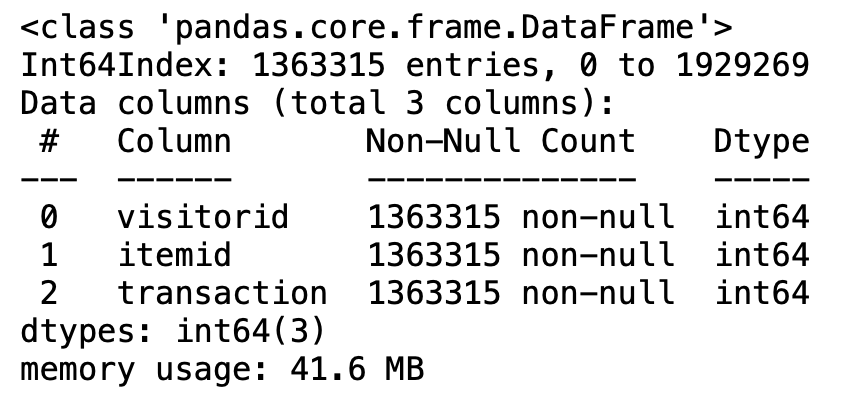
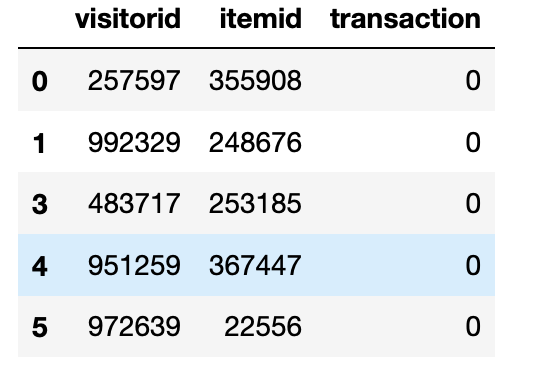
**Документация**

1. **Входные данные для обучения (какой формат, какие факторы).**

Для обучения модели **SVD** из библиотеки **surprise** необходим специальный формат данных (который получается путем считывания датасета через Dataset.load\_from\_df()). Изначально датасет должен иметь вид 3 столбцов: **user\_id, item\_id, rating** (в моем случае **transaction** - наличие/отсутствие покупки товара).



**2. Трансформации исходного датасета, чтобы получить датасет для обучения.**

events = pd.read\_csv("events.csv")

train, test = train\_test\_split(events, test\_size=0.3, shuffle=False)

properties = pd.concat([pd.read\_csv("item\_properties\_part1.csv"),pd.read\_csv("item\_properties\_part2.csv")])

top\_properties\_5 = properties.drop\_duplicates(['itemid', ‘property’]).\groupby("property"['itemid'].count().sort\_values(ascending=False)[:5]

properties\_filtered\_5 = properties[properties['property'].isin(set(top\_properties\_5.index))]

properties\_filtered\_5 = properties\_filtered\_5[['itemid', 'property', 'value']]

properties\_filtered\_5 = properties\_filtered\_5.drop\_duplicates()

items\_5 = properties\_filtered\_5['itemid']

items\_5 = items\_5.drop\_duplicates()

train\_5 = train[train.itemid.isin(items\_5)]

coll\_filter\_data = train\_5[['visitorid', 'itemid', 'event']]

coll\_filter\_data['event'].replace({'view': 0, 'addtocart': 0, 'transaction': 1}, inplace=True)

coll\_filter\_data = coll\_filter\_data.rename(columns = {'event': 'transaction'})

coll\_filter\_data.drop\_duplicates(inplace = True)

coll\_filter\_data.info()

**3. Построение валидации.**

train, test = train\_test\_split(events, test\_size=0.3, shuffle=False) –

значения из датасета **events**, берем 30% в качестве теста и не перемешиваем, т.к. изначальная сортировка в нем идет по времени, а это важно, чтобы тестирование проходило на более новых данных (будто предсказываем поведение клиента в будущем, имея эти действия).

**4. Проведённые эксперименты.**

**SVD, n\_factors=5, n\_epochs=200, lr\_all=0.002, reg\_all=0.05 - Precision@3 = 0.8694 (train), 0.8447 (test)**

SVD, n\_factors=5, n\_epochs=50, lr\_all=0.002, reg\_all=0.05 - Precision@3 = 0.8278 (train), 0.8305 (test)

SVD, n\_factors=5, n\_epochs=30, lr\_all=0.001, reg\_all=0.1 - Precision@3 = 0.8142 (train), 0.8208 (test)

SVD, n\_factors=10, n\_epochs=100, lr\_all=0.001, reg\_all=0.1 - Precision@3 = 0.8255 (train), 0.8280 (test)

SVD, n\_factors=10, n\_epochs=50, lr\_all=0.001, reg\_all=0.05 - Precision@3 = 0.8177 (train), 0.8229 (test)

SVD, n\_factors=10, n\_epochs=30, lr\_all=0.002, reg\_all=0.05 - Precision@3 = 0.8209 (train), 0.8265 (test)

Кроме того, был проведен **GridSearchCV** с сеткой (откуда и были подобраны итоговые гиперпараметры):

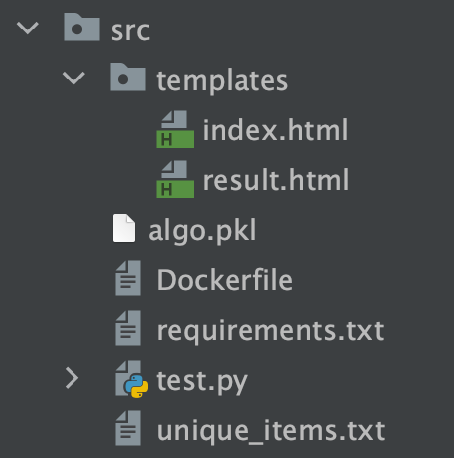
**param\_grid = {'n\_factors': [5, 10, 20], 'n\_epochs': [20, 30, 50], 'lr\_all': [0.002, 0.005, 0.01], 'reg\_all': [0.02, 0.05, 0.1]}**

**5. Docker-образ сервиса, его общее устройство.**

Команда для запуска образа:

**docker pull saiiiok/my\_rep:ipatov\_diploma**

Образ размещен в репозитории, нужно его оттуда загрузить вышеуказанной командой.

В нем размещены следующие файлы: рекомендательная модель, 2 html-файла, список уникальных товаров, список с python-библиотеками для установки (*numpy, surprise, flask, flask\_wtf*), сам докер-файл и непосредственно тело программы.

**6. API.**

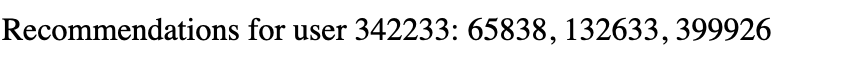
Сервис предоставляет возможность получить рекомендацию 3 товаров для пользователя с введенным User ID.

Он работает по адресам: 127.0.0.1:5000 /, /index, /result.

При GET-запросе эти страницы показывают одно и тоже, а именно:



После ввода информации (User ID) - при корректном вводе (введено целое число) и нажатии кнопки «Рассчитать» - будет отправлен POST-запрос, который запустит расчет предсказания рекомендаций для указанного пользователя, в ответ (по адресу /result) будет получено сообщение (формат - string).



В случае ввода некорректных данных (формат - не целое число), при отправке такого POST-запроса (нажатие кнопки «Рассчитать») в ответ (по адресу /result) будет получено следующее сообщение:

