Robust Predictive Models on MOOCs : Transferring Knowledge across Courses

ABSTRACT
As MOOCs become a major player in modern education, questions about how to improve their effectiveness and reach are of increasing importance. If machine learning and predictive analytics techniques promise to help teachers and MOOC providers customize the learning experience for students, differences between platforms, courses and iterations pose specific challenges. In this paper, we develop a framework to define classification problems across courses, provide proof that ensembling methods allow for the development of high-performing predictive models, and show that these techniques can be used across platforms, as well as across courses. We thus build a universal framework to deploy predictive models on MOOCs and demonstrate our case on the dropout prediction problem.

Keywords
Transfer Learning, Ensembling methods, Stacking, MOOCs, Dropout prediction

1. INTRODUCTION
As Massive Open Online Courses (MOOCs) continue to become a vital part of modern education, it becomes more and more necessary to increase their effectiveness and reach. Along with learning science and design, data analytics is known to be one of the fields most likely to improve this new education experience ([1]). Predictive analytics are particularly promising, allowing researchers to design real-time interventions and to adapt course content based on student behavior ([7],[9],[6],[3]).
Ideally, these predictive analytics would act in ways similar to an experienced teacherâĂŤone who is able to identify different students, and to adapt her actions accordingly. However, because the data available for training models is often significantly different then the data to which those models will be applied, it can be challenging to fully realize this promise ([8]). A predictive analytics system for MOOCs should be able to build on accumulated "past data" to make accurate predictions about an ongoing class. Thanks to the vast offerings of MOOC databases like edX and Coursera, there is now a plethora of past data available, both across and within a given course.
But this diversity of available courses also means the goal of real-time prediction is easier set than accomplished. Courses may come from different platforms, focus on different topics, or occur at different times. They may have more or less homework, span different lengths of time, or require different levels of involvement. As platforms evolve, courses may also morph to include new information or fulfill shifting demands. Such changes typically affect the behavior of students.
This raises a number of questions and challenges for a data or learning scientist. Given data from a set of repeatedly offered MOOC courses, key questions that shape the design of relevant predictive analytics methods are as follows:

Purpose Can I use past courses to predict outcomes within an ongoing course?

Data What data should I exploit to build my predictions? Is data from a single course enough, or should I use several courses?

Method What method will achieve good efficacy if I use data from a single course? Several courses?

In this paper, we address the challenges inherent in building predictive models that perform well across courses. We answer the questions, mentioned above, that a MOOC analyst would ask about the prediction objectives, the data, and the methods used to build such models. We also address whether such methods are able to perform well across courses on the same platform, and on different platforms.
This paper is divided into five sections. The remainder of this first section explores the available literature regarding MOOC dropout prediction and ensembling methods in machine learning. Section 2 introduces the formal notations, assumptions, and data sets we used to prove our case. Section 3 details different methods that prove useful for building robust models that transfer well to new courses. Section 4 presents the evaluation metrics, and showcases the effectiveness our techniques on the dropout prediction problem. Section 6 evaluates the potential impact of such techniques, and summarizes the key findings and 7 conclusions.

Literature review
Even before the recent e-learning boom, concerned researchers

have attempted to predict dropout. One major obstacle facing such attempts is the difficulty of building robust predictive algorithms. While working with early e-learning data, the authors of [7] improved the performance of their learning algorithm by merging several predictive algorithms together, namely Support Vector Machines, Neural Networks, and Boosted Trees.
Since then, almost all dropout studies have been conducted on MOOC data. Some researchers (like the authors of [9], who studied the effects of collaboration on the dropout rate of students) focus on understanding drivers of dropout among students. Others develop feature extraction processes and algorithms capable of pinpointing at-risk students before they drop out. If a MOOC is able to identify such students early enough, these researchers reason, it may be possible for educators to intervene. In [6], Halawa et. al. used basic activity features and respective performance comparison to predict dropout one week in advance. The authors of [2] included more features, as well as an integrated framework that allowed users to apply these predictive techniques to MOOC courses from various eligible platforms.
As MOOC offerings proliferate, the ability to "transfer" statistical knowledge between courses is increasingly crucial, especially if one wants to predict dropout in real time. Unfortunately, it is often difficult to take models built on past courses and apply them to new ones. In [3], Boyer and Veeramachaneni showed that models built on past courses don't always yield good predictive performance when applied to new courses.
Because there is generally only one dataset available per course, the ability to build robust models on MOOCs has naturally accompanied the rise of ensemble methods. Over the past twenty years, a flourishing predictive literature has appeared, offering various techniques for choosing and ensembling models in order to achieve high-performing predictors. A technique called "stacking" has proven particularly promising. In [5], Svzeroski et. al. showed that stacking models usually perform as well as the best classifiers. They also confirmed that linear regression is well-suited to learning the metamodel, and introduced a novel approach based on tree models. The authors of [4] demonstrated the possibility of incrementally adding models to the "ensembling base" from a pool of thousands. Sakkis et. al. [10] used the stacking method to solve spam filtering problems, finding that it significantly improved performance over the benchmark.
In this paper, we explore a framework conducive to building robust predictive models applicable to MOOCs. Although we do address dropout prediction specifically, we also consider the broader possibilities for building predictive models from a set of courses.

