




 [Caellwyn](#) / [ou_student_predictions](#)

<> Code

 Issues Pull requests Actions Projects 1 Wiki Security master ▾

...

[ou_student_predictions](#) / README.md

Caellwyn Update README.md ...

 History 1 contributor

Raw

Blame



183 lines (98 sloc) 18.6 KB

Predicting Student Success Using Virtual Learning Environment Interaction Statistics

Directory:

[Exploratory Data Analysis](#)[First Simple Model](#)[Model Development](#)[Data](#)[Collection of Model Candidates](#)[Various Assorted Charts](#)[Working Environment](#)[Presentation](#)[Report Notebook](#)

The Problem



image by Goran Ivos, courtesy of [Unsplash](#)

Online learning has been a growing and maturing industry for years now, and in a way the internet has always been about learning. [Khan Academy](#) is, to me, a quintessential step in self-driven learning and massive open online courses (MOOCs) like [Coursera](#) create an even more structured approach to online, self-drive learning. I took my first online course on coding fundamentals from [edX](#) in 2013.

While these are an amazing resource, they also have a high dropout and failure rate. Many of us, accustomed to the accountability features in traditional face-to-face learning, struggle with the open-ness of these systems and the lack of a person keeping us accountable. It can be easy to fall through the cracks and fade away.

I set out to find ways to use data to help online courses improve student success by identifying students who are in danger of failing or withdrawing early. My theory is that students in need of special intervention to improve their chances of success can be identified by the way they interact with the online learning system itself.

This project seeks to create a model of student interactions with self-paced learning modules that can predict student outcomes after the first half of the course. My goal is to make this model generalizable by removing correlations with specific courses and instead capturing the underlying functions that can predict the success of students in any self-paced online learning environment. I realize this is a lofty goal, but I believe it is possible to do to some extent. I believe that many students who are struggling will exhibit particular patterns of behavior in how they interact with the learning environment.

The Data

I used a dataset from [Open University](#) based out of Great Britain. They are an entirely online university offering accredited degrees to distance learners. They released anonymized data from seven courses, three from the social sciences and four STEM courses, from the 2013 and 2014. These courses are repeated twice per year and each repetition is called a 'presentation'. Not all presentations of each course are included in the dataset. To learn more about how the data was selected and anonymised, please visit this [US National Institute of Health](#) site that holds the data for scientific inquiry (like mine!)

This dataset contains regional, demographic, and personal data about students, but I was only interested in data about how they interacted with the learning environment. In the StudentVle.csv file available in the [anonymiseData.zip](#) file the university provided information about each activity that each student interacted with, including the date and number of clicks on that activity.

Features for Predictive Modeling

I did not use most of the data in the dataset as it was very specific to those students and the region. My goal is to capture the relationship between student behavior and success, not their background, gender, disability, or socioeconomic situation. These were also not highly correlative to student success anyway when I explored them.

Instead I used 6 features that a virtual learning environment would be able to collect anonymously:

Features:

Activity Statistics

1. Number of total clicks during the presentation of the course.
2. Number of total activities completed

3. Number of days worked

It was the last statistic that I felt would be uniquely predictive. I spent 14 years teaching students and one thing I learned from research and experience is that spreading learning out over more days, rather than cramming more learning on fewer days, helps with persistence and with learning retention. It helps with persistence because it causes the student to form a habit of learning. We are creatures of habit. Cramming a lot of learning into fewer days is also more exhausting and less pleasant, creating a sense of aversion to the act. Frequent reinforcement of learning also serves to reinforce neuronal axons that are the physical manifestation of learning. I'll show what I found on this later on in this notebook.

The next statistics I pulled to use to train my model were the actions on assessments.

Assessment Statistics

1. Average assessment score
2. Number of assessments completed.

The first was an obvious choice, and the second is meaningful because these courses were in part self-paced. I found that doing more assessments earlier was correlated to greater success, as you will see in the data exploration section later.

Course Repetition

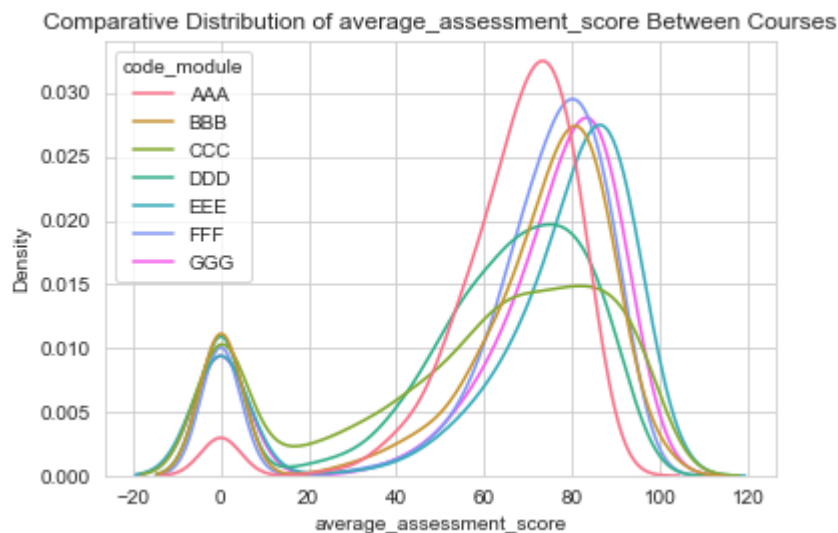
The final statistic I took was the number of times the student had repeated the same course. To be completely honest this is not data directly from interactions, and is in fact a sort of meta-data. However it should be available in any learning environment that tracks student histories across courses and is correlated to success.

Correlations in the Data

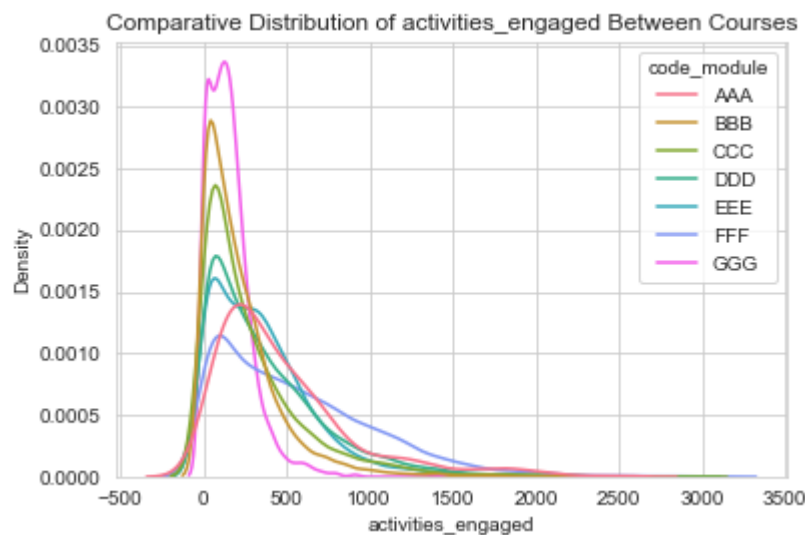
Repeat Students After preparation I had data on 22,424 individual students who registered for classes 24,743 times. The data from the Open University dataset was organized into registrations so that is how I organized my data. I recognize that those 2,319 registrations that represent multiple registrations by the same student violates the independence of observations, and would invalidate inferential modeling. When I do that kind of analysis I will remove the duplicates.

Courses (code_module)

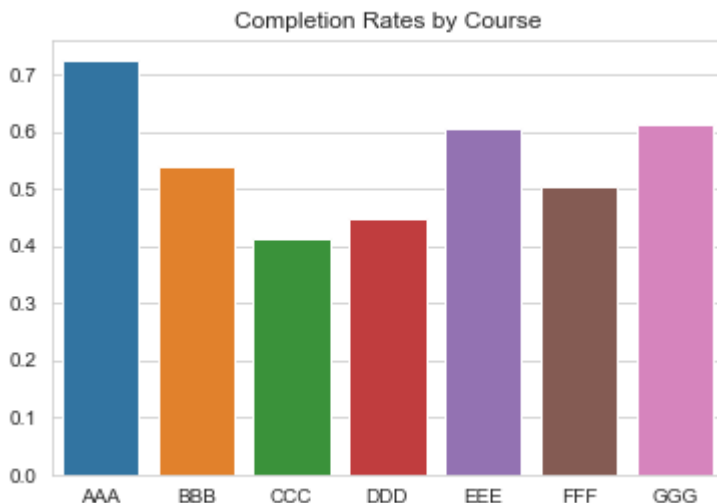
Unfortunately, I had bigger problems. The data is divided into 7 different courses with different numbers of activities, students, success rates, and average assessment scores. I want my model to be generalizable to many online courses outside of this dataset and to capture underlying relationships between student behavior and their success in a course.



