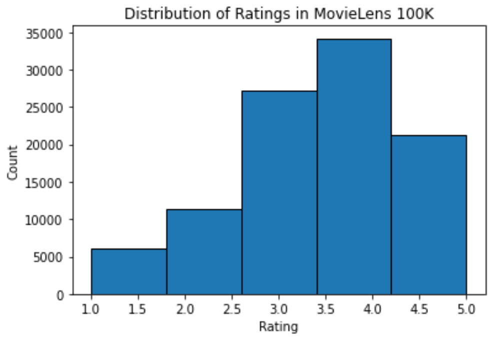
Collaborative Filtering Recommender System For Movie

Mehrdad Mohammadian

Dataset

Movie-lens 100k: https://grouplens.org/datasets/movielens/100k/



number of users: 943

number of items: 1682

matrix sparsity: 0.936953

Code

Tools: Python, Surprise Library

Prediction Algorithm: KNNWithMeans

KNNWithMeans: A basic collaborative filtering algorithm, taking into account the mean ratings of each user.

The prediction $\hat{r}_{ui}$ is set as:

For user-based system:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

For Item-based system:

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

Metric Results in test set

MAE: 0.7395

RMSE: 0.9436

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j|$$

go to the next page ...

Collaborative Filtering Recommender System For Movie

Mehrdad Mohammadian - fall 2021

Dataset: MovieLens 100k

Import Libs

```
In [ ]:  # install surprise lib
         ! pip install scikit-surprise
```

```
In [165]: import pandas as pd
          from surprise import KNNWithMeans
          from surprise import accuracy
          from surprise.model_selection import GridSearchCV
          from sklearn.model_selection import train_test_split
          from surprise import Reader, Dataset
```

```
In [371]: reader = Reader(rating_scale=(1, 5))
```

Find The Best HyperParameters

Load Dataset

```
In [374]: cols = ['user_id', 'movie_id', 'rating', 'timestamp']
          data = pd.read_csv('u.data',  sep='\t', names=cols)
          data.drop(columns='timestamp', inplace=True)
          data = Dataset.load_from_df(data, reader)
```

Grid Search

```
In [381]: sim_options = {
              "name": ["cosine", "pearson"],
              "user_based": [False, True],
          }
          bsl_options = {
              'method': ['sgd'],
              'learning_rate': [0.0005, 0.005]
          }
          param_grid = {
              "sim_options": sim_options,
              'bsl_options': bsl_options
          }

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

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
```

Results

```
In [382]: gs.best_params
```

```
Out[382]: {'mae': {'bsl_options': {'learning_rate': 0.0005, 'method': 'sgd'},
            'sim_options': {'name': 'pearson', 'user_based': False}},
           'rmse': {'bsl_options': {'learning_rate': 0.0005, 'method': 'sgd'},
            'sim_options': {'name': 'cosine', 'user_based': False}}}
```

Train The Model

Load Dataset

```
In [384]: cols = ['user_id', 'movie_id', 'rating', 'timestamp']
          data = pd.read_csv('u.data',  sep='\t', names=cols)
          data.drop(columns='timestamp', inplace=True)
```

```
In [385]: data.head()
```

Out[385]:

	user_id	movie_id	rating
0	196	242	3
1	186	302	3
2	22	377	1
3	244	51	2
4	166	346	1

```
In [504]: sparse = data.pivot(index='user_id', columns='movie_id', values='rating')
```

```
In [505]: sparse
```

Out[505]:

movie_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	3
user_id																														
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	2.0	5.0	5.0	5.0	5.0	5.0	3.0	4.0	5.0	4.0	1.0	4.0	4.0	3.0	4.0	3.0	2.0	4.0	1.0	3
2	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	4.0	4.0	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	4.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	3.0	NaN	NaN	4.0	3.0	NaN	NaN	NaN	4.0	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
939	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
940	NaN	NaN	NaN	2.0	NaN	NaN	4.0	5.0	3.0	NaN	NaN	4.0	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
941	5.0	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
942	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
943	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	4.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	4.0	4.0	NaN	NaN	4.0	4.0	NaN	Na

943 rows × 1682 columns

```
In [386]: X = data.copy()
          y = data['user_id']
          X_train, X_test, _ ,_ = train_test_split(X, y, test_size = 0.20, stratify=y, random_state=42)
```

```
In [387]: print('Data Shape:', X_train.shape, X_test.shape)
```

```
Data Shape: (80000, 3) (20000, 3)
```

```
In [388]: X_train = Dataset.load_from_df(X_train[['user_id', 'movie_id', 'rating']], reader)
          X_test = Dataset.load_from_df(X_test[['user_id', 'movie_id', 'rating']], reader)

          X_train = X_train.build_full_trainset()
          X_test = X_test.build_full_trainset()
          X_test = X_test.build_testset()
```