2. PROBLEM SETTING AND DATA SETS

We place ourselves in the context of using past courses to build a predictive model for a unseen course. We use the term *source* courses for those courses whose data is used to build (train) the predictive models, and the term *target* course for the initially "unseen" course. We consider the general problem of predicting for each student i an outcome y_i^t at time t in the future. We suppose that we have access to information about each student's behavior through a set of features: for example, a behavioral vector $x_i^t \in R^d$ describes the behavior of the student i at time t.

ID	Name	Platform	Students	Weeks
C_0	6002x13	edX	29,050	14
C_1	6002x12	edX	51,394	14
C_2	201x13	edX	12,243	9
C_3	3091x12	edX	24,493	12
C_4	3091x13	edX	12,276	14
C_5	aiplan_001	Coursera	9,010	5
C_6	aiplan_002	Coursera	6,608	5
C_7	aiplan_003	Coursera	5,408	5
C_8	animal_001	Coursera	8,577	5
C_9	animal_002	Coursera	5,431	5
C_{10}	astrotech_001	Coursera	6,251	6
C_{11}	codeyourself_001	Coursera	9,338	7
C_{12}	criticalthinking_1	Coursera	24,707	5
C_{13}	criticalthinking_2	Coursera	15,627	5
C_{14}	criticalthinking_3	Coursera	11,761	5

Figure 1: Summaries of courses used for experiments. The first set contain five courses from edX platform (Harvard-MIT), the second set contain ten courses from the EDI platform (University of Edinburgh).

We assume that the *source* courses and the *target* course share a non-empty set of behavioral features, such that we can restrict ourselves to this set when building our predictive models. As we will see below, this hypothesis is often verified in practice. In this context, our goal is to learn the statistical mapping from the behavior vector $x^{w'}$ of a student in week w' to their particular outcome y^w in week w. To do this, we propose to learn a statistical model by leveraging data (both $x^{w'}$ and y^w) from previous courses.
Data sets: Our experiments are based on two sets of MOOC courses. The source set, on which we built and validated our methods, consists of five courses, and was provided by the edX platform. Its attributes are described in table 1. This dataset initially contained log files describing students' behavior on the platform. For each student in these courses, we extracted a set of 21 features on a weekly basis. The second, or "target," set of courses was given by the University of Edinburgh (EDI) and consists of 10 courses from Coursera, whose attributes are also described in table 1. The courses in this second set are shorter in duration (only 5 weeks), and contain less detailed features. They share only 11 features with the first set of courses.
To build a robust framework that could achieve reliable predictive models, we initially designed, trained and validated our different methods on the first set of 5 courses from edX. At the very end of this paper, we apply these models to the second set of courses. When building models on one course and applying its predictions to another, two issues must be overcome. First, the two courses might not share the same features (for example, the grade for p-sets during week 1 might be available for some courses and not for others). Second, they might not span the same number of weeks. We overcame these issues by only considering features and timespans common to all courses. Therefore, we first used 21 features and 9 weeks when we trained, tested and validated our models on the edX courses. We then restricted ourselves to an 11-feature, 5-week scheme when testing our procedure on the EDI courses.

3. METHODS

In this section, we describe the different approaches used to build a predictive model for dropout. We first describe common practices, and explore whether a single course can be used to build a predictive model for another course. We then explain how the aggregation of several data sources can be used to improve the predictive power of a model. Finally, we describe how a type of machine learning technique called "Ensembling methods" can be used to further boost the predictive power of models built from different courses.

