# Рекомендательная система топ3 товара для клиента. Основана на алгоритме xgboost.

## 1.Инструкция к использованию:

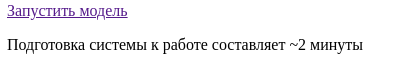
Система поставляется в докер образе sawa25proj.tar.

Следует загрузить образ и запустить выполнение командами:

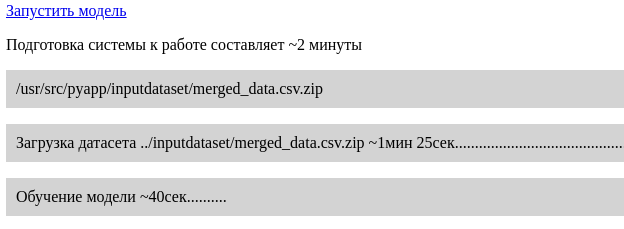
* sudo docker load -i sawa25proj.tar
* sudo docker run -d --rm -v $PWD/inputdataset:/usr/src/pyapp/inputdataset -p 5000:5000 --name=sawa25proj sawa25proj

При этом папка inputdataset доступна для обновления датасета merged\_data.csv.zip при необходимости, имя жестко задано.После активирования образа требуется первоначальная инициализация, запустить которую нужно через веб браузер по адресу: <http://127.0.0.1:5000>

На первой странице содержится ссылка для запуска модели.

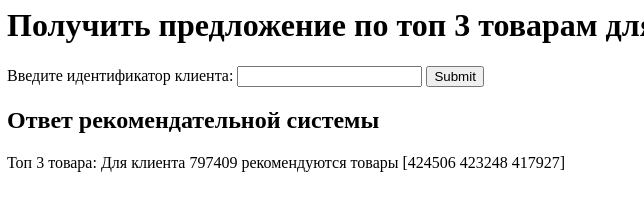


Переход по ссылке запускает код загрузки датасета и обучения модели.



По завершении этого процесса модель готова к работе. С этого момента доступен ввод через вебформу, а так же через api функции.

В поле ввода вебформы следует ввести visitorid клиента, ответом системы будет топ 3 itemid, рекомендуемые для данного клиента.



Если заданного клиента нет в датасете, то будет предложено топ 3 товара по популярности их покупок всеми клиентами.

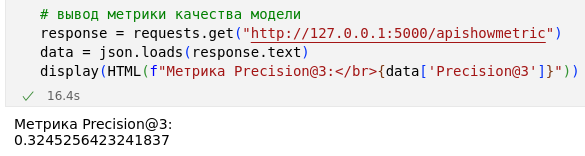
Модель обучена на первых 80% датасета в хронологическом порядке событий. Это означает, что оставшиеся 20% событий являются более поздними событиями. Для клиентов прогнозируются будущие покупки на основе их прошлых действий и комплекса других факторов, включая других клиентов с их покупками и многих производных признаков. Корректность прогнозов подтверждается тем, что на вход модели подается срез первых 80% датасета и для заданного клиента модель определяет товары, которые бы этот клиент купил в будущем, т.е. эти предлагаемые товары присутствуют в оставшихся 20% датасета более поздних событий.

По команде <http://localhost:5000/apishowmetric> можно получить вывод метрики {"Precision@3":0.3245256423241837}

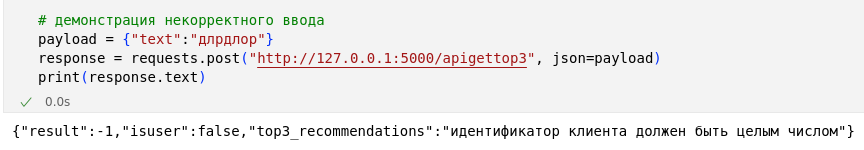
Если введенный visitorid не является числом, то система предупредит о недопустимости такого ввода.

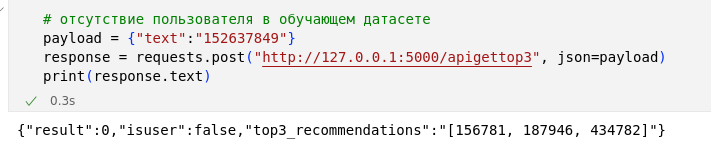
Если заданный visitorid не содержится в тренировочном датасете, то система предложит наиболее популярные товары из всех.

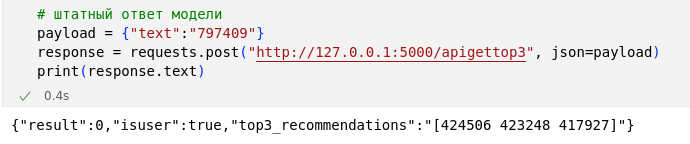
Работа с моделью доступна через api. Отдельно предоставлена функция для расчета технической метрики:



Сервис api возвращает словарь, где   
**result**=0:удача/-1:ошибка;  
**isuser**=true:такой visitorid найден в базе/false: не было такого клиента в обучающем датасете;  
**top3\_recommendations**= "[424506 423248 417927]" – строка с тремя рекомендованными к покупке товарами itemid;







## 2.Подготовка данных включает:

Код подготовки содержится в нотбуке start.ipynb в папке проекта на гитхаб:https://github.com/sawa25/DIPDIP. Принятая к MVP модель является XGBoost и приведена в соответствующей секции в нотбуке.

Краткое содержание подготовки данных:

* Очистка некорректных записей
* Генерация новых факторов
* Компоновка частей в единый датасет, поступающий на обучение.

**Входные датасеты:**

events — датасет с событиями. Колонки:

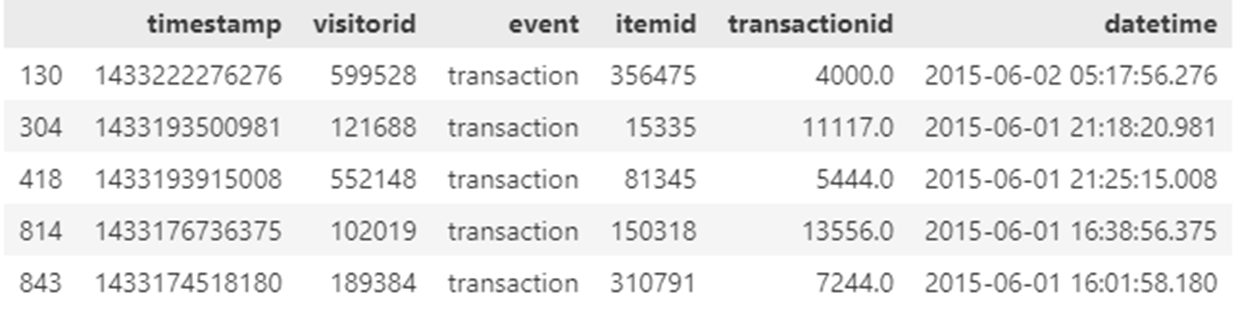
# timestamp — время события

# visitorid — идентификатор пользователя

# event — тип события ['view', 'addtocart', 'transaction']

# itemid — идентификатор объекта

# transactionid — идентификатор транзакции, если она проходила



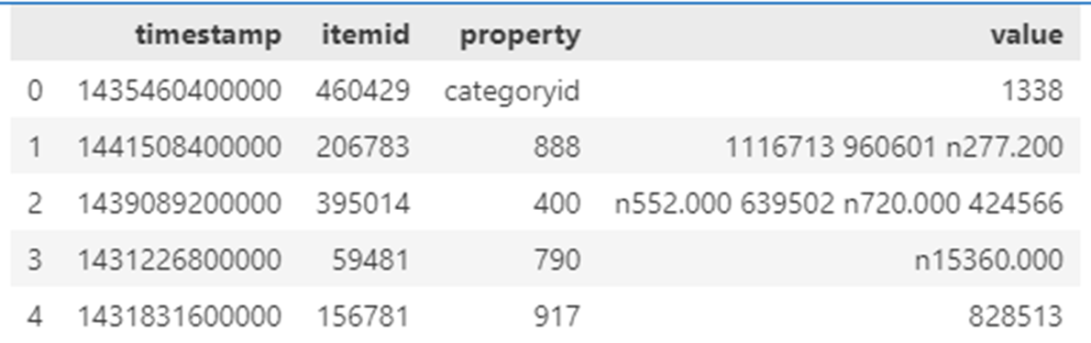
Properties – свойства товаров. Колонки:

# timestamp — момент записи значения свойства

# item\_id — идентификатор объекта

# property — свойство (числовой код)

# value — значение свойства (список буквенночисловых кодов)



Краткое описание трансформации исходных датасетов (код трансформации приведен в нобуке start.ipynb в секции XGBoost):

1.Очистка некорректных записей предлагается по причине наличия событий транзацкий без добавления в корзину товара. Исходный датасет так же содержит записи транзакций без просмотра, но такие записи принято решение не удалять, если есть факт добавления в корзину. Т.о. первый шаг – удаление из датасета строк транзакций, где для сочетания visitorid itemid нет события 'addtocart'.

2.Объединение двух частей item\_properties\_part1.csv.zip, item\_properties\_part2.csv.zip в единый датасет properties

3.Генерация производных факторов для датасета **events** на основе признака event\_datetime.

4.Первичное уменьшение размерности датасета **events** : анализ показал, что 99% просмотров приходится на 10082 товаров из 213782, которые никогда не куплены (т.е. эти товары только смотрели и/или добавляли в корзину, но не покупали), поэтому отброшено (213782-10082) товаров, приходящихся на 1% просмотров пользователями.

5.Генерация признака cumulative\_purchase, показывающего количество покупок данного одинакового товара одним и тем же пользователем, т.к. обнаружилось, что некоторые товары с течением времени покупаются клиентом повторно до нескольких раз, значит можно рекомендовать клиенту товары, которые он уже купил ранее.

6.Уменьшение размерности датасета properties , оставляем только 20 наиболее популярных свойств товаров.

7. Дальнейшее уменьшение размерности датасета properties за счет признака value, - анализ показал, что 99% самых часто встречаемых значений свойств составляют 347 из 1327297, поэтому отброшено (1327297-347), - имеется в виду буквенно-цифровые коды, которые получены путем разделения параметров value по пробелу.

8. Дальнейшее уменьшение размерности датасета properties – подсчет частоты покупок каждого товара из датасета **events** и отбрасывание(из properties) рядов с товарами, которые никогда ни кем не были куплены (уменьшив т.о. с 13563669 до 1042036 записей).

9.Далее используем MultiLabelBinarizer для трансформации списка свойств товара в бинарную матрицу признаков, где каждый столбец соответствует конкретному свойству, а каждая строка представляет товар (itemid). Значение 1 говорит о наличии свойства у товара, а 0 — об отсутствии. После кодирования, мы агрегируем (используя .max()) бинарные признаки по каждому itemid, чтобы каждый товар представлялся одной строкой с перечнем своих свойств. Форма матрицы получилась (1042036 - товаров, 366 - бинарных признаков товаров).

10.Далее очень долговременная операция склеивания подготовленных датасета событий и датасета свойств товаров в единый датасет с учетом признака event\_datetime, по которому соединять максимально близкие по дате/времени события установления свойств товаров и события просмотра/корзина/покупки. merged\_data.csv.zip - итоговый упакованный датасет, готовый к использованию моделью, сохранен для дальнейшего использования.

11.Кодирование целевого столбца в 1/0 (есть событие transaction для товара или нет), отбрасывание нескольких рабочих столбцов, не несущих смысла для обучения (цифровые коды-идентификаторы), деление на train/test как 80/20 без перемешивания, чтобы сохранить хронологическую зависимость событий покупки/не покупки товаров на основе всех предыдущих событий и значений свойств. Обучение модели xgboost.

12.Расчет технической метрики Precision@3=precision\_at\_k().

13.Подготовка рабочей функции get\_top3\_recommendations()– определение рекомендованных товаров для клиента с заданным идентификатором на основе предыдущих действий клиента и связанных с этим событий и факторов товаров.

Далее на основе кода из нотбука построено приложение fast\_api, включающее модель gomodel.py и сервис api, использующий функции модели main.py.

/fast\_api

├── app

│ ├──templates

│ │ └── index.html

│ ├── \_\_init\_\_.py

│ ├── main.py

│ └── gomodel.py

└── inputdataset

└── merged\_data.csv.zip

Приложение завернуто в докер контейнер и готово к запуску.