**Project 3**

**Recommender Systems**

Junrui Zhu 606530444

Liwei Tan 206530851

Tianxiang Xing 006530550

1. **Question 11: For each of Popular, Unpopular and High-Variance test subsets, design a naive collaborative filter for each trimmed set and evaluate its performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE**

* **naïve\_filter.py**

1. Find the popular, unpopular and high-variance datasets based on their definition.
2. Compute the cross-fold RMSE as follows:

RMSE of the oringinal dataset: 1.1781

RMSE of popular dataset: 1.1722

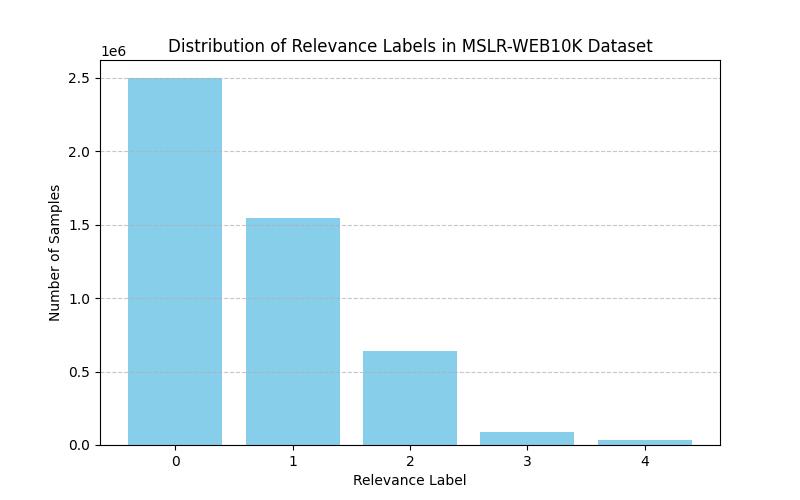
RMSE of umpopular dataset: 2.7826

RMSE of high variance dataset: 3.2018

1. **Question 13: Data Understanding and Preprocessing: Use the provided helper code for loading and pre-processing Web10k data. Print out the number of unique queries in total and show distribution of relevance labels.**

* **web10k.py**

1. The number of unique queries in total: 10000
2. The distribution of relevance labels is shown in the figure below:

****

Comment: Most of the labels have low relevance.

1. **Question 14: LightGBM Model Training: For each of the five provided folds, train a LightGBM model using the ’lambdarank’ objective. After training, evaluate and report the model’s performance on the test set using nDCG@3, nDCG@5 and nDCG@10.**

* **light\_gbm.py**

1. Configuration of the network:

params = {

            'objective': 'lambdarank',

            'metric': 'ndcg',

            'device': 'gpu',  # Use GPU for faster training

            'gpu\_platform\_id': 0,

            'gpu\_device\_id': 0,

            'learning\_rate': 0.1,

            'num\_leaves': 31,

            'min\_data\_in\_leaf': 20,

            'verbose': -1,

            'lambda\_l1': 1.0,

            'lambda\_l2': 1.0

        }

Set early stopping after 5 rounds without improvement on validation set.

Set boost\_round = 800.

1. The performance on test set:

# Performance on Fold1 test set:

# nDCG@3: 0.5354

# nDCG@5: 0.5349

# nDCG@10: 0.5393

# Performance on Fold2 test set:

# nDCG@3: 0.5403

# nDCG@5: 0.5367

# nDCG@10: 0.5392

# Performance on Fold3 test set:

# nDCG@3: 0.5309

# nDCG@5: 0.5317

# nDCG@10: 0.5388

# Performance on Fold4 test set:

# nDCG@3: 0.5427

# nDCG@5: 0.5428

# nDCG@10: 0.5484

# Performance on Fold5 test set:

# nDCG@3: 0.5428

# nDCG@5: 0.5448

# nDCG@10: 0.5499

1. Comments: Our network has an approximate 0.72 training score across all 5 folders and 3 nDCGs. The reason of bad performance on test set is, perhaps, the high relevance labels are too rare, making the model unable to perform on test set.
2. **QUESTION 15: Result Analysis and Interpretation: For each of the five provided folds, list top 5 most important features of the model based on the importance score.**

