ML в FastBoard

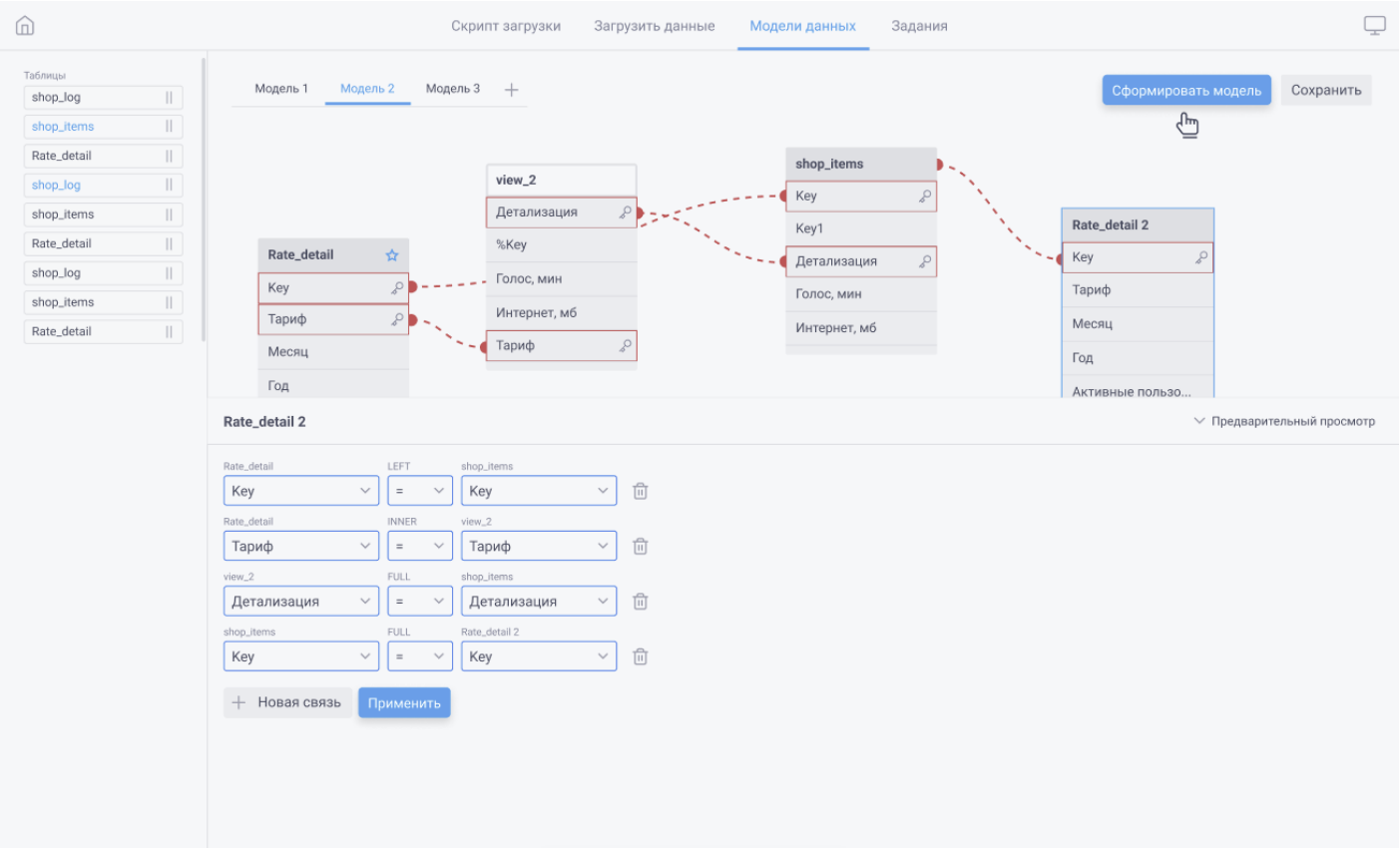
- Модуль подготовки модели данных с помощью ИИ
- Модуль интеллектуального управления и проверки качества данных
- Нейровизуализации
- Нейроконнекторы

Модуль подготовки модели данных с помощью ИИ

Цель

Автоматическое формирование связей между загруженными таблицами в модели данных: определение ключевых полей и типов соединений между ними в зависимости от их данных.

Концепт для интерфейса



В диспетчере данных в разделе Модель данных необходимо добавить 3 управляющих кнопки:

1. Сформировать модель – отправляет на бэкенд набор данных и метаданных по столбцам таблиц в итоговой модели: названия столбцов, их типы данных, количество уникальных значений, поля в сортировке. Вызывается модель ИИ для формирования связей между таблицами. Отображает в интерфейсе результаты работы модели.
2. Применить – применяет предложенные связи, соединяет таблицы в единую модель данных.
3. Отменить – удаляет все предложенные связи, возвращает модель данных в состояние, предшествующее использованию ИИ.

Кнопки "Сохранить" и "Удалить" появляются только тогда, когда нажата кнопка "Сформировать модель" и исчезают после нажатия одной из них.

Любой переход в другой раздел вызывает системное сообщение, оповещающее пользователя о том, что все изменения будут отменены.

Сформированная ИИ модель представляет собой набор полей и их соединений в области предварительного просмотра; пунктирные линии между полями таблиц в области модели.

Сформированная, но не примененная модель позволяет пользователю вносить изменения в созданные связи, удалять их и добавлять новые.

Концепт для ML

Модель получает из интерфейса набор данных и метаданных по столбцам таблиц в итоговой модели: названия столбцов, их типы данных, количество уникальных значений, поля в сортировке. Используя эту информацию, а также некоторые вычисляемые характеристики, необходимо определить поля для создания связей между предложенными таблицами, тип связи (вид Join) и условия соединения (On). Для этого можно выполнить:

- Анализ уникальности полей
- Частотный анализ
- Оценку названий полей
- Определение полей в сортировке
- Поиск совпадающих значений
- Методы машинного обучения

Кроме того, при формировании связей существует ряд ограничений. Так, не должно быть создано модели, в которой существует хотя бы одно "кольцо" – замкнутая связь между 3 и более таблицами.

В качестве выхода модели необходимо предоставить пары полей в таблицах, тип Join (Inner, Left, Right или Full) и условие соединения On (=, >, < и т.д.).

Задание на разработку ML

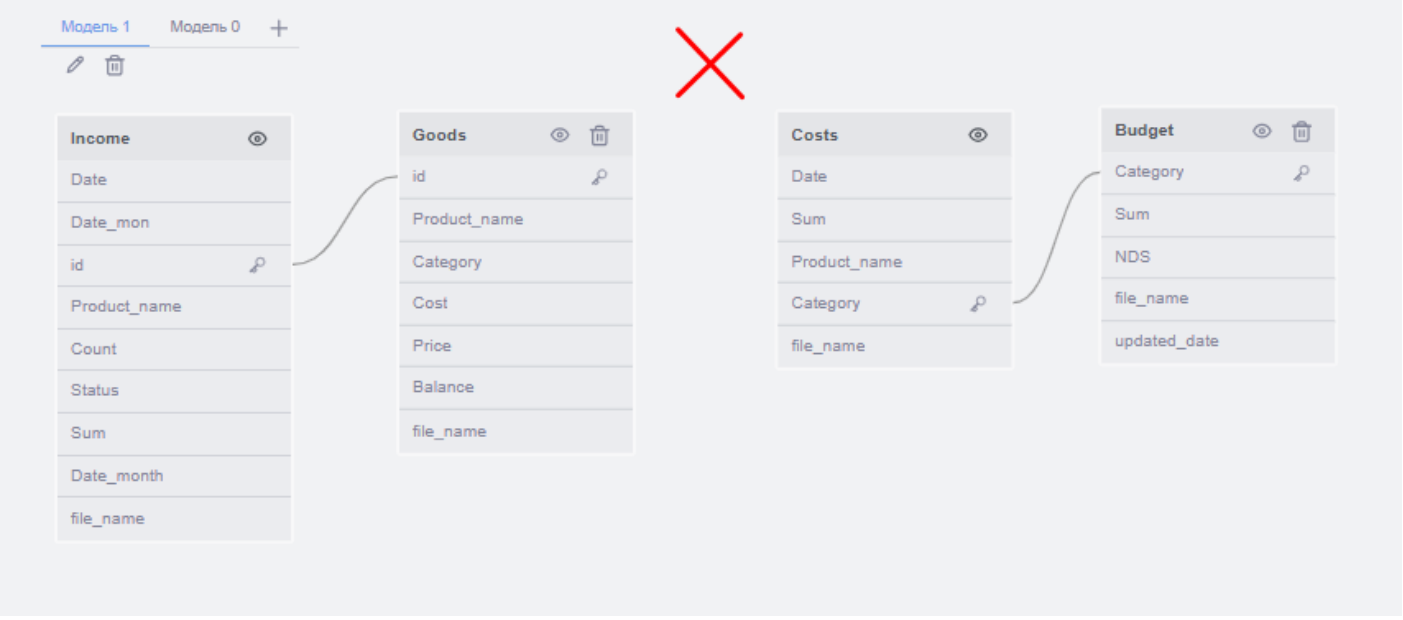
На вход модели ИИ попадают следующие данные из итоговой модели данных проекта:

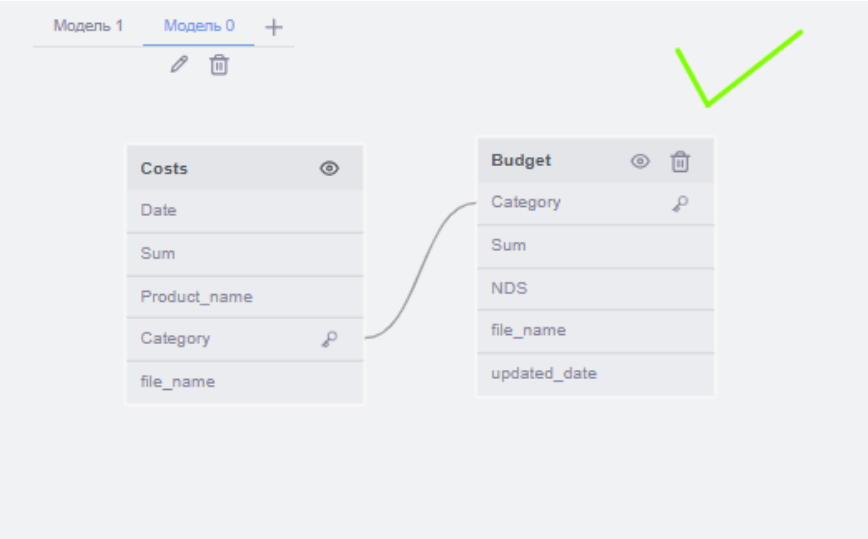
- Названия таблиц вида Table
- Названия столбцов с принадлежностью таблице вида Table.Column
- Типы данных для каждого из столбцов вида Int32, String, Decimal (2)
- Сами значения из столбца, если объём невелик, или подвыборка из столбца (н-р, как результат сэмплирования), если данных достаточно много
- Информация о полях в сортировке и первичных ключах вида Order By (список полей со всех таблиц) и Primary Key (список полей со всех таблиц)
- Разрешено ли использовать одну таблицу более чем в одной модели (булево)

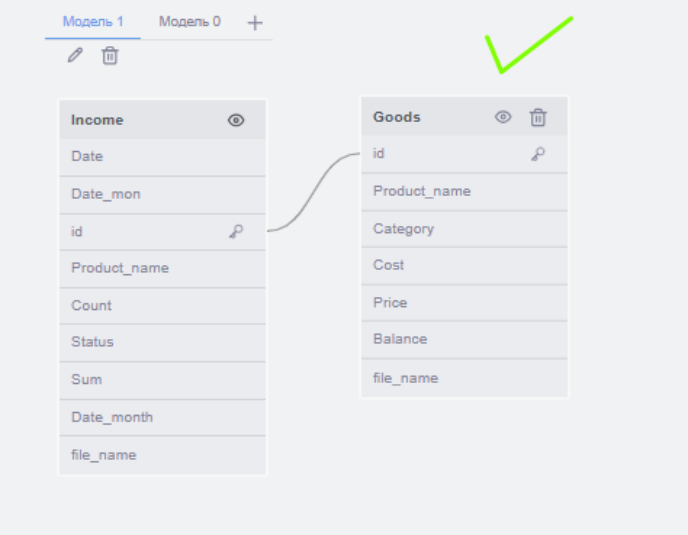
Для получения наиболее корректного результата рекомендуется разбить работу модели на 4 блока: определение групп таблиц для связей, определение типа Join и условия соединения в зависимости от контекста каждой группы, классификацию столбцов внутри групп для выявления ключей соединения, формирование связей между таблицами (выход модели).

Формирование групп для соединения

Для FastBoard принципиально важно, чтобы все имеющиеся в модели данных таблицы были соединены. **Если между таблицами нет связи, то они должны быть разделены на разные группы**.







