Stacking Classifiers for Early Detection of Students at Risk

Eitel J. M. Lauría[1], Edward Presutti[2], Maria Kapogiannis[2] and Anuya Kamath[2]
[1]*School of Computer Science & Mathematics, Marist College, Poughkeepsie, NY, U.S.A.*
[2]*Data Science & Analytics Group, Marist College, Poughkeepsie, NY, U.S.A.*

Keywords: Early Detection, Stacked Ensembles, Learning Management Systems, Student Information Systems, Predictive Modeling, Supervised Learning.

Abstract: A stacked ensemble is a machine learning method that involves training a second stage learner to find the optimal combination of a collection of based learners. This paper provides a methodology to create a stacked ensemble of classifiers to perform early detection of academically at-risk students and shows how to organize the data for training and testing at each stage of the stacked ensemble architecture. Experimental tests are carried out using college-wide data, to demonstrate how the stack can be used for prediction.

1 INTRODUCTION

In the last decade, a number of research projects and initiatives have flourished that tackle the issue of monitoring student academic performance (Pistilli and Arnold, 2010; Romero et al., 2008; Smith et al., 2012). Generally, researchers use machine learning algorithms (e.g. linear models, Bayesian learners, maximum margin algorithms, and decision trees) to develop predictive models that try to identify academically at-risk students.

Our own work has taken a similar path (Jayaprakash et al., 2014). The goal of our predictive modeling framework has been to detect, relatively early in the semester -three to six weeks into the semester, considering fifteen-week semesters-, those students who are experiencing academic difficulty during the course, using student data. This task is re-expressed as a binary classification process with the purpose of discriminating between students in good standing and academically at-risk students. The predictive modeling framework derives its input data from student academic records, enriched with student demographic and aptitude data, and data collected from the learning management system (LMS). The LMS data consists of student interaction with the LMS, as well as all gradable events stored in the LMS gradebook tool, such as assignment grades.

The outcomes at our institution and in several other colleges and universities where our predictive modeling framework was applied have been very promising (Lauría et al., 2016), but there is still considerable room for improvement. There are several strategies that we are actively pursuing to improve these performance metrics: a) develop better predictors from the current data, through feature engineering; b) include new predictors by considering additional data sources (e.g. extract student engagement metrics from social network sources); c) apply enhanced machine learning techniques that can provide more accurate and stable predictions.

This paper tackles the latter strategy, introducing a stacked ensemble architecture to build early detection models. This machine learning method is mostly absent and minimally referenced in the learning analytics literature. Hence, the paper makes two relevant contributions: 1) it provides a methodology for building a stacked ensemble architecture learnt from data; 2) it provides proof of concept of how stacked ensembles can be applied in the context of early detection of academically at-risk students. We first outline the development of stacked ensemble learning. Then we detail the methodology used to build a two-stage stack. We follow with a description of the experiment, including the data, methods, analyses and results of this study. The paper ends with a summary of our conclusions, limitations of the study and pointers to future work.

Lauría, E., Presutti, E., Kapogiannis, M. and Kamath, A.
Stacking Classifiers for Early Detection of Students at Risk.
DOI: 10.5220/0006781203900397
In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018)*, pages 390-397
ISBN: 978-989-758-291-2
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

2 STACKED ENSEMBLE LEARNING

Ensemble methods are not a new concept: in fact, the approach is embedded in most of the state of the art machine learning algorithms (e.g. bagged neural nets, random forests, gradient boosted trees), which use either *bagging* or *boosting* to combine results from a set of learners in order to increase accuracy and reduce the variability of the prediction. Ensemble systems (Polikar, 2006) take the above approach one step further by combining multiple learners trained with different learning algorithms to make predictions, using some voting criterion by the constituent models (e.g. majority voting). Ensembles tend to thrive on diversity: they yield better results if their constituent learning algorithms, and therefore the learners trained with them are substantially different (Lam, 2000).

