[추천시스템] 데이터탐색 및 테스트모델 적용

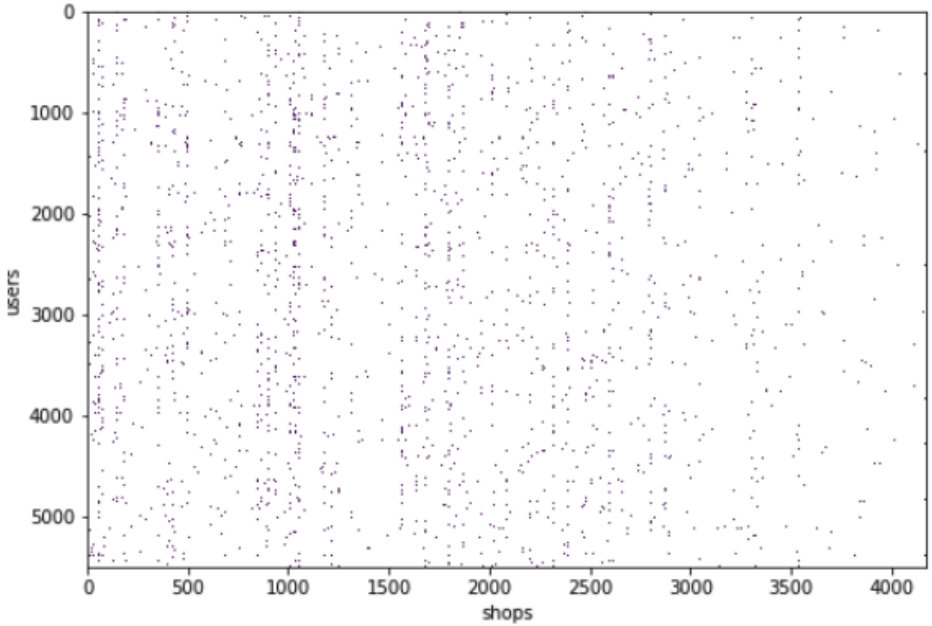
Info.

- 지라티켓: DATASVC-2119 - Jira 이슈가 존재하지 않거나 볼 권한이 없습니다.
- 작업노트북: https://david-i03.woowa.in:8891/#/notebook/2E242K9GS
- 목차

작성완료

1. 데이터 정보
   a. 테이블: sbsvc.ord
   b. 추출지역: 송파구
   c. 추출일: 2018/1/1 ~ 2018/12/31 (1년)
   d. 필터링 기준
      i. 디바이스별 주문수 70건 이상 → 유니크 디바이스 수: 5,507
      ii. 업소는 전체 이용 → 유니크 업소수 4,564
   e. 유저 by 업소 매트릭스 (5507 x 4564)

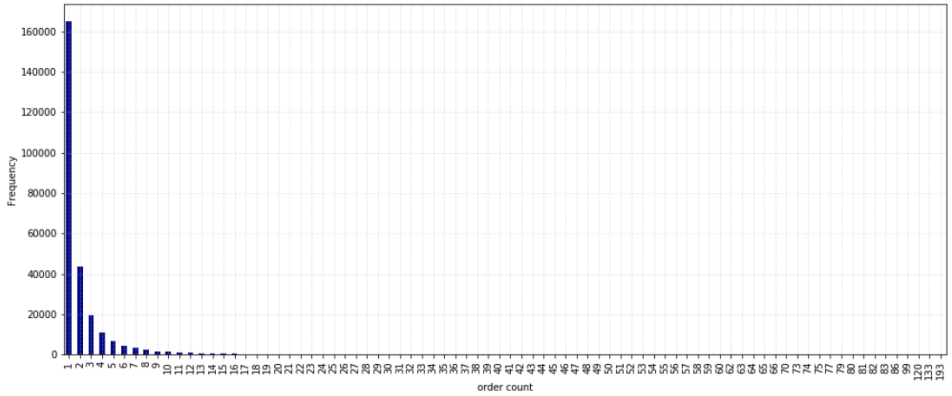
f.



g.

dvc_id	shop_no	count
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	394911	4
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	403253	2
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	430284	1
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	451300	1
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	533525	1
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	538774	1
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	544722	1
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	548266	2
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	572859	3
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	585137	2
OPUD00165420-5688-4B48-8799-85F3AEFF95C9	585594	1

h. 유저별 & 주문수별 빈도의 분포



i.

