Python 추천시스템 패키지

```
import warnings
warnings.filterwarnings('ignore')
```

I. Surprise Package

- 다양한 추천 알고리즘을 쉽게 적용
- 사이킷런과 유사한 API 구조

1) Install Package

```
!pip install surprise
```

```
Requirement already satisfied: surprise in /usr/local/lib/python3.7/dist-packages (0.1)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from sc
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from s
```

2) Import Package

```
import surprise

print(surprise.__version__)
```

```
1.1.1
```

II. Surprise Dataset

user_id(사용자), item_id(아이템), rating(평점)으로 된 데이터 세트만 적용 가능

- 3개의 컬럼만 로딩하고 나머지는 제외

1) MovieLens Dataset

- 로컬 디렉토리에 저장 후 로딩
  - 'ml-100k' : 10만개 평점 데이터
  - 'ml-1m' : 100만개 평점 데이터

```
from surprise import Dataset

data = Dataset.load_builtin('ml-100k')
```

```
!ls -l /root/.surprise_data/
```

```
total 4
drwxr-xr-x 3 root root 4096 Apr  1 02:37 ml-100k
```

2) train_test_split( )

```
from surprise.model_selection import train_test_split

trainset, testset = train_test_split(data,
                                     test_size = 0.3,
                                     random_state = 2045)
```

III. SVD 기반 잠재 요인 협업 필터링

1) fit( ) : 추천 알고리즘 학습

- SVD(Singular Vector Decomposition)

```
from surprise import SVD

algo = SVD()
algo.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fddb4f0ae50>
```

2) test( )

- 사용자-아이템 평점 데이터 세트 전체에 대한 추천을 예측
  - uid : 사용자 아이디
  - iid : 영화(아이템) 아이디
  - r_ui : 실제 평점

- est : Suprise 추천 예측 평점
- details : 처리 결과 로그(True-예측값 생성할 수 없는 데이터)

```
predictions = algo.test(testset)

print('prediction type :',type(predictions), ' size:',len(predictions))
print('₩n', 'prediction 결과의 최초 5개 추출', '₩n')
predictions[:5]
```

```
prediction type : <class 'list'>  size: 30000

 prediction 결과의 최초 5개 추출

[Prediction(uid='13', iid='531', r_ui=3.0, est=2.864833634615578, details={'was_impossible':
 Prediction(uid='567', iid='246', r_ui=4.0, est=4.142685893445308, details={'was_impossible':
 Prediction(uid='243', iid='1148', r_ui=3.0, est=3.2643723361200094, details={'was_impossible
 Prediction(uid='346', iid='241', r_ui=4.0, est=3.3147075831498043, details={'was_impossible'
 Prediction(uid='868', iid='1285', r_ui=2.0, est=2.5996754392940904, details={'was_impossible
```

- 'uid', 'iid', 'est' 값 추출

```
[(pred.uid, pred.iid, pred.est) for pred in predictions[:3]]
```

```
[('13', '531', 2.864833634615578),
 ('567', '246', 4.142685893445308),
 ('243', '1148', 3.2643723361200094)]
```

3) predict( )

- 개별 사용자의 아이템에 대한 추천 평점 예측
  - 'uid', 'iid'는 문자열로 입력
  - 'r_ui' : 기존 평점 정보는 선택 사항
- test( )는 모든 사용자와 아이템에 대해서 predict( )를 반복적으로 수행한 결과

```
uid = str(196)
iid = str(302)

pred = algo.predict(uid, iid)
print(pred)
```

```
user: 196        item: 302        r_ui = None   est = 4.38   {'was_impossible': False}
```

4) rmse( )

- 예측 평점과 실제 평점과의 오차 평가

```
from surprise import accuracy

accuracy.rmse(predictions)
```

```
RMSE: 0.9379
0.9378999936994787
```

IV. Data Preprocessing

1) user_id(사용자), item_id(아이템), rating(평점)

- 컬럼 Header 제거 필요

```
import pandas as pd

rurl = 'https://raw.githubusercontent.com/rusita-ai/pyData/master/ratings.csv'
ratings = pd.read_csv(rurl)

ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

2) index 와 header제거된 파일 생성

```
ratings.to_csv('ratings_noh.csv',
               index = False,
               header = False)
```

3) Suprise - Reader( )

- line_format : 컬럼의 순서 나열
- sep : 컬럼 구분자
- rating_scale : 평점 단위를 0.5(최소) ~ 5(최대)로 설정

```
from surprise import Reader
```

```
reader = Reader(line_format = 'user item rating timestamp',
                sep = ',',
                rating_scale = (0.5, 5))

data = Dataset.load_from_file('ratings_noh.csv',
                              reader = reader)
```

4) SDV 테스트

- n_factors : 잠재 요인(K) 크기 Hyperparameter