Набор входных таблиц со столбцами необходимо разбить по группам таким образом, чтобы выполнялись следующие правила:

- Ни одна таблица не должна встречаться более чем в одной группе, **если не установлена соответствующая настройка**
- Могут быть группы из одной таблицы, если это имеет логическое обоснование, причём их число должно быть минимизировано
- Должна быть как минимум одна группа из более чем одной таблицы
- Не должно быть создано группы, в которой существует хотя бы одно "кольцо" – замкнутая связь между 3 и более таблицами

Формирование групп происходит в зависимости от контекста использования таблиц. Модель должна определять этот контекст, опираясь на названия таблиц и другие входные данные (пример: связь между Income и Goods для определения результатов продаж по категориям товаров).

На этапе группировки НЕ определяются ключевые поля и НЕ создаются связи между ними.

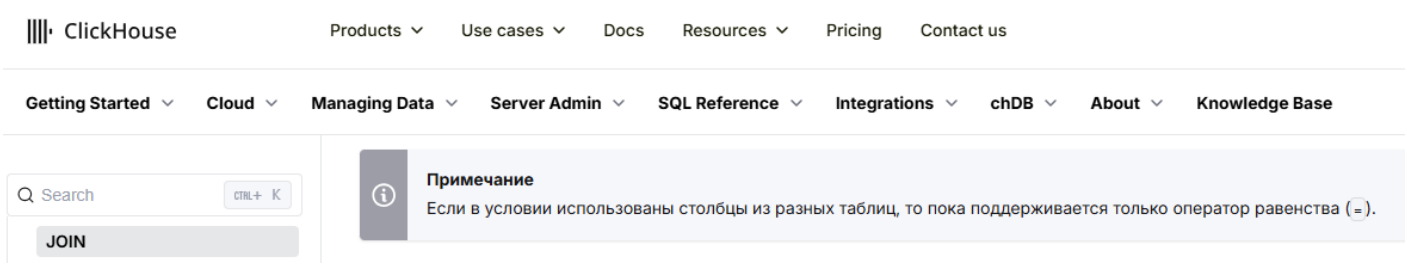
Определение типа и условия соединения

Исходя из выбранного контекста модель должна определять тип связи между таблицами внутри группы:

- Если необходимо сохранить все строки из обеих таблиц в итоговом результате, то FULL JOIN
- Если необходимо сохранить все строки из одной таблицы в итоговом результате, то LEFT/RIGHT JOIN
- Если нет необходимости сохранить все строки, а в итоговом результате должны быть только значения из пересекающихся строк, то INNER JOIN

Необходимость сохранения строк также должна определяться контекстом соединения таблиц. Например, при соединении таблиц Income и Goods нет смысла сохранять все строки какой-то из таблиц – пользователя не интересуют продажи товаров, не описанных в таблице Goods, или товары из этой таблицы, которые не продавались и не отражены в таблице Income. А вот при соединении таблицы с данными о сотрудниках и таблицы с данными о их трудозатратах стоит выполнить Left Join по таблице сотрудников, поскольку в итоговом результате особенно важно видеть тех сотрудников, у которых отсутствует информация об их трудозатратах.

> Для соединения различных таблиц в условии ON возможно использовать ТОЛЬКО оператор равенства.



Другие типы JOIN и условий ON

Определение ключевых полей для соединения

В рамках этого блока необходимо выполнить задачу классификации для разделения полей на обычные и ключевые, сформировать пары из ключевых полей для построения связей, отсеять наименее подходящие ключевые поля и пары таких полей.

Для выполнения задачи классификации кроме метаданных (таких как названия полей и таблиц) предоставляются также данные из полей (значения ячеек). В случае, если объём исходных данных слишком велик, на вход модели может поступать лимитированная или сэмплированная часть изначальной выборки. Количество строк для каждой таблицы, поступающей на вход, не будет превышать 1 000 000.

В качестве "сигналов" о том, что поле является ключевым, может выступать следующая информация:

- В названии поля содержится сочетание букв "id", "key", "code", "ref", "num", "no", написанные отдельно, слитно с другими словами или через нижнее подчёркивание
- В поле много уникальных значений. Такое поле может быть *первичным* ключом для соединения. Чтобы проверить эту информацию, необходимо число уникальных значений разделить на число всех значений в поле. Чем ближе это значение к 1, тем выше шанс, что поле является ключевым
- В поле много повторяющихся значений. Такое поле может быть *внешним* ключом для соединения. Чтобы проверить эту информацию необходимо выполнить частотный анализ
- Поле используется в сортировке или является Primary Key. Соответствующая информация передаётся из модели данных. В ClickHouse сортировка используется в том числе и для создания первичных ключей, поэтому поля в сортировке представляют повышенный интерес при формировании связей
- В других таблицах есть поля с таким же или похожим (содержащим текст полностью или частично) названием
- В таблице, в которой находится это поле, не было найдено других ключевых полей, или итоговая вероятность присвоению другим полям класса "ключевое" ниже, чем для этого поля
- Булевы значения, списки, массивы и вещественные числовые типы (Float, Decimal) минимизируют шанс того, что поле является ключевым

Используя данную информацию и примеры построения связей в обучающих наборах необходимо сформировать список ключевых полей для каждой из таблиц. Далее необходимо перебрать все возможные пары ключевых полей **ВНУТРИ ГРУПП** и определить, возможна ли связь между ними и её приоритет по следующим правилам (если выполнены все правила, то однозначно присваиваем указанный приоритет, иначе проверяем следующий, если для него совпадений больше – устанавливаем его):

1. Высокий приоритет:

- между таблицами с такими или похожими названиями в обучающем наборе часто устанавливаются связи
- поля имеют одинаковое название
- названия полей типовые
- в одном поле 100% уникальных значений, в другом много повторяющихся
- значительная часть данных совпадает (от 50% для присоединяемой таблицы)
- поле с уникальными значениями используется в сортировке или имеет первичный ключ

2. Средний приоритет:
   - между таблицами с такими или похожими названиями в обучающем наборе иногда устанавливаются связи
   - поля имеют похожее название
   - названия полей содержат типовые сочетания букв
   - в одном поле много (более 50%) уникальных значений, в другом есть повторяющиеся
   - совпадающих данных мало (10-50% для присоединяемой таблицы)
   - в таблице с полем с уникальными значениями нет полей в сортировке или с первичными ключами
3. Низкий приоритет:
   - между таблицами с такими или похожими названиями в обучающем наборе связи устанавливаются редко или совсем никогда
   - поля отличаются по названию
   - названия полей не содержат типовые сочетания букв
   - в обоих полях мало (менее 50%) уникальных значений, а частота повторений не превышает 1% от общего объёма выборки для каждого из значений
   - совпадающих данных нет или очень мало (до 10% для присоединяемой таблицы)
   - в одной или обеих таблицах есть поле в сортировке или с первичным ключом, но это поле не из рассматриваемой пары
4. Нулевой приоритет:
   - Существует по меньшей мере одно правило, полностью препятствующее созданию связи (например, из секции Важно или если запрещено повторное использование таблиц)

> Важно! Если в таблице уже есть поле для связи с присоединяемой таблицы, то это ИСКЛЮЧАЕТ возможность создания ещё одной связи с этой таблицей. Невозможность создания связи между двумя таблицами по нескольким полям актуальна для релиза 1.4.0. В дальнейшем планируется добавить эту возможность – появится необходимость пересмотреть логику определения связи между ключевыми полями

Порядок применения правил:

- Проверяется число выполнения правил для каждого из приоритетов, начиная с высокого и заканчивая низким

- Если в одном из приоритетов наблюдается абсолютное большинство выполнений правил, то устанавливаем этот приоритет
- Если в нескольких правилах наблюдается одинаковое число выполнений правил, то устанавливаем больший приоритет из них

Порядок формирования связей:

1. Сначала расставляются связи с высоким приоритетом для полей, которые были классифицированы как "ключевые" с наибольшей вероятностью
2. После установления каждой связи происходит пересмотр всех остальных: если появилось по меньшей мере одно правило, полностью препятствующее созданию связи, то устанавливаем нулевой приоритет; если в одной из рассматриваемых таблиц появилась связь с другой таблицей – понижаем приоритет на 1 позицию, но не опускаем до нулевого
3. Далее последовательно расставляем остальные связи по мере снижения приоритета и вероятности присвоения класса "ключевое" используемым полям, возвращаясь к пункту 2 после установления каждой связи

Выход модели:

- Группы таблиц: название группы (типовое с номером) и список таблиц в группе
- Связи между таблицами вида Таблица1 INNER/LEFT/RIGHT/FULL JOIN Таблица2 ON Таблица1.поле = Таблица2.Поле
- (Дополнительно) При возможности нейросеть должна генерировать название для каждой группы одним-двумя короткими словами (будет использоваться в качестве названия модели), ChatGPT +- справился с этой задачей:

1. **Income, Goods** — "Прибыль от товаров"
2. **Income, Goods, Budget** — "Финансовый план"
3. **Budget, Costs** — "Расходный бюджет"
4. **DDS, PL** — "Движение финансов"
5. **Shops, Sales, Sellers** — "Торговая сеть"
6. **City, Street, Shops** — "Расположение магазинов"

Требования:

Время работы модели не должно превышать 3 сек.

Работа на кластере (на будущее)

Поскольку в KX соединение JOIN для больших объёмов данных работает плохо, необходимо минимизировать их применение, соединив наибольшее возможное число таблиц с фактами в одну (например, с помощью UNION ALL).

Порядок действий:

- Нейросеть должна определить, какие из имеющихся таблиц можно отнести к справочникам (к которым будут подтягиваться данные из одной общей таблицы). Справочник можно идентифицировать по наличию пар "ключ-значение", согласно которым возможно формирование связей с таблицей фактов. Остальные таблицы относятся к фактическим.
- Фактические таблицы объединяются скриптом в одну таблицу:
  SELECT Столбцы FROM Таблица1 UNION ALL Столбцы FROM Таблица2 UNION ALL Столбцы FROM Таблица3...
  Скрипт не запускается, а используется для создания таблицы предварительного просмотра, демонстрируемой в отдельном модальном окне.
- Если пользователя устраивает структура, он нажимает кнопку "Продолжить". Иначе – "перегенерировать" (модель запускает ещё одну попытку классификации таблиц на справочники и факты).
- После подтверждения срабатывает сгенерированный скрипт как в обычной ситуации.

Модуль интеллектуального управления и проверки качества данных

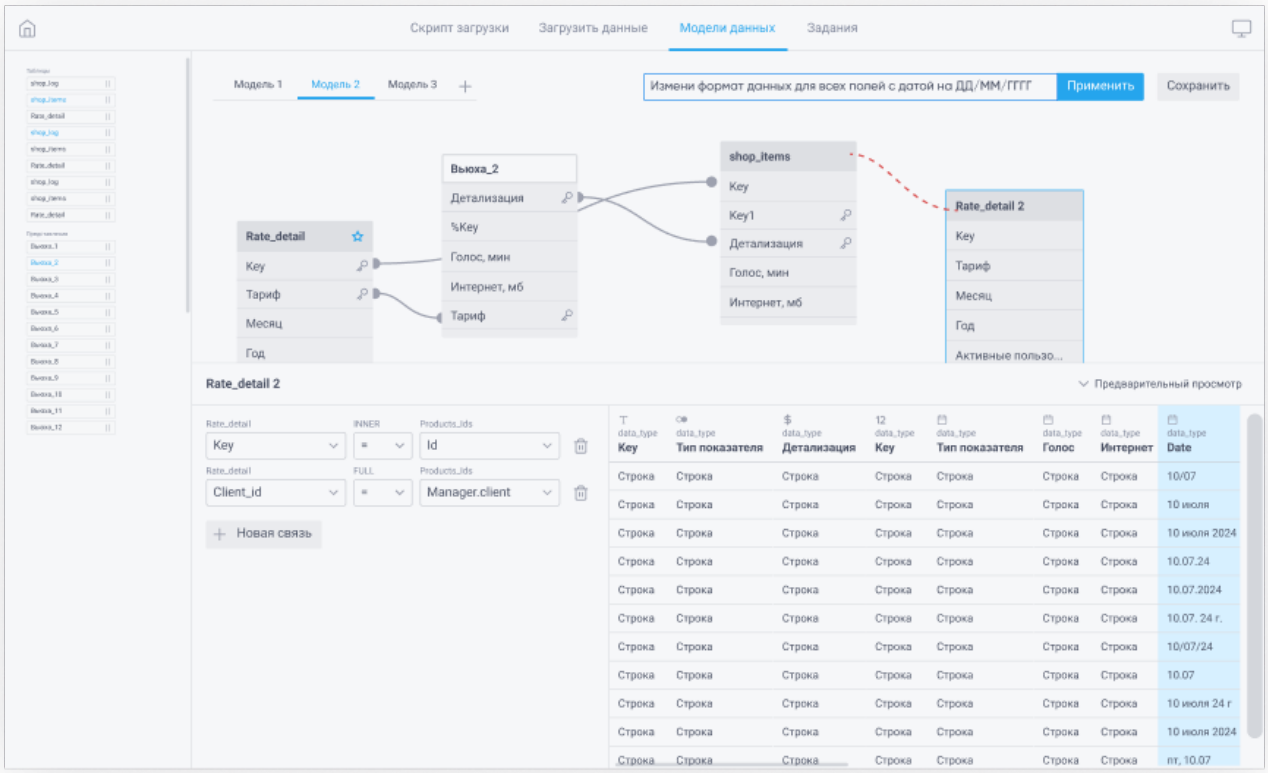
Задача:

Создание и внедрение в редактор модели данных модуля проверки качества данных с использованием языковой модели нейронной сети для обработки текстовых сообщений от пользователя.

Концепт:

Фронт: текстовая строка для ввода сообщения от пользователя с кнопками «Применить», «Сохранить» и «Отменить». При нажатии на кнопку «Применить» запускается обработка текстового сообщения пользователя языковой моделью. После обработки сообщения и внесения моделью изменений в имеющуюся таблицу в секции «Предварительный просмотр» отображается измененная таблица. При нажатии на кнопку «Сохранить» внесенные изменения вносятся в скрипт и меняют модель данных. При нажатии на кнопку «Отменить» (при условии, что не была нажата кнопка «Сохранить») все предложенные моделью изменения сбрасываются, таблица для предварительного просмотра возвращается в первоначальное состояние.

ML: языковая модель для обработки текстовых сообщений от пользователя. Языковая модель обрабатывает 2 вида запросов (и приводит к ним иные): задачи редактирования и задачи поиска. Превращает запрос от пользователя в SQL-код, который используется для создания демонстрационной таблицы или окна с ответом. Демонстрационная таблица хранится временно до тех пор, пока пользователь не подтвердит её использование, не отменит выбор или не выйдет из модели данных.



Функциональные требования:

Фронт: интеграция в модуль работы с моделями данных текстовой строки с 3 управляющими кнопками:

1. Текстовое поле имеет неограниченную длину по числу символов. При достижении видимой границы текстового поля справа оно должно автоматически расширяться вниз с переносом текста на следующую строку. Расширенное поле должно быть видно только при выборе строки запроса нажатием на него. Нажатие на любое пустое место на странице должно приводить к автоматическому скрытию поля до размеров первоначальной строки.

2. Кнопка «Применить» отправляет POST-запрос через API на бэкенд. В теле запроса находится текст из текстового поля и SQL-код из фронта. После нажатия на кнопку «Применить» другие кнопки (в т.ч. для перехода в скрипт загрузки или возвращения на дашборд) остаются недоступными до окончания работы модели. Из бэкенда возвращается SQL-запрос, который формирует временную таблицу для просмотра. Временная таблица добавляется в список таблиц, выносится на модель, курсор устанавливается на эту таблицу.

3. Кнопка «Сохранить». По умолчанию недоступна (некликабельна). Активируется после формирования таблицы предварительного просмотра. При нажатии на кнопку SQL-код интегрируется во фронт и происходит перезапись модели: сохранение названия старой таблицы – удаление старой таблицы – переименование временной таблицы – сохранение временной таблицы как обычной.

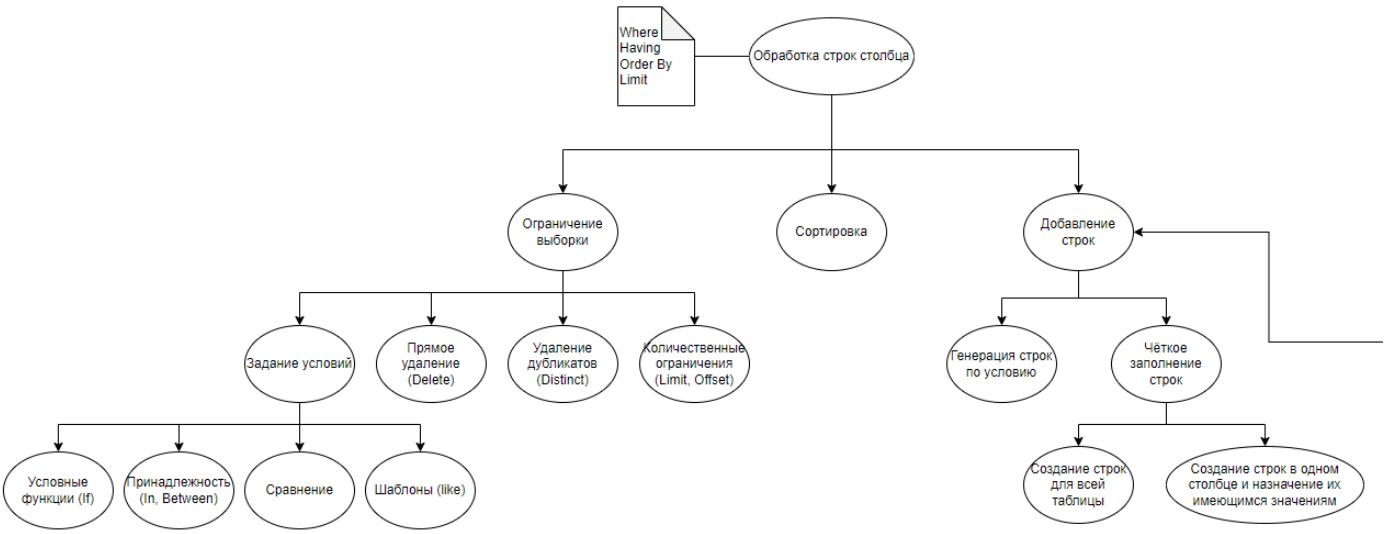
При нажатии на кнопку «Сохранить» должно появляться системное сообщение с кнопками для продолжения или отмены и текстом: "Применение изменений приведёт к перезаписи имеющейся таблицы. Все существующие связи между таблицами будут потеряны, их необходимо создать заново. Вы уверены, что хотите продолжить?"

4. Кнопка «Отменить» - удаляет временную таблицу, возвращает в таблицу предварительного просмотра результат работы скрипта без изменений от языковой модели.

ML: языковая модель принимает текстовый запрос и исходный SQL-код. Возвращает измененный SQL-код или текстовое сообщение. Решает задачи редактирования строк столбцов, столбцов таблицы, таблиц модели, а также задачи поиска.

При удачном срабатывании модели при решении задач редактирования должен отправляться только SQL-запрос. При решении задач поиска или при неудачном срабатывании модели должны отправляться текстовые сообщения с результатом поиска или ошибкой.

Ошибка: неправильно указано правило для решения поставленной задачи. Текст ошибки: "Неверно указано правило для обработки. Пожалуйста, измените запрос"



1. Обработка строк:

1.1. Ограничение выборки – уменьшение числа строк в столбце по одному из следующих принципов:

1.1.1. Прямое удаление.
Пример: "Удали все строки, где Расходы пустые".
ALTER TABLE Таблица DELETE WHERE Расходы is not null

1.1.2. Удаление дубликатов.
Пример: "Оставь только неповторяющиеся значения в столбце Доходы".

SELECT DISTINCT ... FROM Таблица ИЛИ
OPTIMIZE TABLE Таблица DEDUPLICATE

1.1.3. Количественные ограничения на число значений с начала / конца .
Пример: "Оставь только первые 100 строк в таблице".
SELECT ... FROM Таблица LIMIT / OFFSET Значение

1.1.4. Задание условий - ограничение выборки одним из следующих способов:

1.1.4.1. **Условные функции**.
Пример: "Если в столбце нет числовых значений, то оставь его без изменений. Иначе умножь все числовые значения на 100".
SELECT IF (Условие, Результат, Иначе)

1.1.4.2. Принадлежность – проверка на наличие значения в массиве / множестве / диапазоне. Использует операторы **IN, EXIST, BETWEEN**.
Пример: "Оставь только даты в диапазоне от января 2023 до апреля 2024".

1.1.4.3. **Сравнение**. Использует операторы для сравнения как с фиксированными значениями, так и для сравнения столбцов между собой.
Пример: "Оставь только положительные значения в столбце Прибыль"; "Оставь только строки, в которых Доход больше Расхода"

1.1.4.4. **Условия по шаблонам** – поиск по значениям строк в соответствии с заданным шаблоном.
Пример: "Оставь строки, где в названии проекта содержатся кавычки"

1.2. **Сортировка по столбцу**.
Пример: "Отсортируй столбец Количество по убыванию, пустые строки в конце".
ORDER BY Количество desc
Пример 2: "Отсортируй: задачи с большим отклонением факта от плана в начале".
ORDER BY Fact - Plan desc

1.3. Добавление / изменение строк – расширение или заполнение таблицы одним из следующих способов:

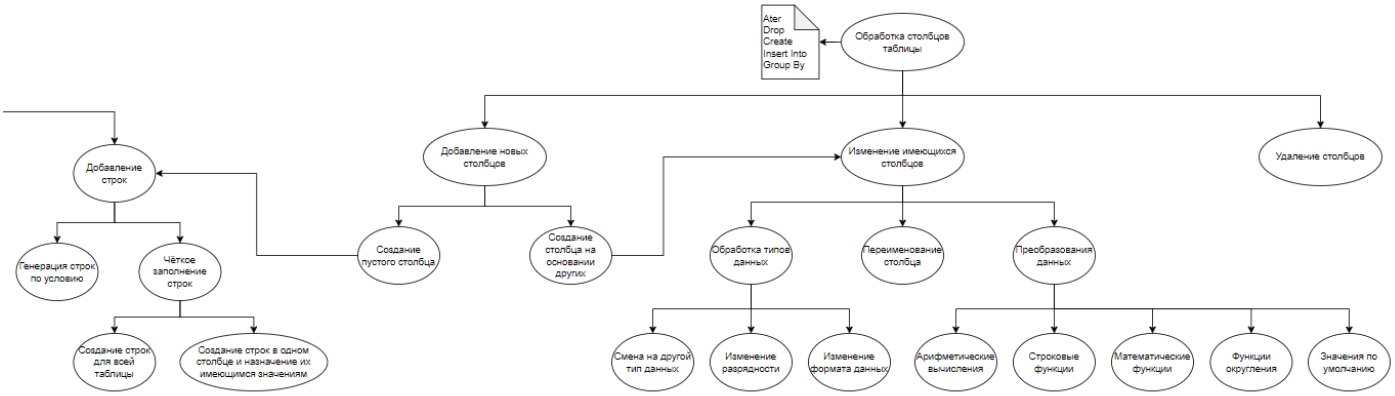
1.3.1. Генерация строк по условию – добавление новых строк с использованием генератора значений. Неуказанные столбцы должны заполняться пустыми значениями.
Пример: "Добавь в таблицу 50 строк, где Название компании – ООО "Домик", Доходы варьируются от 20000 до 30000 с равномерными шагами между значениями".
Должен формироваться массив строк, которые после будут добавлены с помощью INSERT INTO.
ВАЖНО! Задача нуждается в оценке затрат времени: если генерация строк

моделью с последующей вставкой в БД выполняется быстрее, чем в КХ, то выполнять таким образом. Иначе модель должна отдавать код SQL для генерации строк в КХ.

1.3.2. Чёткое заполнение строк – пользователь сам вводит все нужные ему значения:

1.3.2.1. Создание строк для всей таблицы. Не указанные поля заполняются пустыми значениями (NULL):
Пример: "Добавь две строки: Вася, апельсины, 5000, Краснодар. Миша, бананы, 10000, Сочи".
INSERT INTO TABLE Таблица VALUES (Значение столбца 1, Значение столбца 2, ..., Значение столбца n)
Пример 2: "Вставь новые значения: Пользователь Вася, Товар Апельсины, Количество 5000, Город Краснодар"
INSERT INTO TABLE Таблица (Пользователь, Товар, Количество, Город)
VALUES ('Вася', 'Апельсины', 5000, 'Краснодар')
Ошибка: в таблице нет столбцов для некоторых из указанных значений. Текст ошибки: "В таблице нет столбцов для некоторых из указанных значений. Пожалуйста, проверьте список доступных столбцов и измените запрос"
Ошибка: попытка подставить данные неподходящего типа. Текст ошибки: "Для данных {данные} выбран столбец с неподходящим типом. Пожалуйста, измените запрос"

1.3.2.2. Создание строк в одном столбце и назначение их имеющимся значениям. Аналогично предыдущему пункту, но заполняется только одно поле для заполненной строки.
Пример запроса: "Добавь название города Краснодар ко всем строкам, где цена больше 5000". Пример до выполнения: "Москва, 3000, яблоки", "NULL, 6000, бананы", "Ростов, 7000, апельсины". Пример после выполнения: "Москва, 3000, яблоки", "Краснодар, 6000, бананы", "Ростов, 7000, апельсины".
Альтернативный запрос: "Замени название города на Краснодар во всех строках, где цена больше 5000". Пример до выполнения: "Москва, 3000, яблоки", "NULL, 6000, бананы", "Ростов, 7000, апельсины". Пример после выполнения: "Москва, 3000, яблоки", "Краснодар, 6000, бананы", "Краснодар, 7000, апельсины".
Ошибки аналогичны предыдущему пункту.



2. **Обработка столбцов**:

2.1. Добавление новых столбцов (ADD COLUMN). Модель должна не только создавать столбец, но и самостоятельно определять его тип данных (при отсутствии запроса от пользователя) и выполнять проверку на существование столбца с указанным названием:

2.1.1. Создание пустого столбца.
Пример: "Добавь в таблицу Фрукты столбец Количество"
ALTER TABLE Таблица ADD COLUMN Столбец

2.1.2 Создание столбца на основании других – модель должна считать метаданные со столбца-основания, создать новый столбец, заполнить согласно запросу (подробно в п.2.2.). Использует MATERIALIZED – Материализованное вычисляемое выражение. Такое значение не передаётся при вставке, а вычисляется и сохраняется как физический столбец.
Пример: "Создай столбец Стоимость, в котором будет считаться произведение столбцов Цена и Количество"
ALTER TABLE Таблица ADD COLUMN Стоимость MATERIALIZED Цена * Количество
Альтернативный вариант – использовать ALIAS (В отличие от MATERIALIZE, результат вычисления не сохраняется как физический столбец, а вычисляется на лету при выборках из данного поля.):
ALTER TABLE Таблица ADD COLUMN Стоимость ALIAS Цена * Количество

2.2. Изменение столбцов:

2.2.1. Обработка типов данных (MODIFY COLUMN):

2.2.1.1. Смена на другой тип данных – модель должна выполнять проверку на возможность изменения типа данных столбца на указанный.
Пример: "Измени тип данных для столбца Количество с текстового на числовой"
Ошибка: пользователь предлагает неподходящий тип данных. Текст ошибки: "Невозможно изменить тип данных столбца на указанный. Пожалуйста, выберите подходящий тип данных"

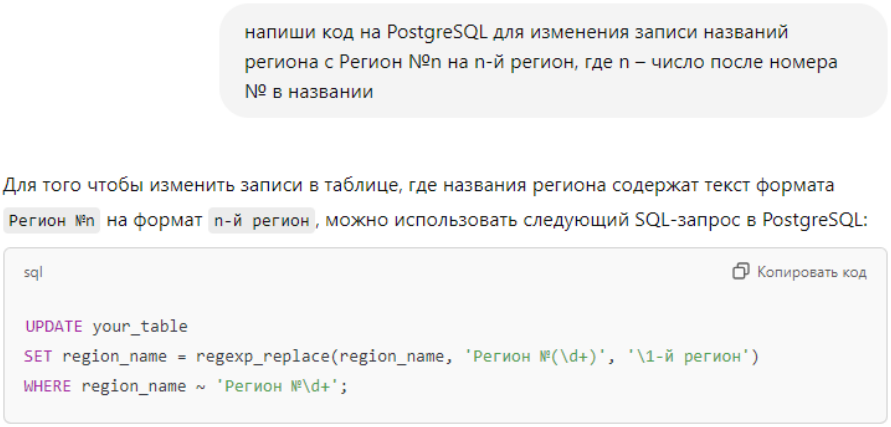
ALTER TABLE Таблица MODIFY COLUMN Столбец Тип данных

2.2.1.2. Изменение разрядности – модель должна проверять наличие свойства «разрядность» в типе данных столбца таблицы.
Пример: "Добавь два знака после запятой в столбце Расходы"
Ошибка: у типа данных выбранного столбца нет разрядности. Текст ошибки: "Изменить разрядность указанного столбца невозможно – это свойство отсутствует"

2.2.1.3. (???) Изменение формата данных – сложная функция, которая преобразует представление исходных данных в формат, удобный пользователю. Модель должна определять положение исходных данных по столбцам, точки разбиения, функцию разбиения и сборки нужного формата. При этом результат сложного запроса всегда будет строкой.
Пример: "Измени записи названий региона с Регион №n на n-й регион, где n – число после номера № в названии"

```
select id, concat (arr[2], '-й регион') as name from (select *, regexp_split_to_array(name, '№')
as arr from region r) f*
```

*запрос составлен на PostgreSQL и является тестовым
Запрос в ClickHouse:
ALTER TABLE Таблица UPDATE Регион = concat(regexp_substr(Регион, '\d+'), '-й регион')
Пример неудачной обработки запроса языковой моделью (установлен 1-й регион во всех названиях):



2.2.2. Переименование столбца (RENAME COLUMN). Используется и для изменения регистра в названии.
Пример: "Измени название столбца Фрукты на Овощи".
ALTER TABLE Таблица RENAME COLUMN Столбец TO Новое название

2.2.3. Преобразования данных:

2.2.3.1. **Арифметические вычисления**. Вычисление суммы, разности, произведения данных и т.д.

Пример: "Добавь ко всем числам, меньшим 5000, в столбце Доходы число 500"
Ошибка: над указанным типом данных невозможно проводить арифметические вычисления. Текст ошибки: "Над указанным типом данных невозможно проводить арифметические вычисления. Пожалуйста, измените запрос"

2.2.3.2. **Строковые функции**. Соединение и разъединение строк, поиск символов, определение длины и т.д.
Пример: "Соедини значения в столбце Название региона с его порядковым номером в столбце Id"
Ошибка: для указанного типа данных неприменимы строковые функции. Текст ошибки: "Для указанного типа данных неприменимы строковые функции. Пожалуйста, измените запрос"

2.2.3.3. **Математические функции**. Вычисление экспонент, логарифмов, поиск корней и т.д.
Пример: "Замени данные в столбце Наклон на синус от имеющихся значений в этом столбце"
Ошибка: для указанного типа данных неприменимы математические функции. Текст ошибки: "Для указанного типа данных неприменимы математические функции. Пожалуйста, измените запрос"

2.2.3.4. **Функции округления**.
Пример: "Округли значения в столбце Прибыль до 2 знаков после запятой"
Ошибка: для указанного типа данных неприменимы функции округления. Текст ошибки: "Для указанного типа данных неприменимы функции округления. Пожалуйста, измените запрос"

2.2.3.5. Установка **значений по умолчанию**.
Пример: "Установи в качестве базового значения в столбце Город значение Краснодар".

2.3. **Удаление столбцов** (DROP COLUMN).
Пример: "Удали столбец Фрукты"
Ошибка: столбца с указанным именем не существует. Текст ошибки: "Столбца с указанным именем не существует"

3. Обработка таблиц:

3.1. Изменение таблиц – любой запрос, содержащий задание на изменение таблицы, работает либо с её столбцами (п.2.), либо с её строками (п.1.)

3.2. **Добавление таблиц** (CREATE TABLE):

3.2.1. **С описанием структуры** – модель должна создавать пустую таблицу, в которой должны быть описаны столбцы, могут быть описаны первичный ключ и движок. Если движок не описан, то создаётся такой же, как и в редакторе скрипта по умолчанию (MergeTree).
Пример: "Создай таблицу Регионы с полями ID (ключевое) и Название".
Ошибка: не указано ни одно поле в новой таблице. Текст ошибки: "При создании новой таблицы необходимо указать хотя бы один столбец для создания"

3.2.2. **Со структурой, аналогичной другой таблице:**
Пример: "Создай таблицу Доходы на основании таблицы Расходы".
Ошибка: таблицы-основания не существует. Текст ошибки: "Вы указали несуществующую таблицу в качестве основания для создания новой. Пожалуйста, выберите существующую таблицу или проверьте корректность указанного названия таблицы"

3.2.3. **С автозаполнением по запросу** – модель должна формировать названия столбцов и их тип данных исходя из запроса на создание выборки:
Пример: "Создай таблицу Адресы, заполнив её названиями городов из таблицы Города и названиями улиц из таблицы Улицы".
Ошибка: таблицы-основания или столбцов из таблицы-основания не существует. Текст ошибки: "Вы указали несуществующую таблицу / несуществующие столбцы

в качестве основания для создания новой. Пожалуйста, выберите существующую таблицу / существующие столбцы или проверьте корректность указанного названия таблицы / столбцов"

3.3. Удаление таблиц (DROP TABLE).
Пример: "Удали таблицу Прибыль".
Ошибка: таблицы для удаления не существует. Текст ошибки: "Вы указали несуществующую таблицу для удаления. Пожалуйста, выберите существующую таблицу или проверьте корректность указанного названия таблицы"

4. Поисковые функции. На выходе модель возвращает текстовое описание элемента модели данных по запросу от пользователя (без SQL-кода):

4.1. Сколько раз встречается значение, указанное пользователем, в столбце / таблице.

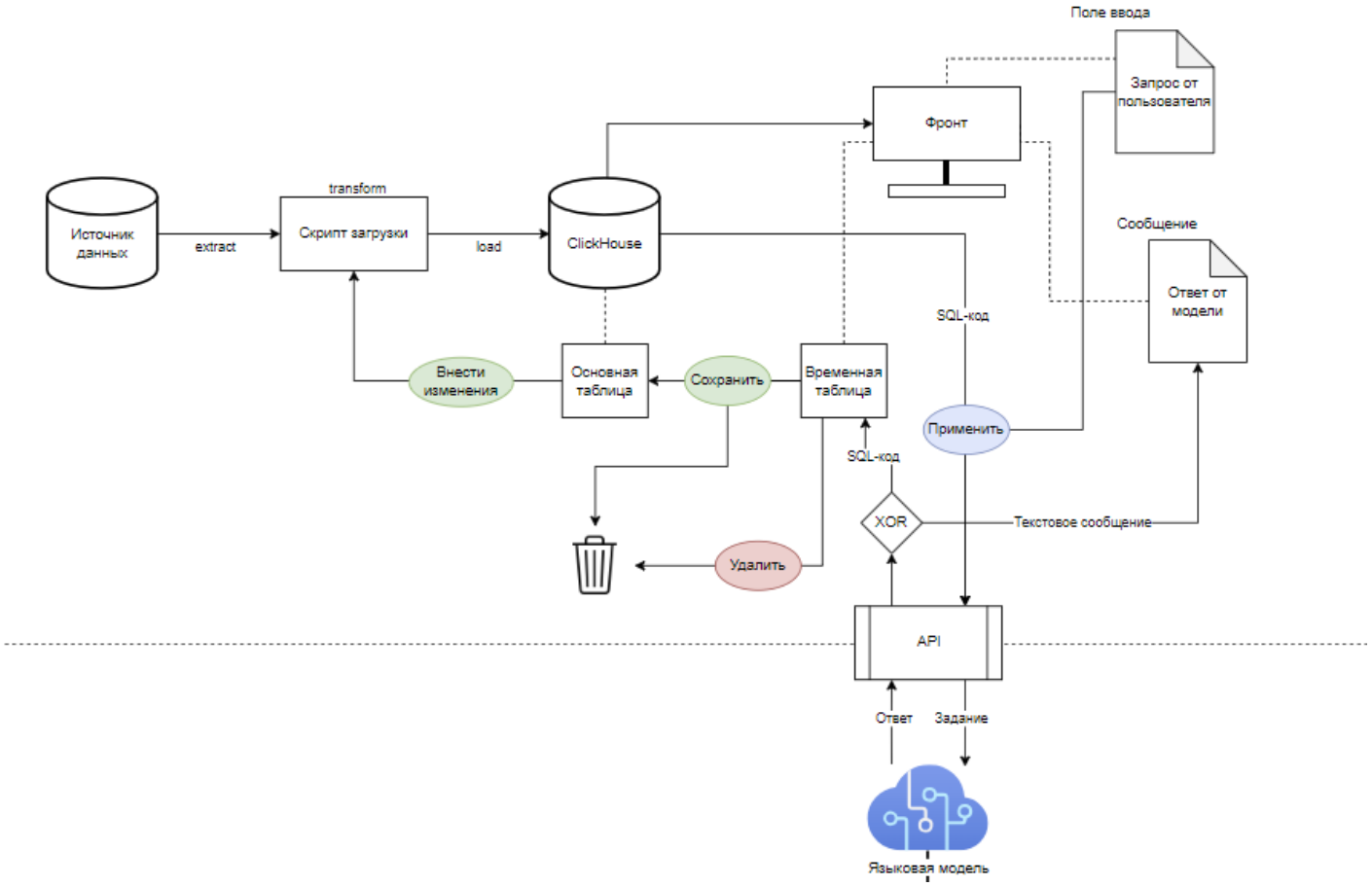
4.2. Наиболее часто / редко встречающееся значение, указанное пользователем, в столбце / таблице.

4.3. Наибольшее / наименьшее значение (самая длинная / короткая строка), указанное пользователем, в столбце.

4.4. Порядковый номер в столбце значения, указанного пользователем.

4.5. Количество значений указанного формата в столбце / таблице, или значений указанного типа в таблице.
Пример: "Сколько раз встречается слово Краснодар в столбце Город?".
Пример: "Сколько раз встречаются столбцы со строковым типом данных в таблице Продажи?".

Архитектура взаимодействия с моделью:

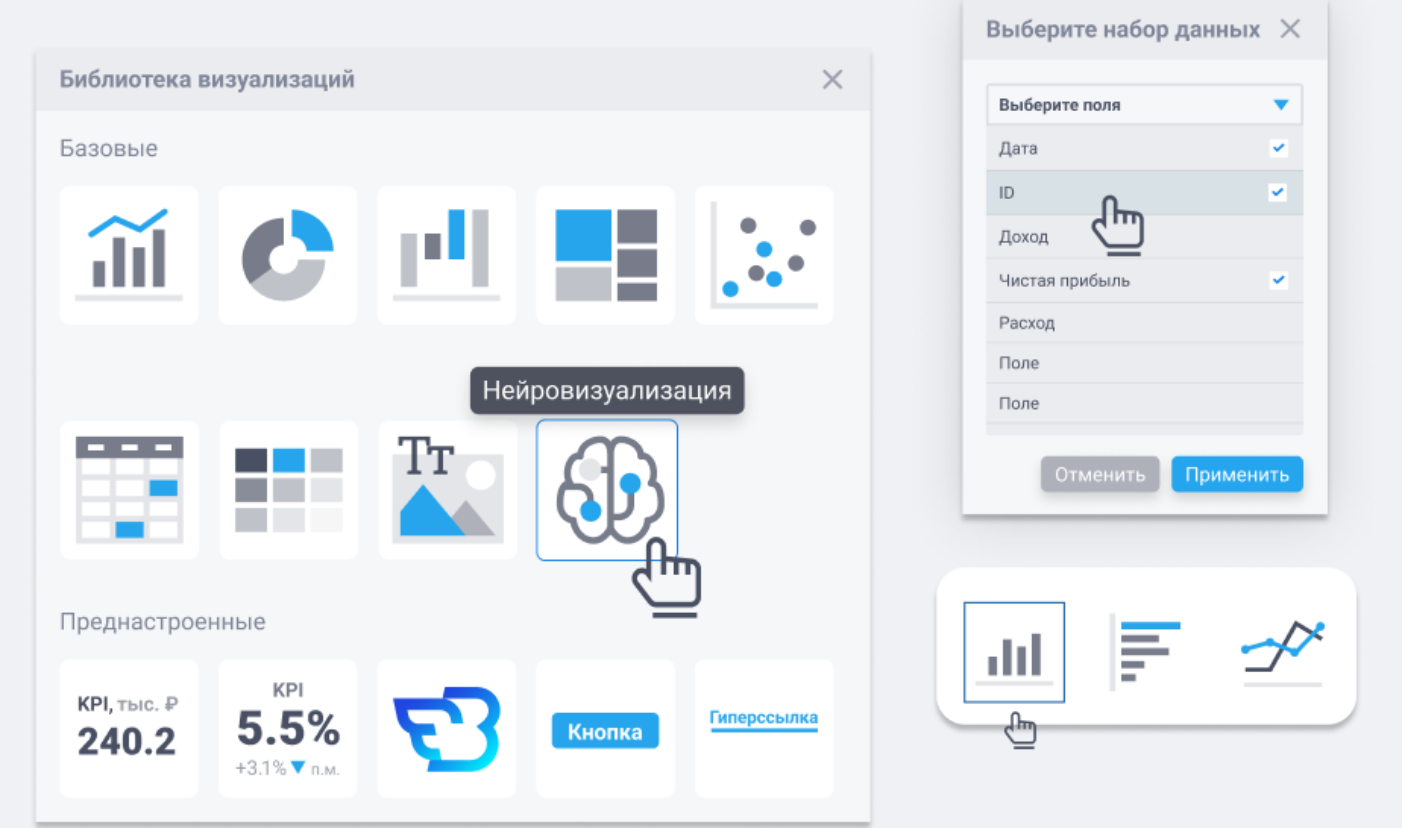

Поле ввода
Запрос от пользователя
Фронт
transform
Источник данных
extract
Скрипт загрузки
load
ClickHouse
Сообщение
Ответ от модели
SQL-код
Внести изменения
Основная таблица
Сохранить
Временная таблица
Применить
SQL-код
XOR
Текстовое сообщение
Удалить
API
Ответ
Задание
Языковая модель

Нейровизуализации

Цель

Интеллектуальная система, которая на основе выбранных пользователем столбцов из модели данных генерирует несколько логически обоснованных визуализаций и предлагает их на выбор. Каждая визуализация должна сопровождаться подписью с указанием использованных типов полей и контекста, в котором она может использоваться (сравнение, распределение, состав или отношение)

Концепт для интерфейса



В список визуализаций, доступных для выбора на дашборде, необходимо добавить модуль **«Нейровизуализация»**. При нажатии на этот модуль в окне должен открываться список с доступными моделями данных на выбор и полями в этих моделях. Пользователь может выбрать до 4-х полей, после чего система автоматически предложит ему от 2 до 6 наиболее подходящих вариантов визуализации на основе предоставленных данных.

Концепт ML

В модели используются 4 расходящиеся ветки для определения контекста выбранных данных. Нужна модель ИИ или алгоритм с использованием разведочного анализа данных, позволяющие выполнить задачу классификации входных данных на наиболее подходящие из следующих классов задач: Сравнения, Распределения, Состава (Структуры) или Отношения. На входе модель получает от пользователя только столбцы данных (от 1 до 4) из итоговой таблицы. Возможно подключение вероятностной модели для определения наилучшего контекста.

После определения контекста задачи требуется применение алгоритма для выбора конкретной визуализации. На выходе модель должна предоставлять следующие данные:

- Наборы столбцов (от 2 до 6) с разделением на показатели (данные для отображения) и разрезы (поля для группировки). Для начала типом агрегации для передачи в модель для всех данных будет "Сумма"!
- Тип визуализации (диаграмма, водопад и т.д.) для каждого набора столбцов для отрисовки на фронте.
- Контекст использования одним словом (1 из 4 классов задач, определенных моделью) к каждому набору столбцов.

Задание на ML-разработку

Задача классификации

На вход модели поступают от 2 до 4 столбцов из модели данных. Тип данных у каждого столбца может быть одного из двух видов:

- **Числовой** – типы данных Int32, Int64 и т.д., Float, Decimal и иные целочисленные и вещественные типы данных, для которых определены арифметические (сложение, вычитание, умножение, возведение в степень) и логические (больше, меньше, равно, в промежутке) операции. В основном выступают в качестве показателей.
- **Категориальный** – строковые (String, FixedString), булевы типы данных, дата и время, списки, массивы, а если проще – все остальные, нечисловые типы данных. Не используются в вычислениях (за исключением даты и времени) в визуализациях, выступают чаще в качестве разрезов.

При необходимости модель может извлекать из данных статистическую информацию: математическое ожидание, дисперсию, отклонения и т.д.

Основная задача модели – на основании имеющихся данных вероятностно классифицировать наборы столбцов по следующим классам:

- **Сравнение** – пользователь смотрит на разницу (динамику) каждого показателя (числового типа) в разрезе категориальных данных (сравнение по разрезам). Пример: динамика цен на продукты по месяцам; разница в поголовье скота на разных фермах; уровень заработной платы по сотрудникам; динамика курса валют за последний год и т.д.
- **Отношение** – пользователь смотрит на различия между совокупностями показателей (данными числового типа) для одной категории (сравнение по показателям). Пример: показатели роста и веса у студентов; содержание белков, жиров и углеводов в молочных продуктах; соотношение память – количество ядер – диагональ для моделей ноутбуков; количество продаж и прирост выручки фирм в регионе и т.д.
- **Состав или структура** – пользователь смотрит на эффект (различия, прирост, долю) отдельных категорий по конкретному числовому показателю (возможно, составному). Пример: доли продаж дочерних компаний в общем объёме продаж холдинга; прибыль/убыток от продаж магазина по категориям товаров; площадь, занимаемая каждым жилым комплексом в доле от общей площади строительства по городам, регионам, странам и т.д.
- **Распределение** – пользователь смотрит на размещение категориальных признаков согласно их числовым показателям. Пример: карта конкурентов (разделение на 4 категории по числовым осям и распределение на ней компаний в зависимости от их показателей); распределение затрат на отделы компании; результаты соревнований по бегу для определения победителя.

> Распределение в целом очень похоже и на сравнение, и на отношение (даже по выбору визуализаций). Тем не менее, если неверно отнести набор данных не к этому классу, то дальнейшее применение алгоритма по подбору визуализации может дать негативный результат.
> В отличие от отношения для распределения важнее визуализировать то, к какой категории принадлежит какой показатель, чем то, как показатели соотносятся между собой (к какой группе отнести конкурента, а не какой конкурент доминирует над

> остальными); от сравнения – чем то, какая разница между показателями разных категорий (какому спортсмену выдать медаль, а не выявить отставание конкретного спортсмена от победителя).

Наборы столбцов необходимо собирать по всем доступным вариантам из тех, что предоставил пользователь. Если на входе модель получает:

- **2 столбца** от пользователя, то есть только **1 вариант** их сочетания;
- **3 столбца** от пользователя, то есть **4 варианта** их сочетания (3 пары из двух столбцов и 1 тройка);
- **4 столбца** от пользователя, то есть **11 вариантов** их сочетания (6 пар, 4 тройки и 1 четвёрка).

Для КАЖДОГО варианта сочетания столбцов модель должна рассмотреть ВСЕ 4 контекста использования (класса). В итоге получится **от 4 до 44 возможных выходных результатов** вида "набор данных – класс".

Далее необходимо расставить вероятности для каждого из выходных результатов. Для решения именно этой задачи требуется модель ИИ. Модель должна проанализировать метаданные каждого столбца из набора данных, рассчитать все необходимые дополнительные величины (например, статистические параметры) и определить, насколько каждый класс подходит каждому набору данных в процентном соотношении.

Важным требованием является наличие **границы отсечения** процентно-незначимых наборов выходных данных. Значение этой границы можно получить опытным путём или установить искусственно с возможностью дальнейшей корректировки в процессе использования модели (например, установить границу равной 85%). После получение процентного соответствия классов наборам столбцов возможен один из следующих результатов сравнения с границей:

- **Не более чем 1** набор данных прошёл границу отсечения. В таком случае принимаем удовлетворительными два набора данных с наибольшими значениями процентного соответствия выбранному классу.
- Границу отсечения прошли **от 2 до 6** (включительно) наборов данных. В таком случае принимаем удовлетворительными именно эти наборы данных.
- Границу отсечения прошли **более 6** наборов данных. В таком случае принимаем удовлетворительными шесть наборов данных с наибольшими значениями процентного соответствия выбранному классу.

Удовлетворительные наборы данных вместе с контекстом использования (классом) отправляются на вход алгоритма выбора визуализации.

Алгоритм выбора визуализации

* **Тип данных** читать как **столбец с типом данных** (н-р, числовой тип данных в показателе = столбец с числовым типом данных в показателях).

Представляет собой обычное дерево решений, первая ветка которого определяется классом набора данных. Дальнейший выбор зависит от типов данных набора и некоторых других характеристик, записанных условиями.

> До перехода на одну из веток дерева решений должна производиться проверка: если все выбранные пользователем столбцы имеют категориальный тип данных (не числовой), то вместо перехода на любую из веток эти данные собираются в одну таблицу (в разрезах) без других вариантов. Вместо класса ставится подпись: "Для получения корректного результата необходимо выбрать хотя бы одно числовое поле".

> Аналогичная проверка должна выполняться и для числовых типов данных, но с одним исключением: если хотя бы в одном столбце из набора не более 12 уникальных значений, то набор данных необходимо пропустить на одну из веток (этот числовой столбец будет играть роль категориального). Если таких столбцов нет, то все данные собираются в одну таблицу (в разрезах) без других вариантов. Вместо класса ставится подпись: "Для получения корректного результата необходимо выбрать хотя бы одно категориальное поле".

> Если на выходе модели представлено более одной таблицы, то из всех таблиц должна остаться только одна (с наибольшим числом столбцов), а остальные – быть удалены.
> *Если на выходе есть и таблицы, и другие визуализации, то из таблиц остаётся одна, а другие визуализации проходят без изменений

Сравнение:

1. Есть столбцы категориального типа И Все столбцы категориального типа имеют тип данных "дата (дата и время)" И Уникальных значений в категориальных типах НЕ более 12:
    1. Только один столбец числовой:
        - Визуализация **Комбинированная**; тип графика – **линия**, категориальные типы данных в разрезах, числовой тип данных в показателе (один линейный график).
    2. Несколько столбцов числовые (хотя бы один – категориальный):
        - Визуализация **Комбинированная**; тип графика – **линия**, категориальные типы данных в разрезах, числовые типы данных в показателях (несколько линейных графиков).
2. Нет столбцов категориального типа ИЛИ Есть столбцы категориального типа, отличающегося от типа данных "дата (дата и время)" ИЛИ Уникальных значений в категориальных типах более 12:
    1. Только один столбец числовой:

- Визуализация **Комбинированная**; тип графика – **столбик**, категориальные типы данных в разрезах, числовой тип данных в показателе (одна столбчатая визуализация).

2. Несколько столбцов числовые:
   - Визуализация **Комбинированная**; тип графика – **столбик**, категориальные типы данных в разрезах, числовые типы данных в показателях (несколько столбчатых визуализаций).
3. Все столбцы – числовые:
   1. Хотя бы в одном столбце НЕ более 12 уникальных значений:
      - Визуализация **Комбинированная**; тип графика – **столбик**, ОДИН числовой тип данных с наименьшим числом уникальных значений в разрезах, остальные числовые типы данных в показателях (одна или несколько столбчатых визуализаций).
   2. Во всех столбцах более 12 уникальных значений:
      - Визуализация **Таблица**; числовые типы данных в разрезах.

Отношение:

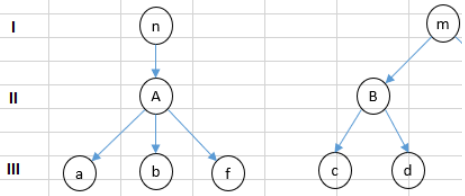
1. В наборе только один столбец с числовым типом данных ИЛИ более одного столбца с категориальным типом данных (при наличии хотя бы одного с числовым):
   - **СООБЩИТЬ ОБ ОШИБКЕ!** – модель неверно классифицировала контекст как Отношение. Исключить ветку из рассмотрения.
2. В наборе два столбца с числовым типом данных и один столбец с категориальным:
   - Визуализация **Пузырьковая**; категориальный тип данных – в разрезе, числовые типы данных – в показателях (без размера пузырьков).
3. В наборе три столбца с числовым типом данных и один столбец с категориальным:
   - Визуализация **Точечная**; категориальный тип данных – в разрезе, числовые типы данных – в показателях, один (случайным образом определенный) – в размере пузырька.
4. В наборе нет столбцов с категориальным типом данных (все столбцы – числовые):
   1. В наборе три столбца и хотя бы в одном столбце НЕ более 12 уникальных значений:
      - Визуализация **Пузырьковая**; ОДИН числовой тип данных с наименьшим числом уникальных значений в разрезах, остальные числовые типы данных в показателях (без размера пузырьков).
   2. В наборе четыре столбца и хотя бы в одном столбце НЕ более 12 уникальных значений:
      - Визуализация **Точечная**; ОДИН числовой тип данных с наименьшим числом уникальных значений в разрезах, остальные числовые типы данных в показателях, один (случайным образом определенный) – в размере пузырька.
   3. В наборе суммарно один или два столбца:
      - **СООБЩИТЬ ОБ ОШИБКЕ!** – модель неверно классифицировала контекст как Отношение. Исключить ветку из рассмотрения.

Состав или структура:

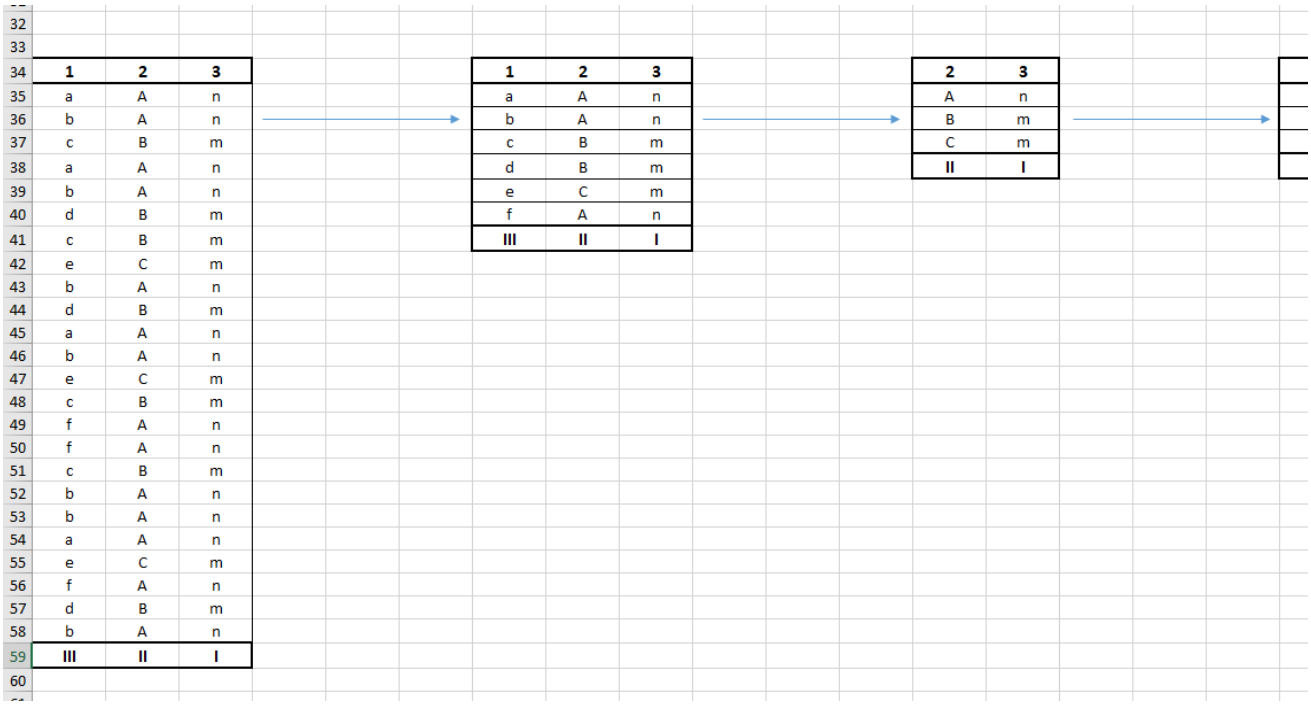
1. В наборе только один столбец с числовым типом данных:
    1. Алгоритмы на выявление **иерархических связей** в наборе данных, при которых каждому дочернему элементу соответствует только один родительский (можно использовать любой другой, вписывающийся в концепцию):
    Алгоритм 1: оставляем только категориальные столбцы, удаляем дубликаты строк (только для алгоритма), выполняем перебор всех возможных комбинаций категориальных столбцов. Если найдётся хотя бы одна комбинация, в которой каждому уникальному значению n-го столбца будет соответствовать только одно уникальное значение (n+1)-го столбца, то алгоритм выполнен.

1	2	3
a	A	n
b	A	n
c	B	m
a	A	n
b	A	n
d	B	m
c	B	m
e	C	m
b	A	n
d	B	m
a	A	n
b	A	n
e	C	m
c	B	m
f	A	n
f	A	n
c	B	m
b	A	n
b	A	n
a	A	n
e	C	m
f	A	n
d	B	m
b	A	n
III	II	I

1	2	3
a	A	n
a	A	n
a	A	n
a	A	n
b	A	n
b	A	n
b	A	n
b	A	n
b	A	n
b	A	n
f	A	n
f	A	n
f	A	n
c	B	m
c	B	m
c	B	m
c	B	m
d	B	m
d	B	m
d	B	m
e	C	m
e	C	m
e	C	m
III	II	I



    Алгоритм 2: оставляем только категориальные столбцы, удаляем дубликаты строк выполняем перебор всех возможных комбинаций категориальных столбцов. Проверяем первый столбец в комбинации. Если все значения в нём уникальные, то рассматриваем набор данных без этого столбца и переходим ко второму (по счёту в изначальном наборе) столбцу. Удаляем все дубликаты строк из таблицы, если все оставшиеся во втором столбце значения уникальные, то рассматриваем набор без этого столбца и переходим к третьему. Продолжаем до последнего столбца. Если удастся хотя бы в одной комбинации сократить выборку до последнего столбца, то алгоритм выполнен.



1	2	3
a	A	n
b	A	n
c	B	m
a	A	n
b	A	n
d	B	m
c	B	m
e	C	m
b	A	n
d	B	m
a	A	n
b	A	n
e	C	m
c	B	m
f	A	n
f	A	n
c	B	m
b	A	n
b	A	n
a	A	n
e	C	m
f	A	n
d	B	m
b	A	n
III	II	I

1	2	3
a	A	n
b	A	n
c	B	m
d	B	m
e	C	m
f	A	n
III	II	I

2	3
A	n
B	m
C	m
II	I

Если алгоритм выполнен полностью:

- Визуализация **Дерево**; категориальные типы данных в разрезах в порядке, обратном порядку в успешном алгоритме (сначала родительские, потом дочерние), числовые типы в показателях.

2. Если алгоритм не выполнен, то:
   - Визуализация **Круговая**; категориальные типы данных в разрезах, числовые типы в показателях.

2. В наборе РОВНО два числовых столбца и один категориальный:
   - Визуализация **Водопад**; категориальный тип данных в разрезе, если есть столбцы с названием "план" и/или "факт", то используем их в одноименных показателях, иначе: первый столбец в показателе "план", второй – в показателе "факт".
3. В наборе больше одного числового столбца И не выполнен пункт 2:
   1. Хотя бы в одном категориальном столбце столбце более 12 уникальных значений:
      - Визуализация **Комбинированная**; тип графика – **столбик**, объединить столбцы, категориальные типы данных в разрезах, числовые типы данных в показателях (накопительные столбчатые визуализации).
   2. Во всех столбцах более 12 уникальных значений:
      - Визуализация **Комбинированная**; тип графика – **линия**, установить непрозрачность области равной 50%, категориальные типы данных в разрезах, числовые типы данных в показателях (диаграмма области).

Распределение:

1. В наборе больше двух числовых столбцов:
   - **СООБЩИТЬ ОБ ОШИБКЕ!** – модель неверно классифицировала контекст как Распределение. Исключить ветку из рассмотрения.
2. В наборе один числовой столбец:

1. Хотя бы в одном категориальном столбце столбце более 12 уникальных значений:
    - Визуализация **Комбинированная**; тип графика – **столбик**, категориальные типы данных в разрезах, числовые типы данных в показателях (несколько столбчатых визуализаций).
2. Во всех столбцах более 12 уникальных значений:
    - Визуализация **Комбинированная**; тип графика – **линия**, установить толщину линии равной 0, толщину точки равной 6, категориальные типы данных в разрезах, числовые типы данных в показателях (точечный график).
3. В наборе два числовых столбца:
    - Визуализация **Пузырьковая**; категориальный тип данных – в разрезе, числовые типы данных – в показателях (без размера пузырьков).

Выход модели

Модель передаёт на фронт следующую информацию:

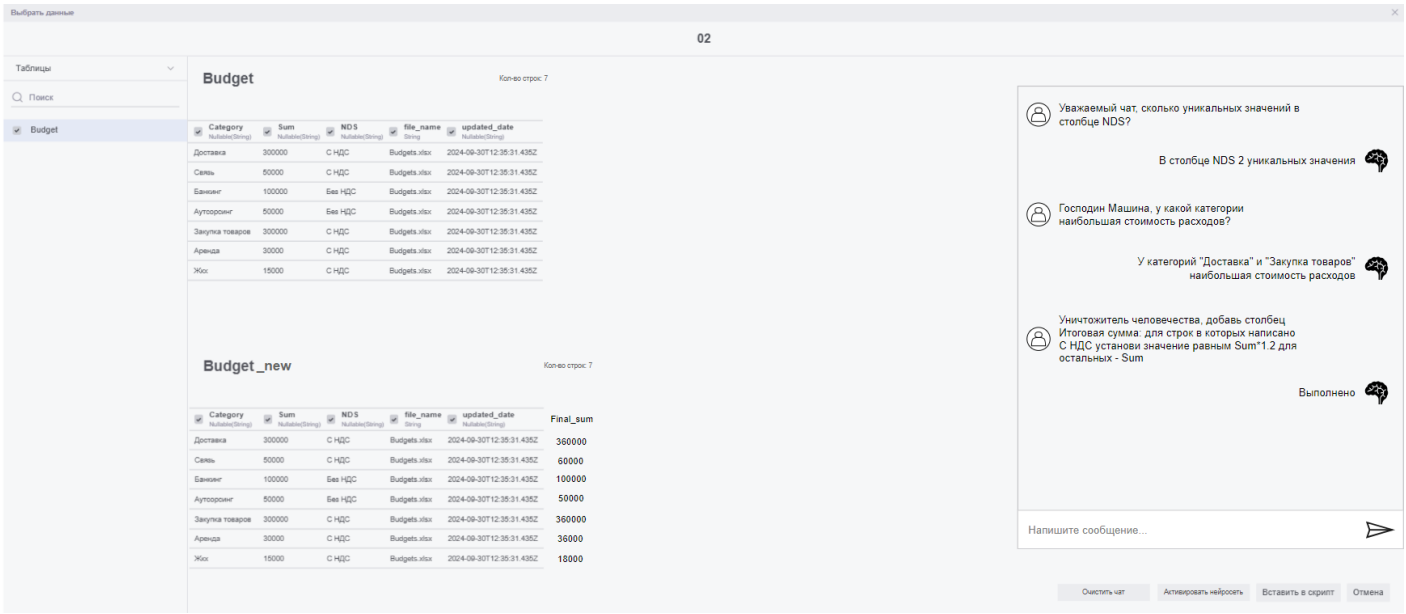
- Итоговые наборы данных, которые прошли границу отсечения классификатора и получили визуализацию;
- Контекст использования – один из четырёх присвоенных классов;
- Тип визуализации;
- Характеристики визуализации (распределение данных по разрезам и показателям, тип графика, толщина линии и т.д.)

Нейроконнекторы

Цель

Создание инструмента для **Data Discovery** в секции подключения, обеспечивающего удобное для пользователя изучение набора данных и формирование скрипта загрузки без необходимости использовать SQL.

Концепт для интерфейса



- Кнопка для активации/деактивации ИИ модели.
- При активации появляется редактируемая таблица под исходной и разблокируется чат с моделью справа от таблиц. При неактивной модели писать в чат невозможно, редактируемая таблица скрыта.
- Исходная таблица блокируется для изменений на время работы модели.
- Одновременно редактировать можно только одну таблицу, но вопросы остаются по всем таблицам в подключении (в чате могут быть запросы к нескольким таблицам последовательно, но не параллельно).
- В чате можно обмениваться сообщениями с моделью. Если направлено задание на изменение данных, то модель отправляет код для редактирования временной таблицы и даёт ответ вида "Выполнено"/"Невозможно выполнить". Если модели задан вопрос, она даёт полноценный текстовый ответ на основании рассматриваемых данных.

- При деактивации модели (нажатием на кнопку) все предложенные ИИ изменения отменяются, однако чат остаётся и его можно возобновить кнопкой для активации модели. Переписка в чате сохраняется для каждого отдельного подключения.
- Должна быть кнопка, позволяющая очистить чат перед выходом.
- При активной ИИ модели кнопка "Вставить в скрипт" добавляет код для редактируемой нейросетью таблицы, а не для исходной.

Концепт ML

Языковая модель, способная обрабатывать запросы от пользователя и давать ответ либо преобразовывать их в код на языке ClickHouse. Должна уметь распознавать и выполнять следующие типы запросов:

- Поиск в данных: по запросу от пользователя модель должна уметь находить необходимые столбцы, строки и значения, сравнивать их с заданными, запоминать введённые условия, находить максимум, минимум, среднее и т.д. Пример: "Напиши, есть ли в столбце Cost значения больше 20000?".
- Анализ использования: модель должна уметь определять контекст использования данных из источника, назначение отдельных столбцов, область применения (задачи, которые можно решать с помощью данных). Пример: "Объясни, для чего тут столбец Month_num, если есть столбец Month?" или "Достаточно ли этих данных, чтобы проанализировать динамику расходов за 2024 год?".
- Работа с типами данных: модель должна уметь определять и подставлять подходящий тип данных столбца. Пример: "Выбери наиболее подходящий тип данных для столбца Cost" или "Ты неправильно определил тип данных столбца Cost. Смени его на вещественный".
- Обработка данных: модель должна уметь писать код для трансформации таблицы, изменения имеющихся столбцов, создания новых на основании запроса (в т.ч. с использованием агрегирующих функций). Пример: "Добавь столбец Profit как результат выражения (Price - Cost) * Count - 5000" или "Транспонируй текущую таблицу".
- Фильтрация и ограничение: модель должна уметь оставлять в наборе данных только те столбцы, строки и значения, которые удовлетворяют запросу от пользователя. Пример: "Удали все значения больше 20000" или "Оставь только первые 5000 строк".

Задание на разработку ML

При обработке запросов модель должна уметь определять столбцы и таблицы, даже если пользователь совершает ошибки в написании их названий либо пишет их на другом языке.

>

> При обучении модели необходимо учитывать, что организация работы с данными происходит на ClickHouse, поэтому особенно важно обрабатывать особенности этого языка запросов, такие как использование сортировки для индексации и специфичность наименований типов данных.

> **ВАЖНО!** Нейросеть не работает с готовым SQL-кодом или таблицей (не создаётся секции ALTER и запросов на изменение). Её задача - написать код для формирования модели данных как результата выборки из БД ClickHouse по шаблону, подставляя в него фрагменты, позволяющие решить поставленную пользователем задачу.

Вход модели:

Таблицы со всеми столбцами и значениями в них + промпт от пользователя. Задачи от пользователя отправляются последовательно через чат, в то время как исходные таблицы неизменны.

Необходимо учитывать, что число значений в таблице может превышать десятки и сотни миллионов при том, что ограничивать или сэмплировать выборку нельзя (поскольку результат работы модели будет использован в итоговом скрипте), но для демонстрационного варианта отображаются только первые 10 строк.

Модель должна определить решаемую задачу в зависимости от запроса пользователя. Если пользователь неудовлетворен результатом работы модели, то нейросеть должна пересмотреть решаемую задачу.

Поиск в данных ("Что и где находится?")

Не возвращает SQL-кода ни для таблицы-примера, ни для добавления в скрипт загрузки.

Модель должна по запросу от пользователя определить зону поиска в данных (локализовать до столбца, набора столбцов, интервала строк и т.д.) после чего выполнить сам запрос на поиск и написать ответ в чате. Решаемые в рамках данной задачи запросы *могут* включать в себя:

- Поиск конкретного значения (числового или текстового).
  Запрос вида: "Есть ли среди Компаний AVA?".
  Возвращает ответ вида: "{Количество} {Значение} есть/отсутствует в {Зона поиска}" – "AVA есть среди компаний".
- Вывод всех уникальных значений в столбце. При первом запросе модель должна выдавать 10 самых часто встречающихся результатов с припиской "и ещё n", где n: (число уникальных значений)-10. При повторном запросе от пользователя выдаёт все значения через запятую (даже если их 100500).
- Поиск/подсчёт значений из диапазона/в диапазоне (набор строк, элемент строки, диапазон числовых значений).

Запрос вида: "У скольких работников из таблицы зарплата больше 100 000?".
Возвращает ответ вида: "{Количество} {Значение} есть/отсутствует в {Зона поиска}" – "У 28 работников зарплата больше 100 000".

- Поиск агрегированного значения из числа следующих: минимум, максимум, среднее, сумма, разность, произведение, частное, статистические функции (мода, медиана, дисперсия, ско).
  Возвращает предупреждение о больших затратах времени на выполнение операции вида "Этот запрос может выполняться длительное время. Напишите "да", если вы уверены, что хотите получить результат. Иначе можете продолжить чат".
  Запрос вида: "Найди проект с наибольшими затратами".
  Если пользователь напишет "Да", то модель должна выдать ответ вида: "{Запрашиваемое агрегированное значение} в {Зона поиска} – {Результат работы модели}" – "Проект с наибольшими затратами – AVA". Иначе продолжает отвечать на другие запросы.
- Другие запросы на поиск в данных.

Вид ответа может меняться в зависимости от сути вопроса пользователя. Модель должна определять последовательность логики запроса (когда пользователя интересует показатель, а когда - его аналитический разрез). Например, ответом на вопрос: "В каком городе наибольшее число жителей?" должно быть название города, а результатом поиска по запросу: "Найди наибольшее число жителей среди представленных городов" – число жителей.

Анализ использования ("Зачем и где применяется?")

Не возвращает SQL-кода ни для таблицы-примера, ни для добавления в скрипт загрузки.

Модель должна по запросу от пользователя определить и описать в ответе в чате контекст применения набора данных: решаемую задачу, экономическое применение, назначение столбцов.

> Для решения этой задачи необходимо анализировать не только метаданные (названия таблиц и столбцов), но и содержимое этих таблиц.

- Решаемая задача – ответ на вопросы вида: "Что я могу сделать с этими данными?".
  Модель должна определять спектр задач по типу: "Вы можете использовать эти данные для определения финансовых показателей компании за 2024 год".
- Экономический контекст – ответ на вопросы вида: "Где используются эти данные?".
  Модель должна определять сферу применения по типу: "Предположительно, эти данные используются в сфере продаж".
- Назначение столбцов – детализация решаемой задачи в области применения отдельных столбцов. Ответ на вопросы вида: "Зачем мне нужен этот столбец?".
  Модель должна определять назначение столбцов по типу: "Это вспомогательный столбец, который нужен для организации корректной сортировки".
- Другие запросы на выявление области применения данных.

Работа с типами данных

Типы данных

Модель должна уметь:

- Определять текущий тип данных указанного пользователем столбца. Эта информация передаётся вместе с метаданными, поэтому для решения этой задачи необходимо лишь передать в ответе имеющееся значение.
- Определять наиболее подходящий тип данных. Причём самый простой из подходящих (не выбирать словари или LowCardinality). Для решения этой задачи модель должна обрабатывать имеющиеся в столбце данные и на их основании предлагать нужный тип.
- Менять тип данных. Менять можно только в том случае, если значения столбца можно хранить в конечном типе данных. Модель должна уметь переводить данные как в конкретный тип ("Измени тип данных столбца Доход на вещественный"), так и в подходящий ("Измени тип данных столбца Расход на наиболее подходящий"). Измененный тип данных попадает в итоговый скрипт загрузки.

Обработка данных

Модель должна уметь изменять имеющиеся столбцы, создавать новые, сортировать по указанным столбцам.

- Изменение столбцов – модель записывает выражение на языке ClickHouse для текстового запроса от пользователя на редактирование существующего столбца. В качестве выражения может выступать комбинация арифметических и агрегирующих операций над столбцами, числами и текстом. Результат – вместо передачи названия существующего столбца в скрипт загрузки передаётся это выражение.
  Пример: "Сделай так, чтобы в столбце Сумма результат выводился в миллионах, а не в 1000".
  Результат (в Select): toFloat32("Sum")/1000.
- Создание столбцов – модель записывает выражение на языке ClickHouse для текстового запроса от пользователя на формирование нового столбца. В качестве выражения может выступать комбинация арифметических и агрегирующих операций над столбцами, числами и текстом. Результат – после списка с названиями существующих столбцов добавляется это выражение.
  Пример: "Создай столбец Дельта как разность между доходами и расходами".
  Результат (в Create): "Delta" Int32.
  Результат (в Select): toInt32("Income")-toInt32("Cost").
- Переименование столбцов – в секции CREATE вместо существующего имени столбца модель добавляет введенное пользователем.
- Сортировка столбцов – модель добавляет секцию ORDER BY согласно запросу от пользователя.

Фильтрация и ограничение

Модель должна уметь конструировать запросы для секций WHERE, LIMIT и OFFSET:

- Ограничение по условию – модель формирует секцию WHERE: интерпретирует текст от пользователя как выражение, содержащее оператор сравнения, принадлежности или шаблон для столбцов, чисел и текста.
  Пример: "Оставь только те значения в столбце Доход, которые больше 500 000"
  Результат (в Where): toInt32("Income")>500000
- Ограничение по числу значений:
  - "Оставь только первые N значений" – модель добавляет секцию LIMIT N.
  - "Оставь последние N значений" – модель добавляет секцию OFFSET N.
  - "Оставь значения, начиная с N и заканчивая M" – модель добавляет секции LIMIT N и OFFSET M.

> Если пользователь пытается ввести запрос, не имеющий отношения к формированию скрипта загрузки или изучению данных, то модель должна выдать ответ вида: "Я могу помочь только с формированием скрипта загрузки или изучением данных.
> Пожалуйста, уточните Ваш запрос или составьте новый таким образом, чтобы он имел отношение к представленным данным!"

Шаблон кода для скрипта загрузки (выход модели)

```
Table "Название таблицы"

Create @@@ -- шаблонная секция, подставляются из модели только названия столбцов и их типы данных
CREATE TABLE IF NOT EXISTS "Название таблицы"(
  "Название столбца 1" type, -- тип данных, если допускается null, то Nullable(type) // ЗАДАЧА НА РАБОТУ С ТИПАМИ ДАННЫХ
  "Название столбца 2" type, -- название столбца может быть изменено моделью
  ...
  "Название столбца n" type
) ENGINE = MergeTree ()

ORDER BY
  tuple ()
@@@

Delete @@@ -- ещё одна шаблонная секция
  ALTER TABLE "Название таблицы" DELETE WHERE 1=1
@@@
```

```
Source "Название подключения"

Read @@@ -- основная секция для обработки моделью
SELECT
  "Название столбца 1", -- перенос столбцов из из источника
  "Название столбца 2",
  ...
  "Название столбца k",
  Выражение для изменения столбца 1, -- ЗАДАЧА НА ИЗМЕНЕНИЕ СТОЛБЦОВ
  Выражение для изменения столбца 2,
  ...
  Выражение для изменения столбца m,
  Выражение для нового столбца 1 as "Название нового столбца 1", -- ЗАДАЧА НА СОЗДАНИЕ НОВЫХ СТОЛБЦОВ
  Выражение для нового столбца 2 as "Название нового столбца 2",
  ...
  Выражение для нового столбца n as "Название нового столбца n"
FROM
  "Название таблицы"
WHERE
  toType("Название столбца k") Условие 1 -- для каждого столбца, используемого в WHERE, у которого изменился тип данных, необходимо указывать тип из Create (например, для типа Int32 будет toInt32 и т.д.) // ЗАДАЧА НА ОГРАНИЧЕНИЕ
  ...
ORDER BY
  "Название столбца k" тип сортировки -- ЗАДАЧА НА СОРТИРОВКУ
  ...
LIMIT a
OFFSET b -- ЗАДАЧА НА ОГРАНИЧЕНИЕ

Optimize @@@ -- шаблонная секция на удаление дубликатов
OPTIMIZE TABLE "Название таблицы" DEDUPLICATE
@@@
```

Пример выхода модели для скрипта загрузки

```
Table "Costs"

Create @@@
```

```
CREATE TABLE IF NOT EXISTS "Costs" (
  "Date" Date,
  "Sum" Int32,
  "Product_name" Nullable (String),
  "Category" Nullable (String),
  "file_name" Nullable (String),
  "new_object" Int32
) ENGINE = MergeTree ()

ORDER BY
  tuple ()
@@@

Delete @@@
ALTER TABLE "Costs" DELETE WHERE 1=1
@@@

Source "01"

Read @@@
SELECT
  "Date",
  "Sum",
  "Product_name",
  "Category",
  "file_name",
  toInt32("Sum")+666
FROM
  "Costs"
WHERE
  toInt32("Sum")>5000
ORDER BY
  "Product_name" desc
LIMIT 10
OFFSET 5
@@@

Optimize @@@
OPTIMIZE TABLE "Costs" DEDUPLICATE
@@@
```

Выход модели в секции подключения:

- Ответ языковой модели в чате (либо "Выполнено/Не выполнено", если поступил запрос на обработку данных).
- Код для изменения выборки из 10 строк для предпросмотра (СОСТЫКОВАТЬСЯ С БЭКОМ!)

"Память" модели

Для каждого подключения нейросеть хранит собственную "Память", включающую в себя по меньшей мере историю переписки с пользователем и код для изменения выборки из 10 строк для предпросмотра. Кроме того, существует один из трёх вариантов (обсуждаемых с заказчиком), в зависимости от которого в памяти модели может храниться:

☐ Ничего – модель обращается напрямую к базе ClickHouse. Этот вариант возможен в том случае, если в секции подключения останется только редактор скрипта, а инструмент Data Discovery будет выделен в отдельную задачу.

☐ Кэшированную таблицу объёмом до 1 млн строк

☐ Всю таблицу целиком

Задание на разработку для фронт- и бэкенда

Расположение элементов (см. **Концепт**):

- Список доступных таблиц (вместе с поисковой строкой), исходная таблица и кнопки "Вставить в скрипт" и "Отмена" остаются на своих местах за тем исключением, что ширина исходной таблицы ограничивается в 70% от свободного места в окне для таблицы. При достижении этой границы снизу должен появляться ползунок для прокрутки.
- Подпись "Кол-во строк" перемещается к названию таблицы в левую часть окна.
- Вся правая часть окна до его границы будет занята чатом с нейросетью. Расстояние между чатом и таблицей должно составлять хотя бы 5% от ширины окна.
- Под исходной таблицей будет располагаться редактируемая (созданная нейросетью) таблица, равная по высоте исходной (ограничение на отображение первых 10 строк).
- Кнопки "Очистить чат" и "Активировать нейросеть" располагаются рядом с кнопками "Вставить в скрипт" и "Отмена".

Передача данных:

☐

Вариант 1 – передача всех данных

При нажатии на кнопку "Активировать модель" запускается **асинхронная** передача данных из источника в модель ИИ: сначала отправляется набор метаданных (название таблицы и столбцов, типы данных, количество строк), потом – сами значения из таблицы. Поскольку размер таблицы может быть достаточно велик, подобная операция может занимать длительное время.

Сценарий взаимодействия с чатом во время загрузки данных:

- Началась загрузка. Если спустя 2 секунды загрузка не завершается, то в чате появляется сообщение: "Инициализирована загрузка данных в модель. Вы можете писать запросы в чат до окончания загрузки, но обработка содержимого столбцов таблицы будет недоступна".
- Пользователь деактивировал модель ДО окончания загрузки (
- Выход из окна выбора данных нажатием на кнопку "Отмена" или крестик считается деактивацией модели).

  ☐ Вариант 1. Для данного подключения модель сохраняет скачанные данные и брейкпоинт в последней скачанной строке. После повторной активации модели скачивание данных возобновляется с точки остановки, а в чате моментально выдаётся сообщение: "Возобновлена загрузка данных в модель. Вы можете писать запросы в чат до окончания загрузки, но обработка содержимого столбцов таблицы будет недоступна".

  ☐ Вариант 2. Выдаётся окно для подтверждения деактивации модели с предупреждением: "В случае деактивации в следующий раз процесс загрузки придётся начинать сначала!". При подтверждении модель деактивируется, передача данных прекращается, скачанные данные удаляются.
- Пользователь написал запрос на работу с данными (не метаданными) ДО окончания загрузки. Модель выдаёт ответ: "Для обработки Вашего запроса необходимо скачать данные из таблицы. Пожалуйста, дождитесь окончания загрузки!"
- Пользователь деактивировал модель ПОСЛЕ окончания загрузки.

  ☐ Вариант 1. Для данного подключения модель сохраняет скачанные данные в кэшированной таблице. Плюс: После повторной активации не выдаётся никаких сообщений, пользователь может писать запросы. Минус: необходимо хранить скачанные данные.

  ☐ Вариант 2. Выдаётся окно для подтверждения деактивации модели с предупреждением: "В случае деактивации в следующий раз процесс загрузки придётся начинать сначала!". При подтверждении модель деактивируется, скачанные данные удаляются. Плюс: не нужно дополнительно хранить огромные массивы данных. Минус: затраты времени на повторное скачивание данных.

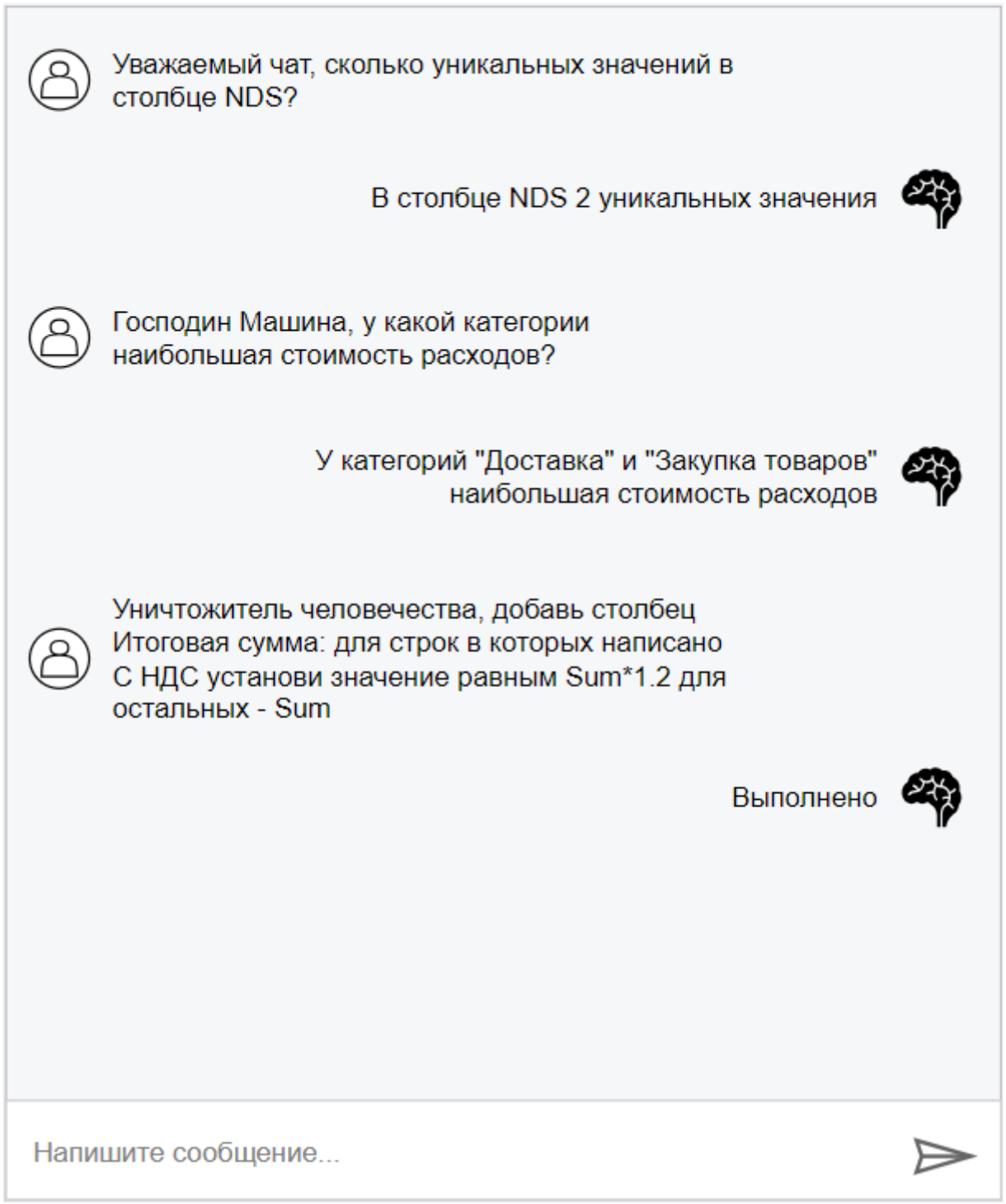
Вариант 2 – Передача выборочной совокупности

При нажатии на кнопку "Активировать модель" запускается сэмплирование (или ограничение с помощью Limit - НЕ РЕКОМЕНДУЕТСЯ!) исходной таблицы до 1 000 000 строк. Данная операция не занимает длительного времени, переданная таблица кэшируется.

> Данный вариант покрывает 95% задач, решаемых новой моделью. Невозможно будет обрабатывать некоторые поисковые запросы, например, вывод точного числа уникальных значений, или поиск одного конкретного значения, не попавшего в сэмплированную выборку. При возможности пренебречь данными задачами вариант 2 является наиболее предпочтительным.

Чат с моделью

Выглядит как переписка. Работает как переписка. В общем, это переписка. С нейросетью.



Особенности:

- В данном чате нет необходимости удалять или изменять сообщения, поскольку каждый запрос чат обрабатывает отдельно и может выдавать различные ответы (например, при анализе использования). Сами сообщения в чате можно выделять и копировать содержимое, а потом вставлять в запрос.
- На каждый запрос модель выдаёт только один ответ. Давать ссылку на ответ нейросети кнопкой "переслать" нельзя – модель запоминает собственные ответы и может отвечать на контекстные вопросы к предыдущим обсуждениям в рамках одного подключения.

- В текстовой строке ввода запроса нельзя прикреплять фото, видео, документы и т.п.
- Пока не нажата кнопка "Активировать нейросеть" строка для ввода сообщения недоступна пользователю (однако история переписки сохраняется).

Кнопки и обработки

Параллельные обработки

> При активированной нейросети водить изменения в исходную таблицу вручную ЗАПРЕЩАЕТСЯ – все поля становятся некликабельными до окончания работы с моделью.

> Вносить вручную изменения пользователь может только в текущую редактируемую таблицу, которая находится под исходной и появляется в момент активации нейросети.

Кнопка "Активировать нейросеть"

При нажатии на кнопку разблокируется чат и появляется редактируемая таблица, а сама кнопка меняет название на "Деактивировать нейросеть". При деактивации нейросети появляется диалоговое сообщение вида: "При деактивации результат обработки таблицы нейросетью не будет добавлен в скрипт загрузки. Если Вы хотите использовать обработанную таблицу, нажмите на кнопку "Вставить в скрипт" при активной модели" с вариантами ответа "Деактивировать" и "Вернуться".

После деактивации блокируется чат и исчезает редактируемая таблица. История переписки сохраняется.

Кнопка "Вставить в скрипт"

При активной нейросети нажатие на эту кнопку передаёт в скрипт загрузки SQL-код, сформированный моделью нейросети.

Кнопка "Очистить чат"

При нажатии на кнопку появляется диалоговое сообщение вида: "Подтвердите, что Вы хотите очистить чат с моделью" с вариантами ответа "Да/Нет". При подтверждении удаляются все сообщения из чата с моделью и отправляется сигнал об очистке памяти модели для текущего подключения.

Данная кнопка доступна как при активированной, так и при деактивированной модели.

Редактируемая таблица

Под исходной таблицей при активированной модели располагается редактируемая таблица. Взаимодействие с ней должно быть полностью аналогично исходной таблице: можно отмечать используемые колонки вручную, менять тип данных (в будущем). Все вносимые вручную изменения передаются в формируемый нейросетью SQL-код для создания подключения. В редактируемую таблицу могут добавляться новые столбцы в зависимости от работы модели. Для редактируемой таблицы также подписывается кол-во строк.