2. 모델 구축 및 성능 평가
    a. 구축 환경
        i. 서버: https://david-i03.woowa.in:8891/#/notebook/2E242K9GS
        ii. Python 라이브러리: Surprise
    b. Baseline Model (https://surprise.readthedocs.io/en/stable/prediction_algorithms.html#similarity-measure-configuration)
        i. Cost function

$$minimize \sum (r_{ui} - \hat{r}_{ui})^2 + \lambda(\sum_u {b_u}^2 + \sum_i {b_i}^2)$$

, where $\hat{r}_{ui} = \bar{r} + b_u + b_i$

        1.
        ii. Using Alternating Least Squares (ALS) with default options
            1. reg_i (The regularization parameter for item) → default: 10
            2. reg_u (The regularization parameter for users) → default: 15
            3. n_epochs (The number of iteration) → default: 10

4. Python code

```
# http://surpriselib.com/
from surprise import Reader, Dataset, SVD, evaluate, BaselineOnly
from surprise.model_selection import cross_validate
from surprise.model_selection import train_test_split
from surprise.model_selection import GridSearchCV
from surprise import accuracy

# create a reader object
reader = Reader()

# read dataset
data = Dataset.load_from_df(df1, reader)

# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# ALS option
bsl_options = {'method': 'als',
               'n_epochs': 10,
               'reg_u': 15,
               'reg_i': 10}
# algo
algo = BaselineOnly(bsl_options=bsl_options)

# fit
model = algo.fit(trainset)
predictions = algo.test(testset)

# Then compute RMSE
print(accuracy.rmse(predictions))
print(accuracy.mae(predictions))

# CV
cross_validate(algo, data, verbose=True, cv=5)
```

5. Cross Validation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	1.4943	1.4850	1.5088	1.4930	1.4826	1.4928	0.0092
RMSE (testset)	2.9123	2.9007	3.0640	2.8504	2.8806	2.9216	0.0742
Fit time	1.06	0.75	0.82	0.77	0.75	0.83	0.12
Test time	0.69	0.76	0.56	0.72	0.80	0.70	0.08

a.

6. 테스트셋 검증 결과
    a. MAE: 1.4971
    b. RMSE: 2.9699

iii. Using Stochastic Gradient Descent (SGD) with default options

1. reg: The regularization parameter of the cost function that is optimized → default: 0.02
2. learning_rate → default: 0.005
3. n_epochs (The number of iteration → default: 20
4. Python Code

```
# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# SGD option
bsl_options = {'method': 'sgd',
               'n_epochs': 20,
               'learning_rate': 0.005,
               'reg': 0.02}
# algo
algo = BaselineOnly(bsl_options=bsl_options)

# fit
model = algo.fit(trainset)
predictions = algo.test(testset)

# Then compute RMSE
print(accuracy.rmse(predictions))
print(accuracy.mae(predictions))

# cv
cross_validate(algo, data, verbose=True, cv=5)
```

5. Cross Validation 결과

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	1.4934	1.5043	1.4970	1.4742	1.4880	1.4914	0.0101
RMSE (testset)	2.8703	3.0295	2.9834	2.7712	3.0118	2.9333	0.0982
Fit time	1.50	1.90	1.80	1.71	1.22	1.63	0.24
Test time	0.91	0.72	0.93	0.89	0.48	0.78	0.17

