Data Science Foundry for MOOCs

Sebastien Boyer, Ben U. Gelman, Benjamin Schreck, Kalyan Veeramachaneni
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA- 02139
sebboyer, kalyan@csail.mit.edu
bschreck@mit.edu, bgelman@gmu.edu

***Abstract*—In this paper, we present the concept of data science *foundry* for data from Massive Open Online Courses. In the *foundry* we present a series of software modules that transform the data into different representations. Ultimately, each online learner is represented using a set of variables that capture his/her online behavior. These variables are captured longitudinally over an interval. Using this representation we then build a predictive analytics stack that is able to predict online learners behavior as the course progresses in real time. To demonstrate the efficacy of the *foundry*, we attempt to solve an important prediction problem for Massive Open Online Courses (MOOCs): who is likely to stopout? Across a multitude of courses, with our complex *per-student* behavioral variables, we achieve a predictive accuracy of 0.7 AUCROC and higher for a one-week-ahead prediction problem. For a two-to-three-weeks-ahead prediction problem, we are able to achieve 0.6 AUCROC. We validate, via transfer learning, that these predictive models can be used in real time. We also demonstrate that we can protect the models using privacy-preserving mechanisms without losing any predictive accuracy.**

I. Introduction

Massive Open Online Courses (MOOCs) generate large amounts of very fine-grained data capturing student interactions with the online platform. Students, or learners, interact with the platforms in four different modes: they browse the course material; they make submissions for assignments, exams and lecture exercises; they interact and collaborate with each other via forums and/or other collaborative platforms (such as Google Hangout); and finally, they provide feedback to the course developers, instructors, and researchers via surveys. In some very special cases, they also participate in peer-grading.

As is typical for an online, large-scale platform, the data presented numerous challenges before we could even begin to develop predictive models. Our first analysis, which was limited to one course, implemented ad-hoc procedures for data collection, curation and data transformation and representation [13]. To analyze data at scale and for multiple courses, we took a different approach, addressing the challenges systematically and creating an end-to-end software for data science and predictive analytics. In this paper, we present an end-to-end predictive analytics platform, we call *"data science foundry"*. The foundry applies a series of transformations to the data starting from the raw data, making the data amenable for an analytics purpose after each transformation. Our specific contributions through this paper are:

The data science foundry: Our system transforms data from its raw form (log files) into a structured representation, then from structured representation to a per learner longitudinal variable representation. Then it transforms the data to make it ready to build predictive models and derive insights. At each step of the process we store the intermediary format thus allowing reuse. For example, per-learner longitudinal data could be used for multiple purposes. It could be used to train predictive models, perform correlative analysis. The *foundry*'s source code is open source[1]. This enables anyone with MOOC data to follow the steps in this paper and produce predictive models.

Design of curation and processing software: The first step in the *foundry* is to transform the raw data into the data schema we proposed. We created software that converts the raw data to this schema, and made it available as open source. We used this software to curate approximately 160 million log lines across multiple courses.

Per-learner longitudinal variables: Predictive models depend on good set of variables. We gathered ideas for variables via a crowd-sourcing experiment. As a result we assembled a list of longitudinal variables and extracted them on a per-learner basis. In total we extracted 18 variables on a weekly basis for 169,215 learners.

Predictive models for 6 different courses and insights: The next step in the *foundry* transforms the per-learner longitudinal data into features for a supervised machine learning problem. To demonstrate, we attempted to solve an important prediction problem: figuring out whether a student is likely to leave a course based on the click stream events generated by his or her interactions. Prediction of this event allows instructors, researchers and platform providers to design interventions that may reduce the dropout rates. Across 6 courses we were able to define 560 prediction problems and we trained models for each of them. Our predictive models successfully predict several weeks ahead.

Transfer learning: We are interested in using these predictive models in real time. We examined whether models learnt one course could be successfully applied to future

[1]The code is available at: https://github.com/MOOCdb/

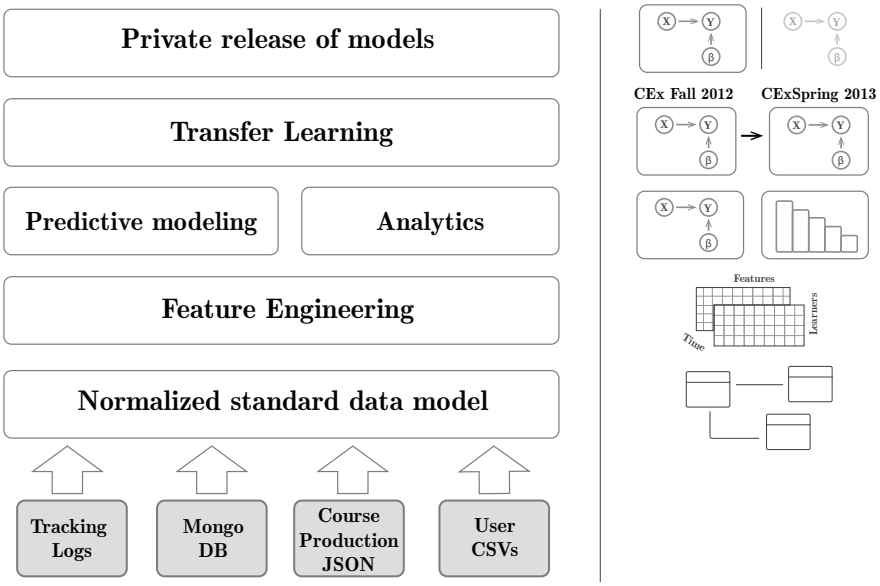

Fig. 1. A data science foundry for the data from Massive Open Online Courses. The data originally stored in multiple formats goes through a number of transformations. At the first step we convert the data into a standardized data schema. We then generate a number of quantitative descriptors characterizing the online learner's behavior (a process called feature engineering). Over this representation of the learner we are able to define a number of prediction problems, build and analyze models and present insights. Finally, to be able to use the models we employ transfer learning methods. Should the user want to release the models we provide privacy preserving mechanisms for releasing the models.

offerings of a course, or to entirely separate courses. We provide a transfer learning method to be able to adapt the models for new courses.

