Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 3 (2024) 346 - 354 e-ISSN: 2580-0760

Comparative Analysis of Hybrid Model Performance Using Stacking and Blending Techniques for Student Drop-Out Prediction in MOOC

Muhammad Ricky Perdana Putra[1], Ema Utami[2]
[1,2]Magister of Informatics Engineering, Universitas Amikom Yogyakarta, Yogyakarta, Indonesia
[1]muhammad.ricky@students.amikom.ac.id, [2]ema.u@amikom.ac.id

Abstract

Despite being in high demand as a lifelong learner and academic material supplement, the implementation of Massive Open Online Courses (MOOC) has problems, one of which is the dropout rate (DO) of students which reaches 93%. As one of the solutions to this problem, Machine Learning can be utilized as a risk management and early warning system for students who have the potential to drop out. The use of ensemble techniques to build models can improve performance, but previous research has not reviewed the most optimal ensemble technique for this case study. As a form of contribution, this study will compare the performance of models built from stacking and blending techniques. The algorithms used in the base model are KNN, Decision Tree, and Naïve Bayes, while the meta-model uses XGBoost. These algorithms are used to build models with stacking and blending techniques. The experimental results using stacking are 82.53% accuracy, 84.48% precision, 94.12% recall, and 89.04% F1-Score. Meanwhile, blending obtained 83.39% accuracy, 85.31% precision, 94.21% recall, and 89.54% F1-Score. These results are supported by model testing using k-fold cross-validation and confusion matrix techniques which show the same results. That is, blending is 0.86% higher than stacking so it can be concluded that blending has better performance than stacking in the MOOC student dropout prediction case study.

Keywords: *machine learning, classification, stacking, blending, MOOC*

How to Cite: Muhammad Ricky Perdana Putra and Ema Utami, "Comparative Analysis of Hybrid Model Performance Using Stacking and Blending Techniques for Student Drop Out Prediction In MOOC", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 8, no. 3, pp. 346 - 354, Jun. 2024.
DOI: https://doi.org/10.29207/resti.v8i3.5760

1. Introduction

Massive Open Online Course (MOOC) was developed to fulfil the needs of lifelong learners or as a supplement to formal education learning [1]. Although the enthusiasm of students is quite high, the reality of MOOC implementation is not free from problems. One of them is the dropout rate, which reaches 66% to 93% [2], [3]. The causes vary, ranging from lack of social support, motivation, and perseverance [4], difficulty understanding the material, lack of interaction with the instructor [5], lack of understanding of learning goals and intentions [6] and lack of peer support [7] -[9].

The impact of students who drop out includes difficulty getting adequate employment and income in the future, thus worsening economic and social conditions [4], while MOOC organisers can affect reputation, rankings, and income [10]. Therefore, risk management and early warning systems specifically for students who have the potential to drop out are needed, one of which is by utilising ML technology such as using classification algorithms.

As a prediction system, ML can provide notifications to learners and instructors on a regular basis. For the learners, this can be used as motivation to continue completing the course. As for the instructor, it can be used as a basis for providing motivation and special attention because it can reduce the potential for dropouts by 14% [11]. In addition, for the course organisers, the prediction results can be used to simplify the learning path or adjust the material provided [12].

There are several classification algorithms that are popular and used by previous researchers including Logistic Regression (LR), K-Nearest Neighbor (KNN), and Random Forest (RF) [2]. The research was conducted by Zengxiao Chi, Shuo Zhang, and Lin Shing. The dataset used from the HarvardX Platform MOOC in the range of 2012 to 2013 with data totalling 416,921 rows and 21 features. After the pre-processing

Received: 19-03-2024 | Accepted: 16-05-2024 | Published Online: 01-06-2024

stage, 241,992 lines were filtered. Model testing was done with 5-fold cross-validation. In this study, RF obtained the highest accuracy value of 91.72%. Even so, these results can still be improved.

One way to improve the performance of ML models is ensemble learning. The ensemble is done by combining models in the first layer called weak learners that function as complex non-linear feature conversions and the second layer called meta learners utilizes residues from previous models [13]. There are four techniques for building ensemble models including bootstrap aggregating (bagging), boosting, stacking, and blending. An example of a model built from bagging techniques is Random Forest, while an example of boosting is the Extreme Gradient Boosting (XGBoost) algorithm [14].

Stacking and blending techniques are built from multiple base models and one meta-model. The difference between the two techniques lies in the division of the dataset. If stacking uses a dataset that is divided into two for training and testing, in blending the training data is separated into ensemble and blender [13], [14]. The ensemble data is used to train the base model and tested with blender data. The results are combined with blender data as new attributes trained on the metamodel and then tested with testing data. This makes blending not use overlapping data compared to stacking which in fact stacks prediction results data together for training and testing.

In the case study of dropout prediction in a massive open online course (MOOC), the use of ensemble algorithms built with stacking or blending techniques can improve the performance of the prediction model. One of the ensemble models proposed by Kumar et al. is called Ensemble Deep Learning Network (EDLN) [15]. The dataset used is KDD Cup 2015 which contains student activity logs on XuetangX MOOC from China. The data selected is the first five weeks. The results of the study obtained an accuracy value of 97.4%. The accuracy value is still not confirmed for complex data during a one-course period.

Research conducted by Shou et al. by building a Multiscale Full Convolutional Network and Variational Information Bottlenecks (MFCN-VIB) [16]. The model can overcome noise in student behaviour time series data that may cause interference. The dataset used is the same as previous research, namely KDD Cup 2015. The results of this study are a precision value of 0.887, recall 0.960, F1-Score 0.922, and AUC 0.872. One of the weaknesses of this research is that the model built is quite complex so the execution time is longer, namely 133 seconds. In addition, the accuracy value is not written.

