Balancing Fined-tuned Machine Learning Models between Continuous and Discrete Variables - A Comprehensive Analysis using Educational Data

Efthyvoulos Drousiotis[1], Panagiotis Pentaliotis[1], Lei Shi[2], and Alexandra I. Cristea[2]

[1] Department of Electrical Engineering Electronics, University of Liverpool, Liverpool, UK {e.drousiotis, p.pentaliotis}@liverpool.ac.uk
[2] Department of Computer Science, Durham University, Durham, UK {lei.shi, alexandra.i.cristea}@durham.ac.uk

Abstract. Along with the exponential increase of students enrolling in MOOCs [26] arises the problem of a high student dropout rate. Researchers worldwide are interested in predicting whether students will drop out of MOOCs to prevent it. This study explores and improves ways of handling notoriously challenging continuous variables datasets, to predict dropout. Importantly, we propose a fair comparison methodology: unlike prior studies and, for the first time, when comparing various models, we use algorithms with the dataset they are intended for, thus 'like for like.' We use a time-series dataset with algorithms suited for time-series, and a converted discrete-variables dataset, through feature engineering, with algorithms known to handle discrete variables well. Moreover, in terms of predictive ability, we examine the importance of finding the optimal hyperparameters for our algorithms, in combination with the most effective pre-processing techniques for the data. We show that these much lighter discrete models outperform the time-series models, enabling faster training and testing. This result also holds over fine-tuning of pre-processing and hyperparameter optimisation.

Keywords: Neural Networks · Tree-based Algorithms · Educational Data Mining · Feature Engineering · MOOCs

1 Introduction

With the rapid development of the Internet and in combination with the growing training demands, the education industry has changed the way it operates. Massive Open Online Course (MOOC) platforms were introduced to the world, which has attracted millions of users [26]. This led to a revolution of big data in learning, with more resources and anonymised datasets for exploration. Over the years, an undeniable challenge in online learning became to find ways to reduce and predict students' dropout rates, which fall roughly at 77%-87% . Many studies have been conducted to explore learning behavioural patterns, through statistical modelling and machine learning, towards predicting students' dropout [11, 19].

Nonetheless, the majority of the studies, such as [30, 31], used the same dataset and variables to implement predictive models, without taking into consideration the type of variables each model uses for maximising its performance. For example, [30] trained a time-series, Long Short-Term Memory (LSTM) model, using the same dataset that was used to train other non-time series machine learning models, including Logistic Regression, Random Forest, and Gradient Boosting Decision Tree (GBDT). The results showed that time-series models, such as LSTM, outperformed other machine learning models (i.e., Linear Regression and Decision Tree), and achieved higher accuracy, precision and recall when compared using data from their 'natural' environment (continuous/time-series variables). However, we argue that previous methods did not consider the functionality of the algorithms and how they could perform best, according to their nature. Some very preliminary previous research [10] has hinted that it may be a good practice to use sequential time-series 'as is', or first convert the dataset into discrete-variables, for obtaining enhanced metrics (precision, recall, f1-score, accuracy) on predicting students' dropout, when the models would be tuned, and the datasets would be pre-processed. The current paper aims to determine if traditional fine-tuning and optimisation methods can change this, or if the conversion into discrete variables is as robust as we assumed. We use the same application of predicting 'completers' and 'non-completers' with the same dataset to analyse this. We examine thus the following research question:

Can pre-processing, fine-tuning and hyperparameter optimisation change the balance between using time-series 'as is', or converting them into discrete variables?

The main contributions of this study are thus to perform, for the first time, a wide-scale analysis, showing, firstly, that discrete-feature methods outperform sequential time-series methods, on both discrete and sequential datasets. Secondly, we show that this result is further consistent, when performing model hyperparameter optimisations and optimal feature engineering. Our results, furthermore, outperform all other studies using the same dataset [24, 15, 21, 20, 17] in predicting dropouts. Moreover, as we compare several approaches, our work also shows that methods of capturing uncertainty outperform the others. This supports the more generic approach to converting the dataset, whenever possible, into the appropriate formats (in our case, time-series into discrete), which helps a different kind of predictive model than the default applied in previous studies, achieving faster training, testing, and predicting, as well as higher predictive accuracy and in general better performance compared to using multi-layer neural networks.

2 Related Work

Learning Analytics (LA) is the process of analysing and reporting multiple learners' data to understand and optimise their performance and the learning environment. Many recent studies focused on classifying students into 'completers' (i.e. students who completed the course) and 'non-completers'. Some of them [19, 2] used traditional machine learning algorithms (e.g. Decision Tree, Logistic

Regression, Random Forest), while others, such as [12, 14], used more advanced algorithms (e.g. Neural Networks).

A few recent studies also [30, 31] utilised both traditional machine learning algorithms and more advanced ones (Neural Networks). These studies [30, 31] used the same dataset to train both Neural Networks and machine learning models (time-series), which showed that Neural Networks outperformed the other machine learning techniques.

However, Tensorflow[3]suggested that to train an LSTM, it is best to use a time-series dataset, while [13] suggested using discrete variables to train a tree-based algorithm (either categorical or continuous variables).

Moreover, some works [16, 32] suggested that artificial Recurrent Neural Networks (RNN) with memory, such as Long-Short-Term-Memory (LSTM), are generally considered as superior models for time-series tasks, due to their nature – the way they operate and handle data. On the other hand, [28] suggested that traditional machine learning algorithms, such as Logistic Regression, Random Forest and GBDT, produce better results with discrete-variable data. The only study we could find that compared four benchmark models with their intended datasets [10], lacked, however, a thorough examination of possible outcomes after pre-processing and hyperparameter optimisation.

In our case, we convert the time-series data, through feature engineering, into discrete variables, and train each model on the type of data it can process best.

Furthermore, some current works used a combination of different types of data, when those were available, in order to obtain higher accuracy with the LSTM model. For example, [22, 23] examined a combination of time-series data and other discrete data, which included features such as first quiz results, the number of playbacks of a video. The primary key aspect of our methodology is that we do not use any other data than video interactions, to *fairly* analyse different formats of this data for different algorithms.

Several studies, such as [24, 15, 17], conducted student dropout prediction (from MOOCs) on the same dataset that we use in this paper. Some of them [21, 24, 20] did not perform any hyperparameter optimisation on their predictive models, but they pre-processed their data; whilst others, such as [15], performed a basic feature engineering and hyperparameters tuning. Specifically, [24] used AME, a meta-embedding technique, through which they derived the optimal embedding for each sequence of object embedding vectors, and a temporal classification, which modelled the temporal nature of the data over multiple days. [20] filtered the logs, which contained unrelated events from users, rows with missing values, and columns with no helpful information. [17] highlighted the unbalanced nature of the dataset, and to achieve better classifier performance, they applied the synthetic minority oversampling technique known as SMOTE on the training set. However, generating synthetic examples may increase the overlapping of classes and introduce additional noise, while the initial distribution of the dataset is compromised, such that it no longer corresponds to real-world data. Finally, [15] performed thorough hyperparameter tuning, while the data

[3] https://www.tensorflow.org/tutorials/structured$_data/time_series$

pre-processing techniques used were not to explore or enhance the data, but just to make the data compatible with their predictive model. As mentioned in Section 1, we explore further the results of [10] with optimised hyperparameters, and optimal pre-processing techniques.

Unlike prior work, this study shows, in a comprehensive way, that it is beneficial to convert time-series data into discrete variables, when testing several machine learning algorithms. Moreover, to ensure a fair comparison, we are using the same data for several testing cases, and we explore in depth the algorithmic performance, by hyperparameter optimising all the machine learning algorithms used.

3 Method

3.1 Dataset and Data Preparation

The dataset used in this study is comprised of 300,000 interactions performed by 2,000 unique students that were registered on XuetangX[4] (launched in October 2013, one of the largest MOOC platforms in the world). The dataset contains two modules delivered in 2014 in a Self-Paced Mode (SPM), where a student can have a more flexible schedule and study during the hours that suit them the best. In order to make a fair, controlled comparison and strengthen our claims, we trained all the models (with and without tuned hyperparameters, with and without pre-processed data) with the time-series dataset and discrete variable dataset. In particular, we converted the time-series dataset into a discrete-variables dataset. In the time-series dataset, the input variables include the type of actions and the time each action was performed for all the 300,000 interaction entries. For constructing the discrete-variables dataset, we used the time-series dataset and counted the number of unique actions for each student. The table is populated by the ID, the Truth (pass or dropout) and the 14 unique types of actions the students performed, namely, *click courseware, click forum, click info, click progress, click about, close courseware, create comment, load video, create thread, pause video, play video, problem get, seek video*, and *stop video*. In total, there are 14 unique types of actions, so we engineered 14 features for 14 input variables for our predictive models. Considering the LSTM model's pre-processing in preparation of the dataset, the actions performed by each student were sequentially grouped, according to the time they were performed. Thus, the essence of the time-series was preserved while still considering the unique actions performed. Afterwards, the actions were transformed into a sequence of binary numbers (see Table 1), to retain the categorical (nominal, i.e. no intrinsic ordering to the categories) nature of the actions.

3.2 Data Pre-Processing / Features Engineering

Our feature engineering techniques aimed not to change the data distribution, but only to feed the data into the models in the most efficient way.

[4] http://moocdata.cn/data/user-activity

Table 1. Time-Series Dataset

ID	Action	Time	Truth [5]
...	...	...	...
561867	pause_video	2015-10-25T10:52:06	0
561867	play_video	2015-10-25T10:52:09	0
561867	pause_video	2015-10-25T10:58:42	0
1368125	click_about	2015-10-05T15:43:55	1
1368125	click_info	2015-10-05T15:45:53	1
708122	pause_video	2015-10-04T21:41:30	1
708122	play_video	2015-10-04T21:24:40	1
...	...	...	...

Table 2. Sample distribution per classification category

	Dropout	Continuing Study
Sample Number	76470	20059
Percentage of Sample	79.22%	20.78%

For the tree-based algorithms, we adopted the Term Frequency-Inverted Document Frequency (TF-IDF) technique to feed the data into our models. TF-IDF is a statistical measure that estimates how important and relevant a word is to a document. Here, we examined the importance of each action compared to the total number of actions. Generally, an action's importance increases with the number of times an action appears in the current input, which, at the same time, is counterbalanced by the frequency of that action in general.

LSTMs use sequences of numerical values, so we transformed the sequences of actions performed by each student, which are identified by words, into numbers. Specifically, we gathered all the student's actions and ordered them according to their timestamps. By doing so, we retained the time-series nature of the action sequence. We then created a dictionary of the unique words from the actions and encoded them using One-Hot Encoding. One-Hot Encoding morphs the unique numbers into sequences of 0s and 1s, which maintain the sequences' categorical nature for the LSTM.

3.3 Building the Models

We implemented an LSTM model and several tree-based machine learning models, including Decision Tree, Random Forest, and BART. In this section, we introduce how these models were built.

LSTMs can process long sequences of data (in our case, sequences of actions). In the current study, we used LSTMs to train on the temporal sequence of actions performed by the students. For the discrete variables dataset training, we considered that all the inputs were from a single timestamp, meaning that all the actions had been executed at once (i.e., no temporal order) compared

to the continuous variables dataset training, where each action had a different timestamp, adding a subtle continuous/temporal feature to the data.

A single standard LSTM unit is composed of a cell vector (c_t) eq. (3), a hidden vector (h_t) eq. (5), an input gate (i_t) eq. (1), an output gate (o_t) eq. (4) and a forget gate (f_t) eq. (2). A cell remembers values over time intervals ($t-1$, t); and the three gates regulate the flow of information, by computing a series of functions for the cell vector and the hidden vector [18].

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c \tag{3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{4}$$

$$h_t = o_t tanh(c_t) \tag{5}$$

Decision Tree is a non-parametric, supervised machine learning model, which learns simple decision rules inferred from the data variables. It can be used for both classification and regression tasks. Specifically, we used the CART model [6], which traverses the binary tree given a new input record, where the tree is trained by a greedy algorithm on the training data, to pick splits in the tree. The main reason of using CART is the algorithmic transparency provided, as it being proven to be a trustworthy baseline in prior researches[29].

Random Forest(RF) [5] is a supervised machine learning method formed on ensemble learning (the combination of different types of algorithms, or the same algorithms applied many times, to create a more precise predictive model). An RF receives the prediction from each sub-tree and chooses the solution that is in majority, by voting. Forests and specifically RF according to [9], tend to outperform the rest of the classification algorithms (based on a large-scale comparison of 179 classification algorithms emerging from 17 learning families over 121 datasets).

