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

Лабораторная работа №4

«Создание рекомендательной модели»

по дисциплине

«Методы машинного обучения»

ИСПОЛНИТЕЛЬ:

Крюков Г.М.
Группа ИУ5-21М

_____

"__"_____2022 г.

Москва, 2022

Цель: Изучение разработки рекомендательных моделей.

Задание:

1. Выбрать произвольный набор данных (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).

Ход выполнения:

Датасет: Набор отзывов пользователей на пищевые товары с сервиса Amazon
Recommendation based on Amazon food - https://www.kaggle.com/code/saurav9786/recommendation-based-on-amazon-food-review/data

Поля:
Id: unique identifier
ProductId: Unique identifier for the product
UserId: Unqiue identifier for the user
ProfileName: Profile name of the user
HelpfulnessNumerator: Number of users who found the review helpful
HelpfulnessDenominator: Number of users who indicated whether they found the review helpful or not
Score: Rating between 1 and 5
Time: Timestamp for the review
Summary: Brief summary of the review
Text: Text of the review

Текст программы:

```
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_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 collections import defaultdict
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib_venn import venn2
%matplotlib inline
sns.set(style="ticks")
```

▾ Чтение и обработка данных

```
[ ] data = pd.read_csv('/content/Reviews.csv')
    data.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
[ ] data.shape
```

```
(568454, 10)
```

```
[ ] review_data = data[data['Text'].notnull()]
    review_data.shape
```

```
(568454, 10)
```

```
[ ] # проверим есть ли пропущенные значения
    data.isnull().sum()
```

```
Id                         0
ProductId                  0
UserId                     0
ProfileName               16
HelpfulnessNumerator       0
HelpfulnessDenominator     0
Score                      0
Time                       0
Summary                   27
Text                       0
dtype: int64
```

```
[ ]  # Удаление строк, содержащих пустые значения
     data_full = data.dropna(axis=0, how='any')
     (data.shape, data_full.shape)
```

```
     ((568454, 10), (568411, 10))
```

```
[ ]  data_full.keys()
```

```
     Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
            'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
           dtype='object')
```

```
[ ]  product_ids = data_full['Id'].values
     product_ids
```

```
     array([     1,      2,      3, ..., 568452, 568453, 568454])
```

```
[ ]  products = data_full['ProductId'].values
     products[0:5]
```

```
     array(['B001E4KFG0', 'B00813GRG4', 'B000LQOCH0', 'B000UA0QIQ',
            'B006K2ZZ7K'], dtype=object)
```

```
[ ]  reviews = data_full['Text'].values
     reviews[:5]
```

```
     array(['I have bought several of the Vitality canned dog food products and have found them al
            'Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small siz
            'This is a confection that has been around a few centuries.  It is a light, pillowy ci
            'If you are looking for the secret ingredient in Robitussin I believe I have found it.
            'Great taffy at a great price.  There was a wide assortment of yummy taffy.  Delivery
           dtype=object)
```

```
[ ]  %%time
     tfidf = TfidfVectorizer()
     description_matrix = tfidf.fit_transform(reviews)
     description_matrix
```

```
     CPU times: user 35.9 s, sys: 659 ms, total: 36.6 s
     Wall time: 39.4 s
```

```
[ ]  description_matrix
```

```
     <568411x120250 sparse matrix of type '<class 'numpy.float64'>'
             with 30647740 stored elements in Compressed Sparse Row format>
```

