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

Лабораторная работа №4
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группы ИУ5-22М
ЧжаоЛян

Москва — 2022 г.

In [83]:
```
#импортируем библиотеки
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_e
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, manhattan_distances
# 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")
```

Чтение и обработка данных

In [194]:
```
data = pd.read_csv(r'C:\Users\80667\Desktop\文件\ИУ5\研一下\MMO\lab\lab4\Highest Holywood Grossing Movies.csv')
data.head()
```

Out[194]:

	Unnamed: 0	Title	Movie Info	Distributor	Release Date	Domestic Sales (in $)	International Sales (in $)	World Sales (in $)	Genre	Movie Runtime	License
0	0	Star Wars: Episode VII - The Force Awakens (2015)	As a new threat to the galaxy rises, Rey, a de...	Walt Disney Studios Motion Pictures	December 16, 2015	936662225	1132859475	2069521700	['Action', 'Adventure', 'Sci-Fi']	2 hr 18 min	PG-13
1	1	Avengers: Endgame (2019)	After the devastating events of Avengers: Infi...	Walt Disney Studios Motion Pictures	April 24, 2019	858373000	1939128328	2797501328	['Action', 'Adventure', 'Drama', 'Sci-Fi']	3 hr 1 min	PG-13
2	2	Avatar (2009)	A paraplegic Marine dispatched to the moon Pan...	Twentieth Century Fox	December 16, 2009	760507625	2086738578	2847246203	['Action', 'Adventure', 'Fantasy', 'Sci-Fi']	2 hr 42 min	PG-13
3	3	Black Panther (2018)	T'Challa, heir to the hidden but advanced king...	Walt Disney Studios Motion Pictures	NaN	700426566	647171407	1347597973	['Action', 'Adventure', 'Sci-Fi']	2 hr 14 min	NaN
4	4	Avengers: Infinity War (2018)	The Avengers and their allies must be willing ...	Walt Disney Studios Motion Pictures	NaN	678815482	1369544272	2048359754	['Action', 'Adventure', 'Sci-Fi']	2 hr 29 min	NaN

In [195]:
```
data.shape
```

Out[195]:
```
(918, 11)
```

In [197]:
```
description_data = data[data['Movie Info'].notnull()]
description_data.shape
```

Out[197]:
```
(918, 11)
```

In [199]:
```
title = description_data['Title'].values
title[0:5]
```

Out[199]:
```
array(['Star Wars: Episode VII - The Force Awakens (2015)',
       'Avengers: Endgame (2019)', 'Avatar (2009)',
       'Black Panther (2018)', 'Avengers: Infinity War (2018)'],
      dtype=object)
```

In [200]:
```
descriptions = description_data['Movie Info'].values
descriptions[0:5]
```

Out[200]:
```
array(['As a new threat to the galaxy rises, Rey, a desert scavenger, and Finn, an ex-stormtrooper, must join Han Solo and Chewbacca to search for the one hope of restoring peace.',
       "After the devastating events of Avengers: Infinity War, the universe is in ruins. With the help of remaining allies, the Avengers assemble once more in order to reverse Thanos' actions and restore balance to the universe.",
       'A paraplegic Marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.',
       "T'Challa, heir to the hidden but advanced kingdom of Wakanda, must step forward to lead his people into a new future and must confront a challenger from his country's past.",
       'The Avengers and their allies must be willing to sacrifice all in an attempt to defeat the powerful Thanos before his blitz of devastation and ruin puts an end to the universe.'],
      dtype=object)
```

In [201]:
```
description_data.keys()
```

Out[201]:
```
Index(['Unnamed: 0', 'Title', 'Movie Info', 'Distributor', 'Release Date',
       'Domestic Sales (in $)', 'International Sales (in $)',
       'World Sales (in $)', 'Genre', 'Movie Runtime', 'License'],
      dtype='object')
```

In [202]:
```
wine_ids = description_data['Unnamed: 0'].values
wine_ids
```

Out[202]:
```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
```

```
754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766,
767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779,
780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792,
793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805,
806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831,
832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844,
845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857,
858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870,
871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883,
884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896,
897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,
910, 911, 912, 913, 914, 915, 916, 917], dtype=int64)
```

In [203]:
```
%%time
tfidf = TfidfVectorizer()
description_matrix = tfidf.fit_transform(descriptions)
description_matrix
```

```
Wall time: 30.9 ms
```

Out[203]:
```
<918x5511 sparse matrix of type '<class 'numpy.float64'>'
        with 21478 stored elements in Compressed Sparse Row format>
```