BART is a Bayesian version of a tree ensemble model, where the estimation is given by a sum of Bayesian CART trees. More information can be found in [7]. Bayesian methods are known to be better at modelling uncertainty[8], which addresses part of our aim in comparing these methods.

For the tree-based models we performed hyperparameter optimisation through Grid Search and Random Search techniques, as they are widely used in the state of the art research [3]. Moreover, we tested the model with 10-Fold Cross Validation, which is a widely used technique to ensure full usage of all available data.

The hyperparameter optimisation for the LSTM was performed with the help of an online open source application 'Weights & Biases'[6] [4]. The sweep method used to tune the model was 'Bayes' [27]. The 'Bayes' method uses a Gaussian Process (GP) Expected Improvement Markov Chain Monte Carlo (GP EI MCMC) technique to calculate the posterior distribution from a prior, and

[6] https://wandb.ai/site

the 'expected improvement' of a parameter. A Gaussian Process distribution on prior functions, is chosen to express assumptions about the function being optimised [25][27]. The 'expected improvement' is the acquisition function (eq. (6)), used to construct a utility function from the model posterior for our Bayesian optimisation, i.e. the main deciding factor for the MCMC [27]. If that parameter improves the F1-score of training, as we have requested from the MCMC search to monitor, the parameters are tuned, respectively.

$$a_{EI}(x; \{x_n, y_n\}, \theta) = \sigma(x; \{x_n, y_n\}, \theta)(\gamma(x)\Phi(\gamma(x)) + N(\gamma(x); 0, 1)) \quad (6)$$

where:

$a_{EI}(x; \{x_n, y_n\}, \theta)$ is the acquisition function that depends on the previous observations, and the GP hyperparameters;

$\Phi(\cdot)$ is the cumulative distribution function of the standard normal;

$N(\gamma(x); 0, 1)$ is the prior distribution with noise $(0, 1)$;

$\sigma(x; \{x_n, y_n\}, \theta)$ is the predictive variance function, and:

$$\gamma(x) = \frac{f(x_{best}) - \mu(x; \{x_n, y_n\}, \theta)}{\sigma(x; \{x_n, y_n\}, \theta)} \quad (7)$$

where:

$\mu(x; \{x_n, y_n\}, \theta)$ is the predictive mean function;

$x_{best} = argmin_{xn} f(x_n)$ is the best current value.

Each training, with continuous or discrete variables, was performed 100 times. We used uniform distributions, as we had no prior belief information for the hyperparameters. The priors were either uniform or integer uniform, depending on the parameter checked. After monitoring tuning, the parameters providing the highest F1-score was chosen from 100 iterations. Then, to obtain the optimal hyperparameters, we run the resulting optimal algorithm 10-fold, to obtain the median ROC curve and the median testing results.

The Random Forest and Decision Tree models were implemented using the scikit-learn version 1.0.1 in Python[7]. The BART model was implemented using the BART package in R[8]. The LSTM model was implemented using Keras version 2.2.5[9].

The overall purpose of our methodological setting was to:

- Compare the two datasets in their primitive forms, without any data preprocessing or hyperparameter optimisation (sequential time-series and discrete).
- Train and test the models using the two datasets applying hyperparameters optimisation and feature engineering techniques (sequential time-series and discrete).

[7] https://pypi.org/project/scikit-learn/

[8] https://cran.r-project.org/web/packages/BART/BART.pdf

[9] https://keras.io/api/layers/recurrent$_l ayers/lstm/$

– Compare and contrast all the above to find out if and when we should optimise model parameters and apply data pre-processing methods.

To evaluate our predictive models' performance, we utilised standard, comprehensive metrics: Precision, Recall, F-1 score and Accuracy. Moreover, we produced a ROC curve (receiver operating characteristic curve) for each model, i.e. the graph showing the performance of the classification models at all classification thresholds. This curve plots two parameters, the True Positive Rate and the False Positive Rate. This allowed us to also explore the Area under the ROC Curve (i.e., AUC) measure.

This way, we ensured a thorough, *fair* comparison of the algorithms under study.

4 Results and Discussions

We present the results of our study in Table 3, comprising of three tree-based models (Decision Tree, Random Forest, BART) and an LSTM model, for 4 test cases (Q1: Discrete dataset without hyperparameter optimisation and feature engineering, Q2: Discrete dataset with hyperparameter optimisation and feature engineering, Q3: Continuous dataset without hyperparameter optimisation and feature engineering, Q4: Continuous dataset with hyperparameter optimisation and feature engineering).

