In [ ]:
```
# !pip uninstall -y numpy
# !pip install numpy==1.26.0
# !pip install pandas surprise scikit-learn seaborn matplotlib datetime
```

In [ ]:
```
!pip install --force-reinstall numpy==1.24.3 scikit-surprise
```

```
Collecting numpy==1.24.3
  Using cached numpy-1.24.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl.metadata (5.6 kB)
Collecting scikit-surprise
  Using cached scikit_surprise-1.1.4-cp311-cp311-linux_x86_64.whl
Collecting joblib>=1.2.0 (from scikit-surprise)
  Using cached joblib-1.5.0-py3-none-any.whl.metadata (5.6 kB)
Collecting scipy>=1.6.0 (from scikit-surprise)
  Using cached scipy-1.15.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl.metadata (61 kB)
Using cached numpy-1.24.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (17.3 MB)
Using cached joblib-1.5.0-py3-none-any.whl (307 kB)
Using cached scipy-1.15.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (37.7 MB)
Installing collected packages: numpy, joblib, scipy, scikit-surprise
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.3
    Uninstalling numpy-1.24.3:
      Successfully uninstalled numpy-1.24.3
  Attempting uninstall: joblib
    Found existing installation: joblib 1.5.0
    Uninstalling joblib-1.5.0:
      Successfully uninstalled joblib-1.5.0
  Attempting uninstall: scipy
    Found existing installation: scipy 1.15.3
    Uninstalling scipy-1.15.3:
      Successfully uninstalled scipy-1.15.3
  Attempting uninstall: scikit-surprise
    Found existing installation: scikit-surprise 1.1.4
    Uninstalling scikit-surprise-1.1.4:
      Successfully uninstalled scikit-surprise-1.1.4
ERROR: pip's dependency resolver does not currently take into account all the p
ackages that are installed. This behaviour is the source of the following depen
dency conflicts.
jaxlib 0.5.1 requires numpy>=1.25, but you have numpy 1.24.3 which is incompati
ble.
jax 0.5.2 requires numpy>=1.25, but you have numpy 1.24.3 which is incompatibl
e.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.24.3 whic
h is incompatible.
blosc2 3.3.2 requires numpy>=1.26, but you have numpy 1.24.3 which is incompati
ble.
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.24.3 which is inco
mpatible.
pymc 5.22.0 requires numpy>=1.25.0, but you have numpy 1.24.3 which is incompat
ible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.24.3 which is in
compatible.
albumentations 2.0.6 requires numpy>=1.24.4, but you have numpy 1.24.3 which is
incompatible.
albucore 0.0.24 requires numpy>=1.24.4, but you have numpy 1.24.3 which is inco
mpatible.
Successfully installed joblib-1.5.0 numpy-1.24.3 scikit-surprise-1.1.4 scip
```

```
y-1.15.3
```

In [ ]:
```
# !pip uninstall -y pandas
# !pip install pandas==2.2.2
```

In [ ]:
```
# !pip uninstall surprise -y
# !pip install --no-binary :all: scikit-surprise
```

In [ ]:
```
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



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, classific
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegre
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squa
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, m

from surprise import SVD
from surprise import Dataset
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")
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

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

In [ ]:

```
df = pd.read_csv('/content/gdrive/My Drive/MMO/anime.csv')
# df1=df.drop(columns=['English name','Japanese name','Type', 'Aired', 'Premie
df=df.drop(columns=['MAL_ID','English name','Japanese name','Type', 'Aired', '

#df.drop_duplicates(subset=['Name'])

