ChatGPT

Постановка задачи и план решения: Выявление мошеннических переводов в мобильном банкинге

1. Постановка задачи и бизнес-контекст

Цель проекта: разработать модель машинного обучения для выявления мошеннических переводов в системе мобильного интернет-банкинга ForteBank. Модель должна автоматически классифицировать входящие транзакции как мошеннические либо легитимные (чистые), позволяя банку своевременно блокировать подозрительные операции. Ключевой бизнес-эффект – снижение финансовых потерь от мошенничества и повышение безопасности клиентов. Успешное решение позволит защитить средства клиентов, повысить доверие к мобильному банку и соответствовать регуляторным требованиям по борьбе с мошенничеством.

Бизнес-эффект и приоритеты: В банковском секторе финансовое мошенничество приносит огромные убытки, и эффективное их предотвращение экономит миллионы долларов. Например, по оценкам Nilson Report мировые потери от карточного мошенничества могут достичь $403 млрд за 10 лет [1]. Решение этой задачи напрямую уменьшит долю невыявленных мошеннических переводов, тем самым экономя средства банка и клиентов. Кроме того, уменьшится число ложных срабатываний (false positives), что улучшит клиентский опыт – честных пользователей не будут беспокоить лишними проверками. Согласно современным подходам, комбинирование продвинутых моделей (например, графовых нейросетей GNN с бустинговыми деревьями) помогает одновременно повысить точность и снизить число ложных тревог [2]. В итоге, эффективная модель анти-фрода увеличит доверие клиентов и обеспечит соответствие требованиям регуляторов по обеспечению прозрачности и безопасности финансовых операций.

Метрики успеха: Основной критерий качества – способность модели **точно и своевременно обнаруживать мошенничества при минимуме ложных тревог**. Для оценки предлагаются следующие метрики:

- **ROC-AUC** – площадь под ROC-кривой, учитывающая соотношение чувствительности (True Positive Rate) и специфичности (False Positive Rate) по всем порогам. AUC является устойчивой метрикой для несбалансированных данных и позволит сравнивать модели по общей дискриминационной способности.
- **Precision, Recall, F1-score** – особенно по классу “мошенничество”. **Recall (Полнота)** на мошенническом классе критически важен (пропущенные мошенничества несут прямые убытки). **Precision (Точность)** важна для контроля ложных срабатываний (чтобы не блокировать легитимные переводы без причины). **F1** как гармоническое среднее даст общее ощущение баланса между ними.
- **PR-AUC (Precision-Recall AUC)** – дополнительная метрика, фокусирующаяся на качестве обнаружения редкого класса. При сильно несбалансированных данных (мошенничества ~1% всех транзакций) PR-кривая более информативна.
- **Confusion Matrix (матрица ошибок)** – для интерпретации результатов: сколько мошеннических транзакций правильно поймано (TP), сколько пропущено (FN), сколько

ложных тревог (FP). На презентации матрица ошибок на тестовых данных четко покажет распределение результатов.

- **Бизнес-метрика:** доля предотвращенных потерь или сумма “спасенных” средств. Можно оценить, какую сумму денег защитила модель, блокируя мошеннические переводы (например, суммировать суммы по правильно выявленным мошенническим транзакциям). Также можно вести учет **False Positive Rate** – доли ошибочно заблокированных легитимных транзакций (желательно держать ниже оговоренного порога, например <0.5%).

Ограничения и допущения: Проект имеет ряд особенностей и ограничений: - **Несбалансированные данные:** Мошеннические транзакции составляют малую долю (в предоставленной выборке ~1–2% или 165 из ~13k транзакций). Модель должна быть устойчива к дисбалансу классов – возможно, потребуется техника балансировки (oversampling мошеннических, undersampling легитимных, генерация синтетических примеров) или использование специальных потерь/весов классов. - **Временная компонента:** Данные транзакций имеют временные метки (2024-2025 гг.). Важно предотвратить утечки данных во времени (**data leakage**): при формировании обучающей и тестовой выборок нельзя использовать информацию из будущего для предсказания прошлого. Желательно разделять данные по времени (например, обучить на более ранних месяцах, валидировать на более поздних) для имитации реального развёртывания модели на будущих данных. - **Скорость и латентность:** В боевом применении модель может применяться в режиме реального времени во время совершения перевода, поэтому алгоритм должен работать достаточно быстро. Однако на этапе хакатона упор делается на качество модели; оптимизация по скорости и внедрение в продакшен – на будущее. Тем не менее, предложенное решение должно учитывать возможность реалтайм-скоринга (например, выбирать алгоритмы, способные быстро инференсить на продакшн-инфраструктуре). - **Отсутствие некоторых данных:** Модель будет обучаться только на предоставленных данных. Мы предполагаем, что существенных внешних данных (например, исторических данных о мошенниках вне выборки) у нас нет. Поэтому упор делается на признаки, извлекаемые из имеющихся транзакций и логов. Кроме того, возможно, не все транзакции имеют соответствующие поведенческие метрики (у нас 13k транзакций и ~8.6k записей с паттернами, некоторые транзакции могли не иметь предшествующей активности для расчета паттернов). - **Интерпретируемость:** Для банковского решения важна интерпретация – нужны объяснимые модели (Explainable AI), чтобы жюри и потенциально регуляторы могли понять логику выявления мошенничества. Поэтому выбор алгоритмов и подходов должен учитывать возможность объяснения (например, использование моделей, интерпретируемых через SHAP/LIME, а также генерация бизнес-правил из модели). - **Конфиденциальность:** Данные клиента обезличены (есть анонимные идентификаторы), но команда должна соблюдать лучшие практики безопасности данных. В рамках хакатона это подразумевает работу только с предоставленными файлами, без утечки личных данных.

Данные и доступные признаки: Для решения задачи даны два набора данных, содержащие как информацию о транзакциях, так и агрегированные поведенческие характеристики клиентов: - **Датасет транзакций** (`транзакции_в_Мобильном_интернет_Банкинге.csv`): включает каждую транзакцию (перевод) с полями: - *Уникальный идентификатор клиента (*`cst_dim_id`*)* – ID отправителя перевода (обезличенный). - *Дата совершенной транзакции (*`transdate`*)* – дата совершения перевода (без времени, округлено до дня). - *Дата и время транзакции (*`transdatetime`*)* – точная метка времени перевода. - *Сумма перевода (*`amount`*)* – сумма операции в тенге. - *Уникальный ID транзакции (*`docno`*)* – внутренний идентификатор документа/перевода. - *Идентификатор получателя (*`direction`*)* – зашифрованный ID получателя средств (возможно хэш счета или телефона получателя). - *Целевой признак мошенничества (*`target`*)* –

метка мошеннической операции (1 – мошенничество, 0 – чистая). Это наша целевая переменная для обучения.

Всего транзакций ~13,114 за период с **30.11.2024** по **29.08.2025**. Мошеннических помечено всего ~165 случаев (≈1.3%), что подтверждает сильный дисбаланс. Данные отсортированы по времени. Наблюдается, что каждый клиент может совершать несколько переводов, и некоторые получатели повторяются между отправителями – это потенциал для выявления **fraud-rings** (групповых схем).

- **Датасет поведенческих паттернов** (`поведенческие_паттерны_клиентов_3.csv`): содержит предрассчитанные агрегированные признаки активности клиента в мобильном приложении, привязанные к дате транзакции. У каждого записа есть:
- *Уникальный идентификатор клиента (`cst_dim_id`) и Дата транзакции (`transdate`)* – эти поля можно использовать для соединения с основным датасетом транзакций.
- **Признаки устройства и окружения:** например, `monthly_os_changes` (число разных версий ОС, использованных клиентом за последние 30 дней до даты транзакции), `monthly_phone_model_changes` (число разных моделей телефона за 30 дней – отражает как часто клиент менял устройство). Также сохраняются категориальные признаки последней сессии перед транзакцией: `last_phone_model` (модель телефона) и `last_os` (версия ОС).
- **Признаки частоты логинов:** `logins_last_7_days` и `logins_last_30_days` – количество уникальных сессий за последние 7 и 30 дней соответственно; производные от них – среднее число логинов в день за 7 дней (`login_frequency_7d`) и за 30 дней (`login_frequency_30d`), а также отношение и процент изменений: например, `freq_change_7d_vs_mean` – относительное изменение недавней активности к средней (показывает, стал ли клиент заходить чаще или реже чем обычно), и доля логинов 7 дней от логинов 30 дней.
- **Признаки регулярности активности:** метрики интервалов между сессиями: `avg_login_interval_30d` (средний интервал между входами в секундах за 30 дней), `std_login_interval_30d` и `var_login_interval_30d` (стандартное отклонение и дисперсия этих интервалов), `ewm_login_interval_7d` (экспоненциально взвешенное среднее интервала за последние 7 дней, с большим весом недавним сессиям). Также нестандартные метрики: `burstiness_login_interval` – показатель “взрывности” паттерна логинов ((std − mean)/(std + mean)), и `fano_factor_login_interval` – коэффициент Фано (variance/mean) для интервалов. Наконец, `zscore_avg_login_interval_7d` – Z-скор среднего интервала за 7 дней относительно среднего за 30 дней (на сколько стандартных отклонений недавний средний интервал отличается от привычного).

Всего ~8,588 записей с такими паттернами. Каждый такой запись соответствует **одной транзакции** из датасета транзакций (судя по полям `cst_dim_id` и `transdate`). Не для каждой транзакции есть запись – возможно, у части переводов не было предшествующих сессий (например, если клиент не логинился перед переводом, или данные о логинах неполные). Нужно будет соединить два датасета по ключам (`cst_dim_id`, `transdate`) и, вероятно, по времени транзакции (например, если в один день несколько переводов, паттерн может быть общим, либо учитывается последняя сессия перед каждым переводом).

Ценные признаки (инсайты из данных): Предоставленные признаки уже отражают важные поведенческие паттерны, которые, как предполагается, помогут отличать мошенника от

обычного клиента. На основании предметной области и данных можно выделить несколько групп потенциально **важных признаков**:

- **Подозрительное устройство и окружение:** частая смена устройства или версии ОС (`monthly_phone_model_changes`, `monthly_os_changes`) может быть индикатором мошенника, который входит в аккаунт жертвы с разных устройств или эмулирует их (в норме клиент обычно пользуется 1-2 устройствами). Необычная модель телефона или ОС (например, очень старая версия Android, либо эмулированная среда) тоже могут косвенно указывать на мошенничество. **Инсайт:** Мошенники часто используют эмуляторы или виртуальные среды, поэтому метрика разных устройств и версий ОС за короткий период будет выше нормы.
- **Аномалии в частоте логинов:** если аккаунт жертвы обычно редко входит (скажем, 1 раз в день), а за последние 7 дней наблюдается всплеск логинов (рост `logins_last_7_days` и высокий `freq_change_7d_vs_mean`), это может указывать, что злоумышленник интенсивно проверяет и управляет аккаунтом перед переводом. Или наоборот, возможно длительная неактивность, а потом внезапный вход и перевод – это тоже аномалия. Признаки вроде `freq_change_7d_vs_mean` и `zscore_avg_login_interval_7d` прямо указывают на отличие недавнего поведения от привычного – такие "скачки" часто характерны для компрометации аккаунта [3]. Как отмечалось в решениях по кредитному мошенничеству, **особенно информативны признаки, сравнивающие недавнюю активность с исторической нормой** [3].
- **Нерегулярность, "рывки" в поведении:** высокая **вариативность интервалов** между сессиями (`std_login_interval_30d`, `burstiness_login_interval`) может означать нестабильное поведение. Например, мошенник может совершить серию быстрых логинов/операций (низкие интервалы, "burst") после долгого перерыва. Коэффициент **burstiness** и **Fano-factor** будут высокими, если поведение нерегулярно – это может быть индикатором взлома (у обычного пользователя может быть более ритмичный график входов).
- **Характеристики транзакции:**
- **Сумма перевода:** мошенники могут выводить крупные суммы средств (в пределах лимитов) – сумма значительно выше средней для данного клиента может быть красным флагом. Стоит рассчитать отклонение суммы текущего перевода от среднего чека клиента (если есть история переводов клиента) или от средних сумм по всем клиентам.
- **Время совершения:** необычное время перевода (например, глубокая ночь или выходной) для конкретного клиента может сигнализировать о мошенничестве. Нужно проверить распределение: если клиент обычно переводил деньги днем в будни, а подозрительная транзакция произошла ночью в выходной, это подозрительно.
- **Получатель (destination)**: хотя идентификатор получателя зашифрован, можно генерировать признаки по нему: например, **количество разных отправителей, переводивших этому получателю**. Часто мошенники используют счет-"мул" для вывода средств от многих жертв. Если наш клиент отправляет деньги на счет, который также получил переводы от многих других клиентов (возможно, тоже жертв), то такой получатель сильно подозрителен. Признак "кол-во уникальных отправителей на данного получателя" или "общая сумма, полученная на этот dest за последние N период" может помочь. Также можно отметить, делал ли **конкретно этот клиент перевод этому получателю раньше** (если нет – первый перевод на новый счет, и он же крупный – это подозрительно).
- **Частота переводов клиента:** если клиент вдруг сделал несколько переводов подряд (особенно на разные счета) за короткое время – это может быть "разделение суммы" мошенником, чтобы не привлекать внимание. Признак: число переводов клиента за последние 24 часа или 7 дней.

- **Комбинации признаков (feature interactions):** Могут оказаться ценными перекрестные признаки: например, *“смена устройства + высокая сумма”* (если одновременно перевод крупный и сделан с нового устройства, риск выше). Или *“время суток + burstiness логинов”* (ночной перевод после серии быстрых логинов). На Kaggle в топовых решениях часто создают пары/тройки признаков для улавливания взаимодействий [4]. Мы тоже можем рассмотреть генерацию взаимодействий: напр., категориальные связки (модель устройства × признак того, мошенничество ли; или кластеризация устройств по риску).

Важно учесть мета-информацию задания: По условиям хакатона, вероятно, оговорены критерии оценки, помимо метрик качества. Обычно жюри оценивает: - **Иновативность решения:** использование нестандартных подходов (например, графовый анализ связей между счетами, применение LLM для объяснений) может добавить баллов. - **Завершенность решения:** наличие всего pipeline от данных до деплоя, и готовность к демонстрации. - **Презентация результатов:** понятная структура, визуализации, объяснение бизнес-ценности. - **Следование методологии хакатона:** зачастую рекомендуется CRISP-DM или иной структурированный подход. Мы разбили решение на этапы: EDA, фичи, моделирование, оценка, и т.д., что соответствует лучшим практикам. - **Командная работа и использование AI-агентов:** отдельный упор задачи – предложить команду AI-агентов (специализированных помощников), что отражает современный тренд AutoML и AI помощников. Это демонстрирует умение автоматизировать и ускорять части ML-процесса с помощью ИИ.

В целом, задача сформулирована как **классификация с сильным дисбалансом**, требующая богатого **feature engineering**, аккуратной валидации по времени и **интерпретируемости**. Далее опишем, как распределить работу между различными AI-агентами и построить эффективный конвейер решения.

2. Роли и задачи AI-агентов для решения задачи

Для решения задачи предлагается команда специализированных AI-агентов. Каждый агент – это условный модуль (возможно реализованный на основе Large Language Model или AutoML инструмента), отвечающий за свою часть pipeline. Разделение ролей позволяет параллельно и эффективно справиться со сложной задачей. Ниже перечислены предполагаемые роли агентов, их зоны ответственности и примеры промптов для их запуска:

Feature Engineer Agent

Задачи и ответственность: Этот агент отвечает за анализ сырых данных и генерацию новых признаков (feature engineering). Он исследует распределения, взаимосвязи между переменными, выявляет какие свойства транзакций и поведения пользователей могут быть предиктивны для мошенничества. Feature Engineer агент: - Проводит первичную разведывательную анализ данных (EDA) для нахождения аномалий и важных корреляций. Например, проверяет, отличаются ли показатели `monthly_os_changes` или `burstiness_login_interval` у мошеннических транзакций против обычных. - Предлагает новые признаки: агрегаты (суммы, средние, счетчики) по клиенту или получателю, взаимодействия признаков (произведения, соотношения), временные характеристики (день недели, час дня), флаги аномалий (например, флаг нового получателя для клиента). - Обеспечивает трансформацию признаков: нормализация сумм, лог-преобразование, encoding категориальных (модель телефона, ОС) — возможно через target encoding или embedding. - Следит, чтобы не было утечки целевой переменной: каждый новый признак должен рассчитываться только на основе данных доступных до момента транзакции. - Готовит итоговый датасет признаков для моделирования.

