MOOC Dropout Prediction with Model Stacking

Qi Zhao and Kangcheng Wei

Team: QIQI

Abstract. We are asked to predict the probability of the event that a student will drop out a course. We firstly extracted many features from the huge dataset. Then we used ensemble learning machine and model stacking technique to get the final result, which ranked the 1st in 68 teams.

Keywords: Kaggle, Machine Learning, Feature Engineering

1 Data Preparation and Feature Engineering

First of all, let's describe the dataset. Each user ID can choose multiple courses and thus has multiple enrollment ID's. Each course may have multiple enrollment ID's.
The dataset comes in 2 parts: the Log Activity table records all the activities and the time when the activities happened within each enrollment ID; the Enrollment ID table contains all the enrollment ID's user ID's and course ID's.

1.1 Data Observations

We firstly observed all the tables. There are total 120542 enrollment ID's in the two tables, of which the first 72325 are our training set and the rest are the test set. Since there is no missing data, all the observations can be fully utilized.

1.2 Feature Engineering

Events Firstly, there are 7 variables (access, discussion, navigate, pageclose, problem, video, wiki) representing 7 distinct activities in the Log Activity table. Since the matrix will be too sparse if we simply see each unique time as a feature, we count the frequency of each activities as 7 features of each. (We built a simple Logistic Regression model after this which scored about 0.35.)

Time and Dates The time and date data are also important, thus we used a package in R ("chron") that can help us extract the duration of the dates and times.
In the final training data, for the certain user, we got these information: "totaldays" means the number of days between the users first activity and last activity; "presentday" means the number of days that the user had at least one activity; "absentday" means the number of days that the user had no activity; "hours" means the total number of hours that the user are studying; "endmonth" is the month that the last activity occurred.
We also made many other attempt. So we add these variables: "holidays" is the times that the user studied during holiday like Christmas, Thanksgiving, etc.; "weekday" and "weekend" record the number of days that the ID studied during weekdays and weekends.

Course Information Next we look at the Enrollment ID table. At first glance, this table looks messy so that we didn't utilize this data. However, it turns out to be very important in our result. Here are the features we extracted:
From the variable "Course ID", we got the variable "coursepopu" which means the number of students that enrolled in this course. From the variable "User ID", we got the "coursenum" which means the courses that a certain user enrolled in.

Merging the new features with the real label, we can get another two variables: "droprate" which means the drop rate of one certain course in the training set and "avgcdr" which means the average drop rate of the courses that the certain user enrolled in.

1.3 Data Processing

After we obtained the information that we want, we need to process the data using some techniques like Box-Cox or Square Root transformation.

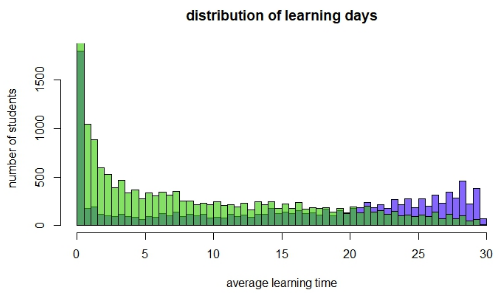

Fig. 1. The distribution of the learning days for dropout student and finished student

2 Model Building and Stacking

While building the model, we mainly used the "Scikit-Learn" package in Python which plays an important part in this project. Since the evaluation score that this competition use is Log-Loss, we can simply compute the approximate Log-Loss of our models so that we don't waste our limited submission chances, which turns out to be surprisingly accurate because of the huge volume of the dataset.

2.1 Single Models

Since all our variables are numerical rather than factors or characters, we tried almost all the commonly used machine learning models.

Logistic Regression As mentioned above, our first logistic regression model failed. The logistic model is a traditional statistical model for predicting probability for a binary event, which should be suitable for this project. However, the logistic model did not give us a good result.

The reason that this model failed is pretty obvious: we didn't have enough features at that time. However, after getting more independent variables, although improved, the result does not seem to be so satisfying according to the high Log-Loss. Therefore, we conclude that the possible reason might be the logistic regression model is not ideal when handling the imbalanced dataset.
However, it boosted our result in the next stage, which will be explained in detail in the next section.

Supported Vector Machine Supported Vectored Machine (SVM) is one of the most historical, but also most commonly used machine learning models in supervised learning. What it does is that it map the data into higher dimensional space. As what we thought, it performs slightly better than the Logistic regression when using the Gaussian Kernel trick, but the Log-Loss of 0.33 is not a significant improvement comparing to those ensemble methods like Gradient Boosting Machine (GBM).

Boosting Methods Boosting methods involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. However, it also tends to be more likely to over-fit the training data.
We used the Adaboost, the GBM and the popular XGBoost method, which all yields great results, with an accuracy of about 87% and a Log-Loss of about 0.31.
It is not difficult to see why they performs better especially on imbalanced dataset. They tend to give the misclassified data more weight in the training process. Besides, the XGBoost model outperforms other ensemble methods because it has a regularization part in its loss function, which can reduce the risk of over-fitting a model.

2.2 Model Separation

The core technique in this report is that we actually built this model for two different parts of data. Since some users enrolled in multiple courses, and some of their enrollment ID are actually included in the training set, which gives us more information.
For example, a user chose 28 course and among the 28 courses 21 are actually in the training set. In the 21 courses, 20 courses were dropped by the user, this definitely gives us more information in this case. However if the user in the test only enrolled in one course or all the courses the user enrolled in are in the test set, the information is not useful anymore.
Therefore we separate the data into two parts. We decide to put the enrollment ID into the first new training and test set if the user of this ID has more than 5 courses in the training set; the rest are put into the second new training and test set. For the new training and test set we have 2 new features which are the course that occurs and dropped in the old training set. After that, we used the

same technique on two data set and simply merge the result with the enrollment ID.

2.3 Model Stacking

Having read many articles, we learned that model stacking is a model ensembling technique that can generate a new model from multiple predictive models. The technique is straight-forward: first we choose several machine learning algorithms and generate some results; then, we use these results as the predictors and build a new parameter.
In the first step, we chose 9 models including GBM, XGBoost, SVM, Neural

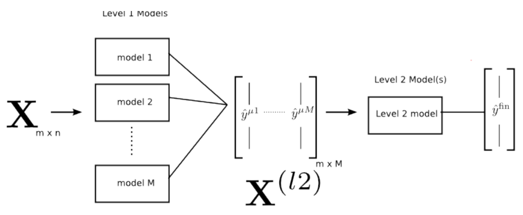

Fig. 2. The model stacking steps

Network, etc. Most are simple and relatively unrelated model. And in the second stage we used those results as new predictors and built a new Logistic Regression Model. In the real dataset, we got a Log-Loss score of about 0.29 and the accuracy is about 0.89, which is a significant improvement of previous result.

3 Results

Each time we built a model, we also see the result of the confusion matrix and analyze the True Negative and False Negative Rate, which can give us a more detailed idea about how the models perform and how to improve the model.
We try to increase the result based on the nature of the Log-Loss. According to the calculation method, we can try to reduce the Log-Loss by making the probability more extreme when we are very confident.

4 Summary

Our team has been cleaning the data and submitting results since day one which gives us a huge advantage considering most teams only started to submit the results in the last two weeks. Also, we have learned a lot.
First, we learned many great techniques in feature engineering, like how to treat the time and date data which is important in using the log data.

Secondly, through this process, we get a big idea of how to manage a data mining project and we are more familiar with the machine learning models, their advantages and disadvantages.
Although we did well in this project, we feel that there are still many things that we wanna learn. For example, we still don't know how to use the order of each activity as a feature, which may contain a lot of information.

References

1. Ben Gorman: A Kaggler's Guide to Model Stacking in Practice (2016)
2. Ali K M Pazzani M J : Error reduction through learning multiple descriptionsM (1996)