   a.
6. 테스트셋 검증 결과
   a. MAE: 1.4942
   b. RMSE: 2.9633

iv. Grid Search

1. Python Code

```
# grid search for the best parameter
# set params
param_grid = {'bsl_options': {'method': ['als', 'sgd'],
                              'learning_rate': [0.005, 0.001],
                              'reg': [1, 0.1, 0.2],
                              'reg_u': [10, 15],
                              'reg_i': [10, 15],
                              'n_epochs': [10, 20]
```

```
                                }
            }

# grid search
algo = BaselineOnly

gs_base = GridSearchCV(algo, param_grid, measures=['rmse', 'mae'], cv=5)

gs_base.fit(data)

# best RMSE score
print(gs_base.best_score['rmse'])
print(gs_base.best_score['mae'])

# combination of parameters that gave the best RMSE score
print(gs_base.best_params['rmse'])
print(gs_base.best_params['mae'])

# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# the best parameter
algo = gs_base.best_estimator['rmse']

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

# Use the new parameters with the train data
bls_best_options = {'reg_u': 15,
                    'learning_rate': 0.005,
                    'n_epochs': 20,
                    'reg': 1,
                    'method': 'als',
                    'reg_i': 10}

algo = BaselineOnly(bls_best_options)

algo.fit(trainset)
test_pred = algo.test(testset)

print(accuracy.rmse(test_pred, verbose=True))
print(accuracy.mae(test_pred, verbose=True))
```

2. Cross Validation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	1.4990	1.4793	1.4989	1.4944	1.4950	1.4933	0.0073
RMSE (testset)	2.9957	2.9066	2.9001	2.8525	2.9538	2.9218	0.0490
Fit time	1.28	1.22	1.17	1.28	1.38	1.27	0.07
Test time	0.49	0.82	0.52	0.48	0.49	0.56	0.13

a.

3. 테스트셋 검증 결과 (with the best parameter)
    a. MAE: 1.4904
    b. RMSE: 2.8143

c. Matrix Factorization (with SVD)
    i. Cost function

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

$$\sum_{r_{ui} \in R_{train}} \left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2\right)$$

1.

    ii. Grid Search

1. Python Code

```
# set params
param_grid = {'n_factors': [2, 10],
              'lr_all': [0.001, 0.01, 0.1],
              'reg_all': [0.01, 0.001, 0.1],
              'n_epochs': [10, 20]
              }

# grid search
algo = SVD

gs_mf = GridSearchCV(algo, param_grid, measures=['rmse', 'mae'], cv=5)

gs_mf.fit(data)


algo = gs_mf.best_estimator['rmse']

print(gs_mf.best_score['rmse'])
print(gs_mf.best_params['rmse'])

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# Use the new parameters with the train data
algo = SVD(reg_all=0.01, lr_all=0.001, n_factors=2, n_epochs=20)

algo.fit(trainset)
test_pred = algo.test(testset)

print(accuracy.rmse(test_pred, verbose=True))
print(accuracy.mae(test_pred, verbose=True))
```

2. Cross Validation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	1.5026	1.5150	1.5072	1.5084	1.5079	1.5082	0.0040
RMSE (testset)	2.8904	2.9841	2.9141	2.9883	2.8341	2.9222	0.0584
Fit time	4.73	4.86	4.19	4.29	4.56	4.53	0.25
Test time	0.75	0.76	0.70	0.71	0.84	0.75	0.05

   a.
3. 테스트셋 검증 결과(with the best options)
   a. MAE: 1.5152
   b. RMSE: 2.9297
   c.

d. Matrix Factorization (with Non-Negative MF)
   i. Cost function: 위 SVD와 동일
   ii. Grid Search

1. Python Code

```
# set params
param_grid = {'n_factors': [2, 10],
              'reg_pi': [0.001, 0.01, 0.1],
              'reg_ui': [0.01, 0.001, 0.1],
              'n_epochs': [10, 20]
              }

# grid search
algo = NMF

gs_nmf = GridSearchCV(algo, param_grid, measures=['rmse', 'mae'], cv=5)

gs_nmf.fit(data)

algo = gs_nmf.best_estimator['rmse']

print(gs_nmf.best_score['rmse'])
print(gs_nmf.best_params['rmse'])

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# Use the new parameters with the train data
algo = NMF(reg_pi=, reg_ui=, n_factors=, n_epochs=)

algo.fit(trainset)
test_pred = algo.test(testset)

print(accuracy.rmse(test_pred, verbose=True))
print(accuracy.mae(test_pred, verbose=True))
```

