarity, overall accuracy, $F1_{macro}$, and $F1_{micro}$) than the rest transformation methods which proving to be the best model for student performance prediction.

Figure 2: Comparison of averaged evaluation results for all transformation methods.

Table 2: Performance comparison using different problem transformation methods (underline – best transformation method in terms of a single metric; **bold** – best within a single transformation method.)

Problem transformation	Base classifier	Evaluation measures				
		Hamming	Jaccard	Accuracy	$F1_{micro}$	$F1_{macro}$
LP	RF	0.324 ± 0.040	0.619 ± 0.061	0.231 ± 0.047	0.776 ± 0.042	0.754 ± 0.052
	SVM	0.320 ± 0.070	0.649 ± 0.081	<u>0.264 ± 0.080</u>	0.787 ± 0.061	0.765 ± 0.072
	KNN	0.334 ± 0.027	0.603 ± 0.046	0.186 ± 0.034	0.760 ± 0.034	0.727 ± 0.047
	MLP	0.334 ± 0.033	0.590 ± 0.063	0.220 ± 0.049	0.759 ± 0.041	0.735 ± 0.050
BR	RF	0.314 ± 0.029	0.620 ± 0.052	0.206 ± 0.027	0.775 ± 0.036	0.745 ± 0.045
	SVM	0.307 ± 0.041	0.638 ± 0.059	0.203 ± 0.033	0.788 ± 0.041	0.760 ± 0.048
	KNN	0.340 ± 0.035	0.582 ± 0.057	0.188 ± 0.033	0.749 ± 0.042	0.723 ± 0.051
	MLP	0.336 ± 0.034	0.589 ± 0.053	0.196 ± 0.034	0.756 ± 0.038	0.726 ± 0.045
CC	RF	0.313 ± 0.031	0.618 ± 0.052	0.228 ± 0.030	0.777 ± 0.036	0.750 ± 0.045
	SVM	<u>0.307 ± 0.040</u>	0.640 ± 0.055	0.236 ± 0.035	0.789 ± 0.039	0.765 ± 0.047
	KNN	0.335 ± 0.027	0.585 ± 0.054	0.189 ± 0.023	0.754 ± 0.037	0.729 ± 0.048
	MLP	0.336 ± 0.030	0.585 ± 0.054	0.199 ± 0.036	0.755 ± 0.038	0.727 ± 0.046
LP + RAKEL _o	RF	0.314 ± 0.063	0.652 ± 0.078	0.254 ± 0.069	0.792 ± 0.056	0.771 ± 0.067
	SVM	0.312 ± 0.064	0.654 ± 0.079	0.250 ± 0.072	<u>0.794 ± 0.056</u>	0.774 ± 0.066
	KNN	0.328 ± 0.037	0.616 ± 0.058	0.196 ± 0.038	0.769 ± 0.042	0.738 ± 0.056
	MLP	0.321 ± 0.056	0.640 ± 0.074	0.242 ± 0.062	0.784 ± 0.054	0.761 ± 0.065

7.2 Impact of feature selection on prediction performance

The original dataset contains a total of 26 features after preprocessing. Using Pearson's and Chi-square features selection methods, this was reduced to 18 features. Figure 3 shows the impact of feature selection by comparing the average prediction performance of LP + RAKEL_o model trained on the dataset with and without feature selection. The figure shows there is almost no change in performance when feature selection is used. Although in some cases feature selection improves the performance of prediction models, in this case it only removes irrelevant features without changing the performances. This indicates that almost around 30% of the features were redundant and by removing them the storage and running time is improved.

Figure 3: Comparison of the impact of feature selection (FS) on performance.

8 Conclusion and recommendation

This paper has presented a student performance prediction using a multilabel learning method that learns an ensemble of LP classifiers where each classifiers train a subset of the set of labels that are partitioned using RAKEL_o. The evaluation results conducted on four base-classifiers show that the student prediction performance model generated better results when RAKEL_o is used to partition the label space of the student's dataset. Originally, the multi-label classification using LP transformation method was compared to other well-known problem transformation methods: binary relevance and class chains and produced lower performance in terms of most evaluation measures used. However, LP classifiers was boosted when the label space is partitioned with RAKEL_o and produced better results in terms of almost all evaluation schemes than binary relevance and class chains.

As a future work, we will evaluate the proposed multi-label ensemble model on student dataset with more training samples and higher label spaces. Specifically, the model can produce pronounced results if the dataset and label space are of large sizes. Therefore, we will consider training this model on datasets collected nationwide to predict students' performance and take appropriate measures ahead of time to improve students' performance.

Funding Statement: This research did not receive any specific grant from funding agencies in the public, commercial, or any other sectors.

Acknowledgement: The authors would like to acknowledge Ethiopian Institute of Technology – Mekelle for supporting this work during the collection of dataset by writing a letters of request to high school administrators.