3.1 Naive approaches

Simple models: When building a supervised predictive model out of data sources, the first logical step involves training a single model on a particular dataset. Although plenty of classification algorithms exist, there is no systematic a-priori method to determine which one is best suited to a particular problem. This is the first hurdle that must be overcome when building a robust model.
The second hurdle involves choosing which prior course to train the model on. Although the first course s_1 may have a distribution closer to that of our target course t, s_2 may have more samples, resulting in a better predictive model. Hence, we must choose both an algorithm and a prior course that, working together, will be most suitable for predicting outcomes in the new course.
Merging sources: Alternatively, one may ask, why not use all the data from all the courses? Could learning a predictive model on the concatenated data from all courses $\{s_1, s_2, ..., sn\}$ result in a model that transfers better to new courses? This mitigates the problem of choosing among courses, but certainly does not solve the need to choose a a modeling approach, as it raises a number of new questions. First, concatenating obscures the differences between courses, preventing a predictive model from making predictions within the environment of the original sample. Second, if the courses have different numbers of students, concatenating them can overweight the influence of the larger data sets. Though this may not be a concern in cases where all datasets are drawn from a single distribution, in our case, combining the datasets is likely to limit the overall information available.
Although those concerns could be addressed using different tricks (for example, adding a "dataset" feature to account for the particularities of models, or undersampling bigger datasets to balance their weight in the concatenated set), we instead sought a different and more systematic approach to building robust models.

3.2 Ensembling methods to improve transfer of models in MOOCs

In this section, we leverage a type of machine learning technique called "Ensembling methods," often used to aggregate different predictive models. These techniques are now widely practiced after their successful deployment in the *Netflix*[1] challenge, in which hundreds of teams competed to build a precise recommendation system. They are used both in the industry and in public competitions, such as those held by Kaggle [2]), to improve the predictive power of models trained and tested on a single dataset([3]).
In this paper, we ask whether ensembling methods can in fact help in transferring models trained on one or more courses. What additional parameter tuning or methods do we have to develop to make this transference possible? Ordinarily, a data scientist uses ensembling techniques by selecting different subsets of features and training examples, learning algorithms, or parameters, and then building a set of predictors to ensemble. In the context of MOOCs, which have multiple courses, there is a natural split in the data we can exploit. We will demonstrate that in the some cases (for short term predictions), these approaches outperform the performance of the transferred predictive models built on a single course data and from a single algorithm.
We will discuss the different methods explored with respect to the three following dimensions :

- A set of pre-trained predictors $E = \{p_1, ..., p_n\}$
- A set of rules to combine the predictions of different algorithms. We call these rules "voting rules" and note them $R = \{R_1, ..., R_p\}$
- A structure S, which specifies in which order and to which predictions these rules should be applied.

Predictors: The first step in building a transferring method for dropout prediction is to train a set of predictive algorithms on data available from past courses. Given N source courses and P predictive models to train, this produces Nx$P = H$ predictors $\{p_1, ..., p_H\}$.
We trained four classification algorithms (RandomForest, Logisitic Regression, SVM, and Nearest Neighbors) on each course. For each of these algorithms, we used 5-fold cross-validation to optimize the parameters. We note that for each of the past available courses, we left a holdout subset of 20% for a later-stage parameters optimization.

Fusing methods: One can combine a set of underlying predictions $\{p_1(x), ..., p_H(x)\}$ in infinite ways. Below, we list three common ways of ensembling that have been proven to perform well over a broad range of applications:

- Averaging (R_1). The most common fusion method consists of averaging the predictions of different underlying predictors.
$$p_{norm}(x) = \frac{1}{H}\sum_{i=1}^{H} p_i(x)$$
- Normalized averaging (R_2). When combining disparate predictive methods, some predictors might produce estimations in $\{0.49, 0.51\}$, while others produce estimations in $\{0, 1\}$. To account for the diversity of ranges from one predictor to the next, one can normalize the predictions of each predictor before averaging them.
$$p_{avg}(x) = \frac{1}{H}\sum_{i=1}^{H} \frac{p_i(x) - \min_{z\in t} p_i(z)}{\max_{z\in t} p_i(z) - \min_{z\in t} p_i(z)}$$
- Rank voting (R_3) In addition to differences in the range of probabilities, may also differ in how fast they

[1]http://www.netflixprize.com/
[2]https://www.kaggle.com/
[3]https://inclass.kaggle.com/c/mooc-dropout-prediction

vary with the input. To mitigate this behavior (which might cause the overall prediction to overweight very sensitive predictors), one can rank the probabilities within the target set first, then average and normalize the resulting ranks of different .

$$p_{rank}(x) = \frac{1}{H}\sum_{i=1}^{H}\frac{rank(p_i(x)) - 1}{N_t}$$

where $rank(p_i(x))$ refers to the rank of sample $p_i(x)$ in the set $\{p_i(z), z \in t\}$

We note that none of these techniques assume anything about the relative performance of different algorithms. We call those voting schemes "symmetric" because they treat each predictor in the same way. Our next set of methods allows us to fuse predictions by accounting for the varying performances of different predictors, and allowing us to put more weight on the "best" predictors. To identify these weights, we use the holdout subset of our source courses, and develop a method known as *stacking* as follows:

- Stacking (R_4) We concatenate all the holdout subsets from all source courses $X_{HO} \in R^{N_{HO} \times d}$ and apply all pretrained predictors $\{p_1(x), ..., p_H(x)\}$ on this dataset. The output of this procedure $Y_{HO} \in R^{N_{HO} \times H}$ is then considered as a new training dataset. We apply a logistic regression on this output to learn the weights for each predictor.

Structures: The last component of an ensembling method is the structure, within which predictions are merged together. Two example structures are shown in figure 2. Structures can influence the final performance of the method. Given a set of predictors and a set of fusing methods, the

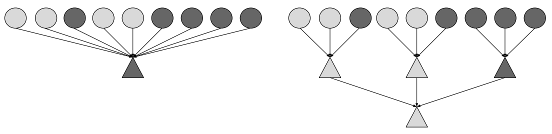

Figure 2: Illustration of two structures used to combine the same set of predictions using a simple voting rule (R_1) (color code is 1 for blacks and 0 for whites). The two different structures result in two different predicted outcomes.

"structure" is the sequence in which said predictors are fused in order to produce the final output.

Learning the structure: We posit that the structure of votes could influence the performance of the overall ensembling method. Due to the potentially arbitrary number of "layers," the number of possible structures is infinite. We restrict ourselves to structures with a high degree of symmetry. We enumerate a subset of structures in the figure 4. We then use algorithm 1 to compare the performance of the preselected structures. Our goal is to find the structure that will yield the highest performing predictor when applied to target courses.
For this comparison to be independent of the choice of target course, we consider each one of the five edX courses as the target course successively, calling them C_0, C_1, C_2, C_3 and C_4). The remaining four act as source courses. We then

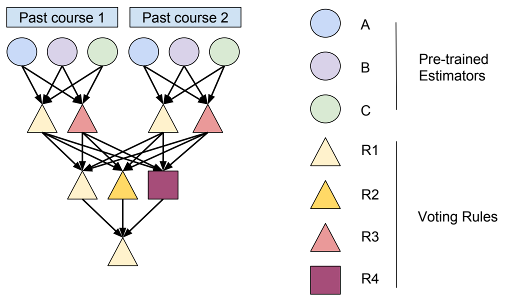

Figure 3: Example of a complex voting structure built on top of two data sources (past courses). The predictions of the different predictors are first aggregated by course then aggregated across courses.

aggregate our results by averaging the performance over the five permutations. We also remark that, in order to learn the metamodel necessary to the stacking rule, we separate all source course into a train and a validation set, as explained in algorithm 1.

```
Data: Full Data for the 5 edX courses
Result: Evaluate performance of structures
for problem in P do
    initialization : Split each dataset into a training and
     a validation subset (80% - 20%);
    for t in set of courses do
        Train each of the predictive algorithms (Random
         Forest, SVM, Logistic Regression, Nearest
         Neighbors) on each train set for the 4 remaining
         courses;
        for s in set of Structures do
            if s requires training then
                concatenate validation sets for 4 remaining
                 courses;
                train s on this dataset;
            else
                pass
            end
            measure ROC AUC : $AUC_s^{p,t}$;
        end
    end
end
```
Algorithm 1: Comparing performance of different ensembling structures.

4. RESULTS ON DROPOUT PREDICTION

MOOC platforms offer courses that span a particular length of time, typically around 12 weeks. A large cohort of students register for each of these courses, but only a fraction of this cohort usually remains at the end of the class.
We consider the common problem of predicting which students will remain in the class. Specifically, given a "current week" w_c and a "prediction week" w_p our goal is to identify which of the students present in the class at week w_c will have dropped out by week w_p. We call this particular problem (w_c, w_p), and we remark that, given a particular course lasting W weeks, there exist exactly $\frac{W.(W-1)}{2}$ potential problems of this type.

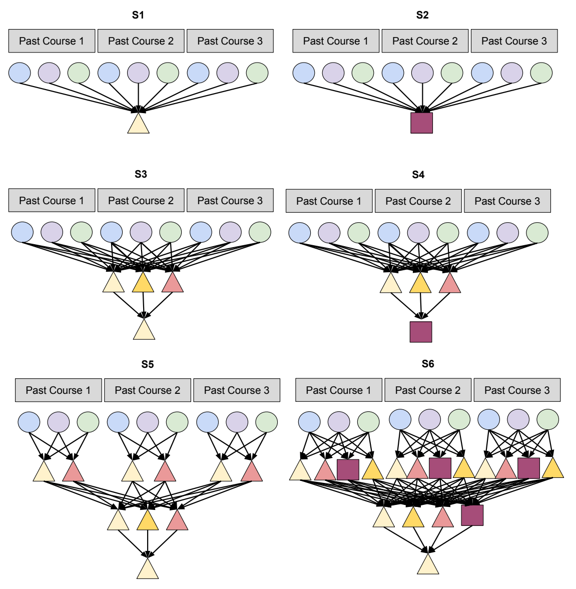

Figure 4: The six structures evaluated on the MOOCs dropout prediction problem. Only $S2, S4$ and $S6$ require training on the validation data because of the presence of a "stacking vote rule". (Note that for simplicity the diagrams shows only 3 courses and 3 predictive algorithms per courses whereas we used 4 and 4).

In the first to parts of this section, we use the five courses from edX (described in 1) to experiment and build our predictive models. We noted that the five courses from edX shared 21 behavioral features altogether. In the third part of this section, we show that these models indeed perform significantly better than our benchmark, even on courses from a different platform, Coursera.

4.1 Performance metrics and benchmarks

Evaluation metrics: To measure the performance of our predictive algorithm, we rely on the AUC-ROC metric, which is commonly used in dropout prediction. Because not all courses last for the same amount of time, we restrict ourselves to problems acceptable for all courses; i.e. the set

$$P = \{(w_c, w_p) \ s.t. \ w_c < w_p \text{ and } w_p < W_{course} \ \forall \ course\}$$

For the five courses used in this study, $W = 12$, meaning we can experiment on $|P| = 66$ different prediction problems. When comparing the performance of algorithms between problems, it becomes clear that some situations are intrinsically more difficult to predict than others. For instance, a short-term prediction problem (e.g, $(6, 5)$) will generally yield higher performance than a long term problem (e.g, $(6, 1)$). Similarly, some courses are more suited for predictions than others, due to the size of the student cohort or the volatility of students within that cohort.
To mitigate this, we *normalize* the performance, and use the following metric to measure the performance of an algorithm a on a problem p and on *target* course t :

$$DAUC_a^{p,t} = \max_{a' \in A} \left(AUC_{a'}^{p,t} \right) - AUC_a^{p,t}$$

In other words, we subtract the actual AUC of an algorithm from the best observed AUC of any other algorithm on this particular problem for this particular *target* course. In this configuration, a lower DAUC should be considered to indicate a better performance. In particular, $DAUC_a^{p,t} = 0$ exactly means that a is the best algorithm for this particular problem and target course.
To appreciate this metric over different problems, we display both the mean and the variance of the DAUC. In order to account for the different performance on different types of problems, we introduce two sets of problems, for which we choose to average the DAUC:

Two subset of problems

P Mean ROC AUC obtained on the 66 available problems

***P**$_s$* Mean ROC AUC obtained on three 'short term' prediction problems ($\{(5, 6), (8, 9), (11, 12)\}$)

Benchmarks : A simple approach to building predictive models is to train a classifier on a *source* course and use it to make predictions on the *target* course. In figure 5, we report the results obtained by training four different classification algorithms on a *source* course (for course 1 to 4) and applying it to the *target* course (C_0). We use 5-fold cross-validation on the training set, and we tune the parameters independently for each method, each source and each prediction problems.
A more systematic approach consists of building predictive models on the concatenation of all available data. In addition to avoiding the hurdle of having to 'guess' which course should be chosen as the *source* course, this approach also allows us to leverage more (and more diverse) data to train predictive models.
As shown in figure 5, we observed that, regardless of the algorithms used, models trained on the concatenation of all available data sources always performed better than the best models trained an a single course. This is true both in terms of average DAUC over all problems in P, and in terms of variance of the DAUC across those same problems.

4.2 Building robust models

Improvement through Merging Methods :
Concatenating the data from past courses undoubtedly improved the algorithms' predictive power, both in terms of average DAUC and variance. To further improve the average performance, and to reduce the variance of our dropout prediction system, we then leveraged the ensembling methods presented above. Instead of restricting ourselves to a choice of a single predictive algorithm, we trained four of them (SVM, Random Forest, Logistic Regression and Nearest Neighbors) and merged their predictions using a simple R_1 voting rule.
Figure 5 shows the average DAUC and its variance for different algorithms, as well as their "merged" version through an R_1 rule. Comparing the result obtained by the "merged" method with those of the four single algorithms, we observe that the merged method always performs comparably to the best single algorithm, beating all competitors on courses C_2 and C_4, and behaving comparably on courses C_1 and C_3). This is true both in terms of average performance and in terms of variance.

Next, we apply this same R_1 rule to merge the predictions built on the data concatenated from all available *source* courses. Here, the results unveil a lower DAUC average and variance for the "merged" method on the concatenated data than for any other algorithm on the same data. Moreover, this method performs better than those "merged" methods trained only on a single course, in terms of both average and variance. Through an "all-algo all-data" kind of method, we have achieved a more reliable and more accurate predictive model, on average, over all possible prediction problems.
In the next section we will see that, for certain type of problems, it is possible to improve this model significantly by using a more complex type of ensembling method called stacking.

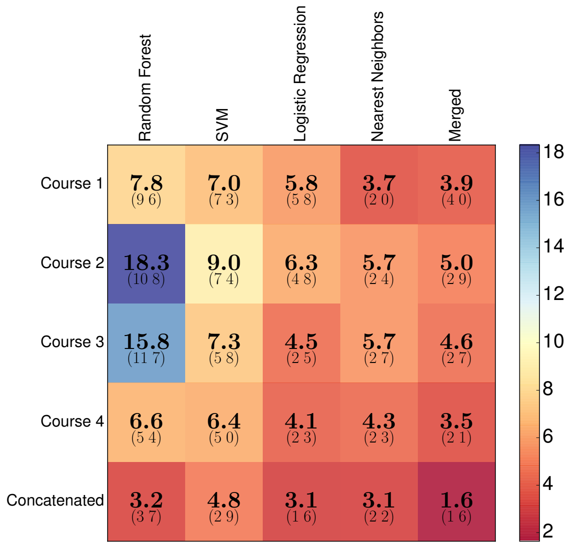

Figure 5: Average (standard deviation) of the DAUC (x100) for all predictions problems (P) on target course C_0. The x-axis contains different predictive algorithms, the y-axis contains different data source.

Optimizing the vote Structure:
The figures above show promising results for ensembling methods in the context of dropout predictions. This encouraged us to explore different ensembling methods to further improve the and performance and/or reliability of our dropout prediction system.
Our ensembling strategy uses all available estimators described above (those built on a single-source course as well as those built on the concatenated data). It then applies one of the manually pre-selected structures as shown in figure 4. We then use algorithm 1 to learn the structure.
Figure 6 displays the DAUC obtained by different ensembling structures according to algorithm 1. We differentiate our observations according to the subset of problems over which the average is computed (P,P_s and P_l).

- Over all problems (average over P), we first remark that the structure has only a very small impact on both the average performance and the variance of the predictive method. We also remark, however, that structure $S6$ yields slightly better results, both in terms of average DAUC and in terms of variance.

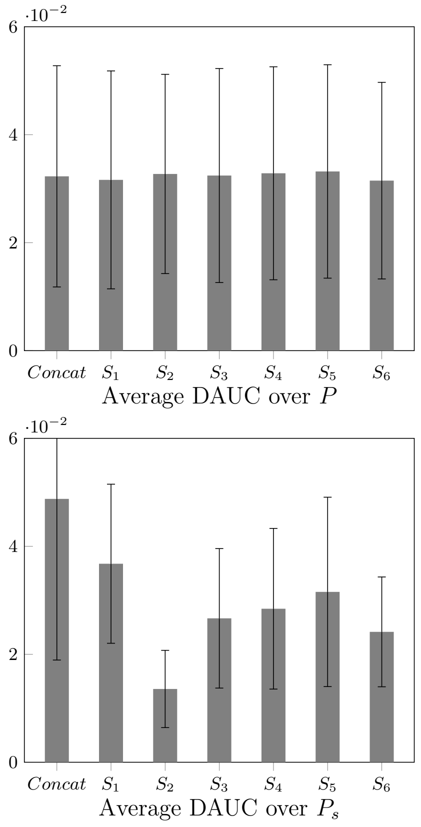

Figure 6: Average DAUC over each edX course taken as target, as computed by algorithm 1.