2. Cross Vaildation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	1.5242	1.5197	1.5138	1.5360	1.5410	1.5270	0.0101
RMSE (testset)	3.0588	2.9238	2.9212	3.0897	3.0300	3.0047	0.0697
Fit time	3.15	2.99	3.68	3.11	3.25	3.24	0.24
Test time	0.46	0.48	0.45	0.65	1.09	0.63	0.24

   a.
3. 테스트셋 검증 결과(with the best options)
   a. MAE: 1.6077
   b. RMSE: 3.0842

e. Neighborhood Model (user-based, item-based)
   i. modeling with the default options

1. Python Code

```
## user based
# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# cosine, MAD
algo = KNNBasic(user_based=True, name='MAD')

algo.fit(trainset)
test_pred = algo.test(testset)

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

print(accuracy.rmse(test_pred, verbose=True))
print(accuracy.mae(test_pred, verbose=True))


## item based
# split the dataset
trainset, testset = train_test_split(data, test_size=.25)

# cosine, MAD
algo = KNNBasic(user_based=False, name='MAD')

algo.fit(trainset)
test_pred = algo.test(testset)

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

print(accuracy.rmse(test_pred, verbose=True))
print(accuracy.mae(test_pred, verbose=True))
```

2. 테스트셋 검증 결과 (with default options, similarity=MAD)
    a. user based
        i. MAE: 1.4708
        ii. RMSE: 2.9640
    b. item based
        i. MAE: 1.4788
        ii. RMSE: 3.0154

f. 모델별 성능 종합
    i. RMSE 기준, 검증셋에서 가장 높은 성능을 보인 모델은
    ii. 계산 속도 측면에서는 Baseline 모델이 가장 우수함
    iii. RMSE 기준 모델별 성능

1.

Model	mean CV performance	std in CV	mean fitting(Min.)	TEST performance
Baseline	2.921	0.049	1.27	2.814
CF MF	2.922	0.058	4.53	2.929

CF NMF	3.004	0.069	3.24	3.084
CF user-based	2.984	0.028	4.27	2.964
CF item-based	2.9849	0.069	4.46	3.015

g. Prediction 예시
   i. 특정 사용자 실데이터

dvc_id	shop_no	count
OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4	569112	1
OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4	663918	1
OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4	715370	5
OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4	787424	4

1.

   ii. 예측 결과

1. Python code

```
algo.predict('OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4', '787424', r_ui=1, verbose=True)
```

2. user: OPUD0120DBD4-C5FC-4CCD-B5FD-E2FACDE6A2D4 item: 787424 r_ui = 1.00 est = 3.67

h. Content-based
   i. 배라 업소의 태그, 테마와 같은 Contents를 이용해 유사도 계산
   ii. Popularity 모델과 같이 Cold-starting 문제 해결책로 이용 가능
   iii. 구축 과정

shop_no	shop_nm	rgn2_nm	content
653188	스트릿츄러스 신천점	송파구	디저트·커피, 1인분, 꿀맛, 썸남썸녀랑, 바삭바삭, 에이드, 츄라이, 음료판매, ...
670255	서울쌀면	송파구	아시안, 1인분, 담백, 겨울, 쌀국수, 쌀덕후, 회식, 점심, 육개장, 퓨전한식,...
741054	엘로우샤크	송파구	한식, 1인분, 단짠단짠, 사무실, 한끼식사되는, 덮밥덕후, 밥집, 고기성애자, 닭...
624728	죽이야기 가든파이브점	송파구	한식, 1인분, 간이약한, 쌀덕후, 호박죽, 매운맛불낙죽, 내방, 진한국물, 속이든...
829286	조가네 갑오징어	송파구	한식, NEW, 1인분, 갑오징어전골, 갑오징어숙회
576845	가락시장 용기수산	송파구	수산시장, 주말, 한강공원, 기분내기, 친구들이랑, 저녁, 부드러운, 내방, 비쥬얼...
770935	구어드림	송파구	한식, 1인분, 생선구이, 계장
813392	스무디킹 가든파이브점	송파구	디저트·커피, 스무디, 커피
653587	탐나는제주흑돈	송파구	고기, 한식, 육식인간, 주말, 양념돼지갈비, 회식, 저녁, 목살, 오겹살, 육즙이...
633214	쌍다리돼지불백 잠실점	송파구	고기, 한식, 돼지불고기비빔밥, 한끼식사되는, 돼지불백비빔, 돼지불고기, 돼지불백,...
748275	선우소곱창	송파구	고기, 한식, 한우곱창, 곱창전골

1.