In [204]:
```
description_matrix
```

Out[204]:
```
<918x5511 sparse matrix of type '<class 'numpy.float64'>'
        with 21478 stored elements in Compressed Sparse Row format>
```

Фильтрация на основе содержания. Метод k-ближайших соседей

In [205]:

```
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
```

In [211]:
```
test_id = 11
print(title[test_id])
print(descriptions[test_id])
```

```
The Lion King (2019)
After the murder of his father, a young lion prince flees his kingdom only to learn the true meaning of responsibility and bravery.
```

In [213]:
```
test_matrix = description_matrix[test_id]
test_matrix
```

Out[213]:
```
<1x5511 sparse matrix of type '<class 'numpy.float64'>'
        with 19 stored elements in Compressed Sparse Row format>
```

In [215]:
```
skr1 = SimplerKnnRecomender(description_matrix, wine_ids, title, descriptions)
```

In [217]:
```
# 15 вин, наиболее похожих на Leon Beyer 2012 Gewurztraminer (Alsace)
# в порядке убывания схожести на основе косинусного сходства
rec1 = skr1.recommend_for_single_object(15, test_matrix)
rec1
```

Out[217]:

	id	title	overview	dist
28	28	The Lion King (1994)	Lion prince Simba and his father are targeted ...	240306.892038
98	98	The Chronicles of Narnia: The Lion, the Witch ...	Four kids travel through a wardrobe to the lan...	200576.652269
656	656	Road to Perdition (2002)	A mob enforcer's son in 1930s Illinois witness...	184047.066277
55	55	Transformers: Dark of the Moon (2011)	The Autobots learn of a Cybertronian spacecraf...	168420.183255
692	692	The Prince of Egypt (1998)	Egyptian Prince Moses learns of his identity a...	154062.199815
811	811	Taken 3 (2014)	Accused of a ruthless murder he never committe...	152976.091052
136	136	Cars (2006)	A hot-shot race-car named Lightning McQueen ge...	144525.338355
261	261	Madagascar: Escape 2 Africa (2008)	The Madagascar animals fly back to New York Ci...	144363.018617
291	291	TRON: Legacy (2010)	The son of a virtual world designer goes looki...	138190.022874
88	88	Harry Potter and the Half-Blood Prince (2009)	As Harry Potter begins his sixth year at Hogwa...	133599.663106
171	171	Beauty and the Beast (1991)	A prince cursed to spend his days as a hideous...	133211.378317
732	732	The Green Hornet (2011)	Following the death of his father, Britt Reid,...	131311.022328
237	237	The Polar Express (2004)	On Christmas Eve, a young boy embarks on a mag...	130620.983631
462	462	The Last Airbender (2010)	Follows the adventures of Aang, a young succes...	130011.210592
347	347	Who Framed Roger Rabbit (1988)	A toon-hating detective is a cartoon rabbit's ...	129374.053893

Out[218]:

	id	title	overview	dist
28	28	The Lion King (1994)	Lion prince Simba and his father are targeted ...	1.232634e+06
98	98	The Chronicles of Narnia: The Lion, the Witch ...	Four kids travel through a wardrobe to the lan...	1.264455e+06
656	656	Road to Perdition (2002)	A mob enforcer's son in 1930s Illinois witness...	1.277461e+06
55	55	Transformers: Dark of the Moon (2011)	The Autobots learn of a Cybertronian spacecraf...	1.289635e+06
692	692	The Prince of Egypt (1998)	Egyptian Prince Moses learns of his identity a...	1.300721e+06
811	811	Taken 3 (2014)	Accused of a ruthless murder he never committe...	1.301556e+06
136	136	Cars (2006)	A hot-shot race-car named Lightning McQueen ge...	1.308033e+06
261	261	Madagascar: Escape 2 Africa (2008)	The Madagascar animals fly back to New York Ci...	1.308157e+06
291	291	TRON: Legacy (2010)	The son of a virtual world designer goes looki...	1.312867e+06
88	88	Harry Potter and the Half-Blood Prince (2009)	As Harry Potter begins his sixth year at Hogwa...	1.316359e+06
171	171	Beauty and the Beast (1991)	A prince cursed to spend his days as a hideous...	1.316654e+06
732	732	The Green Hornet (2011)	Following the death of his father, Britt Reid,...	1.318096e+06
237	237	The Polar Express (2004)	On Christmas Eve, a young boy embarks on a mag...	1.318620e+06
462	462	The Last Airbender (2010)	Follows the adventures of Aang, a young succes...	1.319082e+06
347	347	Who Framed Roger Rabbit (1988)	A toon-hating detective is a cartoon rabbit's ...	1.319565e+06