A different type of ensemble methods, called *stacked ensemble learning (*or more loosely*, stacking or super learning)*, takes a different tack at making predictions, using the predicted values of a group of base learners to feed a second-stage learner, which finds the optimal hypothesis function from the combination of the base learners. Succinctly, deterministic voting is replaced by a statistical learning approach: the predictions of the first stage models become the inputs of the second stage model, which in turn delivers the final predictions of the stack. The second-stage learner, in turn, can be implemented as a single classifier or as an ensemble. Stacked ensemble architectures, which are the focus of this paper, typically perform better than any specific trained models in the stack: the whole is more than the sum of its parts. Stacked ensembles can have several stages, where the predictions of one stage act as input data for the next (this paper describes two-stage stacks). Stacked ensembles were initially introduced by Wolpert (Wolpert, 1992), but it wasn't until 2007 that van der Lan et al (van der Laan et al., 2007) provided proof that a 'super learner' ensemble represents an asymptotically optimal system for learning. Stacked ensembles have gained considerable momentum lately, with these machine learning frameworks performing at the top of the Netflix Prize competition (Sill et al., 2009) as well as the Kaggle competitions (Kaggle Team, 2017). There are two basic approaches used to train stacked ensembles: (i) using cross-fold-validation (the out-of-fold is used to train the first stage -base-models, and their predictions on the test set is used to train the next layer); or (ii) using a holdout set. In this paper we propose a variation of the latter using three independent datasets, which prevents data leakage (the unforeseen addition of information in the training data, which leads to underestimating the models' generalization error), as the first and second stage models use different training data.

3 BUILDING A STACKED ENSEMBLE FOR EARLY DETECTION

3.1 Methodology

Three independent data sets **A, B**, and **C**, extracted through random partitioning from a single data set are used, each of them with a data schema [*X, y*] made up of a vector of predictors *X* and a response variable *y*. The response variable is binary, indicating whether a student is academically at risk. **A** is used for training the first stage**, B** is used for testing the first stage and training the second stage, **C** is used for testing the second stage (more details on the use of the data files follow).

In stack learning (i.e. training and testing) mode:

- **Step(i)** corresponds to Stage1: Training, where k base models $\mathcal{M}_1 \ldots \mathcal{M}_k$ are trained using dataset $\mathbf{A}[X; y]$ and classification algorithms $\mathcal{C}_1 \ldots \mathcal{C}_k$. The notation $_{cv}$ indicates model tuning using cross-validation.
- In **Step(ii)**, corresponding to Stage 1: Testing, trained models $\mathcal{M}_1 \ldots \mathcal{M}_k$ are applied on dataset $\mathbf{B}[X; y]$, and produce k prediction vectors $\hat{y}^B_{1..k}$, and k probability vectors $\hat{p}^B_{1..k}$ associated with the prediction (a measure of the confidence of the prediction). These values make up an $(n_B \times 2 \cdot k)$ matrix $(\hat{Y}_B; \hat{P}_B) \equiv (\hat{y}^B_1 \ldots \hat{y}^B_k; \hat{p}^B_1 \ldots \hat{p}^B_k)$, where n_B is the number of observations in data set **B,** and $2 \cdot k$ is the number of columns made up by predictions and probabilities $(\hat{y}^B_1 \ldots \hat{y}^B_k; \hat{p}^B_1 \ldots \hat{p}^B_k)$.
- In **Step(iii)**, data set $\mathbf{B}[X; y]$ is augmented by adding to it matrix $(\hat{Y}_B; \hat{P}_B) \equiv (\hat{y}^B_1 \ldots \hat{y}^B_k; \hat{p}^B_1 \ldots \hat{p}^B_k)$, resulting in data set $\mathbf{B}^{(aug)}[X; \hat{Y}_B; \hat{P}_B; y]$.
- **Step(iv)** corresponds to Stage 2: Training. In it, the augmented data set $\mathbf{B}^{(aug)}[X; \hat{Y}_B; \hat{P}_B; y]$ is used to train a second stage model $\mathcal{M}_{SS}$ using a

chosen classification algorithm $\mathcal{C}_{SS}$. As before, the notation $\mathbf{B}^{(aug)}[X;\hat{Y}_B;\hat{P}_B;y] \underline{cv}\ \mathcal{C}_{SS}$ refers to the use of cross-validation to train and tune model $\mathcal{M}_{SS}$ using classification algorithm $\mathcal{C}_{SS}$.

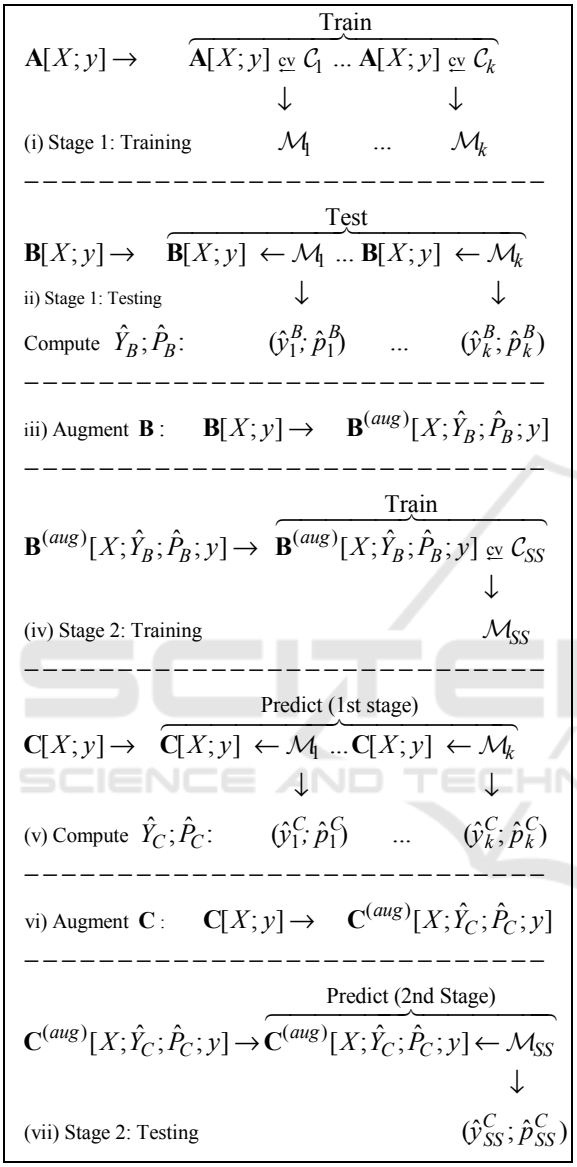

Figure 1: Training and testing a two-stage stack with k classifiers in the first stage, one classifier in the second stage, and 3 independent data sets A, B, C.

It is common practice to train the second stage using only the prediction pairs, yielding $\mathbf{B}^{(aug)}[\hat{Y}_B;\hat{P}_B;y] \underline{cv}\ \mathcal{C}_{SS}$. The latter is the approach followed for the experiments in this paper (see section 4.2.3).

- In **Step(v)**, base models $\mathcal{M}_1 \ldots \mathcal{M}_k$ are applied on dataset $\mathbf{C}[X;y]$ to compute prediction vectors $\hat{y}^C_{1..k}$ and probability vectors $\hat{p}^C_{1..k}$, depicted as an $(n_C \times 2\cdot k)$ matrix $(\hat{Y}_C;\hat{P}_C) \equiv (\hat{y}^C_1 \ldots \hat{y}^C_k;\hat{p}^C_1 \ldots \hat{p}^C_k)$, with n_C observations in data set **C,** and $2\cdot k$ columns $(\hat{y}^C_1 \ldots \hat{y}^C_k;\hat{p}^C_1 \ldots \hat{p}^C_k)$.
- In **Step(vi)**, data set $\mathbf{C}[X;y]$ is augmented by adding to it the $(n_C \times 2\cdot k)$ matrix $(\hat{Y}_C;\hat{P}_C)$, resulting in data set $\mathbf{C}^{(aug)}$ with schema $\mathbf{C}^{(aug)}[X;\hat{Y}_C;\hat{P}_C;y]$.
- In **Step(vii)**, corresponding to Stage 2: Testing, trained second-stage model $\mathcal{M}_{SS}$ is applied on $\mathbf{C}^{(aug)}[X;\hat{Y}_C;\hat{P}_C;y]$ -or on $\mathbf{C}^{(aug)}[\hat{Y}_C;\hat{P}_C;y]$ if only $(\hat{Y}_C;\hat{P}_C)$ were used to train the second stage- and produce the stack predictions and probabilities $(\hat{y}^C_{SS};\hat{p}^C_{SS})$, represented as an $(n_C \times 2)$ matrix, where n_C is the number of observations in data set **C,** and 2 is the number of columns, corresponding to $\hat{y}^C_{SS}$ and $\hat{p}^C_{SS}$. Stack predictions $\hat{y}^C_{SS}$ can be evaluated using the actual response variable y in data set **C** for comparison.

After the stack is trained, tuned and tested, it can be used to make predictions on new data **D**. Figure 2 depicts the two-stage stack making predictions on incoming (and therefore unlabeled) data **D.**

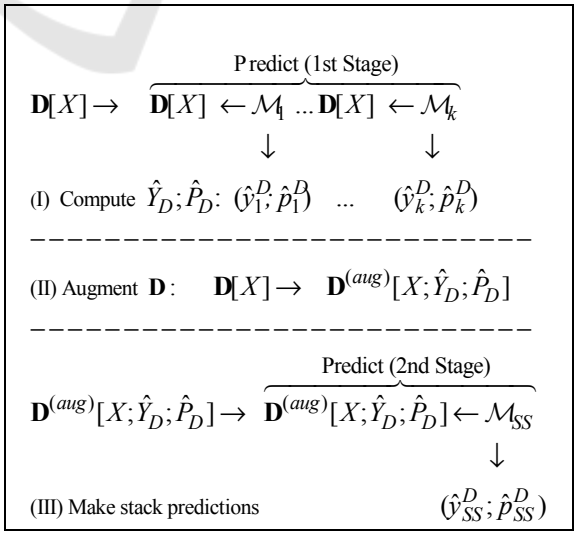

Figure 2: Using the stack for prediction on new data.

- In **Step(I)**, base models $\mathcal{M}_1 \ldots \mathcal{M}_k$ are applied on data set $\mathbf{D}[X]$ to compute matrix $(\hat{Y}_D; \hat{P}_D) \equiv (\hat{y}_1^D \ldots \hat{y}_k^D; \hat{p}_1^D \ldots \hat{p}_k^D)$.
- In **Step(II),** $\mathbf{D}[X]$ is augmented to $\mathbf{D}^{(aug)}[X; \hat{Y}_D; \hat{P}_D]$.
- Finally, in **Step(III)**, the stack computes predictions and probabilities $(\hat{y}_{SS}^D, \hat{p}_{SS}^D)$ for each of n_D records in $\mathbf{D}^{(aug)}$.

3.2 Considerations and Best Practices

The use of three independent datasets (**A, B**, and **C)** in this methodology guarantees that models are tested using new and unseen data: base models are trained using dataset **A** and tested using new and unseen dataset **B**, from where predictions are derived; dataset **B**, augmented with those predictions, is subsequently used to train the second stage model, which is then tested with new and unseen dataset **C.** This approach avoids data leakage, which can cause models to over-represent their generalization error and therefore overfit. The approach is feasible, provided there is enough data to produce three independent datasets, which is our case.

Models are typically trained and tuned using a resampling method, such as cross-validation. This guarantees that the models' hyperparameters are optimized for the data and task at hand before they are tested on new data.

It is advisable to use learning algorithms that can produce probability values as well as predictions (e.g. trees, Bayesian learners, logistic regression). Probabilities reinforce predictions when acting as inputs of the downstream (e.g. second) stage.

Learning algorithms used to train base models should be as different as possible so that the predictions made by them have relatively low correlations (< 0.75 - 0.80). If predictions are highly correlated, it indicates that the base models map very similar hypothesis functions, which defeats the purpose of using a stack.

4 EXPERIMENTAL SETUP

In the experiments we investigated the following: a) the use of a two-stage stack learnt from data, in the manner described in the previous section, to build an early detection tool of academically at-risk students structured as a binary classifier (two classes on a course by course basis): at-risk, and students in good academic standing; b) the (relative) performance of the two-stage stack when compared to stand-alone classifiers.

4.1 Datasets

Undergraduate student data from ten semesters (Fall 2012 - Spring 2017) were extracted, cleaned, transformed and aggregated into a complete dataset (no missing data), with each record -the unit of analysis- corresponding to a course taken by a student in a given semester, using the record format depicted in Table 1. Four sources of data were used to compile the dataset: a) student demographic and aptitude data; b) course grades and course related data; c) student activity data of the first six weeks of each semester, collected by the learning management system (LMS) tools; d) a composite score adding up the partial contributions to the student's final grade in the first six weeks of the semester, collected by the LMS's gradebook tool (i.e. student grades on specific gradable events, such as assignments and exams).