```
trainset, testset = train_test_split(data,
                                     test_size = .3,
                                     random_state = 2045)

algo = SVD(n_factors = 50,
           random_state = 2045)
algo.fit(trainset)

predictions = algo.test(testset)

accuracy.rmse(predictions)
```

```
RMSE: 0.8711
0.871106664601276
```

V. pandas DataFrame

- 판다스 DataFrame에서 데이터 로딩
  - Dataset.load_from_df( )

```
from surprise import Reader, Dataset

ratings = pd.read_csv(rurl)
reader = Reader(rating_scale = (0.5, 5.0))

# ratings DataFrame 에서 컬럼은 사용자 아이디, 아이템 아이디, 평점 순서 준수
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)

trainset, testset = train_test_split(data,
                                     test_size = .3,
                                     random_state = 2045)

algo = SVD(n_factors = 50,
           random_state = 2045)
algo.fit(trainset)

predictions = algo.test( testset )

accuracy.rmse(predictions)
```

```
accuracy.rmse(predictions)
```

```
RMSE: 0.8711
0.871106664601276
```

VI. Cross Validation

- cross_validate( )

```
from surprise.model_selection import cross_validate

ratings = pd.read_csv(rurl)
reader = Reader(rating_scale = (0.5, 5.0))
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']],
                            reader)

algo = SVD(random_state = 2045)

cross_validate(algo,
               data,
               measures = ['rmse', 'mae'],
               cv = 5,
               verbose = True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8801  0.8792  0.8756  0.8708  0.8714  0.8754  0.0038
MAE (testset)     0.6769  0.6716  0.6737  0.6694  0.6690  0.6721  0.0029
Fit time          5.27    5.07    5.17    5.03    5.16    5.14    0.08
Test time         0.30    0.17    0.28    0.16    0.31    0.24    0.06
{'fit_time': (5.267619848251343,
  5.07274317741394,
  5.165613651275635,
  5.026052236557007,
  5.164974689483643),
 'test_mae': array([0.6769408 , 0.6715915 , 0.67370635, 0.66943497, 0.66896851]),
 'test_rmse': array([0.8800656 , 0.87922792, 0.87563498, 0.87084541, 0.87144143]),
 'test_time': (0.2976977825164795,
  0.17068076133728027,
  0.2832474708557129,
  0.16249442100524902,
  0.3090200424194336)}
```

VII. Hyperparameter Tuning

- GridSearchCV( )
  - 약 5분

```
%%time

from surprise.model_selection import GridSearchCV
```

```
# 하이퍼파라미터 딕셔너리 형태 지정
# 'n_epochs' : SGD 수행 시 반복 횟수
# 'n_factors' : 잠재 요인(K) 크기
param_grid = {'n_epochs': [20, 40, 60],
              'n_factors': [50, 100, 200]}

gs = GridSearchCV(SVD,
                  param_grid,
                  measures = ['rmse', 'mae'],
                  cv = 3)
gs.fit(data)
```

```
CPU times: user 4min 18s, sys: 613 ms, total: 4min 18s
Wall time: 4min 19s
```

- 결과 확인

```
# 최저 RMSE 점수
print(gs.best_score['rmse'])

# 최적 하이퍼파라미터 조합
print(gs.best_params['rmse'])
```

```
0.8771275338686033
{'n_epochs': 20, 'n_factors': 50}
```

VIII. 개인화 영화 추천

1) Train Dataset

- 'ratings_noh.csv'

```
from surprise import Reader
from surprise.dataset import DatasetAutoFolds

reader = Reader(line_format = 'user item rating timestamp',
                sep = ',',
                rating_scale = (0.5, 5))

# 'ratings_noh.csv' 파일로 DatasetAutoFolds 클래스 생성
data_folds = DatasetAutoFolds(ratings_file = 'ratings_noh.csv',
                              reader = reader)

# 전체 데이터를 학습데이터로 생성
trainset = data_folds.build_full_trainset()
```

2) 영화 정보 확인

- 사용자가 아직 평점을 매기지 않은 영화
  - 'userId' == 9
  - 'movieId' == 42

```
murl = 'https://raw.githubusercontent.com/rusita-ai/pyData/master/movies.csv'
movies = pd.read_csv(murl)

