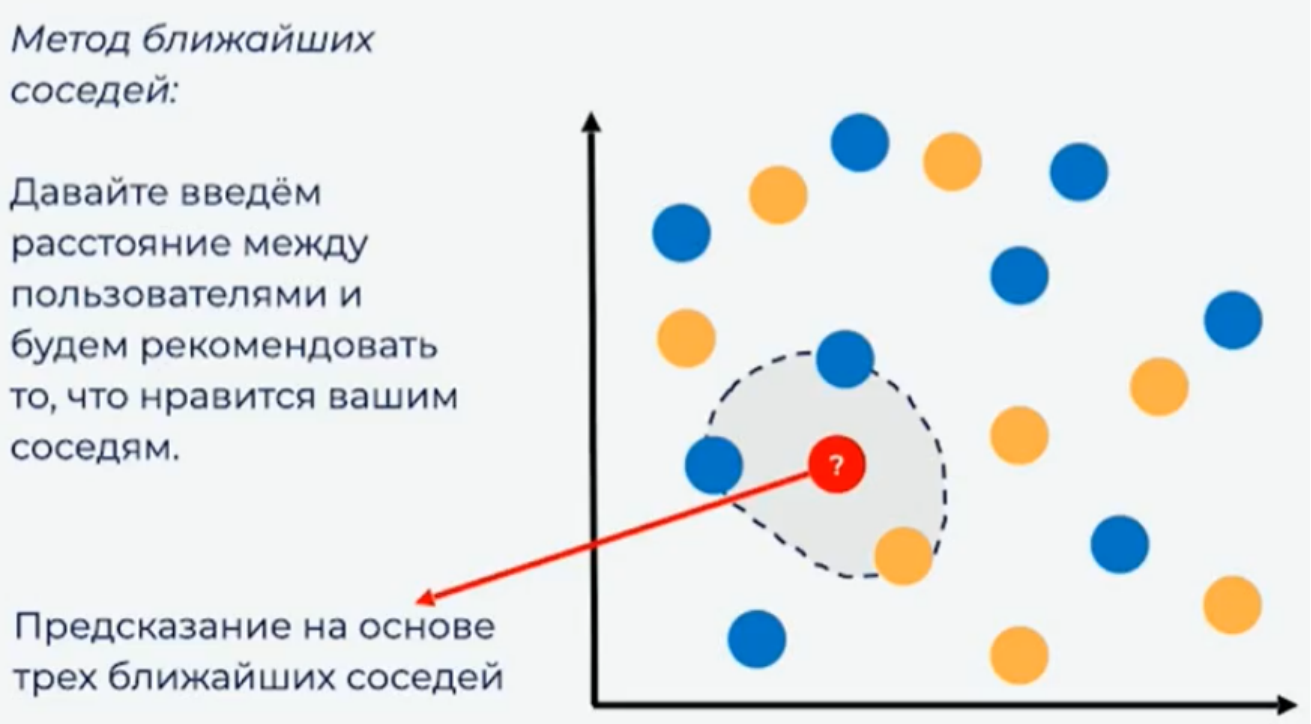
Рекомендательные системы

Всем снова ку!)

Последнюю неделю я изучал нейронные сети, однако понял, что в эти дебри мне лучше не суваться, поэтому давайте пропустим большой курс SkillBox, посвященные этому, и перейдем сразу в систему рекомендаций. Это алгоритм, который использует данные пользователей, для того, чтобы выявлять похожие запросы среди пользователей.

Как это работает

Допустим, вы оценили фильм определенного жанра в 8/10 баллов. И так несколько фильмов. После этого, среди всех пользователей вы начинаете занимать какое-то пространство, Даже можно сказать вектор ответов. Выглядит это так:



Таким образом можно найти соседей среди ваших ответов, и посмотреть, что нравится им. Если вы оцениваете разные фильмы примерно одинаково, то скорее всего то, что понравится этому человеку, понравится и вам!