Firstly, we observed that BART is the most robust model - it maintains its high predictive accuracy for all 4 test cases (Table 3). Specifically, BART outperforms all the other models - and did not overfit in any of the test cases, achieving accuracies from 80% to 92% (see Table 3).

Secondly, for the LSTM for the 4 test cases, we observed overfitting (Table 3), which was caused by training on a (relatively) moderate amount of data. Decision Tree overfitted on the 3 out of 4 test cases (see Table 3), while hyperparameters optimisation and feature engineering did not enhance the model performance as expected. Random Forest overfitted as well on 2 out of 4 test cases, which indicates that it did not benefit from either feature engineering or data transformation. For all 4 test cases, BART showed an impressive ability to determine whether a student would pass or drop out, while hyperparameter optimisations and feature engineering improved its performance.

Thirdly, we used the Area Under the Curve (AUC) (see Figs. 1-4) as a criterion to measure the models' ability to discriminate the test cases. The closer the ROC curve to the upper left corner, the more efficient the test was. By taking into consideration the results (see Table 3) and comparing them with the ROC curves, we validated the fact that BART is the most consistent model, as it was not affected by neither the imbalanced nature of the data nor the low level of hyperparameters optimisation and it has an improved ability to discriminate the test values in comparison with the other four models (Decision Tree, Random Forest, BART and LSTM).

Fourthly, we observed the improved performance of Decision Tree and Random Forest models when they were trained on the dataset (discrete data) they

Table 3. Results: Comparison of Decision Tree, Random Forest, BART on discrete and continuous data, with/out hyperparameter optimisation (4 test cases)

		DT	RF	BART	LSTM
Precision	Q1	0.64	0.77	**0.87**	0.40
	Q2	0.74	0.80	**0.88**	0.81
	Q3	0.60	0.67	**0.81**	0.41
	Q4	0.74	0.72	**0.87**	0.52
Recall	Q1	0.66	0.69	**0.96**	0.34
	Q2	0.63	0.71	**0.97**	0.66
	Q3	0.59	0.58	**0.98**	0.34
	Q4	0.65	0.66	**0.98**	0.51
F1	Q1	0.65	0.71	**0.91**	0.29
	Q2	0.66	0.75	**0.92**	0.68
	Q3	0.59	0.60	**0.89**	0.29
	Q4	0.68	0.68	**0.92**	0.50
Accuracy	Q1	0.77	0.85	**0.90**	0.32
	Q2	0.83	0.88	**0.92**	0.82
	Q3	0.82	0.86	**0.88**	0.32
	Q4	0.88	0.89	**0.89**	0.80

are suited for, as they overfitted when trained on the 'unsuitable' dataset (sequential data). The LSTM model did not perform as well as the tree-based models, and especially not as expected for the continuous variables. That is possibly because LSTMs are known to require a large amount of data in order to be efficiently trained [1].

Our results suggest that, whenever possible, it would be beneficial to convert the time-series dataset into a discrete variables dataset and apply Bayesian methods, such as BART, as it is highly likely to produce better performance, especially when the time-series datasets are not populated enough.

Moreover, our results highlight the necessity of always finding the best hyperparameters for the models based on the data they are trained on, and the most efficient and effective data pre-processing techniques, as they can dramatically improve the models' performances and prevent overfitting. However, converting time-series datasets into discrete datasets can often be time-consuming.

5 Conclusions, Limitations and Future work

In summary, this paper presents the results of a comparison study with 4 test cases, swapping continuous and discrete datasets, as well as training with/without hyperparameter optimisation, on four different state-of-the-art algorithms (Decision Tree, LSTM, BART, Random Forest).

