Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4
по дисциплине «Методы машинного обучения»
«Создание рекомендательной системы»

ИСПОЛНИТЕЛЬ:

Цветкова Алена
Группа ИУ5-21М

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

Москва, 2021

Цель работы: изучение разработки рекомендательных моделей

Исходные данные ratings:

	book	user	rating
0	1	314	5
1	1	439	3
2	1	588	5
3	1	1169	4
4	1	1185	4

```
min = ratings['rating'].min()
max = ratings['rating'].max()
median = ratings['rating'].median()
mean = ratings['rating'].mean()
print(" Минимальная оценка: {0}\n Максимальная оценка: {1}\n Медианное значение: {2}\n Среднее значение {3}"
      .format(min, max, median, mean))
```

```
Минимальная оценка: 1
Максимальная оценка: 5
Медианное значение: 4.0
Среднее значение 3.8565335989797873
```

SVD-разложение

In [16]:
```
from sklearn.model_selection import train_test_split

train_ratings, test_ratings = train_test_split(ratings, test_size=0.3)

print('Размер обучающей выборки: {}'.format(train_ratings.shape))
print('Размер тестовой выборки: {}'.format(test_ratings.shape))
```

```
Размер обучающей выборки: (687229, 3)
Размер тестовой выборки: (294527, 3)
```

In [161]:
```
train_matrix = np.zeros((cnt_users, cnt_books))
for line in train_ratings.itertuples():
    train_matrix[line.user - 1, line.book - 1 ] = line.rating - 1

test_matrix = np.zeros((cnt_users, cnt_books))
for line in test_ratings.itertuples():
    test_matrix[line.user - 1, line.book - 1] = line.rating - 1
```

In [162]:
```
test_matrix.shape
```

Out[162]:
```
(53424, 10000)
```

In [163]:
```
train_matrix
```

Out[163]:
```
array([[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.]])
```

In [164]:
```
u, s, vt = svds(train_matrix, k=250)
s_diag_matrix = np.diag(s)
# предсказываем
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)

print (X_pred.shape, test_matrix.shape)

# выводим метрику
print('User-based CF MSE: ' + str(rmse(X_pred, test_matrix)))
```

```
(53424, 10000) (53424, 10000)
User-based CF MSE: 2.7622900204837655
```

In [165]:
```
u, s, vt = svds(train_matrix, k=100)
s_diag_matrix = np.diag(s)
# предсказываем
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)

print (X_pred.shape, test_matrix.shape)

# выводим метрику
print('User-based CF MSE: ' + str(rmse(X_pred, test_matrix)))
```

```
(53424, 10000) (53424, 10000)
User-based CF MSE: 2.832179678621353
```

Заполним пропуски в матрице средней оценкой

In [149]:
```
r_average = ratings['rating'].mean()
train_matrix[train_matrix == 0] = np.NaN
```

In [150]:
```
r_average
```

Out[150]:
```
3.8565335989797873
```

In [153]:
```
train_matrix = pd.DataFrame(train_matrix).fillna(r_average).to_numpy()
```

In [154]:
```
train_matrix
```

Out[154]:
```
array([[3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336],
       [3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336],
       [3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336],
       ...,
       [3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336],
       [3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336],
       [3.8565336, 3.8565336, 3.8565336, ..., 3.8565336, 3.8565336,
        3.8565336]])
```

In [156]:
```
test_matrix
```

Out[156]:
```
array([[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.]])
```

In [160]:
```
u, s, vt = svds(train_matrix, k=250)
s_diag_matrix = np.diag(s)
# предсказываем
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)

print (X_pred.shape, test_matrix.shape)

# выводим метрику
print('User-based CF MSE: ' + str(rmse(X_pred, test_matrix)))
```

```
(53424, 10000) (53424, 10000)
User-based CF MSE: 1.244571565908985
```

Для каждой строчки заполняем пропуски средней оценкой по строке – это учитывает манеру проставления оценок.

In [132]:
```
i = 0
while i < train_matrix.shape[0]:
    print(i)
    k = 0
    train_str = train_matrix[i]
    train_av_2 = train_str.mean()
    if train_av_2 > 0:
        train_av = train_str[train_str != 0].mean()
    else:
        train_av = r_average
    print(train_av)
    while k < len(train_matrix[i]):
        if train_matrix[i][k] == 0:
            train_matrix[i][k] = train_av
        k = k + 1
    print(train_matrix[i])
    i = i + 1
```

In [139]:
```
train_matrix
```

Out[139]:
```
array([[2.5       , 2.5       , 2.5       , ..., 2.5       , 2.5       ,
        2.5       ],
       [3.5       , 3.5       , 3.5       , ..., 3.5       , 3.5       ,
        3.5       ],
       [2.5       , 2.5       , 2.5       , ..., 2.5       , 2.5       ,
        2.5       ],
       ...,
       [3.15384615, 3.15384615, 3.15384615, ..., 3.15384615, 3.15384615,
        3.15384615],
       [4.        , 4.        , 4.        , ..., 4.        , 4.        ,
        4.        ],
       [3.27272727, 3.27272727, 3.27272727, ..., 3.27272727, 3.27272727,
        3.27272727]])
```

In [146]:
```
u, s, vt = svds(train_matrix, k=250)
s_diag_matrix = np.diag(s)
# предсказываем
X_pred = np.dot(np.dot(u, s_diag_matrix), vt)

print (X_pred.shape, test_matrix.shape)

# выводим метрику
print('User-based CF MSE: ' + str(rmse(X_pred, test_matrix)))
```

```
(53424, 10000) (53424, 10000)
User-based CF MSE: 0.8080460521776914
```

Рекомендации с помощью TfidfVectorizer по ключевым словам для книг

```
books_result.head()
```

	book_id	title	original_title	authors	average_rating	tags	book_desc	genres
0	2767052	The Hunger Games (The Hunger Games, #1)	The Hunger Games	Suzanne Collins	4.34	to-read fantasy favorites currently-reading yo...	None	None
1	3	Harry Potter and the Sorcerer's Stone (Harry P...	Harry Potter and the Philosopher's Stone	J.K. Rowling, Mary GrandPré	4.44	to-read fantasy favorites currently-reading yo...	None	None
2	41865	Twilight (Twilight, #1)	Twilight	Stephenie Meyer	3.57	to-read fantasy favorites currently-reading yo...	None	None
3	2657	To Kill a Mockingbird	To Kill a Mockingbird	Harper Lee	4.25	to-read favorites currently-reading young-adul...	The unforgettable novel of a childhood in a sl...	Classics|Fiction|Historical|Historical Fiction...
4	4671	The Great Gatsby	The Great Gatsby	F. Scott Fitzgerald	3.89	to-read favorites currently-reading young-adul...	Alternate Cover Edition ISBN: 0743273567 (ISBN...	Classics|Fiction|Academic|School|Literature|Hi...

In [180]:
```
tf1 = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
tfidf_matrix1 = tf1.fit_transform(books_result['tags'])
cosine_sim1 = linear_kernel(tfidf_matrix1, tfidf_matrix1)
```

In [181]:
```
tfidf_matrix1
```

Out[181]:
```
<10000x144268 sparse matrix of type '<class 'numpy.float64'>'
        with 2100874 stored elements in Compressed Sparse Row format>
```

In [182]:
```
titles = books['title']
indices = pd.Series(books.index, index=books['title'])

def tags_recommendations(title):
    idx = indices[title]
    sim_scores = sorted(list(enumerate(cosine_sim1[idx])), key=lambda x: x[1], reverse=True)[1:15]
    book_indices = [i[0] for i in sim_scores if i[1] >= 0.3]
    book_corr = [i[1] for i in sim_scores if i[1] >= 0.3]
    return titles.iloc[book_indices], book_corr
```

In [187]:
```
res_title, res_corr = tags_recommendations('It')
```

In [189]:
```
res_df = pd.DataFrame(res_title)
res_df ['corr'] = res_corr
res_df
```

Out[189]:

	title	corr
242	Misery	0.656433
552	Needful Things	0.654482
304	Pet Sematary	0.646338
674	Firestarter	0.642161
348	'Salem's Lot	0.640320
704	Thinner	0.632258
669	The Dead Zone	0.617296
910	Gerald's Game	0.595837
1338	The Mist	0.582776
1138	The Tommyknockers	0.582351
943	Desperation	0.577643
914	Insomnia	0.575003
793	Doctor Sleep (The Shining, #2)	0.568818
236	Carrie	0.565448