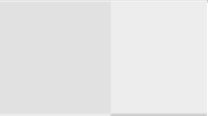
**Команда**

****

12. Предиктивная модель для рекомендации продуктов банка   
(кейс от Сбера)

**03 июня - 16 июня 2024 г.**

| Команда | Роли |
| --- | --- |
| Дмитрий Александрович Кожемяко | Капитан, DS developer |
| Иван Михайлович Черных | DS developer |
| Александр Андреевич Ермоленко | DS developer |
| Сергей Александрович Борисовский | Fullstack developer |
| Дарья Александровна Орлова | Data Analyst |

Куратор команды от ЛЦТ: Кристина (@krislct)

* Подробнее о фичах по гео, транзакциях и диалогам можно посмотреть здесь: [ссылка](https://docs.google.com/spreadsheets/d/1k-VmFNZEMRYDlroP6Bss-BNT-ZUdGs2ygX_Jym4cNqI/edit?gid=0#gid=0)
* Подробнее о ходе работ можно посмотреть здесь: [ссылка](https://docs.google.com/spreadsheets/d/1UEw8heslMy-hNOawX51f8UvLDqi5eMlRmSSIw45g0Fk/edit?gid=0#gid=0&range=A2:M26)

# **Постановка задачи**

Предсказание склонности клиента к покупке — важная задача для многих компаний, позволяющая повысить эффективность рекомендаций и конверсию в покупку.

При совместном использовании данных из различных источников можно составить более полный портрет клиента и более точные рекомендации.

Необходимо разработать мультимодальную модель, позволяющую прогнозировать выдачу продуктов в течение следующего месяца.

Обязательное условие: использование библиотеки PyTorch-LifeStream в решении задачи.

# **Общее описание решения**

## **Подход к решени**ю

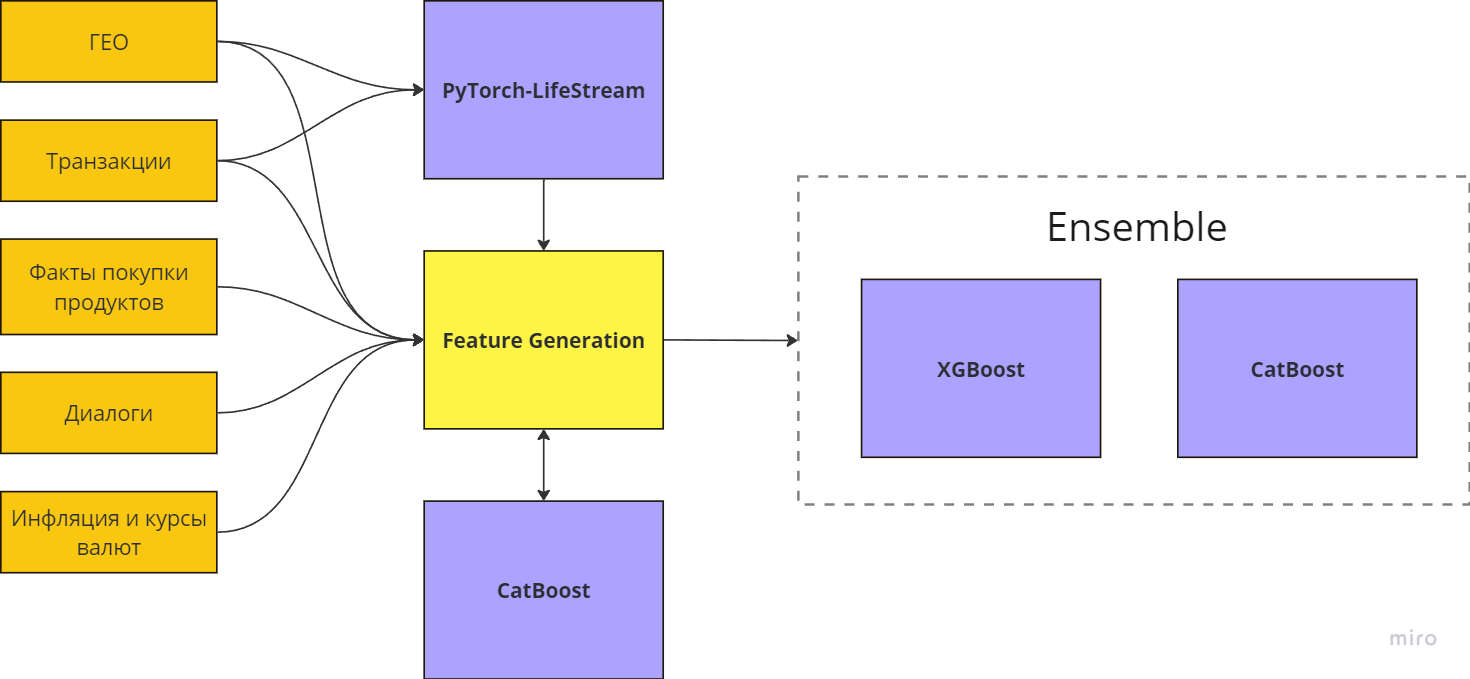


Рисунок 1 - Архитектура решения

Решение можно разделить на три основных этапа:

1. **Подготовка данных для генерации признаков:**

* снижение размерности данных, преобразование в удобные форматы;
* удаление/заполнение пропусков;
* препроцессинг для обучения Pytorch-LifeStream;
* подготовка внешних данных по курсам валют и инфляции;

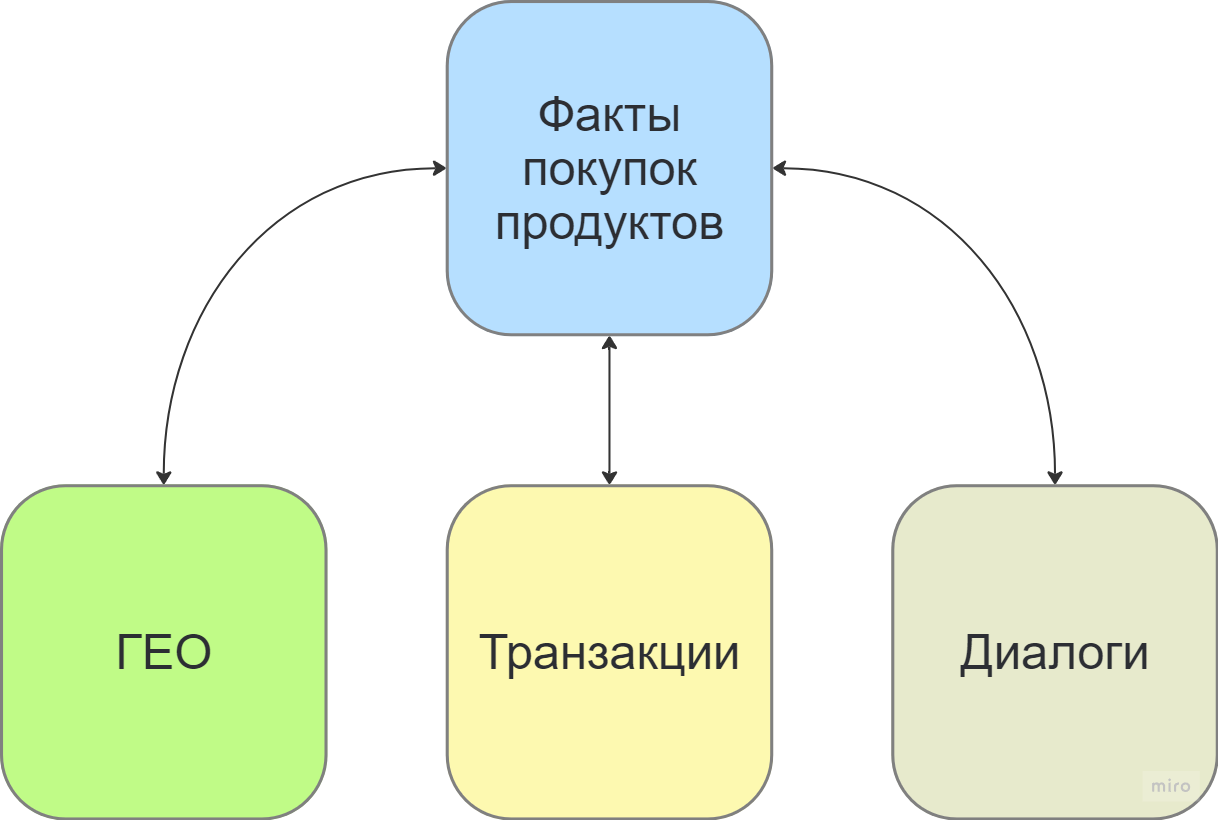
**2. Фичинженеринг:**

* обучение Pytorch-LifeStream для генерации эмбеддингов на основе сырых данных;
* генерация фичей от полученых эмбеддингов от Pytorch-LifeStream и полученных по диалогам (различные расстояния между векторами, кластеризация);
* генерация признаков на основе данных, различные агрегации и т.д.
* фильтрация данных,

**3. Создание и обучение модели:**

* создание ансамбля на основе Catboost и XGBoost с различными параметрами;
* подбор гиперпараметров при помощи Optuna для каждой модели;
* объединение результатов работы моделей при помощи взвешенного усреднения вероятностей;

**При построении мультимодальной архитектуры придерживались следующей концепции:**

1. В качестве базовой единицы мы выбрали пары клиент-месяц и все агрегаты данных строили на этой группировке. Мы обеспечили связь между данными Факты продаж-ГЕО, Факты продаж-Транзакции, Факты продаж-Диалоги.
2. Идеальный вариант представлен на схеме ниже.

Однако есть проблемы связности данных. Данные не всегда равносильны. 

Таблица 1 - Связность данных.

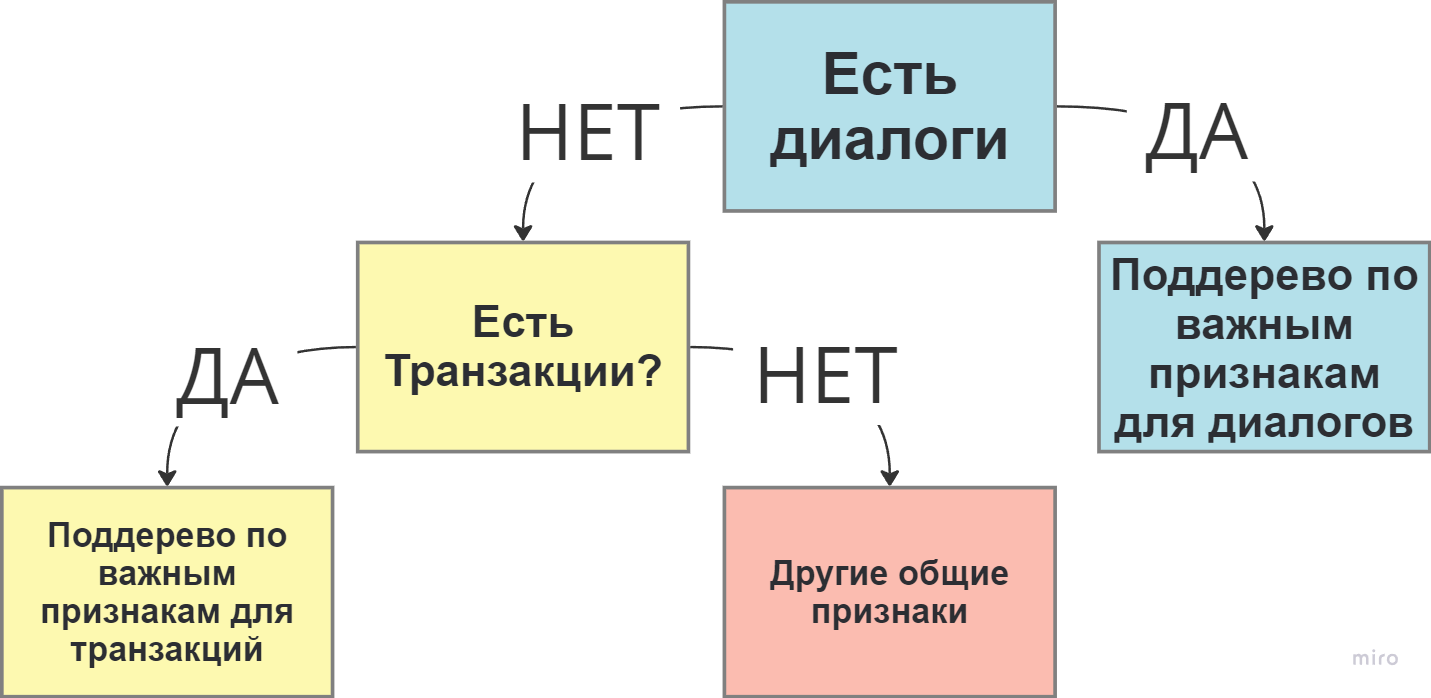
|  | Всего | Кто **не** покупал ничего | Кто покупал хоть что-то | Кто **не** покупал Продукт 1 | Кто покупал Продукт 1 | Кто **не** покупал Продукт 2 | Кто покупал Продукт 2 | Кто **не** покупал Продукт 3 | Кто покупал Продукт 3 | Кто **не** покупал Продукт 4 | Кто покупал Продукт 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Нет в geo и dlg и trx | 129477 | 129435 | 42 | 129455 | 22 | 129471 | 6 | 129463 | 14 | 129470 | 7 |
| Нет в geo и dlg | 501810 | 489815 | 11995 | 497135 | 4675 | 501075 | 735 | 496954 | 4856 | 498846 | 2964 |
| Нет в dlg и trx | 288201 | 288057 | 144 | 288131 | 70 | 288182 | 19 | 288157 | 44 | 288174 | 27 |
| Нет в geo и trx | 171524 | 171412 | 112 | 171452 | 72 | 171517 | 7 | 171499 | 25 | 171504 | 20 |
| Нет в dlg | 1031488 | 973621 | 57867 | 1009325 | 22163 | 1028152 | 3336 | 1007478 | 24010 | 1016208 | 15280 |
| Нет в geo | 650665 | 610078 | 40587 | 630174 | 20491 | 648693 | 1972 | 635912 | 14753 | 641812 | 8853 |
| Нет в trx | 358345 | 357945 | 400 | 358076 | 269 | 358321 | 24 | 358270 | 75 | 358280 | 65 |

Например, по данной таблице наличия/отсутствия видов данных, можно легко увидеть, что если у клиента нет транзакций, то практически всегда он не купит никакой продукт. Такой зависимости нет ни у диалогов, ни у геоданных.

Таким образом при мультимодальности необходимо учитывать уровень влияния каждого вида данных.

Важно отметить, что для мультимодальных данных хорошо подходят решающие деревья. Так например, нам даны модальности диалоги, гео, транзакции и модель должна строить на их основе прогноз. Если добавить простую фичу например, говорящую о наличии у клиента данных о диалоге “да или нет”, тогда модель может построить разные варианты деревьев в случае когда есть такая информация и в случаях когда нет.

Данное особенность деревьев дает преимущество для мультимодальных задач. Однако если подумать данную идею можно расширить и для линейных моделей.

Отдельно стоит обратить внимание на мультилейбловую сторону задачи, в которой необходимо сделать предсказание по четырем таргетам.

Отличной техникой является создание промежуточной модели, которая делает предсказание каждого таргета через кросс-валидацию. Т.е. предсказания покупки продуктов 1 2 3 можно использовать при прогнозировании покупки продукта 4. Таким образом мы можем заполнить предсказания во всех данных без риска лика данных. Жизненно пользу данного подхода возможно описать следующим примером, что некоторые продукты могут дополнять друг друга, либо это может быть показателем общей лояльности клиента к приобретению продуктов.

В качестве развития глобально правильней было бы обучать векторное представление по всем данным сразу плюс с фичами, однако мощности не позволяют такое провернуть. Также необходимо подумать о том что данные можно более тесно связывать между собой, например частота диалогов относительно частоты транзакций и т.д. Однако время и мощности не позволяют реализовать все идеи.

* 1. **Проблема больших данных**

Организаторами предоставлен довольно большой датасет > 15 Гб в сжатом виде (parquet).

Первая и основная проблема с которой столкнулись: как обрабатывать и рассчитывать фичи под такой объем данных. Довольно много времени заняло изучение данных и выбор способов обработки.

В итоге для обучения эмбеддингов на PyTorch-LifeStream использовали PySpark. А для генерации фичей произвели сэмплирование данных таким образом, чтобы учесть все случаи, когда клиенты покупали продукты, и было множество примеров, когда не покупали. Т.к. дисбаланс в данных чрезмерно большой, то получилось произвести “безболезненное” исключение негативных примеров.

* 1. **Структура проекта**

**Препроцессинг:**

* Preprocessor\_geo.ipynb - препроцессинг на PySpark для данных по геохешей.
* Preprocessor\_trx.ipynb - препроцессинг на PySpark для данных по транзакцияи.
* Train\_emb\_new.ipynb - генерация эмбеддингов транзакций при помощи Pytorch-LifeStream.
* Train\_emb\_geo.ipynb - генерация эмбеддингов геохешей при помощи Pytorch-LifeStream.
* dialog\_preprocessing.ipynb - генератор фичей на основе эмбеддингов.
* compression\_data.py - модуль по сжатию данных без потери данных (за счет умного изменения типа данных для каждой колонки).
* gen\_sample\_pairs\_Client\_Month.ipynb - кастомный сэмплер для оптимальной выборки пар клиент-месяц. (см. раздел Дисбаланс).
* create\_normalize\_trx.ipynb - нормализация цен транзакций за счет использования открытых данных ЦБ России по инфляции и курсам валют. (см. раздел Внешние источники данных).
* calc\_good\_bad\_cos\_sim\_v2.ipynb - модуль расчета средних "хороших" (когда покупали) и "плохих" (когда не покупали) векторов и расчет расстояния рассматриваемых объектов до "хороших"/"плохих" примеров.
* create\_geo\_features.py - модуль по формированию гео-фичей. Формируются фичи относительно геохешей уровней 4/5/6, включая фичи: популярности геохешей у клиентов, покупающих продукты, количество уникальных пользователей геохешей, количество транзакций в рамках геохеша и т.д.
* create\_agg\_geo\_v7\_1.ipynb - генератор расширенных фичей на основе ГЕО.
* create\_agg\_dialog\_v7\_2.ipynb - генератор расширенных фичей на основе диалогов.
* create\_agg\_trx\_v7\_1.ipynb - генератор расширенных фичей на основе транзакций.
* create\_agg\_ptls\_v7\_1.ipynb - генератор расширенных фичей на основе эмбеддингов.
* create\_agg\_target\_v2.ipynb - генератор расширенных фичей на основе таргетов.

**Модeль:**

* Optuna\_Ansamble.ipynb - ансамблевый метод с использованием Optuna.
* baseline\_v1\_8\_7.ipynb - кросс-валидация и объединение всех фичей.

**Данные дополнительные:**

* client\_without\_dlg.csv - список клиентов, у которых нет диалогов.
* client\_without\_geo.csv - список клиентов, у которых нет гео-данных.
* client\_without\_trx.csv - список клиентов, у которых нет транзакций.
* list\_uniq\_client\_target.csv - список уникальных клиентов.
* esult\_sample\_Client\_Month\_df\_12\_06\_2024.parquet - расширенный сэмпл клиентов используемый для генерации эбеддингов.

**Ссылка для** [**скачивания**](https://disk.yandex.ru/d/D6HiwFPZa9LEOg) **данных**

**Установите необходимые зависимости:**

* pip install requirements.txt

**Используйте Jupyter notebooks для генерации и обработки фичей:**

* jupyter notebook Preprocessor\_geo.ipynb
* jupyter notebook Preprocessor\_trx.ipynb
* jupyter notebook Train\_emb\_new.ipynb
* jupyter notebook Train\_emb\_geo.ipynb
* jupyter notebook dialog\_preprocessing.ipynb
* jupyter notebook gen\_sample\_pairs\_Client\_Month.ipynb
* jupyter notebook create\_normalize\_trx.ipynb
* jupyter notebook calc\_good\_bad\_cos\_sim\_v2.ipynb
* jupyter notebook create\_geo\_features.py
* jupyter notebook create\_agg\_geo\_v7\_1.ipynb
* jupyter notebook create\_agg\_dialog\_v7\_2.ipynb
* jupyter notebook create\_agg\_trx\_v7\_1.ipynb
* jupyter notebook create\_agg\_ptls\_v7\_1.ipynb
* jupyter notebook create\_agg\_target\_v2.ipynb

**Используйте Jupyter notebooks для обучения модели:**

* jupyter notebook Optuna\_Ansamble.ipynb
* jupyter notebook baseline\_v1\_8\_7.ipynb

Весь код доступен на GitHub по ссылке: [GitHub](https://github.com/DimkKozhem/Multimodel)

# **EDA**

* 1. **Полнота данных**

Поставленной задачей требовалось предсказать покупку 4 продуктов для 140 488 клиентов банка, из них:

* транзакции - 94 173 клиента (67%);
* гео-информация - 72 509 клиентов(52%);
* диалоги - 44 439 клиентов (32%);
* транзакции, гео-информация и диалоги - 111 161 пользователей (79%).  
  1. **Дубликаты в данных**

Выявлено наличие дубликатов в тестовых данных test\_target\_b. По одному клиенту в один отчетный месяц есть несколько записей таргета, таких записей 31 691.

Найденные пары клиент-отчет не содержали фактов покупки товаров, поэтому они были исключены из этих данных.

* 1. **Геоданные**

В среднем о клиенте 5 раз в месяц есть метки по геоданным (чаще всего встречается 3 разных геохеша уровня 4). Есть предположение, что в гео-данных содержат информацию о входе одно из мобильных приложений Сбера. Однако не исключено, что там используется и иная геоинформация.

Важно, что если это мобильное приложение Банка, то довольно логично делать связку гео-данные-транзакции-диалогами со службой поддержки.

* 1. **Транзакции**

Анализ полей транзакций:

* “event\_type” - имеет 56 уникальных значений отнесена к категориальному признаку;
* поле “event\_subtype” имеет 58 уникальных значений отнесена к категориальному признаку;
* поле “currency” имеет 11 уникальных значений отнесена к категориальному признаку
* поле “src\_type11” имеет 80 уникальных значений отнесена к категориальному признаку;
* поле “src\_type12” имеет 364 уникальных значений отнесена к категориальному признаку;
* поле “dst\_type11” имеет 86 уникальных значений отнесена к категориальному признаку;
* поле “dst\_type12” имеет 432 уникальных значений отнесена к категориальному признаку;

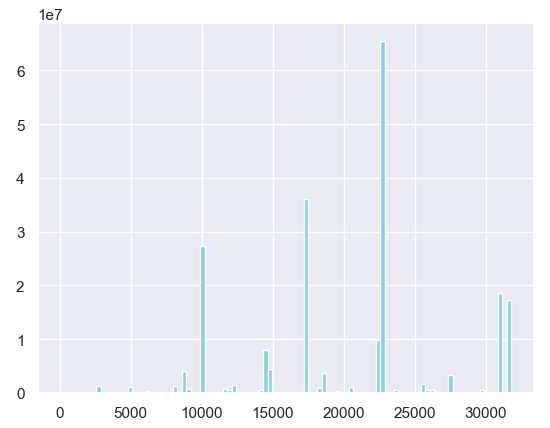


Рисунок 2 - распределений значений “dst\_type12”.

* поле “src\_type21” имеет 36246 уникальных значений отнесена к непрерывному признаку/в pytorch-lifestream к категориальному;

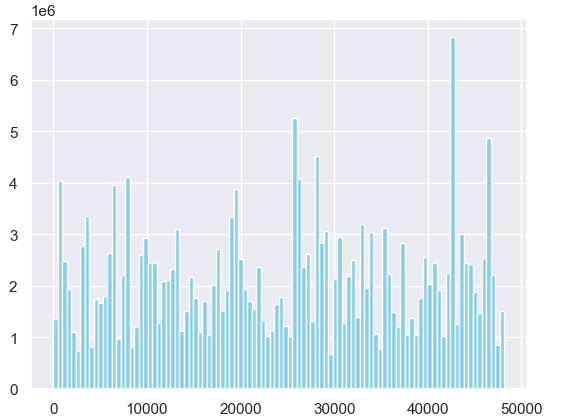


Рисунок 3 - распределений значений “src\_type21”.

* поле “src\_type22” имеет 88 уникальных значений отнесена к категориальному признаку;
* поле “src\_type31” имеет 2317 уникальных значений отнесена к непрерывному признаку/в pytorch-lifestream к категориальному;

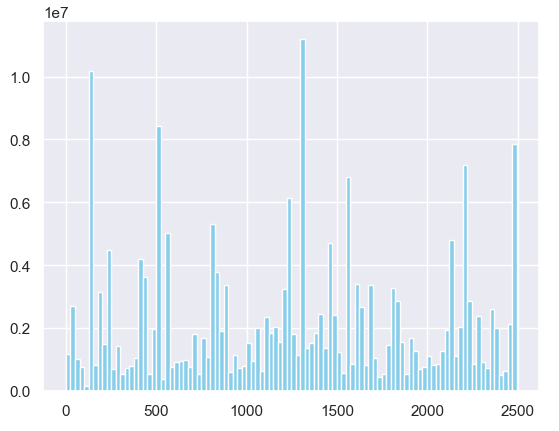


Рисунок 4 - распределений значений “src\_type31”.

* поле “src\_type32” имеет 89 уникальных значений отнесена к непрерывному признаку

В итоге, анализ транзакций показал, что в данных есть очень много категорий, для которых можно сделать фичи на основе популярности категорий использованных при покупки продукта 1/2/3/4.

* 1. **Таргет**

В таблице ниже содержится информация об общем количестве приобретенных продуктов всеми клиентами в каждом месяце.

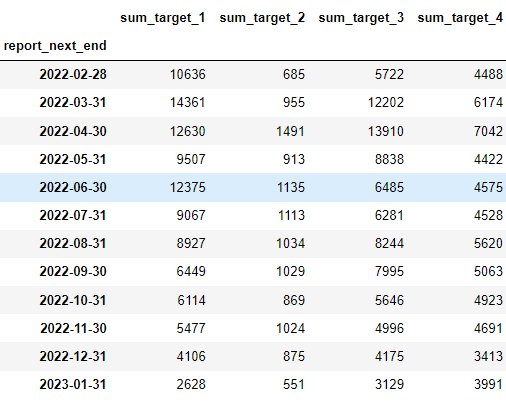
Анализ популярности продуктов:

* продукт 1 - наиболее часто покупали в марте, апреле и июне;
* продукт 2 - пользовался наибольшим спросом в апреле;
* продукт 3 и 4 - активно покупали в марте и апреле.

Продукт 1 в начале 2022 года был самым популярным среди всех других продуктов, а к концу года таргет 1 показывает значительное отток покупателей, и самыми стабильными в количестве покупателей продукт 3 и продукт 4.

Предполагается, что повышенный интерес к продуктам в определенные месяцы может быть связан с сезонностью продукта или с проводимыми промо акциями банка.

Таблица 2 - Информация о приобретенных продуктов по месяцам.



* 1. **Внешние данные**

В исходных данных представлены сумма транзакций и типы валют. Данные для обучения используются за целый год, поэтому мы учитываю инфляция. Также при наличии транзакций по нескольким различным валютам несправедливо работать как с одной, необходимо произвести нормализацию к одному виду, например к рублям.

Среди внешних источников данных были использованы:

* Открытые данные ЦБ РФ по инфляции

График инфляции и ключ.ставки за каждый месяц доступны по ссылке: <https://www.cbr.ru/hd_base/infl/>

Данные представлены за каждый месяц. На основе данных ЦБ по инфляции для каждого периода рассчитано значение в виде кумулятивного процента (накопительный процент с учетом прошлых месяцев).

Далее составлен словарь для каждого месяца с указанием на сколько надо умножить сумму транзакции, чтобы привести все к текущим значениям. И данная операция была проведена над всеми данными.

* Открытые данные ЦБ РФ по курсам валют  
  Курсы валют за каждый день доступны по ссылке: <https://www.cbr.ru/currency_base/dynamics/>

Валюты в данных представлены в зашифрованном виде

11.0 167351850

1.0 47897

14.0 39889

7.0 21909

17.0 2115

9.0 2074

….

Можно привести самые первые транзакции к рублям, если сделать предположение, что чаще всего используются рубли (11), далее это доллары (1) и евро (14), далее другая валюта. Другими популярными валютами являются, например, белорусские рубли и юани. Кроме долларов и евро, курс у остальных валют колеблется в районе 20 руб.

Т.к. данных довольно много, то было применено простое приведение сумм к рублям:

* рубли (код 11), оставлен без изменения;
* валюты с кодами 1 и 14 были умножены на среднее значение доллара и евро за указанный период 2022 год это ~70 рублей;
* остальная валюта умножена на значение 20.

В итоге нормализованные цены также попадали в топ важных фичей по каждому продукту.

* 1. **Особенности тестовых данных**

Предсказание необходимо сделать на следующие три месяца:

* 2022-11-30 - 46 086 клиентов;
* 2022-12-30 - 45 525 клиентов;
* 2023-01-31 - 8 877 клиентов.

Данную особенность необходимо учитывать при обучении модели. Т.е. например пары клиент-месяц в конце года для обучения модели более важны, чем пары вначале года. Это отдельно учитывается в сэмплере данных.

* 1. **Распределение клиентов по данным**

Не для всех клиентов есть все данные по геохешам, диалогам и транзакциям. По наличии данных в разрезе клиентов можно посмотреть в таблице ниже.

Таблица 3 - Информация о клиентах

|  | **geo\_clients** | **dialog\_clients** | **target\_clients** | **trx\_clients** | **submitе\_clients** |
| --- | --- | --- | --- | --- | --- |
| **geo\_clients** | 789051 | 259373 | 625392 | 602230 | 72509 |
| **dialog\_clients** | 259373 | 408228 | 311143 | 338084 | 44439 |
| **target\_clients** | 625392 | 311143 | 994380 | 715400 | 140488 |
| **trx\_clients** | 602230 | 338084 | 715400 | 1081371 | 94173 |
| **submitе\_clients** | 72509 | 44439 | 140488 | 94173 | 140488 |

У каждого клиента встречается хотя бы два типа данных:

- гео + транзакции включают всех клиентов из диалогов;

- диалоги + транзакции включают всех клиентов из гео;

- гео + диалоги включают всех клиентов из транзакций.

К тому же есть 129 477 клиентов, у которых есть данные о купленных продуктах (таргет), но нет ни геохешей, ни диалогов и ни транзакций.

В сабмите есть 29 327 клиентов у которых также нет геохешей, диалогов и транзакций.

И эти 29 327 есть только в фактах продаж.

Таким образом можно сделать выводы:

1. Делать подмодели для каждого типа данных не имеет смысла (т.к. нет таких клиентов, которые встречаются только в геохешах, диалогах или транзакциях).
2. Ключевым источником признаков для модели должны быть факты ранее совершенных покупок, т.е. все фичи должны строиться на базе фактов продаж.
3. Имеет смысл делать эмбеддинговое представление фичей по данным гео+транзакции+диалоги+факты покупок, в данном случае PyTorch-LifeStream как нельзя кстати.
   1. **Распределение таргетов по клиентам**

Таблица 4 - Распределение таргетов.

|  | **geo\_clients** | **dialog\_clients** | **target\_clients** | **trx\_clients** | **submitе\_clients** |
| --- | --- | --- | --- | --- | --- |
| **target 1** | 72557 | 70885 | 93048 | 92779 | 17451 |
| **target 2** | 8084 | 6720 | 10056 | 10032 | 1803 |
| **target 3** | 65728 | 56471 | 80481 | 80406 | 14812 |
| **target 4** | 39319 | 32892 | 48172 | 48107 | 8670 |

По Продукту 2 имеет меньше всего примеров покупки.

# **Преобразования данных**

* 1. **Заполнение пропусков**

Помимо обычного пропуска данных необходимо учитывать отсутствие в принципе какого-то вида данных у пользователя.

В используемых мультимодальных данных встречаются такие ситуации, когда есть информация о транзакциях, но при этом нет данных о диалогах или геоданных.

Самый оптимальный способ заполнения это нулями, он оптимально подходит к исходным данным, когда по сути может быть ситуация, что данных нет в принципе. При заполнении данных например медианной, надо быть аккуратнее и следить за распределением как обучающей выборке так и в тестовой (сабмит). Учитывая большое количество сложных фичей и мультимодальность, то приоритетным способом заполнения остается заполнение нулями.

* 1. **Большие данные. Решение проблемы**

Исходный датасет даже в сжатом виде (parquet) занимает более 15 Гб. В связи с чем возникает проблема больших данных.

Примененные техники уменьшения размерности данных:

* исключение с малым количеством транзакций;
* использовать PySpark при препроцессинги данных;
* итерационное исключение “неважных” фичей;
* использование Polars;
* заменить тип данных id клиента на “category”;
* заменить float64 на float32;
* фильтрация клиентов (оптимальный отбор объектов для обучения)
  1. **Предобработка данных**

В решении использовали несколько различных подходов и моделей, под которые сделали различные преобразования данных.

1. Расчет агрегатов: расчет осуществлялся по агрегации за прошедший месяц.
2. С помощью внешних источников данных:

* проведена нормализация цен относительно инфляции и по курсу валют;
* все транзакций были приведены к рублям.

1. Далее возникает дилемма: использовать ли для прогнозирования только последний месяц или весь период до текущего репорта. С одной стороны, для точности лучше использовать весь период, однако будет получаться, что для более ранних таргет-отчетов у нас мало данных за период, а максимум данных будет для более поздних таргет-отчетов. С другой стороны, можно принять, что горизонт событий у всех – это 1-2 месяца, т.е. что было вначале года, не так сильно влияет на поведение клиента в конце года. Также гораздо проще считать все фичи по месяцам. Поэтому принято решение использовать помесячные данные.
2. Для транзакций есть категориальные данные. В целях создания дополнительных фичей, была произведена агрегация транзакций по популярным значениям категории.

Таблица 5 - Популярные категории.

| **Категориальный признак** | **Популярные значения внутри категории** | **% Покрытия транзакций** |
| --- | --- | --- |
| **event\_type** | [54, 38, 37, 51, 25, ] | 79,6% |
| **event\_subtype** | [55, 49, 18, 29, 47, ] | 79,1% |
| **src\_type11** | [19, 149] | 86,7% |
| **src\_type12** | [344, 902, 1081] | 78,9% |
| **dst\_type11** | [364, 869, 433, 1166, 988, 852, ] | 89,8% |
| **dst\_type12** | [22652, 17340, 10049, 30836, 31488, ] | 74,5% |
| **src\_type22** | [70, 41, 85, ] | 28,7% |
| **src\_type32** | [81, 26, 4, 17, 25, 67, ] | 47,0% |

Популярные значения категорий покрывают от 30% до 90% всех транзакций.

По итогам экспериментов выявлено, что фичи, построенные на базе популярных типов транзакций, дают наибольший прирост. Точнее 2/3 самых важных фичей для всех 4-х таргетов, – это фичи, построенные на данных агрегатах по типам транзакций.

Вследствие чего список популярных категорий был расширен.

Таблица 6 - Расширенные популярные категории.

| **Категориальный признак** | **Популярные значения внутри категории** | **% Покрытия транзакций** |
| --- | --- | --- |
| **event\_type** | [54, 38, 37, 51, 25, 40, 56, 41, 52, 16, 17, 26, 44, 36, 15, 9 , 48, 5 , 45, ] | 95,6% |
| **event\_subtype** | [55, 49, 18, 29, 47, 51, 9, 10, 12, 16, 19, 8, 14, 50, 22, 33, 43, 56, 2, 1,] | 95,9% |
| **src\_type11** | [19, 149, 72, 70, 128, 122, 1, 180, 0,] | 98,5% |
| **src\_type12** | [344, 902, 1081, 370, 201, 45, 682, 456, 451, 189, 1013, 93, 1019, 0, 429, 959, ] | 95,3% |
| **dst\_type11** | [364, 869, 433, 1166, 988, 852, 1302, 813, 308, 0, 592, 1171, 1121, ] | 98,1% |
| **dst\_type12** | [22652, 17340, 10049, 30836, 31488, 22548, 14606, 14906, 8693, 18723, 17444, 27453, 25683, 0, 12409, 2898, 7985, 5086, 20641,] | 95,5% |
| **src\_type22** | [70, 41, 85, 87, 48, 56, 20, 58, 51, 60, 21, 31, 26, ] | 54,4% |
| **src\_type32** | [81, 26, 4, 17, 25, 67, 77, 44, 76, 21, 24, 61, 51, 82, 55, 74, 16, ] | 74,8% |
| **src\_type21** | [42507, 25721, 19370, 6272, 8099, 13013, 32995, 25714, 29127, 28251, 26311, 12863, 17316, 28887, ] | 11,0% |
| **src\_type31** | [148, 1323, 1235, 1553, 2128, 2486, 500, 814, 409, 515, 242, 2202, 439, 1835, 844, 568, 189,] | 32,7% |

* 1. **Контроль лика данных**

Так как в задаче присутствуют временные ряды, то важно обеспечить отсутствие лика данных. Контроль лика данных осуществлялся на всех этапах работы с данными.

# **Генерация фичей**

Для бустинга фичи генерировались для пар “клиент-месяц”. По сгенерированным эмбедингам PyTorch-LifeStream, помимо прямого использования в качестве отдельной фичи, также рассчитывались статистики. В итоге, фичей получилось более 3000. Селекция фичей проводилась следующим образом - если для всех таргетов у фичи важность таргета меньше порога 0.1 то она удалялась.

* 1. **Pytorch-LifeStream**

В качестве входной точки был использован бейзлайн от организаторов. Генерацию выполняли для данных транзакций и геохешей. Для разработки решения использовали документацию PyTorch-LifeStream из репозитории [GitHub](https://github.com/dllllb/pytorch-lifestream).

Препроцессинг предложенный в бейзлане нам не подошёл из-за большого объема данных, воспользовались решением на основе PySpark. В качестве базовой модели выбрана модель CoLES, так как она неплохо зарекомендовало себя на похожих соревнований.

На итог мы обучили по две модели для каждого типа данных. Модели отличались между собой по количеству параметров используемых для генерации эмбеддингов. Размер обученных эмбеддингов был длинной 256 и 32.

Предложенная схема инференса в базовом решении нам не подошла из-за большего требования к памяти и времени обработки данных. Для решения задачи было решено реализовать свою функцию фильтрации данных по месяцам, что позволило значительно увеличить скорость генерации эмбеддингов для целевого датасета.

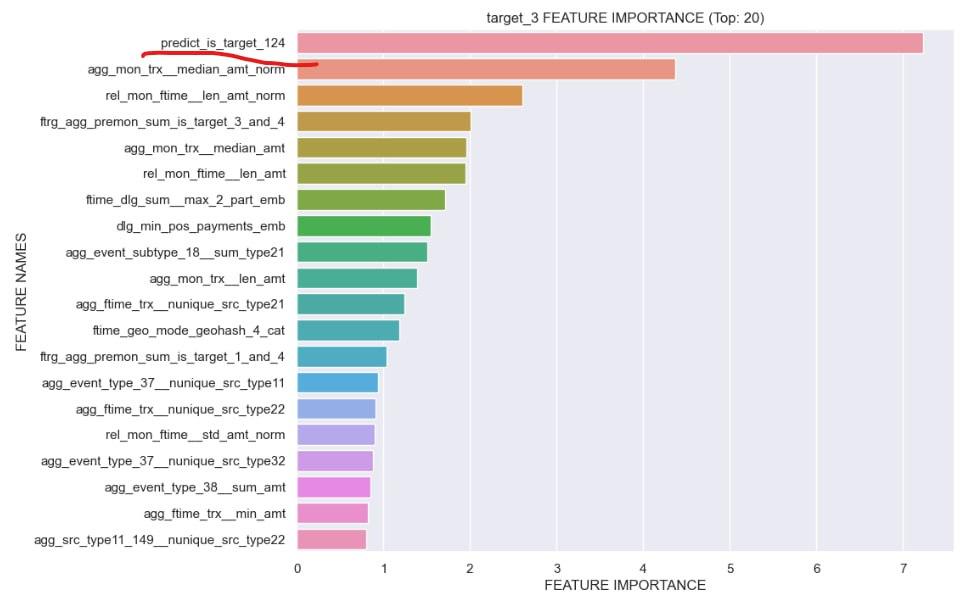
Для сгенерированных эмбеддингов были замерены косинусное, евклидово и манхентовское расстояния до “эталонного эмбеддинга”. “Эталонный эмбеддинг” был определен как средний эмбеддинг для каждого таргета когда была покупка и когда не было покупки. Также эмбеддинги были кластеризованы при помощи КMeans.

* 1. **Фичинженеринг на предоставленных данных**

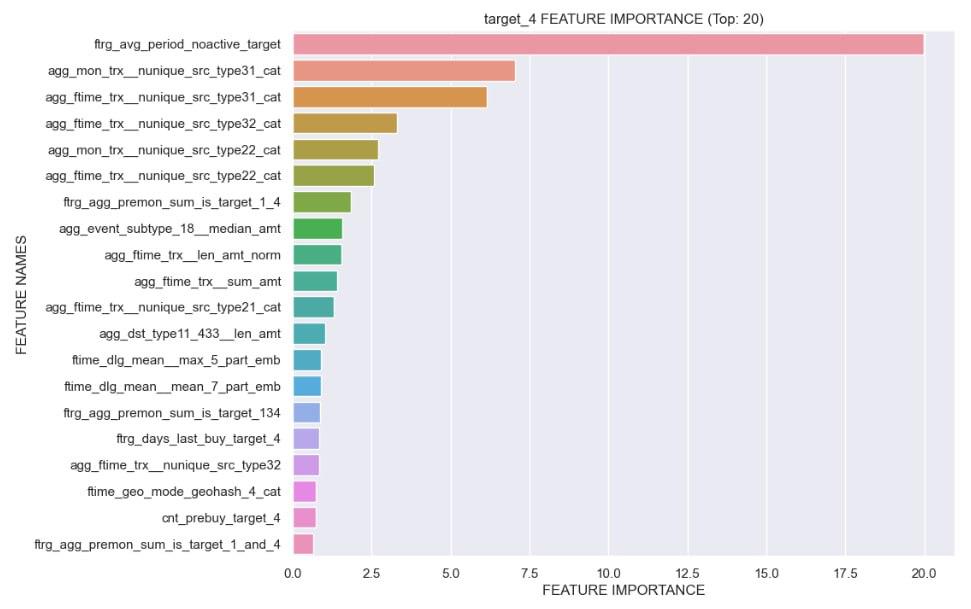
Основные по важности фичи по результатам тестирования моделей:

* предсказание покупки соседних товаров (получено с помощью Cross-validation);
* типы транзакций клиента (в агрегированном виде как по прошлому месяцу так и за весь период);
* факт наличия каждого вида данных;
* эмбеддинги построенные на базе PyTorch-LifeStream (Особенно для Продукта 1 и для Продукта 3, где они входит в топ 7);
* признаки “неактивности” клиента (сколько по времени не совершались действия)
* статистика по эмбеддингам разговоров (для Продукта 1 особенно);
* агрегаты на базе геоданных.
  1. **Важные фичи**

1. Предсказания других таргетов дает заметный прирост. То есть для обучения модели например, для таргета 3, использовались предсказания таргетов 1/2/4. И так по всем 4м таргетам. Для таргетов 1,3,4 это фича попадает в топ-20 по важности.

Рисунок 4 - Распределений важных фичей для таргета 1,3,4.

2. На важных фичей для Продукта 1 самая важная фича – это период неактивности по покупкам. Т.е. среднее время, когда пользователь не покупал никакие продукты, например, 3 месяца ничего не покупал. Это период неактивности.

Рисунок 4 - Распределений важных фичей для таргета 1.

1. **Что пробовали и не получилось**

**При решении задачи не успели реализовать:**

* получить максимальный скор на эмбеддингах PyTorch-LifeStream без подключения дополнительных фичей;
* построить эмбеддинг для истории покупок на основе PyTorch-LifeStream;
* построить связи между геохешами из-за особенности предоставленных данных;
* сделать фильтрацию клиентов для обучения под каждый таргет;
* использование кластеров клиентов для прогнозирования, построение для каждого кластера свой отдельной мультимодели.

**При решении задачи не сработало:**

* при попытке уменьшить объем данных пробовали перевод категорий из типа object в тип category. При работе с данными в памяти действительно объем уменьшался где-то на 50-100 Мб, но при сохранении, например, транзакций, объем паркет файла увеличивался с 3 Гб до 17 Гб. В итоге отказались от этого подхода.
* из-за большого объема использование некоторых under сэмплингов данных, типа TomekLinks, слишком затратно по времени.
* для работы с большим количеством данных использовали Polars, он хоть и давал скорость, но не ту, которая требовалась. В итоге самое лучшее решение было предварительно сэмплировать данные.

1. **Результаты экспериментов**

**Тут таблица экспериментов**

1. **Итоговый результат на лидерборде**

**ТУТ скрин с DSWORK**

Подвал!!!!

При решении задачи придерживались следующего алгоритма:

1. Сформировали статистики по каждой сущности.
   1. Определили “слабые” и “сильные” стороны данных;
   2. Посчитали отношения вида данных к клиентам и фактам продаж.
2. В качестве объекта для предсказания использовали пару «клиент — отчётный месяц».
3. Проблему с большим объемам данных решали различными методами, в итоговом решении использовали сэмплирование пар «клиент — отчётный месяц»(кастомный сэмплинг).
4. Далее все фичи и эмбеддинги рассчитывали только для сэмплированного набора данных, что значительно упростило и ускорило процесс подготовки данных.
5. Библиотеку Pytorch-LifeStream для формирования эмбеддингов пар «клиент — отчётный месяц» для датасета транзакций и геоаналитики.
6. Проведена большая работа по формированию дополнительных фичей. Процесс формирования фичей отражен в отдельном разделе. Отдельный набор фичей формировался на базе эмбеддингов, построенных с помощью Pytorch-LifeStream.
7. В качестве финальных моделей использован ансамбль моделей.
8. Также на каждом этапе проводилась кросс-валидация на бустинг-модели, для формирования дополнительных признаков в качестве предсказаний по таргетам, а также для проверки стабильности работы модели на всех фолдах (обобщающая способность модели).
9. Для подбора гиперпараметров ансамбля моделей использовали библиотеку Optuna.
10. Качества модели выполнения на основе нескольких метрик.
    1. **Гео фичи**

| № | Титл | Описание | Статус |
| --- | --- | --- | --- |
| 1 | Сгенерировать доп хэши разных уровней | Попробовать сагрегировать более верхнеуровневый хэш уровня 3, либо наборот более низкий 7 | На подумать |
| 2 | Связать Гео-адрес с конкретными транзакциями (через клиента и дату-время) | ГЕО. Можно принять такое допущение, что клиенты из более дорогих районов предпочитают одни продукты (типа вклады), а клиенты из "дешевых" районов предпочитают другие продукты (например кредиты).  Связать ГЕО адреса с транзакциями (через клиента и дату-время) | В работу |
| 3 | Связать Гео-адрес с общими транзакциями клиента (только через клиента без конкретной даты ) | Связать ГЕО адреса с транзакциями без конкретной даты и типа транзакции, просто чтобы определить гео-терминал в дорогом районе или нет (через клиента). Надо рассчитать для каждого "гео адреса" суммы из транзакций по клиентам (распределять просто взвешено) | В работу |
| 4 | Рассчитать частоту каждого геохэша - общую | Надо сделать для каждого гео адреса частоту транзакций (тоже взвешанно по клиентам).  Считаем общее кол-во транзакций осуществленных на каждом терминале за каждый месяц | Сделано |
| 5 | Рассчитать частоту каждого геохэша - по каждому клиенту |  | Сделано |
| 6 | Связь гео и диалогов | Рассчитать средний эмбеддинг по диалогам клиента. тоже сделать связку гео-адрес и средний тип диалогов | В работу |
| 7 | Частота преобретения клиентом продуктов пользующихся определеным банкоматом | Т.е. если 50% клиентов пользуясь определнным терминалом в итоге преобретали продукт 1, тогда для этого банкомата для продукта 1 будет значение 0.5 | Сделано |
| 8 | Популярные Топ-1 геохеши у клиентов | Расчет топ-1 популярных хешей для каждого клиента, и также расчет процента посещения этих топ-1 относительно всех посещаемых геохешей (по аналогии с софтмакс) | Сделано |
| 9 | Популярные Топ-5 геохеши у клиентов | Расчет топ-5 популярных хешей для каждого клиента, и также расчет процента посещения этих топ-5 относительно всех посещаемых геохешей (по аналогии с софтмакс) | Сделано |
| 10 | Связка клиента и используемые им геохеши в отчетный период | Связка клиента и используемые им банкоматы в отчетный период. С указанием: частоты использования каждого банкомата | Сделано |
| 11 | Близость геохешей | Можно посчтиать условную близость сравнивая геохешей по разным уровням, за счет вложенности одного уровня в другой | В работу |
| 12 | Биннанизация хешкодов | Подразумевается, что близкие хэш-коды лежат рядом друг с другом и это поможет отделить города районы и прочее | В работу ! |
| 13 | Определить к чему относиться гео событие к звонку в поддержку или к транзакции | Если человек заходил в Сбер приложение, но не оплаичвал, значит он либо общался с поддержкой либо просто провреял баланс | В работу |
| 14 | Посчитать для каждого геохеша кол-во транзакций и объекм в это же время |  | В работу |
| 15 | Посчитать для каждого геохеша кол-во диалогов в это время |  | В работу |
| 16 | Кол-во месяцев активности клиента |  | Сделано |

* 1. **Фичи по транзакциям**

| Транзакции | Титл | **Описание** | Статус |
| --- | --- | --- | --- |
| 1 | Накопительная сумма | Считаем, что если есть клиенты которые кладут средства через банкомат, то есть и те которые получаеют их переводом (например ЗП) | На подумать |
| 2 | Нормализция цен относительно инфляции | Данные предоставлены за год, значит есть влияние инфляции.  Можно помочь модели, если мы нормализцем цены относительно инфляции  На сайте ЦБ есть сведения об инфляции. | Сделано |
| 3 | Нормализция цен относительно валют | В транзакциях представлены суммы в различных валютах  Можно привести их к рублям, если сделать такое предположение, что чаще всего используются рубли (11), далее это доллары (1) и евро (14), далее другая валюта самой популярной другой валютой являются белорусские рубли и юани и другие валюты, кроме долларов и евро курс у остальных колеблиться в районе 20 руб  Курс валют доступен на сайте ЦБ | Сделано |
| 4 | Временные фичи | Определить по верменам транзакций параметры: Дни недели, выходные или будни, часы вечерние утренние дневные | В работу |
| 5 | Агрегация числовых значнеий по клиенту за период | Например сумма и кол-во транзкаций за месяц по клиенту + дополнительные виды агрегаций, медиана, дисперсия и прочее. Также не только по сумме транзакции, но и по другим числовым признакам | Сделано |
| 6 | Агрегация числовых значнеий по клиенту за период по популярным значениям категорий | Например сумма транзкаций за месяц по клиенту для категории CAT1 для значения ITEM1.  ITEM1 выбирается как самые частые значения категорий.  Например для категории "event\_subtype" агрегации будут считаться для значений категорий: 55, 49, 18, 29, 47 (т.к. самые частые категории) | Сделано |
| 7 | Определение популярной валюты | Через currency, либо это рубли (категория "11") либо другая валюта  Определение популярной currency у клиента currency = 11 - рубли, остальное не рубли.  Можно посчитать долю рублевых платежей относительно других валют, как в кол-ве, так и в сумме | Сделано |
| 8 | Наиболее распространенный тип транзакций |  | Сделано |
| 9 | Кол-во месяцев активности клиента |  | Сделано |
| 10 | Отклонение любой другой фичи за прошлый месяц относительного среднего значения:  - сумма транзакций  - кол-во транзакций  - частота использования определенной категории продукта | Надо сделать обалденную фичу:  Отклонение любой другой фичи за прошлый месяц относительного среднего значения за прошлые 6 месяцев  Например суммы транзакций сильно увеличились (относительно прошлых периодов) и человек в следующий месяц купил продукт  Охх как жаль, что данных очень много не получается все хотелки рассчитать на слабых мощностях | Сделано |
| 11 | Влияние дня недели | Будни/выходные | В работу |
| 12 | Влияние суток | Утро/день/вечер/ночь | В работу |
| 13 | Влияние сезонности | Официальные праздники | В работу |
| 14 | Шаблоны транзакций | % повторяющихся транзакций | В работу |
| 15 | Наличие данных о транзакциях | Просто факт есть или нет. Очень важная фича для деревьев и мультимодальности | Сделано |
| 16 | Наличие данных о транзакциях в прошлом месяце | Просто факт есть или нет. Очень важная фича для деревьев и мультимодальности | В работу |
| 17 | Связать транзкации с диалогами и гео инфомрацией по времени транзакций |  | В работу |

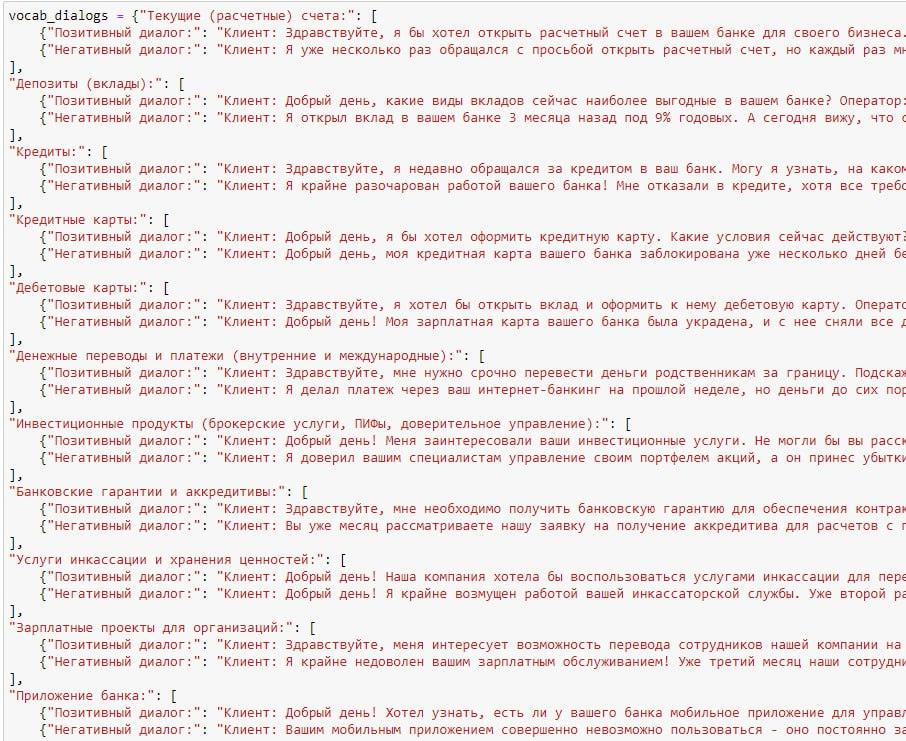
* 1. **Фичи по диалогам**

| № | Титл | Описание | Статус |
| --- | --- | --- | --- |
| 1 | Рассчет "продоваемых" эмбеддингов | Первое что на ум приходит:  1. Выбрать всех клиентов у которые покупали продукт 1/2/3/4  2. У этих клиентов выбрать все эмбеддинги диалогов за последний месяц  3. Усреднить эмбеддинги - это будет эаталонный эмбеддинг отображающий диаолги после которых произошла покупа  4. Далее абсолютно для каждого пользователя рассчитать близость до среднего эмбеддинга по продукту 1/2/3/4 и эту близость подавать как фичу в бустинг  Это один из вариантов, первое что в голову пришло | Сделано |
| 2 | Расчет среднего эмбединга для клиентов не купивших ни одного продукта | Есть эмбеддинги диалогов.  2. Более сложно.  2.1. Надо взять все эмбеддинги по которым люди через месяц покупали продукт №1. Т.е. если клиент купил продукт 31.05.2022, то взять диалоги с 01.04.2022 по 30.04.2022  2.2. Посчитать средний эмбеддинг для "хороших" диалогов (простое среднее либо умное - про умное отдельно расскажу после расчета обычного)  2.3. Взять все эмбеддинги по которым люди через месяц точно не покупали продукт №1. Но брать только по тем клиентам у которых текущая модель не сомневаясь сказала не покупает.  2.4. Посчитать средний эмбеддинг для "плохих" диалогов  2.5. И так же сделать для 2 3 и 4 продуктов  2.6. Далее всем остальным эмбеддингам в трейне и тесте посчитать удаленность до среднего "хорошего" и удаленность до среднего "плохого"  Это кажется как будто будет очень мощная фича | Сделано |
| 3 | Кластеризовать эмбеддинги | Надо кластеризовать их, чтобы получить категорию класса по ним | Сделано |
| 4 | Кол-во диалогов пользователя с банком | В идеале нужно поянть как ведут себя люди которые не общаются с поддержкой но покупают продукты | Сделано |
| 5 | Кол-во диалогов в разные дни |  | Сделано |
| 6 | Понять ГЕО это покупки или Сбер прилоджение | Если у людей есть такое что ГЕО сильно меньше чем транзакций, то значит ГЕО это не гео терминалы, а просто инфа о местонахождении человка при пользовании СберОнлайн .  По дням проверять | На подумать |
| 7 | Понять Диалоги это через приложение или нет | Если Диалоги чаще всего имеют связку с ГЕО, тогда это Диалоги через приложение СберОнлайн |  |
| 8 | Выявление общих обсуждаемых тем? Нужно ли это или лучше, наверно, кластеризовать диалоги | Latent Dirichlet Allocation (LDA) | На подумать |
| 9 | Связка будни/выходные/праздники/не праздники и обсуждаемые темы |  | В работу |
| 10 | Определение тональность/изменение тональности диалога | Возможно? и нужно ли это? | В работу |
| 11 | Кол-во месяцев активности клиента |  | Сделано |
| 12 | Статистики по эмбеддингам диалогов | Статистики считаются как по всему прошедшему периоду до отчетной даты так и только по последнему месяцу:  - Средний эмбеддинг диалога  - Сумма значений эмбеддинга диалога  - Дисперсия эмбеддингов диалога: Дисперсия эмбеддингов диалога. Высокая дисперсия может указывать на большое разнообразие тем в общение.  - Минимальное/максимальное расстояние между эмбеддингами: Минимальное и максимальное расстояние (например, евклидово или косинусное) между парами эмбеддингов в диалоге. Это может отражать сходство или различие тем в диалоге.  - Средняя разница между соседними эмбеддингами: Среднее расстояние между соседними диалогами. Это может показывать, насколько гладко меняются темы в диалоге.  - Энтропия эмбеддингов диалога: Энтропия распределения эмбеддингов реплик в диалоге. Высокая энтропия может указывать на большое разнообразие тем.  - Длина диалога в эмбеддингах: Сумма расстояний между первым и последним эмбеддингом диалога, проходя через все промежуточные эмбеддинги. Это может отражать длину или сложность диалога. | Сделано |
| 13 | PCA средних эмбеддингов диалогов |  | Сделано |
| 14 | Сдеалть статисстику по частям эмбеддинга | Фича сумму всех значений в эмбеддинге выстреливает для некоторых таргетов.  Поэтому есть смысл сделать набор фичей по частям эмбеддинга, разбив эмбеддинг напримре на 10 или 20 частей | Сделано |
| 15 | Близость диалогов к каким либо тематикам | На основе диалогов сформированы 13 тематик, для каждой из которых сделан позитивный и негативный пример. Далее сделаны эмбеддинги по всему этому, и для каждого диалога в тренировочной выборке указана косинусная близость ко всему этому:  - общая близость к негативу или позитиву  - близость к каждому из 13 типов  - близость к негативу или позитиву по каждому из 13 типов | Сделано |

### 

### Близость диалога к определенной тематике банковских продуктов

Сформированы примеры диалогов оператора Банка с клиентами. На основе диалогов сформированы 13 тематик, для каждой из которых сделан позитивный и негативный пример. Далее сделаны эмбеддинги по каждой категории и отдельно позитивный или негативный диалог внутри тематики. Далее для каждого диалога в тренировочной выборке указана косинусная близость ко всем сформированным диалогам:  
- общая близость к негативу или позитиву  
- близость к каждому из 13 типов  
- близость к негативу или позитиву по каждому из 13 типов



Использовалась библиотека SentenceTransformer и модель DeepPavlov/rubert-base-cased

По итогам моделирования в топ важных фичей попали близость к негативным и позитивным диалогам следующих тематик:

* Кредитные карты
* Денежные переводы и платежи
* Текущие (расчетные) счета
* Кредиты
* Приложение банка
* Кэшбэк
  1. **Фичи на базе таргета**

| № | Титл | Описание | Выводы | Статус |
| --- | --- | --- | --- | --- |
| 1 | Регулярные типы продуктов | Проверить, что если продукты цикличные, например подписка ежемесячная |  | Проверить |
| 2 | Дополняющие продукты (иключающие продукты) | Что если продукты это два вида подписаок и часто клиенты покупают оба продукта, чем ни один.  Проверить, что если есть продукты которые друг-друга исключают. Например вклад и кредит | - Клиенты приобревшие продукт №1 с вероятностью в 10 раз выше преобретут Проудкт№2, чем если бы не преобретали 1ый продукт  - Клиенты приобревшие продукт №1 с вероятностью в 8 раз выше преобретут Проудкт№3, чем если бы не преобретали 1ый продукт  - Клиенты приобревшие продукт №1 с вероятностью в 8 раз выше преобретут Проудкт№4, чем если бы не преобретали 1ый продукт  Аналогично наблюдается связь с другими продуктами | Проверено |
| 3 | Факт приобретения клиентом когда-либо продукта 1 или 2/3/4 | Если человек покупал уже продукты, то вероятность повторного приобретения увеличивается, также меняется и вероятность преобретения других продуктов | Однако клиенты в тренировочных и в тестовых данных не пересекаются, это значит неизвестно какие продукты приобретал клиент банка до этого  В чистом виде не получиться использовать  Можно конечно указывать вероятность преобретения ранее полученную на модели | Сделано |
| 4 | Расширеный факт приобретения клиентом когда-либо группы продуктов | По сути человек мог приобрести 1 или 2, либо 3 или 4 продукт  'is\_target\_1\_2'  'is\_target\_1\_3'  'is\_target\_1\_4'  'is\_target\_2\_3'  'is\_target\_2\_4'  'is\_target\_3\_4' |  | Сделано |
| 5 | Второй расширеный факт приобретения клиентом когда-либо группы продуктов | По сути человек мог приобрести 1 И 2, либо 3 И 4 продукт  'is\_target\_1\_and\_2'  'is\_target\_1\_and\_3'  'is\_target\_1\_and\_4'  'is\_target\_2\_and\_3'  'is\_target\_2\_and\_4'  'is\_target\_3\_and\_4'  'is\_target\_cnt' - кол-во продуктов |  | Сделано |
| 6 | Сколько именно продуктов клиент купил ранее из пунктов 3 и 4 |  |  | Сделано |
| 7 | Сколько в среднем в месяц клиент покупает продуктов | это отноешение "Сколько именно продуктов клиент купил ранее" на кол-во рассматриваемых месяцев |  | Сделано |
| 8 | Двухэтапное прогнозирование | Кросс-валидация  На 1м этапе прогнозируем вероятность для всех 4х продуктов  На 2м этапе подаем прогнозы 3х дургих продуктов при предсказании конкретного  (т.е. использовать предыдущих предиктов по другим таргетам)  Важно не допустить лика данных |  | Сделано |
| 9 | Отдельно предсказываем вообще человек готов купить любой из 4х продуктов | Делаем общий таргет на 4е продукта.  1ая модель предсказывает вообще вероятность покупки любого продукта  2ая модель предсказывает какой конкретно продукт человек может приобрести |  | Сделано |
| 10 | Для каждого класса свой UnderSampling |  |  | В работу |
| 11 | Определить какие продукты имеют повторяемость | Т.е. если определенный продукт человек купил только один раз, то скорее всего это какой-нибудь ипотека  И наборот если каждый месяц, то это какая-нибудь подписка на СберПлюс |  | На подумать |
| 12 | Общее кол-во приобретенных продуктов всеми клиентами в каждом месяце | Идея такая, что в определенный месяц происходила какая-нибудь акция.  Для обучения использовать прошлый и позопрошлый месяцы | Классная фича, занес в EDA | Сделано |
| 13 | Средний интервал между покупками куплеными Продуктами | Среднее количество месяцев между покупками продуктов для каждого клиента |  | Сделано |
| 14 | Частота покупок за последние 30/60/90/180/365 | Количество покупок продуктов за указанные периоды времени |  | Сделано |
| 15 | Доля покупок по продуктам | Доля покупок клиента для каждого продукта. |  | Сделано |
| 16 | Прошедшее время с момента первой и последней покупки | Количество дней с момента первой и последней покупки продукта |  | Сделано |
| 17 | Время неактивности | Время неактивности, сколько времени не совершались покупки |  | Сделано |

* 1. **Фичи на базе эмбеддингов PyTorch-LifeStream**

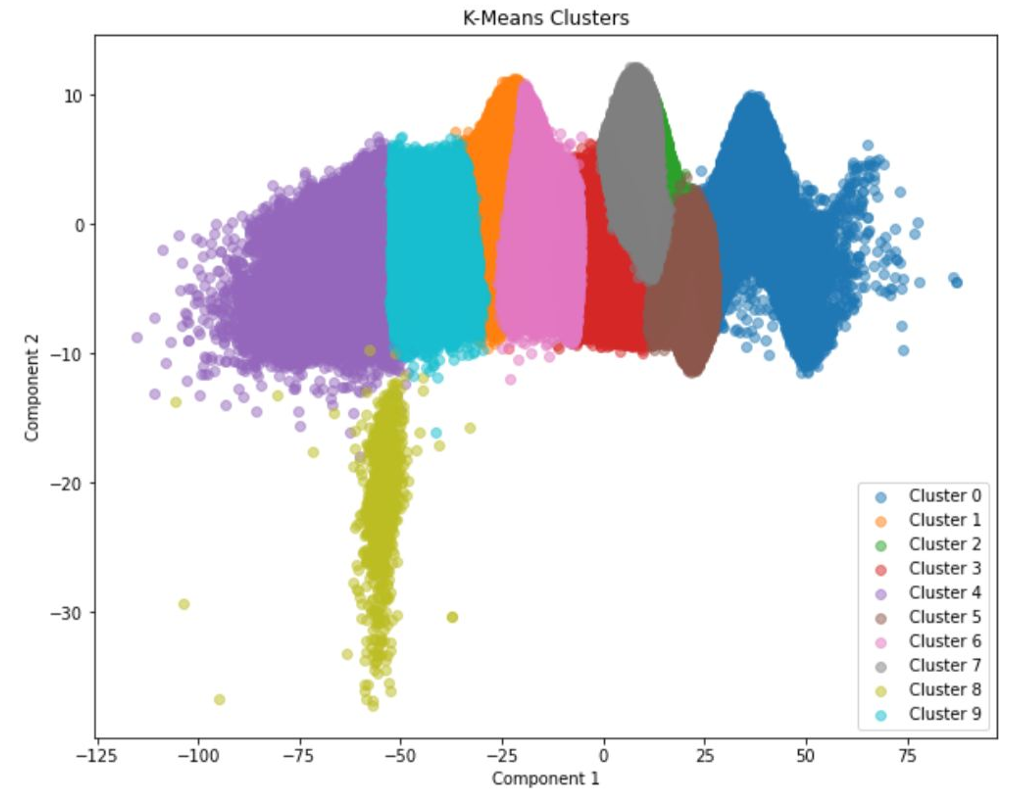
| № | Титл | **Описание** | Статус |
| --- | --- | --- | --- |
| 1 | Расчет эмбедингов на основе PTLS для всех видов данных | Расчет эмбедингов на основе PTLS для всех видов данных: гео, транзакции, таргеты  Длина итогового эмбеддинга 256 | Сделано |
| 2 | Расчет статистик по эмбеддингам | Расчет статистик: min, max, sum, std, mean, median и прочее | Сделано |
| 3 | Разбивка эмбеддинга на части | Разбиваем эмбеддинга для каждого вида данных на части по 16 элемента, всего 16 частей. | Сделано |
| 4 | Расчет статистик по частям эмбеддингов | Расчет статистик: min, max, sum, std, mean, median и прочее | Сделано |
| 5 | Близость к "хорошим" и "плохим" эмбеддингам  (Расчет среднего эмбединга для клиентов не купивших ни одного продукта) | Близость к "хорошим" и "плохим" эмбеддингам  2.1. взять все эмбеддинги по которым люди через месяц покупали продукт №1. Т.е. если клиент купил продукт 31.05.2022, то взять эмбеддинги с прошлых месяцев  2.2. Посчитать средний эмбеддинг для "хороших" объектов (простое среднее)  2.3. Взять все эмбеддинги по которым люди через месяц точно не покупали продукт.  2.4. Посчитать средний эмбеддинг для "плохих" объектов  2.5. И так сделать для всех продуктов 1 2 3 и 4  2.6. Далее каждому эмбеддингу в трейне и тесте посчитать удаленность до среднего "хорошего" и удаленность до среднего "плохого" | Сделано |
| 6 | Кластеризовать эмбеддинги | Надо кластеризовать их, чтобы получить категорию класса по ним | Сделано |

* 1. **Фичи на базе кластеризации**

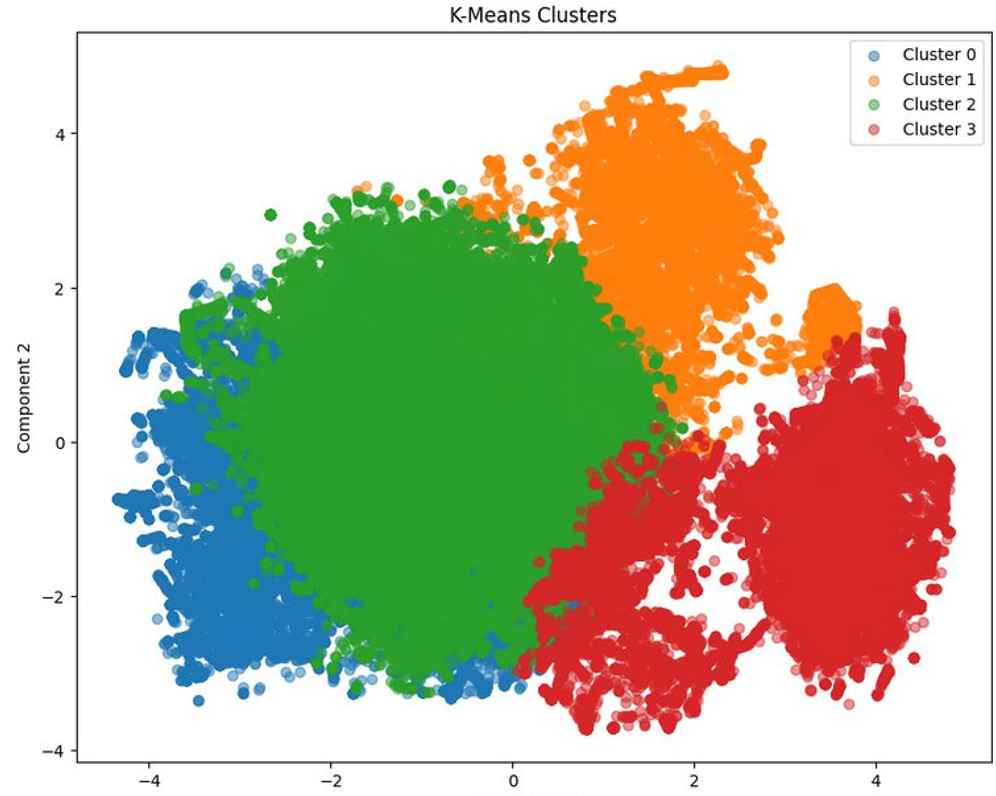
Все векторные представления (эмбеддинги диалогов, эмбеддинги транзакций и гео) были дополнительно обработаны с помощью кластеризатора с целью группировки объектов по кластерам.

В качестве кластеризатора использовался K-means.

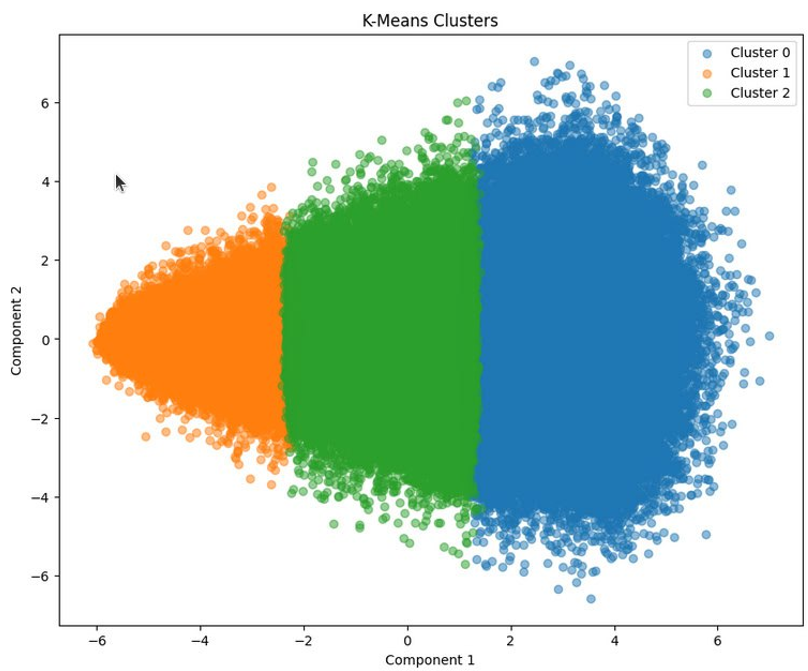
Эмбеддинг диалогов:



Эмбеддинг транзакций, обученный с помощью PyTorch-LifeStream:



Эмбеддинг гео данных, обученный с помощью PyTorch-LifeStream:

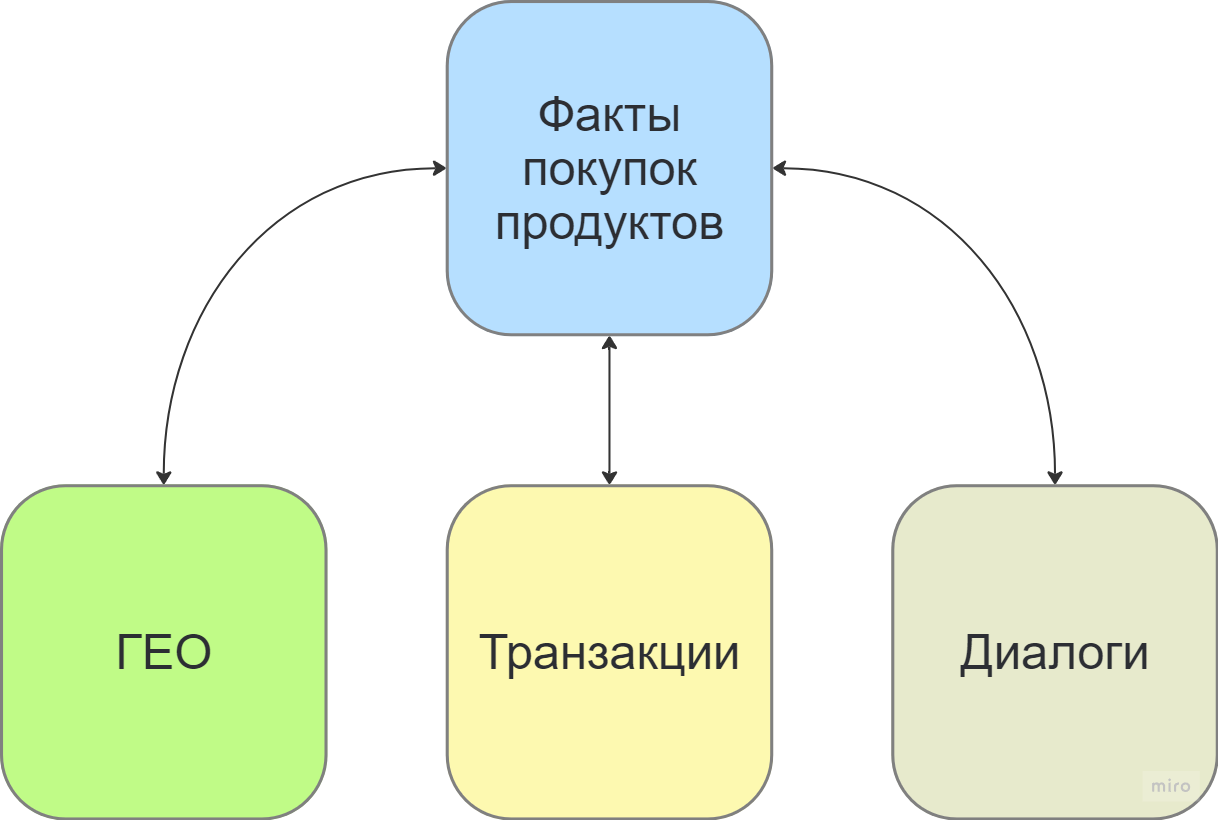


# **Мультимодальность данных**

При построении мультимодальной архитектуры решения важно взять единую точку связи различных данных. В нашем случае такими точками являются:

* Клиент
* Временные периоды
* Таргеты (Факты продаж)

В качестве базовой единицы мы выбрали пары клиент-месяц и все агрегаты данных строили на этой группировке. Мы обеспечили связь между данными Факты продаж-ГЕО, Факты продаж-Транзакции, Факты продаж-Диалоги.



В идеале необходимо прийти к связи все со всеми:

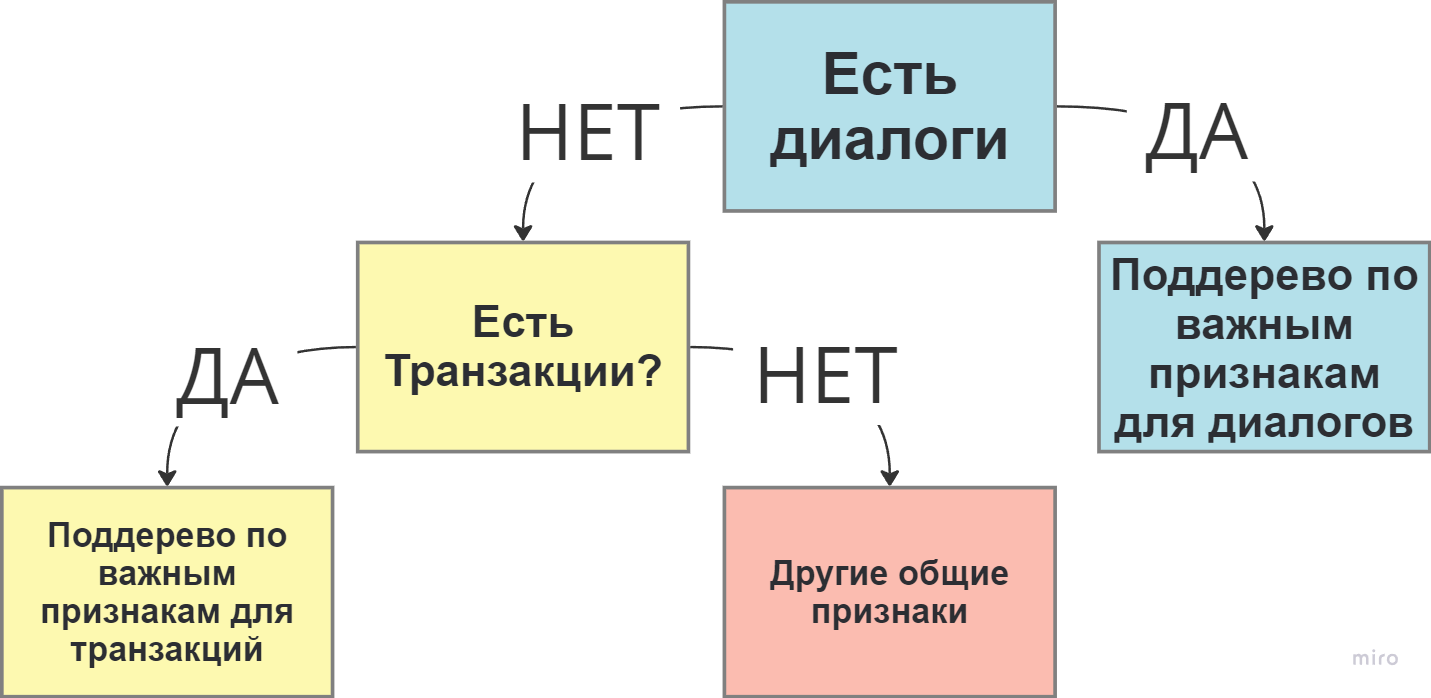


|  | Всего | Кто **не** покупал ничего | Кто покупал хоть что-то | Кто **не** покупал Продукт 1 | Кто покупал Продукт 1 | Кто **не** покупал Продукт 2 | Кто покупал Продукт 2 | Кто **не** покупал Продукт 3 | Кто покупал Продукт 3 | Кто **не** покупал Продукт 4 | Кто покупал Продукт 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Нет в geo и dlg и trx | 129477 | 129435 | 42 | 129455 | 22 | 129471 | 6 | 129463 | 14 | 129470 | 7 |
| Нет в geo и dlg | 501810 | 489815 | 11995 | 497135 | 4675 | 501075 | 735 | 496954 | 4856 | 498846 | 2964 |
| Нет в dlg и trx | 288201 | 288057 | 144 | 288131 | 70 | 288182 | 19 | 288157 | 44 | 288174 | 27 |
| Нет в geo и trx | 171524 | 171412 | 112 | 171452 | 72 | 171517 | 7 | 171499 | 25 | 171504 | 20 |
| Нет в dlg | 1031488 | 973621 | 57867 | 1009325 | 22163 | 1028152 | 3336 | 1007478 | 24010 | 1016208 | 15280 |
| Нет в geo | 650665 | 610078 | 40587 | 630174 | 20491 | 648693 | 1972 | 635912 | 14753 | 641812 | 8853 |
| Нет в trx | 358345 | 357945 | 400 | 358076 | 269 | 358321 | 24 | 358270 | 75 | 358280 | 65 |

Однако есть проблемы связности данных. Данные не всегда равносильны. Например, по данной таблице наличия/отсутствия видов данных, можно легко увидеть, что если у клиента нет транзакций, то практически всегда он не купит никакой продукт. Такой зависимости нет ни у диалогов, ни у геоданных.

Таким образом при мультимодальности необходимо учитывать уровень влияния каждого вида данных.

Важно отметить, что для мультимодальных данных хорошо подходят решающие деревья. Так например, нам даны модальности диалоги, гео, транзакции и модель должна строить на их основе прогноз. Если добавить простую фичу например, говорящую о наличии у клиента данных о диалоге “да или нет”, тогда модель может построить разные варианты деревьев в случае когда есть такая информация и в случаях когда нет.



Данное особенность деревьев дает преимущество для мультимодальных задач. Однако если подумать данную идею можно расширить и для линейных моделей.

Отдельно стоит обратить внимание на мультилейбловую сторону задачи, в которой необходимо сделать предсказание по четырем таргетам.

Отличной техникой является создание промежуточной модели, которая делает предсказание каждого таргета через кросс-валидацию. Т.е. предсказания покупки продуктов 1 2 3 можно использовать при прогнозировании покупки продукта 4. Таким образом, мы можем заполнить предсказания во всех данных без риска лика данных. Жизненную пользу данного подхода возможно описать следующим примером, что некоторые продукты могут дополнять друг друга, либо это может быть показателем общей лояльности клиента к приобретению продуктов.

В качестве развития глобально правильней было бы обучать векторное представление по всем данным сразу плюс с фичами, однако мощности не позволяют такое провернуть. Также необходимо подумать о том что данные можно более тесно связывать между собой, например частота диалогов относительно частоты транзакций и т.д. Однако время и мощности не позволяют реализовать все идеи.

# **Дисбаланс**

Представленные данные имеют очень большой дисбаланс по классам.

| **Продукт** | **Не купили (0)** | **Купили (1)** | **% 1го класса** |
| --- | --- | --- | --- |
| 1 Продукт | 10163527 | 83177 | **0,82%** |
| 2 Продукт | 10237080 | 9624 | **0,09%** |
| 3 Продукт | 10175082 | 71622 | **0,70%** |
| 4 Продукт | 10198143 | 48561 | **0,48%** |
| Любой из 4х | 10044361 | 202343 | **2,01%** |

* Самый низкий процент 1го класса для Продукта № 2 - 0,09%
* Самый высокий процент 1го класса для Продукта №1 - 0,82%
* Клиенты, которые приобретали в определенный месяц хотя бы один продукт, составляют только 2% от общего числа данных

Необходимо также учитывать, что данные разбиты не только по клиентам, но и по месяцам. Однако, для модели в качестве объекта подается связка клиент + отчетный период, поэтому это разные объекты.

С таким большим дисбалансом при обучении моделей “в лоб” появлялась проблема, что все предсказания всегда выдавались только как “нулевой” класс.

Для решения этой проблемы были применены такие техники как:

* Использование алгоритма уменьшения мажорного класса через RandomUnderSampling
* Расчет соотношения классов и подача в модель соотношений классов для борьбы с дисбалансом при обучении (class\_weights)
* Исключение “слабых” объектов из “нулевого” класса. Под слабыми объектами понимаются, например, клиенты с малым кол-ом транзакций.
* Кастомный сэмплер 1. Выбираем пользователей, которые покупали продукт, и далее выбираем их прошлые месяцы. Если они там уже не покупали продукты, то идеальный негативный класс, т.к. состояние клиентов похоже, но в одном случае не покупалось, а через месяц купил, значит что-то изменилось. Кастомный сэмплер 1 не показал эффективность.
* Кастомные сэмплеры 2. В ходе тестирования моделей определили валидационную и тестовую выборку. К ним добавили пары клиент-месяц из сабмита, добавили клиентов, которые покупали хотя бы раз продукт, и рандомно добавили пары клиент-месяц из тех случаев, когда ничего не покупалось клиентом. Данное решение позволило сильно уменьшить объем данных, оставляя пространство для генерации новых фичей и проверки различных гипотез. В ИТОГЕ ЭТО ВЫБРАНО?



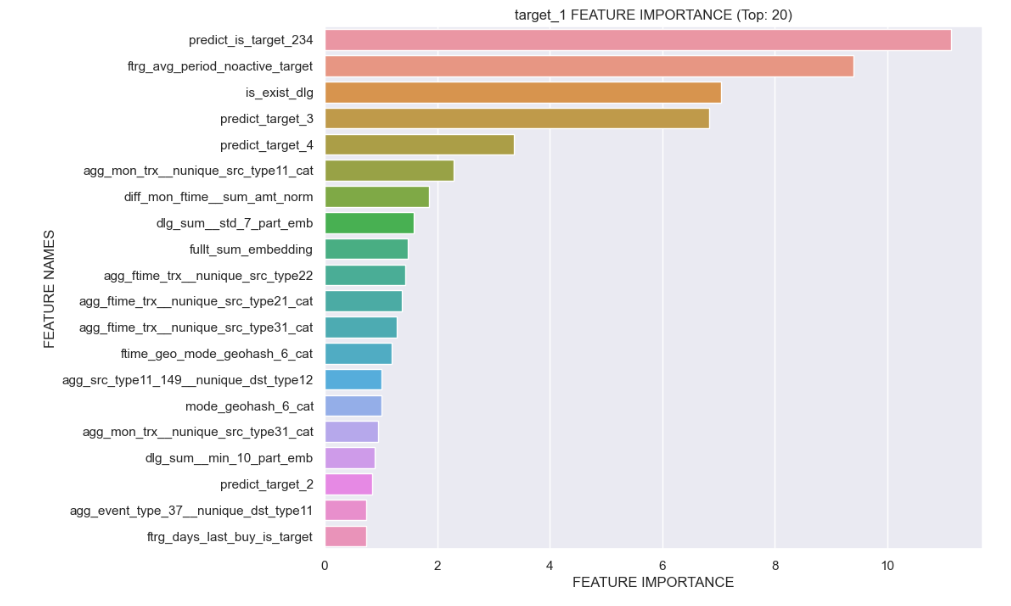
Фильтрация 0го класса позволила побороться с дисбалансом, что помимо точности модели, также дало возможность быстрее проводить моделирование и обработку данных.

# **Моделирование**

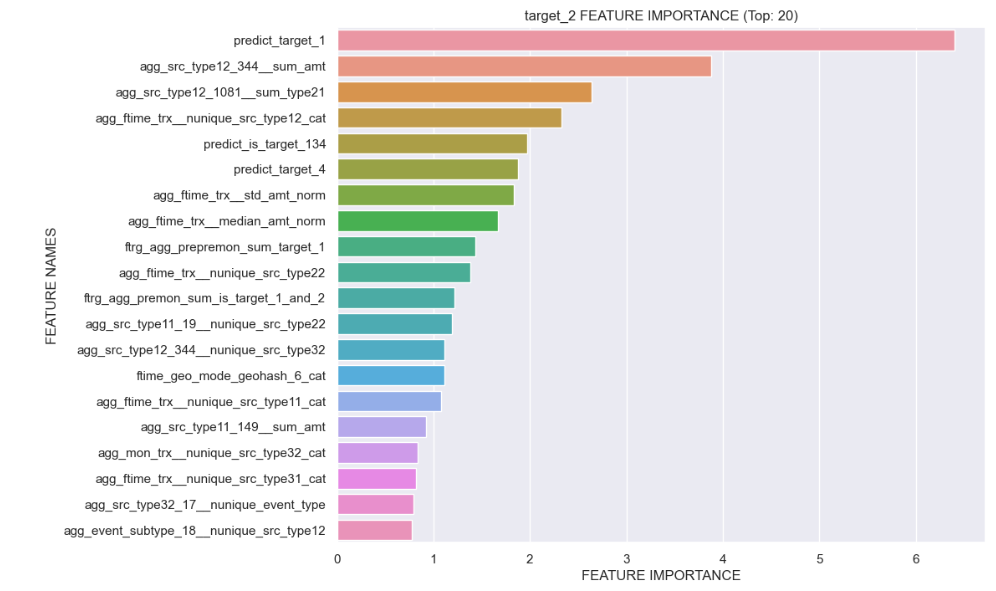
Процесс моделирования строился итерационно:

1. Подготовка фичей
2. Построение модели
3. Анализ локальных метрик
4. Анализ данных, поиск инсайтов
5. Повтори п.1-4

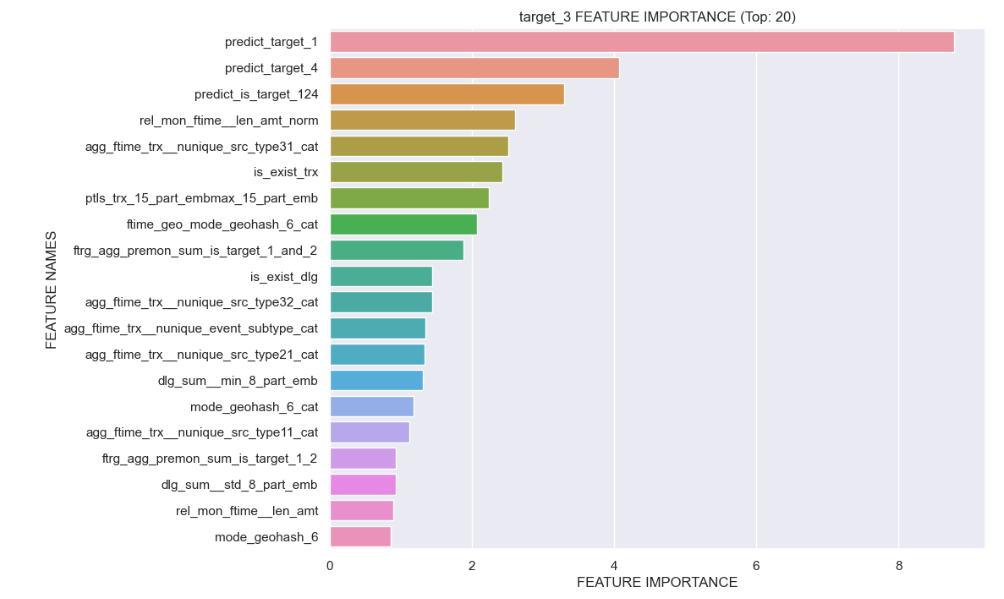
Важность фичей для предсказания Продукта 1.



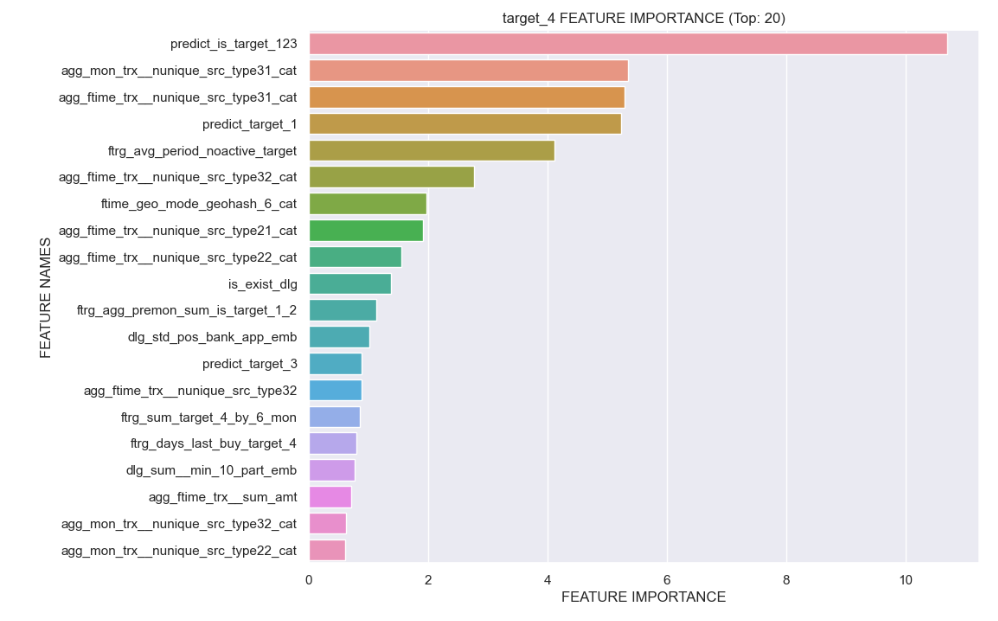
Важность фичей для предсказания Продукта 2.



Важность фичей для предсказания Продукта 3.



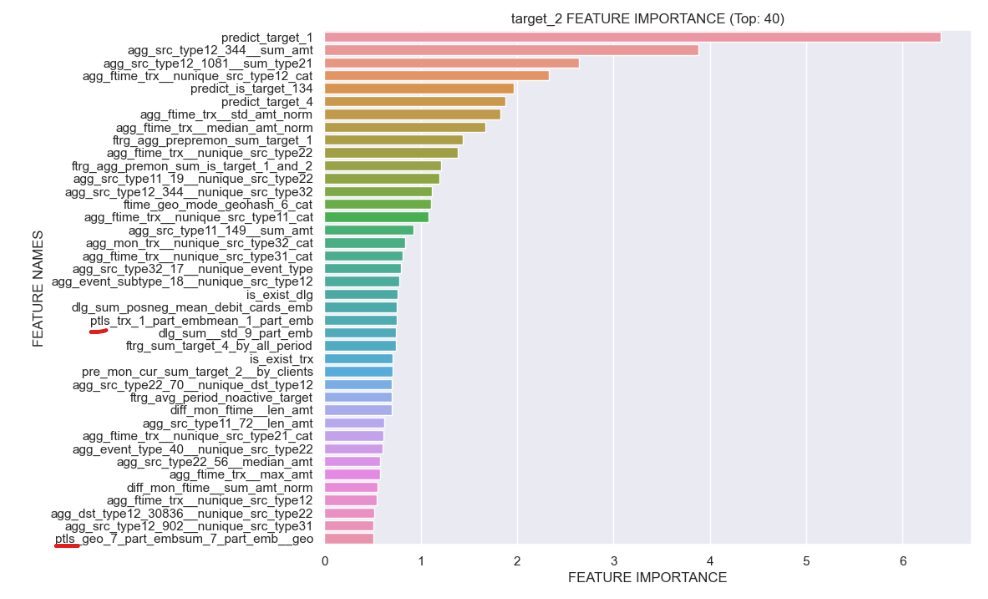
Важность фичей для предсказания Продукта 4.



Основные по важности фичи это:

* Предсказание покупки соседних товаров (получено с помощью Cross-validation)
* Типы транзакций клиента (в агрегированном виде как по прошлому месяцу так и за весь период)
* Факт наличия каждого вида данных
* Эмбеддинги построенные на базе PyTorch-LifeStream (Особенно для Продукта 1 и для Продукта 3, где они входит в топ 7)
* Признаки “неактивности” клиента (сколько по времени не совершались действия)
* Статистика по эмбеддингам разговоров (для Продукта 1 особенно)
* Агрегаты на базе геоданных

Если посмотреть чуть больше важных фичей для Продукта 2, то можно заметить, что фичи на основе PyTorch-LifeStream, также имеют значимую роль. При чем одна из фичей это разбитие эмбеддинга на группы и подсчет различных статистик.



* 1. **Учет дополнительных метрик**

При большом дисбалансе классов, что мы и наблюдаем в датасете, большинство данных будет принадлежать отрицательному классу. Это приводит к тому, что количество истинно отрицательных результатов часто доминирует над истинными положительными, что может привести к некорректным оценкам.

Поэтому помимо ROC-AUC при моделировании использовались дополнительные метрики

Такие метрики, как PR-AUC (Precision-Recall AUC), F1 Score, P4 и Gini могут быть более эффективными в подобных случаях.

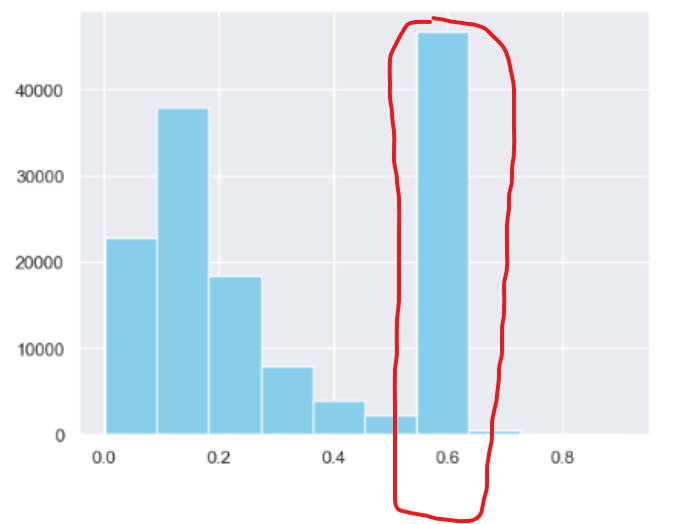
* Кривая PR-AUC показывает, как выбор порога влияет на точность классификатора, а также помогает выбрать лучшее значение дискриминационного порога для определенного баланса классов.
* F1 Score объединяет в себе информацию о точности и полноте, а точность и полнота, не зависят от соотношения классов и, следовательно, могут применяться в условиях несбалансированных выборок.
* P4 - расширение *F*1-меры, обладающее симметрией относительно инверсии классов.
* Gini лучше показывает различие в предсказательной способности модели на всех уровнях порогов.

Дополнительно контролировалось распределение предсказаний, что помогало оперативно понять изменение качества модели.

Так например выглядят распределения по различным таргетам.

| Распределение предсказания по Продукту 1  для данных сабмита | Распределение предсказания по Продукту 2  для данных сабмита |
| --- | --- |
|  |  |
| Распределение предсказания по Продукту 3  для данных сабмита | Распределение предсказания по Продукту 4  для данных сабмита |
|  |  |

Например видно, что у продукта № 1 есть проблемы с уверенностью предсказаний.



Для более чем 40.000 клиентов модель не может предсказать, и ставит 0.5. Это может быть вызвано, например особенностью данных. Так например как раз по 40.000 пользователям нет данных по гео, транзакциям и диалогам.

Посмотрим распределение по предсказаниям таргетов по обучающей выборке:

| Распределение предсказаний по Продукту 1  по обучающей выборке | Распределение предсказания по Продукту 2  по обучающей выборке |
| --- | --- |
|  |  |
| Распределение предсказания по Продукту 3  по обучающей выборке | Распределение предсказания по Продукту 4  по обучающей выборке |
|  |  |

Видно, что модель в 1 таргете менее уверена в целом.

При углубленном анализе выяснилось, что проблема в первую очередь из-за клиентов по которым нет никакой информации ни транзакций, ни гео, ни диалогов. После дополнительного анализа решено таких пользователей исключать из обучающей выборки, чтобы помочь модели сконцентрироваться на пользователях с данными. В тестовых данных для таких пользователей проставляются 0, что означает, что клиент не приобретет продукт в следующем месяце. Здесь присутствует жизненная логика, что если человек никак не использует услуги банка, либо это вновь пришедший клиент, то меньше вероятность, что он будет приобретать дополнительные продукты. Также были добавлены признаки наличия у клиента каждого вида информации.

В итоге после этих манипуляций модель по первому таргету стала формировать прогнозы с более правильным распределением. В дальнейшем мы вернули часть пар клиент-месяц с отсутствующими транзакциями и добавили признак наличия транзакций, чтобы проверить, как модель сама справляется с данной задачей.

| Новое распределение предсказания по Продукту 1 для данных сабмита | Новое распределение предсказания по Продукту 2 для данных сабмита |
| --- | --- |
|  |  |
| Новое распределение предсказания по Продукту 3 для данных сабмита | Новое распределение предсказания по Продукту 4 для данных сабмита |
|  |  |

# 

* 1. **Pytorch-LifeStream**

Создание эмбеддингов на PyTorch-LifeStream

В качестве входной точки был использован бейзлайн от организаторов. Однако в связи большим объемом исходных данных его пришлось переписать под использование PySpark. Генерацию выполняли для данных транзакций и геохешей. Для разработки решения использовали документацию PyTorch-LifeStream из репозитории [GitHub](https://github.com/dllllb/pytorch-lifestream).

Препроцессинг предложенный в бейзлане нам не подошёл из-за большого объема данных, воспользовались решением на основе PySpark. В качестве базовой модели выбрана модель CoLES, так как она неплохо зарекомендовало себя на похожих соревнований.

Для итоговой модели обучили по две модели для каждого типа данных. Модели отличались между собой по количеству параметров используемых для генерации эмбеддингов. Размер обученных эмбеддингов был длинной 256 и 32.

Предложенная схема инференса в базовом решении нам не подошла из-за большего требования к памяти и времени обработки данных. Для решения задачи было решено реализовать свою функцию фильтрации данных по месяцам, что позволило значительно уменьшить скорость генерации эмбеддингов для каждого датасета.

Для сгенерированных эмбеддингов были замерены косинусное, евклидово и манхентовское расстояния до “эталонного эмбеддинга”. “Эталонный эмбеддинг” был определен как средний эмбеддинг для каждого таргета когда была покупка и когда не было покупки. Также эмбеддинги были кластеризованы при помощи КMeans.

# **Проверенные гипотезы**

* Гипотеза № 1. Гипотеза повышения вероятности приобретения продукта, если был приобретен ранее другой продукт.

Проверена гипотеза влияния факта приобретения одного продукта на вероятность приобретения других продуктов.

Что если продукты – это два вида подписок и часто клиенты покупают оба продукта, чем ни один.

Так определено, что:

- Клиенты, которые приобрели продукт №1, с вероятностью в 10 раз выше приобретут Продукт №2, чем если бы не приобретали 1ый продукт.

- Клиенты, которые приобрели продукт №1, с вероятностью в 8 раз выше приобретут Продукт №3, чем если бы не приобретали 1ый продукт.

- Клиенты, которые приобрели продукт №1, с вероятностью в 8 раз выше приобретут Продукт №4, чем если бы не приобретали 1ый продукт.

Однако, клиенты в тренировочных и в тестовых данных не пересекаются, это значит, неизвестно какие продукты приобретал клиент банка до этого. В чистом виде не получиться использовать.

Можно, конечно, указывать вероятность приобретения, ранее полученную на модели.

* Гипотеза № 2 ДОПИСАТЬ ИЛИ УБИРАЕМ ЭТОТ РАЗДЕЛ?

# **Оригинальность решения**

1. Анализ распределения таргетных данных от тестовых

## Уменьшение размера датасета Применены решения по оптимизации типов данных. Например, произведены такие преобразования: - перевод значений из float64 в float32 - перевод временных меток из object в тип datetime - для целочисленных значений осуществлен перевод из float64 в соответствующий int (int8, int16, int32), в зависимости от допустимых максимальных/минимальных значений и т.д.

Это позволило значительно уменьшить объем данных, например, объем транзакций уменьшился с 21,6 Гб до 8,7 Гб без потери данных (в 2,5 раза).

Решение по уменьшению объема датасета реализовано в качестве отдельной функции.

1. Защита от Shake-up: кросс-валидация на фолдах, стремимся снизить разброс. Для каждого таргета проводился анализ стабильности предсказания по фолдам разброс точности на фолдах по всем таргетам примерно находится в диапазоне 1-2%.
2. Нормализация сумм транзакций относительно внешних данных по инфляции и курсам валют ЦБ России.
3. Формирование примеров диалогов оператора Банка с клиентами по разным тематикам и расчет близости диалогов клиента к каждой тематике, а также анализ негативного и позитивного диалога.

# **Техническое решение**

* 1. **Тестирование**

В качестве программного тестирования решения используются конструкции типа “assert”.

Например, при разделении данных на трейн/тест/вал производиться проверка, что нет ситуаций когда клиенты между train/test/val пересекаются чтобы избежать лика данных:

| **assert** len(set(X\_train['client\_id'].values)&set(X\_test['client\_id'].values)) == 0, 'Ошибка в разделение клиентов X\_train и X\_test' **assert** len(set(X\_test['client\_id'].values)&set(X\_val['client\_id'].values)) == 0, 'Ошибка в разделение клиентов X\_val и X\_test' **assert** len(set(X\_train['client\_id'].values)&set(X\_val['client\_id'].values)) == 0, 'Ошибка в разделение клиентов X\_train и X\_val' |
| --- |

* 1. **Уменьшение размера датасета** ЭТО УЖЕ ЕСТЬ В ОРИГИНАЛЬНОСТИ

Применены решения по оптимизации типов данных. Например, произведены такие преобразования:

- перевод значений из float64 в float32

- перевод временных меток из object в тип datetime

- для целочисленных значений осуществлен перевод из float64 в соответствующий int (int8, int16, int32), в зависимости от допустимых максимальных/минимальных значений

и т.д.

Это позволило значительно уменьшить объем данных, например, объем транзакций уменьшился с 21,6 Гб до 8,7 Гб без потери данных (в 2,5 раза).

Решение по уменьшению объема датасета реализовано в качестве отдельной функции.

* 1. **Разбиение на train/test/val**

Разделение данных на тренировочные, тестовые и валидационные производилось по следующим принципам:

* Разделение на трейн и тест производилось независимо для каждого таргета
* Разделение производилось по клиентам, т.е. важно, чтобы в трейне и тесте клиенты не совпадали
* Поэтому сначала схлопывали клиентов с признаком “хотя бы раз покупал продукт в любом месяце”
* Далее делили клиентов стратификацией по факту приобретения продукта
* Далее возвращали все отчетные месяцы для клиентов train/test/val выборок
  1. **Инструменты**
* PyTorch-LifeStream
* PySpark  
  Препроцессинг данных с использованием Spark из-за больших объемов фичей

@Dimk\_88 напиши плиз один абзац про использование spark почему и как это улучшило скорость и потребление памяти. Добавлю в документацию

* Optuna
* Polars
* imbalanced-learn
* SentenceTransformer
* KMeans
* CatBoost
* XGBoost
  1. **Не получилось**
* При попытке уменьшить объем данных пробовали перевод категорий из типа object в тип category. При работе с данными в памяти действительно объем уменьшался где-то на 50-100 Мб, но при сохранении, например, транзакций, объем паркет файла увеличивался с 3 Гб до 17 Гб. В итоге отказались от этого подхода.
* Из-за большого объема использование некоторых under сэмплингов данных, типа TomekLinks, слишком затратно по времени.
* Для работы с большим количеством данных использовали Polars, он хоть и давал скорость, но не ту, которая требовалась. В итоге самое лучшее решение было предварительно сэмплировать данные.

# **Анализ экспериментов**

| **№** | **Дата и время** | **Название** | **Краткое описание** | **Фичи** | **1 Target ROC-AUC** | **2 Target ROC-AUC** | **3 Target ROC-AUC** | **4 Target ROC-AUC** | **Общий ROC-AUC** | **F1 Score** | **PR-AUC** | **DSWorks** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 05.06.2024  23:00:00 | CatBoost Базовая модель | Catboost на агргеационных фичах транзакций | 100 фичей агрегаты транзакций | 0,6586 | 0,7584 | 0,7357 | 0,7786 | 0,7328 |  |  |  |
| 2 | 06.06.2024  22:00:00 | CatBoost Базовая модель | Catboost на агргеационных фичах транзакций | 368 фичей агрегаты транзакций | 0,6731 | 0,791 | 0,7514 | 0,7951 | 0,7527 |  | T4: 0.0387 |  |
| 3 | 07.06.2024  20:00:00 | CatBoost Базовая модель | Catboost на агргеационных фичах транзакций и гео фичах. Удалена часть клиентов с нулевыми транзакциями | 383 фичей агрегаты транзакций и гео | 0,705 | 0,7854 | 0,7318 | 0,8043 | 0,7566 |  | T1: 0.0240  T2: 0.0109  T3: 0.0292  Т4: 0.0390 |  |
| 4 | 07.06.2024  22:00:00 | CatBoost Базовая модель | Модель 3 + эксперименты с дисбалансом | 383 фичей агрегаты транзакций и гео | 0.7059 |  |  |  | #DIV/0! |  | Т1: 0.0254 |  |
| 5 | 07.06.2024  22:40:00 | CatBoost Базовая модель | Модель 3 + эксперименты с дисбалансом | 383 фичей агрегаты транзакций и гео | 0,7078 |  |  |  | 0,7078 |  | Т1: 0.0248 |  |
| 6 | 08.06.2024  14:30:00 | CatBoost Базовая модель | Модель 3 + фичи по таргетам. Покупал ли ранее клиент продукты | 422 фичей агрегаты транзакций и гео | 0,6738 | 0,7878 | 0,7172 | 0,8135 | 0,7481 |  | T1:0.2639  T2: 0.1354  T3: 0.3256  T4: 0.3780 |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  | #DIV/0! |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

# 