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

Лабораторная работа №4
По курсу «Методы машинного обучения»

«Создание рекомендательной модели»

ИСПОЛНИТЕЛЬ:

Лосева Светлана Сергеевна
Группа ИУ5-24М

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Цель работы:

Изучение разработки рекомендательных моделей.

Задание:

1. Выбрать произвольный датасет, предназначенный для построения рекомендательных моделей.
2. Опираясь на материалы лекции, сформировать рекомендации для одного пользователя (объекта) двумя произвольными способами.
3. Сравнить полученные рекомендации (если это возможно, то с применением метрик).

Описание задания:

Для выполнения лабораторной работы возьмём датасет, который содержит информацию о 17,562 аниме и предпочтениях 325,772 разных пользователей. Для выполнения лабораторной работы нам потребуются лишь некоторые колонки.

Выполнение работы:

1. Фильтрация на основе содержания. Данную фильтрацию будем проводить по жанрам. Предпочитаемое нами аниме – «Кровь триединства». Манхэттенское и Евклидово расстояния дают приблизительно равные результаты
2. Коллаборативная фильтрация. Метод user-based.

Вывод:

Была проделана работа по разработке рекомендательной модели.

```
!pip install numpy pandas scikit-surprise sklearn seaborn matplotlib datetime
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.19
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.1
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f(
    |████████████████████████████████| 11.8MB 324kB/s
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-packages (0.(
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (0.:
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Collecting datetime
  Downloading https://files.pythonhosted.org/packages/73/22/a5297f3a1f92468cc737f8ce7
    |████████████████████████████████| 61kB 5.9MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/di:
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-package:
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package:
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-package:
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-package:
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loca]
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pa(
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-package:
Collecting zope.interface
  Downloading https://files.pythonhosted.org/packages/bb/a7/94e1a92c71436f934cdd2102{
    |████████████████████████████████| 256kB 42.2MB/s
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux_
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25(
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, zope.interface, datetime
Successfully installed datetime-4.3 scikit-surprise-1.1.1 zope.interface-5.4.0
```

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_erro
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, manhattan_dis
from surprise import SVD, Dataset, Reader
from surprise.model_selection import PredefinedKFold
from collections import defaultdict
from surprise.accuracy import rmse
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib_venn import venn2
%matplotlib inline
sns.set(style="ticks")
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
df = pd.read_csv('/content/gdrive/My Drive/MMO/anime.csv')
#df1=df.drop(columns=['English name','Japanese name','Type', 'Aired', 'Premiered',  'Produ
df=df.drop(columns=['MAL_ID','English name','Japanese name','Type', 'Aired', 'Premiered',
#df.drop_duplicates(subset=['Name'])
df.head (10)
```

	Name	Score	Genders	Episodes	Studios	Source	Duration	Rating
0	Cowboy Bebop	8.78	Action, Adventure, Comedy, Drama, Sci-Fi, Space	26	Sunrise	Original	24 min. per ep.	R - 17+ (violence & profanity)
1	Cowboy Bebop: Tengoku no Tobira	8.39	Action, Drama, Mystery, Sci-Fi, Space	1	Bones	Original	1 hr. 55 min.	R - 17+ (violence & profanity)
2	Trigun	8.24	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	26	Madhouse	Manga	24 min. per ep.	PG-13 - Teens 13 or older

```
df.shape
```

```
(17562, 27)
```

Robin Drama, ... or older

```
name = df['Name'].values
name[0:30]
```

```
array(['Cowboy Bebop', 'Cowboy Bebop: Tengoku no Tobira', 'Trigun',
       'Witch Hunter Robin', 'Bouken Ou Beet', 'Eyeshield 21',
       'Hachimitsu to Clover', 'Hungry Heart: Wild Striker',
       'Initial D Fourth Stage', 'Monster', 'Naruto', 'One Piece',
       'Tennis no Ouji-sama', 'Ring ni Kakero 1', 'School Rumble',
       'Sunabouzu', 'Texhnolyze', 'Trinity Blood', 'Yakitate!! Japan',
       'Zipang', 'Neon Genesis Evangelion',
       'Neon Genesis Evangelion: Death & Rebirth',
       'Neon Genesis Evangelion: The End of Evangelion',
       'Kenpuu Denki Berserk', 'Koukaku Kidoutai',
       'Rurouni Kenshin: Meiji Kenkaku Romantan - Tsuioku-hen',
       'Rurouni Kenshin: Meiji Kenkaku Romantan',
       'Rurouni Kenshin: Meiji Kenkaku Romantan - Ishinshishi e no Chinkonka',
       'Akira', '.hack//Sign'], dtype=object)
```