Another ensemble model proposed by Fu et al. called CLSA is a combination of Convolutional Neural Network (CNN) and Bi Long Short-Term Memory (LSTM) [17]. The dataset used is the same as previous research, namely KDD Cup 2015 which has been pre-processed so that 60 thousand activity log data from 12 thousand student data are randomly selected and 7 features are selected related to behavioural characteristics in the first to fifth week. With CLSA, the accuracy was increased by 2.8% from the basic model to 87.6%.

Although the three studies above have concluded that ensemble can improve model performance, they still do not explain the most optimal ensemble technique to use between stacking and blending. Therefore, this study will conduct a comparative analysis of the performance of ensemble algorithms with stacking and blending techniques to determine the most optimal technique for improving model performance. To get accurate results, the datasets and algorithms used are made the same.

In order not to widen the research conducted, there are several limitations, namely the type of data used is single data and not time series data, hybrid models built for academic research purposes and not used for implementation in MOOCs and not for optimal learning path customization. The research starts from a literature study, model building is done with Google Colab using Python programming language and supported by libraries from SK-Learn, obtaining test results and conducting descriptive analysis to determine the most optimal ensemble technique.

2. Research Methods

This research compares the prediction performance of hybrid models built with stacking and blending techniques. In order to get an equal comparison, the base learner algorithms used include KNN, Decision Tree (DT), and Naïve Bayes. The meta-learner algorithm used is Extreme Gradient Boosting commonly abbreviated as XGBoost. An explanation of the flow and reasons for selecting the four algorithms will be explained in the next paragraph below.

The dataset used in this study is the same as the previous three studies, namely KDD Cup 2015. The dataset was uploaded by contributor Anas Nofal on Kaggle.com and can be downloaded for free (https://www.kaggle.com/datasets/anasnofal/mooc-data-xuetangx) and processed into frequency per activity log. The raw data (http://moocdata.cn/data/user-activity) is in JSON (JavaScript Object Notation) format, while the data that has been processed based on frequency is presented in tabulations.

The class distribution in the training data is 137,237 DO classes and 43,476 non-DO classes. The percentage comparison is 75.95% and 24.05%. Meanwhile, the class distribution in the test data is 33,896 DO classes and 11,033 non-DO classes. The percentage comparison is 75.44% and 24.56%. This is in accordance with the results of previous research [2][3]. Visually, the class distribution on the training and test data is presented in Figure 1 and Figure 2.

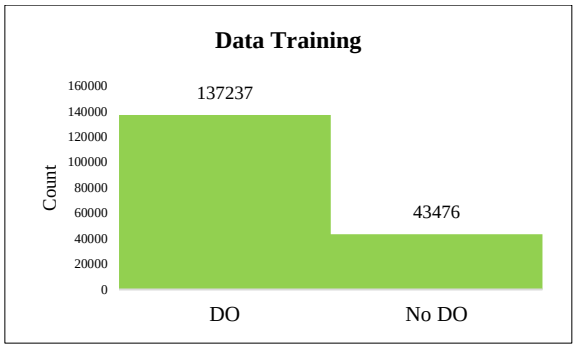

Figure 1. Training Data Class Distribution

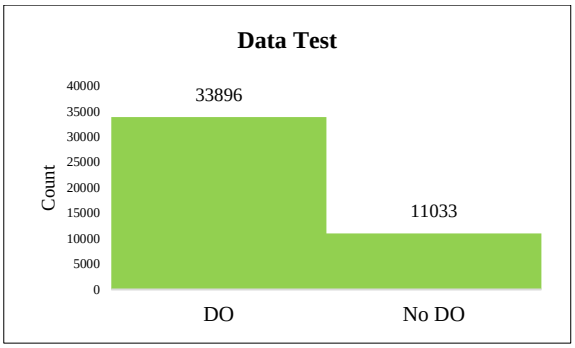

Figure 2. Testing Data Class Distribution

The research flow begins with a literature study aimed at finding research gaps or practical problems related to the case used as the object of research. Furthermore, collecting datasets from the site previously described. The pre-processing stage carried out is (1) ensuring data in the form of numerical numbers, (2) replacing empty values (null) to 0, (3) feature selection, and (4) data scaling which aims to optimize the potential for increasing data accuracy [18]. The type of scaling performed is Standard Scaler with the aim of making the average zero and variance one [19].

In the feature selection sub-stage, was done manually by selecting features related to user activity logs only and 22 features were successfully selected. The manual feature selection method can be further optimized by utilizing feature weighting techniques or genetic algorithms so that only strong features are selected. This can be used as a topic for further research. Then, to find out the correlation between features, it is visualized with a heatmap as in Figure 3.

