2022-ccbda HW5 311511036 洪愷尹

Recommendation System with PySpark

Getting started

- Upload data to google drive `/content/drive/MyDrive/2022-CCBDA/`

```
└── 2022-CCBDA
    ├── train_student.csv
    ├── train_meta.csv
    ├── test_public.csv
    ├── test_private.csv
    └── test_all.csv
```

- Open notebook on Google Colab

Method

1. Import the necessary functions from PySpark's ML module, including StringIndexer, IndexToString, and Pipeline. The StringIndexer function is used to convert string values in the input data into numerical indexes. This is necessary because the ALS algorithm can only work with numerical data.
2. A list of indexers is created for each column in the training data, except for the rating column. The Pipeline function is then used to fit and transform the training data using these indexers, resulting in a new dataframe with indexed columns.
3. The ALS model is then created, specifying the user, item, and rating columns, as well as several other parameters. The transform method is applied to the test data to make predictions, and the na function is used to fill any `NaN` values in the predictions with the mean rating value. The model's performance is evaluated using the RegressionEvaluator function, which calculates the RMSE.
4. The same process is repeated for the public dataset, and the resulting predictions are displayed using the show method.

Difference from baseline method

My method is mainly based on TA method.

1. I started with a baseline code that utilized the Alternating Least Squares (ALS) algorithm to build a recommendation model. I first modified the split ratio of the training and testing datasets, increasing the proportion of training data. This led to a 1.1% improvement in performance.
2. Next, I attempted to further improve the model by using `ParamGridBuilder` and `CrossValidator` to select the best model from a pool of potential model settings. However, this modification did not improve performance and actually resulted in worse performance compared to the baseline. I suspect that the specific parameters I tuned for the `als.rank` and `als.regParam` may not have been reasonable choices.
3. Lastly, In the evaluation stage, I found that the value chosen to replace the NaN values in the predictions had a significant impact on RMSE of the model. Through trial and error, I determined that replacing the NaN values with 4.35 resulted in the best performance. This suggests that carefully considering the value used to fill in missing data can significantly impact the model's accuracy.

Submission

- Add column (U_I, rating) to csv file header manually.
- Upload to Kaggle competition server.