Дисклеймер: в данном кейсе используются замаскированные «живые» данные настоящей задачи, при этом все названия в кейсе подобраны случайно, сохраняя смысл данных, все совпадения с реально существующими – случайны.

# Контекст

В далеком 2171 году ученые нашли древний артефакт, который помог решить одну из главных проблем всех Data Scientist-ов и аналитики в целом: отсутствие параллельных миров. Теперь любой Data Scientist может протестировать любую гипотезу на реальных данных в параллельной вселенной, не затрачивая ресурсы в нашей.

Параллельная вселенная выглядит примерно вот так:

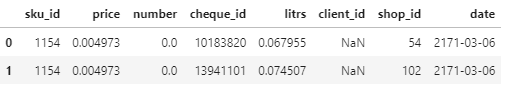


В этой вселенной некоторые характеристики перепутаны (например, реальное количество проданных товаров или их названия). Однако, все взаимосвязи в данных параллельного мира идентичны взаимосвязям из нашей вселенной.

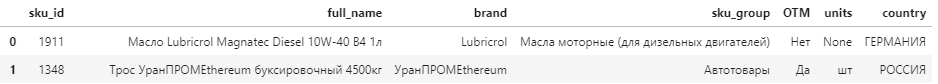
Вы работаете в **Организации**, управляющей сетью магазинов, на которых продается топливо для космических кораблей и шаттлов, а также различные другие товары. Перед вами ставят следующую аналитическую задачу:

# Исходные данные

# Транзакционные данные продаж – transactions



* sku\_id – уникальный идентификатор товара
* price – цена, по которой был продан товар
* number–количество товаров (если не топливо)
* cheque\_id – уникальный идентификатор чека
* litrs – количество литров (если товар - топливо)
* client\_id – уникальный идентификатор клиента (если клиент «представился» при покупке)
* shop\_id – уникальный идентификатор магазина
* date – дата транзакции

Данные о товарах – nomenclature  


* sku\_id – уникальный идентификатор товара
* full\_name–полное наименование товара
* brand–наименование торговой марки
* sku\_group – группа, к которой принадлежит товар
* OTM – признак собственной торговой марки
* units – единица измерения для количества
* country – страна производства товара

Формат всех источников - .parquet. Про чтение файлов в данном формате см. [здесь](#_Прочие_комментарии).

# Задача

Для увеличения продаж товаров из следующих групп:

* вода
* сладкие газированные напитки, холодный чай
* кофейные напитки с молоком
* энергетические напитки
* снеки
* соки и сокосодержащие напитки

вам необходимо разработать рекомендательную систему, которая будет предлагать покупателям 20 дополнительных товаров в чек.

**Пример**: покупатель приходит на кассу с 2-мя товарами: напитком «*Local-Cola»* и чипсами «*Sya'l*». Алгоритм должен предложить 20 товаров, которые пользователь вероятнее всего захочет добавить в свою корзину (в порядке убывания релевантности). На практике кассир предложит 1й по порядку товар, из имеющихся в наличии.

В качестве метрики качества рекомендаций используется **mean average precision at 20**, для понимания которой нужно освоить 2 промежуточные метрики:

**1) Precision at K** – количество релевантных элементов в рекомендации, если взять только верхние K рекомендованных товаров:

где принимает значение 1, если на -м месте стоит релевантный товар, и 0, если нет.

**2) Average precision at 20** – средний для всех от 1 до 20 в случаях, когда на -м месте порекомендован релевантный товар:

Например, если мы порекомендовали верно только первые два и последний товары, то метрика будет рассчитана следующим образом:

Итого, **Mean average precision at 20** – это средний для всех чеков тестовой выборки:

# Формат предоставления результатов

* **Оформленный репозиторий с кодом проекта** (в виде .zip файла или ссылки на github/аналог), удовлетворяющий следующим критериям:
  + воспроизводимый
  + содержащий Jupyter Notebook (или несколько) c визуализациями и комментариями к проделанным шагам

Язык решения – python, но если вы им не владеете, допускаются решения так же и на R.

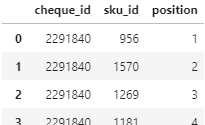
* **Рекомендации для чеков** из файла transactions-for\_submission. В каждом чеке (cheque\_id) пропущен 1 случайный товар, относящийся к какой-то из следующих групп (sku\_group):
  + вода
  + сладкие газированные напитки, холодный чай
  + кофейные напитки с молоком
  + энергетические напитки
  + снеки
  + соки и сокосодержащие напитки

Необходимо предоставить .parquet файл с результатами рекомендаций, содержащий поля:

* + cheque\_id – уникальный идентификатор чека
  + sku\_id–уникальный идентификатор товара
  + position **–** порядковый номер товара в предложенной рекомендации

Для каждого cheque\_id должно быть представлено 20 рекомендованных sku\_id.

В качестве примера в задании находится файл submission-example.parquet



Файл с результатами необходимо именовать в формате   
submission-*{surname\_name}*.parquet

* **Краткий отчет для Организации** в любом формате (pdf/ppt/doc и т.п.) о финальной модели и протестированных гипотезах. Также приветствуются дополнительные комментарии:
  + по имеющимся данным
  + чтобы вы сделали, имея больше ресурсов/времени/данных (каких?)

# Критерии оценки

При оценке работ будут учитываться:

* внимательность к данным
* методичность и последовательность выводов
* полнота и корректность решения
* аккуратность и читаемость кода
* информативность и понятность отчета

Значение метрики на тестовой части («submission») при проверке будет учитываться в последнюю очередь, но будет выполнять функции ориентира. Результаты будут опубликованы после сбора всех решений.

# Обратная связь и вопросы

Вопросы по кейсу пишите на платформе. Там же будут публиковаться ответы, о чем вы будете получать уведомления по email. В случае форс-мажора пишите на наш почтовый ящик Data Science:  
[ds\_drp@gazprom-neft.ru](mailto:ds_drp@gazprom-neft.ru)

# Прочие комментарии

Для чтения .parquet файлов вам понадобиться pyarrow

conda install -c conda-forge pyarrow

затем

pandas.read\_parquet(‘data.parquet’)

Удачи!