	Click About	Click Courseware	Click Forum	Click Info	Click Progress	Close Courseware	Close Forum	Create Comment	Create Thread	Delete Comment	Delete Thread	Load Video	Pause Video	Play Video	Problem Check	Problem Check Correct	Problem Check Incorrect	Problem Get	Problem Save	Reset Problem	Seek Video	Stop Video
Click About	1,00	0,48	0,37	0,66	0,27	0,40	0,01	0,25	0,16	0,07	0,03	0,43	0,33	0,23	0,21	0,23	0,09	0,02	0,09	0,06	0,18	0,05
Click Courseware	0,48	1,00	0,36	0,68	0,52	0,87	0,01	0,21	0,17	0,08	0,04	0,85	0,63	0,49	0,47	0,49	0,26	0,05	0,20	0,11	0,37	0,10
Click Forum	0,37	0,36	1,00	0,55	0,34	0,32	0,02	0,57	0,38	0,25	0,21	0,28	0,23	0,19	0,19	0,22	0,11	0,03	0,12	0,03	0,17	0,02
Click Info	0,66	0,68	0,55	1,00	0,53	0,60	0,01	0,30	0,24	0,11	0,07	0,57	0,45	0,37	0,38	0,41	0,20	0,04	0,16	0,11	0,30	0,06
Click Progress	0,27	0,52	0,34	0,53	1,00	0,45	0,01	0,16	0,16	0,05	0,03	0,41	0,33	0,27	0,44	0,46	0,27	0,04	0,20	0,12	0,25	0,05
Close Courseware	0,40	0,87	0,32	0,60	0,45	1,00	0,01	0,22	0,19	0,09	0,05	0,85	0,62	0,52	0,49	0,49	0,27	0,07	0,19	0,09	0,40	0,12
Close Forum	0,01	0,01	0,02	0,01	0,01	0,01	1,00	0,01	0,01	0,00	0,00	0,01	0,01	0,01	0,02	0,00	0,03	0,00	0,00	0,00	0,01	0,00
Create Comment	0,25	0,21	0,57	0,30	0,16	0,22	0,01	1,00	0,38	0,38	0,24	0,20	0,15	0,12	0,12	0,16	0,05	0,02	0,09	0,00	0,12	0,01
Create Thread	0,16	0,17	0,38	0,24	0,16	0,19	0,01	0,38	1,00	0,16	0,24	0,18	0,13	0,12	0,14	0,15	0,08	0,03	0,07	0,02	0,11	0,02
Delete Comment	0,07	0,08	0,25	0,11	0,05	0,09	0,00	0,38	0,16	1,00	0,51	0,07	0,05	0,04	0,04	0,05	0,02	0,01	0,03	0,00	0,04	0,00
Delete Thread	0,03	0,04	0,21	0,07	0,03	0,05	0,00	0,24	0,24	0,51	1,00	0,04	0,03	0,03	0,03	0,03	0,02	0,01	0,01	0,00	0,03	0,00
Load Video	0,43	0,85	0,28	0,57	0,41	0,85	0,01	0,20	0,18	0,07	0,04	1,00	0,68	0,59	0,50	0,51	0,27	0,06	0,18	0,10	0,45	0,11
Pause Video	0,33	0,63	0,23	0,45	0,33	0,62	0,01	0,15	0,13	0,05	0,03	0,68	1,00	0,97	0,38	0,37	0,24	0,05	0,16	0,10	0,64	0,06
Play Video	0,23	0,49	0,19	0,37	0,27	0,52	0,01	0,12	0,12	0,04	0,03	0,59	0,97	1,00	0,35	0,33	0,22	0,05	0,15	0,09	0,66	0,06
Problem Check	0,21	0,47	0,19	0,38	0,44	0,49	0,02	0,12	0,14	0,04	0,03	0,50	0,38	0,35	1,00	0,86	0,77	0,09	0,17	0,14	0,33	0,05
Problem Check Correct	0,23	0,49	0,22	0,41	0,46	0,49	0,00	0,16	0,15	0,05	0,03	0,51	0,37	0,33	0,86	1,00	0,52	0,08	0,29	0,15	0,30	0,05
Problem Check Incorrect	0,09	0,26	0,11	0,20	0,27	0,27	0,03	0,05	0,08	0,02	0,02	0,27	0,24	0,22	0,77	0,52	1,00	0,10	0,15	0,13	0,23	0,02
Problem Get	0,02	0,05	0,03	0,04	0,04	0,07	0,00	0,02	0,03	0,01	0,01	0,06	0,05	0,05	0,09	0,08	0,10	1,00	0,04	0,01	0,04	0,00
Problem Save	0,09	0,20	0,12	0,16	0,20	0,19	0,00	0,09	0,07	0,03	0,01	0,18	0,16	0,15	0,17	0,29	0,15	0,04	1,00	0,13	0,15	0,01
Reset Problem	0,06	0,11	0,03	0,11	0,12	0,09	0,00	0,00	0,02	0,00	0,00	0,10	0,10	0,09	0,14	0,15	0,13	0,01	0,13	1,00	0,07	0,01
Seek Video	0,18	0,37	0,17	0,30	0,25	0,40	0,01	0,12	0,11	0,04	0,03	0,45	0,64	0,66	0,33	0,30	0,23	0,04	0,15	0,07	1,00	0,04
Stop Video	0,05	0,10	0,02	0,06	0,05	0,12	0,00	0,01	0,02	0,00	0,00	0,11	0,06	0,06	0,05	0,05	0,02	0,00	0,01	0,01	0,04	1,00

Figure 3. Feature Correlation Heatmap

After pre-processing, data splitting is done. The stacking technique does not require data splitting anymore because it only uses training data and test data. Meanwhile, the blending technique requires splitting the training data with a ratio of 60:40 so that it becomes ensemble data totalling 108,427 and blender data totalling 72,286. The class distribution on the ensemble data is 82,515 DO classes and 25,912 non-DO classes. And then, the class distribution on the blender data is 54,722 DO classes and 17,564 non-DO classes. Visually, the class distribution on the training and test data is presented in Figure 4 and Figure 5.

The ready data will be subjected to data training and testing processes with three algorithms in the first layer including KNN, Decision Tree, and Naïve Bayes. The stacking technique performs training and testing using training data and test data. While the blending technique conducts training with ensemble data and testing with

blender data and testing data. In stacking the test results are collected into one new data frame and in blending the test results are used as new features in the blender data and testing data.