Our results conclude that researchers should transform data into suitable forms when feasible, and they should always try to identify the optimal data pre-processing techniques, as this can improve model performance. We have

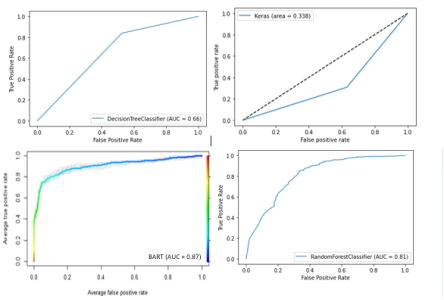

Fig. 1. Discrete data: No Pre-Processing, No Hyperparameter Optimisation

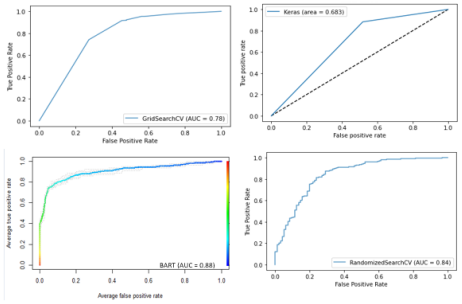

Fig. 2. Discrete data: Hyperparameter Optimisation, Feature Engineering

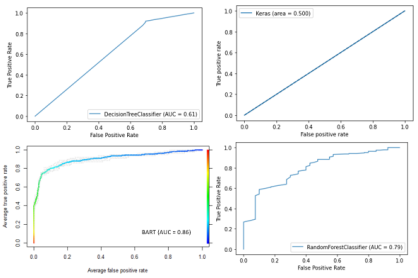

Fig. 3. Continuous data: No Pre-Processing, No Hyperpar. Optimisation

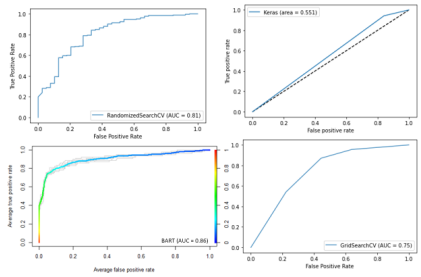

Fig. 4. Continuous data: Hyperparameter Optimisation, Features Engineering

(**For all ROC Curves above:** Upper left - Decision Tree, Upper right - LSTM, Bottom Left - BART, Bottom Right - Random Forest)

shown that this process assists different types of predictive models to obtain higher performance and enhanced learning ability. Different from other studies, we propose, for the first time, a *fair comparison*; for this, we trained our predictive models not only based on the data type they are indicated for (time-series data for LSTMs and discrete data for tree-based models) but also with all the test cases, for obtaining unbiased results. It is also worth mentioning that we have noted that BART is the only model which did not overfit in any of the 4 test cases, rendering it the ideal model for producing not only benchmarks but also high quality results. The other 3 models (Decision Tree, Random Forest and LSTM) were overfitted in some of the cases, indicating that we should be very cautious when trying to improve the performance of our predictive models.

The main limitations of this study are those related to the data. We used only one dataset, as being the largest available currently. However, more and larger datasets would be useful for further comparisons, as LSTM models especially tend to perform better when being trained on more data. Moreover, de-

mographic or personal information of each student (unavailable in the dataset) might provide the models with more meaningful connections to variables for the classifications and thus allow better performance. It is also important to note that, as some students might prefer to download videos and watch them locally, not all the actions (i.e. interacting with a video player such as 'play' and 'pause') could be fully captured through the online learning platforms.

For future work, we plan to add more predictive models for comparison, including Bayesian and non-Bayesian models, to validate and strengthen our conclusion on the Bayesian models being less overfit-prone. Moreover, to explore further the capabilities of the LSTM we could consider the time intervals of the actions.

References

1. Adadi, A.: A survey on data-efficient algorithms in big data era. In: Journal of Big Data (2021), https://link.springer.com/article/10.1186/s40537-021-00419-9citeas
2. Alamri, A., Alshehri, M., Cristea, A., Pereira, F.D., Oliveira, E., Shi, L., Stewart, C.: Predicting MOOCs Dropout Using Only Two Easily Obtainable Features from the First Week's Activities. In: Intelligent Tutoring Systems, vol. 11528, pp. 163–173. Springer International Publishing, Cham (2019), http://link.springer.com/10.1007/978-3-030-22244-4 20
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
4. Biewald, L.: Experiment tracking with weights and biases (2020), software available from wandb.com
5. Breiman, L.: Random forests. Machine Learning **45**, 5–32 (2004)
6. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees (1983)
7. Chipman, H.A., George, E.I., McCulloch, R.E.: BART: Bayesian additive regression trees (Oct 2010). https://doi.org/10.1214/09-AOAS285
8. Clyde, M., George, E.I.: Model uncertainty. Statistical science **19**(1), 81–94 (2004)
9. Delgado, M.F., Cernadas, E., Barro, S., Amorim, D.G.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**, 3133–3181 (2014)
10. Drousiotis, E., Pentaliotis, P., Shi, L., Cristea, A.I.: Capturing fairness and uncertainty in student dropout prediction–a comparison study. In: International Conference on Artificial Intelligence in Education. pp. 139–144. Springer (2021)
11. Drousiotis, E., Shi, L., Maskell, S.: Early predictor for student success based on behavioural and demographical indicators. In: International Conference on Intelligent Tutoring Systems. pp. 161–172. Springer (2021)
12. Fei, M., Yeung, D.: Temporal Models for Predicting Student Dropout in Massive Open Online Courses. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW). pp. 256–263 (Nov 2015), iSSN: 2375-9259
13. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Proceedings of the Sixteenth International Conference on Machine Learning. p. 124–133. ICML '99, Morgan Kaufmann Publishers Inc., San Francisco (1999)
14. Gardner, J., Yang, Y.: Modeling and Experimental Design for MOOC Dropout Prediction: A Replication Perspective. Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019) p. 10 (2019)