You can see above that there are significantly different distributions of assessment scores for different courses.



Similarly, you can see that students engaged different numbers of activities depending on the course.



Finally we see a worryingly high difference in completion rates between the courses.

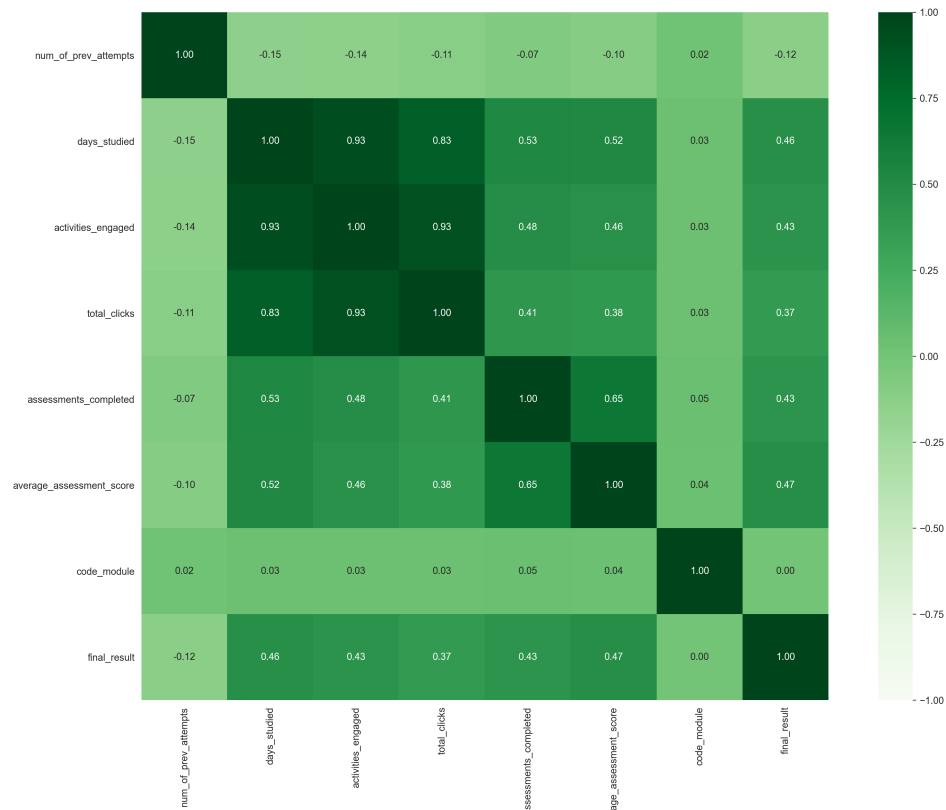
These correlation would make my model accuracy vary by course, my model's ability to predict success specific to these courses, rather than applying broadly

In order to correct this problem, I took two steps:

1. I scaled the features of each course separately to that course. Activities completed, assessment scores, etc. are now all relative to the mean for the individual courses.
2. I used [Imbalanced Learn's SMOTE class](https://imbalanced-learn.org/stable/generated/imblearn.over_sampling.SMOTE.html) to create synthetic data for the minority class in each course to balance the success rates. This method creates data which is statistically similar to the data in that class using a K-Nearest Neighbors approach.

The data features and targets were then no longer correlated to the courses. My model could focus on the behavior itself, albeit relative to how other students perform in the same course.

Variable Correlations



The above chart shows correlations between each variable my model uses for predictions, as well as course (`code_module`) and `final_result`. If you look along the row for `code_module` you will see that, for the most part, the correlations between courses and features have been removed, and the correlation between courses and `final_result` has been completely removed.

If you look down the final result row you can also see how strongly each of the aggregate features I extracted are to the final result.

1. Days Studied: .35
2. Activities Engaged: .33
3. Total Clicks: .27
4. Assessments Completed: .33
5. Average Assessment Score: .43

I was very excited to see `days_studied` with such a high correlation. My hypothesis from the start was that spreading learning out over more days would improve persistence and learning and I see validation here.

You can also see that each of these statistics are also highly correlated to each other, especially `days_studied`, `activities_engaged`, and `total_clicks`. These variables all represent how much work a student puts in and it's very reasonable that they would correlate. The number and score of assessments is strongly, but less well correlated, which is interesting to me. BTW, the number of assessments completed varies by student because students can work ahead on assessments.

