

## ▼ Лабораторная работа №4

### "Создание рекомендательной модели"

Выполнила: Пинская Н.М.

Группа: ИУ5-21М

**Цель лабораторной работы:** изучение продвинутых способов предварительной обработки данных для дальнейшего формирования 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).

## ▼ Масштабирование признаков

```
#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
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")
```

```
# Подключение к google диску
```

```

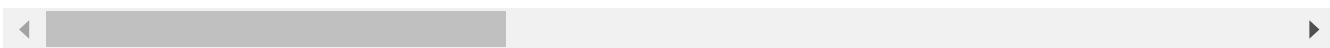
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

# Вывод содержимого папки на диске
import os
data_root = '/content/drive/MyDrive/MMO'
print(os.listdir(data_root))

['pulitzer-circulation-data.csv', 'mmsa-icu-beds2.csv', 'movies.csv.zip', 'movies2.csv.zip', 'movies1.


```



## ▼ Чтение данных

```
df_md_all = pd.read_csv('/content/drive/MyDrive/MMO/rekko_challenge_rekko_challenge_2019/bookmarks.csv')
```

```
df_md_all.head()
```


	user_uid	element_uid	ts	
0	301135	7185	4.430516e+07	
1	301135	4083	4.430516e+07	
2	301135	10158	4.430516e+07	
3	301135	2693	4.430516e+07	
4	301135	2181	4.430515e+07	

```
df_md_all.shape
```

```
(948216, 3)
```

```
df_rt = pd.read_csv('/content/drive/MyDrive/MMO/rekko_challenge_rekko_challenge_2019/ratings.csv')
```

```
df_rt.head()
```

	user_uid	element_uid	rating	ts	
0	571252	1364	10	4.430517e+07	
1	63140	3037	10	4.430514e+07	
2	443817	4363	8	4.430514e+07	
3	359870	1364	10	4.430506e+07	
4	359870	3578	9	4.430506e+07	

```
df_rt.shape
```

```
(438790, 4)
```

```
df_book = pd.read_csv('/content/drive/MyDrive/MMO/anime.csv')
```

```
df_book.head()
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama&#039;	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266



```
df_book.shape
```

```
(12294, 7)
```

Убедимся, что в нашем рабочем датафрейме не будет записей с отсутствующим жанром:

```
df_book_with_genre = df_book[df_book['genre'].notnull()]
df_book_with_genre = df_book_with_genre[~df_book_with_genre['genre'].str.isspace()]
```

```
name = df_book_with_genre['name'].values
name[0:6]
```

```
array(['Kimi no Na wa.', 'Fullmetal Alchemist: Brotherhood', 'Gintama°',
      'Steins;Gate', 'Gintama&#039;',
      'Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou'],
      dtype=object)
```

```
types = df_book_with_genre['type'].values
types[0:5]
```

```
array(['Movie', 'TV', 'TV', 'TV', 'TV'], dtype=object)
```

```
genre = df_book_with_genre['genre'].values
genre[0:3]
```

```
array(['Drama, Romance, School, Supernatural',
      'Action, Adventure, Drama, Fantasy, Magic, Military, Shounen',
      'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen'],
      dtype=object)
```

Векторизуем описания с помощью Tf-Idf Vectorizer

```
tfidf_v = TfidfVectorizer()
genre_matrix = tfidf_v.fit_transform(genre)
genre_matrix
```

```
<12232x47 sparse matrix of type '<class 'numpy.float64'>'
  with 41638 stored elements in Compressed Sparse Row format>
```

И с помощью CountVectorizer:

```
countv = CountVectorizer()
genre_matrix_co = countv.fit_transform(genre)
genre_matrix_co

<12232x47 sparse matrix of type '<class 'numpy.int64'>'
  with 41638 stored elements in Compressed Sparse Row format>
```

```
class SimpleKNNRecommender:
```

```
def __init__(self, X_matrix, X_names, X_notes, X_descr):
    """
    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(
        {'Anime Name': pd.Series(X_names, dtype='str'),
         'Type': pd.Series(X_notes, dtype='str'),
         'Genre': pd.Series(X_descr, 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
```

Выберем тестовый образец, на основе которого мы будем давать рекомендации:

```
test_anime_name = 1000
name[test_anime_name]

'Lady Lady!!'
```

Зададим его матрицу:

```
test_anime_matrix = genre_matrix[test_anime_name]
test_anime_matrix

<1x47 sparse matrix of type '<class 'numpy.float64'>'
  with 2 stored elements in Compressed Sparse Row format>
```

```
skr1 = SimpleKNNRecommender(genre_matrix, name, types, genre)
```

```
test = df_book_with_genre.iloc[test_anime_name]
test
```

```
anime_id      2227
name          Lady Lady!!
genre         Drama, Historical
type          TV
episodes      21
rating        7.73
members       2681
Name: 1000, dtype: object
```

Делаем рекомендацию на основании описания векторизованного Tf-idf и косинусного расстояния:

```
rec1 = skr1.recommend_for_single_object(15, test_anime_matrix)
rec1
```

	Anime Name	Type	Genre	Dist
8118	Anime Roukyoku Kikou Shimizu no Jirochouden	TV	Comedy, Drama, Historical	921100.489503
8082	Ajisai no Uta	OVA	Comedy, Drama, Historical	921100.489503
5738	91 Days Recap	Special	Action, Drama, Historical	884330.883302
620	91 Days	TV	Action, Drama, Historical	884330.883302
8892	Huckleberry no Bouken (Movie)	Movie	Adventure, Drama, Historical	869097.562312
10353	Souya Monogatari	TV	Adventure, Drama, Historical	869097.562312
6257	Apfelland Monogatari	Movie	Adventure, Drama, Historical	869097.562312
10213	Shiroyi Kiba White Fang Monogatari	Special	Adventure, Drama, Historical	869097.562312
6095	Huckleberry no Bouken	TV	Adventure, Drama, Historical	869097.562312
5549	Luiza Shounen Higurashi	Special	Adventure, Drama, Historical	869097.562312

```
test_anime_matrix_co = genre_matrix_co[test_anime_name]
test_anime_matrix_co
```

```
<1x47 sparse matrix of type '<class 'numpy.int64'>'
  with 2 stored elements in Compressed Sparse Row format>
```

```
8888      Okae Jouni      Movie      Drama, Fantasy, Historical      867753.000000
```

```
skr2 = SimpleKNNRecommender(genre_matrix_co, name, types, genre)
```

Делаем рекомендации по описаниям векторизованным CountVectorizer и на основе Евклидова расстояния:

```
skr2.recommend_for_single_object(15, test_anime_matrix_co, cos_flag = False)
```

	Anime Name	Type	Genre	Dist
8738	Hanshin Awaji Daishinsai ni Manabu: Boku wa, A...	OVA	Drama, Historical, Kids	1000000.0
4386	Ashita Genki ni Nare!: Hanbun no Satsumaimo	Movie	Drama	1000000.0
10067	Sabaku no Takara no Shiro	OVA	Drama	1000000.0
10062	Ryouma 30 Seconds	TV	Historical	1000000.0
8826	Hi no Tori: Hagoromo-hen	Movie	Drama	1000000.0
10061	Ryoukan-san	Movie	Historical	1000000.0
265	Kaze Tachinu	Movie	Drama, Historical, Romance	1000000.0
5777	Fuyu no Hi	Movie	Historical	1000000.0
2734	Tokyo Magnitude 8.0 Recap	Special	Drama	1000000.0
3144	Saikyou Bushouden: Sangoku Engi	TV	Historical	1000000.0
3347	Sangokushi (1985)	Special	Historical	1000000.0
10048	Romangeun Eobsda	Movie	Drama	1000000.0
244	Tsumiki no Ie	Movie	Drama	1000000.0
8235	Bonobono: Kumomo no Ki no Koto	Movie	Drama	1000000.0
8765	Heijoukyou: Yasukerashi Miyako	Movie	Historical	1000000.0

---

✓ 0s completed at 6:34 PM

