Лабораторная работа №4 студентки группы ИУ5-21М Дьяконовой Светланы

```
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 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")
```

```
df = pd.read_csv('BI_Software_recommendation_dataset.csv')
```

```
df.head()
```

```
   product_id         category       industry Business_scale user_type
\
0      100001  Data Management      Utilities          Large  Business

1      100002     Database/ERP           Food          Large  Business

2      100003    Data Analysis  Manufacturing          Large  Business

3      100004    Data Analysis             IT         Medium  Business

4      100005     Benchmarking           Food         Medium   Analyst


  no_of_users  deployment       OS mobile_apps      pricing  rating
0      Single       Cloud    Linux           Y     Freemium     4.5
1      Single  On-Premise      Mac           Y     Freemium     4.5
2      Single  On-Premise    Linux           N  Open Source     5.0
3    Mulitple  On-Premise      Mac           Y  Open Source     5.0
4    Mulitple       Cloud  Windows           N   Enterprise     4.2
```

```
df['new_col'] = df.apply(lambda row: str(row.category)+'
'+str(row.industry)+' '+str(row.Business_scale)+' '+str(row.user_type)
+' '+str(row.no_of_users)+' '+str(row.deployment)+' '+str(row.OS)+'
'+str(row.mobile_apps)+' '+str(row.pricing)+' '+str(row.rating), axis
= 1)

df.head()
```

```
   product_id         category       industry Business_scale user_type
\
0      100001  Data Management      Utilities          Large  Business

1      100002     Database/ERP           Food          Large  Business

2      100003    Data Analysis  Manufacturing          Large  Business

3      100004    Data Analysis             IT         Medium  Business

4      100005     Benchmarking           Food         Medium   Analyst


  no_of_users  deployment       OS mobile_apps      pricing  rating  \
0      Single       Cloud    Linux           Y     Freemium     4.5
1      Single  On-Premise      Mac           Y     Freemium     4.5
2      Single  On-Premise    Linux           N  Open Source     5.0
3    Mulitple  On-Premise      Mac           Y  Open Source     5.0
4    Mulitple       Cloud  Windows           N   Enterprise     4.2

                                             new_col
0  Data Management Utilities Large Business Singl...
```

```
1  Database/ERP Food Large Business Single On-Pre...
2  Data Analysis Manufacturing Large Business Sin...
3  Data Analysis IT Medium Business Mulitple On-P...
4  Benchmarking Food Medium Analyst Mulitple Clou...
```

```
from sys import int_info
class SimpleKNNRecommender:
    def __init__(self, X_ids, 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
        tfidfv = TfidfVectorizer()
        self._X_matrix = tfidfv.fit_transform(X_overview)
        self.df = pd.DataFrame(
            {'id': pd.Series(X_ids, dtype='int'),
            'overview': pd.Series(X_overview, dtype='str'),
            'dist': pd.Series([], dtype='float')})


    def recommend_for_single_object(self, K: int, \
                X_object: int, 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
        """
        X_matrix_object = self._X_matrix[X_object]
        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
```

```
rec_movie = df['product_id'].values[0] - 100001
knnr = SimpleKNNRecommender(df['product_id'].values,
df['new_col'].values)
rec1 = knnr.recommend_for_single_object(10, rec_movie)
rec1
```

```
        id                                             overview
dist
66  100067  Data Management Utilities Small Analyst Single...
685269.207785
77  100078  Data Management Pharma Large Analyst Mulitple ...
590908.698584
64  100065  Data Management Fashion Large Business Single ...
515360.044282
31  100032  Data Management Pharma Small Business Mulitple...
473842.811435
27  100028  Data Management Fashion Medium Business Single...
453097.740876
9   100010  Data Management Telecommunications Medium Anal...
452410.038123
28  100029  Data Analysis Utilities Small Analyst Mulitple...
408531.607726
61  100062  Data Analysis Manufacturing Small Business Sin...
407760.364082
94  100095  Data Analysis Marketing Large Business Single ...
397049.216552
2   100003  Data Analysis Manufacturing Large Business Sin...
383417.983629
```

```
ui_df = pd.read_csv('Dataset.csv')

ui_df.head()
```

```
   user_id  item_id  rating  timestamp
0        0       50       5  881250949
1        0      172       5  881250949
2        0      133       1  881250949
```

```
3      196      242       3  881250949
4      186      302       3  891717742
```

```
t_df = pd.read_csv('Movie_Id_Titles.csv')
```

```
t_df.head()
```

```
   item_id              title
0        1   Toy Story (1995)
1        2   GoldenEye (1995)
2        3  Four Rooms (1995)
3        4  Get Shorty (1995)
4        5     Copycat (1995)
```

```
t_id = ui_df['item_id'].values[0]
print(t_df['title'].values[t_id-1])
```

```
Star Wars (1977)
```

```
rec_df = pd.read_csv('Dataset.csv')
rec_df.columns = ['user_id', 'item_id', 'rating', 'timestamp']
rec_df.dropna(inplace=True)
rec_df.reset_index(drop=True, inplace=True)
rec_df
```

```
        user_id  item_id  rating  timestamp
0             0       50       5  881250949
1             0      172       5  881250949
2             0      133       1  881250949
3           196      242       3  881250949
4           186      302       3  891717742
...         ...      ...     ...        ...
99998       880      476       3  880175444
99999       716      204       5  879795543
100000      276     1090       1  874795795
100001       13      225       2  882399156
100002       12      203       3  879959583

[100003 rows x 4 columns]
```

```
def create_utility_matrix(data):
    itemField = 'item_id'
    userField = 'user_id'
    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

user_item_matrix, users_index, items_index =
create_utility_matrix(rec_df)
user_item_matrix
```

```
     1     2     3     4     5     6     7     8     9     10    ...
1673  \
0     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
1     5.0   3.0   4.0   3.0   3.0   5.0   4.0   1.0   5.0   3.0  ...
0.0
2     4.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   2.0  ...
0.0
3     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
4     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
..    ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...
...
939   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   5.0   0.0  ...
0.0
940   0.0   0.0   0.0   2.0   0.0   0.0   4.0   5.0   3.0   0.0  ...
0.0
941   5.0   0.0   0.0   0.0   0.0   0.0   4.0   0.0   0.0   0.0  ...
0.0
942   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
943   0.0   5.0   0.0   0.0   0.0   0.0   0.0   0.0   3.0   0.0  ...
0.0

     1674  1675  1676  1677  1678  1679  1680  1681  1682
0     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
```

```
2     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
..    ...   ...   ...   ...   ...   ...   ...   ...   ...
939   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
940   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
941   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
942   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
943   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[944 rows x 1682 columns]
```

```
user_item_matrix__test = user_item_matrix.loc[[943]]
user_item_matrix__test
```

```
     1     2     3     4     5     6     7     8     9     10    ...
1673  \
943   0.0   5.0   0.0   0.0   0.0   0.0   0.0   0.0   3.0   0.0  ...
0.0

     1674  1675  1676  1677  1678  1679  1680  1681  1682
943   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[1 rows x 1682 columns]
```

```
user_item_matrix__train = user_item_matrix.loc[:942]
user_item_matrix__train
```

```
     1     2     3     4     5     6     7     8     9     10    ...
1673  \
0     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
1     5.0   3.0   4.0   3.0   3.0   5.0   4.0   1.0   5.0   3.0  ...
0.0
2     4.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   2.0  ...
0.0
3     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
4     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
0.0
..    ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...
...
938   4.0   0.0   0.0   0.0   0.0   0.0   4.0   0.0   3.0   0.0  ...
0.0
939   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   5.0   0.0  ...
0.0
940   0.0   0.0   0.0   2.0   0.0   0.0   4.0   5.0   3.0   0.0  ...
0.0
941   5.0   0.0   0.0   0.0   0.0   0.0   4.0   0.0   0.0   0.0  ...
0.0
942   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  ...
```

```
0.0

     1674  1675  1676  1677  1678  1679  1680  1681  1682
0     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
1     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
..    ...   ...   ...   ...   ...   ...   ...   ...   ...
938   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
939   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
940   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
941   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
942   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[943 rows x 1682 columns]
```

```
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
Sigma = np.diag(S)
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]

test_user = np.mat(user_item_matrix__test.values)
tmp = test_user * Ur * np.linalg.inv(Sr)
test_user_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])

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

```
array([[ 0.89418772],
       [ 0.64379767],
       [-0.37125572],
       [-0.32433101],
       [-0.13024201],
       [ 0.9524136 ],
       [-0.39081573],
       [ 0.37839479],
       [ 0.99091292],
       [-0.38074192]])
```

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

```
array([ 0.89418772,  0.64379767, -0.37125572, -0.32433101, -
0.13024201,
        0.9524136 , -0.39081573,  0.37839479,  0.99091292, -
0.38074192])
```

```
recommended_user_id = np.argsort(-cos_sim_list)[0]
recommended_user_id
```

```
577
```

```
movieId_list = list(user_item_matrix.columns)

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

```
GoldenEye (1995) - 5.0
Dead Man Walking (1995) - 3.0
Seven (Se7en) (1995) - 4.0
Usual Suspects, The (1995) - 5.0
Braveheart (1995) - 4.0
Taxi Driver (1976) - 4.0
Rumble in the Bronx (1995) - 4.0
Bad Boys (1995) - 4.0
Apollo 13 (1995) - 4.0
Crimson Tide (1995) - 4.0
Net, The (1995) - 3.0
Billy Madison (1995) - 4.0
Clerks (1994) - 5.0
Star Wars (1977) - 4.0
Legends of the Fall (1994) - 1.0
Natural Born Killers (1994) - 3.0
Outbreak (1995) - 4.0
Professional, The (1994) - 5.0
Pulp Fiction (1994) - 5.0
Quiz Show (1994) - 4.0
```

```
i=1
recommended_user_item_matrix =
user_item_matrix.loc[[list(user_item_matrix.index)[577]]]
for idx, item in
enumerate(np.ndarray.flatten(np.array(recommended_user_item_matrix))):
    if item > 0:
        print('{} - {}'.format(t_df['title'].values[idx], item))
        if i==20:
            break
        else:
            i+=1
```

```
Toy Story (1995) - 5.0
Get Shorty (1995) - 4.0
Copycat (1995) - 4.0
Twelve Monkeys (1995) - 2.0
Babe (1995) - 4.0
Seven (Se7en) (1995) - 2.0
Usual Suspects, The (1995) - 4.0
Mr. Holland's Opus (1995) - 3.0
Braveheart (1995) - 5.0
Birdcage, The (1996) - 4.0
Apollo 13 (1995) - 5.0
Batman Forever (1995) - 3.0
Crimson Tide (1995) - 4.0
Net, The (1995) - 2.0
To Wong Foo, Thanks for Everything! Julie Newmar (1995) - 4.0
Dolores Claiborne (1994) - 3.0
Hoop Dreams (1994) - 5.0
I.Q. (1994) - 4.0
Star Wars (1977) - 4.0
Outbreak (1995) - 4.0
```