Фильтрация на основе содержания. Метод k-ближайших соседей

```
[ ]  class SimplerKnnRecomender:
       def __init__(self, X_matrix, X_ids, X_title, X_overview):
             """
             Входные параметры:
             X_matrix - обучающая выборка (матрица объект-признак)
             X_ids - массив идентификаторов объектов
             X_title - массив названий объектов
             X_overview - массив описаний объектов
             """
             #Сохраняем параметры в переменных объекта
             self._X_matrix = X_matrix
             self.df = pd.DataFrame(
                 {'id': pd.Series(X_ids, dtype='int'),
                 'title': pd.Series(X_title, dtype='str'),
                 'overview': pd.Series(X_overview, 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
```

```
[ ]  test_id = 8389
     zee_user = data_full['UserId'].values
     zee_user = zee_user[test_id]
     #print(products[test_id])
     #print(reviews[test_id])
```

```
[ ]  test_matrix = description_matrix[test_id]
     test_matrix
```

```
<1x120250 sparse matrix of type '<class 'numpy.float64'>'
        with 34 stored elements in Compressed Sparse Row format>
```

```
[ ]  skr1 = SimplerKnnRecomender(description_matrix, product_ids, products, reviews)
```

```
[ ]  # 15 товаров, наиболее похожих на B000UA0QIQ
     # в порядке убывания схожести на основе косинусного сходства
     rec1 = skr1.recommend_for_single_object(15, test_matrix)
     rec1
```

	id	title	overview	dist
292703	292722	B005MSH1XY	i love soda and love root beer i have been exc...	369636.521643
158580	158593	B000A6DKKG	Hi, I am a HUGE Root Beer Fanatic...	362035.510324
101270	101278	B000IUPND6	I was really excited to find this product but ...	355553.961402
321848	321872	B001KUSLGY	This soda is OK. It has a weird aftertaste. Th...	345784.351133
372705	372734	B002U5A86E	I recommend A&W root beer for those fans who a...	344396.570288
47773	47778	B003SBTY2S	To me this soda did taste like a mild root bee...	343412.024333
364891	364920	B002AK2O4I	I had a sampler pack of this when I got my sod...	335151.815628
487392	487430	B002G0CA6O	This is probably one of the better diet sodas ...	334287.666579
487399	487437	B002G0CA6O	I've only tried 3 flavors of this soda so far:...	331811.598182
416362	416396	B003SBU2VA	This is the best natural soda I have found and...	329239.760597
353400	353426	B007J6KLAW	I recently had the opportunity to try out a <a...	327895.091678
479895	479931	B0001BVH9G	The root beer has a very good taste like root ...	325474.351109
194804	194820	B000OTBRKY	My husband is a HUGE Root beer fan. So, I got ...	323155.701530
149585	149597	B000E8WIAS	I just got this in last night and tried it tod...	320611.864630
101277	101285	B000IUPND6	Would love to tell you what it actually tasted...	320556.794841

```
# При поиске с помощью Евклидова расстояния получаем иной результат
rec2 = skr1.recommend_for_single_object(15, test_matrix, cos_flag = False)
rec2
```

	id	title	overview	dist
544827	544870	B002LMXFCU	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
378615	378644	B00126P0HE	T H I S C H O C O L A T E I S A D D I C T I...	1.000000e+06
487826	487864	B002LMA8FC	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
299584	299606	B002LMQRA2	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
388799	388832	B001G7QG5O	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
324226	324250	B002LN566C	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
187986	188002	B002LMQ6OO	v e r y g o o d v e r y g o o d v e r y g o o ...	1.000000e+06
292703	292722	B005MSH1XY	i love soda and love root beer i have been exc...	1.122821e+06
158580	158593	B000A6DKKG	Hi, I am a HUGE Root Beer Fanatic...	1.129570e+06
101270	101278	B000IUPND6	I was really excited to find this product but ...	1.135294e+06
321848	321872	B001KUSLGY	This soda is OK. It has a weird aftertaste. Th...	1.143867e+06
372705	372734	B002U5A86E	I recommend A&W root beer for those fans who a...	1.145079e+06
47773	47778	B003SBTY2S	To me this soda did taste like a mild root bee...	1.145939e+06
364891	364920	B002AK2O4I	I had a sampler pack of this when I got my sod...	1.153125e+06
487392	487430	B002G0CA6O	This is probably one of the better diet sodas ...	1.153874e+06

```
# Манхэттэнское расстояние дает несколько иные результаты поиска
rec3 = skr1.recommend_for_single_object(15, test_matrix,
                                        cos_flag = False, manh_flag = True)
rec3
```