- Over the short-term problems, (average over P_s), we observe a lot more difference across the different structures. By far, structure $S2$ is the best performer for this type of problem, with a DAUC of 0.013 on average compared to 0.049 for the merged method discussed in the previous section.

- Over long-term problems (average over P_l) the difference between structures is significant, and thus not as big as for the short term problems. The best structure here is $S4$.

4.3 Transferring across MOOC platforms

Having achieved robust methods for dropout prediction on different predictive problems, we now test our method on a new set of courses, composed of 10 courses from the University of Edinburgh. Rather than testing this method on the holdout course as explained in algorithm 1, this set of courses present the additional difficulty of being derived from another MOOC platform (Coursera), thus having potentially very different statistics for the features used in our models. For example, overlap between the features of our 5 first courses (from edX) and the features of those new courses is not total. Whereas our initial 5 courses shared 21 common features, they share only 12 features with this new set of courses. In figure 7 we report the DAUC obtained on average over all possible prediction problems and over all ten

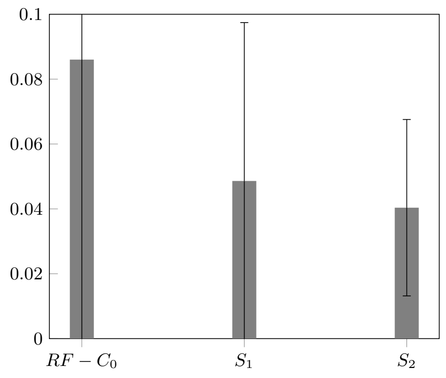

Figure 7: Average DAUC over all prediction problems on the 10 UDI courses taken as target (only edX courses are taken as source courses).

ID	Name	AUC short	AUC all
C_5	aiplan_001	0.82	0.75
C_6	aiplan_002	0.79	0.70
C_7	aiplan_003	0.81	0.74
C_8	animal_001	0.73	0.64
C_9	animal_002	0.75	0.67
C_{10}	astrotech_001	0.77	0.67
C_{11}	codeyourself_001	0.84	0.74
C_{12}	criticalthinking_1	0.71	0.63
C_{13}	criticalthinking_2	0.80	0.71
C_{14}	criticalthinking_3	0.78	0.70

Figure 8: AUC achieved by S_2 ensembling method built on the five edX courses and applied on the 10 UDI courses.

courses (taken as target). We display the results for a simple Random Forest algorithm built on the first edX course, for an ensembling method based on the S_1 structure, and finally for the best-performing ensembling method (from the experiment in the previous sub section) based on S_2 structure. All the ensembling methods are built on top of estimators from all the five edX courses (for the four algorithms : Rrandom Forest, SVM, Logistic Regression, Nearest Neighbor). Table 8 reports the absolute performance of the best technique (S_2) structure in terms of average AUC across different prediction problems for each course.
We remark first that the S_2 performs again significantly better than both the simple algorithm and the simple ensembling method. We also note that the absolute performance achieved by this best ensembling technique is relatively high, given the small number of features available and the different origins of the two set of courses.

5. KEY FINDINGS

Our key findings can be summarized in three categories, corresponding to the three sets of questions described in the introduction:

Purpose We showed that even though MOOC courses span different numbers of weeks and have different characteristics, one can usually find sufficient overlap between courses to perform nontrivial prediction tasks.

Data We showed that using more courses as training data improved the predictive power significantly. We also proved that this predictive power was sufficient to apply the model built on one particular MOOC platform to another platform.

Method First we showed that, both in the case of a single course model and in the case of a model built from several courses, using simple ensembling methods between algorithms significantly improved the performance. When compared to a single algorithm trained on all available courses, a simple ensemble methods improved the AUC by an average of 1.5 to 3.2 points. Secondly, we proved that in certain use cases (for instance, short term dropout prediction problems), using more complex ensembling structure can significantly boost performance. For short term prediction problems, using a S_2-like structure of ensembling resulted in no less than a 4 point AUC improvement on average.

Finally, our best method was successful when applied to a set of unseen courses. On the ten never-before-seen Coursera courses, our method obtained a 0.70 average AUC overall and a 0.78 average AUC on short term prediction problems (one week ahead). This completes our case that a high-performance predictive model can be built from a set of previous courses, and that ensembling methods appear to be a suitable framework to build such models.

6. DISCUSSION

When trying to estimate the actual benefit of such techniques on the real life of students and teachers on MOOC platforms, one has to make several assumptions that may only be verified after several years of implementation.
The main assumption is the possibility to reduce churn of

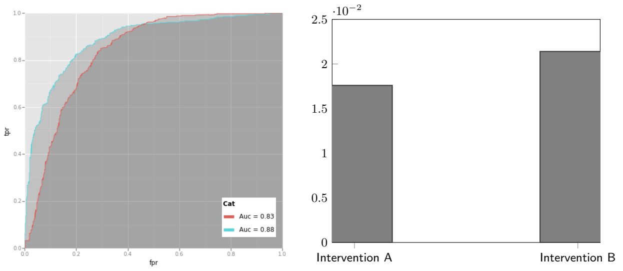

Figure 9: Estimated increase in number of students completing a typical MOOC course. Intervention A refers to an intervention based on a simple predictive model built on a course. Intervention B refers to an intervention of type S_2. See description for numerical assumptions.

students through personalized intervention. This is not obvious, as many argue that most dropout students were intrinsically not interested in the content of the class, and could therefore not be fruitfully intervened with. Most MOOC providers, however, agree that a good chunk of each cohort could be prevented from dropping out of the class thanks to some customized and well-adapted interventions. Identifying dropout students (the example describe in this paper) could enable a concrete set of interventions to be done, with extra resource help, additional videos or motivating resources particularly tailored to potential "dropout" students. For our purpose we will assume that a tailored intervention will save 1% of all potential dropout students.
Given a fixed false-positive rate, arguably necessary to design an intervention targeted for dropout students, the purpose of the predictive methods described above can be understood as the maximization of the true-positive rate: the ratio of predicted dropout students to the number of total dropout students.
Taking a weekly intervention framework, in which an intervention is conducted for potential dropout students at the end of each week, we showed in the previous section that ensembling methods (particularly the S_2 structure) were able to perform around 0.05 AUC point better than other more straightforward models (particularly an "all-algo all-sources" method). In figure 9, we show the example of two ROC AUC separated by 0.05. We remark that with a constraint of 10% on the false positive rate, we obtain a difference of around 20% in the true positive rate .
Given a typical MOOC class − 10 weeks long, starting with 10.000 students, and with a typical weekly dropout rate of 20% per week we display in figure 9 the simulated data of the number of students completing the course. When an intervention based on a straightforward predictive model is simulated, it increases the number of students finishing the course by around 1.7%, whereas an S_2 based predictive model would increase it by around 2.1% (an additional 50 student completions overall).

7. CONCLUSION

In this paper, we developed a framework to address the main challenges faced when applying predictive analytics to MOOCs: How to build models that transfer well across courses and platforms?
To do this, we used ensembling methods, as well as a broad range of algorithms and a rigorous training procedure. We explored different variations of these techniques and reported the results obtained on a first set of five courses from the edX platform. We introduced a novel performance metric, allowing for performance comparison across prediction problems and target courses. These results show that ensembling methods improved the accuracy of prediction, both on average and in terms of variance. We also showed that "stacking" (or learning metamodels on top of a set of base predictors) can significantly boost performance in the case of short term prediction problems.
Eventually, we tested the method developed in a first part (on the first set of five courses from edX MOOC platform) on ten courses from the University of Endinburgh MOOC platform. We reported the results obtained in terms of AUC and showed that the method developed performed very well on those new courses, too.
We argue that our paper demonstrates a robust framework to develop predictive algorithms that are transferable across online courses.

8. REFERENCES

[1] P. Adamopoulos. What makes a great mooc? an interdisciplinary analysis of student retention in online courses. 2013.
[2] S. Boyer, B. U. Gelman, B. Schreck, and K. Veeramachaneni. Data science foundry for moocs. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.
[3] S. Boyer and K. Veeramachaneni. Transfer learning for predictive models in massive open online courses. In *Artificial Intelligence in Education*, pages 54–63. Springer, 2015.
[4] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.
[5] S. Džeroski and B. Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004.
[6] S. Halawa, D. Greene, and J. Mitchell. Dropout prediction in moocs using learner activity features. *Experiences and best practices in and around MOOCs*, 7, 2014.
[7] I. Lykourentzou, I. Giannoukos, V. Nikolopoulos, G. Mpardis, and V. Loumos. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education*, 53(3):950–965, 2009.
[8] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.
[9] B. Poellhuber, M. Chomienne, and T. Karsenti. The effect of peer collaboration and collaborative learning on self-efficacy and persistence in a learner-paced continuous intake model. *International Journal of E-Learning & Distance Education*, 22(3):41–62, 2008.
[10] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. *arXiv preprint cs/0106040*, 2001.