Private release of models and analysis: Finally, we developed a privacy-preserving mechanism that could allow users to release the models. Users can analyze the impact of privacy-preserving mechanisms on predictive accuracy. In our experiments we found that the models are still able to perform well.

We proceed by describing the data produced by the edX platform in Section II. We then present the challenges of *curating*, *conditioning* and *normalizing* the data, and how we addressed them. In Section III, we present the result of the complex process of feature engineering to develop a longitudinal representation of a learner. Given the longitudinal representation of learners, we next present the prediction problem we are interested in and the multiple versions of this problem in Section IV. In section V we present different types of models we built for the prediction problems. This section also summarizes the results achieved via predictive models for all 6 courses. We deduce which features mattered, and provide insights into our prediction problems in Section VI. In section VII we focus on how these predictive models could be used in real settings. We compare multiple methodologies that could enable transfer learning. Finally, in Section VIII, we present the privacy mechanisms we employed that will allow us to release these models for use, and analyze the affect of these privacy mechanisms on the models' predictive accuracy.

II. From raw log data to student trajectory

Figure 2 shows how a typical course on edX is organized. The course is split into multiple weeks; each week is a *unit*, and a *unit* consists of multiple *sequences*. When clicked on, a *sequence* displays a number of browsable *panels* that feature intertwined presentations of lecture videos and lecture exercises. Each lecture video is split into many smaller segments to enhance assimilation, as it is known that shorter videos are better assimilated by learners [5]. Often, exercises are intertwined with the videos, to give learners the opportunity to assess how well they understood the material. Figure 2 shows one such lecture exercise—in this case, panel number 3. The exercise is broken down into two distinct problems, which are identified internally as unique *modules*. At the bottom of the page, there is a link to the discussion thread that corresponds with this particular lecture exercise (in this case, the thread is named S6E0).

While the learners browse this hierarchy of resources and progress through the course structure, their data is recorded as a sequence of log lines. Figure 3 shows one such log line, describing when the learner navigates to the *panel* shown in Figure 2. In the log line, the module is represented by a *uri* (a unit usually not visible externally) and the *page* represents the *url*, often recorded at the *unit* or *sequence* level.

During a typical course, it is not uncommon to have hundreds of millions of these log lines detailing the learner's every activity on the platform. Our first challenge is to represent and describe this activity succinctly. We represent it via a learner trajectory, defined as a *sequence* of 5-tuples consisting of $< id, t, u, a, m >$, where id is learner id, t is the timestamp, u is a location of the learner's activity that contains the information regarding the hierarchy of the module as well as the module with which the learner is engaged, a is the action taken by the learner, if any, at this location in the course, and m is any metadata that corresponds to this activity. As with many online systems, while defining a learner trajectory is conceptually straightforward, assembling the trajectory from

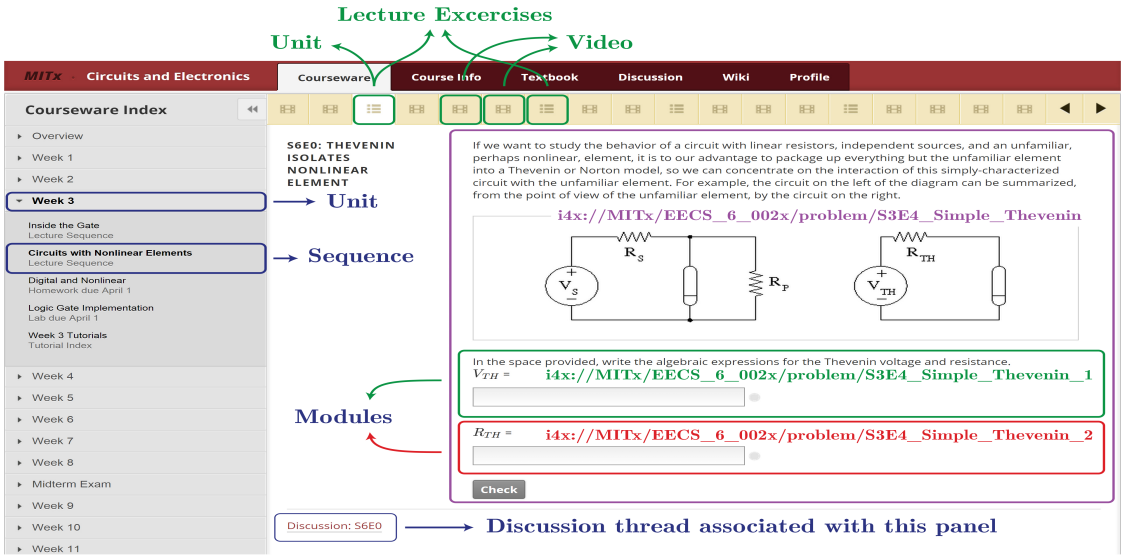

Fig. 2. A snapshot of a course structure as offered by edX. Hierarchy of different components is also shown.

the log data generated by this platform is not. Below, we explain how we conditioned and curated this data before building the trajectory.

Data conditioning: When working with this type of data, a few issues commonly arise. There may be missing `page` information or inadequate information about the hierarchy, (that is, the `page` information may be at the *panel* level or the *sequence* level rather than the modular level), or a *module* may be missing entirely. To reconstruct a coherent trajectory for the learner, we draw inferences using sequential log lines from the same learner. We may infer the *module* if it is missing, or add information to construct u such that information is captured at the right depth level. Figure 4 shows a few examples of such inferences. In the figure, the user attempts the same problem multiple times. However, in the second log line the *page* information is missing, and in the third log line the *module* information is missing. By knitting together these three consecutive log lines, we can infer the persistent location of the user.

Data curation: Once we have conditioned the log lines for the learner, our next step is to curate the data. A number of processing steps allow us to remove the redundant entries generated whenever a learner attempts a problem (such attempts are recorded in both browser and server events). Another step tags each `page` in log line with the type of resource it is; that is, a *book*, *video*, *lecture problem*, etc. This is achieved by creating a dictionary of names used for different resources as they appear in the *page url*, then comparing the page *url* in the log line to this dictionary.

Data normalization: Once we have conditioned and curated each log line carefully, we normalize the data into a set of tables, as documented in the work [14]. There are three tables, each of which stores information pertaining to one of three categories: *browsing*, *submissions* and *collaborative events* the learner engages in. We further refined a number of *meta-information fields* to accommodate more nuanced aspects of a digital learning ecosystem. These meta-information fields can capture detailed items, including deadlines for problem completion, whether or not the grading was done by peers, and release dates for different content. We then extended the schema so that data from a different platform, *Coursera*, could be translated as well without any loss of information.

These steps of conditioning, curation and normalization allow us to replicate the analytics we perform on one course for multiple courses across universities and platforms. Currently, about 100 course offerings are being translated using this software stack. We present results from a subsample of 6 courses that we have access to. Table I presents the details of the 6 courses we analyzed.

III. Representing the Learner Behavior

Our next step is to quantitatively characterize learners' online behavior from *web logs* and *click stream* data. First, we extract per-learner time sequences of click stream events. These sequences are primitive, but if they are formulated into *variables* that abstract learners' behavior via *feature engineering*, they could help to gauge learners' intent, interest, and motivation in the absence of verbalized or visual feedback. Two types of variables exist. They are:

Variables that capture per learner behavior with respect to a resource: For example, consider two variables such as: *total time spent of the video* and the *number of pauses while watching the video*. When these two variables are evaluated for all the learners and analyzed they can reveal patterns; if too many learners *pause* too many times, the video could be fast and/or confusing.

Per-learner longitudinal variables: A longitudinal study involves repeated observation of the same variables over time. A variable is usually an aggregate or a statistic of some observations defined for that time interval. In the context of the MOOC, we can define the time interval to be a *week*, a *day* or a *time corresponding to the module/unit* or divide the course

```
{
  "username": "John",
  "event_source": "browser",
  "event_type": "problem_check"
  "agent": "Mozilla/5.0 (Windows NT 6.1; rv:10.0.10) Gecko/20100101 Firefox/10.0.10"
  "ip": "128.230.212.64"
  "module_id": "i4x://MITx/EECS_6_002x/problem/S6E0_Simple_Thevenin"
  "page": "https://6002x.mitx.mit.edu/courseware/6.002_Spring_2012/Week_3/↩
     Circuits_with_Nonlinear_Elements/"
}
```


Fig. 3. An example log line as generated by user attempting to submit an answer to a problem.

Fig. 4. An example of how we link multiple log lines and fill in the missing information to get the complete picture of the learner engagement.

	2012		2013			
Course	6002x	3091x	3091x	1473x	6002x	201x
Abreviation	CEx12	CHx12	CHx13	GPx13	CEx13	ESx13
Start Date	09/05	10/09	02/05	02/12	3/3	15/04
Number of weeks	14	12	14	13	14	9
Number of students	51,394	24,493	12,276	39,759	29,050	12,243
Number of log lines	60M	40M	8M	34M	19M	6M

TABLE I

DETAILS ABOUT THE 6 MASSIVE OPEN ONLINE COURSES OFFERED VIA EDX PLATFORM. THE COURSES SPAN A PERIOD OF 2 YEARS SINCE THE INCEPTION OF EDX AND HAVE A TOTAL OF 169,215 STUDENTS/LEARNERS. THE COURSES ALTOGETHER GENERATED A TOTAL OF 165 MILLION LOG LINES.

into two periods - before and after midterm. Some example of these variables are: *How much time each student spent on the course website during the past week*, and *The average length of time before the deadline that a learner starts to work on an assignment*.

A learner can be quantitatively characterized in numerous ways. To generate ideas and to prioritize which ones we should extract first we sought the help of a crowd. The experiment and its results are described in detail in [15]. The resulting feature ideas that we incorporated in our software are presented in the Table II.

The outcome at this stage is the transformation of the click stream data in the schema to a temporal representation of the learner characteristics. This is given by:

$$< l_{id}, f_{id}, t_{id}, fValue >, \tag{1}$$

where l_{id} is the id of the learner, f_{id} is the id of the feature, t_{id} is the id of the time interval, and $fValue$ is the numeric value of the feature.

IV. FORMULATING REAL TIME PREDICTIVE PROBLEMS

Given the temporal representation of the learner in eq. 1 one can formulate a number of prediction problems, examine impact of a certain behavior (quantified by a variable) at time t_{id} on the future behaviors. Below we describe, how we can construct numerous prediction problems from the representation we detailed in the previous section. In general we can define a prediction problem as

$$l_{id}, f_{id}, t_{m+n} = \Psi\left(l_*, f_*, t_{1\ldots m}\right), \tag{2}$$

where Ψ is the predictive model, l_* is all the learners in the data, f_* represents all the features and $t_{1\ldots m}$ represents the time slices till m. We are trying to predict the value of a specific feature f_{id} n steps ahead for the learner l_{id}. To more precisely define the formulation of the prediction problems, we first define the temporal units for our data, and the concept of lead and lag.

Weeks as time units: Temporal prediction of a future event requires us to assemble explanatory variables along a time axis. This axis is subdivided to express the time-varying behavior of variables so they can be used for explanatory

TABLE II
LIST OF FEATURES PER STUDENT PER WEEK EXTRACTED

Feature number	Definition
1	Whether the student has stopped out or not
2	Total time spent on all resources
3	Number of distinct problems attempted
4	Number of submissions [1]
5	Number of distinct correct problems
6	Average number of submissions per problem (x_4 / x_5)
7	Ratio of total time spent to number of distinct correct problems (x_2 / x_8). This is the inverse of the percent of problems correct
8	Ratio of number of problems attempted to number of distinct correct problems (x_6 / x_8)
9	Average time to solve problems
10	Duration of longest observed event
11	Total time spent on lectures
12	Difference in average number of submissions per problem when compared to previous week.
13	Difference in ratio of total time spent to number of distinct correct problems compared to previous week.
14	Difference in Ratio of number of problems attempted to number of distinct correct problems compared to previous week.
15	Difference in average time to solve problems compared to previous week.
16	Number of correct submissions
17	Percentage of the total submissions that were correct (x_{16} / x_4)
18	Average time between a problem submission and problem due date over each submission that week
19	Standard deviation of duration of the events for the learner for that week

[1] In our terminology, a submission corresponds to a problem attempt. In MOOCs, students could submit multiple times to a single problem. We therefore differentiate between problems and submissions.

purposes. In the courses we consider, content was assigned and collected on a weekly basis, where each week corresponded to a module. Owing to the regular modular structure, we decided to define time slices as weekly units. Time slices started the first week in which course content was offered, and ended after the final exam had closed (which is usely after the fourteenth or fifteenth week). To precisely define the timing of our predictions, we introduce two new definitions

Definition 1: The **lead**, n, represents how many weeks in advance we will attempt to predict a feature value f_{id}.

