((بسمه تعالی))

گزارش پیاده‌سازی پروژه

پرداختن به مشکل شروع سرد در پالایش همکارانه

از طریق یادگیری بدون برچسب مثبت و پیش‌بینی چند هدف

فهرست مطالب

مقدمه ۲

دیتاست ۲

پیش پردازش ۳

شرح کلی پیاده‌سازی و ارزیابی مدل‌ها ٦

نتایج ارزیابی ۱۱

کتابخانه‌ها ۱۱

منبع ۱۲

مقدمه

این پروژه بر اساس مقاله [1] و توسط پایتون و در محیط google colab پیاده‌سازی شده‌است.

در این مقاله، یک رویکرد دو مرحله‌ای جدید برای رسیدگی به مسئله شروع سرد آیتم‌ها ارائه شده است. به این صورت که در ابتدا، تعاملات آیتم - کاربر را در محیط یادگیری بدون برچسب مثبت (PU)[1] مشاهده می‌کنیم. منظور از محیط یادگیری PU این است که جفت‌های کاربر-آیتم بدون هیچ گونه تعامل قبلی، داده‌های بدون برچسب هستند و نباید به عنوان نمونه‌های منفی در نظر گرفته شوند. در ادامه بر اساس رویکردی مبتنی بر پالایش همکارانه، ماتریس تعامل کاربران و آیتم‌های گرم (آیتم‌هایی با تعاملات قبلی) را بازسازی می‌کنیم و امتیازات از دست رفته را به دست می‌آوریم. در این پروژه از مدل SVD برای بازسازی ماتریس تعامل کاربران و آیتم‌های گرم استفاده شده است. سپس این مدل توسط معیار NDCG ارزیابی می‌شود. در مرحله دوم، یک رگرسور چندمنظوره قیاسی(MTR)[2] بر روی این ماتریس تعامل بازسازی‌شده آموزش داده می‌شود و سپس تعاملات را برای آیتم‌های جدیدی که وارد سیستم می‌شوند، پیش‌بینی می‌کند. در این پروژه برای این منظور از مدل جنگل تصادفی(RF) استفاده شده است و ارزیابی این مدل توسط معیار MAE صورت گرفته است.

دیتاست

در این پروژه از دیتاست MovieLens 1M استفاده شده است که شامل امتیازات صریح کاربران در مورد فیلم‌ها است و شامل سه فایل جدا تحت عنوان‌های users، movies و ratings است. که عملا از users هیچ استفاده‌ای نشده است.

```
[4]  # Reading users file
     users = pd.read_csv("/content/drive/MyDrive/RS/ml-1m/users.dat", sep="::", engine="python", header=None,
                         names=['user_id', 'gender', 'zipcode', 'age_desc', 'occ_desc'])

     # Show the first few rows of the DataFrame
     users.head()
```

	user_id	gender	zipcode	age_desc	occ_desc
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

[1] positive unlabeled
[2] multi-target regressor

```
[4]  # Reading users file
     users = pd.read_csv("/content/drive/MyDrive/RS/ml-1m/users.dat", sep="::", engine="python", header=None,
                         names=['user_id', 'gender', 'zipcode', 'age_desc', 'occ_desc'])

     # Show the first few rows of the DataFrame
     users.head()
```

	user_id	gender	zipcode	age_desc	occ_desc
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

```
     # Reading movies file
     movies = pd.read_csv("/content/drive/MyDrive/RS/ml-1m/movies.dat", sep="::", engine="python", header=None,
                         names=['movie_id', 'title', 'genres'], encoding='latin-1')

     # Show the first few rows of the DataFrame
     movies.head()
     # print(movies.movie_id.unique().shape[0])
```

	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

دیتاست اصلی مورد استفاده، دیتاست ratings است که شامل ۶۰۴۰ کاربر و ۳۷۰۶ فیلم منحصر به فرد است.

```
[6]  # count the number of unique users and movies.
     n_users = ratings.user_id.unique().shape[0]
     n_movies = ratings.movie_id.unique().shape[0]
     print('Number of users = ' + str(n_users) + ' | Number of movies = ' + str(n_movies))

     Number of users = 6040 | Number of movies = 3706
```

