Data Scientist. Рекомендательные системы.

Брюханов Д. В.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

План работы:

1. Постановка задачи, исходные данные и что с ними нужно сделать.
2. Подсчет топ 10 самых популярных товаров для каждого пользователя и оценка этого подсчета – как идеал, к которому мы будем стремиться.
3. Roadmap работы, какие подходы и какие библиотеки использовались.
    3.1 Implicit ALS и kNN
    3.2 LightFM
    3.3 Нейросеть
4. Подведение итогов и вывод.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Постановка задачи:

Нам нужно разработать рекомендательную систему подбора товаров, исходя из индивидуальных особенностей каждого пользователя. Представим задачу графически.

Нам нужно:

Вот это

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered
0	2539329	1	1	2	8	NaN	196	1.0	0.0
1	2539329	1	1	2	8	NaN	14084	2.0	0.0
2	2539329	1	1	2	8	NaN	12427	3.0	0.0
3	2539329	1	1	2	8	NaN	26088	4.0	0.0
4	2539329	1	1	2	8	NaN	26405	5.0	0.0

Превратить вот в это

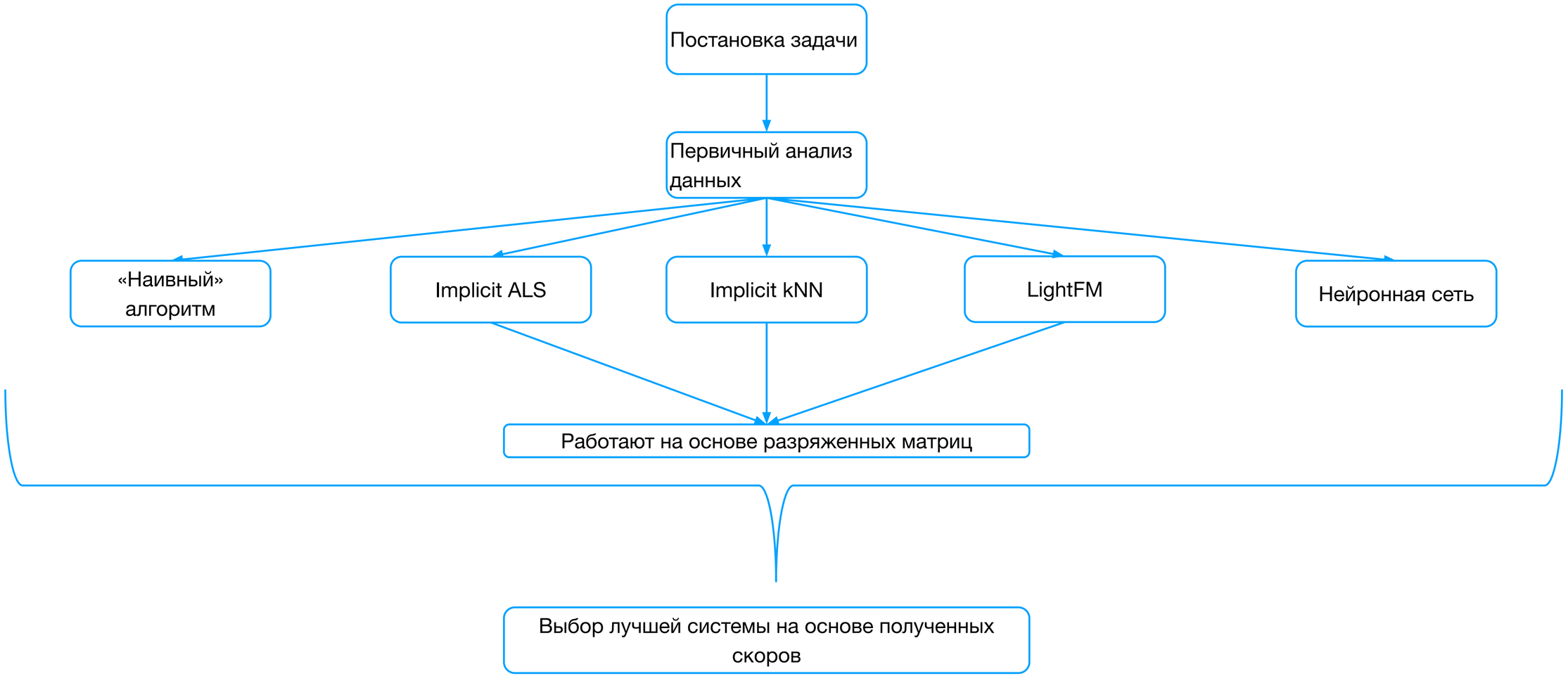
	user_id	product_id
0	1	10326 10258 13032 13176 38928 41787 17122 1303...
1	2	17758 17872 47209 5907 32052 47766 48210 35917...
2	3	8021 42557 28373 1005 12845 18599 1005 22035 1...
3	7	42265 14332 9598 39121 33740 519 19678 29894 2...
4	13	42248 32850 1689 18168 18168 42248 41480 5652 ...

Получив при этом скор (MAP@10) >= 0,2094

Но, стремится мы будем сюда □ > 0,25

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Roadmap экспериментов:



Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Разбор алгоритмов:

Первичный анализ и «Наивный алгоритм»

С ним все просто: мы группируем данные по пользователю и считаем кол-во купленных за всю историю продуктов. В результате будем иметь для каждого пользователя такую картину:

	product_id	amount
0	33754	17
1	21903	14
2	24838	14
3	17704	13
4	46667	13

Повторим эту операцию 100 000 раз (для каждого пользователя) и преобразовав к итоговому виду, получим скор 0.27841.
Это будет наш эталон, к нему мы будем стремиться, однако мы не можем назвать такой подход рекомендательной системой, ведь мы ничего нового не предлагаем пользователю, а принимаем в учет только те товары, про которые он сам прекрасно знает и, более того, любит, т к это его топ.

LightFM

(Матрица формируется не из конкретных данных, а из их *индексов*)

- Нельзя просто так взять и сделать матрицу из списка продуктов/пользователей, необходим маппинг по индексам.
- При создании модели можно настраивать и изменять: (функцию потерь, специальные доп. фичи, кол-во эпох обучения, кол-во ядер процессора)
- Лучший скор - 0.05358
- Ходят слухи, что алгоритм может быть довольно эффективен при правильной подготовке тех самых фичей.

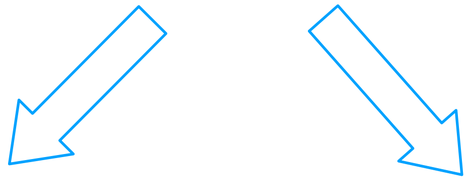
Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Разбор алгоритмов:

Implicit

(При подготовке данных необходимо категориальное кодирование)

- Удобная и быстрая подготовка данных
- Возможность явно передать прямо в библ. модель кол-во товаров, которое мы хотим рекомендовать.
- Идеальный вариант, если нужно что-то кому-то быстро предложить



ALS (Alternating Least Squares)

- Основные параметры настройки факторы и итерации (factors, iterations)
- Наиболее удачный вариант в нашем случае: factors=30, iterations=8
- Лучший скор - 0.05377
- Существенно не дотягивает до наших целей

kNN (k Nearest Neighbours)

- Основные параметры настройки - собственно кол-во ближайших соседей
- Наиболее удачный вариант в нашем случае: k=10 (10 соседей)
- Лучший скор - 0.12458
- Также не дотягивает до таргета, но уже неплохо

Примечание: *в один прекрасный день перестал устанавливаться в Colab стандартным способом (!pip install). Помогла только такая матерная конструкция:*
!sudo -H python3 -m pip install implicit --no-cache --force-reinstall --log ./implicit.txt

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Несколько слов о разряженных матрицах.

Ключевым элементов в работе вышеописанных библиотек является т н разряженная матрица (sparse matrix). Чтобы получить эту самую матрицу, сначала нам требуется преобразовать входные данные. Считаем кол-во добавлений каждого товара в корзину для каждого пользователя, кодируем, и после этого с помощью scipy превращаем в разряженную матрицу:

Исходная группировка

	user_id	product_id	add_to_cart_order
0	1	196	10
1	1	10258	9
2	1	10326	1
3	1	12427	10
4	1	13032	3
...	...	...	...
9459060	206209	43961	3
9459061	206209	44325	1
9459062	206209	48370	1
9459063	206209	48697	1
9459064	206209	48742	2

Кодированная разряженная матрица

```
(0, 195)      10.0
(0, 10216)    9.0
(0, 10282)    1.0
(0, 12375)    10.0
(0, 12976)    3.0
(0, 13119)    2.0
(0, 14026)    1.0
(0, 17053)    1.0
(0, 25032)    8.0
(0, 25978)    2.0
(0, 26294)    2.0
(0, 30318)    1.0
(0, 35802)    1.0
(0, 38766)    1.0
(0, 39489)    1.0
(0, 41609)    1.0
(0, 45950)    3.0
(0, 49016)    2.0
(1, 22)       1.0
(1, 78)       1.0
(1, 1550)     6.0
(1, 1992)     4.0
(1, 2562)     2.0
(1, 3137)     1.0
(1, 4049)     1.0
:     :
(99999, 26099)        1.0
(99999, 26391)        1.0
(99999, 26522)        1.0
```

Для работы на понадобится 2 такие матрицы: одна «**item-user**» и другая «**user-item**». Первую будем использовать для обучения алгоритма, а вторую для предсказаний. Можно так же транспонировать одну матрицу и использовать для того и другого, но лучше срабатывает первый способ.

После того как матрицы готовы остается только обучить на одной из них алгоритмы и после предсказать(порекомендовать) на другой требуемый топ товаров для желаемого пользователя или списка пользователей.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Алгоритм на основе нейронной сети.

- В качестве вишенки на торте и чего-то по-настоящему рабочего было решено воспользоваться нейронной сетью.

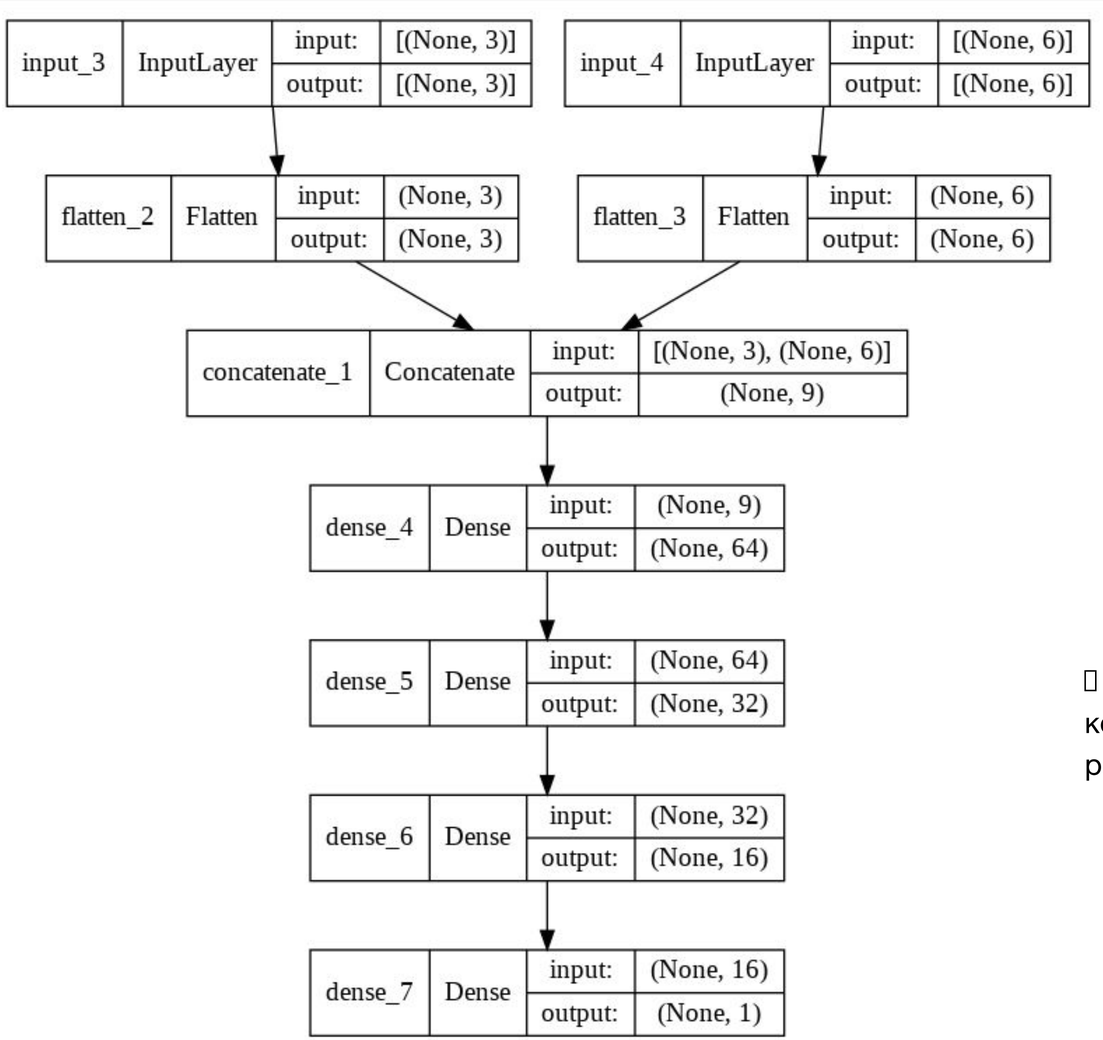
- Исходный набор признаков был преобразован и дополнен, и в итоге получилось 3 признака для пользователей и 6 для продуктов. В качестве таргета был взят факт повторного заказа того или иного продукта конкретным юзером. Весь сет был разбит на трэйн и тест выборки по принципу разницы между первым и последним номера заказа. Если разница была больше 1, то такие строки уходили в трэйн, если меньше – то в тест.

- Идея была в том, чтобы по величине сырой сигмоиды после предсказаний судить о «силе» или весе того или иного продукта для конкретного пользователя. Отсортировав все эти продукты по убыванию значения сигмоиды, можно было получить топ N. Не всегда это был топ 10, иногда больше, иногда меньше, ведь с некоторыми продуктами некоторые пользователи могли вовсе не взаимодействовать.

- Итоговый сет из 10 продуктов добивался случайным образом из исторического топ10 для каждого юзера. Также он добивался и рандомом из всех возможных продуктов, но добивать из топ10 было выгодней для скора приблизительно на 0.015. Повторы естественно исключались.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Архитектура нейросети:



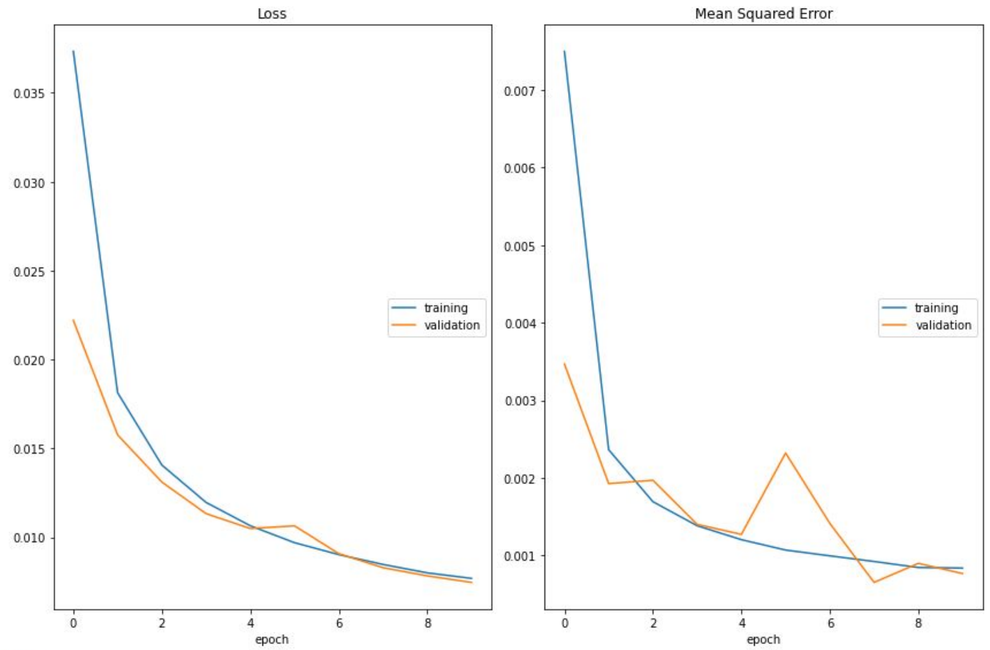
- 2 входа отдельно для продуктовых и пользовательских признаков
- Превращаем данные в одномерный тензор
- Объединение
- Некоторый стек полносвязных слоев. Он менялся в ходе экспериментов. Менялось кол-во слоев, выходное пространство первого слоя и последующих, значение регуляризации l2, общего для всех слоев.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Интересные наблюдения.

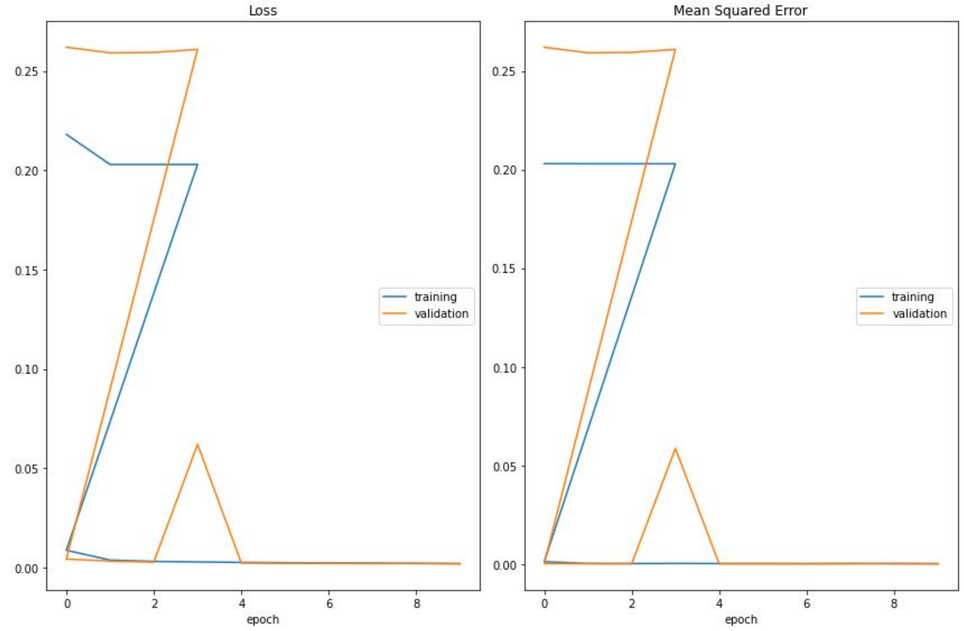
Приведу две наиболее удачные конфигурации сетей: одна дала наилучшие финальные рекомендации, другая наивысшую точность предсказаний (accuracy):

Архитектура показана на пред. слайде, Adam, lr=1e-3, reg. l2=0.01




```
Loss
       training              (min:    0.008, max:    0.037, cur:    0.008)
       validation            (min:    0.007, max:    0.022, cur:    0.007)
Mean Squared Error
       training              (min:    0.001, max:    0.007, cur:    0.001)
       validation            (min:    0.001, max:    0.003, cur:    0.001)
56702/56702 [==============================] - 256s 5ms/step - loss: 0.0077 - mse: 8.3894e-04 - val_loss: 0.0075 - val_mse: 7.6820e-04
```

4 dense(128, 64, 32, 16), Adam, lr=1e-3, reg. l2=0.001




```
Loss
       training              (min:    0.002, max:    0.218, cur:    0.002)
       validation            (min:    0.002, max:    0.262, cur:    0.002)
Mean Squared Error
       training              (min:    0.000, max:    0.203, cur:    0.000)
       validation            (min:    0.000, max:    0.262, cur:    0.000)

Epoch 00010: val_mse did not improve from 0.00009
56702/56702 [==============================] - 251s 4ms/step - loss: 0.0020 - mse: 2.2548e-04 - val_loss: 0.0018 - val_mse: 1.1837e-04
```

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Формирование рекомендаций по предсказаниям нейросети:

Вид итогового датафрейма, отсортированного по величине выхода сигмоиды:

	user_id	product_id	predictions
0	1	196	0.975069
1098	1	12427	0.974666
1309	1	10258	0.970747
1408	1	25133	0.967029
1590	1	46149	0.929090
...	...	...	...
692610	206209	31477	0.695112
844490	206209	38730	0.683582
620478	206209	6567	0.670001
885094	206209	22920	0.654735
68584	206209	14197	0.507085

1079141 rows × 3 columns

	user_id	product_id
0	1	196 12427 10258 25133 46149 38928 35951 39657 ...
1	2	47209 24852 18523 19156 1559 33754 16589 7781 ...
2	3	39190 47766 21903 17668 43961 38167 24852 2389...
3	7	40852 37602 21137 31683 13198 30391 27690 4727...
4	13	27086 4210 27435 43086 42248 14197 34382 41926...
...	...	...
99995	206202	38837 12919 17038 16797 41177 17459 22220 4585...
99996	206206	29326 21850 45139 38167 24852 23892 6846 24852...
99997	206207	44632 36011 13176 39180 33787 33754 48086 2734...
99998	206208	34213 13176 21137 43086 35939 47626 35688 3428...
99999	206209	9405 24852 16168 39216 31477 38730 6567 22920 ...

100000 rows × 2 columns

В результате цель была достигнута и получен **MAP@10 - 0.27080**, что очень близко к оценке исторического топ10 пользователей.

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox

Вывод:

- Исходя из опыта работы с Implicit сложилось впечатление, что это просто не под Windows, минимум под Docker, хотя Colab-у теоретически должно быть все равно, однако же работало это ниже ожидаемого уровня по данной задаче. kNN должен был показать 0.2, судя по опыту решения этого кейса на просторах интернета.

- LightFM. Его плюсы только в том, что устанавливается он просто и легко, ну и пожалуй он весьма вариативен в настройке модели. Думаю, поигравшись с этим настройками можно было немного разогнать скор. Но все же в простоте и точности существенно проигрывает Implicit kNN.

- Нейросеть в очередной раз показала себя как ультимативный универсальный оптимизатор, однако пайплайн с ней самый нетривиальный, я бы даже сказал изысканный, и подготовка данных самая долгая.

- Поэтому подводя итог, хочется сказать что: если важна скорость и допустимо сделать продукт «на коленке», то выбор падает однозначно на Implicit, если же важна точность – то тогда нейросеть.

Спасибо за внимание!

Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox Skillbox