df.head (10)
```

Out[ ]:

	Name	Score	Genders	Episodes	Studios	Source	Duration	Ra
0	Cowboy Bebop	8.78	Action, Adventure, Comedy, Drama, Sci-Fi, Space	26	Sunrise	Original	24 min. per ep.	R - (vio profa
1	Cowboy Bebop: Tengoku no Tobira	8.39	Action, Drama, Mystery, Sci-Fi, Space	1	Bones	Original	1 hr. 55 min.	R - (vio profa
2	Trigun	8.24	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	26	Madhouse	Manga	24 min. per ep.	PG Teen or
3	Witch Hunter Robin	7.27	Action, Mystery, Police, Supernatural, Drama, ...	26	Sunrise	Original	25 min. per ep.	PG Teen or
4	Bouken Ou Beet	6.98	Adventure, Fantasy, Shounen, Supernatural	52	Toei Animation	Manga	23 min. per ep.	Chi
5	Eyeshield 21	7.95	Action, Sports, Comedy, Shounen	145	Gallop	Manga	23 min. per ep.	PG Teen or
6	Hachimitsu to Clover	8.06	Comedy, Drama, Josei, Romance, Slice of Life	24	J.C.Staff	Manga	23 min. per ep.	PG Teen or
7	Hungry Heart: Wild Striker	7.59	Slice of Life, Comedy, Sports, Shounen	52	Nippon Animation	Manga	23 min. per ep.	PG Teen or
8	Initial D Fourth Stage	8.15	Action, Cars, Sports, Drama, Seinen	24	A.C.G.T.	Manga	27 min. per ep.	PG Teen or
9	Monster	8.76	Drama, Horror, Mystery, Police, Psychological,...	74	Madhouse	Manga	24 min. per ep.	R+ - N

10 rows × 27 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(17562, 27)
```

In [ ]:
```
name = df['Name'].values
```

```
name[0:30]
```

Out[ ]:
```
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 Chinkonk
a',
       'Akira', '.hack//Sign'], dtype=object)
```

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

Out[ ]:
```
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, Supernatural',
       '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)
```

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

Out[ ]:
```
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)
```

In [ ]:
```
%%time
tfidfv = TfidfVectorizer()
genders_matrix = tfidfv.fit_transform(gender)
genders_matrix
```

```
CPU times: user 126 ms, sys: 4.24 ms, total: 130 ms
Wall time: 210 ms
```

Out[ ]:
```
<Compressed Sparse Row sparse matrix of dtype 'float64'
	with 57900 stored elements and shape (17562, 48)>
```

Фильтрация на основе содержания по жанрам

In [ ]:
```
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 объек
        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
```

In [ ]:
```
trinity_blood_ind =17
name[trinity_blood_ind]
```

Out[ ]: 'Trinity Blood'

In [ ]:
```
trinity_blood_matrix = genders_matrix[trinity_blood_ind]
trinity_blood_matrix
```

Out[ ]:
```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 3 stored elements and shape (1, 48)>
```

In [ ]:
```
skr1 = SimpleKNNRecommender(genders_matrix, name, gender, rating)
```

Выведем 10 аниме похожих на "кровь триединства"

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

Out[ ]:

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

In [ ]:

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

Out[ ]:

	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
5133	Hellsing: Psalm of the Darkness	Action, Supernatural, Vampire, Seinen	R - 17+ (violence & profanity)	873084.219101
7168	JoJo no Kimyou na Bouken (TV)	Action, Adventure, Supernatural, Vampire, Shounen	R - 17+ (violence & profanity)	866381.795488
11103	Kizumonogatari III: Reiketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	866273.999806
11102	Kizumonogatari II: Nekketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	866273.999806

In [ ]:

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

Out[ ]:

	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
1644	Master Mosquiton '99	Action, Adventure, Comedy, Supernatural, Vampire	PG-13 - Teens 13 or older	442110.174192
6104	Nyanpire The Animation	Comedy, Supernatural, Vampire	G - All Ages	452136.772271
14352	Sirius	Action, Historical, Supernatural, Vampire	R - 17+ (violence & profanity)	470536.077177
5133	Hellsing: Psalm of the Darkness	Action, Supernatural, Vampire, Seinen	R - 17+ (violence & profanity)	503816.992368
7168	JoJo no Kimyou na Bouken (TV)	Action, Adventure, Supernatural, Vampire, Shounen	R - 17+ (violence & profanity)	516949.135819
11103	Kizumonogatari III: Reiketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	517157.616582
11102	Kizumonogatari II: Nekketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	517157.616582

In [ ]:

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

Out[ ]:

	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
11103	Kizumonogatari III: Reiketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	717746.413634
11102	Kizumonogatari II: Nekketsu-hen	Action, Mystery, Supernatural, Vampire	R - 17+ (violence & profanity)	717746.413634
1644	Master Mosquiton '99	Action, Adventure, Comedy, Supernatural, Vampire	PG-13 - Teens 13 or older	762750.970781
11524	Answer	Music, Supernatural, Vampire	G - All Ages	791202.630915

Коллаборативная фильтрация. Метод user-based

In [ ]:

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

Out[ ]:

	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
20	1	91500	2.5	1425942647
21	1	91542	5.0	1425942618
22	1	92439	5.0	1425941424
23	1	96821	5.0	1425941382
24	1	98809	0.5	1425942640
25	1	99114	4.0	1425941667
26	1	112552	5.0	1425941336
27	2	5	3.0	867039249
28	2	25	3.0	867039168
29	2	32	2.0	867039166

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

Out[ ]: 1726

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

Out[ ]: 10475

In [ ]:
```
# 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()

    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
```

In [ ]:
```
%%time
user_item_matrix, users_index, items_index = create_utility_matrix(df_user)
```

```
CPU times: user 3.41 s, sys: 335 ms, total: 3.75 s
Wall time: 3.8 s
```

In [ ]:
```
user_item_matrix
```

Out[ ]:

	1	2	3	4	5	6	7	8	9	10	...	131015	98248	32721	1
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	
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	
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	
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	
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	
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	
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	
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	
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	

1726 rows × 10475 columns

In [ ]:

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

Out[ ]:

	1	2	3	4	5	6	7	8	9	10	...	131015	98248	32721	1
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	

1 rows × 10475 columns

In [ ]:

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

Out[ ]:

	1	2	3	4	5	6	7	8	9	10	...	131015	98248	32721	1
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	
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	
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	
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	
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	
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	
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	
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	
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	

1725 rows × 10475 columns

Построение модели на основе SDV

In [ ]:

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

```
CPU times: user 1min 31s, sys: 6.77 s, total: 1min 38s
Wall time: 1min
```

In [ ]:

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

Out[ ]: (10475, 10475)

In [ ]:

```
# Матрица соотношения между объектами и латентными факторами
V.shape
```

Out[ ]: (1725, 1725)

In [ ]:

```
S.shape
```

Out[ ]: (1725,)

In [ ]:

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

Out[ ]: (1725, 1725)

In [ ]:
```
# Диагональная матрица сингулярных значений
Sigma
```

Out[ ]:
```
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, 5.61424353e-14, 0.00000000e+00],
        [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
         0.00000000e+00, 0.00000000e+00, 1.12554711e-14]])
```

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

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

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

In [ ]:
```
tmp = test_user * Ur * np.linalg.inv(Sr)
tmp
```

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

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

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

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

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

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

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

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

Out[ ]:
```
855
```

In [ ]:
```
movieId_list = list(user_item_matrix.columns)

df1 = pd.read_csv('/content/gdrive/My Drive/MMO/anime.csv')

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

Out[ ]:

	userId	movieId	rating	Name
0	1	110	1.0	Chuuka Ichiban!
1	1	147	4.5	Kimi ga Nozomu Eien
2	1	858	5.0	Gunparade Orchestra
3	1	1221	5.0	Demashita! Powerpuff Girls Z
4	1	1246	5.0	Yuugo: Koushounin
...	...	...	...	...
105874	1726	38798	0.5	Tsuma ga Kirei ni Natta Wake
105875	1726	40815	3.0	Honzuki no Gekokujou: Shisho ni Naru Tame ni w...
105876	1726	40819	3.0	Kyonyuu Princess Saimin
105877	1726	42725	2.0	Sen no Hana Sen no Sora
105878	1726	45447	2.0	Moshi Juexing Zhi Suyuan

105879 rows × 4 columns

In [ ]:

```
# 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
1 - Kimi ga Nozomu Eien - 2.5
1 - Igano Kabamaru - 3.0
2 - Happy☆Lesson (TV) - 3.5
2 - Tenshi ni Narumon! - 2.0
3 - Black Magic M-66 - 3.0
4 - Tsuki wa Higashi ni Hi wa Nishi ni: Operation Sanctuary - 2.0
4 - Princess Rouge - 3.0
4 - Herlock Saga: Nibelung no Yubiwa - 3.0
4 - Urayasu Tekkin Kazoku - 2.0
6 - The☆Doraemons: Strange, Sweets, Strange? - 2.5
8 - Afro Samurai - 2.0
8 - Renketsu Houshiki - 4.0
8 - Umineko no Naku Koro ni - 3.0
9 - Starship Operators - 0.5
9 - Sekai Meisaku Douwa: Mori wa Ikiteiru - 3.0
9 - Kagaku Kyuujo-tai TechnoVoyager - 2.5
9 - Hinadori no Saezuri - 3.0
9 - Fluximation - 3.0
9 - Mahjong Hishouden: Naki no Ryuu - 2.0
10 - PopoloCrois - 3.0
11 - To Heart Omake - 3.0
12 - Seikai no Monshou - 3.0
12 - Marmalade Boy - 2.0
12 - Fate/stay night - 2.0
12 - Kashou no Tsuki: Aki Kyougen - 3.0
12 - Pokemon Movie 05: Mizu no Miyako no Mamorigami Latias to Latios - 3.0
12 - UFO Princess Valkyrie 4: Toki to Yume to Ginga no Utage - 3.0
12 - Saiunkoku Monogatari 2nd Season - 3.0
12 - Mitsubachi Maya no Bouken - 3.0
12 - Makasete Iruka! - 3.0
15 - Binzume Yousei - 2.0
15 - Magical Canan - 3.0
15 - Koutetsu Tenshi Kurumi 2 - 4.5
15 - Ginga Nagareboshi Gin - 3.0
15 - Yuugen Kaisha - 3.0
15 - Sakura Taisen: Katsudou Shashin - 3.0
15 - Kizuna - 2.5
15 - Masuda Kousuke Gekijou Gag Manga Biyori - 3.0
15 - Idol Project - 4.0
15 - Tono to Issho: Ippunkan Gekijou - 2.0
15 - AEIOUdan: Douro no Tadashii Watari Kata - 3.0
16 - Pokemon: Senritsu no Mirage Pokemon - 2.5
16 - Saiunkoku Monogatari Recaps - 3.0
16 - Mitsubachi Maya no Bouken - 2.0
16 - Zenryoku Usagi - 1.0
17 - Mugen no Ryvius - 1.0
17 - Macross: Do You Remember Love? - 4.0
17 - One Piece Film: Strong World - 4.0
17 - Kojin Jugyou - 1.0
```

In [ ]:

```
# 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_m
    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
1 - Kimi ga Nozomu Eien - 3.5
2 - Tenshi ni Narumon! - 2.0
4 - Tsuki wa Higashi ni Hi wa Nishi ni: Operation Sanctuary - 4.0
4 - Herlock Saga: Nibelung no Yubiwa - 4.0
8 - Doraemon Movie 12: Nobita no Dorabian Nights - 2.0
9 - Variable Geo - 2.5
9 - Future GPX Cyber Formula - 5.0
9 - Baldr Force Exe Resolution - 3.0
10 - Futatsu no Spica - 3.0
11 - Mushishi - 5.0
12 - Last Exile - 4.5
12 - Mahou Sensei Negima! - 3.5
12 - Pita Ten - 2.5
12 - Mahoutsukai ni Taisetsu na Koto - 3.5
12 - Seikai no Monshou - 4.0
12 - Marmalade Boy - 5.0
12 - Fate/stay night - 1.5
12 - Shoujo Kakumei Utena: Adolescence Mokushiroku - 1.0
12 - Armitage III: Dual-Matrix - 2.5
12 - Wata no Kuni Hoshi - 1.0
12 - Injuu Gakuen La☆Blue Girl: Fukkatsu-hen - 3.0
12 - Hokuto no Ken Movie - 0.5
12 - UFO Princess Valkyrie 4: Toki to Yume to Ginga no Utage - 4.0
12 - Saiunkoku Monogatari 2nd Season - 2.5
12 - Zettai Muteki Raijin-Oh (1992) - 4.0
12 - Karasu Tengu Kabuto - 3.5
12 - Makasete Iruka! - 5.0
15 - Dragon Drive - 3.5
15 - Magical Canan - 4.0
```

In [ ]: