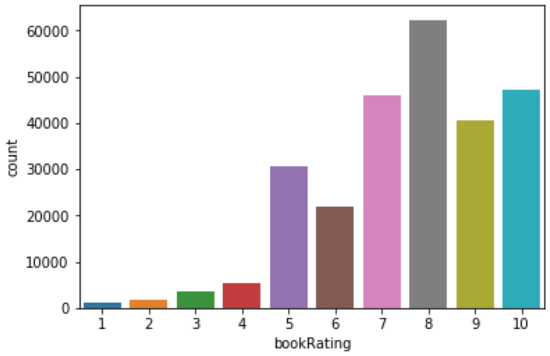
Machine Learning Techniques final report

-ELTeam

Data Prepossessing

- *Step 1: Read in all features of users and books but drop some features that are URL of image of books because we think they're less useful for prediction.*
- *Step 2: Make all values have their own "correct" values and data types. For example, year of publications of the books have some values that contain strings which look like names of publisher so we need to correct them and change the data type of feature to integer. Moreover, other features such as users' locations contain a lot of mistakes or various words actually have the same meaning (USA, America…), so we manually fill in a few missing values and modify others by coding.*
- *Step 3: Crate dummy variables, or say one-hot vectors of some features like location (country) and age. For age, we first categorize them to the range of age class and then create their dummy variables. For example, a 24-year-old user's age will be categorized to the age class of 21~30.*
- *Step 4: Read in train data, check its format and plot a simple figure that shows counts of each rating score.*
  At this point, a simple popularity based recommendation system can be built based on count of user ratings for different books. Besides, we find that books authored by J.K. Rowling are most popular based on this criterion.



- *Step 5: Calculate mean, standard error and number of rating (include explicit rating), grouped by users and books because we think that if a user tends to give higher rating on average in the past, he or she may give higher in the future too. Thus we make it another feature of the user so do standard error and count.*
- *Step 6: All features are combined together and the training data is ready.*

Three Models chosen

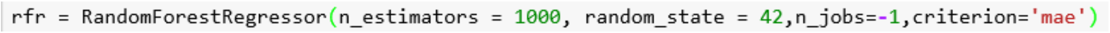
- Model 1: Random Forest Classifier

```
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42,n_jobs=-1)
```

This task is a classification problem, so we choose Random Forest Classifier intuitively. Moreover, because features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data, height of trees and not bootstrapping. To get deterministic behavior, we set random state a fixed number 42. The criterion measures the impurity here is gini.

- *Efficiency:*
  The running time of this algorithm is highly depend on the number of trees and size or height of them. After several times of attempt, we decide to use 1000 trees and let each tree fully grown. The combination of parameters takes about twenty to thirty minutes every time running.
- *Scalability:*
  As for scalability, random forest is easily to be parallelized so has good scalability because several parts of the model can be done independently. We can directly parallel construction of the trees and the parallel computation of the predictions through setting the parameter n_jobs.
  If n_jobs=k then computations are partitioned into k jobs, and run on k cores of the machine.
- *Popularity:*
  Due to the good classification performance, scalability, and ease of use, random forest gains a huge popularity in machine learning.
- *Interpretability:*
  Random forest originally be designed to mimic decision making of human. Thus it naturally has good interpretability for us.

- Model 2: Random Forest Regressor



```
rfr = RandomForestRegressor(n_estimators = 1000, random_state = 42,n_jobs=-1,criterion='mae')
```

Though people usually use regression to predict value that is continuous, rather than class. We still give it a try and think it may give better prediction due to more meticulous output (a real number). After we get the real number output, we round it to the nearest integer which is final prediction. Here we still use 1000 trees and set random state 42. Besides, different from classifier, regressor gets other ways of measuring the quality of a split: mean square error (MSE) and mean absolute error (MAE).

- *Efficiency:*
  Similar to random forest classifier, the cost of time significantly depends on the number of estimators and whether tress are pruned or not. Here, we still set the number of trees 1000 that is same as random forest classifier. However, random forest regressor takes several times of time than classifier. We think it may due to the different ways of measuring impurity. As mentioned above, the functions to evaluate the quality of split are
  MSE:

$$\mathrm{MSE(Data)} = \frac{1}{N}\sum_{n=0}^{N-1}(y_n - \bar{y})^2 \text{ where } \bar{y} = \text{average of } \{y_n\}$$

  and MAE:

$$\mathrm{MAE(Data)} = \frac{1}{N}\sum_{n=0}^{N-1}|y_n - \bar{y}|$$

  Compare to GINI:

$$\text{GINI(Data)} = 1 - \sum_{k=1}^{K} \left(\frac{\sum_{n=1}^{N} [\![y_n = k]\!]}{N}\right)^2$$