Table 1: Features in input data sets.

Predictors	Data type
Gender	Categorical
Age	Numeric
Class (Freshman, Sophomore, Jr, Sr.)	Categorical
Aptitude Score (e.g. SAT)	Numeric
Cumulative GPA	Numeric
Course size	Numeric
Discipline (SCI, LA, CSM, BUS, SBS, CA)	Categorical
LMS Total Activity (weeks 1-6 + sum)	Numeric x 7
Login (weeks 1-6 + sum)	Numeric x 7
Content Read (weeks 1-6 + sum)	Numeric x 7
Gradebook Composite Score (wks 1-6)	Numeric
Target feature: Academic_Risk (Yes=at risk; No=good standing)	

LMS student activity data was recorded as weekly frequency ratios, normalized with the average and standard deviation of each course. Three LMS student activity metrics were considered (number of weekly logins, number of weekly accesses to content, weekly overall activity). Summations of all 6 weeks x 3 metrics were also added as features, for a total of 21 LMS student activity metrics.

Each record included the final grade of the student in the course as the target feature of the unit of analysis, recoded into a binary variable to classify students in two categories (good standing and

academically at-risk), using a threshold grade of acceptable academic performance: undergraduate students with less than a letter grade C are considered at risk.

The full dataset was randomly sampled and a total of 31029 records of students taking courses was extracted (a 35% sample of the total data over 10 semesters).

4.2 Methods

We performed eight batches of experiments, using two different configurations of classification algorithms for the first-stage (base) models; two different sets of predictors to train the base models; and two different algorithms for the second-stage model. Each batch was repeated 10 times with varying random generator seeds to account for variation in predictive performance due to the data; in each run the data was randomly partitioned into datasets A, B and C with 10343 records each. This amounted to a total of 80 runs in the experiment (2 x 2 x 2 x 10). The data proved to be highly imbalanced (only 4.1% of the records had at-risk students). Instead of balancing the training data, the probability thresholds (aka probability cutoffs) that determine the prediction were adjusted to account for the imbalance in the target feature.

4.2.1 First-stage Classifiers

Four classification algorithms, organized in two configurations of three base classifiers each, were used in the experiment.

- XB: XGBtree (Chen and Guestrin, 2016), a recent implementation of the gradient boosted tree algorithm
- NN: A feed-forward neural network (multilayer perceptron) with one hidden layer and varying number of units.
- RF: The Random Forests algorithm (Breiman, 2001), a variation of bagging applied to decision trees.
- NB: The naïve Bayes algorithm with kernel estimation, to estimate the densities of numeric predictors (John and Langley, 1995).

The chosen classifiers are either state-of-the-art (e.g. XB) or well-proven classification algorithms when dealing with training data of mixed types (numeric and categorical). They are also substantially different in their theoretical underpinnings, and should therefore yield non-identical prediction errors (section 4.3 elaborates on the base classifiers' predictions and probabilities, and their correlation). The classifiers were organized in two configurations of base models: XB+NN+RF and XB+NN+NB.

4.2.2 Second-stage Classifiers

The second-stage model was trained using:

- LOG: regularized logistic regression using the the LibLinear library (Fan et al., 2008).
- LMT: Logistic model trees (Landwehr et al., 2005)

Logistic regression is an effective classifier, widely used when dealing with numeric and binary data (which is the case of the second-stage training data). Regularization helps overcome overfitting. Logistic model trees are a variation of the model trees suggested by (Džeroski and Ženko, 2004) as best practice when training second-stage models.

4.2.3 Predictors

The base models were trained using two distinct predictor configurations:

- ALL: All the predictors (as described in Table 1) for all 3 classifiers.
- NoCS: All the predictors for all 3 classifiers, except the LMS Gradebook composite score (there are instructors that do not use the Gradebook, and therefore the model cannot be trained using partial contributions to the final grade as predictors).