In [219]:
```
# Манхэттэнское расстояние дает несколько иные результаты поиска
rec3 = skr1.recommend_for_single_object(15, test_matrix,
                                        cos_flag = False, manh_flag = True)
rec3
```

Out[219]:

	id	title	overview	dist
28	28	The Lion King (1994)	Lion prince Simba and his father are targeted ...	5.746275e+06
736	736	Sausage Party (2016)	A sausage strives to discover the truth about ...	6.199520e+06
692	692	The Prince of Egypt (1998)	Egyptian Prince Moses learns of his identity a...	6.351937e+06
593	593	Dr. Dolittle 2 (2001)	Dolittle must save a forest and a bear's life.	6.407191e+06
344	344	The Longest Yard (2005)	Prison inmates form a football team to challen...	6.455570e+06
735	735	Yes Man (2008)	A man challenges himself to say "yes" to every...	6.495309e+06
98	98	The Chronicles of Narnia: The Lion, the Witch ...	Four kids travel through a wardrobe to the lan...	6.508928e+06
642	642	Indecent Proposal (1993)	A billionaire offers $1,000,000 to a young mar...	6.548398e+06
314	314	Knives Out (2019)	A detective investigates the death of a patria...	6.561577e+06
347	347	Who Framed Roger Rabbit (1988)	A toon-hating detective is a cartoon rabbit's ...	6.586593e+06
775	775	Encanto (2021)	A young Colombian girl has to face the frustra...	6.593280e+06
109	109	The Grinch (2018)	A grumpy Grinch plots to ruin Christmas for th...	6.607736e+06
582	582	The Village (2004)	A series of events tests the beliefs of a smal...	6.641202e+06
294	294	True Grit (2010)	A stubborn teenager enlists the help of a toug...	6.676849e+06
422	422	Cheaper by the Dozen (2003)	With his wife doing a book tour, a father of t...	6.713774e+06

Коллаборативная фильтрация. Метод на основе сингулярного разложения

In [220]:
```
data.head()
```

Out[220]:

	Unnamed: 0	Title	Movie Info	Distributor	Release Date	Domestic Sales (in $)	International Sales (in $)	World Sales (in $)	Genre	Movie Runtime	License
0	0	Star Wars: Episode VII - The Force Awakens (2015)	As a new threat to the galaxy rises, Rey, a de...	Walt Disney Studios Motion Pictures	December 16, 2015	936662225	1132859475	2069521700	['Action', 'Adventure', 'Sci-Fi']	2 hr 18 min	PG-13
1	1	Avengers: Endgame (2019)	After the devastating events of Avengers: Infi...	Walt Disney Studios Motion Pictures	April 24, 2019	858373000	1939128328	2797501328	['Action', 'Adventure', 'Drama', 'Sci-Fi']	3 hr 1 min	PG-13
2	2	Avatar (2009)	A paraplegic Marine dispatched to the moon Pan...	Twentieth Century Fox	December 16, 2009	760507625	2086738578	2847246203	['Action', 'Adventure', 'Fantasy', 'Sci-Fi']	2 hr 42 min	PG-13
3	3	Black Panther (2018)	T'Challa, heir to the hidden but advanced king...	Walt Disney Studios Motion Pictures	NaN	700426566	647171407	1347597973	['Action', 'Adventure', 'Sci-Fi']	2 hr 14 min	NaN
4	4	Avengers: Infinity War (2018)	The Avengers and their allies must be willing ...	Walt Disney Studios Motion Pictures	NaN	678815482	1369544272	2048359754	['Action', 'Adventure', 'Sci-Fi']	2 hr 29 min	NaN

In [221]:
```
data3 = data[1:5500]
```

In [223]:
```
# Количество уникальных дегустаторов
len(data3['Distributor'].unique())
```

Out[223]: 34

In [225]:
```
# Количество уникальных вин
len(data3['Title'].unique())
```

Out[225]: 917

In [226]:
```
# Сформируем матрицу взаимодействий на основе рейтингов
# Используется идея из статьи - https://towardsdatascience.com/beginners-guide-to-creating-an-svd-recommender-system-1f
def create_utility_matrix(data):
    itemField = 'Title'
    userField = 'Distributor'
    valueField = 'Domestic Sales (in $)'

    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 [227]:
```
%%time
user_item_matrix, users_index, items_index = create_utility_matrix(data3)
```

```
Wall time: 19.9 ms
```

In [228]:
```
user_item_matrix
```

Out[228]:

	La La Land (2016)	The Village (2004)	The Grinch (2018)	The Pursuit of Happyness (2006)	Ready Player One (2018)	The Green Mile (1999)	Road to Perdition (2002)	The Other Woman (2014)	Back to the Future Part II (1989)	Get Smart (2008)	...	Saving Mr Banks (2013)
Relativity Media	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Columbia Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
FilmDistrict	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
New Line Cinema	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Metro-Goldwyn-Mayer (MGM)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Dimension Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Screen Gems	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Universal Pictures	0.0	0.0	271094731.0	0.0	0.0	0.0	0.0	0.0	119000002.0	0.0	...	0.0
Roadside Attractions	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Lionsgate	151101803.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Orion Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
20th Century Studios	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Walt Disney Studios Motion Pictures	0.0	114197520.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	83301580.0
DreamWorks Distribution	0.0	0.0	0.0	0.0	0.0	0.0	104454762.0	0.0	0.0	0.0	...	0.0
United Artists	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Miramax	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Artisan Entertainment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
The Weinstein Company	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Twentieth Century Fox	0.0	0.0	0.0	0.0	0.0	0.0	0.0	83911193.0	0.0	0.0	...	0.0
Summit Entertainment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
IFC Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
DreamWorks	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Sony Pictures Entertainment (SPE)	0.0	0.0	0.0	163566459.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Newmarket Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
Revolution Studios	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0

In [229]:
```
# Выделение тестовой строки
user_item_matrix__test = user_item_matrix.loc[['Warner Bros.']]
user_item_matrix__test
```

Out[229]:

	La La Land (2016)	The Village (2004)	The Grinch (2018)	The Pursuit of Happyness (2006)	Ready Player One (2018)	The Green Mile (1999)	Road to Perdition (2002)	The Other Woman (2014)	Back to the Future Part II (1989)	Get Smart (2008)	...	Saving Mr. Banks (2013)	Tomorrowland (2015)	Jason Bourne (2016)	A Beautiful Mind (2001)	D (2
Warner Bros.	0.0	0.0	0.0	0.0	137690172.0	136801374.0	0.0	0.0	0.0	130319208.0	...	0.0	0.0	0.0	0.0	

1 rows × 917 columns

In [230]:
```
#taster_names = description_data['taster_name'].unique()
taster_names = np.delete(data3['Distributor'].unique(), 0)
taster_names = np.delete(taster_names, 7)
taster_names
```

Out[230]:
```
array(['Twentieth Century Fox', 'Sony Pictures Entertainment (SPE)',
       'Paramount Pictures', 'Universal Pictures', 'Warner Bros.',
       'DreamWorks Distribution', 'Lionsgate', 'New Line Cinema',
       'Newmarket Films', 'Summit Entertainment', 'Columbia Pictures',
       'IFC Films', 'TriStar Pictures', 'Orion Pictures',
       'Metro-Goldwyn-Mayer (MGM)', 'Miramax', 'The Weinstein Company',
       'Fox Searchlight Pictures', 'Revolution Studios',
       'Artisan Entertainment', 'Sony Pictures Classics',
       'United Artists', 'Screen Gems', 'USA Films',
       '20th Century Studios', 'STX Entertainment', 'Dimension Films',
       'United Artists Releasing', 'FilmDistrict', 'Focus Features',
       'Relativity Media', 'Roadside Attractions'], dtype=object)
```

In [231]:
```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.loc[taster_names]
user_item_matrix__train
```

Out[231]:

	La La Land (2016)	The Village (2004)	The Grinch (2018)	The Pursuit of Happyness (2006)	Ready Player One (2018)	The Green Mile (1999)	Road to Perdition (2002)	The Other Woman (2014)	Back to the Future Part II (1989)	Get Smart (2008)	...	Saving Mr. Banks (2013)	Tomorr
Twentieth Century Fox	0.0	0.0	0.0	0.0	0.0	0.0	0.0	83911193.0	0.0	0.0	...	0.0	
Sony Pictures Entertainment (SPE)	0.0	0.0	0.0	163566459.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Paramount Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Universal Pictures	0.0	0.0	271094731.0	0.0	0.0	0.0	0.0	0.0	119000002.0	0.0	...	0.0	
Warner Bros.	0.0	0.0	0.0	0.0	137690172.0	136801374.0	0.0	0.0	0.0	130319208.0	...	0.0	
DreamWorks Distribution	0.0	0.0	0.0	0.0	0.0	0.0	104454762.0	0.0	0.0	0.0	...	0.0	
Lionsgate	151101803.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
New Line Cinema	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Newmarket Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Summit Entertainment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Columbia Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
IFC Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
TriStar Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Orion Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Metro-Goldwyn-Mayer (MGM)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Miramax	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
The Weinstein Company	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Fox Searchlight Pictures	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Revolution Studios	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Artisan Entertainment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Sony Pictures Classics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
United Artists	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
Screen Gems	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
USA Films	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
20th Century Studios	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

In [232]:
```
%%time
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
```