Definition 2: The **lag**, m, represents how many weeks of historical variables will be used to build the predictive model. For example, if we use a lead of 5 and a lag of 3, we would use the first 3 weeks of data to predict 5 weeks ahead. Thus, each training data point consists of a learner's feature values for weeks 1, 2 and 3 considered as features. The f_{id} value for week 8 becomes the outcome value or label we want to predict. By this definition considering that we have k features, t time slices we can formulate a total of $k \cdot \sum_{m=1}^{t-1} t - m$ supervised predictive problems.

A. Example prediction problem: Stopout prediction

When we surveyed a variety of studies (including ours), a number of them posed the same prediction problem: could we predict in advance whether a student would stopout of the course? This knowledge would help stakeholders to design strategies for retaining the students that are most likely to stopout, and to present surveys in a timely fashion to understand the reasons for the stopout.[2]

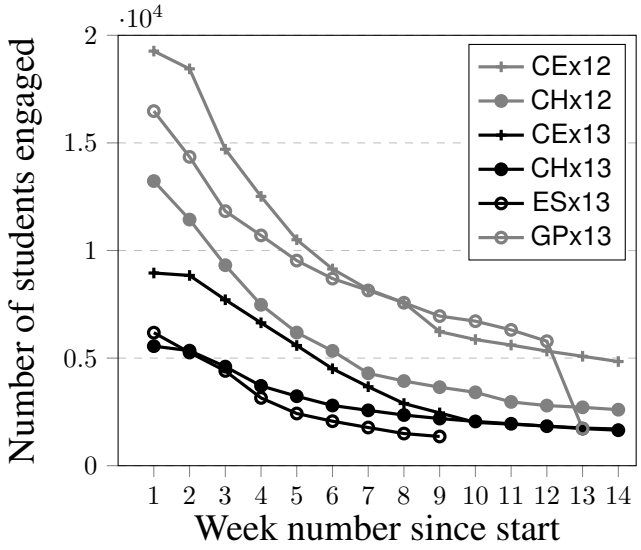

Fig. 5. Students who persist in a course as weeks pass by.

[2]The likelihood of a response to a survey reduces significantly after the student has left the course.

In every course, there is a steady rate of learners that drop out; indeed, only 7-8% of the learners remain by the end of the course, as can be seen in Fig. 5. However, it is hard to tell whether learners stopout because of how the course is delivered, or because it was never their intention to finish the course. Authors in [12] argue that one should analyze stopout rates only for those students who have expressed interest and strong motivation, traits assessed via surveys. In the absence of such surveys, perhaps modeling stopout could lead to some insights. Additionally, we hope that if we are able to predict stopout risk before the actual stopout occurs, customized interventions could be designed to help students.

Next, we had to pin down our definition of stopout. We considered defining it as the learner's last interaction in the course, regardless of the nature of this interaction[1]. Because this definition gives the same weight to a passive interaction (such as viewing a lecture, accessing an assignment, or viewing a Wiki) as it does to a proactive interaction (such as submitting a problem or taking an exam), instead, we used the definition provided by [13]:

Definition 3: The **stopout** time stamp (week) is defined as the last week a learner submitted an assignment or exercise.

Under this definition, 91 individual prediction problems exist for a course that spans 14 weeks. For any given week i, there are $14-i$ prediction problems. Each prediction problem becomes an independent modeling problem which requires a discriminative model. To build these discriminative models, we use the common approach of "flattening out" the data—that is, forming the covariates for the discriminative model by assembling features from different learner-weeks, considered as separate variables.

V. Predictive modeling

In the *foundry* we enable the user to choose from a number of predictive modeling methodologies.

Logistic Regression Logistic regression is a popular predictive model due to its ability to model relationships non-linearly and yet be easily interpretable. It calculates a weighted average of a set of variables, submitted as covariates, as an input to the *logit* function. The input to our *logit* function, z, takes the following form:

$$z = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ...\beta_m * x_m \tag{3}$$

Here, β_1 to β_m are the coefficients for the covariates values, x_1 to x_m. β_0 is a constant and the *logit* function, given by $y = \frac{1}{1+e^{-z}}$. For a binary classification problem, the output of the *logit* function is the estimated probability of a positive training example. We train the model iteratively via maximum likelihood estimation. A testing set comprised of untrained covariates and labels evaluates the performance of the model on the following test data, via calculation of area under the ROC curve.

A. Results for stopout prediction problem

We built logistic regression based stopout predictive models for all the courses presented in Table I. By accounting for different lead and lag, we built a total of 560 models across the courses.

Figure 6 presents the results for all 6 courses for varying lead obtained with the built-in logistic regression algorithm from the *sklearn* machine learning library. As expected, the predictive accuracy decreases as the lead increases. However, in most cases, we are able to maintain 0.6 or above AUC for the prediction problem with the highest lead. In all the courses, it is very easy to predict one week ahead [3]. We get more than 0.7 AUC for a 2-3 weeks ahead prediction problem. The error bars on the line plots represent the deviation for the same lead when applied at different weeks during the course. The large variance in some cases suggests that the prediction problem is harder when the course is at a certain point. Anecdotally, we have found that it is often easier to predict stopout after half the term has passed.