Реализация

В ходе статьи будет использоваться новая для меня библиотека *Surpriselib*. Эта библиотека позволяет быстро анализировать рекомендательные системы.

Теперь вроде всё, мы со всем разобрались, так что погнали!

```
print("Первым делом скачаем библиотеку, написав $ pip install scikit-surprise```")
```

```
Первым делом скачаем библиотеку, написав $ pip install scikit-surprise```
```

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv("recdemo.csv", sep=";") #небольшой dataset
```

```
df # Мне пришлось вручную в эскеле вбивать значения и делать датасет, тк в уроке не нашел этого файла. Поставьте лайк за это если не сложно пахпапх
```

	id	A	B	C	D	E	F
0	1	3.0	4.0	2.0	4.0	1.0	NaN
1	2	3.0	4.0	2.0	4.0	NaN	2.0
2	3	NaN	2.0	5.0	5.0	NaN	5.0
3	4	NaN	NaN	NaN	NaN	4.0	NaN
4	5	3.0	NaN	2.0	4.0	4.0	NaN
5	6	NaN	5.0	5.0	5.0	NaN	5.0
6	7	1.0	NaN	NaN	2.0	NaN	3.0
7	8	NaN	NaN	NaN	NaN	4.0	4.0

```
df_unpivot=pd.melt(df, id_vars=['id'])
df_unpivot.head(10) # преобразует данные в так называемый аккуратный вид
```

	id	variable	value
0	1	A	3.0
1	2	A	3.0
2	3	A	NaN
3	4	A	NaN
4	5	A	3.0
5	6	A	NaN
6	7	A	1.0
7	8	A	NaN
8	1	B	4.0
9	2	B	4.0

```
df_unpivot.dropna(inplace=True) # Убрали строки, где есть пустые значения
df_unpivot.columns=['userID', 'itemID', 'rating'] # Переименовали колонки в удобные нам названия
df_unpivot.head(10)
```

	userID	itemID	rating
0	1	A	3.0
1	2	A	3.0
4	5	A	3.0
6	7	A	1.0
8	1	B	4.0

	userID	itemID	rating
9	2	B	4.0
10	3	B	2.0
13	6	B	5.0
16	1	C	2.0
17	2	C	2.0

```
from surprise import Dataset
from surprise import Reader

print("Reader отвечает за размер шкалы. В данном случае рейтинги у нас от 1 до 5, поэтому так и пишем")

reader = Reader(rating_scale=(1, 5)) # Зададим разброс оценок
data = Dataset.load_from_df(df_unpivot[['userID', 'itemID', 'rating']], reader) #создадим объект, с которым умеет работать библиотека
```

```
Reader отвечает за размер шкалы. В данном случае рейтинги у нас от 1 до 5, поэтому так и пишем
```

Далее нужно нам просто разбить датасет, как ранее мы делали, на train и test, для просмотра точности выполнения операции.

```
trainset= data.build_full_trainset()
testset=trainset.build_anti_testset()
```

```
testset[0:10] # Номер пользователя, Номер фильма, и средний рейтинг, посчитанный за все значения по элементам массива
```

```
[(1, 'F', 3.4642857142857144),
 (2, 'E', 3.4642857142857144),
 (5, 'B', 3.4642857142857144),
 (5, 'F', 3.4642857142857144),
 (7, 'B', 3.4642857142857144),
 (7, 'C', 3.4642857142857144),
 (7, 'E', 3.4642857142857144),
 (3, 'A', 3.4642857142857144),
 (3, 'E', 3.4642857142857144),
 (6, 'A', 3.4642857142857144)]
```

Давайте быстро вспомним, как работает knn метод (вдруг кто забыл). Создадим сэмпл, на основе которого наша модель обучится. Далее найдем 2х ближайших соседей для точки. Как видите, первые два значения, это расстояния до ближайших соседей, и вторые два значения, это номера этих соседей из списка

```
samples = [[0., 0., 0.], [0., .5, 0.], [1., 1., .5], [0.3, .5, 0.2], [.2, 1., .5]]
from sklearn.neighbors import NearestNeighbors
knn=NearestNeighbors(n_neighbors=2)
knn.fit(samples)
print(knn.kneighbors([[1.,1.,1.]]))
```

```
(array([[0.5       , 0.94339811]]), array([[2, 4]], dtype=int64))
```

Теперь давайте посмотрим на реализацию уже в нашей новой библиотеке для этого метода.

