**Отчет по работе с датасетом заказов и созданию модели по поиску фродовых заказов.**

**Обзор данных и подход:**

Датасет, на котором предстояло обучаться, содержит больше 105 тысяч строк. Был дан набор признаков:

* user\_id - идентификатор пользователя
* created\_date - дата создания записи
* nm\_id - идентификатор товара
* total\_ordered - общее количество заказанных товаров
* payment\_type - тип оплаты
* is\_paid - флаг оплаты (булево значение)
* count\_items - количество товаров за последнее время
* unique\_items - количество уникальных товаров за последнее время
* avg\_unique\_purchase - среднее количество уникальных покупок за последнее время
* is\_courier - флаг доставки курьером
* nm\_age - возраст товара
* Distance - расстояние
* days\_after\_registration - дни после регистрации
* number\_of\_orders - количество заказов
* number\_of\_ordered\_items - количество заказанных товаров
* mean\_number\_of\_ordered\_items - среднее количество заказанных товаров
* min\_number\_of\_ordered\_items - минимальное количество заказанных товаров
* max\_number\_of\_ordered\_items - максимальное количество заказанных товаров
* mean\_percent\_of\_ordered\_items - средний процент заказанных товаров
* target - целевая переменная (0 - не требуется блокировка, 1 - требуется блокировка)
* service - тип региона

Был заявлен сильный дисбаланс классов в соотношении 1:10 (класс 1 встречался значительно реже).

**Обработка выбросов**:

- Значения выше 95-го перцентиля заменены на 85-й перцентиль для признаков: `total\_ordered`, `count\_items`, `NmAge` и др.

- Логарифмирование агрегированных признаков (`user\_total\_ordered\_sum\_log`).

2. **Преобразование признаков**:

- Создание временных признаков (`dayofweek`, `hour`) из `CreatedDate`.

- Группировка редких категорий (`PaymentType`: 'RARE' для значений кроме 'CSH' и 'CRD').

- Агрегация данных пользователя: сумма заказов за 7/30 дней, время с последнего заказа.

**Архитектура модели**

- **Алгоритм**: CatBoostClassifier.

- **Причины выбора**:

- Эффективная работа с категориальными признаками без предварительного кодирования.

- Устойчивость к переобучению за счёт регуляризации и ансамблевого подхода.

**Ключевые гиперпараметры**

- **Основные параметры**:

- `max\_depth=8`, `n\_estimators=40`, `learning\_rate=0.05`

- `l2\_leaf\_reg=13` (L2-регуляризация), `min\_data\_in\_leaf=30`

- `subsample=0.7`, `bagging\_temperature=0.5`

- **Кастомные веса признаков**:

- `total\_ordered=4.0`, `user\_total\_ordered\_sum=1.0` (усиление влияния ключевых признаков).

**Результаты**

**Основные метрики (миноритарный класс)**

- **Precision**: 0.906

- **Recall**: 0.139

**Дополнительные метрики**

- **F1-score**: 0.24

- **PR-AUC**: 0.577

- **ROC-AUC**: 0.780

**Матрица ошибок и анализ**

Confusion Matrix:

[[12429 26]

[1554 252]]

Высокий Precision: Модель редко ошибается в предсказании положительного класса.

Небольшой Recall: Ощутимая часть объектов миноритарного класса не распознаётся.

**Рукводство**

Запуск проекта с помощью docker-compose.yml:

----

services:

postgres:

image: postgres:15

container\_name: fraud-postgres

restart: unless-stopped

environment:

POSTGRES\_DB: HakatonWB

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: 123

ports:

- "15432:5432"

volumes:

- postgres\_data:/var/lib/postgresql/data

healthcheck:

test: ["CMD-SHELL", "pg\_isready -U postgres"]

interval: 5s

timeout: 5s

retries: 5

networks:

- fraud-network

fraud-backend:

image: bebekon/fraud-back

container\_name: fraud-backend

restart: unless-stopped

depends\_on:

postgres:

condition: service\_healthy

environment:

- ConnectionStrings\_\_Database=Host=postgres;Port=5432;Database=HakatonWB;User Id=postgres;Password=123;

- ASPNETCORE\_ENVIRONMENT=Development

ports:

- "5050:8080"

- "8081:8081"

networks:

- fraud-network

extra\_hosts:

- "host.docker.internal:host-gateway"

fraud-ml-service:

image: bebekon/fraud-ml

container\_name: fraud-ml-service

restart: unless-stopped

ports:

- "8000:8000"

depends\_on:

fraud-backend:

condition: service\_started

networks:

- fraud-network

fraud-frontend:

image: bebekon/fraud-front

container\_name: fraud-frontend

restart: unless-stopped

depends\_on:

- fraud-backend

- fraud-ml-service

ports:

- "5173:5173"

environment:

- NODE\_ENV=development

volumes:

- /app/node\_modules

- /app/.vite

command: [ "npm", "run", "dev", "--", "--host", "0.0.0.0", "--port", "5173" ]

networks:

- fraud-network

networks:

fraud-network:

driver: bridge

volumes:

postgres\_data:

----

Далее в консоли в текущей директории docker-compose up -d

Для frontend: <http://localhost:5173>

Для backend (swagger-ui): <http://localhost:5050/swagger/index.html>

Для ML (swagger-ui): <http://localhost:8000/docs>