VI. Deriving insights

In data driven discovery, one of the common questions one asks is "What variables were important in predicting a certain outcome?". To generate an answer for this we use randomized logistic regression. We detail the approach below and also present ways to generate insights for multiple variables, multiple courses, and multiple time points simultaneously.

Randomized Logistic Regression We use a randomized logistic regression to assess the importance of features. Our model uses 18 features to model stopout. To assess the importance of the features, a randomized logistic regression repeatedly models a perturbed data set (subsample) with regularization, and works as follows:

- Step 1: Sample without replacement 75% of the training data each time (the variables are normalized ahead of training).
- Step 2: Train a logistic regression model on the sub-sampled data, with a randomized regularization coefficient for each variable. The randomized coefficient β_j is sampled from uniform distribution $[\lambda, \frac{\lambda}{\alpha}]$, where $\alpha \in (0, 1]$, and λ is the regularization coefficient usually used in standard regularized regression approaches. This randomization applies different selection pressures for different variables.
- Step 3: For every covariate, evaluate $b_s^j = \mu(w_j, th)$, where μ is a unit step function, w_j is the coefficients for covariate j, and th is the threshold we set to deem the feature important. When this threshold is set at 0.25, this results in a binary vector, which represents the selection of the covariate. This binary vector is $(lag \times |features|)$ long, where 1 at a location j implies feature $i = j \mod 18$ was present in this model.
- Step 4: Repeat Steps 1, 2 and 3 a total of 200 times.
- Step 5: Estimate the importance of the covariate j by calculating the selection probabilities $\frac{\sum_s b_s^j}{200}$.

Evaluating feature significances: Let $c \in \{1, ..., C\}$ be the set of courses, $p \in \{1, ..., P\}$ be the set of prediction problems for course c (formed by varying lead and lag compatible with the length of c), $w \in \{1, ..., W\}$ a week number (usually we have that $w = lag$) and $f \in \{1, ..., F\}$ be a feature index.

[3]This problem is currently being solved as part of KDD-cup.

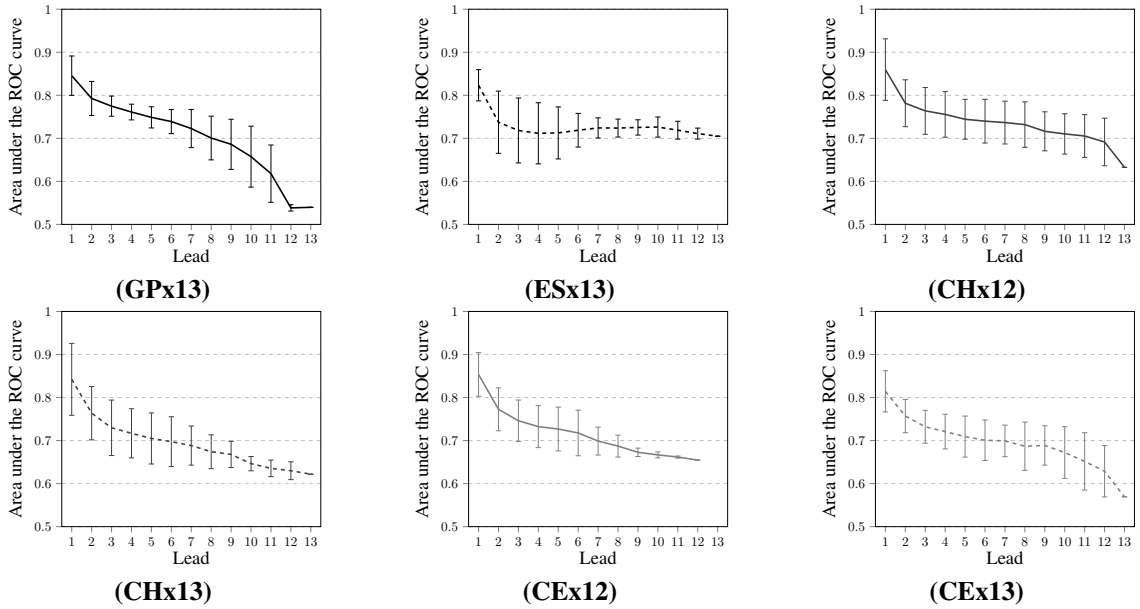

Fig. 6. Area under the curve vs. the lead for all six courses under study. As lead increases the AUC decreases. The error bars show the variance in the AUC when the same lead is applied at different weeks. Most courses have reasonably good AUC for one week ahead.

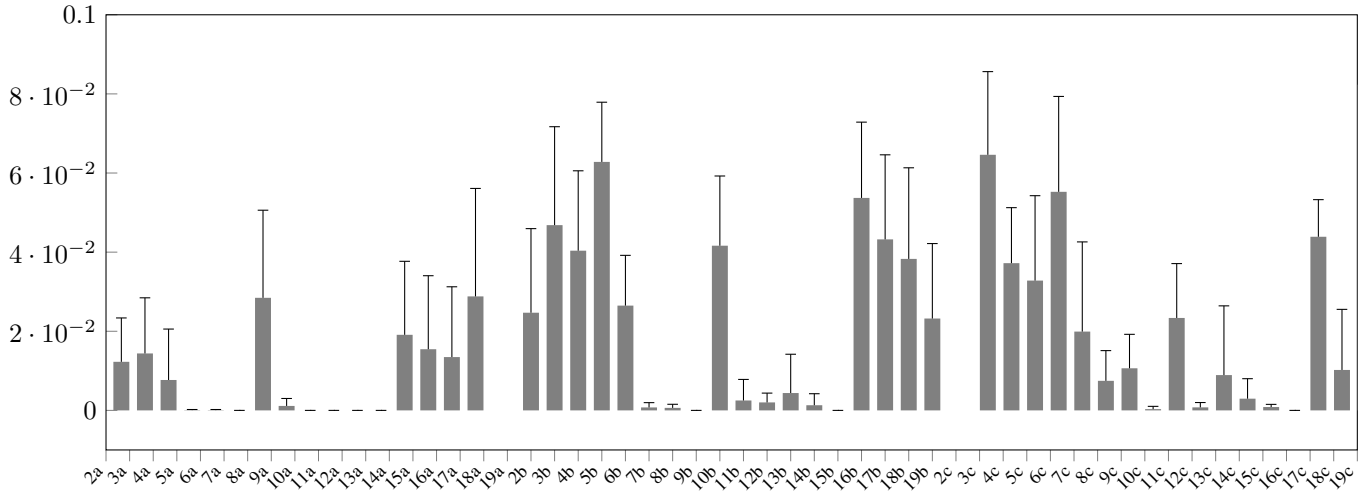

Fig. 7. Feature importance values for variables for 3 weeks when week 4 is targeted for prediction. Variables with suffix 'a' are week 1 features in order as per the Table II, variables with suffix 'b' are week 2 features, and variables with suffix 'c' are week three features in the same order. The features of the last week are most important in the prediction. We observe this in all the prediction problems, *that is*, most recent features matter more.

After running our randomized logistic regression algorithm we are left with the following quantities for each course $c \in C$, and each problem p, $I_{i,w}^{c,p}$ = importance of feature i of week w. First we want to evaluate the relative importance of the feature for each course. We therefore control for the influence of time and prediction problems' particularities by normalizing the importance for each week.

Definition 4: For all courses c and all features $f \in F$, we evaluate the **feature significance** of feature f for course c, via the following quantity:

$$\hat{I}_f^c = \frac{1}{|P|} \sum_p \frac{1}{|W|} \sum_w \left(\frac{I_{f,w}^{c,p}}{\sum_{f' \in F} I_{f',w}^{c,p}} \right)$$

Next, we want to evaluate the influence of time on feature importance. We first fix the week number K for which we want to evaluate the importance, as we will only consider the importance over the K last weeks of each prediction problem (this way we don't consider problems with $lag < K$). To control for courses and problems' particularities, we normalize the importance for each course and problems over the K weeks of interest.

Definition 5: For an integer K, a course c, a week index $z \in \{1, ..., K\}$, $c \in \{1, ..., C\}$ and a feature $f \in \{1, ..., F\}$, we evaluate the **temporal feature significance** of feature f

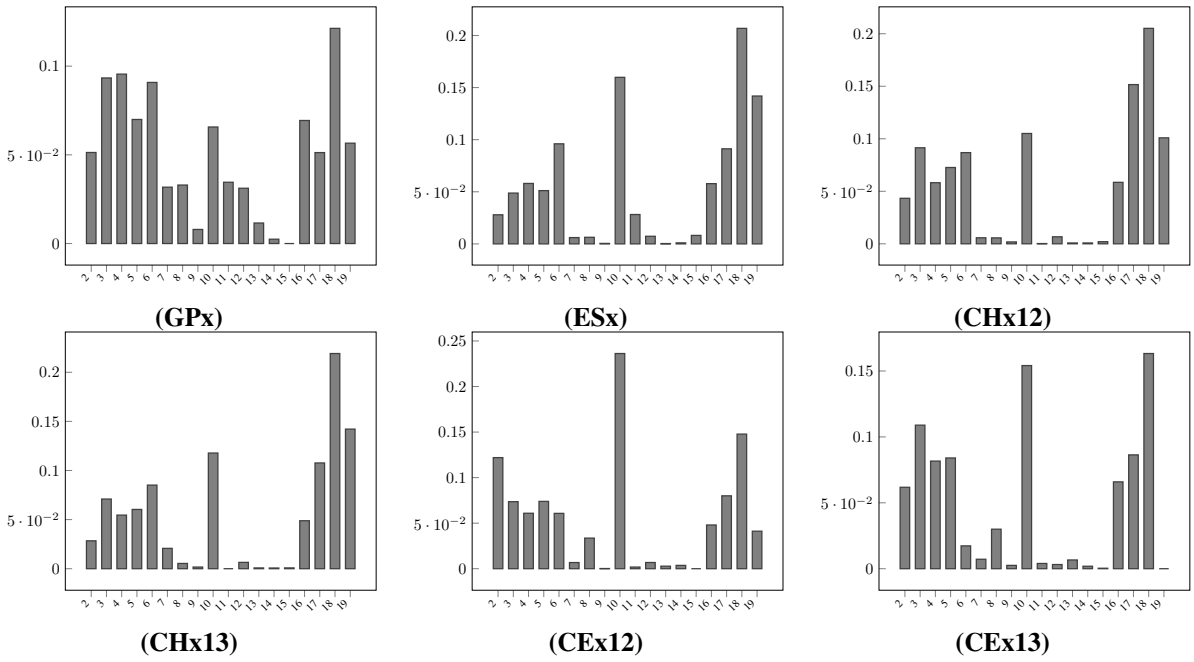

Fig. 8. Feature importances, per course and across all prediction problems. Most important feature is the pre-deadline submission time which measures how much in advance does the learner starts to attempts the problems in the week.

on the zth week by calculating the following quantity:

$$\mathring{I}^{c,K}_{f,z} = \frac{1}{|P|}\sum_{p}\left(\frac{I^{c,p}_{f,W-K+z}}{\sum\limits_{f'\in[1,F],w'\in\Omega(p,K)} I^{c,p}_{f',w'}}\right)$$

where $\Omega(p,K) = [W-K+1, W]$ is the set of K last weeks of the prediction problem.

Finally, it is interesting to aggregate the two above-mentioned quantities to derive across courses' generalities. We therefore introduce the notion of aggregate significance to capture this information.

Definition 6: We evaluate **aggregated feature significance** of feature f as the mean of the feature significance of f for all courses. Similarly, we evaluate **aggregated temporal feature significance** of feature f as the mean of the feature temporal significance of f for all courses.

A. Results achieved using randomized logistic regression

For each of the 560 prediction problems across 6 courses we built 200 logistic regression models resulting in an overall 112,000 models. We derived feature significance values using the methods we presented above.