```
from surprise import KNNBaseline
algo=KNNBaseline(k=1) # по 1 ближайшему соседу.
algo.fit(trainset)
predictions=algo.test(testset) # сделаем прогноз для тестсета
```

```
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
```

```
predictions[0:5]
```

```
[Prediction(uid=1, iid='F', r_ui=3.4642857142857144, est=1.9577925900897966, details={'actual_k':
1, 'was_impossible': False}),
 Prediction(uid=2, iid='E', r_ui=3.4642857142857144, est=1.0422074099102034, details={'actual_k':
1, 'was_impossible': False}),
 Prediction(uid=5, iid='B', r_ui=3.4642857142857144, est=4.082661845573158, details={'actual_k': 1,
'was_impossible': False}),
 Prediction(uid=5, iid='F', r_ui=3.4642857142857144, est=2.0826618455731576, details={'actual_k':
1, 'was_impossible': False}),
 Prediction(uid=7, iid='B', r_ui=3.4642857142857144, est=3.8701641215302014, details={'actual_k':
1, 'was_impossible': False})]
```

- uid - id пользователя
- iid - id фильма по которому искался сосед
- est - та оценка, которая нас интересует.

Давайте перепишем результаты в более удобный формат. Для этого создадим таблицу

```
import warnings

warnings.filterwarnings('ignore')
# для того, чтобы убрать надоедливое напоминание, что опр функция уйдет в будущей версии pandas
df_unpivot1=df_unpivot.copy()
for i in predictions:
    df_unpivot1 = df_unpivot1.append({'userID':i.uid, 'itemID': i.iid, 'rating': i.est},
ignore_index=True)
```

```
df_unpivot1.pivot(index='userID', columns='itemID', values='rating')
```

itemID	A	B	C	D	E	F
userID						
1	3.000000	4.000000	2.000000	4.000000	1.000000	1.957793
2	3.000000	4.000000	2.000000	4.000000	1.042207	2.000000
3	3.311189	2.000000	5.000000	5.000000	4.090446	5.000000
4	3.072513	4.197382	2.072513	4.072513	4.000000	3.976639
5	3.000000	4.082662	2.000000	4.000000	4.000000	2.082662

itemID	A	B	C	D	E	F
userID						
6	3.469084	5.000000	5.000000	5.000000	4.248341	5.000000
7	1.000000	3.870164	1.870164	2.000000	3.691628	3.000000
8	3.095874	1.909554	2.095874	4.095874	4.000000	4.000000

Как видите, теперь в таблице убраны все NaN-ы. Да, это сделали мы)

```
df
```

	id	A	B	C	D	E	F
0	1	3.0	4.0	2.0	4.0	1.0	NaN
1	2	3.0	4.0	2.0	4.0	NaN	2.0
2	3	NaN	2.0	5.0	5.0	NaN	5.0
3	4	NaN	NaN	NaN	NaN	4.0	NaN
4	5	3.0	NaN	2.0	4.0	4.0	NaN
5	6	NaN	5.0	5.0	5.0	NaN	5.0
6	7	1.0	NaN	NaN	2.0	NaN	3.0
7	8	NaN	NaN	NaN	NaN	4.0	4.0

```
print("Теперь давайте сделаем всё то же самое, но для k=3")
```

```
Теперь давайте сделаем всё то же самое, но для k=3
```

```
algo = KNNBaseline(k=3)
algo.fit(trainset)
# Than predict ratings for all pairs (u, i) that are NOT in the training set.
predictions = algo.test(testset)
```