	id	title	overview	dist
299584	299606	B002LMQRA2	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
187986	188002	B002LMQ6OO	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
544827	544870	B002LMXFCU	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
378615	378644	B00126P0HE	T H I S C H O C O L A T E I S A D D I C T I...	5.027534e+06
388799	388832	B001G7QG5O	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
324226	324250	B002LN566C	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
487826	487864	B002LMA8FC	v e r y g o o d v e r y g o o d v e r y g o o ...	5.027534e+06
59769	59775	B001NZW2V6	These are good but are not sweet! Good, Good, ...	5.919712e+06
172767	172782	B00152K9T4	yum yum yum yum yum yum yum yum yum yum yu...	6.090244e+06
159475	159488	B004OQ7A4U	Again this is good stuff but don't buy it here...	6.155053e+06
434639	434674	B002KB41QG	One pound lasted only a couple of days, it is ...	6.193635e+06
349473	349498	B002QX38I0	I bought it as a gift and had no complaints. w...	6.289485e+06
393535	393569	B001ET5XVC	The Garlic paste is very good. I made some gar...	6.299548e+06
484118	484154	B0000TU9TS	This is a great mustard if you like the true m...	6.326417e+06
477594	477630	B0001M0YK8	I was happy to find this in bulk ... it's our ...	6.354051e+06

Коллаборативная фильтрация. Метод на основе сингулярного разложения

```
[ ] data_full.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
[ ] data_short = data_full[30000:55000]
    data_short.shape
```

```
(25000, 10)
```

```
[ ] # Количество уникальных пользователей, оставивших отзывы
    len(data_short['UserId'].unique())
```

```
21584
```

```
# Сформируем матрицу взаимодействий на основе рейтингов
# Используется идея из статьи - https://towardsdatascience.com/beginners-guide-to-creating-an-svd-recommender-system-1fd7326d1f65
def create_utility_matrix(data):
    itemField = 'ProductId'
    userField = 'UserId'
    valueField = 'Score'

    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(data_short)
```

```
CPU times: user 10.5 s, sys: 787 ms, total: 11.3 s
Wall time: 11.2 s
```

```
user_item_matrix
```

	B001KNI2WE	B000MT8AT2	B002Y2QS8U	B003AOCR22	B008N5VC1A	B001FUYPE6	B0001W2W4O	B0000TWLJE
A1XNT9Y7G5J8DP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A1V7Y9VIRWUD5Y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AWMJSP9UNMNRL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A21I6JIITGWD7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3SFW7DKA83D1O	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
A3HZJNP1OQ1JRY	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3I96WV0TK2R1P	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A2SVJWQVC7ZDMT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A14ELYDYX7LOFQ	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3PCUG76EPS57C	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

21584 rows × 3254 columns

```
# Выделение тестовой строки
user_item_matrix__test = user_item_matrix.loc[['A3UCO959VA9MV']]
user_item_matrix__test
```

	B001KNI2WE	B000MT8AT2	B002Y2QS8U	B003AOCR22	B008N5VC1A
A3UCO959VA9MV	0.0	0.0	0.0	0.0	0.0

1 rows × 3254 columns

```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.drop(['A3UCO959VA9MV'], axis=0, inplace=False)
user_item_matrix__train
```

	B001KNI2WE	B000MT8AT2	B002Y2QS8U	B003AOCR22	B008N5VC1A	B001FUYPE6	B0001W2W4O
A1XNT9Y7G5J8DP	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A1V7Y9VIRWUD5Y	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AWMJSP9UNMNRL	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A21I6JIITGWD7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3SFW7DKA83D1O	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
A3HZJNP1OQ1JRY	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3I96WV0TK2R1P	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A2SVJWQVC7ZDMT	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A14ELYDYX7LOFQ	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A3PCUG76EPS57C	0.0	0.0	0.0	0.0	0.0	0.0	0.0