Figure 7 presents the feature importances achieved for a lag=3 and lead=4 prediction problem across all courses. The bars represent the *aggregated temporal feature significance*, and the error bars represent the standard deviation across courses. We summarize our findings with the following remarks:

- The *temporal feature significance* increases when we get closer to the present. That is, a feature from far in the past matters less than the same feature closer to the prediction time of interest. These results can be seen in Figure 7.
- The *feature significance* differs a lot between features. In figure 8 we show the feature importance for different features across different courses. In almost all courses, how long before the deadline the students start attempting problems matters. Additionally, the total time spent on the course matters, while the amount of time spent specifically on books, wikis, and lectures does not matter. In our experience, a variety of stakeholders find these insights very valuable, as they allow for customized design of interventions.

VII. Towards real time use of the predictive models

One of the foundational questions we tackled is, "are we able to use this model to predict stopout in the following courses?" To answer this question, we employ transfer learning approaches. Below, we present two simple approaches, which we will call *offering S* (for source offering) and *target offering T* (for current ongoing courses).

A naive approach: When using samples from a previous course to help classification on a new course, we first wonder if the two tasks (classifying in the first course and classifying in the second course) are similar enough that applying the model learned on the first course to the second one will yield satisfying results. To answer this question, we trained a logistic regression model with an optimized ridge regression parameter on S, and applied it to T. Figure 9 shows the AUCs achieved by this approach for two different problems. They are for 1-week-ahead (left) and 4-week-ahead (right) prediction problem (note that when assessing the model on the same course we used the same ridge factor as well as cross validation methods so that the auc is a test auc as well).

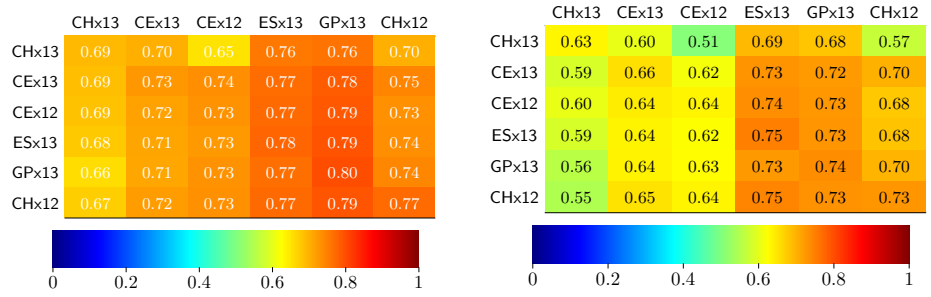

Fig. 9. Performance of a naive transfer learning approach. A 6-by-6 matrix, where i, j shows the AUC achieved when course i's model is used in course j.

After trying the simplest approach—transferring the model directly from one course to another—we wondered if the level of error could be decreased by using more sophisticated approaches that would adapt the model to each new course. This motivated the use of the approach below.

Transductive learning approach: To transfer the model from source to target, we followed our belief that the covariates from the two courses may overlap to a significant degree, and used a sample correction bias [7]. This importance sampling, or IS, method is equivalent to assuming that the covariates are drawn from a common distribution, and that their difference comes from selection bias during the sampling process (out of this general common distribution). In order to correct this sample selection bias, the idea is to give more weight to the learners in the source course that are "close" to the learners in the target course. In this way, the classification algorithm takes advantage of learners in each course that are similar to each other, and barely considers the significantly different ones. For each target sample we predict : $\hat{y}_{Ti} = \arg \max_{y \in \{+1,-1\}} l(x_{Ti}, y, w^*)$. The weights are estimated from the source data by optimizing:

$$\text{where } w^* = \arg\min_{w} \sum_{i=1:n_S} \beta_i \, l(x_{Si}, y_{Si}, w)$$

Note that each learner's data is re-weighted using a parameter β_i in the log likelihood objective function. Finding the optimal β_i for such a re-weighting procedure would be straightforward if we knew the two distributions from which the source and the target learners' data is drawn. To find these in practice, we use a Gaussian kernel

$$k(x, z) = \exp(-\frac{||x - z||^2}{\sigma^2}) \; \forall x, z \in S \cup T$$

to measure the distance between data points. We then compute an average distance to the target domain for each source data point: $\kappa_i = \frac{1}{n_T} \sum_{j=1:n_T} k(x_{Si}, x_{Tj})$ and use a quadratic formulation found in [7] to evaluate the optimal coefficients β_i^*:

$$\beta^* \sim \arg\min_{\beta} \frac{1}{2} \beta^T K \beta - n_S \, \kappa^T \beta s.t. \; \beta_i \in [0, B] \qquad (4)$$

and

$$\left| \sum_{i=1:n_S} \beta_i - n_S \right| \leq n_S \epsilon$$

where $K_{ij} = k(x_{Si}, x_{Sj})$. The objective function is the formulation of the discrepancy between the two empirical distributions' means, the first constraint limits the scope of this discrepancy, and the second constraint ensures that the measure $\beta(x)P_T(x)$ is close to a probability distribution. In our current experiments we note that importance sampling based method did equally well as the naive method.

VIII. Releasing the Models

In the previous sections, we have shown that it is possible to transfer the model from one course to another, either by using a *naive* approach or by calibrating a source model for the new data. Let us now consider that we want to release our models so that users from multiple universities and different platforms can utilize the model to make predictions.

Releasing the model (i.e. the coefficients learnt by the logistic regression algorithm) may raise privacy concerns. Even though it is difficult to reconstruct personal data from only the given coefficients, one can find very sensitive information about a particular student by using both this model and data from other students. This is particularly important when the course (or the prediction problem) has only a few "close" students, as someone could infer information about a subset of students using only their data and this model.

To address the concern, we resort to differential privacy. For logistic regression, a provable ϵ- differential privacy mechanism that vastly reduces the risk of re-identification is offered by [3]. We refer interested readers to [3], [4] for more information about privacy-preserving mechanisms. To achieve privacy, we incorporated ϵ as a measure of the "privacy," and we changed the optimization problem of the logistic regression by adding a "privacy term" to the optimization problem, so that it becomes:

$$w^* = \arg\min \frac{1}{n} \sum_{i=1:n} \log(1 + e^{-y_i . w^T . x_i}) + \frac{1}{2} \lambda w^T . w + \frac{b^T . w}{n}$$

where b is a random vector chosen as follows: We generate d random numbers from a normal distribution centered on 0 and set them as the components of vector b. We then generate a random number from a Γ distribution with parameter d (the number of features) and $\frac{2}{\epsilon}$, and set it as the norm of vector b (dividing all the components by their aggregate sum and multiplying by this new positive number).The resulting direction of the vector follows a uniform distribution over the

angles in the d dimensional space (to interested readers, we suggest [3] for a mathematical proof).

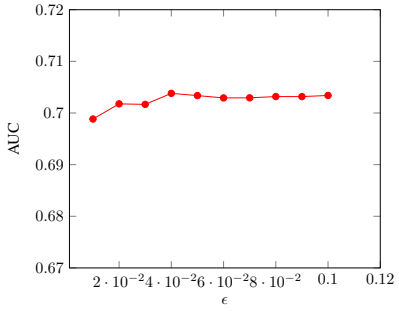

Fig. 10. Effect of ϵ on the predictive accuracy. Lower the ϵ higher the privacy protection. We trained the model on $CHx13$ with different ϵ and tested the model on the course $CEx13$. The performance of the model does not deteriorate much. For a small values of $\epsilon = 0.005$ (high privacy constraint) the AUC goes down to 0.698 compared to the AUC of 0.703 when the model is directly used without perturbation.

We solve the resulting optimization problems for different values of the privacy parameter ϵ on the problem of predicting week 4 using the first three weeks. The results are displayed in Fig. 10.

IX. Related work

Prior to MOOCs, in [8], the authors proposed the combination of three classic algorithms to build a predictive model for stopout within the first e-learning experiments. Focusing on distance education, [10] uses students' non-behavioral data (age, gender, financial status) to predict dropout. For behavioral data prior to MOOCs, prior academic performance has proved to be the highest predictor of stopout [9]. Other studies focused on time invariant variables such as *attendance at class orientation* [16]. The features used in this work are time-varying as in the work of [8]. These studies differ significantly from ours due to the type of data now accessible in the context of MOOCs.

In this context, the study of factors relating to persistence has been of great interest due to high non-completion rates. Using the first MOOC class on edX, [2] provided insightful statistics on the nature of MOOC students and their behavior on the platform. In [11], the authors studied the effect of collaboration on the likelihood of dropout. They used a logistic regression model and achieved 90% accuracy (retrospectively). We designate this as a correlative (and not predictive) model because completion of the final exam was used as an explanatory variable.

More recent work has focused on predictive analytics. The closest work to ours can be found in [6] and [1], both of which use binary variables in an attempt to predict dropout one week in advance. Our work distinguishes itself by the broadness and the flexibility of our framework, and particularly by how we overcame the constraints on the choice of the week to predict the features that could be used. Additionally, we showed that the model could be both reused for new courses, and publicly released without privacy concerns.

Note: We also want to mention the recent competition launched by the SIGKDD: KDD-cup 2015, which deals with predicting stopout in MOOCs. As we write this article, the competition has about 200 competitors. We make three encouraging remarks. First, the popularity of this contest (as well as the problem chosen) shows that there is significant interest in predicting stopout. Second, the organizers of the competition (with whom we haven't been in contact) use the data from an edX platform and use the same AUC metric as we did. Third, as of the day we write this article, the "best" AUC achieved by the competitors for the problem proposed (a 10 days ahead prediction) is similar to the score reached by our methods. The competitors get 0.89, while we achieve between 0.90 and 0.85 for most of the courses. Finally, we emphasize that our work focuses on a large flexible automatic prediction framework for MOOCs, which differs greatly from the aim of the competition's organizers.

References

[1] G. Balakrishnan and D. Coetzee. Predicting student retention in massive open online courses using hidden markov models. In *Technical Report No. UCB/EECS-2013-109*. EECS, University of California, Berkeley, 2013.

[2] L. Breslow, D. E. Pritchard, J. DeBoer, G. S. Stump, A. D. Ho, and D. Seaton. Studying learning in the worldwide classroom: Research into edxs' first mooc. *Research & Practice in Assessment*, 8:13–25, 2013.

[3] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *The Journal of Machine Learning Research*, 12:1069–1109, 2011.

[4] C. Dwork. Differential privacy. In *Encyclopedia of Cryptography and Security*, pages 338–340. Springer, 2011.

[5] P. J. Guo, J. Kim, and R. Rubin. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 41–50. ACM, 2014.

[6] S. Halawa, D. Greene, and J. Mitchell. Dropout prediction in moocs using learner activity features. *Proceedings of the European MOOC Stakeholder Summit (EMOOCS 2014), Lausanne, Switzerland*, 2014.

[7] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2006.

[8] 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.

[9] G. Mendez, T. D. Buskirk, S. Lohr, and S. Haag. Factors associated with persistence in science and engineering majors: An exploratory study using classification trees and random forests. *Journal of Engineering Education*, 97(1):57–70, 2008.

[10] A. Parker. A study of variables that predict dropout from distance education. *International journal of educational technology*, 1(2):1–10, 1999.

[11] 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.

[12] J. Reich. Rebooting mooc research. *Science*, 347(6217):34–35, 2015.

[13] C. Taylor, K. Veeramachaneni, and U.-M. O'Reilly. Likely to stop? predicting stopout in massive open online courses. *arXiv preprint arXiv:1408.3382*, 2014.

[14] K. Veeramachaneni, S. Halawa, F. Dernoncourt, U.-M. O'Reilly, C. Taylor, and C. Do. Moocdb: Developing standards and systems to support mooc data science. *arXiv preprint arXiv:1406.2015*, 2014.

[15] K. Veeramachaneni, U.-M. O'Reilly, and C. Taylor. Towards feature engineering at scale for data from massive open online courses. *arXiv preprint arXiv:1407.5238*, 2014.

[16] A. Wojciechowski and L. B. Palmer. Individual student characteristics: Can any be predictors of success in online classes? *Online Journal of Distance Learning Administration*, 8(2), 2005.