The second stage was trained using first-stage predictions and probabilities (the first-stage predictors were not used as predictors in the second stage). We opted for this approach to isolate the effectiveness of the stack without the contribution of the first-stage predictors in the second stage.

4.2.4 Computational Details

The stacked ensemble system was coded in R, using the *caret* package (Kuhn, 2008) to orchestrate the training and tuning of both the three base models and the second stage model, and perform predictions. R implementations (packages) of the chosen algorithms were used to train the models: *xgboost* for gradient boosted trees; *nnet* for feed forward neural nets; *randomForest*; *naivebayes*; *Rweka* for logistic model trees; *LibLinear* for regularized logistic regression. (Note: There are a handful of turnkey implementations of stacked ensembles, mostly written in R, Python and Java - e.g. H2O.ai-, but we decided to code our own implementation to have a better control over the execution process). The experiments were run on an

Intel Xeon server, 2.90GHz, 8 processors, 64GB RAM. Parallel processing was coded into the system to make use of all 8 cores during training and tuning. Each test run of the stack, including training and tuning of the first and second stage models took, on average, 3 to 4 min. to complete.

4.2.5 Evaluating and Comparing Algorithms

First-stage and second-stage models were trained and tuned using 5-fold cross-validation and a grid search to tune the models' hyperparameters. Features were pre-processed (centered and scaled) as part of the training and tuning process. Table 2 depicts the classification algorithms and their hyperparameters, used to tune the models. Tuning was performed using the area under the curve of the ROC curve (AUC), a widely-used metric to summarize classification performance, especially when dealing with a high imbalance in the data. The area under the curve of the ROC curve (AUC) was averaged over the ten runs of each of the eight experiment batches, computing a mean value and a standard error. Single tailed, paired t-tests were performed for each batch, comparing the mean AUC of the stack relative to the mean AUC of the best performing first-stage model. Sensitivity (1-Type II error) and Specificity (1-Type I error) were also reported for the stack, using the values computed at the minimum distance between the ROC curve and the top right corner of the ROC chart box.

Table 2: Tuning and model hyperparameters.

Algorithm	Hyperparameter(s)
XB	# boosting iterations, min_child_weight, max_depth, col_sample_bytree, shrinkage
NN	size, decay
RF	mtry (# sampled vars)
NB	Laplace correction, usekernel(T/F), Adjust
LOG	cost, loss function, tolerance
LMT	# iterations

4.3 Results and Discussion

Table 3 displays the assessment of mean predictive performance of the stacked ensemble for the eight experiments described in section 4.2. The stack exhibited very good predictive performance when trained with all first stage predictors, outperforming all three base classifiers for both configurations of first base classifiers (XB+NN+RF and XB+NN+NB). For the XB+NN+RF/LMT stack, the mean AUC value was 0.935; for the XB+NN+RF/LOG stack, the mean AUC value was 0.939. The naïve Bayes (NB) algorithm was considerably less performant than the random forests algorithm -compare mean AUC(NB)=0.858 with mean AUC(RF)=0.920-, but all three classification algorithms in the XB+NN+NB configuration are considerably different from each other, as exhibited by the correlations between predicted probabilities generated by the first-stage models in the XB+NN+NB configuration (see Table 4). This probably gave way to high mean AUC values of the stack (mean AUC = 0.933 for both LMT and LOG,.

Table 3: Stack Predictive Performance Results.

Stage 1 Classifiers	Stage 1 Predictors	Stage 2 Classif.	Mean (AUC)				Stack AUC		paired t-test p-value (*)	Sensitiv.	Specif.
			XB	NN	RF	NB	Mean	SE			
XB+NN+RF	ALL	LMT	0.928	0.920	0.925		0.934	0.003	0.020	0.865	0.873
XB+NN+RF	ALL	LOG	0.928	0.920	0.925		0.936	0.002	0.000	0.867	0.875
XB+NN+RF	NoSC	LMT	0.846	0.833	0.834		0.855	0.001	0.001	0.792	0.767
XB+NN+RF	NoSC	LOG	0.846	0.833	0.834		0.855	0.001	0.000	0.792	0.770
XB+NN+NB	ALL	LMT	0.928	0.920		0.858	0.933	0.002	0.057	0.869	0.865
XB+NN+NB	ALL	LOG	0.928	0.920		0.858	0.933	0.002	0.026	0.870	0.865
XB+NN+NB	NoSC	LMT	0.846	0.833		0.775	0.851	0.002	0.115	0.794	0.757
XB+NN+NB	NoSC	LOG	0.846	0.833		0.775	0.852	0.001	0.021	0.791	0.761