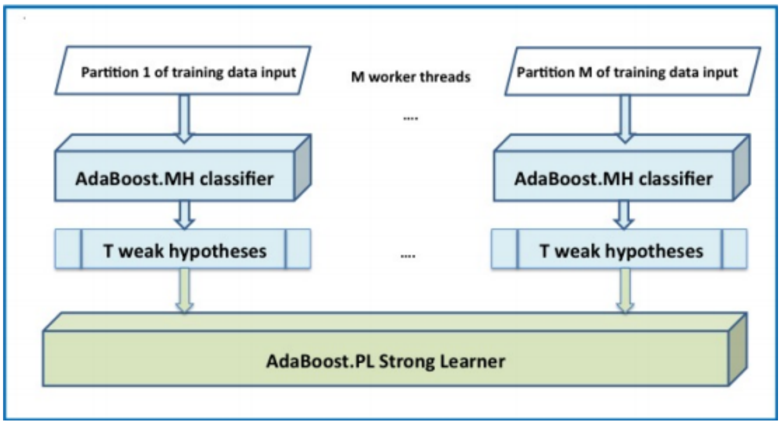
- *Scalability:*
  This algorithm is also a part of random forest family so several parts of the model can be done independently and thus has excellent scalability too. Like random forest classifier, we may parallel the construction of tree and computation of prediction value by setting n_jobs.
- *Popularity:*
  Share the name and the virtues of random forest, this model also has plenty of advocates. Sometimes it may get higher popularity than random forest classifier.
- *Interpretability:*
  The overall idea is same as random forest classifier, only different in impurity function and output value. This model has good interpretability too.

Model 3: Adaptive Boosting Classifier

```
ada = AdaBoostClassifier(n_estimators = 1000, random_state = 42, )
```

Third model we choose is adaptive boosting classifier which is another meta algorithm taught in class. The basic idea of adaBoost is to make several weak learners properly cooperate and give a better result. The base estimator we use here is decision tree. We train 1000 trees and set random state 42 that same as above two models.

- *Efficiency:*
  This model takes the least time among these three models but gives the worst prediction accuracy.
- *Scalability:*
  AdaBoost is scalable according to the paper by Darshan Thaker, Prabhat Nagarajan: A Scalable and Parallel Implementation of the MultiBoost Library's AdaBoost.MH (Adaptive Boosting) Algorithm. The picture of their parallelization idea is:

  (this picture is from the paper mentioned above)
- *Popularity:*
  The popularity of this model has once high but get lower recently. It's fewer and fewer people do research on this algorithm so do implementation. We

think it may because there’re more and more powerful models that give a good result and thus the popularity of adaBoost declined.

- *Interpretability:*
  The base estimator we use is decision tree which is Interpretable. Therefore, this model has good interpretability too.

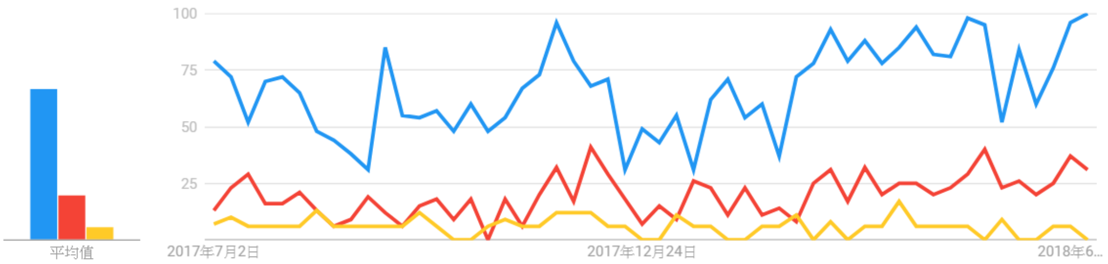
Comparison

- *Comparison table:*

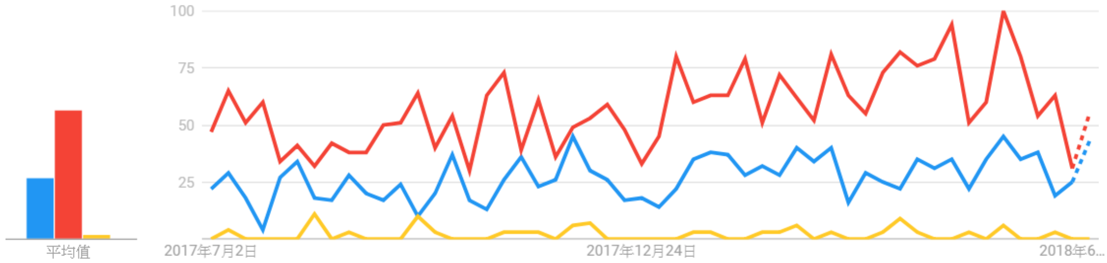
CRITERITION\MODELS	*RFC*	*RFR*	*ADA*
EFFICIENCY	*20~30 mins*	*Whole afternoon*	*5 mins*
SCALIBILITY	*High*	*High*	*Fair*
POPULARITY	*High*	*High*	*Low*
INTERPRETABILITY	*Good*	*Good*	*Good*

- *Additional popularity comparison on google search:*

(blue: RFClassifier, red: RFRegressor, orange: ADA)



(blue: RFClassification, red: RFRegression, orange: ADA)



Recommend approach:

For both track, we recommend

Model 2 Random Forest Regressor

- *Pros*
  This model gives us the best result on out-sample error and thus is the most accuracy model among these three.
- *Cons*
  It takes a lot of time to train these model. As mentioned above, the training process takes at least whole afternoon, if the criterion is set to be mae then it takes even longer.

Work loads for team members:

Due to the different free timeslots of us,

陳伯駒 郭笋宏 *are primarily responsible for data preprocessing and training of these three models*

張如嫻 *is primarily responsible for organizing the overall flow and writing to the report*