```
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
```

```
df_unpivot3 = df_unpivot.copy()
for i in predictions:
    df_unpivot3 = df_unpivot3.append({'userID':i.uid, 'itemID': i.iid, 'rating': i.est},
ignore_index=True)
df_unpivot3.pivot(index='userID', columns='itemID', values='rating')
```

itemID	A	B	C	D	E	F
userID						
1	3.000000	4.000000	2.000000	4.000000	1.000000	2.507435
2	3.000000	4.000000	2.000000	4.000000	2.601745	2.000000
3	3.256100	2.000000	5.000000	5.000000	3.527693	5.000000
4	3.083864	4.197382	2.083864	4.083864	4.000000	3.976639

itemID	A	B	C	D	E	F
userID						
5	3.000000	3.834977	2.000000	4.000000	4.000000	3.013312
6	3.418235	5.000000	5.000000	5.000000	3.590444	5.000000
7	1.000000	3.366011	1.857717	2.000000	3.095321	3.000000
8	2.579863	3.471928	3.463240	4.463240	4.000000	4.000000

Как видите, значения изменились. Так как же нам понять, когда нужно остановиться? Для этого нужно воспользоваться знаниями кросс-платформенной валидации:

Кросс-валидация

```
from surprise.model_selection import cross_validate
```

```
cross_validate(algo,data,cv=2,verbose=True)
```

```
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
Estimating biases using als...
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBaseline on 2 split(s).

                  Fold 1  Fold 2  Mean    Std
RMSE (testset)    1.5231  1.3452  1.4342  0.0889
MAE (testset)     1.1617  1.0102  1.0859  0.0758
Fit time          0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00




{'test_rmse': array([1.52310128, 1.34522382]),
 'test_mae': array([1.16171853, 1.01016551]),
 'fit_time': (0.0010008811950683594, 0.0),
 'test_time': (0.0, 0.0)}
```

Как видите, мы использовали всего 2 фолда (хз что это, но очень интересно). Разбиение в этом методе не нужно на тест и трейн. Оно само само, и слава богу

```
for i in range(1,6):
    algo=KNNBaseline(k=i,verbose=False) # Отключили вывод
    cv=cross_validate(algo,data,measures=['RMSE'],cv=3,verbose=False)
    print(str(i)+'NN:',np.mean(cv['test_rmse']))
```

```
1NN: 1.751042465246463
2NN: 1.7374517019459086
3NN: 1.5664632445650042
4NN: 1.3412878123745717
5NN: 1.4208622735581171
```

Как видите, наибольшее качество у нас при 2 или при 4 соседях

Метод косинусной меры

```
```

```
algo = KNNBaseline(k=5,sim_options= {'name': 'cosine'}, verbose=False)
predictions = algo.fit(trainset).test(testset)
df_unpivot5_cos = df_unpivot.copy()
for i in predictions:
    df_unpivot5_cos = df_unpivot5_cos.append({'userID':i.uid, 'itemID': i.iid, 'rating': i.est},
ignore_index=True)
df_unpivot5_cos.pivot(index='userID', columns='itemID', values='rating')
```

itemID	A	B	C	D	E	F
userID						
1	3.000000	4.000000	2.000000	4.000000	1.000000	3.572362
2	3.000000	4.000000	2.000000	4.000000	2.927003	2.000000
3	2.754335	2.000000	5.000000	5.000000	3.270544	5.000000
4	3.134947	4.197382	2.134947	4.134947	4.000000	3.976639
5	3.000000	3.668772	2.000000	4.000000	4.000000	3.714426
6	2.939492	5.000000	5.000000	5.000000	3.360864	5.000000
7	1.000000	3.442211	2.951617	2.000000	2.802029	3.000000
8	2.700881	3.765123	3.231273	4.431273	4.000000	4.000000

Как видите, сегодня бзе красивых графиков(((
Возможно, позже я буду ориентироваться исключительно на графиках, потому что они довольно интересны и понятны для изучения