These correlations between predictor variables are a problem if I try to do inferential statistics on these features. I would not be able to untangle which variables are actually affecting the results and how, since one changes the other. If I wanted to dig deeper into inference I would have to choose just one of these features to explore and maybe bring back the demographic features that I dropped.

Data Preparation

I've discussed some of the preparation I did, but here is a more comprehensive description:

1. I limited the data to what makes sense for the prediction window, or the points in the course I want to make my predictions on. For one, I won't try to predict the outcomes of students who have already withdrawn. I also won't use any data that I would not yet have by that point in the course.
2. My data target classes are 'Distinction', 'Pass', 'Fail', and 'Withdrawn'. However, my predictor only predicts whether a student needs intervention or not. Those passing or passing with distinction do not need interventions, while those failing or withdrawing do. I will combine the first two into a new class 'No Intervention' and the second two into 'Needs Intervention'. This makes my target binary, easier to predict and easier to evaluate.
3. I will scale all the data course by course. The `CourseScaler` class will fit on each individual course, as indexed by the `'code_module'` variable. It will take the mean and standard deviation for each feature for each course from the training data and use these means and standard deviations to scale each feature for each individual course $((\text{feature} - \text{mean}) / \text{standard deviation})$ for both the training and the testing data. This will remove the correlations between `code_module` and the activity and assessment statistics.

4. I will use the `smotecourses()` function to apply [imblearn.over_sampling.SMOTE](#) to the training data to create synthetic data within the minority class to balance the classes. Each course in the training set will then have equal number of students needing intervention and not needing interventions (as far as the model knows for training purposes). This will remove the correlation between `code_module` and `final_result`.
5. Once those are done, I can drop the `code_module` column, because I don't want my model considering that in its predictions.
6. Finally, of course I split the data into training data and testing data to validate my models and identify and prevent overfitting.

Model Development

Custom Tools

My preprocessing needs are unique and don't work well with off the shelf Sci-Kit Learn and Imbalanced Learn pipelines, cross-validation, and grid search tools. My preprocessing steps need the `code_module` feature in order to scale and balance each course separately, but that feature should not be passed to the model. Because of this I had to code my own versions of each of these that would fit a scaler to each course in the training set, use that fit scaler to scale both the training and test set, and then use the Imbalanced Learn SMOTE sampler to balance the classes in only the training set. Once I had these coded my model development could proceed.

Logistic Regression

I used a logistic regression model for a baseline model. A true baseline would be a random guessing model with a 50% accuracy, but an untuned logistic regression model was already able to achieve a 76.7% accuracy on both average cross-validation scores and the validation set. A logistic regression model seeks the best fit line to model the linear relationship between the predictor variables, X , and the target variable, y , which is a linear regression. It then applies a sigmoid function to that line to assign probabilities that each observation belongs in one class or the other. For my purposes, if the probability of an observation belonging a class is greater than .5, then I will predict that it belongs to that class. I used my custom `cross_validate` function to remove the course bias while preventing overfitting.

Using the custom gridsearch tool to optimize the model's regularization hyperparameters only resulted in a .001% gain in accuracy, probably due to the fact that the model was not suffering from overfitting in the first place.

Decision Tree

The logistic regression model was pretty successful, but I thought, perhaps the problem was not as linear as that model likes. A decision tree is a promising candidate for this problem because it does not assume the independence of the features or linear relationship between features and target. Instead it seeks to find the best way to divide and subdivide the data in a tree structure based on the values of different variables. Once the tree is built, predictions are made by sending an observations down the tree on a path to the predicted class as it reaches each split in the tree is sent in one or the other direction. You can think of it like a deterministic Pachinko machine!

This model averaged 77% on cross-validation scores, but only 71% on the validation set. Looking closer, there was more variance in the cross validation scores than with the logistic regression. I thought it might be overfitting the data.

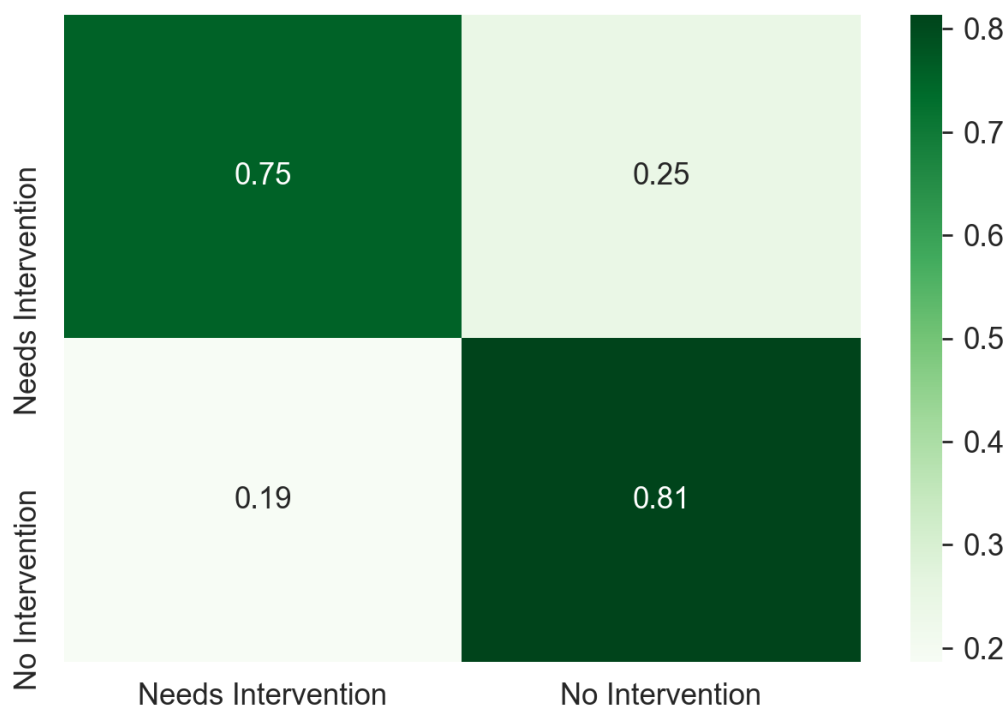
Random Forest

This is an interesting extension to the decision tree model. It creates a whole forest of decision trees and trains each one on a subset of the data and a subset of the features. This is a technique called bagging, or [Bootstrap AGGregation](#) (check the link for more on this). It works on the principle that a bunch of bad predictors, on average will be more accurate than one good predictor. This worked for Francis Galton in [guessing the weight of an ox](#), maybe it will work here!

In fact, this bagging strategy reduced the overfitting problem, achieving a consisted 76-79% accuracy on cross-validation and the average cross-validation score and the accuracy on the validation set were 77.7% and 77.5% respectively. This shows that while the bias is still a little high, we've corrected the overfitting that the decision tree was prone to.

Final Model: eXtreme Gradient Boosting

XGBoost models have gained a lot of popularity recently and won a lot of Kaggle competitions. It uses another popular idea called [boosting](#). That's a pretty involved wikipedia article, but the TLDR is that it's a similar ensemble method like random forest, but whereas random forest trains a bunch of trees in parallel and takes the aggregate of their predictions, boosting stacks the trees on top of each other and each one tries to improve on the one below it by predicting where the previous one made mistakes. I think of it as like a line of poor students each grading the next one's paper, which is an analysis of the previous one's paper. Each one gets a lot wrong, but something right so the right answers percolate through and some of the wrong answers get corrected at each step.



The XGBoost model outperformed the others with a 79% accuracy on both average validation scores and the test set. It correctly identifies 75% of students who are on track to fail the course while only incorrectly flagging 19% of those who are on track to pass. The bias is lower than the others, and there is no overfitting. This bodes well for predicting on new data.

Future deployment

This model can be useful in flagging students for intervention in online courses. However, preprocessing data is crucial. In order to properly scale new data the CourseScaler transformer would need to be fitted on at least one entire course worth of data, or would at least need to be provided with the mean and standard deviation of each feature for that course. It does not lend itself to web deployment, but could be integrated into the backend analytics of an online education provider to help target students for support.

Summary:

I built an eXtreme Gradient Boosted tree-based model that can, at the halfway point of an online course, determine with almost 80% accuracy who does or does not need additional intervention to pass. I worked to make the model generalizable to new data from new courses and new learning systems by scaling data by course and removing course-based bias from the training data. This can be useful to online education organizations to automatically flag students needing additional help or intervention.

Next Steps:

1. There are many more model types I can try with this data. I could use an SVM to attempt to draw a hyperplane to divide my datapoints into the two classes, or I could use deep learning networks to take a more abstract approach to solving the problem.
2. I could try larger prediction window and focus more on failing students than withdrawing ones, or a smaller prediction window.
3. I would love to try this model and preprocessing routine out on new datasets to see how well it generalizes.
4. I could try removing some of the outliers.
5. Finally, I could dig deeper into the errors my model is making to see if I can see what trips it up. what are the commonalities among students that are misclassified