```
Wall time: 12 ms
```

In [233]:
```
# Матрица соотношения между дегустаторами и латентными факторами
U.shape
```

Out[233]: (917, 917)

In [234]:
```
# Матрица соотношения между объектами и латентными факторами
V.shape
```

Out[234]: (32, 32)

In [235]:
```
S.shape
```

Out[235]: (32,)

In [236]:
```
Sigma = np.diag(S)
Sigma.shape
```

Out[236]: (32, 32)

In [237]:
```
# Диагональная матрица сингулярных значений
Sigma
```

Out[237]:
```
array([[2.31559108e+09, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 2.03551447e+09, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 2.00999035e+09, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        1.00723831e+08, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 8.35040170e+07, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 8.34823520e+07]])
```

In [238]:
```
# Используем 3 первых сингулярных значения
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]
# Матрица соотношения между новым дегустатором и латентными факторами
test_user = np.mat(user_item_matrix__test.values)
test_user.shape, test_user
```

```
0.00000000e+00, 1.01704370e+08, 1.00492203e+08, 0.00000000e+00,
0.00000000e+00, 1.10485654e+08, 8.43511970e+07, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 9.04183420e+07,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.06580051e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
2.56393010e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.33378256e+08, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 1.86848418e+08, 1.01157447e+08,
1.32384315e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.34451603e+08, 0.00000000e+00, 1.30444603e+08,
0.00000000e+00, 1.17450119e+08, 0.00000000e+00, 1.13721571e+08,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 8.41584610e+07, 0.00000000e+00,
0.00000000e+00, 1.03804407e+08, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.65493908e+08, 2.49975996e+08, 0.00000000e+00,
2.10614939e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.07353792e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
```

In [239]:
```
tmp = test_user * Ur * np.linalg.inv(Sr)
tmp
```

Out[239]:
```
matrix([[-1.,  0.,  0.]])
```

In [240]:
```
test_user_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])
test_user_result
```

Out[240]:
```
array([-1.,  0.,  0.])
```

In [241]:
```
# Вычисляем косинусную близость между текущим дегустатором
# и остальными дегустаторами
cos_sim = cosine_similarity(Vr, test_user_result.reshape(1, -1))
cos_sim[:10]
```

Out[241]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

In [242]:
```
# Преобразуем размерность массива
cos_sim_list = cos_sim.reshape(-1, cos_sim.shape[0])[0]
cos_sim_list[:10]
```

Out[242]:
```
array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])
```

In [243]:
```
# Находим наиболее близкого дегустатора
recommended_user_id = np.argsort(-cos_sim_list)[0]
recommended_user_id
```

Out[243]:
```
4
```

In [244]:
```
test_user
```

Out[244]:
```
matrix([[0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         1.37690172e+08, 1.36801374e+08, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 1.30319208e+08, 0.00000000e+00, 0.00000000e+00,
         2.32906145e+08, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00768056e+08,
         1.00920329e+08, 8.69077460e+07, 1.17154724e+08, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 3.28828874e+08, 0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 2.00081192e+08, 1.71479930e+08,
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Подготовка обучающей и тестовой выборки,
кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»
[Электронный ресурс] // GitHub. — 2019. — Режим доступа:
https://github.com/
ugapanyuk/ml_course/wiki/LAB_KNN (дата обращения: 05.04.2019).
[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] //
Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/
stable/ (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. —

Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2019).
[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode:
http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).
[5] dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. — 2017. — Access
mode: https://www.kaggle.com/dronio/SolarEnergy (online; accessed: 18.02.2019).
[6] Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow.
— 2017. — Access mode: https://stackoverflow.com/a/44823381 (online; accessed:
20.02.2019).
[7] scikit-learn 0.20.3 documentation [Electronic resource]. — 2019. — Access mode: https:
//scikit-learn.org/ (online; accessed: 05.04.2019).