```
gender=df['Genders'].values
gender[0:30]
```

```
array(['Action, Adventure, Comedy, Drama, Sci-Fi, Space',
       'Action, Drama, Mystery, Sci-Fi, Space',
       'Action, Sci-Fi, Adventure, Comedy, Drama, Shounen',
       'Action, Mystery, Police, Supernatural, Drama, Magic',
       'Adventure, Fantasy, Shounen, Supernatural',
       'Action, Sports, Comedy, Shounen',
       'Comedy, Drama, Josei, Romance, Slice of Life',
       'Slice of Life, Comedy, Sports, Shounen',
       'Action, Cars, Sports, Drama, Seinen',
       'Drama, Horror, Mystery, Police, Psychological, Seinen, Thriller',
       'Action, Adventure, Comedy, Super Power, Martial Arts, Shounen',
       'Action, Adventure, Comedy, Super Power, Drama, Fantasy, Shounen',
       'Action, Comedy, Sports, School, Shounen',
```

```
           'Action, Shounen, Sports', 'Comedy, Romance, School, Shounen',
           'Action, Adventure, Comedy, Ecchi, Sci-Fi, Shounen',
           'Action, Sci-Fi, Psychological, Drama',
           'Action, Supernatural, Vampire', 'Comedy, Shounen',
           'Action, Military, Sci-Fi, Historical, Drama, Seinen',
           'Action, Sci-Fi, Dementia, Psychological, Drama, Mecha',
           'Drama, Mecha, Psychological, Sci-Fi',
           'Sci-Fi, Dementia, Psychological, Drama, Mecha',
           'Action, Adventure, Demons, Drama, Fantasy, Horror, Military, Romance, Seinen,
           'Action, Mecha, Police, Psychological, Sci-Fi, Seinen',
           'Action, Historical, Drama, Romance, Martial Arts, Samurai, Shounen',
           'Action, Adventure, Comedy, Historical, Romance, Samurai, Shounen',
           'Samurai, Historical, Drama, Shounen',
           'Action, Military, Sci-Fi, Adventure, Horror, Supernatural, Seinen',
           'Game, Sci-Fi, Adventure, Mystery, Magic, Fantasy'], dtype=object)
```

```
rating=df['Rating'].values
rating[0:30]
```

```
array(['R - 17+ (violence & profanity)', 'R - 17+ (violence & profanity)',
       'PG-13 - Teens 13 or older', 'PG-13 - Teens 13 or older',
       'PG - Children', 'PG-13 - Teens 13 or older',
       'PG-13 - Teens 13 or older', 'PG-13 - Teens 13 or older',
       'PG-13 - Teens 13 or older', 'R+ - Mild Nudity',
       'PG-13 - Teens 13 or older', 'PG-13 - Teens 13 or older',
       'PG-13 - Teens 13 or older', 'PG - Children',
       'PG-13 - Teens 13 or older', 'R - 17+ (violence & profanity)',
       'R+ - Mild Nudity', 'R - 17+ (violence & profanity)',
       'PG-13 - Teens 13 or older', 'PG-13 - Teens 13 or older',
       'PG-13 - Teens 13 or older', 'R - 17+ (violence & profanity)',
       'R+ - Mild Nudity', 'R+ - Mild Nudity', 'R+ - Mild Nudity',
       'R - 17+ (violence & profanity)', 'PG-13 - Teens 13 or older',
       'R - 17+ (violence & profanity)', 'R+ - Mild Nudity',
       'PG-13 - Teens 13 or older'], dtype=object)
```