References

- [1] D. Tripathi, D. R. Edla, and R. Cheruku, "Hybrid credit scoring model using neighborhood rough set and multi-layer ensemble classification," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1543–1549, 2018.
- [2] C. Romero and S. Ventura, "Educational data mining: a review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 601–618, 2010.
- [3] R. d. Baker, "Data mining for education.? in international encyclopedia of education, edited by b. mcgaw, p. peterson, and e. baker," 2011.
- [4] C. Romero, S. Ventura, and P. De Bra, "Knowledge discovery with genetic programming for providing feedback to courseware authors," *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 425–464, 2004.
- [5] A. M. Shahiri, W. Husain *et al.*, "A review on predicting student's performance using data mining techniques," *Procedia Computer Science*, vol. 72, pp. 414–422, 2015.
- [6] U. K. Pandey and S. Pal, "Data mining: A prediction of performer or underperformer using classification," *arXiv preprint arXiv:1104.4163*, 2011.
- [7] S. T. Hijaz and S. R. Naqvi, "Factors affecting students' performance: A case of private colleges in bangladesh," *Journal of sociology*, vol. 3, no. 1, pp. 44–45, 2006.
- [8] M. Shovon, H. Islam, and M. Haque, "An approach of improving students academic performance by using k means clustering algorithm and decision tree," *arXiv preprint arXiv:1211.6340*, 2012.
- [9] R. Asif, A. Merceron, and M. K. Pathan, "Predicting student academic performance at degree level: a case study," *International Journal of Intelligent Systems and Applications*, vol. 7, no. 1, p. 49, 2014.
- [10] H. Agrawal and H. Mavani, "Student performance prediction using machine learning," *International Journal of Engineering Research and Technology*, vol. 4, no. 03, pp. 111–113, 2015.
- [11] P. Cortez and A. M. G. Silva, "Using data mining to predict secondary school student performance," 2008.
- [12] M. Pandey and S. Taruna, "A comparative study of ensemble methods for students' performance modeling," *International Journal of Computer Applications*, vol. 103, no. 8, 2014.
- [13] A. Satyanarayana and M. Nuckowski, "Data mining using ensemble classifiers for improved prediction of student academic performance," 2016.
- [14] N. Iam-On and T. Boongoen, "Improved student dropout prediction in thai university using ensemble of mixed-type data clusterings," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 497–510, 2017.

- [15] P. Kumari, P. K. Jain, and R. Pamula, "An efficient use of ensemble methods to predict students academic performance," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–6.
- [16] E. A. Yekun, "Dataset for Student Performance Prediction," 2020. [Online]. Available: <https://doi.org/10.7910/DVN/WHBU4P>
- [17] L. Ladha and T. Deepa, "Feature selection methods and algorithms," *International journal on computer science and engineering*, vol. 3, no. 5, pp. 1787–1797, 2011.
- [18] D. R. Edla, D. Tripathi, R. Cheruku, and V. Kuppli, "An efficient multi-layer ensemble framework with bpsogsa-based feature selection for credit scoring data analysis," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 6909–6928, 2018.
- [19] D. Tripathi, D. R. Edla, R. Cheruku, and V. Kuppli, "A novel hybrid credit scoring model based on ensemble feature selection and multilayer ensemble classification," *Computational Intelligence*, vol. 35, no. 2, pp. 371–394, 2019.
- [20] D. Tripathi, D. R. Edla, V. Kuppli, A. Bablani, and R. Dharavath, "Credit scoring model based on weighted voting and cluster based feature selection," *Procedia computer science*, vol. 132, pp. 22–31, 2018.
- [21] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.
- [22] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *2008 eighth IEEE international conference on data mining*. IEEE, 2008, pp. 995–1000.
- [23] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *European conference on machine learning*. Springer, 2007, pp. 406–417.
- [24] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDDWM)*, vol. 3, no. 3, pp. 1–13, 2007.
- [25] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2004, pp. 22–30.
- [26] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.
- [27] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence and loss minimization in multi-label classification," *Machine Learning*, vol. 88, no. 1-2, pp. 5–45, 2012.
- [28] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2010.
- [29] P. Szymański, T. Kajdanowicz, and K. Kersting, "How is a data-driven approach better than random choice in label space division for multi-label classification?" *Entropy*, vol. 18, no. 8, p. 282, 2016.
- [30] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE transactions on knowledge and data engineering*, vol. 20, no. 2, pp. 172–188, 2007.
- [31] E. Abbe, "Community detection and stochastic block models: recent developments," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6446–6531, 2017.
- [32] T. P. Peixoto, "Nonparametric bayesian inference of the microcanonical stochastic block model," *Physical Review E*, vol. 95, no. 1, p. 012317, 2017.
- [33] P. Szymański and T. Kajdanowicz, "A scikit-based python environment for performing multi-label classification," *arXiv preprint arXiv:1702.01460*, 2017.
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [35] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern recognition*, vol. 45, no. 9, pp. 3084–3104, 2012.