پیش پردازش

بنابر آنچه در مقاله [1] تعیین شده است، در ابتدا کاربران و آیتم‌هایی با تعاملات کم‌تر نسبت به یک آستانه (برای این دیتاست، مقدار این آستانه ۳۰ است) از دیتاست ratings حذف می‌شوند.

```
[7]  # Drop users and items with less than 30 interactions from ratings

     # Compute the number of interactions for each user and each movie
     user_interactions = ratings.groupby('user_id')['movie_id'].count()
     movie_interactions = ratings.groupby('movie_id')['user_id'].count()

     # Get a list of active users and active movies
     active_users = user_interactions.loc[user_interactions >= 30].index.tolist()
     active_movies = movie_interactions.loc[movie_interactions >= 30].index.tolist()

     # Filter the ratings dataset to include only interactions from active users and active movies
     ratings_filtered = ratings[(ratings['user_id'].isin(active_users)) & (ratings['movie_id'].isin(active_movies))]
     # print(ratings_filtered)

     # Compute the number of dropped users and items
     num_dropped_users = len(user_interactions) - len(active_users)
     num_dropped_movies = len(movie_interactions) - len(active_movies)
     print(f"Dropped {num_dropped_users} users and {num_dropped_movies} items.")

     Dropped 751 users and 870 items.
```

سپس برای داشتن آیتم‌های یکسان، آیتم های داخل movies نیز مطابق با آیتم‌های باقی مانده قرار داده می‌شوند.

```
# Drop items with less than 30 interactions from movies

# Perform a left join on the movies and ratings datasets
merged = pd.merge(movies, ratings_filtered, on='movie_id', how='left')
# Drop rows with null values in the rating column
merged = merged.dropna(subset=['rating'])
# Keep only the movie_id, title, and genres columns from the merged dataset
merged = merged[['movie_id', 'title', 'genres']]

# Drop duplicates from the merged dataset
merged = merged.drop_duplicates()

# Perform a left join on the merged dataset and the movies dataset
final_movies = pd.merge(merged, movies, on=['movie_id', 'title', 'genres'], how='left')
final_movies
```

	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
2831	3948	Meet the Parents (2000)	Comedy
2832	3949	Requiem for a Dream (2000)	Drama
2833	3950	Tigerland (2000)	Drama
2834	3951	Two Family House (2000)	Drama
2835	3952	Contender, The (2000)	Drama|Thriller

2836 rows × 3 columns

از دیتاست movies که حاوی ویژگی genres است برای در نظر گرفتن ویژگی های اضافی برای فیلم ها استفاده شده است. دیتاست movies شامل ۱۸ نوع ژانر مختلف از فیلم‌ها است.

```
There are 18 genre labels.
Counter({'Animation': 96,
         "Children's": 225,
         'Comedy': 948,
         'Adventure': 254,
         'Fantasy': 64,
         'Romance': 373,
         'Drama': 1079,
         'Action': 453,
         'Crime': 160,
         'Thriller': 418,
         'Horror': 264,
         'Sci-Fi': 246,
         'War': 125,
         'Musical': 103,
         'Documentary': 54,
         'Mystery': 92,
         'Film-Noir': 35,
         'Western': 53})
```

یکی از چالش‌هایی که در پروژه با آن مواجه هستیم تبدیل امتیازات و ویژگی آیتم‌ها به شکلی است که برای مدل قابل فهم باشد. در ادامه، دو ماتریس ساخته می‌شوند. ماتریس اول برای ویژگی genre برای هر فیلم به حالت

یک ماتریس با مقادیر باینری تبدیل شده است. به این صورت که اگر فیلم در یک ژانر خاص قرار بگیرد مقدار آن فیلم برای آن ژانر ۱ است و مقدار ۰ به معنی این است که در آن ژانر نیست.

```
# we need to manipulate the genres column so that each genre is represented as a separate binary feature
# "1" indicates that the movie falls under a given genre, while "0" does not.
genres = list(genres_counts.keys())

for g in genres:
    final_movies[g] = final_movies['genres'].transform(lambda x: int(g in x))
final_movies[genres].head()
```