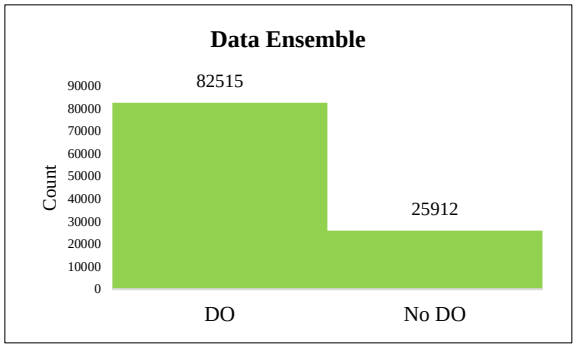

Figure 4. Ensemble Data Class Distribution

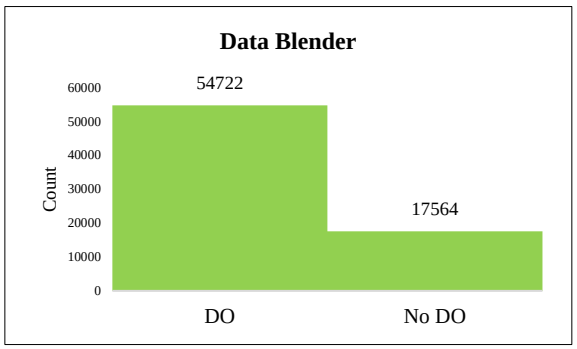

Figure 5. Blender Data Class Distribution

KNN was chosen because it was widely used by previous researchers such as Nithya and Umarani [20] and Chi et al. [2]. The advantages of KKN are the flexibility of the *k* value that can be changed according to certain needs or conditions, for example, based on mean error, suitable for binary classification, and effective on data that is complex enough to produce more accurate predictions [21]. The value of *k* in the KNN algorithm is determined based on the results of pre-research using the dataset presented in graph form in Figure 6, showing that *k*=9 has the lowest mean error value.

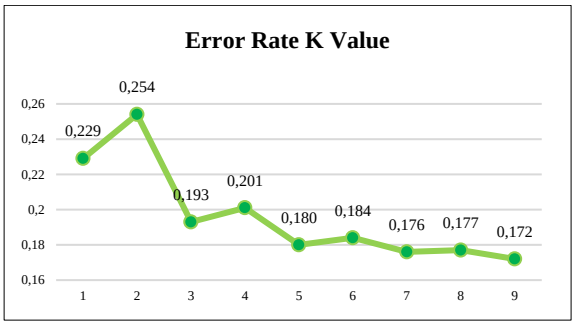

Figure 6. Error Rate *k* Value

KNN aims to find the closest distance or the highest similarity value. The stages in building the KNN model are (1) determining the value of *k*, (2) calculating the Euclidean distance with the formula in Formula 1, (3) determining the closest distance with the minimum distance in K, (4) the nearest neighbor label and the dominant label are used to predict the new data class. In addition, KNN prediction can be determined from the similarity formula written in Formula 2.

$$D = \sqrt{(X1 - Y1)^2 + (X2 - Y2)^2} \quad (1)$$

$$Similarity\ (T,S) = \frac{\sum_{i=1}^{n} f\ (T_i,\ S_i)\ x\ w_i}{\sum_{i=1}^{n} w_i} \quad (2)$$

DT as the second base model was chosen because it can find unexpected data patterns, is suitable for classification [22], produces acceptable accuracy values, and can handle numeric data [23]. Therefore, in the pre-processing stage, all data is ensured to be numeric in order to produce maximum accuracy in DT. Previous studies that applied DT are Park and Yoo [24], and Moreno-Marcos et al. [8].

There are three components in a decision tree including roots, branches, and leaves. The feature used as the root or root node is determined through the gain formula in Formula 3. To find the gain value, it is necessary to know the entropy through the formula in Formula 4. After all attributes become branches, the leaves can be determined whose values are classification labels.

$$Entropy\ (S) = \sum_{i=1}^{n} - pi\ *\ \log_2 pi \quad (3)$$

$$Gain\ (S,A) =\ Entropy\ (S)\ - \sum_{i=1}^{n} \frac{|S_i|}{|S|} *\ Entropy\ (S_i) \quad (4)$$

Similar to the other two base model algorithms, Naïve Bayes functions as a non-linear feature converter. Some of the advantages of Naïve Bayes are simplicity, fast training and execution time, and good performance [17][18]. The Naïve Bayes algorithm is processed based on the equation proposed by Thomas Bayes and known as Bayes Theorem with the formula written in Formula 5 and can determine the probability value of the target class.

$$P\ (H|X) = \frac{P\ (X|H)\ P(H)}{P(X)} \quad (5)$$

The notation in Bayes Theorem is divided into two variables, X as the sample data of the unknown class and C as the hypothesis that X is the class data. P(X|C) is the probability based on the conditions in the hypothesis, P(X) is the probability of the observed sample data, and P(C) is the probability of the hypothesis C. The largest probability will be chosen as the prediction result. Previous research using Naïve Bayes is Zheng et al. [25].

The second layer uses the XGBoost algorithm. New frame data in stacking is used as training data and testing is done using testing data. In addition, in blending, blender data that has been added with features is used as training data and testing data that has been added with features is used as test data. Then, the test results are presented in tabulated form for easy reading and understanding.

The selection of the XGBoost algorithm is used as a meta-learner to utilize the previous model residue in the form of base model prediction results. XGBoost is an

algorithm that applies the concept of Gradient Boosting Decision Tree (GBDT) [26] while improving performance by adjusting iterative learning features to reduce the loss function [27]. The use of XGBoost was conducted by Wunnasri et al. [28] which is intended for the first phase of the model as a classification algorithm.

The advantage of XGBoost is that the computational process is 10 times faster and the accuracy value is higher than Random Forest [29]. The prediction concept in XGBoost utilizes a decision tree. Formula 6 is a differentiable loss function to measure whether the model that has been built matches the training data and Formula 7 determines the complexity of the model [30]. As the complexity of the model increases, the corresponding score will decrease in value.

$$\gamma_1 = \sum_k^K f_k\,(X_i), f_k \in F \quad (6)$$

$$obj(\theta) = \sum_{i=0}^{n} l\,(y_i, \gamma_1) + \sum_k^K \Omega\,(f_k) \quad (7)$$

To validate the prediction results of the model that has been built, *k*-fold cross-validation and confusion matrix techniques are performed. To perform validation testing, all datasets that have been split will be combined into one and then processed or will be split again based on the iteration of the *k*-fold. Each *k*-fold iteration is calculated for accuracy, precision, recall, and F1-Score. These results will be compared and analyzed to produce a conclusion.

The research flow is designed and structured to get comparable results by making the same treatment, starting from the dataset used, pre-processing, the algorithm used, and the test validation technique. The difference is the separation of datasets, the training for the second layer model, namely stacking, uses the test results from the first layer which are stacked while blending uses the test results from the first layer to be used as additional features. Visually, the research flow is presented in Figure 7.