Пример промпта для запуска агента:

> **Feature Engineer Agent Prompt:** *“Ты – эксперт по feature engineering для задач обнаружения финансового мошенничества. У тебя есть две таблицы: (1) транзакции с полями (дата/время, сумма, получатель, клиент, метка мошенничества) и (2) агрегированные паттерны поведения клиентов (частота логинов, смена устройств и т.д. до момента транзакции). Проанализируй доступные признаки и придумай, как их преобразовать или дополнить, чтобы улучшить выявление мошеннических транзакций. Учти дисбаланс классов и временные зависимости. Перечисли новые потенциальные признаки (с обоснованием), которые стоит добавить к датасету.”*

Fraud Pattern Detector Agent

Задачи и ответственность: Агент по обнаружению мошеннических паттернов фокусируется на поиске **нетривиальных, скрытых закономерностей**, характерных для мошенничества. Он может применять как аналитические, так и ML/статистические методы помимо основного моделирования: - Выполняет поиск аномалий и кластерный анализ. Например, может применить алгоритмы кластеризации или автоэнкодеры на поведенческие метрики, чтобы выявить группы пользователей с необычным поведением (которые могут соответствовать мошенникам). - Строит **граф связей** между клиентами и получателями: узлы – клиенты и счета-получатели, ребро – транзакция. Затем анализирует этот граф (метрики центральности, сообщества). Может обнаружить, что несколько разных клиентов переводили деньги одному получателю (возможная “муловая” сеть), или один клиент перевел многим новым получателям за короткое время. Такие паттерны сложно уловить обычной моделью, но графовый анализ их выявляет. При наличии ресурсов может даже применить **Graph Neural Network (GNN)** для классификации узлов транзакций, получив эмбеддинги связей [5] [6]. - Использует правила и шаблоны мошенничества: например, ищет последовательности действий (вход в аккаунт – сразу перевод на большой сумму – сразу вывод денег) или сопоставляет с известными сценариями социальной инженерии. - Передает найденные шаблоны и дополнительные признаки, характеризующие эти шаблоны, обратно Feature Engineer или напрямую в модель. Например, может добавить признак “подозрительный получатель” (1, если получатель имеет $\geq$N разных отправителей).

Пример промпта для запуска агента:

> **Fraud Pattern Detector Agent Prompt:** *“Ты – аналитический агент по поиску мошеннических схем. Имеется граф транзакций: узлы ‘клиент’ отправляет сумму узлу ‘получатель’. Также есть временные последовательности логинов клиентов. Найди скрытые паттерны, характерные для мошенничества. Например, выяви группы связанных аккаунтов, аномальные последовательности действий или общие черты мошеннических случаев. Опиши эти паттерны и предложи способ, как использовать их в модели (новые признаки или правила). Особое внимание – повторяющиеся получатели, всплески активности аккаунта, смена устройства перед переводом.”*

AutoML Agent

Задачи и ответственность: AutoML-агент отвечает за автоматизацию выбора моделей, их гиперпараметров и построение оптимального ансамбля: - Пробует различные алгоритмы машинного обучения на подготовленных признаках: градиентный бустинг деревьев (LightGBM, XGBoost, CatBoost), логистическая регрессия, Random Forest, а также, при возможности, нейросетевые подходы (например, простой многослойный персептрон, либо более сложную архитектуру, если обнаружены последовательностные паттерны). - Проводит

гиперпараметрический поиск (Grid Search, Random Search или Bayesian optimization) по ключевым параметрам моделей (глубина деревьев, learning rate, пороги регуляризации и т.д.) с использованием кросс-валидации. Учитывает временную структуру – например, использует валидацию по времени (time-based split) либо stratified K-fold, где стратифицирует по мошенничеству. - Может автоматически попробовать инженерные приемы: отбор признаков (feature selection) на основе важности, генерацию полиномиальных признаков или бининг переменных. - Рекомендует лучшую модель или ансамбль моделей. Например, может обнаружить, что **ансамбль бустинговых моделей дает наилучший результат**, что соответствует практикам победителей Kaggle (часто объединяют XGBoost + LightGBM + CatBoost) [4]. Также проверяет, не улучшит ли качество стеккинг или блендинг нескольких подходов. - Следит, чтобы модель не переобучилась: сравнивает качество на кросс-валидации и холдаут-наборе.

Пример промпта для запуска агента:

> **AutoML Agent Prompt:** *"Ты – AutoML-агент. У тебя есть обучающий набор данных с меткой мошенничества и ~N признаков, подготовленных предыдущими агентами. Твоя задача – подобрать наилучшую модель/ансамбль для классификации. Попробуй следующие алгоритмы: логистическая регрессия, RandomForest, XGBoost, LightGBM, CatBoost, нейросеть. Оптимизируй гиперпараметры (укажи диапазоны) под метрику AUC-ROC (основная) и F1 (вторичная). Учитывай дисбаланс – например, используй параметр scale_pos_weight или класс-веса. Найди лучшую модель, сравнив результаты кросс-валидации, и выдай ее параметры. Также оцени, даст ли ансамблирование прирост (например, усреднение или стеккинг топ-3 моделей). Верни описание лучшей модели и ее метрик на валидации."*

Data Validator Agent

Задачи и ответственность: Этот агент обеспечивает качество данных на всех этапах: - Проверяет исходные данные на пропущенные значения, дубликаты, аномальные выбросы. Например, убеждается, что во всех транзакциях корректно указаны суммы (неотрицательные, не NULL) и метки (только 0/1). Если есть пропуски в поведенческих признаках, решает, как их заполнить (imputation) – например, если у клиента не было логинов за 30 дней, ставит 0 или особое значение. - Следит за корректностью объединения датасетов: нет ли транзакций без соответствующих паттернов и как с ними поступить (например, если для ~4500 транзакций нет поведенческих данных, то агент может пометить их специальным флагом или заполнить нулями, чтобы модель понимала отсутствие активности). - Выявляет возможные утечки: например, случай, если признак формируется с использованием целевой переменной (в данной задаче маловероятно, но, например, убедиться, что мы не используем будущее знание – транзакции будущих дней – при расчете текущих признаков). - Оценивает распределения признаков: сравнивает тренинговый и тестовый (или валидационный) сеты – нет ли сильного дрейфа (distribution shift). Если дрейф есть, сообщает о необходимости коррекции (например, пересчитать нормировки или учесть сезонность). - Валидирует модельные прогнозы: смотрит на калибровку вероятностей, строит контрольные графики (калибровочные кривые, learning curve). Проверяет, что модель адекватно экстраполирует (нет ли ситуаций, когда подает невозможные значения и получает неоправданные прогнозы). - Обеспечивает воспроизводимость: фиксирует случайные сиды, проверяет, что pipeline работает стабильно на разных запусках.

Пример промпта для запуска агента:

> **Data Validator Agent Prompt:** *"Ты – агент контроля качества данных. Проведи аудит данных и подготовленного feature-set перед обучением модели. 1) Проверь, есть ли пропущенные или некорректные значения в признаках (например, null в логинах или отрицательные суммы транзакций). 2) Предложи способы заполнения или обработки таких аномалий. 3) Убедись, что объединение таблиц прошло верно – насколько уменьшилась выборка, нет ли систематического отбора. 4) Проверь распределение ключевых признаков в обучающей vs тестовой выборке (при временном разделении) – нет ли значимого сдвига. 5) Убедись, что никакой признак не "подсказывает" напрямую целевое значение (например, не сформирован с использованием target). Отчитай о любых проблемах и способах их решения."*

Evaluator Agent

Задачи и ответственность: Агент-оцениватель занимается всесторонней оценкой обученной модели: - Считает основные метрики на контрольном наборе: ROC-AUC, PR-AUC, Accuracy, Precision/Recall/F1, а также бизнес-метрики (например, сумму предотвращенных мошенничеств при выбранном пороге). - Генерирует понятные визуализации результатов: **ROC-кривую**, график Precision-Recall, **Confusion Matrix**. Он же может подобрать оптимальный порог классификации, ориентируясь на максимум F1 или фиксируя нужный Recall. Например, рисует кривую зависимости TPR/FPR от порога или использует метод оптимизации порога по максимуму метрики. - Формирует **отчет по ошибкам**: какие типичные случаи FP (ложные срабатывания) и FN (пропущенные мошенничества) – анализирует их признаки. Это важно для последующего улучшения модели и для объяснения жюри, что команда понимает слабые места. Например, может выясниться, что большинство FN – это транзакции с очень малыми суммами (модель их не распознала, считая безопасными), или что FP случаются у необычных но честных клиентов (например, клиент часто меняет телефон, но не мошенник). - Проводит **валидацию на устойчивость**: если есть разбиение по времени (например, обучили на первых 6 месяцах, валидируем на последних 2 месяцах), проверяет, что качество не проседает во времени. - Отдельно оценивает **Explainability**: готовит данные для интерпретации – считает важности признаков (feature importance) для моделей (например, средняя gain в деревьях или permutation importance). Строит **SHAP values** для выборки и готовит визуализации: summary plot (распределение влияния признаков), bar-chart топ-10 признаков по важности. Эти результаты он передает Explainer-агенту для генерации текстовых объяснений. - Проверяет модель на **перекрестную валидацию** (если применимо) – чтобы убедиться, что результат не случаен. Например, 5-fold CV AUC std.