# userId = 9 의 movieId 데이터 추출
# movieId = 42 데이터 확인
movieIds = ratings[ratings['userId'] == 9]['movieId']

if movieIds[movieIds == 42].count() == 0:
  print('사용자 아이디 9는 영화 아이디 42의 평점 없음', '₩n')

print(movies[movies['movieId'] == 42])
```

```
사용자 아이디 9는 영화 아이디 42의 평점 없음

    movieId                   title             genres
38       42  Dead Presidents (1995)  Action|Crime|Drama
```

3) SVD - fit( )

- 추천 알고리즘 학습

```
algo = SVD(n_epochs = 20,
           n_factors = 50,
           random_state = 2045)
algo.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fddb1015250>
```

4) SVD - predict( )

- 개별 사용자의 아이템에 대한 추천 평점 예측(est = 2.96)
  - 'uid', 'iid'는 문자열로 입력
  - 'r_ui' : 기존 평점 정보는 선택 사항

```
uid = str(9)
iid = str(42)

pred = algo.predict(uid, iid, verbose = True)
```

```
user: 9          item: 42         r_ui = None   est = 2.96   {'was_impossible': False}
```

5) get_unseen_surprise( )

- 사용자가 평점을 주지 않은 영화 목록을 반환
  - 사용자가 이미 평점을 준 영화 목록을 제거

```
def get_unseen_surprise(ratings, movies, userId):
    # 'userId' 사용자가 평점을 매긴 모든 영화 리스트 생성
    seen_movies = ratings[ratings['userId'] == userId]['movieId'].tolist()

    # 모든 영화 movieId 리스트 생성
    total_movies = movies['movieId'].tolist()

    # 모든 영화 movieId 중 이미 평점을 매긴 영화의 movieId를 제외하고 리스트 생성
    unseen_movies = [movie for movie in total_movies if movie not in seen_movies]

    print(' 평점 매긴 영화수 :' , len(seen_movies), '₩n', ₩
          '추천대상 영화수 :', len(unseen_movies), '₩n', ₩
          '전체 영화수:',len(total_movies))

    return unseen_movies

unseen_movies = get_unseen_surprise(ratings, movies, 9)
```

```
평점 매긴 영화수 : 46
추천대상 영화수 : 9696
전체 영화수: 9742
```

6) recomm_movie_by_surprise( )

- 최종적으로 사용자에게 영화를 추천
  - top-10

```
def recomm_movie_by_surprise(algo, userId, unseen_movies, top_n = 10):

    # predict() 를 평점이 없는 영화에 반복 수행한 후 결과를 List 객체로 저장
    predictions = [algo.predict(str(userId), str(movieId)) for movieId in unseen_movies]

    # predictions List 객체는 surprise의 Predictions 객체를 원소로 가지고 있음
    # [Prediction(uid='9', iid='1', est=3.69), Prediction(uid='9', iid='2', est=2.98)]
    # 'est' 값으로 정렬하기 위해서 sortkey_est( ) 함수 정의
    # sortkey_est( ) 함수는 List 객체의 sort( ) 함수의 키 값으로정렬 수행
    def sortkey_est(pred):
        return pred.est

    # sortkey_est( ) 반환값의 내림 차순으로 정렬하고 top_N개의 최상위 값 추출
    predictions.sort(key = sortkey_est, reverse = True)
    top_predictions = predictions[:top_n]

    # top_N으로 추출된 영화의 정보 추출
```

```
    # 영화 아이디, 추천 예상 평점, 영화 제목
    top_movie_ids = [int(pred.iid) for pred in top_predictions]
    top_movie_rating = [pred.est for pred in top_predictions]
    top_movie_titles = movies[movies.movieId.isin(top_movie_ids)]['title']
    top_movie_preds = [(id, title, rating) for id, title, rating in ₩
                       zip(top_movie_ids, top_movie_titles, top_movie_rating)]

    return top_movie_preds
```

7) 최종 추천 결과

```
top_movie_preds = recomm_movie_by_surprise(algo,
                                           9,
                                           unseen_movies,
                                           top_n = 10)

print('##### top-10 추천 영화 리스트 #####', '₩n')
for top_movie in top_movie_preds:
    print(top_movie[1], ':', top_movie[2])
```

```
##### top-10 추천 영화 리스트 #####

Pulp Fiction (1994) : 4.292320110925793
Shawshank Redemption, The (1994) : 4.280575006684376
Schindler's List (1993) : 4.223054673027752
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964) : 4.1918592929885
Godfather, The (1972) : 4.191330334239697
Rear Window (1954) : 4.175787864055172
Monty Python and the Holy Grail (1975) : 4.167985211593734
Lawrence of Arabia (1962) : 4.139061529698701
Goodfellas (1990) : 4.133207055423318
Fight Club (1999) : 4.131297142412626
```

#

#

#

The End

#

#

#

✓ 0초 오전 11:49에 완료됨