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

Отчет
Лабораторная работа № 4
По курсу «Методы машинного обучения»

Создание рекомендательной модели

ИСПОЛНИТЕЛЬ:
Попов Илья Андреевич
Группа ИУ5-23М
_____

"__"_____2022 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.
_____

"__"_____2022 г.

Москва 2022

Popov I.A. IU5-23M lab4

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).

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.metrics.pairwise import cosine_similarity
import surprise
from surprise import KNNBasic, Reader, Dataset, accuracy
from surprise.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib_venn import venn2
```

In [109…
```
raw_data = pd.read_csv('reddit_user_data_count.csv', sep=',')
#movies = pd.read_csv('movies.csv', sep=',')
```

In [115…
```
raw_data.shape
```

Out[115… (1738737, 3)

In [147…
```
#количество уникальный пользователей
unique_users = raw_data['user'].unique()
unique_users.shape[0]
```

Out[147… 37845

In [157…
```
#и-за огромного размера выборки решено оставить данный только от 1000 случайных пользователей
np.random.seed(10)
users_to_retain = 1000
remove_n = unique_users.shape[0] - users_to_retain
reduced_data = raw_data

drop_users = np.random.choice(unique_users, remove_n, replace=False)
for user in drop_users:
    reduced_data = reduced_data.mask(reduced_data['user'] == user)
reduced_data = reduced_data.dropna()
reduced_data
#вычисления данной ячейки были выполнены на более мощном пк и экспортированы в виде reduced_dataset.csv
```

Out[157…

	user	subreddit	count
0	------Username------	AskReddit	20
1	------Username------	Barca	9
2	------Username------	FIFA	4
3	------Username------	MMA	5
4	------Username------	RioGrandeValley	3
...	...	...	...
1738732	zzzayah	teenagersnew	6
1738733	zzzayah	tonsilstones	1
1738734	zzzayah	trees	1
1738735	zzzayah	wallstreetbets	1
1738736	zzzayah	youngpeopleyoutube	1

1738737 rows × 3 columns

In [2]:
```
reduced_data = pd.read_csv('reduced_dataset.csv', sep=',')
reduced_data = reduced_data.drop('Unnamed: 0', axis=1)
```

In [159…
```
reduced_data.head()
```

Out[159…

	user	subreddit	count
0	-_sohcahtoa_-	Advice	2.0
1	-_sohcahtoa_-	AskReddit	54.0

2	-_sohcahtoa_-	BlackPeopleTwitter	15.0
3	-_sohcahtoa_-	Calgary	3.0
4	-_sohcahtoa_-	CrapperDesign	1.0

In [155…
```
reduced_data.shape
```

Out[155… (48308, 3)

In [156…
```
#количество уникальный пользователей
unique_users = reduced_data['user'].unique()
unique_users.shape[0]
```

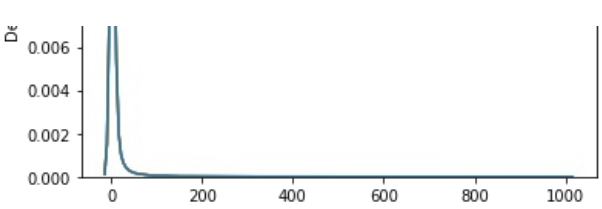
Out[156… 1000

In [160…
```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]
    new_data = dataset
    new_data[column] = all_data.reshape((size,))

    return all_data.reshape((size,)), filled_data, missed_data, new_data

def research_impute_numeric_column(dataset, num_column, const_value=None):
    strategy_params = ['mean', 'median', 'most_frequent', 'constant']
    strategy_params_names = ['Среднее', 'Медиана', 'Мода']
    strategy_params_names.append('Константа = ' + str(const_value))

    original_temp_data = dataset[[num_column]].values
    size = original_temp_data.shape[0]
    original_data = original_temp_data.reshape((size,))

    new_df = pd.DataFrame({'Исходные данные':original_data})

    for i in range(len(strategy_params)):
        strategy = strategy_params[i]
        col_name = strategy_params_names[i]
        if (strategy!='constant') or (strategy == 'constant' and const_value!=None):
            if strategy == 'constant':
                temp_data, _, _, _ = impute_column(dataset, num_column, strategy, fill_value_param=const_value)
            else:
                temp_data, _, _, _ = impute_column(dataset, num_column, strategy)
            new_df[col_name] = temp_data

    sns.kdeplot(data=new_df)
```

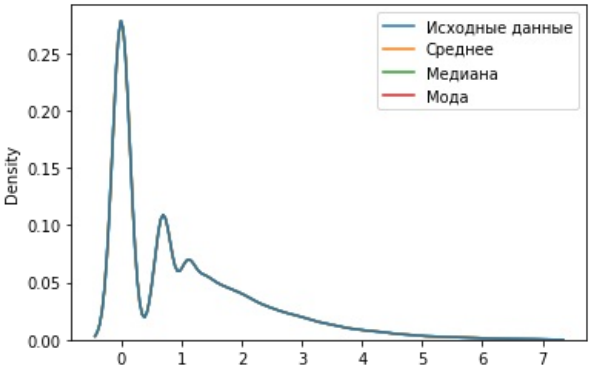
In [161…
```
raw_data.isna().sum()
```

Out[161…
```
user         0
subreddit    0
count        0
dtype: int64
```

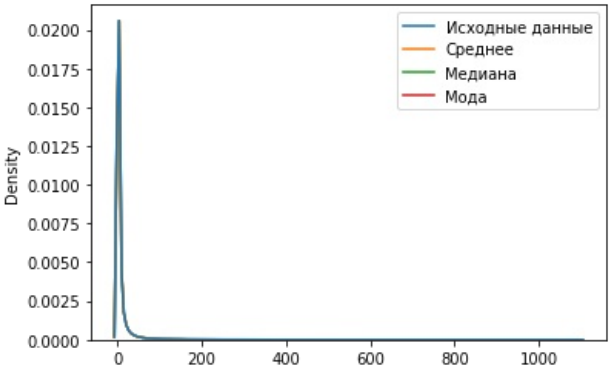
In [162…
```
#У большенства пользователей всего 1 сообщение
research_impute_numeric_column(reduced_data, 'count')
```



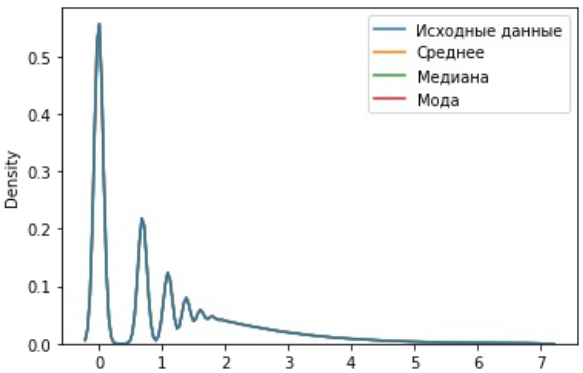
In [163…
```
#нормализуем признак с помощёю логарифма
reduced_data['count'] = np.log(reduced_data['count'])
research_impute_numeric_column(reduced_data, 'count')
```



In [99]:
```
#те же графики для необработаных данных
research_impute_numeric_column(raw_data, 'count')
```



In [100…
```
raw_data['count'] = np.log(raw_data['count'])
research_impute_numeric_column(raw_data, 'count')
```



In [164…
```
#количество уникальных пользователей
reduced_data['user'].unique().shape[0]
```

Out[164… (1000,)

In [169…
```
#количество уникальных тем
reduced_data['subreddit'].unique().shape[0]
```

Out[169… 12500

In [187…
```
reduced_data.shape
```

Out[187… (48308, 3)

In [3]:
```
def create_utility_matrix(data):
    itemField = 'subreddit'
    userField = 'user'
    valueField = 'count'

    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))}
    items_index = {items[i]: i for i in range(len(items))}
    user_ids = [users_index[i] for i in users_index]
    item_ids = [items_index[i] for i in items_index]

    pd_dict = {item: [0.0 for i in range(len(user_ids))] for item in item_ids}

    for i in range(0,data.shape[0]):
        item = items_index[itemList[i]]
        user = users_index[userList[i]]
        value = valueList[i]
        pd_dict[item][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 [4]:
```
itemField = 'subreddit'
userField = 'user'
valueField = 'count'
data = reduced_data

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))}
items_index = {items[i]: i for i in range(len(items))}
user_ids = [users_index[i] for i in users_index]
item_ids = [items_index[i] for i in items_index]
len(userList)
```

Out[4]: 48308

In [5]:
```
user_item_matrix, users_index, items_index = create_utility_matrix(reduced_data)
```

In [201…
```
user_item_matrix
```

Out[201…

	0	1	2	3	4	5	6	7	8	9	...	12490	12491	12492	12493	12494	12495	12496	12497	12498	12499
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	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	0.0	...	0.0	0.0	0.000000	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	0.0	...	0.0	0.0	0.000000	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	0.0	...	0.0	0.0	0.000000	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	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	3.091042	0.0	0.0	0.0	0.0	0.0	0.0	0.0
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1000 rows × 12500 columns

In [203…
```
# Выделение тестовой строки
user_item_matrix__test = user_item_matrix.loc[[0]]
user_item_matrix__test
```

Out[203…

	0	1	2	3	4	5	6	7	8	9	...	12490	12491	12492	12493	12494	12495	12496	12497	12498	12499
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 12500 columns

In [204…
```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.loc[1:]
user_item_matrix__train
```

Out[204…

	0	1	2	3	4	5	6	7	8	9	...	12490	12491	12492	12493	12494	12495	12496	12497	12498	12499
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	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	0.0	...	0.0	0.0	0.000000	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	0.0	...	0.0	0.0	0.000000	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	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	3.091042	0.0	0.0	0.0	0.0	0.0	0.0	0.0
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0

999 rows × 12500 columns

In [205…
```
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
```

In [206…
```
# Матрица соотношения между пользователями и латентными факторами
U.shape
```

Out[206… (12500, 12500)

In [207…
```
# Матрица соотношения между объектами и латентными факторами
V.shape
```

Out[207… (999, 999)

In [208…
```
S.shape
```

Out[208… (999,)

In [209…
```
Sigma = np.diag(S)
Sigma.shape
```

Out[209… (999, 999)

In [210…
```
# Диагональная матрица сингулярных значений
Sigma
```

Out[210…
```
array([[1.01200990e+02, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 5.26707401e+01, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 4.39235795e+01, ...,
```

```
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        2.73055702e-15, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 2.49762824e-15, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 1.41899783e-15]])
```

In [211…
```
# Используем 3 первых сингулярных значения
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]
```

In [212…
```
# Матрица соотношения между новым пользователем и латентными факторами
test_user = np.mat(user_item_matrix__test.values)
test_user.shape, test_user
```

Out[212…
```
((1, 12500), matrix([[0., 0., 0., ..., 0., 0., 0.]]))
```

In [213…
```
tmp = test_user * Ur * np.linalg.inv(Sr)
tmp
```

Out[213…
```
matrix([[-0.03880942,  0.09265148, -0.08130747]])
```

In [214…
```
test_user_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])
test_user_result
```

Out[214…
```
array([-0.03880942,  0.09265148, -0.08130747])
```

In [217…
```
# Вычисляем косинусную близость между текущим пользователем
# и остальными пользователями
cos_sim = cosine_similarity(Vr, test_user_result.reshape(1, -1))
cos_sim[:10]
```

Out[217…
```
array([[ 0.66371526],
       [-0.44975709],
       [-0.52118672],
       [ 0.35570285],
       [ 0.00617574],
       [-0.67155382],
       [-0.63123467],
       [ 0.58759067],
       [ 0.03824613],
       [ 0.13179207]])
```

In [218…
```
# Преобразуем размерность массива
cos_sim_list = cos_sim.reshape(-1, cos_sim.shape[0])[0]
cos_sim_list[:10]
```

Out[218…
```
array([ 0.66371526, -0.44975709, -0.52118672,  0.35570285,  0.00617574,
       -0.67155382, -0.63123467,  0.58759067,  0.03824613,  0.13179207])
```

In [219…
```
# Находим наиболее близкого пользователя
recommended_user_id = np.argsort(-cos_sim_list)[0]
recommended_user_id
```

Out[219…
```
341
```

In [244…
```
#и его имя
recommended_user_name = np.unique(np.array(userList))[recommended_user_id]
recommended_user_name
```

Out[244…
```
'Muhfukin_uhh'
```

In [246…
```
#исходный пользователь
basic_user_name = np.unique(np.array(userList))[0]
basic_user_name
```

Out[246…
```
'-_sohcahtoa_-'
```

In [243…
```
def user_subs(user_name):
    user_threads = reduced_data.where(reduced_data['user'] == user_name)
    return user_threads.dropna()
```

In [254…
```
ru_subs = user_subs(recommended_user_name)
ru_subs
```

Out[254…

	user	subreddit	count
16476	Muhfukin_uhh	AddisonRae	0.000000
16477	Muhfukin_uhh	Afrodisiac	0.000000
16478	Muhfukin_uhh	AmateursHour	0.000000
16479	Muhfukin_uhh	AnimalsBeingDerps	0.000000
16480	Muhfukin_uhh	AskAsians	0.000000
...	...	...	...
16562	Muhfukin_uhh	watch_dogs	2.397895
16563	Muhfukin_uhh	whosthatpornstar	0.000000
16564	Muhfukin_uhh	xxxcaptions	0.000000
16565	Muhfukin_uhh	yakuzagames	3.332205
16566	Muhfukin_uhh	youseeingthisshit	2.079442

91 rows × 3 columns

In [255…
```
bu_subs = user_subs(basic_user_name)
bu_subs
```

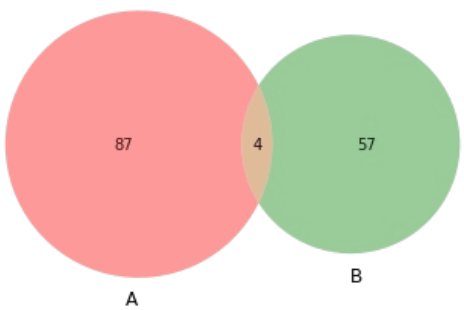
Out[255…

	user	subreddit	count
0	-_sohcahtoa_-	Advice	0.693147
1	-_sohcahtoa_-	AskReddit	3.988984
2	-_sohcahtoa_-	BlackPeopleTwitter	2.708050
3	-_sohcahtoa_-	Calgary	1.098612
4	-_sohcahtoa_-	CrapperDesign	0.000000
...	...	...	...
56	-_sohcahtoa_-	thewalkingdead	0.693147
57	-_sohcahtoa_-	trashy	2.995732
58	-_sohcahtoa_-	untrustworthypoptarts	0.000000
59	-_sohcahtoa_-	videos	0.693147
60	-_sohcahtoa_-	worldnews	1.098612

61 rows × 3 columns

In [256…
```
#имеются 4 одинаковые темы
venn2([set(ru_subs['subreddit']),set(bu_subs['subreddit'])])
```

Out[256…
```
<matplotlib_venn._common.VennDiagram at 0xbf9c56d108>
```



In [286…
```
#выведем список общих тем
np.intersect1d(np.array(ru_subs['subreddit']), np.array(bu_subs['subreddit']))
```

Out[286…
```
array(['AskReddit', 'BlackPeopleTwitter', 'boottoobig',
       'relationship_advice'], dtype=object)
```

In [280…
```
#общие темы понагляднее
def similar_subs(user_a, user_b):
    intersection = np.intersect1d(np.array(user_a['subreddit']), np.array(user_b['subreddit']))
    user_a_values, user_b_values = pd.DataFrame(), pd.DataFrame()
    for sub in intersection:
        user_a_values = pd.concat([user_a_values, user_a.where(user_a['subreddit'] == sub).dropna()])
        user_b_values = pd.concat([user_b_values, user_b.where(user_b['subreddit'] == sub).dropna()])

    return user_a_values, user_b_values
```

In [281…
```
df1, df2 = similar_subs(ru_subs, bu_subs)
```

In [284…
```
df1
```

Out[284…

	user	subreddit	count
16481	Muhfukin_uhh	AskReddit	1.609438
16482	Muhfukin_uhh	BlackPeopleTwitter	0.693147
16521	Muhfukin_uhh	boottoobig	1.098612
16556	Muhfukin_uhh	relationship_advice	0.000000

In [285…
```
df2
```

Out[285…

	user	subreddit	count
1	-_sohcahtoa_-	AskReddit	3.988984
2	-_sohcahtoa_-	BlackPeopleTwitter	2.708050
24	-_sohcahtoa_-	boottoobig	0.693147
52	-_sohcahtoa_-	relationship_advice	1.098612

In [7]:
```
reader = Reader(rating_scale=(0, 8))

# The columns must correspond to user id, item id and ratings (in that order).
data = Dataset.load_from_df(reduced_data, reader)
sim_options = {'name': 'cosine',
               'user_based': False  # compute  similarities between items
               }

trainset, testset = train_test_split(data, test_size=.25)
algo = KNNBasic(sim_options=sim_options)
algo.fit(trainset)
predictions = algo.test(testset)

accuracy.rmse(predictions)
#algo.fit(data)
#pred = algo.predict(uid, iid, r_ui=4, verbose=True)
```

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
RMSE: 46.2016
```

Out[7]:
```
46.20159490290392
```

In [27]:
```
for p in testset:
    if p[0] == '-_sohcahtoa_-':
        print(p)
```

```
('-_sohcahtoa_-', 'AskReddit', 54.0)
('-_sohcahtoa_-', 'relationship_advice', 3.0)
('-_sohcahtoa_-', 'clevercomebacks', 1.0)
('-_sohcahtoa_-', 'gaming', 4.0)
('-_sohcahtoa_-', 'holdmyfries', 12.0)
('-_sohcahtoa_-', 'untrustworthypoptarts', 1.0)
```

```
('-_sohcahtoa_-', 'WWE', 8.0)
('-_sohcahtoa_-', 'RoastMe', 8.0)
('-_sohcahtoa_-', 'funny', 4.0)
('-_sohcahtoa_-', 'CrapperDesign', 1.0)
('-_sohcahtoa_-', 'creepy', 1.0)
('-_sohcahtoa_-', 'canada', 6.0)
('-_sohcahtoa_-', 'HumansBeingBros', 2.0)
('-_sohcahtoa_-', 'mildlypenis', 1.0)
('-_sohcahtoa_-', 'pokemongo', 1.0)
('-_sohcahtoa_-', 'WatchItForThePlot', 2.0)
('-_sohcahtoa_-', 'gifs', 3.0)
('-_sohcahtoa_-', 'worldnews', 3.0)
('-_sohcahtoa_-', 'StardewValley', 5.0)
```

In [15]:
```
for p in predictions:
    if p.uid == '-_sohcahtoa_-':
        print(p)
```

```
user: -_sohcahtoa_- item: AskReddit  r_ui = 54.00   est = 8.00   {'actual_k': 37, 'was_impossible': False}
user: -_sohcahtoa_- item: relationship_advice r_ui = 3.00   est = 8.00   {'actual_k': 30, 'was_impossible': Fal
se}
user: -_sohcahtoa_- item: clevercomebacks r_ui = 1.00   est = 8.00   {'actual_k': 25, 'was_impossible': False}
user: -_sohcahtoa_- item: gaming     r_ui = 4.00   est = 8.00   {'actual_k': 35, 'was_impossible': False}
user: -_sohcahtoa_- item: holdmyfries r_ui = 12.00   est = 7.43   {'actual_k': 18, 'was_impossible': False}
user: -_sohcahtoa_- item: untrustworthypoptarts r_ui = 1.00   est = 6.93   {'actual_k': 18, 'was_impossible': F
alse}
user: -_sohcahtoa_- item: WWE        r_ui = 8.00   est = 8.00   {'actual_k': 4, 'was_impossible': False}
user: -_sohcahtoa_- item: RoastMe    r_ui = 8.00   est = 8.00   {'actual_k': 29, 'was_impossible': False}
user: -_sohcahtoa_- item: funny      r_ui = 4.00   est = 8.00   {'actual_k': 35, 'was_impossible': False}
user: -_sohcahtoa_- item: CrapperDesign r_ui = 1.00   est = 8.00   {'was_impossible': True, 'reason': 'User and
/or item is unknown.'}
user: -_sohcahtoa_- item: creepy     r_ui = 1.00   est = 8.00   {'actual_k': 27, 'was_impossible': False}
user: -_sohcahtoa_- item: canada     r_ui = 6.00   est = 8.00   {'actual_k': 19, 'was_impossible': False}
user: -_sohcahtoa_- item: HumansBeingBros r_ui = 2.00   est = 8.00   {'actual_k': 30, 'was_impossible': False}
user: -_sohcahtoa_- item: mildlypenis r_ui = 1.00   est = 6.51   {'actual_k': 23, 'was_impossible': False}
user: -_sohcahtoa_- item: pokemongo  r_ui = 1.00   est = 8.00   {'actual_k': 24, 'was_impossible': False}
user: -_sohcahtoa_- item: WatchItForThePlot r_ui = 2.00   est = 8.00   {'actual_k': 17, 'was_impossible': False
}
user: -_sohcahtoa_- item: gifs       r_ui = 3.00   est = 8.00   {'actual_k': 37, 'was_impossible': False}
user: -_sohcahtoa_- item: worldnews  r_ui = 3.00   est = 8.00   {'actual_k': 35, 'was_impossible': False}
user: -_sohcahtoa_- item: StardewValley r_ui = 5.00   est = 8.00   {'actual_k': 18, 'was_impossible': False}
```

In [16]:
```
for p in predictions:
    if p.uid == 'Muhfukin_uhh':
        print(p)
```

```
user: Muhfukin_uhh item: photocritique r_ui = 5.00   est = 5.55   {'actual_k': 6, 'was_impossible': False}
user: Muhfukin_uhh item: PopSmoke   r_ui = 16.00   est = 8.00   {'was_impossible': True, 'reason': 'Not enough
neighbors.'}
user: Muhfukin_uhh item: DeathStranding r_ui = 1.00   est = 7.46   {'actual_k': 5, 'was_impossible': False}
user: Muhfukin_uhh item: dji        r_ui = 5.00   est = 5.50   {'actual_k': 2, 'was_impossible': False}
user: Muhfukin_uhh item: GalaxyNote8 r_ui = 1.00   est = 5.00   {'actual_k': 1, 'was_impossible': False}
user: Muhfukin_uhh item: photography r_ui = 8.00   est = 4.84   {'actual_k': 23, 'was_impossible': False}
user: Muhfukin_uhh item: AmateursHour r_ui = 1.00   est = 8.00   {'was_impossible': True, 'reason': 'User and/o
r item is unknown.'}
user: Muhfukin_uhh item: DaysGone   r_ui = 1.00   est = 7.25   {'actual_k': 4, 'was_impossible': False}
user: Muhfukin_uhh item: BonerAlert r_ui = 1.00   est = 8.00   {'was_impossible': True, 'reason': 'User and/or
item is unknown.'}
user: Muhfukin_uhh item: NYYankees  r_ui = 3.00   est = 8.00   {'was_impossible': True, 'reason': 'Not enough n
eighbors.'}
user: Muhfukin_uhh item: itookapicture r_ui = 2.00   est = 6.04   {'actual_k': 25, 'was_impossible': False}
user: Muhfukin_uhh item: WomenOfColor r_ui = 1.00   est = 1.00   {'actual_k': 1, 'was_impossible': False}
user: Muhfukin_uhh item: yakuzagames r_ui = 28.00   est = 8.00   {'actual_k': 10, 'was_impossible': False}
user: Muhfukin_uhh item: japanese   r_ui = 1.00   est = 5.67   {'actual_k': 3, 'was_impossible': False}
user: Muhfukin_uhh item: cyberpunkgame r_ui = 3.00   est = 6.21   {'actual_k': 32, 'was_impossible': False}
user: Muhfukin_uhh item: japanpics  r_ui = 6.00   est = 3.98   {'actual_k': 3, 'was_impossible': False}
user: Muhfukin_uhh item: whosthatpornstar r_ui = 1.00   est = 8.00   {'was_impossible': True, 'reason': 'User a
nd/or item is unknown.'}
user: Muhfukin_uhh item: aestheticrain r_ui = 3.00   est = 1.50   {'actual_k': 2, 'was_impossible': False}
user: Muhfukin_uhh item: nextfuckinglevel r_ui = 18.00   est = 5.26   {'actual_k': 40, 'was_impossible': False}
```