```
%%time
tfidfv = TfidfVectorizer()
genders_matrix = tfidfv.fit_transform(gender)
genders_matrix
```

```
CPU times: user 97.4 ms, sys: 0 ns, total: 97.4 ms
Wall time: 109 ms
```

Фильтрация на основе содержания по жанрам

```
class SimpleKNNRecommender:

    def __init__(self, X_matrix, X_Name, X_Genders, X_Rating):
        """
        Входные параметры:
        X_matrix - обучающая выборка (матрица объект-признак)
        X_Name - массив названий объектов
        X_Genders - массив жанров объектов
```

```
        X_Rating - массив возрастного ограничения объектов
        """
        #Сохраняем параметры в переменных объекта
        self._X_matrix = X_matrix
        self.df = pd.DataFrame(
            {'Name': pd.Series(X_Name, dtype='str'),
            'Gender': pd.Series(X_Genders, dtype='str'),
            'Rating': pd.Series(X_Rating, dtype='str'),
            'dist': pd.Series([], dtype='float')})


    def recommend_for_single_object(self, K: int, \
                X_matrix_object, cos_flag = True, manh_flag = False):
        """
        Метод формирования рекомендаций для одного объекта.
        Входные параметры:
        K - количество рекомендуемых соседей
        X_matrix_object - строка матрицы объект-признак, соответствующая объекту
        cos_flag - флаг вычисления косинусного расстояния
        manh_flag - флаг вычисления манхэттэнского расстояния
        Возвращаемое значение: K найденных соседей
        """

        scale = 1000000
        # Вычисляем косинусную близость
        if cos_flag:
            dist = cosine_similarity(self._X_matrix, X_matrix_object)
            self.df['dist'] = dist * scale
            res = self.df.sort_values(by='dist', ascending=False)
            # Не учитываем рекомендации с единичным расстоянием,
            # так как это искомый объект
            res = res[res['dist'] < scale]

        else:
            if manh_flag:
                dist = manhattan_distances(self._X_matrix, X_matrix_object)
            else:
                dist = euclidean_distances(self._X_matrix, X_matrix_object)
            self.df['dist'] = dist * scale
            res = self.df.sort_values(by='dist', ascending=True)
            # Не учитываем рекомендации с единичным расстоянием,
            # так как это искомый объект
            res = res[res['dist'] > 0.0]

        # Оставляем K первых рекомендаций
        res = res.head(K)
        return res
```

```
trinity_blood_ind =17
name[trinity_blood_ind]
```

```
'Trinity Blood'
```

```
trinity_blood_matrix = genders_matrix[trinity_blood_ind]
```

```
trinity_blood_matrix
```

```
<1x48 sparse matrix of type '<class 'numpy.float64'>'
	with 3 stored elements in Compressed Sparse Row format>
```

```
skr1 = SimpleKNNRecommender(genders_matrix, name, gender, rating)
```

Выведем 10 аниме похожих на "кровь триединства"

```
df_new = df[['Name','Genders', 'Rating']]
df_new.loc[df_new['Name']=='Trinity Blood']
```

	Name	Genders	Rating
17	Trinity Blood	Action, Supernatural, Vampire	R - 17+ (violence & profanity)

```
# в порядке убывания схожести на основе косинусного сходства
rec1 = skr1.recommend_for_single_object(10, trinity_blood_matrix)
rec1
```

	Name	Gender	Rating	dist
15595	Yichang Shengwu Jianwenlu	Supernatural, Vampire	PG-13 - Teens 13 or older	938347.176320
16615	Noblesse	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	906943.306038
11461	Noblesse: Awakening	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	906943.306038
1644	Master Mosquiton '99	Action, Adventure, Comedy, Supernatural, Vampire	PG-13 - Teens 13 or older	902269.296938
6104	Nyanpire The Animation	Comedy, Supernatural, Vampire	G - All Ages	897786.169580
14352	Sirius	Action, Historical, Supernatural, Vampire	R - 17+ (violence & profanity)	889297.900037
	Hellsing: Psalm of	Action, Supernatural, Vampire	R - 17+ (violence	

```
# При поиске с помощью Евклидова расстояния получаем почти такой же результат. Разница во
rec2 = skr1.recommend_for_single_object(10, trinity_blood_matrix, cos_flag = False)
rec2
```

	Name	Gender	Rating	dist
15595	Yichang Shengwu Jianwenlu	Supernatural, Vampire	PG-13 - Teens 13 or older	351149.038671
11461	Noblesse: Awakening	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	431408.609005
16615	Noblesse	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	431408.609005

```
# Манхэттэнское расстояние дает приблизительно равные результаты поиска
rec3 = skr1.recommend_for_single_object(10, trinity_blood_matrix,
                                        cos_flag = False, manh_flag = True)
rec3
```

	Name	Gender	Rating	dist
15595	Yichang Shengwu Jianwenlu	Supernatural, Vampire	PG-13 - Teens 13 or older	430178.058232
11461	Noblesse: Awakening	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	573076.976390
16615	Noblesse	Action, Supernatural, Vampire, School	R - 17+ (violence & profanity)	573076.976390
14352	Sirius	Action, Historical, Supernatural, Vampire	R - 17+ (violence & profanity)	637941.562929
6104	Nyanpire The Animation	Comedy, Supernatural, Vampire	G - All Ages	661777.343822
5133	Hellsing: Psalm of the Darkness	Action, Supernatural, Vampire, Seinen	R - 17+ (violence & profanity)	694635.629757
	Kizumonogatari III:	Action, Mystery, Supernatural,	R - 17+ (violence	

Коллаборативная фильтрация. Метод user-based

```
df_user = pd.read_csv('/content/gdrive/My Drive/MMO/ratings.csv')
df_user.head (30)
```

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556
5	1	1968	4.0	1425942148
6	1	2762	4.5	1425941300
7	1	2918	5.0	1425941593
8	1	2959	4.0	1425941601
9	1	4226	4.0	1425942228
10	1	4878	5.0	1425941434
11	1	5577	5.0	1425941397
12	1	33794	4.0	1425942005
13	1	54503	3.5	1425941313
14	1	58559	4.0	1425942007
15	1	59315	5.0	1425941502
16	1	68358	5.0	1425941464
17	1	69844	5.0	1425942139
18	1	73017	5.0	1425942699
19	1	81834	5.0	1425942133

```
# Количество уникальных пользователей
len(df_user['userId'].unique())
```

    1726

23 1 96821 5.0 1425941382

```
# Количество уникальных аниме
len(df_user['movieId'].unique())
```

    10475

```
# Сформируем матрицу взаимодействий на основе рейтингов
def create_utility_matrix(data):
    itemField = 'movieId'
    userField = 'userId'
    valueField = 'rating'

    userList = data[userField].tolist()
    itemList = data[itemField].tolist()
    valueList = data[valueField].tolist()
```

```
    valueList = data[valueField].tolist()

    users = list(set(userList))
    items = list(set(itemList))

    users_index = {users[i]: i for i in range(len(users))}
    pd_dict = {item: [0.0 for i in range(len(users))] for item in items}

    for i in range(0,data.shape[0]):
        item = itemList[i]
        user = userList[i]
        value = valueList[i]
        pd_dict[item][users_index[user]] = value

    X = pd.DataFrame(pd_dict)
    X.index = users

    itemcols = list(X.columns)
    items_index = {itemcols[i]: i for i in range(len(itemcols))}

    return X, users_index, items_index
```

```
%%time
user_item_matrix, users_index, items_index = create_utility_matrix(df_user)
```

```
CPU times: user 4.7 s, sys: 645 ms, total: 5.35 s
Wall time: 4.92 s
```

```
user_item_matrix
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
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.C
2	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.C
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.C
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.C
5	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.C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1722	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	4.0	5.C
1723	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.C
1724	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.C
1725	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	3.0	0.C
1726	1.5	2.5	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	3.5	0.0	0.C

1726 rows × 10475 columns

```
# Выделение тестовой строки
user_item_matrix_test = user_item_matrix.loc[[1726]]
```

```
user_item_matrix__test = user_item_matrix.loc[[1726]]
user_item_matrix__test
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1726	1.5	2.5	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	3.5	0.0	0.0

1 rows × 10475 columns

```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.loc[:1725]
user_item_matrix__train
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
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
2	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
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
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
5	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1721	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
1722	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	4.0	5.0
1723	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
1724	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
1725	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	3.0	0.0

1725 rows × 10475 columns

Построение модели на основе SDV

```
%%time
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
```

```
CPU times: user 1min 35s, sys: 3.48 s, total: 1min 39s
Wall time: 51.6 s
```

```
# Матрица соотношения между пользователями и латентными факторами
U.shape
```

```
(10475, 10475)
```

```
# Матрица соотношения между объектами и латентными факторами
```

```
V.shape
```

    (1725, 1725)

```
S.shape
```

    (1725,)

```
Sigma = np.diag(S)
Sigma.shape
```

    (1725, 1725)

```
# Диагональная матрица сингулярных значений
Sigma
```

```
array([[6.44242259e+02, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 2.90612469e+02, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 2.30453178e+02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        2.21000120e-02, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 5.63516800e-14, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 1.12554711e-14]])
```

```
# Используем 3 первых сингулярных значения
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]
```

```
# Матрица соотношения между новым пользователем и латентными факторами
test_user = np.mat(user_item_matrix__test.values)
test_user.shape, test_user
```

    ((1, 10475), matrix([[1.5, 2.5, 0. , ..., 0. , 0. , 0.]]))

```
tmp = test_user * Ur * np.linalg.inv(Sr)
tmp
```

    matrix([[-0.03210987, -0.00074386, 0.00198715]])

```
test_user_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])
test_user_result
```

    array([-0.03210987, -0.00074386, 0.00198715])

```
# Вычисляем косинусную близость между текущим пользователем и остальными пользователями
cos_sim = cosine_similarity(Vr, test_user_result.reshape(1, -1))
cos_sim[:10]
```

```
array([[0.51386506],
       [0.27446399],
       [0.58918284],
       [0.77620347],
       [0.43780319],
       [0.68410241],
       [0.39757477],
       [0.82922485],
       [0.84720479],
       [0.10977591]])
```

```
# Преобразуем размерность массива
cos_sim_list = cos_sim.reshape(-1, cos_sim.shape[0])[0]
cos_sim_list[:10]
```

```
array([0.51386506, 0.27446399, 0.58918284, 0.77620347, 0.43780319,
       0.68410241, 0.39757477, 0.82922485, 0.84720479, 0.10977591])
```

```
# Находим наиболее близкого пользователя
recommended_user_id = np.argsort(-cos_sim_list)[0]
recommended_user_id
```

```
855
```

```
movieId_list = list(user_item_matrix.columns)
```

```
df_name=df1[['MAL_ID','Name']]
df_merge = pd.merge(df_user, df_name, left_on='movieId', right_on='MAL_ID', how='inner')
df_merge.drop(columns=['MAL_ID','timestamp'])
```

	userId	movieId	rating	Name

```
# Аниме, которые оценивал текущий пользователь:

i=1
for idx, item in enumerate(np.ndarray.flatten(np.array(test_user))):
    if item > 0:
        title = df_merge.at[idx, 'Name']
        id=df_merge.at[idx, 'userId']
        print('{} - {} - {}'.format(id, title, item))
        if i==50:
            break
        else:
            i+=1
```

```
1 - Chuuka Ichiban! - 1.5
11 - Chuuka Ichiban! - 2.5
33 - Chuuka Ichiban! - 3.0
68 - Chuuka Ichiban! - 3.5
82 - Chuuka Ichiban! - 2.0
142 - Chuuka Ichiban! - 3.0
157 - Chuuka Ichiban! - 2.0
217 - Chuuka Ichiban! - 3.0
224 - Chuuka Ichiban! - 3.0
308 - Chuuka Ichiban! - 2.0
457 - Chuuka Ichiban! - 2.5
638 - Chuuka Ichiban! - 2.0
816 - Chuuka Ichiban! - 4.0
848 - Chuuka Ichiban! - 3.0
881 - Chuuka Ichiban! - 0.5
1002 - Chuuka Ichiban! - 3.0
1012 - Chuuka Ichiban! - 2.5
1041 - Chuuka Ichiban! - 3.0
1055 - Chuuka Ichiban! - 3.0
1060 - Chuuka Ichiban! - 2.0
1113 - Chuuka Ichiban! - 3.0
1231 - Chuuka Ichiban! - 3.0
1409 - Chuuka Ichiban! - 3.0
1438 - Chuuka Ichiban! - 2.0
1445 - Chuuka Ichiban! - 2.0
1627 - Chuuka Ichiban! - 3.0
1658 - Chuuka Ichiban! - 3.0
37 - Gunparade Orchestra - 3.0
146 - Gunparade Orchestra - 3.0
340 - Gunparade Orchestra - 3.0
376 - Gunparade Orchestra - 3.0
482 - Gunparade Orchestra - 2.0
483 - Gunparade Orchestra - 3.0
502 - Gunparade Orchestra - 4.5
506 - Gunparade Orchestra - 3.0
507 - Gunparade Orchestra - 3.0
510 - Gunparade Orchestra - 3.0
512 - Gunparade Orchestra - 2.5
557 - Gunparade Orchestra - 3.0
734 - Gunparade Orchestra - 4.0
989 - Gunparade Orchestra - 2.0
1029 - Gunparade Orchestra - 3.0
1170 - Gunparade Orchestra - 2.5
1215 - Gunparade Orchestra - 3.0
```

```
1243 - Gunparade Orchestra - 2.0
1317 - Gunparade Orchestra - 1.0
1426 - Gunparade Orchestra - 1.0
1442 - Gunparade Orchestra - 4.0
1460 - Gunparade Orchestra - 4.0
1465 - Gunparade Orchestra - 1.0
```

```
# Фильмы, которые оценивал наиболее схожий пользователь:
i=1
recommended_user_item_matrix = user_item_matrix.loc[[recommended_user_id+1]]
for idx, item in enumerate(np.ndarray.flatten(np.array(recommended_user_item_matrix))):
    if item > 0:
        title = df_merge.at[idx, 'Name']
        id=df_merge.at[idx, 'userId']
        print('{} - {} - {}'.format(id, title, item))
        if i==30:
            break
        else:
            i+=1
```

```
1 - Chuuka Ichiban! - 2.5
11 - Chuuka Ichiban! - 3.5
82 - Chuuka Ichiban! - 2.0
157 - Chuuka Ichiban! - 4.0
224 - Chuuka Ichiban! - 4.0
752 - Chuuka Ichiban! - 2.0
918 - Chuuka Ichiban! - 2.5
937 - Chuuka Ichiban! - 5.0
955 - Chuuka Ichiban! - 3.0
1080 - Chuuka Ichiban! - 3.0
1145 - Chuuka Ichiban! - 5.0
1299 - Chuuka Ichiban! - 4.5
1316 - Chuuka Ichiban! - 3.5
1338 - Chuuka Ichiban! - 2.5
1408 - Chuuka Ichiban! - 3.5
1409 - Chuuka Ichiban! - 4.0
1438 - Chuuka Ichiban! - 5.0
1445 - Chuuka Ichiban! - 1.5
1465 - Chuuka Ichiban! - 1.0
1490 - Chuuka Ichiban! - 2.5
1718 - Chuuka Ichiban! - 1.0
778 - Kimi ga Nozomu Eien - 3.0
1055 - Kimi ga Nozomu Eien - 0.5
37 - Gunparade Orchestra - 4.0
146 - Gunparade Orchestra - 2.5
201 - Gunparade Orchestra - 4.0
249 - Gunparade Orchestra - 3.5
376 - Gunparade Orchestra - 5.0
480 - Gunparade Orchestra - 3.5
483 - Gunparade Orchestra - 4.0
```