21583 rows × 3254 columns

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

```
CPU times: user 11min 42s, sys: 27.7 s, total: 12min 10s
Wall time: 6min 21s
```

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

```
(3254, 3254)
```

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

```
(21583, 21583)
```

```
[ ] S.shape
```

```
(3254,)
```

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

```
(3254, 3254)
```

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

```
array([[1.04735170e+02, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 9.19262880e+01, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 8.38198611e+01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        1.28484753e-16, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 1.66792097e-17, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 1.08862805e-29]])
```

```
[ ] # 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, 3254), matrix([[0., 0., 0., ..., 0., 0., 0.]]))
```

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

```
matrix([[ 2.33195207e-06, -2.87461502e-07,  3.12486145e-06]])
```

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

```
array([ 2.33195207e-06, -2.87461502e-07,  3.12486145e-06])
```

```
# Вычисляем косинусную близость между текущим пользователем
# и остальными пользователями

cos_sim = cosine_similarity(Vr, test_user_result.reshape(1, -1))
cos_sim[:10]
```

```
array([[-3.27571309e-20],
       [ 9.36924075e-19],
       [-2.24523324e-24],
       [ 6.09694183e-01],
       [ 8.01965140e-01],
       [ 8.16761990e-01],
       [ 8.10025188e-01],
       [ 9.34472861e-01],
       [ 7.81322508e-23],
       [ 9.54802843e-01]])
```

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

```
array([-3.27571309e-20,  9.36924075e-19, -2.24523324e-24,  6.09694183e-01,
        8.01965140e-01,  8.16761990e-01,  8.10025188e-01,  9.34472861e-01,
        7.81322508e-23,  9.54802843e-01])
```

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

```
8389
```

```
[ ] user_item_matrix.iloc[[8389]]
```

	B001KNI2WE	B000MT8AT2	B002Y2QS8U	B003AOCR22	B008N5VC1A
A374POWERPF1X3	0.0	0.0	0.0	0.0	0.0

1 rows × 3254 columns

```
[ ] test_user
```

```
matrix([[0., 0., 0., ..., 0., 0., 0.]])
```

```
[ ]  # Получение названия товара
     product_list = list(user_item_matrix.columns)
     def product_name_by_id(ind):
         try:
             product = product_list[ind]
             #print(wineId)
             #flt_links = data3[data['movieId'] == wineId]
             #tmdbId = int(flt_links['tmdbId'].values[0])
             #md_links = df_md[df_md['id'] == tmdbId]
             #res = md_links['title'].values[0]
             return product
         except:
             return ''
```

```
[ ]  # Товары, которые оценивал юзер
     i=1
     for idx, item in enumerate(np.ndarray.flatten(np.array(test_user))):
         if item > 0:
             film_title = product_name_by_id(idx)
             print('{} - {} - {}'.format(idx, film_title, item))
             if i==20:
                 break
             else:
                 i+=1
```

```
1636 - B009KAQZ9G - 4.0
```

```
# продукты, которые оценивал наиболее схожий юзер:
i=1
recommended_user_item_matrix = user_item_matrix.loc[['A374POWERPF1X3']]
for idx, item in enumerate(np.ndarray.flatten(np.array(recommended_user_item_matrix))):
    if item > 0:
        film_title = product_name_by_id(idx)
        print('{} - {} - {}'.format(idx, film_title, item))
        if i==20:
            break
        else:
            i+=1
```

```
2206 - B000G7P50M - 5.0
```

Вывод:

При выполнении работы выбран датасет и сформированы рекомендации пищевых товаров Amazon для одного пользователя двумя произвольными способами фильтрации на основе отзывов на эти товары.

Как видно, фильтрация на основе содержания и коллаборативная фильтрация показали различные результаты работы в рамках рекомендательных систем.