COMP4332 / RMBI4310 Project 3 Group 8

In this project, we aim to predict the star rating of users to business in the Recommendation System. We have used multiple methods in the project: Collaborative Filtering, Neural Collaborative Filtering, XGBoost.

Model Training

For each model used in the recommendation system, data preparation and hyper-parameter tuning are done. The best hyperparameters are shown as follows with the RMSE of the validation dataset.

Models	**Data preparation**	**Best Hyper parameter**	**Train RMSE**	**Valid RMSE**
Collaborative Filtering	build users and items vocabulary	embedding_size= 30 layer= mlp epoch= 3	0.8602755	1.0318438
Neural Collaborative Filtering	build users and items vocabulary	ncf_size= 40 ncf_layer = mlp Best ncf_epoch = 3	0.9182000	1.0477171
Wide and Deep model	build continuous and category features with 50,30,20 most frequent categories combinations	optimizer = adam epoch = 9 embedded size = 150	0.987072	1.004304
XGBoost[1]	Features engineering	subsample = 0.6 n_estimators = 1000 min_child_weight = 5 max_depth = 5 learning_rate = 0.01 gamma = 1.5 colsample_bytree = 0.6	0.9636991876357113	0.9892214476922436

Data Preprocessing

For XGBoost, the rating tables are merged with the content features tables. Each row in the rating tables is associated with corresponding user's and item's content features. The columns of business data and user data that are redundant and useless to the model training, such as address, name and latitude, are dropped. One-hot encoding has been

applied to transform the categorical variables to dummy variables so that they can be used for model training.

Parameter Tuning

Among the algorithms, as XGBoost performed the best result, the meaning and trained value of each hyper-parameter are shown in the following.

Parameter	Explanation	Range
min_child_weight	Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In a linear regression task, this simply corresponds to the minimum number of instances needed to be in each node. The larger min_child_weight is, the more conservative the algorithm will be.	1, 5, 10
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.	0.5, 1, 1.5, 2, 5
subsample	The number of samples (rows) used in each tree, set to a value between 0 and 1, often 1.0 to use all samples.	0.6, 0.8, 1.0
colsample_bytree	the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.	0.6, 0.8, 1.0
max_depth	The maximum depth of each tree in the random forest classifier model.	3, 5, 7, 10
learning_rate	Gradient boosting involves creating and adding trees to the model sequentially. New trees are created to correct the residual errors in the predictions from the existing sequence of trees. learning_rate is set to slow down the learning in the gradient boosting model which applies a weighting factor for the corrections by new trees when added to the model.	0.01, 0.02, 0.05
n_estimators	The number of trees in the model	200,400,600, 800,1000

Evaluation

According to the result after parameter tuning, XGBoost is the best algorithm with the lowest validation RMSE = 0.9903784184660418. To further understand the performance, we visualize the error distribution (actual value - predicted value) in a graph.[2] Most of the errors are centralized in the range between -1 and 1.

Appendix

1. XGBoost

XGBoost (Supplement information) XGBoost stands for eXtreme Gradient Boosting. It is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. Gradient boosting employs a gradient descent algorithm to minimize errors in sequential models. It builds the model in a stage-wise fashion, and generalizes them by allowing optimization of an arbitrary differentiable loss function. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now. In particular, XGBoost holds the advantages of its fast execution speed and great model performance.

2. Error Distribution of XGBoost