(*) p-values of upper-tailed paired t-tests over 10 runs (n=10) of AUC means of the stack (second-stage classifier) and the best performing first-stage classifier, for each experiment.

slightly smaller than those of the XB+NN+RF configuration). Predictive performance improvement was moderate but consistent, and particularly relevant considering the high AUC values displayed by all base classifiers, and the limited scope of the experiment setup (e.g. number of classifiers put in place). The one-sided paired t-tests were significant (α =0.05) in almost all cases where all the first-stage predictors were present (the XB+NN+NB/LOG configuration was the only exception). This indicates a significant difference in the mean predictive performance -measured by the AUC- between the stacked ensemble and its first-stage models. The stack seemed to extract additional predictive performance from its component classifiers. The sensitivity and specificity values were also high, (min. 0.865 and max. 0.870 for sensitivity; and min. 0.865 and max. 0.873 for specificity), which implies rather low Type I and Type II errors: on the average, only 13.2% of academically at-risk students went undetected; and the stack produced 13.1% of false alarms. The latter number is much higher, considering the high imbalance in the data, strongly skewed towards students in good standing), but less important in the overall context: we seek classifiers with a low Type I error to minimize the number of false negatives while ensuring that not too many false positives are generated

The absence of Gradebook data (partial contributions to the final grade) had an expected negative impact on the predictive performance of the stack, reducing its average AUC to 0.855 for the XB+NN+RF configuration, and to values of 0.851 and 0.852 for the XB+NN+NB configuration. The stack's predictive performance remained superior on average, and significant differences between mean AUC values of the stack and its component classifiers were present in all but one configurations (XB+NN+NB/LMT). Still, it is noteworthy how well the stack performed despite the relatively weaker performance of the naïve Bayes classifier -AUC(NB) = 0.775-. This reveals another advantage of the stacked ensemble architecture: it cushions weaker performances of its components, promoting more stable predictions when faced with varying characteristics of the data.

Table 4 displays correlations between predicted probabilities generated by the first-stage models. For the XB+NN+RF configuration, values were above 0.75, meaning that classifiers, although accurate, represent similar hypothesis functions and therefore make similar errors. Instead, when replacing RF with NB in the XB+NN+NB configuration, the NB classifier predictions exhibited low correlations with those of the NN and XB classifiers. The NB classifier is less accurate, but does not span the same hypotheses space, and therefore produces different errors. This provides an explanation for the good predictive performance of the stack, in spite of the varying predictive performance of the first-stage models. The stack is tuned by carefully considering the number and type of classifiers, and the correlations between predictions produced by base classifiers

Table 4: Correlations of Predicted Probabilities.

	Mean	Std Dev	Min	Max
XB-NN	0.77	0.11	0.44	0.92
NN-RF	0.76	0.11	0.49	0.89
RF-XB	0.85	0.07	0.70	0.96

XB+NN+RF configuration

	Mean	Std Dev	Min	Max
XB-NN	0.77	0.11	0.44	0.92
NN-NB	0.22	0.06	0.09	0.33
NB-XB	0.34	0.06	0.24	0.45

XB+NN+NB configuration

5 LIMITATIONS AND FUTURE WORK

The current research has several limitations. First, the study imposed a stacked ensemble architecture limited to two stages, three base classifiers and one single second-stage classifier. This was done to restrict the amount of time required for the execution of each training process as the experiment was repeated multiple times. The purpose of the study at this preliminary stage is not to identify an optimal architecture but rather to empirically test the effectiveness and stability of the proposed stack ensemble architecture. Likewise, the choice of classifying algorithms was discretionary: the study used well-known classifiers with dissimilar characteristics and a proven record of predictive performance, some being ensembles themselves, to train the base classifiers, with the purpose of attaining good classification metrics while covering as much as possible of the hypotheses space, within the constraints of the computational resources available. Other classification algorithms could have been considered, but we settled for six classification algorithms using the rationale described above.

Nonetheless, the study provides first-time insight of the use of a stacked ensemble architecture in the domain of learning analytics and early detection of academically at-risk students. We recognize that much more could be written about each of these topics. However, we will provide more complete coverage of these topics at the conference.

6 SUMMARY AND CONCLUDING COMMENTS

Stacked ensembles are powerful and flexible machine learning frameworks with the potential of delivering better and more stable predictions. This paper demonstrates how to create a stacked ensemble and perform predictions of academically at-risk students. The impetus of this research stems from the need of introducing novel approaches that can be used in practical settings to predict academic performance and carry out early detection of students at risk. The methodology presented in this paper is the subject of intensive research and exploration at our institution, inclusive of the analysis of different configurations of classifiers, model tuning criteria, arrangements of predictors, and its impact on the stack's predictive performance. A pilot on a group of course sections has been run at the College during Fall 2017 and will continue through Spring 2018 using the stacked ensemble methodology described in this paper. The model output will be visualized through the LMS using a graphical user interface –a dashboard- augmented with statistics generated from the prediction. Hopefully, this paper will provide the motivation for other researchers and practitioners to begin exploring the use of stacked ensembles for predictive modeling in the learning analytics domain.

REFERENCES

Breiman, L., 2001. Random Forests. *Mach Learn 45, 5–32.*

Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. *CoRR abs/1603.02754*.

Džeroski, S., Ženko, B., 2004. Is Combining Classifiers with Stacking Better Than Selecting the Best One? *Mach Learn 54, 255–273.*

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J., 2008. LIBLINEAR: A Library for Large Linear Classification. *J Mach Learn Res 9, 1871–1874*.

Jayaprakash, S. M., Moody, E. W., Lauría, E. J. M., Regan, J. R., Baron, J. D., 2014. Early Alert of Academically At-Risk Students: An Open Source Analytics Initiative. *J. Learn. Anal. 1, 42*.

John, G.H., Langley, P., 1995. Estimating Continuous Distributions in Bayesian Classifiers, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95. Morgan Kaufmann *Publishers Inc., San Francisco, CA, USA, pp. 338–345.*

Kaggle Team, 2017. No Free Hunch: Allstate Claims Severity Competition, 2nd Place Winner's Interview: Alexey Noskov. *Off. Blog Kagglecom*.

Kuhn, M., 2008. Building Predictive Models in R Using the caret Package. *J. Stat. Softw. Artic. 28, 1–26.*

Lam, L., 2000. Classifier Combinations: Implementations and Theoretical Issues, in: Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, *2000 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 77–86.*

Landwehr, N., Hall, M., Frank, E., 2005. Logistic Model Trees. *Mach. Learn. 59, 161–205*.

Lauría, E.J.M., Presutti, E., Sokoloff, M., Guarino, M., 2016. Crossing the Chasm to Big Data: Early Detection of at-Risk Students in a Cluster Computing Environment. Presented at the Proceedings of the 7th International Learning Analytics & Knowledge Conference (LAK'17)- Practitioner Track. *Vancouver, Canada.*

Pistilli, M.D., Arnold, K.E., 2010. Purdue Signals: Mining Real-time Academic Data to Enhance Student Success. Campus 22–24.

Polikar, R., 2006. Ensemble based systems in decision making. *IEEE Circuits Syst. Mag. 6, 21–45*.

Romero, C., Ventura, S., Garcia, E., 2008. Data mining in course management systems: Moodle case study and tutorial. *Comput Educ 51, 368–384*.

Sill, J., Takács, G., Mackey, L.W., Lin, D., 2009. Feature-Weighted Linear Stacking. *CoRR abs/0911.0460.*

Smith, V.C., Lange, A., Huston, D.R., 2012. Predictive Modeling to Forecast Student Outcomes and Drive Effective Interventions in Online Community College Courses. *J. Asynchronous Learn. Netw. 16, 51–61*.

van der Laan, M., Polley, E., Hubbard, A., 2007. Super Learner (Working Paper No. 222), Berkeley Division of Biostatistics Working Paper Series. *University of California, Berkeley*.

Wolpert, D.H., 1992. Stacked generalization. *Neural Netw. 5, 241–259*.