Home-depot product search relevance

Yixin Zhang, Yuantai Xie | 2020 Aug 21

NYU

Results

Model	Details	NDCG metric	Train RMSE	Test RMSE
GBM	{'max_depth': 6, 'n_estimators': 40}	0.988	0.45	0.47
XGBoost	{'max_depth': 4, 'n_estimators': 32}	0.987	0.46	0.48
LightGBM	{‘num_leaves': 15}	0.986	0.52	0.53

Auto machine learning Results

Model	Train RMSE	Test RMSE
H2O	0.46	0.47
SparkBeyond	0.49	0.51

Contents

- Data summary
- Date processing
- Model performance
  - Feature importance
  - Grid search
- Model performance using NDCG metric
  - Lambdarank (LightGBM and XGBoost) vs GBM

Data summary

- Predict the *relevance* of search results with products

	id	product_uid	product_title	search_term	relevance
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.00
3	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	rain shower head	2.33
4	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	shower only faucet	2.67

```
(74067, 5)
```

Data processing

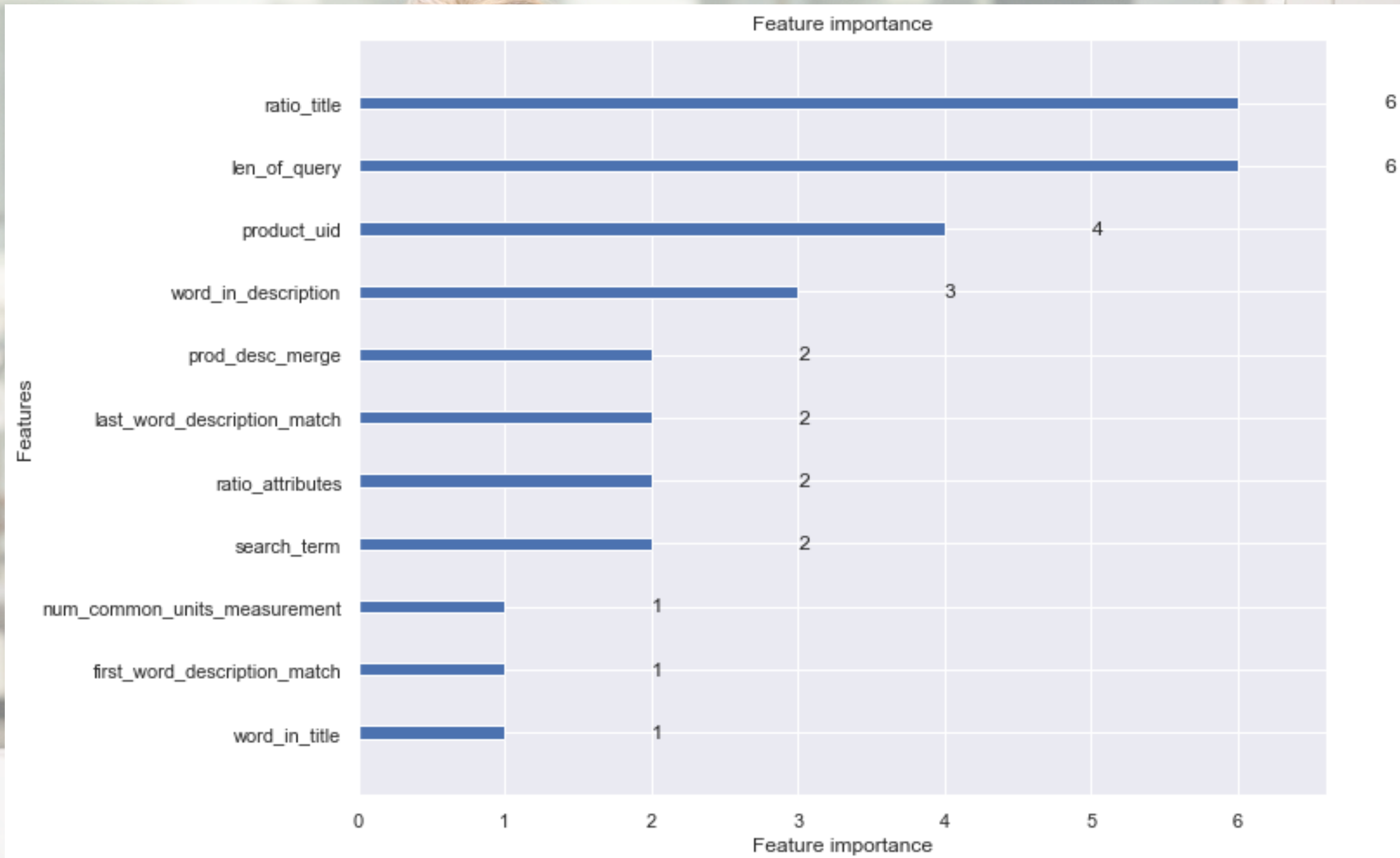
1) Spell checking
   - Use a pre-defined dictionary of spelling mistakes
   - 3,400 pairs
2) Stemming
   - Perform stemming function on variabels *search_term, product_title, product_description, and product_attributes*
3) Create 16 new features
4) Calculate TF-IDF and do feature reduction

Auto ml

VARIABLE IMPORTANCE

6_CVTE:search_term.0	1.00
7_TxtTE:search_term.0	0.22
0_product_uid	0.02
15_NumToCatTE:product_uid.0	0.02
16_NumToCatTE:product_uid.0	0.01
17_Txt:search_term.0	0.01
17_Txt:search_term.8	0.01
17_Txt:search_term.1	0.01
17_Txt:search_term.48	0.00
17_Txt:search_term.7	0.00
17_Txt:search_term.45	0.00
17_Txt:search_term.49	0.00
17_Txt:search_term.18	0.00
17_Txt:search_term.28	0.00

Model performance: Feature importance



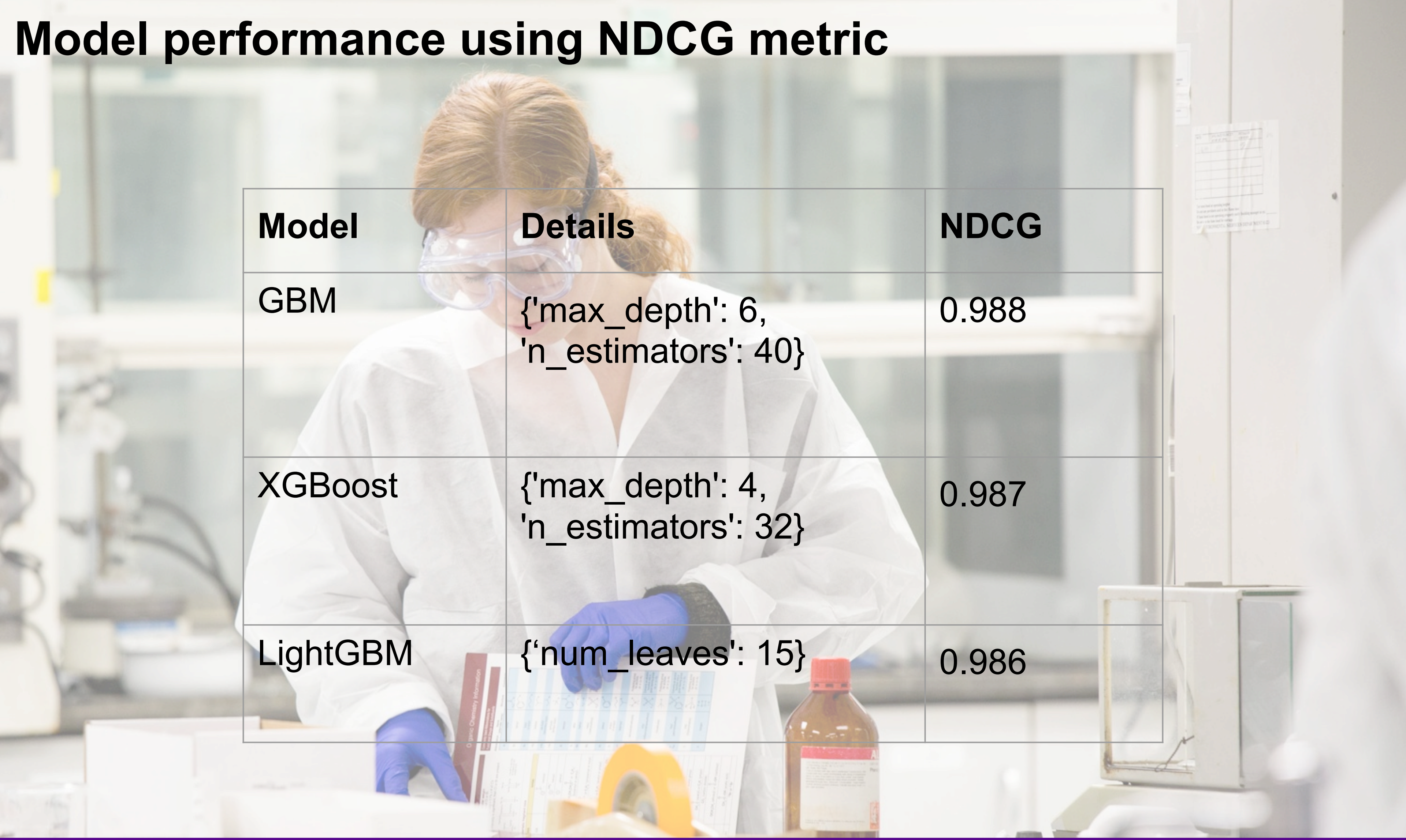
Model performance: Grid Search

Model	Details	CPU times	Wall time
GBM	{'max_depth': 6, 'n_estimators': 40}	16min 3s	16min 8s
XGBoost	{'max_depth': 4, 'n_estimators': 32}	21min 13s	21min 53s
LightGBM	{‘num_leaves': 15}	35min 6s	36min 3s