Train ¶

```
In [391]: options = {
              'bsl_options': {'learning_rate': 0.0005, 'method': 'sgd'},
              'sim_options': {'name': 'pearson', 'user_based': False}
          }
          model = KNNWithMeans(sim_options=options['sim_options'], bsl_options=options['bsl_options'])
          model.fit(X_train)
```

```
Computing the pearson similarity matrix...
Done computing similarity matrix.
```

```
Out[391]: <surprise.prediction_algorithms.knns.KNNWithMeans at 0x7fd365a01f10>
```

Test With Metrics

In [392]:

```
predictions= model.test(X_test)
accuracy.mae(predictions)
accuracy.rmse(predictions)
```

```
MAE:  0.7395
RMSE: 0.9436
```

Out[392]:

```
0.94358288574682
```

Fill sparse matrix with predicted rates

after this section we would have a dense matrix

In [506]:

```
for u_index, row in sparse.iterrows():
    user_num = u_index - 1
    for i_index, item in enumerate(row):
        item_num = i_index+1
        item_rate = sparse.iloc[user_num][item_num]
        if pd.isnull(item_rate):
            sparse.iloc[user_num][item_num] = model.predict(user_num+1, item_num).est
            # print(f'user {user_num+1}  item {item_num} is {item_rate} and predict is {predict.est}')
```

In [507]:

```
sparse
```

Out[507]:

movie_id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
user_id																		
1	5.000000	3.000000	4.000000	3.000000	3.000000	5.000000	4.000000	1.000000	5.000000	3.000000	2.000000	5.000000	5.000000	5.000000	5.000000	5.000000	3.000000	4.0
2	4.000000	3.437731	3.125313	3.591362	3.556621	3.979347	3.967059	4.080176	4.056785	2.000000	3.992702	4.651292	4.000000	4.000000	3.964021	3.373254	2.990827	3.1
3	3.150601	2.658846	2.492210	2.632303	2.867090	3.066745	3.204607	3.064673	3.248655	2.966963	3.337028	3.527199	2.684429	3.356114	2.972057	2.402100	2.694866	2.0
4	5.000000	4.143578	3.727402	4.575867	4.419570	5.000000	4.914418	5.000000	4.554399	5.000000	4.000000	4.909718	4.999127	5.000000	4.522477	4.166339	3.847748	4.0
5	4.000000	3.000000	2.361272	3.116525	2.601415	3.226517	3.419331	3.506003	3.519460	3.276340	3.363945	3.763923	2.871801	3.541435	3.117859	2.852480	4.000000	3.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
939	4.977866	4.037906	4.141668	3.785063	4.483867	4.904866	5.000000	4.680618	5.000000	4.393106	5.000000	5.000000	3.831734	4.410353	5.000000	4.370084	3.906283	4.0
940	3.769243	2.974195	2.536475	2.000000	3.039524	3.473083	4.000000	5.000000	3.000000	3.113876	3.638804	4.000000	2.738687	3.000000	3.643131	2.717572	2.653324	2.8
941	5.000000	3.448209	3.315365	3.997877	3.613486	4.182333	4.000000	4.374142	4.243871	4.138473	4.105365	4.541268	3.731191	4.148017	4.000000	3.385663	3.178876	3.4
942	4.459274	3.791035	3.884371	4.132217	4.145948	4.323595	4.189632	4.631328	4.446121	4.406334	4.457771	4.897983	3.903486	4.418340	4.352290	3.702532	3.661877	3.1
943	3.899458	5.000000	3.140142	3.311576	3.650609	3.423774	3.456891	4.213548	3.000000	3.903049	4.000000	5.000000	3.582276	3.764817	3.822782	3.096024	2.957741	3.0

943 rows × 1682 columns

Recommend movies for a system user

Load movie items dataset

In [534]:

```
i_cols = ['movie_id', 'title' ,'release date','video release date', 'IMDb URL', 'unknown', 'Action', 'Adventure',
'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
movies = pd.read_csv('u.item',  sep='|', names=i_cols, encoding='latin-1')
movies.head()
```

Out[534]:

	movie_id	title	release date	video release date	IMDb URL	unknown	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Mu
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	1	0	0	0	0	0	0	
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	0	0	0	0	0	0	0	
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	0	0	0	0	0	0	0	
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	1	0	0	1	0	0	0	
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	0	1	0	1	0	0	0	

In [558]:

```
def recommend(user_number, num_items):
    for i in  sparse.iloc[user_number].sort_values(ascending=False).head(num_items).keys():
        print(movies.iloc[i]['title'])

# recommend top 10 movie item to user 456
recommend(456, 10)
```

```
Everyone Says I Love You (1996)
Haunted World of Edward D. Wood Jr., The (1995)
Supercop (1992)
Graduate, The (1967)
Guantanamera (1994)
Nikita (La Femme Nikita) (1990)
Evil Dead II (1987)
Patton (1970)
Indiana Jones and the Last Crusade (1989)
Striptease (1996)
```

The End