Report: Milestone 2

Hou Yumeng, Alexis Jacq (Team $\mathbb{M}$)

December 10, 2015

1 Data Description

Similar to the previous dataset we had for milestone 1, we have one dataset about 5478 users' events (behaviours) on MOOC within 4 weeks, and another data file containing their final grades. Our mission is to predict the students' grades based on their behaviours.

2 Feature extraction

2.1 Week Labelling and Aggregation

Firstly, we labelled the events in different weeks, and take them as individual features.
Then, we applied aggregation for users' events during the four weeks. We aggregate and harvest the number of Unique behaviors & Total behaviors (based on EventID).

2.2 Exploring New Features

2.2.1 Normal Feature Selection

We added new features of Unique *Video.∗* events (*Load, Play, Seek, Pause, SpeedChange*), Unique *Forum.Load*, Unique *Problem.Check* and Total *Problem.Check*, most of which have around or above 0.5 correlation coef with final grades.

2.2.2 Exploration Combinations As New Features

We found the the proportion of submission times: $Unique.Problem.Check/(Total.Problem.Check+1)$ helps in the prediction, so we took it as a new feature.
The interactions between solving problems and watching videos or launching in forum, as shown below, are correlated with the grades. So we added these features.
$Problem.Check * Video.Load$
$Problem.Check/(1 + Forum.Thread.Launch)$

2.2.3 Dealing With Submission TimeStamp

We separated the submissions of each student into 4 weeks. In each week, we calculate the average submission TimeStamp / Maximum TimeStamp (of the week). If one didn't submit for a quiz, we take the maximum TimeStamp. After getting the average submission time in each week, we calculated an average of these four results and take all these 5 values as new features. We defined a new feature which normalizes the timestamps with the number of submissions given by *Unique.Problem.Check*, in order to focus on the speed of students to submit:
$Speed = SubmissionTime/(1 + Unique.Problem.Check)$

3 Normalization

Data normalization could be done by PCA and down-sampling/up-sampling, which were quite helpful to improve the results in training with sophisticated models. But it proved no help in simple linear regression. Hence we discarded it in our solution.

4 Model and Parameters

We tried a variety of models including NNET, GBM, LM, RF, SVM, LDA, etc. We tuned the parameters accordingly based on the features of each model. We compared the performance mainly based on RMSE method but also the points got on KAGGLE. Though we kept improving slightly with better parameters (adjusting tuneGrid function). We finally surprisingly found that the simplest linear regression model with good features outperformed other "advanced" models greatly. We used stepwise approach by AIC to remove useless variables and to train our linear model.

5 Outliers Removal

We removed outliers after a first linear regression, which performed better than removal in the beginning. We used the distance between predicted grades derived from the first iteration and real grades, and removed the 0.1% farthest points from the training set and re-applied a second linear regression.

6 Visualization: Performance (RMSE) Comparison

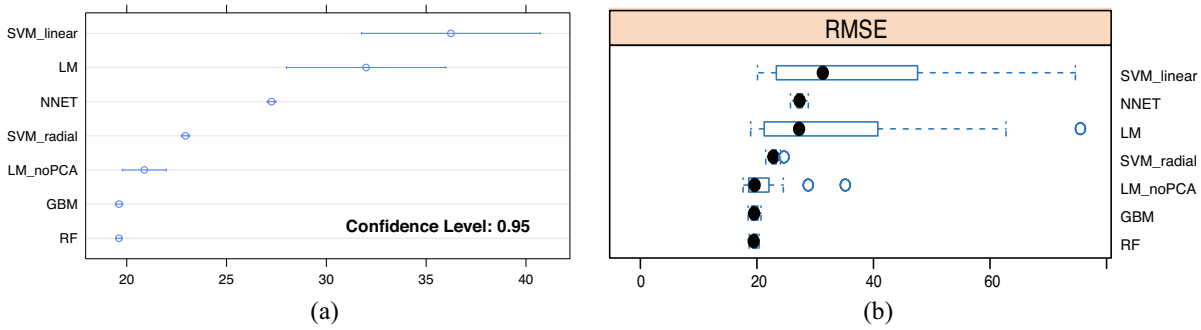

Figure 1: Dotplot(a) and Boxplot(b) for RMSE of different models.

As shown in Figure1, we compared the models we used in terms of RMSE. When roughly tuned, random forest and GBM outperform the others. Linear Regression without PCA is also efficient but not so stable as the two models above. PCA proves no help for simple linear regression. Complex models such as SVM and NNET don't work well in this case and they're hard to tune as well.