	Animation	Children's	Comedy	Adventure	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Sci-Fi	War	Musical	Documentary	Mystery	Film-Noir	West
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

ماتریس دوم، ماتریس تعامل کاربر-آیتم است. این ماتریس تعامل بسیار پراکنده است، یعنی درصد کمی از تعاملات مثبت کاربر-آیتم وجود دارد در حالی که بیشتر جفت‌ها به عنوان صفر علامت‌گذاری شده‌اند. این تنظیم ذاتاً در محدوده داده‌های PU قرار می گیرد. این بدان معناست که کاربر u_i مورد i_j را می‌پسندد اگر امتیاز مثبت داشته باشیم اما وقتی $y(u_i, i_j) = 0$ نتیجه قطعی نیست. در واقع، یک مقدار صفر مبهم است و می‌تواند به این معنی باشد که کاربر آیتم مربوطه را دوست ندارد، بلکه همچنین می‌تواند به این معنی باشد که آیتم هنوز به کاربر ارائه نشده است.

```
# Create a user-item matrix from the ratings dataset
user_item_matrix = ratings_filtered.pivot(index = 'user_id', columns ='movie_id', values = 'rating').fillna(0)
user_item_matrix.head()
```

movie_id	1	2	3	4	5	6	7	8	9	10	...	3937	3943	3945	3946	3947	3948	3949	3950	3951	3952
user_id																					
1	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 2836 columns

شرح کلی پیاده‌سازی و ارزیابی مدل‌ها

از آنجا که این پروژه، یک رویکرد دو مرحله‌ای برای رسیدگی به مسئله شروع سرد آیتم‌ها است، در ابتدا آیتم‌های دیتاست را به صورت تصادفی به دو بخش مجزا تقسیم می‌کنیم. یک قسمت شامل آیتم‌های گرم برای مرحله انتخاب مدل، و دیگری شامل آیتم‌های سرد برای ارزیابی توصیه شروع سرد.

```
# Separate the movies data into warm and cold items
warm_items = final_movies.sample(frac=0.5, random_state=42)
cold_items = final_movies.drop(warm_items.index)
print('Number of warm items:', len(warm_items))
print('Number of cold items:', len(cold_items))

# Get the movie ids of the warm items
# warm_item_ids = warm_items['movie_id'].tolist()
# Separate the user-item matrix based on the warm items
warm_item_ids = warm_items['movie_id'].values
warm_user_item_matrix = user_item_matrix.loc[:, user_item_matrix.columns.isin(warm_item_ids)]
print('User-item matrix for warm items:')
warm_user_item_matrix.head()
```

```
Number of warm items: 1418
Number of cold items: 1418
User-item matrix for warm items:
```

movie_id	3	8	9	14	19	21	24	26	30	31	...	3930	3936	3943	3946	3947	3948	3949	3950	3951	3952
user_id																					
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1418 columns

در این پروژه از مدل SVD با ۲۶ ویژگی پنهان[3](طبق پارامترهای تعیین شده در مقاله) برای برای بازسازی ماتریس تعامل کاربران و آیتم‌های گرم در مرحله اول استفاده شده است.

```
from sklearn.decomposition import TruncatedSVD

# Create SVD model
svd_model = TruncatedSVD(n_components=26, random_state=42)

# Fit SVD model on warm user-item matrix
svd_model.fit(warm_user_item_matrix)
```

```
TruncatedSVD
TruncatedSVD(n_components=26, random_state=42)
```

[3] latent features

ماتریس تعامل کاربران و آیتم‌های گرمی که به عنوان ورودی مرحله دوم به مدل داده می‌شود، به این صورت در نظر گرفته شده که اگر امتیازی از قبل در دیتاست موجود نبود، از مقدار امتیاز پیش بینی شده توسط مدلSVD استفاده می‌شود. در غیر این صورت مقدار امتیاز واقعی مستقیما در ماتریس قرار می‌گیرد.

```
# Predict missing ratings
predicted_ratings = svd_model.inverse_transform(svd_model.transform(warm_user_item_matrix))

# Create new user-item matrix based on actual ratings and predicted ratings
new_user_item_matrix = pd.DataFrame(predicted_ratings, columns=warm_user_item_matrix.columns, index=warm_user_item_matrix.index)
new_user_item_matrix[warm_user_item_matrix > 0] = warm_user_item_matrix[warm_user_item_matrix > 0]
# Print SVD predicting matrix
print('SVD Predicting Matrix:')
# print(new_user_item_matrix.head())
new_warm_user_item_matrix=pd.DataFrame(predicted_ratings, columns=warm_user_item_matrix.columns, index=warm_user_item_matrix.index)
new_warm_user_item_matrix
```

SVD Predicting Matrix:

movie_id	3	8	9	14	19	21	24	26	30	31	...	3930	3936	3943	3946	39
user_id																
1	0.005583	0.124300	-0.012755	0.022333	0.034696	-0.620668	0.269028	0.001234	0.003677	0.051206	...	-0.099339	-0.035061	-0.001930	-0.015127	-0.0734
2	0.231349	0.036482	0.093911	0.074976	0.126905	1.000000	0.166180	0.136580	0.024567	0.149811	...	0.038136	0.143469	-0.074376	0.049394	-0.0061
3	0.126691	0.052836	-0.018930	-0.024144	0.006118	0.459064	0.184469	-0.039941	-0.038194	-0.051926	...	0.030753	0.069450	-0.040373	0.023188	-0.0198
5	-0.281341	0.007083	-0.114459	0.312341	-0.237505	1.717995	1.000000	0.222151	0.332732	-0.050551	...	-0.056431	0.040338	0.134757	-0.034698	0.0121
6	0.433064	0.073913	0.011860	-0.141851	-0.043891	0.011813	0.115508	-0.020307	-0.016056	0.128417	...	0.149743	0.092926	0.085940	0.096718	0.0823
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6035	1.000000	0.051635	-0.041832	0.418575	0.554191	2.000000	1.078232	2.000000	0.154516	2.000000	...	-0.057581	0.072793	-0.020094	-0.067353	-0.0636
6036	-0.023371	0.104167	-0.127492	1.111825	-0.182887	3.000000	2.000000	3.000000	4.000000	0.385661	...	1.122148	0.437804	0.226608	-0.031585	0.2694
6037	-0.284593	-0.032267	0.043971	-0.058239	-0.128803	0.362810	0.515693	-0.136651	-0.034638	-0.070725	...	0.225938	0.031248	-0.018277	0.012869	0.0534
6039	0.214310	0.020286	0.023785	-0.025268	0.056373	0.448227	-0.251988	0.089812	-0.075457	-0.008429	...	0.353344	0.318577	-0.025493	-0.051958	0.0978
6040	-0.503348	-0.004169	0.030778	0.422650	0.070825	0.901077	-0.263229	0.106081	3.000000	-0.135006	...	0.003801	0.059756	0.262879	0.036310	0.0639

5289 rows × 1418 columns

ارزیابی مدل SVD توسط معیار NDCG و بر اساس اعتبارسنجی متقابل ۵-fold [۴] صورت گرفته است.

[4] 5-fold cross validation

```python
from sklearn.model_selection import KFold
from sklearn.metrics import ndcg_score

# Create 5-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

ndcg_scores = []

# Loop over folds
for train_idx, test_idx in kf.split(new_user_item_matrix):
    # Split into training and testing sets
    train = new_user_item_matrix.iloc[train_idx]
    test = new_user_item_matrix.iloc[test_idx]

    # Fit SVD model on training set
    svd_model.fit(train)

    # Predict missing ratings on test set
    predicted_ratings = svd_model.inverse_transform(svd_model.transform(test))

    # Calculate NDCG score
    ndcg_scores.append(ndcg_score(test.values, predicted_ratings, k=10))

print('NDCG Score:', sum(ndcg_scores) / len(ndcg_scores))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1658: FutureWarning: ndcg_score should not be used on negative y_true values. ndcg_score wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1658: FutureWarning: ndcg_score should not be used on negative y_true values. ndcg_score wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1658: FutureWarning: ndcg_score should not be used on negative y_true values. ndcg_score wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1658: FutureWarning: ndcg_score should not be used on negative y_true values. ndcg_score wi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py:1658: FutureWarning: ndcg_score should not be used on negative y_true values. ndcg_score wi
  warnings.warn(
NDCG Score: 0.827030452468067
```