15. Goel, Y., Goyal, R.: On the effectiveness of self-training in mooc dropout prediction. In: Open Computer Science. vol. 10, pp. 246–258 (2020)
16. Hochreiter, S., Yoshua, Informatik, F.F., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies (2001)
17. Hong, B., Wei, Z., Yang, Y.: Discovering learning behavior patterns to predict dropout in mooc. In: 2017 12th International Conference on Computer Science and Education (ICCSE). pp. 700–704 (2017)
18. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR **abs/1508.01991** (2015), http://arxiv.org/abs/1508.01991
19. Jin, C.: MOOC student dropout prediction model based on learning behavior features and parameter optimization. Interactive Learning Environments pp. 1–19 (Aug 2020). https://doi.org/10.1080/10494820.2020.1802300
20. Liang, J., Li, C., Zheng, L.: Machine learning application in moocs: Dropout prediction. In: 2016 11th International Conference on Computer Science Education (ICCSE). pp. 52–57 (2016). https://doi.org/10.1109/ICCSE.2016.7581554
21. Liang, J., Yang, J., Wu, Y., Li, C., Zheng, L.: Big data application in education: Dropout prediction in edx moocs. In: 2016 IEEE Second International Conference on Multimedia Big Data (BigMM). pp. 440–443 (2016)
22. Liu, Z., Xiong, F., Zou, K., Wang, H.: Predicting Learning Status in MOOCs using LSTM (Aug 2018)
23. Mubarak, A.A., Cao, H., Ahmed, S.A.: Predictive learning analytics using deep learning model in MOOCs' courses videos (2021)
24. Pulikottil, S.C., Gupta, M.: Onet – a temporal meta embedding network for mooc dropout prediction. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 5209–5217 (2020). https://doi.org/10.1109/BigData50022.2020.9378001
25. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
26. Rehfeldt, R.A., Jung, H.L., Aguirre, A., Nichols, J.L., Root, W.B.: Beginning the Dialogue on the e-Transformation: Behavior Analysis' First Massive Open Online Course (MOOC). Behavior Analysis in Practice **9**(1), 3–13 (Jan 2016)
27. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2. p. 2951–2959. NIPS'12 (2012)
28. SONG, Y.y., LU, Y.: Decision tree methods: applications for classification and prediction. Shanghai Archives of Psychiatry **27**(2), 130–135 (Apr 2015)
29. Strecht, P., Cruz, L., Soares, C., Mendes-Moreira, J., et al.: A comparative study of classification and regression algorithms for modelling students' academic performance. International Educational Data Mining Society (2015)
30. Tang, C., Ouyang, Y., Rong, W., Zhang, J., Xiong, Z.: Time Series Model for Predicting Dropout in Massive Open Online Courses. In: Artificial Intelligence in Education. pp. 353–357. Cham (2018)
31. Wang, L., Wang, H.: Learning Behavior Analysis and Dropout Rate Prediction Based on MOOCs Data. In: 2019 10th International Conference on Information Technology in Medicine and Education (ITME). pp. 419–423 (Aug 2019)
32. Zhang, X., Liang, X., Zhiyuli, A., Zhang, S., Xu, R., Wu, B.: AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction. IOP **569**, 052037 (Aug 2019), publisher: IOP Publishing