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:

Чичикин Тимофей Дмитриевич
Группа ИУ5-25М

____

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 uninstall -y numpy
# !pip install numpy==1.26.0
# !pip install pandas surprise scikit-learn seaborn matplotlib
datetime

!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_x86_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_x86_64.whl.metadata (61 kB)
Using cached numpy-1.24.3-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.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_64.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 packages that are installed. This behaviour is the source of
the following dependency conflicts.
jaxlib 0.5.1 requires numpy>=1.25, but you have numpy 1.24.3 which is
incompatible.
jax 0.5.2 requires numpy>=1.25, but you have numpy 1.24.3 which is
incompatible.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy
1.24.3 which is incompatible.
blosc2 3.3.2 requires numpy>=1.26, but you have numpy 1.24.3 which is
incompatible.
```

```
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.24.3
which is incompatible.
pymc 5.22.0 requires numpy>=1.25.0, but you have numpy 1.24.3 which is
incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.24.3
which is incompatible.
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 incompatible.
Successfully installed joblib-1.5.0 numpy-1.24.3 scikit-surprise-1.1.4
scipy-1.15.3
```

{"id":"36eb58434fe24e908e7996bbb644813f","pip_warning":{"packages":
["numpy"]}}

```
# !pip uninstall -y pandas
# !pip install pandas==2.2.2

# !pip uninstall surprise -y
# !pip install --no-binary :all: scikit-surprise

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,
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_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity,
euclidean_distances, manhattan_distances

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")

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',     'Producers',     'Licensors'])
df=df.drop(columns=['MAL_ID','English name','Japanese name','Type',
'Aired', 'Premiered', 'Producers',     'Licensors'])

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

df.head (10)
```

{"type":"dataframe"}

```
df.shape
```

```
(17562, 27)
```

```
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, 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)
```

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

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

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]

{"type":"string"}
```

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

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

```
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']
```

{"summary":"{\n \"name\": \"df_new\",\n \"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Trinity Blood\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Genders\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Action, Supernatural, Vampire\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Rating\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"R - 17+ (violence & profanity)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

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

{"summary":"{\n \"name\": \"rec1\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 10,\n   \"samples\": [\n   \"Kizumonogatari III: Reiketsu-hen\",\n   \"Noblesse\",\n   \"Sirius\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 8,\n   \"samples\": [\n   \"Action, Supernatural, Vampire, School\",\n   \"Action, Supernatural, Vampire, Seinen\",\n   \"Supernatural, Vampire\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Rating\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"PG-13 - Teens 13 or older\",\n   \"R - 17+ (violence & profanity)\",\n   \"G - All Ages\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

n   },\n   {\n   \"column\": \"dist\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 23745.324554057453,\n   \"min\": 866273.9998055177,\n   \"max\": 938347.1763200712,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   906943.3060380513,\n   873084.2191007645,\n   938347.1763200712\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"rec1"}

```
# При поиске с помощью Евклидова расстояния получаем почти такой же
результат. Разница во 2 и 3 строке, но расстояния у них одинаковые
rec2 = skr1.recommend_for_single_object(10, trinity_blood_matrix,
cos_flag = False)
rec2
```

{"summary":"{\n \"name\": \"rec2\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 10,\n   \"samples\": [\n   \"Kizumonogatari III: Reiketsu-hen\",\n   \"Noblesse: Awakening\",\n   \"Sirius\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 8,\n   \"samples\": [\n   \"Action, Supernatural, Vampire, School\",\n   \"Action, Supernatural, Vampire, Seinen\",\n   \"Supernatural, Vampire\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Rating\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"PG-13 - Teens 13 or older\",\n   \"R - 17+ (violence & profanity)\",\n   \"G - All Ages\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"dist\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 53290.628867703155,\n   \"min\": 351149.038671413,\n   \"max\": 517157.61658218317,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   431408.60900531115,\n   503816.9923677359,\n   351149.038671413\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"rec2"}

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

{"summary":"{\n \"name\": \"rec3\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n

\"dtype\": \"string\",\n   \"num_unique_values\": 10,\n
\"samples\": [\n   \"Master Mosquiton '99\",\n
\"Noblesse: Awakening\",\n   \"Hellsing: Psalm of the
Darkness\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Gender\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 8,\n   \"samples\": [\n   \"Action,
Supernatural, Vampire, School\",\n   \"Action, Mystery,
Supernatural, Vampire\",\n   \"Supernatural, Vampire\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Rating\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"PG-13 - Teens 13 or older\",\n
\"R - 17+ (violence & profanity)\",\n   \"G - All Ages\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"dist\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
107445.57299444747,\n   \"min\": 430178.05823249306,\n
\"max\": 791202.6309150326,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   573076.9763895547,\n
717746.4136338527,\n   430178.05823249306\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"rec3"}

Коллаборативная фильтрация. Метод user-based

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

{"summary":"{\n \"name\": \"df_user\",\n \"rows\": 30,\n
\"fields\": [\n   {\n   \"column\": \"userId\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 1,\n   \"max\": 2,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   2,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"movieId\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 41850,\n   \"min\": 5,\n   \"max\":
112552,\n   \"num_unique_values\": 30,\n   \"samples\": [\n
5,\n   59315\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"rating\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1.2434887887778048,\n   \"min\": 0.5,\n   \"max\":
5.0,\n   \"num_unique_values\": 9,\n   \"samples\": [\n
3.0,\n   4.5\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":

\"timestamp\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 170537164,\n   \"min\": 867039166,\
n   \"max\": 1425942699,\n   \"num_unique_values\": 30,\n
\"samples\": [\n   867039249,\n   1425941502\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

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

```
1726
```

```
# 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()

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

```
user_item_matrix
```

{"type":"dataframe","variable_name":"user_item_matrix"}

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

{"type":"dataframe","variable_name":"user_item_matrix__test"}

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

{"type":"dataframe","variable_name":"user_item_matrix__train"}

Построение модели на основе SDV

```
%%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
```

```
# 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, 5.61424353e-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]])
```

```python
# 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])
```

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

```
855
```

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

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

```python
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'])
```

{"type":"dataframe"}

```python
# 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
```

```python
# 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
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
```