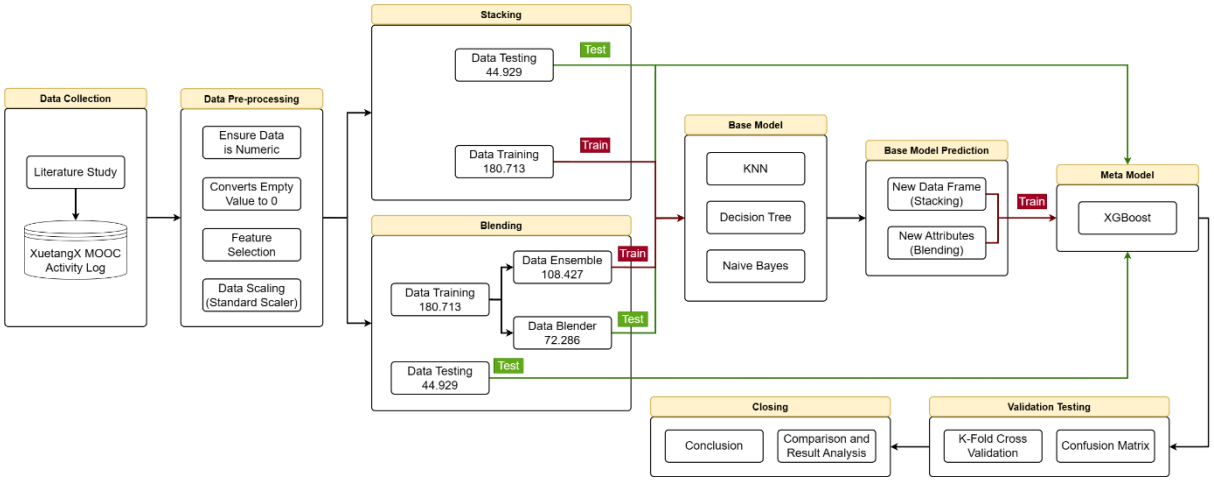

Figure 7. Research Flow

3. Results and Discussions

First, a hybrid model was built with the stacking technique. Training 180,713 data with KNN, Decision Tree, and Naïve Bayes algorithms and testing was done with 44,929 data. The *k*-fold value chosen is five which means the data will be divided into five subsets, one subset is used as testing and the other is used as training. For each iteration, the confusion matrix will be calculated. In addition, the execution time is also calculated to determine the prediction speed.

The test results of the first three layers of algorithms, namely KNN, get an accuracy value of 82.43%, precision of 85.10%, recall of 92.99%, and F1-Score of 88.88%. Decision Tree results are 76.97% accuracy, 84.25% precision, 85.44% recall, and 84.84% F1-Score. Then, the results of Naïve Bayes are accuracy 81.08%, precision 82.47%, recall 95.14%, and F1-Score 88.35%. Furthermore, *k*-fold cross-validation and confusion matrix testing are presented in Table 1.

KNN has the flexibility to determine the *k* value as the nearest neighbor circumference. The greater the value of *k*, the more neighbors there are so that it makes predictions more accurate, especially if the data classification is binary because the label is determined based on the majority of labels. However, prediction using KNN has the disadvantage of requiring a fairly long execution time on average of 213.00 seconds so the KNN model is suitable for predictions that require high accuracy and ignore execution time.

In addition, Naïve Bayes gets the fastest average execution time which is less than one second to be precise 0.29. Although the accuracy value obtained is not as good as KNN. Then, the DT algorithm gets the lowest accuracy value because it is unable to handle the complexity of the attributes used. The more branches that are built, the more complex the decision. So, DT is more suitable for data that has fewer attributes and according to Ang Ji and David Levinson's research bootstrap aggregating (bagging) techniques can overcome these problems [31]. One implementation of the bagging technique is Random Forest which was introduced by Leo Breiman in 2001.

Table 1. Stacking Technique Base Model Testing Results

Model	Test Variable	Iteration					Mean
		1	2	3	4	5	
KNN	Accuracy	82,94	82,64	82,60	82,93	82,79	82,78
	Precision	85,51	85,07	85,35	85,68	85,40	85,40
	Recall	93,93	93,31	93,08	93,10	93,29	93,23
	F1-Score	89,26	89,00	89,05	89,24	89,17	89,14
	Execution Time	222,29	216,17	212,03	206,55	207,95	213,00
Decision Tree	Accuracy	77,06	76,94	76,89	77,44	77,04	77,07
	Precision	84,42	84,19	84,49	84,87	84,62	84,52
	Recall	85,58	85,39	85,23	85,58	85,28	85,41
	F1-Score	85,00	84,79	84,86	85,22	84,95	84,96
	Execution Time	3,46	2,70	3,20	2,36	2,24	2,79
Naïve Bayes	Accuracy	81,38	80,86	80,96	81,58	81,08	81,17
	Precision	82,71	82,16	82,44	82,80	82,39	82,50
	Recall	95,43	95,23	95,22	95,63	95,51	95,41
	F1-Score	88,62	88,22	88,37	88,75	88,47	88,49
	Execution Time	0,29	0,27	0,30	0,28	0,30	0,29

Next, meta-model building with XGBoost. The XGBoost library has been developed by the Distributed Machine Learning Community (DMLC). Residual data collected from the previous three base models are combined into one data frame for retraining with XGBoost. The test results show the accuracy value of the stacking model is 82.53%, precision 84.48%, recall 94.12%, and F1-Score 89.04%. Then, the results of testing the stacking model with *k*-fold cross-validation and confusion matrix techniques are shown in Table 2.

Table 2. Stacking Technique Hybrid Model Testing Results

Model	Test Variable	Iteration					Mean
		1	2	3	4	5	
Ensemble Stacking	Accuracy	83,15	82,70	82,65	83,17	82,80	82,89
	Precision	84,93	83,93	84,64	85,05	85,87	84,88
	Recall	94,59	95,25	94,28	94,45	92,60	94,23
	F1-Score	89,50	89,23	89,20	89,51	89,11	89,31
	Execution Time	211,84	232,16	212,32	212,15	211,59	216,01

Utilization of the previous residue makes XGBoost get an average value at a *k*-fold of 82.89%. While the average accumulation on the base model is 80.34% it can be concluded that the hybrid model built with the stacking technique can improve performance by 2.55%. However, in terms of execution time, XGBoost takes quite a long time, which is an average of 216.01 seconds. This is due to the complexity of the XGBoost algorithm and can be reviewed for further research regarding the most optimal and fast algorithm to be used as a metamodel.

Second, the next experiment is to build a model with the blending technique. Slightly different from stacking, blending does not use the base model test results as training data on the meta-model but the test results will be added to the blender data and testing data as new attributes and will be trained on the meta-model so that initially 22 features become 25 features. The large amount of data and features used can affect performance, but the blending technique can overcome this by separating the training data in each layer so that it does not accumulate like stacking.

The amount of training data or ensemble in the base model is 108,427 and the test data or blender is 72,286. After training and testing data, KNN produces an accuracy of 82.65%, precision of 85.27%, recall of 93.19%, and F1-Score of 89.05%. Decision Tree obtained an accuracy value of 76.90%, precision of 84.28%, recall of 85.41%, and F1-Score 84.84%. Then, Naïve Bayes gets an accuracy value of 81.19%, precision of 82.61%, recall of 95.19%, and F1-Score of 88.46%. The difference in general, the average execution time on blending is faster because the amount of data used is less. The complete results of *k*-fold cross-validation and confusion matrix testing are presented in Table 3.

The data pattern in these results is the same as the base model stacking test. KNN gets the highest score in terms of accuracy. Compared to stacking, KNN on blending gets a higher accuracy value of 0.20% and the execution time is faster. Then, the Decision Tree gets the lowest accuracy value compared to the other two algorithms and there is an increase in the accumulated accuracy value of 0.24%. The same thing happens to Naïve Bayes, which is an increase of 0.12%. Here Naïve Bayes is the algorithm with the fastest execution time which only takes 0.27 seconds to do prediction. As explained earlier, the results are not put together in the frame data as in the stacking technique, but the results are put together with blender data and testing data as new features named 'knn_predictions', 'dt_predictions', 'nb_predictions'. Previously, the features used were 22 features, then three new features were added so that there were 25 features. The assumption of adding these new features is that there is an expected performance improvement when tested with *k*-fold cross-validation and confusion matrix.

Table 3. Blending Technique Base Model Testing Results

Model	Test Variable	Iteration					Mean
		1	2	3	4	5	
KNN	Accuracy	82,74	82,64	82,79	83,07	82,78	82,80
	Precision	85,28	85,26	85,39	85,81	85,40	85,43
	Recall	93,30	93,18	93.28	93.27	93,28	93,26
	F1-Score	89,11	89,04	89,16	89,39	89,17	89,17
	Execution Time	183,42	218,82	188,79	164,11	160,06	183,04
Decision Tree	Accuracy	77,11	77,08	77,37	77,53	77,45	77,31
	Precision	84,40	84,49	84,62	85,12	84,76	84,68
	Recall	85,57	85,40	85,77	85,56	85,72	85,60
	F1-Score	84,98	84,94	85,19	85,34	85,24	85,14
	Execution Time	3,04	2,57	2,16	1,86	1,88	2,30
Naïve Bayes	Accuracy	81,17	81,19	81,14	81,78	81,17	81,29
	Precision	82,59	82,58	82,53	83,31	82,52	82,71
	Recall	91,18	95,25	95,31	95,24	95,42	95,28
	F1-Score	88,44	88,46	88,46	88,88	88,50	88,55
	Execution Time	0,19	0,19	0,21	0,24	0,21	0,21

After training using blender data, then testing using test data. The following are the results of testing the blending technique, namely accuracy 83.39%, precision 85.31%, recall 94.21%, and F1-Score 89.54%. Compared to stacking, there is an increase of 0.86%. The difference as well as the improvement occurred due to the dataset and mechanism in the meta-model. There are fewer datasets in blending so the complexity is lower. The construction of the meta-model from XGBoost which utilizes blender data with new features for training and testing data for testing is more effective than stacking which uses test data for training as well as testing on the meta model. Furthermore, the test results using *k*-fold cross-validation and confusion matrix are presented in Table 4.

Table 4. Blending Technique Hybrid Model Testing Results

Model	Test Variable	Iteration					Mean
		1	2	3	4	5	
Ensemble Blending	Accuracy	83,60	83,56	83,38	83,90	83,46	83,58
	Precision	85,63	85,44	85,46	85,81	85,46	85,56
	Recall	94,22	94,20	94,14	94,44	94,27	94,25
	F1-Score	89,72	89,61	89,59	89,92	89,65	89,70
	Execution Time	365,19	334,87	369,51	391,92	370,49	366,40

Based on testing on *k*-fold cross-validation, the difference in the highest accuracy value in stacking and blending is 0.88%. This proves that the hybrid model built with blending has a better performance value than stacking. However, in terms of execution time, stacking is faster than blending with a difference of 134.41 seconds. This is because the features used for training are fewer in stacking which is only three features while blending is 25 features.

Then, in other confusion matrix values such as precision, recall, and F1-Score, blending is higher than stacking. Precision gives an idea of how the model can predict the positive class correctly among all positive predictions. Recall, also called sensitivity, is an evaluation to describe how well a model can correctly identify the positive class. Finally, the F1-Score reflects the balance between precision and sensitivity. In other words, F1-Score gives an idea of how good the model is at predicting true positives and true negatives.

In general, the results of this study are in line with the research results of previous studies that ensemble can improve the performance value of hybrid models. Blending has better precision, recall, and F1-Score compared to the MFCN-VIB model proposed by Shou et al. and claimed to be able to overcome noise [16]. However, in the variable execution time, MFCN-VIB is faster with a considerable difference of 203.14 seconds. Another difference is in the type of data used, MFCN-VIB uses time series data while blending uses a single time series data.

The CLSA model proposed by Fu et al. and is a combination of CNN and Bi-LSTM algorithms [17] has a lower difference of 1.75% with blending, namely CLSA getting 87.4%. The dataset used is also different because CLSA only used 60 thousand data and selected seven features related to the characteristics of students in the first week to the fifth week. Meanwhile, stacking and blending used all the data in KDD Cup 2015. The number of features used is different and the complexity of the hybrid model built is also different.

Finally, research was conducted by Kumar et al. with an ensemble model called EDLN [15]. The dataset used was only the first five weeks of the course and the amount of data was not written. The accuracy of the model is 97.4%. These results are influenced by the amount of data used and cannot be confirmed for the same results on complex data and over a specific period. In comparison, in this research, the blending achieved an accuracy of 89.35% and was built with all the data used in the 2015 KDD Cup with a total of 225k data, making it more complex.

Despite getting good results, this research still has weaknesses including the hybrid model of the blending

technique still has the longest execution time and there is no testing based on the Area Under Curve (AUC) which can visualize all possible classification thresholds [32]. Future research can experiment with finding the best combination of algorithms that can reduce the execution time to be built in hybrid with blending techniques or apply metaheuristic type optimization techniques.

4. Conclusions

The results prove that ensemble or hybrid models can improve accuracy. This is in line with the results of the three studies listed in the previous section. However, there are some things that differ such as the amount and shape of the data as well as the complexity or combination of algorithms chosen to build the model. Building a hybrid model with stacking gets an accuracy value of 82.53% while blending gets an accuracy value of 83.39%. This means that blending gets 0.86% higher results in the case study of dropout students in MOOCs with data classification in the form of binary, single data and not time series.

In addition, as a result of the additional lines of code in the hybrid model, the model execution time becomes longer. This can be a gap for further research to improvise the model so that the execution time is faster. In addition, related to the related features used, can be reviewed and selected again to ensure the correlation between features is strong, such as using genetic algorithms. Then, it can use metaheuristic optimization techniques such as Particle Swarm Optimization (PSO), Ant Colony, or Komodo Mlipir Algorithm (KMA).

The results of this research are expected to provide inspiration and reference for similar research, namely dropout prediction in MOOCs using ensemble algorithms. In addition, the research results can be applied to the real world as an early warning system that sends regular notifications so as to reduce the potential for students to drop out. The information can be utilized by teachers to provide intensive guidance and for the MOOC system to determine a dynamic learning path so that the course can still be completed by students.

References

[1] L. Ma dan C. S. Lee, "Drivers and barriers to MOOC adoption: perspectives from adopters and non-adopters," *Online Inf. Rev.*, vol. 44, no. 3, hal. 671–684, 2020, doi: 10.1108/OIR-06-2019-0203.

[2] Z. Chi, S. Zhang, dan L. Shi, "Analysis and Prediction of MOOC Learners' Dropout Behavior," *Appl. Sci.*, vol. 13, no. 2, hal. 1–17, 2023, doi: 10.3390/app13021068.

[3] M. Şahin, "A Comparative Analysis of Dropout Prediction in Massive Open Online Courses," *Arab. J. Sci. Eng.*, vol. 46, no. 2, hal. 1845–1861, 2021, doi: 10.1007/s13369-020-05127-9.

[4] F. Agrusti, G. Bonavolontà, dan M. Mezzini, "University dropout prediction through educational data mining techniques: A systematic review," *J. E-Learning Knowl. Soc.*, vol. 15, no. 3, hal. 161–182, 2019, doi: 10.20368/1971-8829/1135017.

[5] J. Chen, J. Feng, X. Sun, N. Wu, Z. Yang, dan S. Chen, "MOOC Dropout Prediction Using a Hybrid Algorithm Based on Decision Tree and Extreme Learning Machine," *Math. Probl. Eng.*, vol. 2019, 2019, doi: 10.1155/2019/8404653.

[6] K. Coussement, M. Phan, A. De Caigny, D. F. Benoit, dan A. Raes, "Predicting student dropout in subscription-based online learning environments: The beneficial impact of the logit leaf model," *Decis. Support Syst.*, vol. 135, no. December 2019, hal. 113325, 2020, doi: 10.1016/j.dss.2020.113325.

[7] A. Alamri *et al.*, "Predicting MOOCs dropout using only two easily obtainable features from the first week's activities," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11528 LNCS, hal. 163–173, 2019, doi: 10.1007/978-3-030-22244-4_20.

[8] P. M. Moreno-Marcos, P. J. Muñoz-Merino, J. Maldonado-Mahauad, M. Pérez-Sanagustín, C. Alario-Hoyos, dan C. Delgado Kloos, "Temporal analysis for dropout prediction using self-regulated learning strategies in self-paced MOOCs," *Comput. Educ.*, vol. 145, hal. 103728, 2020, doi: 10.1016/j.compedu.2019.103728.

[9] C. Jin, "MOOC student dropout prediction model based on learning behaviour features and parameter optimization," *Interact. Learn. Environ.*, vol. 31, no. 2, hal. 714–732, 2020, doi: 10.1080/10494820.2020.1802300.

[10] L. J. Rodríguez-Muñiz, A. B. Bernardo, M. Esteban, dan I. Díaz, "Dropout and transfer paths: What are the risky profiles when analyzing university persistence with machine learning techniques?" *PLoS One*, vol. 14, no. 6, hal. 1–20, 2019, doi: 10.1371/journal.pone.0218796.

[11] Y. Mourdi, M. Sadgal, H. El Kabtane, dan H. E. A. El Abdallaoui, "A Multi-Layers Perceptron for predicting weekly learner commitment in MOOCs," *J. Phys. Conf. Ser.*, vol. 1743, no. 1, 2021, doi: 10.1088/1742-6596/1743/1/012027.

[12] J. Swacha dan K. Muszyńska, "Predicting Dropout in Programming MOOCs through Demographic Insights," *Electron.*, vol. 12, no. 22, 2023, doi: 10.3390/electronics12224674.

[13] J. Niyogisubizo, L. Liao, E. Nziyumva, E. Murwanashyaka, dan P. C. Nshimyumukiza, "Predicting student's dropout in university classes using two-layer ensemble machine learning approach: A novel stacked generalization," *Comput. Educ. Artif. Intell.*, vol. 3, no. November 2021, hal. 100066, 2022, doi: 10.1016/j.caeai.2022.100066.

[14] J. Melvin dan A. Soraya, "Analisis Perbandingan Algoritma XGBoost dan Algoritma Random Forest Ensemble Learning pada Klasifikasi Keputusan Kredit," *J. Ris. Rumpun Mat. dan Ilmu Pengetah. Alam*, vol. 2, no. 2, hal. 87–103, 2023.

[15] G. Kumar, A. Singh, dan A. Sharma, "Ensemble Deep Learning Network Model for Dropout Prediction in MOOCs," *Int. J. Electr. Comput. Eng. Syst.*, vol. 14, no. 2, hal. 187–196, 2023, doi: 10.32985/ijeces.14.2.8.

[16] Z. Shou, P. Chen, H. Wen, J. Liu, dan H. Zhang, "MOOC Dropout Prediction Based on Multidimensional Time-Series Data," *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/2213292.

[17] Q. Fu, Z. Gao, J. Zhou, dan Y. Zheng, "CLSA: A novel deep learning model for MOOC dropout prediction," *Comput. Electr. Eng.*, vol. 94, no. July, hal. 107315, 2021, doi: 10.1016/j.compeleceng.2021.107315.

[18] M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, dan Z. Siddique, "Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance," hal. 5–9, 2021.

[19] A. Ambarwari, Q. Jafar Adrian, dan Y. Herdiyeni, "Analysis of the Effect of Data Scaling on the Performance of the Machine Learning Algorithm for Plant Identification," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 1, hal. 117–122, 2020, doi: 10.29207/resti.v4i1.1517.

[20] S. Nithya dan S. Umarani, "MOOC Dropout Prediction using FIAR-ANN Model based on Learner Behavioral Features," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 9, hal. 607–617, 2022, doi: 10.14569/IJACSA.2022.0130972.

[21] A. Putri *et al.*, "Komparasi Algoritma K-NN, Naive Bayes dan SVM untuk Prediksi Kelulusan Mahasiswa Tingkat Akhir," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 3, no. 1, hal. 20–26, 2023, doi: 10.57152/malcom.v3i1.610.

[22] Z. Saputra, D. Sartika, dan M. H. Irfani, "Prediksi Calon Mahasiswa Penerima KIP Pada Universitas Indo Global Mandiri menggunakan Algoritma Decision Tree," vol. 4, no. 3, hal. 231–240, 2024.

[23] Reza Fauzy, Riki Winanjaya, dan Susiani, "Analisis Tingkat

Kepuasan Pelanggan dengan Menerapkan Algoritma C4.5," *Bull. Comput. Sci. Res.*, vol. 2, no. 2, hal. 41–46, 2022, doi: 10.47065/bulletincsr.v2i2.162.

[24] H. S. Park dan S. J. Yoo, "Early Dropout Prediction in Online Learning of University using Machine Learning," *Int. J. Informatics Vis.*, vol. 5, no. 4, hal. 347–353, 2021, doi: 10.30630/JOIV.5.4.732.

[25] Y. Zheng, Z. Gao, Y. Wang, dan Q. Fu, "MOOC Dropout Prediction Using FWTS-CNN Model Based on Fused Feature Weighting and Time Series," *IEEE Access*, vol. 8, hal. 225324–225335, 2020, doi: 10.1109/ACCESS.2020.3045157.

[26] B. Huang dan C. Wang, "Research on Data Analysis of Efficient Innovation and Entrepreneurship Practice Teaching Based on LightGBM Classification Algorithm," *Int. J. Comput. Intell. Syst.*, vol. 16, no. 1, 2023, doi: 10.1007/s44196-023-00324-4.

[27] D.- Andriansyah dan E. W. Fridayanthie, "Optimization of Support Vector Machine and XGBoost Methods Using Feature Selection to Improve Classification Performance," *J. Informatics Telecommun. Eng.*, vol. 6, no. 2, hal. 484–493, 2023, doi: 10.31289/jite.v6i2.8373.

[28] W. Wunnasri, P. Musikawan, dan C. So-In, "A Two-Phase Ensemble-Based Method for Predicting Learners' Grade in MOOCs," *Appl. Sci.*, vol. 13, no. 3, 2023, doi: 10.3390/app13031492.

[29] S. Y. J. Prasetyo, Y. B. Christianto, dan K. D. Hartomo, "Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan Machine Learning di Wilayah Kabupaten Boyolali dan Purworejo," *Indones. J. Model. Comput.*, vol. 2, no. 2, hal. 25–36, 2019, [Daring]. Tersedia pada: https://ejournal.uksw.edu/icm/article/view/2954

[30] I. M. . Karo, "Implementasi Metode XGBoost dan Feature Importance untuk Klasifikasi pada Kebakaran Hutan dan Lahan," *J. Softw. Eng. Inf. Commun. Technol.*, vol. 1, no. 1, hal. 11–18, 2020.

[31] A. N. G. Ji dan D. Levinson, "Injury Severity Prediction From Two-Vehicle Crash," vol. 1, no. April, hal. 217–226, 2020.

[32] K. Kristiawan dan A. Widjaja, "Perbandingan Algoritma Machine Learning dalam Menilai Sebuah Lokasi Toko Ritel," *J. Tek. Inform. dan Sist. Inf.*, vol. 7, no. 1, hal. 35–46, 2021, doi: 10.28932/jutisi.v7i1.3182.