Пример промпта для запуска агента:

> **Evaluator Agent Prompt:** *"Ты – агент оценки модели. Модель обучена – теперь оцени ее на тестовых данных (временно удержанных). 1) Вычисли метрики: AUC-ROC, AUC-PR, Accuracy, Precision, Recall, F1. 2) Построй confusion matrix – в числах и процентах. 3) Проанализируй: насколько модель хороша в выявлении мошенничества (Recall), каков уровень ложных тревог (Precision). 4) Предложи оптимальный порог для классификации, обоснуй (например, исходя из максимума F1 либо заданного бизнесом соотношения). 5) Проведи error analysis: опиши общие черты пропущенных мошенничеств и ложных срабатываний. 6) Подготовь данные для интерпретации: вычисли важность каждого признака (например, SHAP values на тесте) и выдели топ-10 важных признаков. Представь результаты ясно для дальнейшего использования."*

LLM Explainer Agent

Задачи и ответственность: Это агент на основе Large Language Model, задачей которого является **объяснение работы модели и ее предсказаний на человеческом понятном языке**. Он выполняет роль “переводчика” технических результатов для бизнес-аудитории и создает текст для презентации: - Объясняет **глобальную логику модели**: на основе feature importance и выявленных паттернов описывает, какие факторы чаще всего влияют на решение о мошенничестве. Например: “Модель обращает особое внимание на частую смену устройства и резкий рост активности входов – эти факторы сильно повышают вероятность мошенничества”. - Поясняет **локальные предсказания**: может на лету сгенерировать объяснение для конкретной транзакции. Например: “Транзакция №X помечена моделью как мошенничество с вероятностью 92%, потому что сумма перевода очень велика для данного клиента, перевод совершается ночью, и получатель – новый для клиента. Кроме того, за последний день было 10 входов в аккаунт с разных устройств, что нетипично для него – модель распознала это как признаки компрометации аккаунта.” Такой агент берет значения признаков конкретного случая и их SHAP-влияния, формируя понятный вывод. - Подготавливает текстовые **выводы и рекомендации**: например, “Наше решение способно обнаруживать ~98% мошеннических транзакций, снизив средний ущерб на клиента на X тенге в месяц. Мы рекомендуем внедрить его в antifraud-процесс мобильного банка. Также выявлены новые инсайты: аккаунты, использующие более 3 устройств в месяц, в 5 раз чаще подвержены мошенничеству – стоит усилить контроль при смене устройства.” - Участвует в оформлении финальной презентации: генерирует описания графиков (что показывает ROC, почему важен SHAP и т.п.), формулирует ответы на возможные вопросы жюри (например, о надежности модели, о методах борьбы с дисбалансом, о времени детекции).

Пример промпта для запуска агента:

> **LLM Explainer Agent Prompt:** *“Ты – экспертный AI-ассистент, который умеет ясно объяснять результаты работы модели выявления мошенничества. У тебя есть данные: важность признаков (например: 'monthly_phone_model_changes' – 0.15, 'amount' – 0.12, 'freq_change_7d_vs_mean' – 0.10, и т.д.), примеры SHAP-значений для отдельных транзакций, и общие метрики (AUC, F1). Подготовь объяснение для руководства банка: какие ключевые факторы влияют на решение модели? Как модель ведет себя (например, очень чувствительна к определенным комбинациям признаков)? Приведи пример мошеннической транзакции и объясни, почему модель ее так классифицировала. Объяснение должно быть простым, избегай жаргона, используй понятные аналогии. Также подчеркни, что модель – лишь инструмент помощи и финальное решение за службой безопасности.”*

Каждый агент будет работать в связке с другими. Например, Feature Engineer и Fraud Pattern Detector вместе формируют признаковое пространство; AutoML обучает модели и передает лучшую Evaluator’у; Evaluator и Explainer взаимодействуют для интерпретации. Предусмотрена координация, которую может обеспечивать сам человек-капитан команды или управляющий скрипт (orchestrator), последовательно вызывающий агентов и передающий результаты далее. Таким образом, команда агентов покрывает весь цикл разработки модели от подготовки данных до генерации итогового отчета.

3. Pipeline построения модели (end-to-end)

Для успешного построения решения нужен четкий план действий – **pipeline**, состоящий из последовательных этапов. Ниже представлен предлагаемый конвейер работ, который соответствует и классическому процессу Data Science (CRISP-DM), и требованиям хакатона:

1. **Предварительный анализ данных (EDA) и предобработка:**
2. Загрузка предоставленных датасетов, проверка размеров, типов данных, ознакомление с описанием полей.
3. Очистка данных: обработка пропусков и аномалий. Например, если в поведенческих данных есть `NaN` для некоторых клиентов, решить, чем заменить (0, средним, специальной категорией “нет данных”). Проверка, есть ли дубли транзакций или противоречивые метки.
4. Приведение типов: даты – в формат datetime, числовые поля – в float/int. Разделение возможных составных полей (возможно, `transdatetime` можно разложить на дату и время отдельно, а `transdate` – на год/месяц/день).
5. Соединение таблиц: merge транзакций с паттернами по `cst_dim_id` и `transdate`. После объединения – проверка, сколько транзакций удалось сопоставить с паттернами (например, ~8588 из 13114, остальные ~4500 не имеют записей). Для тех, что не сопоставились, можно присвоить им дефолтные значения паттернов (например, 0 активностей, 1 устройство) и флаг “нет поведенческих данных”.
6. Создание обучающей и тестовой выборки. Рекомендуется разделить данные **по времени**: например, обучить модель на транзакциях до 31 июля 2025, а тестировать на август 2025. Так мы симулируем будущее и проверяем, как модель обобщает на новые данные. Альтернативно, если данных мало, можно использовать k-fold кросс-валидацию с группировкой по клиенту (чтобы транзакции одного клиента не утекали в обучение и тест одновременно). Но предпочтительно временное разделение, т.к. природа мошенничества может меняться со временем.

7. **Выход этого шага:** чистый, объединенный датасет с меткой, разбитый на train/val (и, возможно, hold-out test) с базовыми предобработками (обработанные пропуски, правильные типы). Также отчет EDA: базовые частоты, распределение целевой (сколько % fraud), распределение сумм переводов (например, выявлено, что мошеннические переводы имеют медиану X, большую, чем у легитимных), и т.п. Это поможет далее ориентироваться.

8. **Feature Engineering (генерация новых признаков):**

9. Расчет дополнительных признаков на основе транзакций:
    - Временные: час дня перевода, день недели. Возможно, мошенники чаще активны в нерабочие часы – эти бинарные/категориальные фичи могут помочь.
    - Признаки по получателю (destination): для каждого `direction` рассчитить сколько разных клиентов ему перевели деньги до текущего момента (`dest_unique_senders_count`), общую сумму полученную (`dest_total_received`), флаг нового получателя для данного клиента.
    - Признаки по клиенту: количество переводов клиентом за последние 7/30 дней, средняя сумма переводов клиента (и отношение текущей суммы к этой средней – показатель аномальности суммы).
    - Комбинированные признаки: возможно, произведение или отношение некоторых уже имеющихся: например, `amount` / (средняя сумма по клиенту) – насколько эта

транзакция выбивается из привычных по сумме. Или `logins_last_7_days` * флаг ночной транзакции – есть ли сочетание частых логинов и ночной активности.
    - Векторы/эмбеддинги: при наличии времени можно попробовать обучить простую модель для получения эмбеддингов пользователей или получателей. Например, построить матрицу “клиент-<получатель>” переводов и применить SVD/Matrix Factorization либо Node2Vec на графе клиент-получатель, чтобы получить embeddings, которые захватят скрытые связи (сообщества мошенника и мула). Эти эмбеддинги (несколько чисел на узел) затем добавить как признаки.
10. Обработка категориальных признаков: `last_phone_model` и `last_os` – это строковые поля. Их можно преобразовать через **frequency encoding** (сколько раз встречается данная модель в данных), target encoding (с какой частотой данная модель ассоциируется с fraud в трейне), либо более продвинуто – внедрить эмбеддинг (например, как часть нейросетевой модели, если пойдем в глубокое обучение).
11. Масштабирование признаков: для алгоритмов, требовательных к масштабу (логистическая регрессия, нейросети) – нормализовать/стандартизовать количественные признаки (`amount`, интервалы и т.д.). Для деревьев (LightGBM/XGB) это не критично, они сами справятся.
12. Балансировка классов (часть feature engineering): можно на этапе формирования training set сделать oversample мошеннических транзакций (например, дублировать случайным образом 1-ы или сгенерировать их с небольшим шумом). Однако современный подход – предпочесть правильную функцию потерь или параметр модели (scale_pos_weight). На этапе фичей можно разве что синтетически добавить примеры, но лучше оставить это модели.

13. **Выход этого шага:** обогащенный датасет с новым набором X-признаков. Ожидаемое их количество – десятки, до ~50–100. (Например, исходных ~17 поведенческих + 5 базовых транзакционных + 10 новых = ~30–40). Также список этих признаков и их описание, чтобы потом использовать в интерпретации.

14. **Выбор алгоритма и обучение модели:**

15. Начать с **базовой модели**: например, обучить `LightGBM` на подготовленных данных как baseline. LightGBM хорошо работает с табличными данными и дисбалансом (есть параметр `is_unbalance` или ручная установка веса классов). Проверить его качество (AUC, F1 на валидации). Бейзлайн нужен, чтобы понять, в правильном ли направлении фичи работают.
16. Параллельно попробовать несколько моделей:
    - **Gradient Boosting**: LightGBM, XGBoost, CatBoost – сравнить их. CatBoost автоматически обработает категориальные признаки (что удобно для `phone_model`), XGBoost/LightGBM требуют encoding, но могут быть быстрее.
    - **Логистическая регрессия**: с регуляризацией и отбором признаков – как интерпретируемая простая модель. Вероятно покажет более низкий AUC, но даст бенчмарк и проверит, нет ли линейно разделяющего признака (если логистическая даст хороший результат, значит уже есть сильный линейный сигнал).
    - **Нейронная сеть**: например, многослойный персептрон с несколькими скрытыми слоями. На табличных данных бустинг обычно лучше, но NN можно добавить, особенно если использовать эмбеддинги для категорий и соединений. Также если есть последовательные данные (например, попробовать LSTM/Temporal CNN по временным сессиям клиента), но это, вероятно, выходит за рамки времени хакатона.

- **Ансамбли**: можно заранее предусмотреть бустинг + логрег (stacking) или усреднение моделей. В исследованиях по фроду отмечается, что **стеккинг нескольких бустинговых моделей улучшает результат** [7], а также сохраняет интерпретируемость при применении XAI. Поэтому можем планировать финальный ансамбль XGB + LGBM + CatBoost (взять усредненное решение или мета-модель над ними).

17. **Гиперпараметры:** Настроить ключевые параметры. Например, для LGBM: `num_leaves`, `max_depth`, `learning_rate`, `min_child_samples`, `subsample`, `colsample_bytree`. Для XGB аналогично. Можно воспользоваться готовыми настройками, полученными AutoML Agent (если он выдал). В условиях хакатона, возможно, GridSearch на небольшей сетке или Bayesian optimization с 20–30 итерациями по AUC.
18. **Валидация:** После обучения – оценить на валидационном сете. Сохранить модели и метрики. При необходимости – повтор итерации feature engineering (например, если важным оказался признак, можно еще его комбинации добавить, или если видим недостающие паттерны – вернуться к шагу 2).

19. **Выход этого шага:** обученная **лучшие модель(и)** – например, LGBM с определенными параметрами, дающая высокий ROC-AUC. Также, возможно, сохраненные веса/бинарники модели, готовые для применения. Отчет с результатами кросс-валидации или hold-out (например: “LightGBM показал AUC=0.95, Recall=0.90 на тесте, что лучше, чем XGBoost с AUC=0.94…”).

20. **Оценка качества и тестирование:**

21. Использование Evaluator Agent: рассчитываются все целевые метрики на тестовом наборе **(который не использовался при обучении)**. Для наглядности можно составить табличку: | Model | AUC-ROC | Precision | Recall | F1 | |---------------|---------|-----------|--------|-------| | LightGBM | 0.950 | 0.75 | 0.92 | 0.83 | | XGBoost | 0.945 | 0.72 | 0.90 | 0.80 | | Ensemble (avg)| 0.956 | 0.78 | 0.90 | 0.84 |

    (Цифры условные для иллюстрации). - Построение **ROC-кривой** и вычисление AUC. Демонстрация, что кривая модели сильно выше линии случайного угадывания, особенно в зонах низких FPR. Можно отметить точку на ROC, соответствующую выбранному порогу. - **Precision-Recall анализ:** особенно важен, когда класс редкий. Построить PR-кривую, показать достигнутый баланс precision ~ X при recall ~ Y. Возможно, выбрать порог по максимальному F1 или по бизнес-ограничению (например, “не более 5 ложных блокировок на 1000 транзакций” – тогда выбрать порог, где FPR = 0.5%). - **Confusion matrix:** визуально отразить результаты. Например, из 50 реальных мошенничеств модель нашла 46 (True Positives) и пропустила 4 (False Negatives), при этом ошибочно пометила 10 из 1000 легитимных как мошенничество (False Positives). Такое представление (можно абсолютными числами и процентами) очень наглядно для жюри. - **Анализ ошибок:** перечислить типичные FP и FN случаи, как описано ранее. Возможно, даже разобрать 1-2 конкретных примера FN: “Вот транзакция, которую модель пропустила – у клиента низкая активность, но сумма маленькая и ничего особо подозрительного, однако это была мошенническая транзакция (может, новый тип мошенничества, невидимый для нашей модели)”. И FP пример: “Модель ошибочно заподозрила транзакцию: клиент действительно сменил устройство и перевел крупную сумму ночью, но это был легитимный случай (например, клиент сам купил новый телефон). Такие кейсы нужны, чтобы улучшать модель в будущем – возможно, добавить проверку, что устройство было все-таки зарегистрировано клиентом ранее, и т.д.” - Тестирование также предполагает

проверку на стресс: что если изменить порог, или если поступят данные из другого распределения? Но в условиях хакатона достаточно оффлайн-теста на отложенных данных.

22. **Интерпретируемость и объяснения (Explainability):**

23. Рассчитываются **Feature Importance** модели. Для бустинговых деревьев можно вывести важность (Gain/Split importance), отсортировать топ-10 значимых признаков. Ожидаем, что в топе будут, например, `monthly_phone_model_changes`, `dest_unique_senders_count`, `amount`, `freq_change_7d_vs_mean`, `last_os` и т.п. Это подтвердит здравость модели и покажет, какие факторы решающие.
24. Вычисление **SHAP values** для множества примеров (например, 1000 случайных транзакций из теста). С помощью SHAP создается *summary plot*: точечный график, где по оси X – SHAP влияние, по Y – признаки, цветом – значение признака. Он покажет, как каждый признак влияет: например, для `monthly_phone_model_changes` увидим, что высокие значения (красные) сдвигают SHAP сильно в + (towards fraud), что подтверждает: много смен устройства -> выше вероятность мошенничества. Аналогично, `amount`: красные (большие суммы) правее – увеличивают риск.
25. **Partial Dependence / ICE** (опционально): для одного-двух ключевых признаков построить график зависимости прогноза от значения признака, чтобы показать нелинейность. Например, зависимость вероятности мошенничества от суммы: может быть U-образная (очень маленькие суммы тоже могут быть мошенническими, чтобы не заметили, а средние – безопасней, большие – снова риск). PDP-графики или хотя бы рассуждения об этом добавят глубины.
26. LLM Explainer агент на основе этих данных готовит текстовое объяснение для презентации (см. ниже раздел визуализации и презентации). Он переведет сухие графики SHAP и confusion matrix в понятные инсайты: какие признаки наиболее важны и почему, как модель принимает решение в общих чертах.

27. Важно подчеркнуть, что **модель не является черным ящиком**: мы вооружились XAI инструментами (SHAP, LIME и т.д.). Прозрачность решения – один из критериев (в реальном мире регуляторы требуют объяснять автоматические решения [8]).

28. **Автоматизация и дообучение (MLOps pipeline):**

29. Последним шагом планируется, как модель будет внедрена и поддерживаться. В рамках хакатона можно предоставить план: например, обучение модели оформлено в скрипт/ноутбук, который можно запускать ежемесячно на новых данных.
30. Настроить сохранение модели и версии данных с помощью MLFlow или Weights & Biases – это позволит отслеживать метрики по времени, сравнивать старые и новые модели.
31. Предусмотреть **мониторинг в продакшене:** с помощью сервиса **Evidently.ai** или аналогичных построить дашборд, который будет отслеживать статистики новых поступающих транзакций и предупреждать, если распределение сильно отклонилось (drift) или производительность модели упала (например, увеличилось число пропущенных мошенничеств – значит, мошенники изменили тактику, нужно дообучить модель).
32. Авто-допобучение: можно реализовать pipeline, где новые подтвержденные мошенничества (метки от antifraud-отдела) добавляются в тренировочный набор, и модель периодически переобучается (например, раз в месяц) на расширяющемся датасете, чтобы сохранять актуальность. При этом важно хранить контрольную выборку для объективной оценки модели после каждого дообучения.

33. Если используется облако, настроить CI/CD: например, с помощью GitHub Actions или Jenkins, чтобы при обновлении кода модели автоматически запускалось обучение/тест и деплой обновленной модели в облаке.

Таким образом, pipeline замкнется: новые данные → обновление признаков → обучение/валидция → деплой → мониторинг → при необходимости возвращение на шаг фичей или обучения. На хакатоне мы, конечно, проводим этот цикл в оффлайн-режиме один раз, но показываем жюри готовность решения к промышленной эксплуатации.

4. Архитектура решения и используемые технологии

При реализации решения мы учтем возможности современных облачных сервисов (AWS, Azure, GCP) и мощь LLM, чтобы построить **масштабируемую, надежную и интерактивно объяснимую систему**. Ниже приводится предлагаемая архитектура и стек технологий:

1. Инфраструктура данных и обучения: - **Облачное хранилище данных:** исходные файлы и последующие данные можно хранить в облаке – AWS S3, Azure Blob Storage или Google Cloud Storage. Это обеспечит доступность и версионность данных. В рамках хакатона данные хранятся локально, но в бою – на защищенном хранилище. - **Среда разработки и обучения:** использовать управляемые ноутбуки/инстансы облака. Например, **AWS SageMaker Notebook**, **Azure Machine Learning Studio notebooks** или **Google Colab/Vertex AI Workbench**. Они дают необходимые ресурсы (CPU/GPU) и доступ к данным. В хакатоне, скорее всего, использован локальный Jupyter, но архитектура подразумевает легкую переносимость в облако. - **Вычислительные ресурсы:** если для модели требуются большие вычисления (например, GNN или нейросеть), можно задействовать GPU-инстансы (p2/p3 на AWS, NC-series на Azure, A100 на GCP). Для бустинговых деревьев достаточно CPU, но GPU-реализации (LightGBM GPU, RAPIDS cuML) могут ускорить тренинг. NVIDIA RAPIDS, кстати, успешно применяли для ускорения пайплайнов на Kaggle [9]. - **Контейнеризация:** весь pipeline можно упаковать в Docker-контейнер, чтобы было удобно запускать обучение где угодно. Например, NVIDIA предоставляет контейнеры для финансового фрода с GNN, совмещающего граф и табличные данные [10]. Мы можем собрать свой, включив нужные библиотеки (PyTorch, LightGBM, etc.).

2. Модели и библиотеки: - **Python, sklearn, pandas, numpy:** базовый стек для обработки данных, вычислений, прототипирования. - **LightGBM / XGBoost / CatBoost:** основные ML-модели для табличных данных. В решении, предположительно, лучшим будет LightGBM или CatBoost, учитывая категориальные фичи и скорость. Эти библиотеки хорошо масштабируются и могут быть интегрированы с MLflow для отслеживания. - **PyTorch (и PyTorch Geometric / DGL):** используется если мы внедряем продвинутые модели – например, GNN для анализа графа транзакций, или хотим обучить нейросетевой автоэнкодер/Transformer на последовательностях сессий. PyTorch Geometric или DGL позволят построить GNN, где узлы – клиенты и получатели, ребра – транзакции, и решать задачу классификации ребер (мошенничество или нет) или узлов. Исследования показывают, что добавление **GNN-эмбеддингов в модель повышает точность обнаружения даже на 1-2%, что в деньгах очень значительно** [11] [12]. Если время хакатона не позволит реализовать GNN, можно упомянуть в архитектуре как перспектива. - **AutoML инструменты:** может использоваться `H2O.ai AutoML`, `AutoGluon`, или Optuna for hyperparam tuning. Однако, мы описали собственный AutoML-agent подход. В облаке есть готовые сервисы: AWS SageMaker Autopilot или GCP Vertex AI Tabular AutoML, но на хакатоне, вероятно, будет кастомный подход. Тем не менее, архитектура может включать опционально эти сервисы для быстрого прототипирования и сравнения. - **MLflow / Weights & Biases:** для **экспериментального трекинга** и управления моделями. MLflow (open-source) можно развернуть на сервере или использовать Managed MLflow (Databricks). Weights & Biases (W&B) – облачный сервис, легко

интегрируемый, для логирования метрик, графиков, и даже хранения артефактов модели. Мы можем настроить логирование каждой попытки модели, что особенно полезно при AutoML. Затем лучшую модель помечаем и сохраняем в модельный реестр (Model Registry) – например, MLflow Model Registry, для последующего деплоя. - **LangChain для LLM:** LangChain или аналогичные фреймворки позволят организовать работу LLM Explainer агента. Например, можно с помощью LangChain сделать так, что агент получает на вход описание модели/фичей (как контекст), и потом отвечает на вопросы. LangChain упрощает интеграцию LLM с нашим кодом и данными, можно реализовать цепочку: "возьми SHAP значения -> сформулируй объяснение". Для LLM можно использовать модель OpenAI (Azure OpenAI Service) или локальную большую модель (например, Llama 2) если важна офлайн работа. Azure OpenAI и другие сервисы позволяют подключать GPT-4, что даст высокое качество объяснений. - **SHAP & LIME:** Библиотеки для объяснимости. **SHAP** (Shapley Additive Explanations) – основной выбор, т.к. хорошо работает с деревьями (есть быстрый Tree SHAP). Будет использован для глобального и локального объяснения. **LIME** тоже можно упомянуть – например, для объяснения отдельного случая в презентации можно показать как LIME выделяет конкретные фичи, влияющие на решение, в понятиях близких к линейным. - **Evidently AI:** библиотека для мониторинга данных и качества модели. Мы можем сгенерировать отчет о сдвиге данных, важна ли она в ходе хакатона – не совсем, но для архитектуры закладываем. Например, Evidently можно настроить в пайплайне: после развертывания модель еженедельно получает отчет о дрейфе распределений (например, вдруг пользователи стали чаще менять устройство – это нужно учитывать, или изменилась доля мошенничеств). - **Другие утилиты:** Git для версионности кода, Docker/ Kubernetes для разворачивания (если думать о индустриализации). Сама модель в продакшене может быть задеплоена как **REST API** (например, с помощью Flask/FastAPI, завернутого в Docker, и размещенного на AWS ECS / Azure Container Instances / GCP Cloud Run). Поскольку банковский сервис, можно интегрировать в существующую систему: модель будет вызываться внутри backend мобильного банка или antifraud системы при каждом переводе.

3. Интеграция LLM и агентной системы: - Каждый описанный AI-агент может быть реализован как отдельный компонент. Напрямую это может быть Python-скрипт или функция. Но концептуально можно использовать LLM (например, GPT-4) с подходящим контекстом, чтобы он генерировал решения. Например, Feature Engineer Agent: можно подсунуть LLM описание данных и попросить предложить фичи – он выдаст идеи, которые инженер уже реализует. Fraud Pattern Agent может использоваться для **идеи правил**, но реализация правил – кодом. LLM Explainer Agent – наиболее прямое применение LLM (генерация текстов). - В архитектуре можно изобразить **Orchestrator**, который по цепочке вызывает функции/агентов, передает результаты. Эту роль может выполнять либо управляющий Python-скрипт, либо экспериментально LLM with planning (есть проекты, где GPT сам планирует и вызывает инструменты). Например, можно попробовать GPT-4 + LangChain Agents: дать ему инструменты (питон для данных, autoML-фреймворк, etc.) – однако в реальности на хакатоне надежнее разделить вручную. - В презентации стоит отметить, что такая **многоагентная архитектура** сама по себе инновационна. Она демонстрирует, как ИИ (в виде LLM) может автоматизировать работу DS команды: от анализа данных до подготовки отчетов. Это будущее Data Science – облегчить рутинные части и ускорить поиск решений.

4. Диаграмма архитектуры:
(в тексте опишем, но на слайде можно нарисовать блок-схему):

- **Данные:** (S3 Bucket или БД) -> **Предобработка** (AWS Glue Jobs или Python script) -> **Feature Engineering** (Notebook/Script) -> **Feature Store** (например, S3 Parquet или специализированный хранилище признаков)

- **Training**: Jupyter/SageMaker job, использующий данные из Feature Store, тренирует модель (LightGBM, etc.), логирует в MLflow/W&B. AutoML агент может параллельно запустить несколько таких джобов.
- **Model Registry:** лучшая модель сохраняется (например, MLflow Model Registry or SageMaker Model Registry).
- **Deployment:** развёртывание модели как endpoint. В AWS – SageMaker Endpoint или Lambda + container; Azure – ACI/AKS; GCP – Vertex AI Endpoint.
- **Inference pipeline:** мобильное приложение -> бэкенд банка -> вызов ML endpoint с нужными данными -> модель возвращает вероятность мошенничества -> бизнес-логика принимает решение (если > threshold, запрашивает подтверждение личности или отклоняет перевод). Это контур real-time.
- **Monitoring:** логирование результатов решений (сколько сработок, были ли false positive/ negative отмечены вручную), передача этих логов в мониторинговый сервис. Evidently Dashboard можно встраивать для визуального контроля.
- **Retraining trigger:** например, ежемесячно срабатывает AWS Step Function or Azure Data Factory pipeline: берет новые данные, пересчитывает фичи, обучает модель заново, сравнивает с текущей по качеству, и если лучше – автоматом деплоит.
- **LLM Explainer service:** Опционально, запущен сервис (например, на Azure Functions с Azure OpenAI), куда можно подать ID транзакции, и он сформирует текстовое объяснение для оператора. Такое можно предложить как часть решения: не только флаги ставим, но и объяснение “почему флагнуто”, что крайне ценно для antifraud-аналитиков.

В рамках хакатона реализуется офлайн часть (EDA, обучение, оценка), а архитектуру реального времени и MLOps мы описываем как проектную документацию. Использование облачных решений делает наше решение **масштабируемым и production-ready**: например, NVIDIA уже интегрировала свой AI Blueprint по фроду в AWS и другие экосистемы для простого прототипирования и деплоя [13]. Мы опираемся на открытые технологии (LightGBM, SHAP) и промышленные стандарты (MLflow, Kubernetes), поэтому перенести наше решение из ноутбука в бой будет минимально сложным.

5. Передовые практики и решения из Kaggle и исследований

При разработке стратегии мы ориентировались на успешные подходы из соревнований Kaggle и научных работ по обнаружению банковского мошенничества. Ключевые инсайты и best practices:

- **Массивный feature engineering и агрегации:** Как показал опыт победителей соревнования IEEE-CIS Fraud Detection, богатство признаков – залог высокого качества модели [14]. Лучшие решения генерировали сотни новых признаков: статистики по временным интервалам, количеству транзакций за периоды, комбинации полей в уникальные ID и т.д. В нашем случае мы уже имеем продвинутые агрегаты, но мы расширяем их (по получателям, по историям сумм). Практика Kaggle – создавать **UID** (unique identifier) из нескольких полей, чтобы группировать транзакции. Мы аналогично можем создать, к примеру, “client_dest_pair” = (clientID, destID) и считать по нему статистики (если повторяются), или “device_client” = (модель устройства + клиент) – вдруг у мошенника много жертв с одним устройством. Использование агрегаций по таким группам выявляет скрытые зависимости [4].
- **Взаимодействия признаков и высокомерные категориальные признаки:** В задачах мошенничества часто есть высокоразмерные категориальные признаки (ID устройств, аккаунтов). Kaggle-команды применяли **entity embeddings** – обучение представлений категорий в виде векторов, особенно эффективно в нейросетях. Также использовали прямые взаимодействия: генерировали новые фичи, объединяя категории парами/

тройками (card_id + addr, device + hour и т.п.), если это имело смысл. В нашем решении, помимо явных взаимодействий (например, флаг “новый получатель для клиента”), можно использовать технику **target encoding** для категорий (например, вероятностная частота мошенничества для данной модели телефона).

- **Ансамблирование моделей:** Редко одна модель побеждает всех – чаще ансамбль. На Kaggle чаще всего топовые решения – это ансамбли из нескольких бустинговых деревьев и нейросетей. Так, в IEEE-CIS победитель использовал ансамбль XGBoost + CatBoost + LightGBM [15]. В исследованиях 2025 года также отмечается, что стеккинг бустинговых моделей улучшает показатели [7]. Наш план тоже включает возможность ансамбля: например, усреднить предсказания нескольких бустингов с разными параметрами (bagging effect) или обучить мета-модель (логрег) на выходах нескольких алгоритмов.
- **Борьба с дисбалансом:** На Kaggle обычно оптимизируют либо AUC (нечувствителен к дисбалансу), либо используют методы вроде **stratified sampling**. Часто применяли **очистку от шумов**: т.к. могут быть ошибочно помеченные данные. В научных статьях рекомендуется использовать методы **SMOTE** или **ADASYN** для синтетических миноритарных примеров, однако с осторожностью, чтобы не переобучить. Также популярно было кастомизировать порог: не полагаться на 0.5, а подбирать с учетом максимизации определенной метрики. Мы тоже уделили внимание оптимальному порогу и использовали weight для класса 1 при обучении.
- **Графовые методы и сетевой анализ:** Современные исследования подчеркивают эффективность **Graph Neural Networks (GNN)** для финансового мошенничества [16]. В случаях, когда мошеннические схемы образуют цепочки переводов или группы связанных аккаунтов, графовые методы могут выявить тех, кто “на первый взгляд нормален, но связан с известными мошенниками” [17]. Например, если аккаунт A не мошенник, но он отправил деньги аккаунту B, который получал средства от 5 других жертв – GNN может уловить подобную связь и повысить скор для транзакции A→B. Kaggle пока редко используют GNN из-за сложности, но появляются эксперименты (ноутбуки на Kaggle с NetworkX анализом и PyG). Для нас урок – если есть время/данные, пробуем хотя бы простые **network features**: центральность узла “получатель”, количество треугольников в графе (может указывать на кольцо переводов), PageRank получателя и т.д. Мы это можем сделать Fraud Pattern агентом. Если внедрить GNN, то, как в NVIDIA Blueprint, лучше комбинировать с классической моделью: использовать эмбеддинги узлов из GNN как дополнительные признаки для XGBoost [18].
- **Временные модели и аномалия в поведении:** В некоторых решениях применяли рекуррентные нейросети или HMM для моделирования последовательности транзакций/действий. Например, научные работы предлагают детектировать аномалии в последовательности транзакций по времени (часто мошенник совершает транзакцию, не соответствующую прошлому поведению). У нас поведенческие признаки уже отражают такую идею (Z-скор, частоты). Best practice – всегда иметь признаки “*recent vs history*” (недавняя активность vs средняя) [3], что мы и сделали. Дополнительно, можно было бы использовать алгоритмы вроде Isolation Forest или one-class SVM на поведенческих данных каждого клиента, чтобы построить модель “нормального поведения” и сигналить, если текущая точка сильно выбивается. Но в соревновательных решениях такое редко давало прирост сверх хорошо подобранных фичей, поэтому наш акцент – на supervised модели с богатым feature set.
- **Explainable AI и доверие:** Тренд последних лет – не только искать лучший AUC, но и обеспечить интерпретируемость. В статье 2025 года показано объединение ансамбля XGBoost/LightGBM/CatBoost с XAI методами (SHAP, LIME) дало AUC 0.99 на датасете и при этом сохранило объяснимость каждого решения [7] [19]. Практикующие отмечают, что банки все чаще требуют **понятных объяснений** для каждого алерта, чтобы оператор мог быстро принять решение и обосновать его клиенту или аудитору [8]. Поэтому лучшие

решения на практике – это симбиоз сильной модели и мощных инструментов объяснения. Мы в соответствии применяем SHAP, готовим визуализации и даже планируем LLM для генерации текстовых объяснений.

- **Примеры удачных подходов:** На Kaggle есть конкурсы родственные нашей задаче: помимо IEEE-CIS (кредитные транзакции), был конкурс **TalkingData Mobile Fraud** (реклама, but pattern similar), **Paypal Fraud detection challenge** (от PayPal – закрытый). Победители там в основном использовали градиентный бустинг + тонны признаков, фокусировались на память о прошлых событиях (history of device, IP, etc.). Еще есть примеры с использованием **геолокации** или **IP-адресов** – в нашем датасете их нет, но будь они, стоило бы учесть (например, страна IP).
- **Частота обновления и встраивание модели:** Практика показала, что в антифроде модель быстро устаревает, т.к. мошенники адаптируются. Поэтому важна гибкость: **онлайн-обучение** или частое переобучение. Kaggle этим не занимается, но реальный кейс – отклик на дрейф. Мы предусматриваем в архитектуре мониторинг дрейфа и автоматическое переобучение, основываясь на best practice из MLOps.
- **Дополнительные идеи из research:** Некоторые исследования предлагают рассматривать проблему как ранжирование аномалий или использовать **cost-sensitive learning** (разные веса ошибок, т.к. пропустить мошенничество хуже, чем лишний раз проверить клиента). Мы могли бы включить кастомную функцию потерь, где FN штрафуется выше, добиваясь почти 100% Recall. Но нужно сохранить Precision, иначе служба безопасности захлебнется. Поэтому баланс – ключевой. Современные алгоритмы (CatBoost, XGBoost) позволяют указать weight для классов – мы это используем вместо cost-sensitive ручной функции.
- **Методы противодействия фроду:** Параллельно с моделью часто используются простые правила (например, “если сумма > X и устройство новое – отправить на проверку независимо от модели”). Комбинация **правил и ML** – отличная практика. В hackathon-решении мы фокусируемся на ML, но можем упомянуть, что модель может выдавать скор, а поверх бизнес-правила принимают решение (например, если скор средний, но правило критичное сработало – все равно блокировать, и наоборот).
- **Kaggle код и примеры:** Во время решения можем ссылаться на публичные ноутбуки. Есть примеры ноутбуков с GNN для фрода, с визуализациями. Но главное – опыт Kaggle подтвердил, что табличный бустинг с хорошо продуманными признаками очень силен: даже сложные паттерны выучиваются. Так, в IEEE-CIS победили по сути “грубые” агрегаты по клиентам, что говорит: **domain knowledge + feature engineering > any fancy model** (при сходных данных) [14] . Мы эту философию применяем.

В итоге, наше решение вобрало лучшее из обоих миров: от **сообществ соревнований** – упор на признаки, ансамбли и тщательную валидацию; от **академических исследований** – современные подходы (графы, XAI, адаптивность). Это обеспечивает высокое качество и инновационность, что обязательно отметит жюри.

6. Визуализация результатов и презентация решения

Для успешной защиты решения на финале хакатона необходимо ярко и понятно продемонстрировать ключевые результаты. Наша стратегия визуализации и презентации:

- **Диаграмма Pipeline:** Начать презентацию с общей схемы нашего решения (как данные превращаются в модель и в итоге в предупреждение мошенничества). Это может быть блок-схема, напоминающая архитектуру: Data → Features → Model → Prediction → Action. Такая схема покажет системность подхода команды.
- **Графики качества модели:** Обязательно включаем:

- *ROC-кривая* с подписью AUC. На графике отметить точку нашего выбранного порога. Показать, что AUC высок (например, 0.95), кривая близка к верхнему левому углу – то есть модель значительно лучше случайного угадывания.
- *Precision-Recall кривая* (особенно если AUC-PR ~ высок, например 0.50+, что очень хорошо при 1% позитивов). Можно сказать: “при recall 90% precision составляет ~75%, что означает 3 из 4 сработок – настоящие мошенничества. Это отличный показатель, означающий эффективную фильтрацию.”
- *Confusion Matrix* – визуально (таблица 2x2 или матрица в виде квадратиков). Подписи: True Fraud, False Fraud, etc. Желательно рядом указать бизнес-интерпретацию: “Из 165 мошеннических атак модель поймала 155, пропустив 10 (эти 10 будут убытками, но это лишь 6% от всех атак). Ложно обвинено 50 транзакций из ~13k, то есть 0.4% всех переводов – каждого 250-го честного клиента затронет проверка. Это компромисс, который банк может выдержать ради безопасности.”
- *Histogram of model scores:* опционально, показать распределение предсказанных вероятностей для двух классов. Продемонстрировать, что мошенничества имеют значительно более высокие скоры, хотя есть небольшой overlap.
- **Feature Importance визулизации:**
- *Bar chart mon-10 признаков:* столбики важности (например, из LightGBM gain). На первом месте, допустим, “monthly_phone_model_changes – 28%”, второе “dest_unique_senders_count – 20%”, “amount – 15%”, ... Такой график наглядно показывает, что решение опирается на разумные факторы. Мы рядом поясняем: “ТОП-3 признака: 1) Кол-во смен устройства: мошенники часто меняют устройства; 2) Подозрительность получателя: некоторые получатели собирают деньги от многих клиентов; 3) Сумма перевода: слишком большая сумма часто сигнализирует о мошенничестве”.
- *SHAP summary plot:* если время позволит, это очень впечатляющий график для жюри технического уровня. Он показывает влияние каждого признака на предсказание. Мы можем прям на слайде обвести пару моментов: например, для признака “freq_change_7d_vs_mean” (рост частоты логинов) видно, что большая величина (красная) в основном двигает SHAP в плюс (мошенничество) – значит, недавний всплеск активности типичен для мошенников. Такую интерпретацию и проговариваем. Жюри увидит, что мы не просто нагенерировали фичей, но и понимаем их эффект.
- *Пример локального объяснения:* можно взять один реальный случай из данных (который модель пометила мошенничеством) и расписать его профиль: “Клиент ID123, перевод 500k KZT ночью на счет (direction) со множеством отправителей; за день до этого 12 входов, 3 разных устройства, большой перерыв до этого – модель дала скор 0.98. На графике SHAP видим, что основную вклад внесли: device_changes (SHAP +0.3), amount (+0.2), dest_senders_count (+0.15).” И представить это текстом как история, как сделал бы Explainer агент. Такая **storytelling** форма сделает решение живым.
- **Демонстрация AI-агентов:** раз это часть задания, стоит показать пример работы хотя бы одного. Например, вывести кусочек **prompt и ответа LLM Explainer**. Сделать мини-диалог:
- Вопрос: “Почему модель считает транзакцию X мошеннической?”
- Ответ (от LLM Explainer): “Судя по модели, транзакция X подозрительная, потому что устройство, с которого она проведена, не совпадает с прежними устройствами клиента, и активность клиента за день взлетела перед переводом. Кроме того, получатель платежа ранее получал деньги от 7 разных клиентов за короткий период – это типичный признак мошенничества (мул-счет). Поэтому модель присвоила высокий риск 92%.” Это произведет впечатление, т.к. покажет практическую пользу LLM: понятное объяснение для оператора банка.
- **Сравнение с базовой линией:** стоит включить слайд/график, где мы покажем, как мы улучшали результат: например, “До добавления поведенческих признаков AUC был 0.80,

после стал 0.95; без графовых фичей Recall был 80%, с ними – 90%". Это продемонстрирует вклад наших идей. Можно таблицу или просто утверждения.

- **Презентация бизнес-ценности:** Финальные слайды посвятить тому, чего добилось решение:
- "Модель ловит X% мошенничества, сокращая потери примерно на Y тенге в месяц (если известна средняя сумма мошенничества)."
- "Система объяснима: для каждого алерта генерируется причина, что повышает доверие службы безопасности и клиентов."
- "Решение легко интегрируется в мобильный банк: можно разворачивать в облаке и масштабировать под нагрузку (модель проверяет транзакцию за <100 мс)."
- "Также, подход расширяем: новые атаки можно быстро обучить модель распознавать, pipeline автоматизирован. Данные, поступающие со временем, будут учтены – модель самообучается каждый месяц, оставаясь актуальной."

- Возможно упомянуть, что **команда** (набор агентов) может быть переиспользована для других задач банка – например, для обнаружения мошенничества с кредитами, или аномалий в поведении клиентов (не ограничиваемся переводами).

- **Визуальная привлекательность:** Использовать цвета, иконки (например, значок щита для мошенничества, значок банка, диаграммы). Главное – **не перегружать текстом** на слайдах, а дать ключевые картинки и тезисы. Данный документ служит технической инструкцией, а из него для слайдов отбираются самые важные визуалы и bullets.

- **Учет критериев презентации:** Если жюри оценивает новизну – подчеркнуть наш мульти-агентный подход с LLM (новинка). Если важна глубина проработки – показать, что мы сделали и EDA, и модель, и MLOps план. Если важна применимость – упомянуть, что можем сразу пилотировать это в инфраструктуре ForteBank (например, развернуть в их Azure облаке, если у банка Azure).

Подводя итог, на защите решения мы показываем **глубокое понимание проблемы**, **отличные результаты модели с понятными метриками**, **инновационные элементы (графы, LLM, autoML)** и **бизнес-влияние**. Такой всеобъемлющий подход, подкрепленный наглядными графиками (ROC, confusion matrix, SHAP) и живыми объяснениями, обязательно произведет впечатление на жюри. Мы не только достигли высоких цифр, но и можем доказать, что решение будет работать в реальном мире и приносить пользу банку.

Источники и вдохновение: В процессе решения мы опирались на опыт мирового сообщества (соревнование IEEE-CIS Fraud Detection [14] [4]), новейшие разработки NVIDIA в области графового AI для финансов [16] [6], а также научные работы по объяснимому ИИ в финансовом мошенничестве [7] [19] – все это помогло сформировать лучший подход специально под задачу ForteBank.

---

[1] [2] [5] [6] [10] [11] [12] [13] [16] [17] [18] Supercharging Fraud Detection in Financial Services with Graph Neural Networks (Updated) | NVIDIA Technical Blog
https://developer.nvidia.com/blog/supercharging-fraud-detection-in-financial-services-with-graph-neural-networks/

[3] [4] [9] [14] [15] Leveraging Machine Learning to Detect Fraud: Tips to Developing a Winning Kaggle Solution | NVIDIA Technical Blog
https://developer.nvidia.com/blog/leveraging-machine-learning-to-detect-fraud-tips-to-developing-a-winning-kaggle-solution/

7 8 19 Financial Fraud Detection Using Explainable AI and Stacking Ensemble Methods
https://arxiv.org/html/2505.10050v1