RMIT Classification: Trusted

Experiments on Beer Recommendation System

IDENTIFIERS

- Zhen Qian
- S3888611

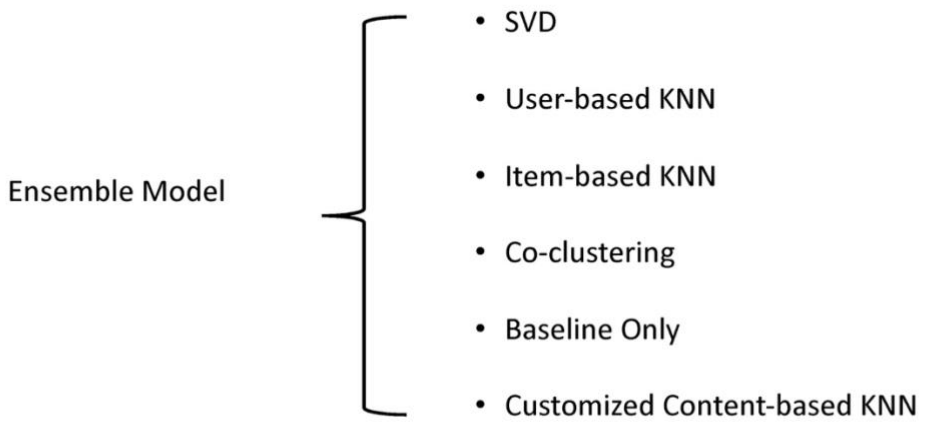
OVERVIEW

The coding for this project were mainly based on the python libraries including Surprise, Sci-kit Learn, and Pandas. A selected number of Surprise built-in algorithms were used to build a recommendation system based on the beer rating dataset. These algorithms were SVD, User-based KNN, Item-based KNN, Co-clustering, and Baseline. The metric used for comparing these algorithms is Mean Absolute Error (MAE).

This project tried to build a content based KNN model using the customisation module Algobase from the Surprise library. This project also tried to build an ensemble model by stacking the Surprise built-in algorithms with the customized algorithm. The meta-algorithm used to wrap everything is linear regression.

MODEL DESCRIPTION

The final estimation model used was an ensemble model stacked with SVD, User-based KNN, Item-based KNN, Content-based KNN, Co-clustering and Baseline. For SVD, default parameters were used. For the User-based KNN and Item-based KNN, the chosen similarity option is cosine similarity matrix. For Co-clustering, default parameters were used, which basically gave us 3 user clusters and 3 item clusters. For the Baseline Only model, default parameters were used. For the customised Content-based KNN model, number of nearest neighbours (k) was set to 40.



RMIT Classification: Trusted

FEATURE ENGINEERING

Three columns in the train, validate, and test dataset were used. They were ReviewerID, BeerID, and Label. Two columns in the features dataset were used for building the content-based model. They were BrewerID and ABV.

EXPERIMENTS

1. Comparing Surprise built-in algorithms based on the pre-split train and validation dataset

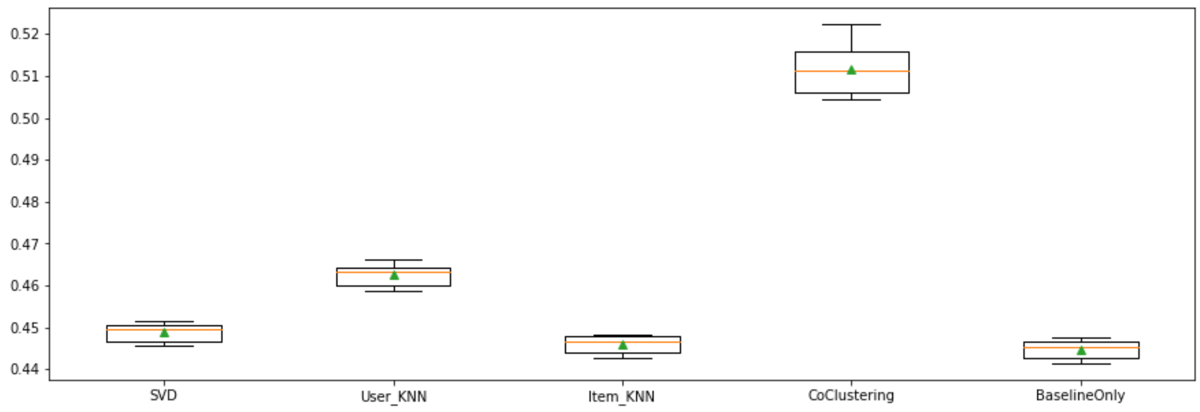
Surprise built-in algorithms were compared in this experiment. The pre-defined train and validation file was loaded as data frames and then converted to a trainset and a valset that can be read by Surprise algorithms. Fit the data into each model and compute the mean absolute error (MAE) and root mean squared error (RMSE) as their metrics. The results can be seen as below:

Algorithm	**MAE**	**RMSE**
SVD	0.4447	0.5928
User_KNN	0.4591	0.6073
Item_KNN	0.4418	0.5904
CoClustering	0.5075	0.6491
BaselineOnly	0.4417	0.5899

In this experiment, the BaselineOnly algorithm had the best performance with the lowest MAE, which is 0.4417, while the performance of Item_KNN is very close to the best.

2. Comparing Surprise built-in algorithms with KFold cross validation

Above built-in algorithms were compared again with 10-fold cross validation. Validation set was not used this time. Train set was split into 10 folds using the stratified fold function from Sci-kit Learn library. The idea is to make BeerID the category variable for splitting train and test set. When using the stratified kfold function, the train set will be split proportionally for each beer ID, so that we can guarantee there is at least one of each item in both train set and test set as much as we can. Apart from the splitting, the rest of the train and evaluation process is similar to the previous experiment. The result can be seen below:



This experiment proved again that BaselineOnly algorithm performed the best for this beer rating dataset.

3. Build a customized content based KNN model

The codes used in this experiment is adapted from the codes Houhaa posted on Kagle. (Houhaa, 2021). The idea is to build a similarity matrix for the items based on the contents of the items. If a user gave a high rating to an item, the he was likely to give high rating to an item with similar contents. In this experiment, the chosen features/contents to build the similarity matrix are BrewerID and ABV.

For the BrewerID, I simply define that the similarity is 1 if the BrewerID is the same for different beers, otherwise the similarity is 0.01. For the ABV, an exponential function was used to scale the difference of ABV between two beers to a number that falls between 0 and 1. The overall similarity was computed by multiplying the brewer similarity and the ABV similarity.

When predicting the rating of a beer from a user, all the ratings that this user had given to other beers were extracted and pooled. Then compute the weighted average rating for the most similar beers (highest in the value of similarity). The result is the prediction for a new beer.

The metrics for this customized content based KNN model were added to previous evaluation table for comparison. The result is shown below:

Algorithm	**MAE**	**RMSE**
SVD	0.4447	0.5928
User_KNN	0.4591	0.6073
Item_KNN	0.4418	0.5904
CoClustering	0.5075	0.6491
BaselineOnly	0.4417	0.5899
Content_KNN	0.5022	0.6759

RMIT Classification: Trusted

4. Build an ensemble model to stack the built-in algorithms and the customized algorithm

The idea of stacking is to use the predictions from the base estimators as inputs to train the final estimator. In this experiment, the train set was split into 5 folds, 4 train folds and 1 test fold. The train folds were used to train the base models and the test fold was used to generate predictions for each model. We got 5 folds of predictions in the end, which was combined as an independent dataset to train the meta model. Linear regression model was used as the meta model to wrap up everything. Then base models were trained on the whole training set.

Similar to previous experiment on cross validation. The 5-fold training process also used stratified kfold function. The BeerID column was set to be the category variable so that at least one item would be in both train split and test split.

The metrics for the ensemble model was added to previous evaluation table for comparison. The result can be seen below:

Algorithm	**MAE**	**RMSE**
SVD	0.4447	0.5928
User_KNN	0.4591	0.6073
Item_KNN	0.4418	0.5904
CoClustering	0.5075	0.6491
BaselineOnly	0.4417	0.5899
Content_KNN	0.5022	0.6759
Ensemble Model	0.4378	0.5843

RMIT Classification: Trusted

LIBRARIES USED

matplotlib==3.4.3
numpy==1.21.2
pandas==1.3.3
scikit-learn==1.0
scikit-surprise==1.1.1
sklearn==0.0

REFERENCES

Houhaa. (2021). Recommendation Systems. https://www.kaggle.com/nouhazouaghi/recommendation-systems