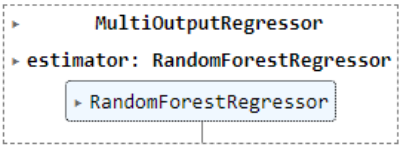
در ادامه بر اساس ماتریس ویژگی‌ها و آیتم‌های سرد و گرم، ماتریس ویژگی آیتم‌های گرم و ماتریس ویژگی آیتم‌های سرد ساخته می‌شود.

```python
warm_items = warm_items.set_index('movie_id').sort_index(ascending=True)
warm_movie_features = warm_items[genres]
warm_movie_features

cold_items = cold_items.set_index('movie_id').sort_index(ascending=True)
cold_movie_features = cold_items[genres]
cold_movie_features
```

در مرحله دوم از مدل Random Forest برای پیش بینی امتیازات کاربران در مورد آیتم‌های سرد بر اساس ماتریس تعامل کاربران و آیتم‌های گرم و ماتریس ویژگی آیتم‌ها استفاده می‌شود. چالش دیگری که در این مدل با آن روبرو هستیم، در مقاله پارامترهای Random Forest به صورت n_estimators=200 (تعداد درختان) و min_samples_leaf=5 (حداقل تعداد نمونه مورد نیاز برای قرار گرفتن در یک گره برگ) تعیین شده است. ولی از آن جایی که با این پارامترها ساخت و ارزیابی مدل، مدت زمان خیلی زیادی طول می‌کشد، در این پروژه پارامتر تعداد درختان به صورت n_estimators=20 کاهش یافته است.

```
# Train a multi-output random forest model
rf_model = MultiOutputRegressor(RandomForestRegressor(n_estimators=20, min_samples_leaf=5, random_state=42))
rf_model.fit(warm_movie_features, new_warm_user_item_matrix.T)
```




```
# Separate the user-item matrix based on the cold items
cold_item_ids = cold_items['movie_id'].values
cold_user_item_matrix = user_item_matrix.loc[:, user_item_matrix.columns.isin(cold_item_ids)]
cold_user_item_matrix.shape

# Use Random Forest model to predict ratings
predicted_ratings_rf = rf_model.predict(cold_movie_features)

# Reshape predictions to match matrix dimensions
predicted_cold_item_user_matrix = pd.DataFrame(predicted_ratings_rf, columns=cold_user_item_matrix.T.columns, index=cold_user_item_matrix.T.index)

print('Predicted Cold User-Item Matrix:')
predicted_cold_item_user_matrix.T
```

Predicted Cold User-Item Matrix:

movie_id	1	2	4	5	6	7	10	11	12	13	...	3916	3918	3920	3921	392
user_id																
1	1.631304	0.197065	0.011844	0.026876	-0.062546	-0.010705	0.056969	0.012714	-0.046099	0.905259	...	0.196115	-0.023681	0.308376	0.026876	0.02687
2	-0.046350	-0.057376	0.252847	0.097540	1.486922	0.082489	0.731711	0.169118	0.159041	-0.153052	...	0.422711	-0.013060	0.422711	0.097540	0.09754
3	0.971571	0.280931	0.077717	0.071362	0.214690	-0.039743	0.540812	-0.002867	0.066080	0.018454	...	0.021365	0.070614	0.135517	0.071362	0.07136
5	0.955541	0.172702	0.735063	0.278255	1.151127	0.178122	0.190764	0.286102	0.116888	0.224021	...	0.500289	0.040096	0.500289	0.278255	0.27825
6	0.537957	0.226741	0.065586	0.047292	0.127729	0.723495	0.221043	0.568726	0.013629	0.437742	...	0.049994	-0.005873	0.126966	0.047292	0.04729
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6035	0.850212	0.441905	0.539793	0.332335	0.205464	0.527248	-0.017220	0.403560	0.149416	0.079911	...	0.470138	-0.039793	0.470138	0.332335	0.33233
6036	2.569537	0.690358	1.852592	0.969320	1.401469	1.113366	0.254940	2.086015	0.933872	2.145526	...	1.832123	1.114387	1.832123	0.969320	0.96932
6037	0.338499	-0.183334	0.588414	0.174088	0.736032	0.192378	0.326079	0.425703	0.121464	-0.029680	...	0.325854	0.264898	0.315476	0.174088	0.17408
6039	0.862402	0.128746	0.448945	0.304891	0.076278	0.337042	-0.013438	0.167687	0.038944	0.247912	...	0.041446	-0.023914	0.041446	0.304891	0.30489
6040	0.603950	-0.201855	1.185430	0.565165	0.592682	0.356304	0.102959	0.796763	0.257635	0.041410	...	1.010672	0.278102	1.010672	0.565165	0.56516

5289 rows × 1418 columns

پس از ساخت ماتریس امتیازات کاربران و آیتم‌های سرد، بر اساس امتیازات پیش بینی شده، مدل به هر کاربر ۱۰ عدد از فیلم‌های دسته آیتم‌های سرد را توصیه می‌کند. فرمت توصیه‌ها به صورت زیر است:

User_id: [movie_id1, movie_id2,…,movie_id10]

```
# Recommend top 10 movies for each user
top_recommendations = {}
for userId in predicted_cold_item_user_matrix.T.index:
    user_ratings = predicted_cold_item_user_matrix.T.loc[userId].sort_values(ascending=False)
    top_recommendations[userId] = list(user_ratings.head(10).index)

print('Top 10 movie recommendations for each user:')
print(top_recommendations)
```

```
Top 10 movie recommendations for each user:
{1: [2102, 588, 2080, 2081, 2096, 595, 661, 1022, 1024, 1029], 2: [1210, 2468, 1722, 1264, 3584, 3705, 969, 1374, 2275, 1371], 3: [3805, 2422, 1287, 592, 2370, 5
```

ارزیابی این مدل توسط معیار MAE و بر اساس اعتبارسنجی متقابل ۵-fold صورت گرفته است.

```
# Evaluate model with 5-fold cross validation
scores = cross_val_score(rf_model, new_warm_user_item_matrix.T, warm_movie_features,
                         cv=5, scoring='neg_mean_absolute_error')

print('MAE scores:', -scores)
print('Average MAE:', -np.mean(scores))
```

```
MAE scores: [0.05103558 0.06359798 0.05003746 0.05541439 0.0768615 ]
Average MAE: 0.059389383225328475
```

نتایج ارزیابی

نتیجه اجرای پروژه برای قسمت اول، یعنی اجرای مدلSVD برای برای بازسازی ماتریس تعامل کاربران و آیتم‌های گرم، به صورت زیر بوده است که نشان دهنده عملکرد خوب مدل در پیش بینی امتیازات از دست رفته است:

```
NDCG Score: 0.827030452468067
```

در انتها بر اساس شناسه کاربران و فیلم ها و همچنین ماتریس تعامل بازسازی شده برای آیتم‌های سرد می‌توانیم این آیتم‌ها را به کاربران پیشنهاد بدهیم.

و نتیجه نهایی اجرای پروژه در ارزیابی مدل Random Forest، بر اساس معیار MAE به صورت زیر بوده است:

```
MAE scores: [0.05103558 0.06359798 0.05003746 0.05541439 0.0768615 ]
Average MAE: 0.059389383225328475
```

باید توجه داشت این مقدار MAE براساس ساخت مدل Random Forest با تعداد ۲۰ درخت تصمیم است و ممکن است با افزایش تعداد درختان دقت نیز افزایش یابد.

کتابخانه‌ها

در طول پروژه و بر اساس نیازهای مختلف، از کتابخانه‌های زیر استفاده شده است:

```
# Import libraries
import pandas as pd
import numpy as np
import math
```

```
from sklearn.decomposition import TruncatedSVD
```

```
from sklearn.model_selection import KFold
from sklearn.metrics import ndcg_score
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
```

منبع

[1] A. Gharahighehi, K. Pliakos and C. Vens, "Addressing the Cold-Start Problem in Collaborative Filtering Through Positive-Unlabeled Learning and Multi-Target Prediction," in IEEE Access, vol. 10, pp. 117189-117198, 2022