2. Python Code

```
## Content-based
# preprocessing
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity

tf = TfidfVectorizer()
tfidf_matrix = tf.fit_transform(cont['content'])


cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

# rec func
def get_recommandations(shop_no, rgn):
    idx = indice[shop_no]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[0:]
    shop_indices = [i[0] for i in sim_scores]
    return print(cont.iloc[shop_indices][cont['rgn2_nm'] == rgn].to_string())


# sample result
get_recommandations("729141", "")
```

3. 추천 결과
   a. 업소번호: "729141", 지역: "송파구"
   b. cosine similarity 기준으로 729141와 가장 유사한 순으로 정렬해서 노출

shop_no	shop_nm	rgn2_nm	content
729141	진우곱창	송파구	고기, 막창, 친구들이랑, 직원이자신있어하는, 저녁, 곱창, 주당, 철판요리, 대창...
730369	부산양곱창	송파구	고기, 한식, 회사동료랑, 바삭바삭, 고기성애자, 막창, 친구들이랑, 특양, 양볶음...
855277	마마곱창	송파구	고기, 한식, NEW, 막창, 곱창
726880	동대문 철판순대곱창	송파구	고기, 한식, 회사동료랑, 야채막창, 소허파곱창, 친구들이랑, 순대, 입소문난, 매...
760005	야채곱창 곱	송파구	한식, 바삭바삭, 고기성애자, 고깃집, 주당, 한잔하고싶을때, 안주로딱좋은, 입에서...
752603	이자카야인	송파구	일식·회, 토시살참스테이크, 채끝등심스테이크, 바삭바삭, 친구들이랑, 깔끔, 저녁,...
748275	선우소곱창	송파구	고기, 한식, 한우곱창, 곱창전골
581401	칼라분식	송파구	분식, 1인분, 쌀쌀할때, 달콤, 바삭바삭, 매콤, 순대, 입소문난, 깔끔, 저녁,...
742390	오리엔탈 키친	송파구	한식, 쌀쌀할때, 골뱅이, 매콤한낙지볶음과쫄면, 저녁에봄비는, 막창, 친구들이랑, ...
678934	이성원쉐프의청년감자탕	송파구	고기, 한식, 주말, 친구들이랑, 매콤, 회식, 저녁, 부드러운, 내방, 진한국물,...
729842	오징어집	송파구	일식·회, 단짠단짠, 도미, 회사동료랑, 광어, 해산물덕후, 깔끔, 우럭, 저녁, ...
758786	하남돼지집	송파구	고기, 한식, 친구들이랑, 고깃집, 입소문난, 회식, 직원이자신있어하는, 점심, 저...
573348	논두렁황소곱창	송파구	고기, 육식인간, 주말, 매콤, 회식, 저녁, 속이든든한, 육즙이흐르는, 안주로딱좋...
745938	모심정 한우국밥	송파구	고기, 포차, 회사동료랑, 육회, 매콤, 불고기, 회식, 직원이자신있어하는, 한잔하...
608327	용대리코다리찜	송파구	고기, 한식, 1인분, 제육볶음, 음료판매, 매콤, 저녁, 내방, 또먹고싶다, 낙지...
813032	컵앤곱	송파구	한식, 1인분, 순대볶음, 곱창
602981	곱창의전설 방이점	송파구	고기, 한식, 1인분, 레알인정, 주말, 저녁에봄비는, 밥집, 음료판매, 입소문난,...

   c.

3. 논의사항

a. 모델별 성능 차이가 크지 않음

   i. 전처리 필요 (단위 변환 등 feature engineering 고려)

b. 지역별, 비즈니스 특성을 고려한 모델 개발이 필요

c. 유저 행동 혹은 업소/메뉴 프로파일링 필요

4. 추후 진행 내용
   a. 1~2주마다 관련 논문이나 책의 내용 정리/공유
      i. Recommendation System 책, Implicit Feedback, Factorization Machine, TF-ranking, MF with Tensorflow, Hashing Trick 등등
   b. 피드백 반영후 테스트 모델 재구축