Intro and code of NDCG metric

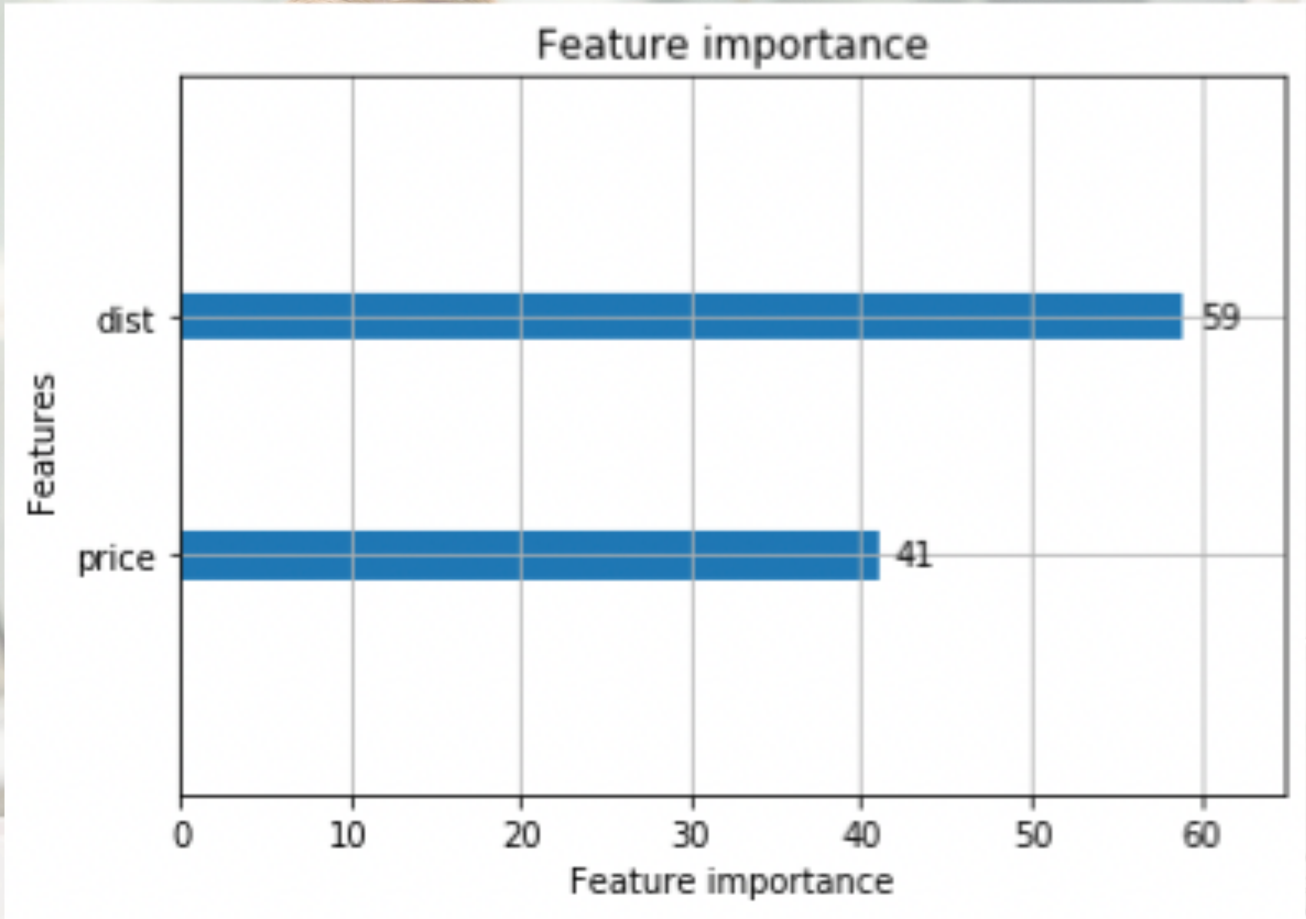
```python
"""
This function calculates ndcg for multiple queries dataset
y_true : array, shape = [n_samples]
        Ground truth (true relevance labels)
y_score : array, shape = [n_samples]
        Predicted scores
k:      int, optional
        Only consider the highest k scores in the ranking. If None, use all outputs.
group: array, shape = [n_groups]
        each element denotes how many items are there in each group
assume all queries have equal weights
"""

def ndcg_score(y_true, y_score, group, k = None):
    avg_ndcg = 0
    index = 0 #next row to be calculated
    count = 0 #number of groups which can provide information
    for i in range(0, len(group)):
        cur_true = y_true[index: index+group[i]-1]
        cur_score = y_score[index: index+group[i]-1]
        index += group[i]

        idcg = dcg_score(cur_true, cur_true, k)
        # when ground truth is equal to 0, we abandon that group which provides no information
        if idcg == 0:
            continue

        cur_ndcg = dcg_score(cur_true, cur_score, k)/idcg
        avg_ndcg = cur_ndcg * 1/(count+1) + avg_ndcg * count/(count+1)
        count += 1

    return avg_ndcg

def dcg_score(y_true, y_score, k = None):
    order = np.argsort(y_score)[::-1]
    y_true = np.take(y_true, order[:k])

    gain = 2 ** y_true - 1

    discounts = np.log2(np.arange(len(y_true)) + 2)
    return np.sum(gain / discounts)
```

Model performance using NDCG metric



Model	**Details**	**NDCG**
GBM	{'max_depth': 6, 'n_estimators': 40}	0.988
XGBoost	{'max_depth': 4, 'n_estimators': 32}	0.987
LightGBM	{‘num_leaves': 15}	0.986

LGBMRanker model running



Thanks for listening!