* **q15.py**

===== Fold1 - Top 5 Features =====

feature\_name  importance\_gain  importance\_split

0   Column\_133     22509.302441               257

1    Column\_54     14422.273663               218

2   Column\_129      6566.901497               814

3   Column\_128      4828.602377               472

4    Column\_14      4605.773697               406

===== Fold2 - Top 5 Features =====

feature\_name  importance\_gain  importance\_split

0   Column\_133     22282.439733               273

1    Column\_54     15825.311511               255

2   Column\_129      7057.400825               917

3   Column\_130      5328.208504              1253

4   Column\_128      4406.970275               524

===== Fold3 - Top 5 Features =====

feature\_name  importance\_gain  importance\_split

0   Column\_133     22420.718331               296

1    Column\_54     13112.206020               221

2   Column\_129      6768.141324               953

3   Column\_128      5197.381736               460

4   Column\_130      5021.586955              1221

===== Fold4 - Top 5 Features =====

feature\_name  importance\_gain  importance\_split

0   Column\_133     22891.101171               273

1    Column\_54     15846.240428               256

2   Column\_129      5640.414552               915

3   Column\_128      4834.048760               476

4   Column\_130      4510.946808              1140

===== Fold5 - Top 5 Features =====

feature\_name  importance\_gain  importance\_split

0   Column\_133     21986.197452               267

1    Column\_54     15900.570613               234

2   Column\_129      5494.489304               929

3   Column\_130      5408.566897              1200

4   Column\_128      5275.257707               529

1. **Question 16:**
2. **For each of the five provided folds: 13 • Remove the top 20 most important features according to the computed importance score in the question 15. Then train a new LightGBM model on the resulted 116 dimensional query url data. Evaluate the performance of this new model on the test set using nDCG. Does the outcome align with your expectations? If not, please share your hypothesis regarding the potential reasons for this discrepancy.**

* **q\_16\_1.py**

1. We use the same network hyperparameters as in question 14 to train this network.
2. The performance after removing top 20 features:

===== Performance after removing top 20 features (Fold1) =====

nDCG@3:  0.4428

nDCG@5:  0.4443

nDCG@10: 0.4520

===== Performance after removing top 20 features (Fold2) =====

nDCG@3:  0.4423

nDCG@5:  0.4434

nDCG@10: 0.4533

===== Performance after removing top 20 features (Fold3) =====

nDCG@3:  0.4416

nDCG@5:  0.4435

nDCG@10: 0.4568

===== Performance after removing top 20 features (Fold4) =====

nDCG@3:  0.4465

nDCG@5:  0.4481

nDCG@10: 0.4586

===== Performance after removing top 20 features (Fold5) =====

nDCG@3:  0.4384

nDCG@5:  0.4426

nDCG@10: 0.4575

1. Comments: This is exactly what we expected, as the nDCG score drops by about 0.1, which is pretty significant. This is because most of the top features are removed, making the remained dataset hard to learn.
2. **Remove the 60 least important features according to the computed importance score in the question 15. Then train a new LightGBM model on the resulted 76 dimensional query-url data. Evaluate the performance of this new model on the test set using nDCG. Does the outcome align with your expectations? If not, please share your hypothesis regarding the potential reasons for this discrepancy.**

* **q16\_2.py**

1. We use the same network hyperparameters as in question 14 to train this network.
2. Performance after removing 60 least important features:

===== Performance after removing bottom 60 features (Fold1) =====

nDCG@3:  0.5220

nDCG@5:  0.5213

nDCG@10: 0.5255

===== Performance after removing bottom 60 features (Fold2) =====

nDCG@3:  0.5180

nDCG@5:  0.5151

nDCG@10: 0.5194

===== Performance after removing bottom 60 features (Fold3) =====

nDCG@3:  0.5113

nDCG@5:  0.5103

nDCG@10: 0.5162

===== Performance after removing bottom 60 features (Fold4) =====

nDCG@3:  0.5157

nDCG@5:  0.5157

nDCG@10: 0.5237

===== Performance after removing bottom 60 features (Fold5) =====

nDCG@3:  0.5348

nDCG@5:  0.5307

nDCG@10: 0.5337

1. Comments: We can see that after removing 60 least important features, there’s hardly any change on the test performance. Although this seems unusual, this phenomenon is understandable given the characteristic of the dataset.

Since nDCG emphasizes top-ranked documents, features affecting lower-ranked results may not show significant performance differences when removed.

The lack of improvement is likely due to the global importance bias, limited redundancy, and local significance of the removed features.