Обнаружение мошенничества в финансовых операциях

Выполнено студентами 3 курса 22ВТ-09.03.03.01-о3:

Овод
Алексей Михайлович

Усачев
Илья Витальевич

27.06.2025

Информация о проекте

README

Fraud Analytic

Fraud Analytic — это потоковая система обнаружения мошенничества в финансовых транзакциях. Она использует Apache Kafka для получения данных, Apache Spark ML для обработки и предсказания, а Cassandra для хранения результатов. Модель машинного обучения обучается на датасете с Kaggle и оценивает вероятность мошенничества в реальном времени. Результаты доступны через Flask API, которые можно визуализировать с помощью Power BI.

Установка и запуск

1. Копируем репозиторий:

Веб-интерфейсы

1. Dashboard (Мониторинг мошеннических операций в реальном времени): Веб-интерфейс на Flask с визуализацией последних данных о мошеннических операциях. Отображает динамику вероятности мошенничества по времени с помощью графиков Chart.js, таблицу последних транзакций и статистику (количество операций, среднюю вероятность, сумму).
   Адрес: http://localhost:5000
2. Flask API (Просмотр данных о мошенничестве): Flask-сервер предоставляет простой API для доступа к данным из Cassandra. Возвращает JSON с записями из таблицы fraud.alerts. Адрес: http://localhost:5000/fraud

Пример ответа:

```
[
  {
    "amount": 316.06,
    "clazz": 1.0,
    "fraud_probability": 0.9089357093357741,
    "prediction": 1.0,
    "time": 93853.0
  }
]
```

3. Spark UI (Мониторинг заданий): После запуска контейнеров вы можете отслеживать выполнение Spark-задач через веб-интерфейс. Адрес: http://localhost:8080

Что можно посмотреть:

- Активные и завершенные задания (Jobs)
- Распределение ресурсов между исполнителями (Executors)
- Логи выполнения задач (Stages)

Авторы

- @lunyamq
- @UsachevDev

Packages

No packages published
Publish your first package

Contributors 2

lunyamq lunya

UsachevDev Ilya Usachev

Languages

Python 94.9% Dockerfile 2.6%
Shell 2.5%

Команда проекта

Lunyamq

Овод Алексей

https://github.com/lunyamq

UsachevDev

Усачев Илья

https://github.com/UsachevDev

Ссылка на репозиторий

https://github.com/lunyamq/fraud-analytic

Актуальность

1. **Стремительный рост объёма электронных платежей и мобильных переводов.**

   С каждым годом всё больше операций проводится онлайн и через мобильные приложения, что расширяет поверхность для мошеннических схем и увеличивает объём данных, требующих моментального анализа.

2. **Усложнение и автоматизация методов мошенников.**

   Традиционные правила и пороговые фильтры не успевают адаптироваться к новым видам атак: сценарии становятся более изощрёнными, используют машинное обучение и распределённые сети ботов.

3. **Высокие репутационные потери для банков и платёжных сервисов.**

   Даже единичный инцидент с утечкой средств или ложным срабатыванием подрывает доверие клиентов, приводит к оттоку пользователей и штрафам за нарушение стандартов безопасности.

4. **Жёсткие требования регуляторов и внутренних политик по защите данных.**

   Законодательство и отраслевые стандарты (PCI DSS, ФЗ-152, PSD2) требуют оперативного мониторинга и отчётности по подозрительной активности, включая уведомления в режиме реального времени.

Цель и задачи

Цель:

- Создать систему, которая в режиме реального времени обнаруживает подозрительные операции и оценивает вероятность мошенничества.

Задачи:

- Проанализировать и подготовить исходный dataset.
- Обучить и протестировать модель для классификации мошеннических операций.
- Организовать потоковую передачу данных с помощью Apache Kafka и Spark Streaming.
- Реализовать запись результатов анализа в масштабируемое хранилище (Cassandra).
- Провести визуализацию полученных результатов и оценить эффективность системы.

Информация о датасете

Источник данных

Данные взяты из открытого репозитория Kaggle Credit Card Fraud Detection, содержащего сведения о транзакциях по банковским картам клиентов одного из европейских банков за два дня.

Объём и состав

Всего ~285 000 записей, из них 492 мошеннических (0,17 %)

Признаки: 28 столбцов с анонимизированными признаками (V1–V28), а также поля Time, Amount и Class (целевой признак: 1 — мошенничество, 0 — нет).

Предварительная обработка

- Выбраны признаки V1–V28 (результат PCA)
- Векторизация признаков через VectorAssembler
- Разделение данных на обучающую и тестовую выборки (70/30)
- Обучение модели RandomForestClassifier

Информация о датасете

creditcard.csv (150.83 MB)

Detail Compact Column 5 of 31 columns

About this file

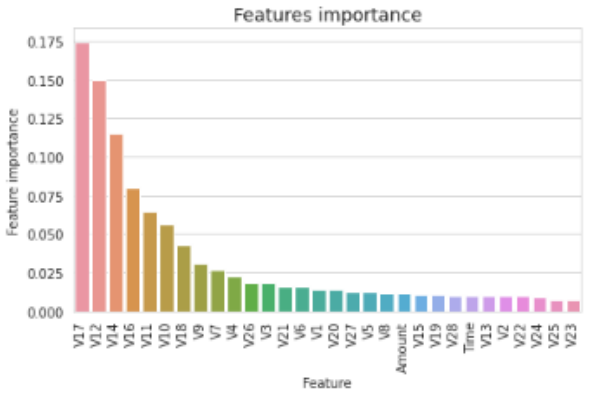
BigQuery Table bigquery-public-data.fraud_detection.comments

# Time Number of seconds elapsed between this transaction and the first transaction in the dataset	# V1 may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)	# V28 abc	# Amount Transaction amount	# Class 1 for fraudulent transactions, 0 otherwise
0 – 173k	-56.4 – 2.45	-15.4 – 33.8	0 – 25.7k	0 – 1
0	-1.3598071336738	-0.0210530534538215	149.62	0
0	1.19185711131486	0.0147241691924927	2.69	0
1	-1.35835406159823	-0.0597518405929204	378.66	0
1	-0.966271711572087	0.0614576285006353	123.5	0
2	-1.15823309349523	0.215153147499206	69.99	0
2	-0.425965884412454	0.0810802569229443	3.67	0
4	1.22965763450793	0.00516776890624916	4.99	0
7	-0.644269442348146	-1.08533918832377	40.8	0

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Информация о датасете

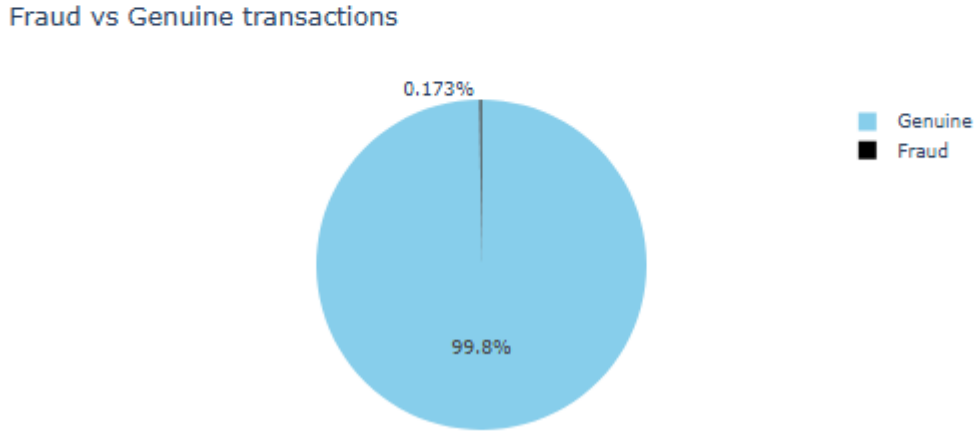
In [29]:
```
tmp = pd.DataFrame({'Feature': predictors, 'Feature importance': clf.feature_importances_})
tmp = tmp.sort_values(by='Feature importance',ascending=False)
plt.figure(figsize = (7,4))
plt.title('Features importance',fontsize=14)
s = sns.barplot(x='Feature',y='Feature importance',data=tmp)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
plt.show()
```



In [4]:
```
labels=["Genuine","Fraud"]

fraud_or_not = raw_df["Class"].value_counts().tolist()
values = [fraud_or_not[0], fraud_or_not[1]]

fig = px.pie(values=raw_df['Class'].value_counts(), names=labels , width=700, height=400, color_d
iscrete_sequence=["skyblue","black"]
             ,title="Fraud vs Genuine transactions")
fig.show()
```



Ход работы и используемые инструменты

Подготовка данных и обучение модели

Загрузка исходного датасета

Построение пайплайна в Apache Spark ML (VectorAssembler → RandomForestClassifier)

Разбиение на обучающую и тестовую выборки (70 %/30 %), кросс-валидация

Сохранение обученной модели

Хранение и визуализация результатов

Apache Cassandra: запись транзакций с метками риска

Power BI: графики для анализа качества модели и распределения вероятностей мошенничества

Flask: мониторинговый веб-интерфейс, передача данных с Cassandra

Потоковая передача данных

Kafka Producer (producer.py на Python): отправка транзакций в топик fraud-topic

Spark Structured Streaming (consumer.py): чтение из Kafka → применение модели → генерация предсказаний

Сообщения в формате JSON, триггер обработки каждые 3 секунды

Установка и запуск

Container CPU usage
8.49% / 1200% (12 CPUs available)

Container memory usage
3.31GB / 7.5GB

Show charts

Search

Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
fraud-analytic	-	-	-	16.14%	2 minutes ago	
processor-1	94ec4bcf0103	fraud-analytic-processor		0.01%	2 minutes ago	
flask-1	9616615583b8	fraud-analytic-flask	5000:5000	0.02%	2 minutes ago	
kafka-1	7fdd63c3348f	confluentinc/cp-kafka:7.5.0	9092:9092	9.46%	2 minutes ago	
cassandra-1	98b7522b709e	cassandra:4.1	9042:9042	6.19%	2 minutes ago	
zookeeper-1	ee7b0c751157	confluentinc/cp-zookeeper:7.5.0		0.21%	2 minutes ago	
spark-1	77f3ad5dac12	bitnami/spark:3.5	8080:8080	0.25%	2 minutes ago	

2. Запускаем Docker и собираем контейнеры:

```
docker-compose up --build
```

3. Подключаемся к Cassandra:

```
docker exec -it fraud-analytic-cassandra-1 cqlsh
```

Установка и запуск

README

4. Добавляем keyspace и таблицу:

```
CREATE KEYSPACE IF NOT EXISTS fraud
  WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

CREATE TABLE IF NOT EXISTS fraud.alerts (
    time double,
    amount double,
    fraud_probability double,
    prediction double,
    clazz double,
    PRIMARY KEY (time, amount)
);
```

5. В новом терминале заходим в контейнер processor:

```
docker exec -it fraud-analytic-processor-1 bash
```

6. Скачиваем dataset:

```
./download.sh
```

7. Обучаем модель:

```
spark-submit model_training.py
```

8. Запускаем `producer.py`:

```
python3 producer.py
```

9. В новом терминале снова заходим в контейнер processor:

```
docker exec -it fraud-analytic-processor-1 bash
```

10. Запускаем `consumer.py`:

```
./sparkrun.sh consumer.py
```

11. После завершения работы producer.py проверяем результаты в Cassandra:

```
SELECT * FROM fraud.alerts;
```

Установка и запуск

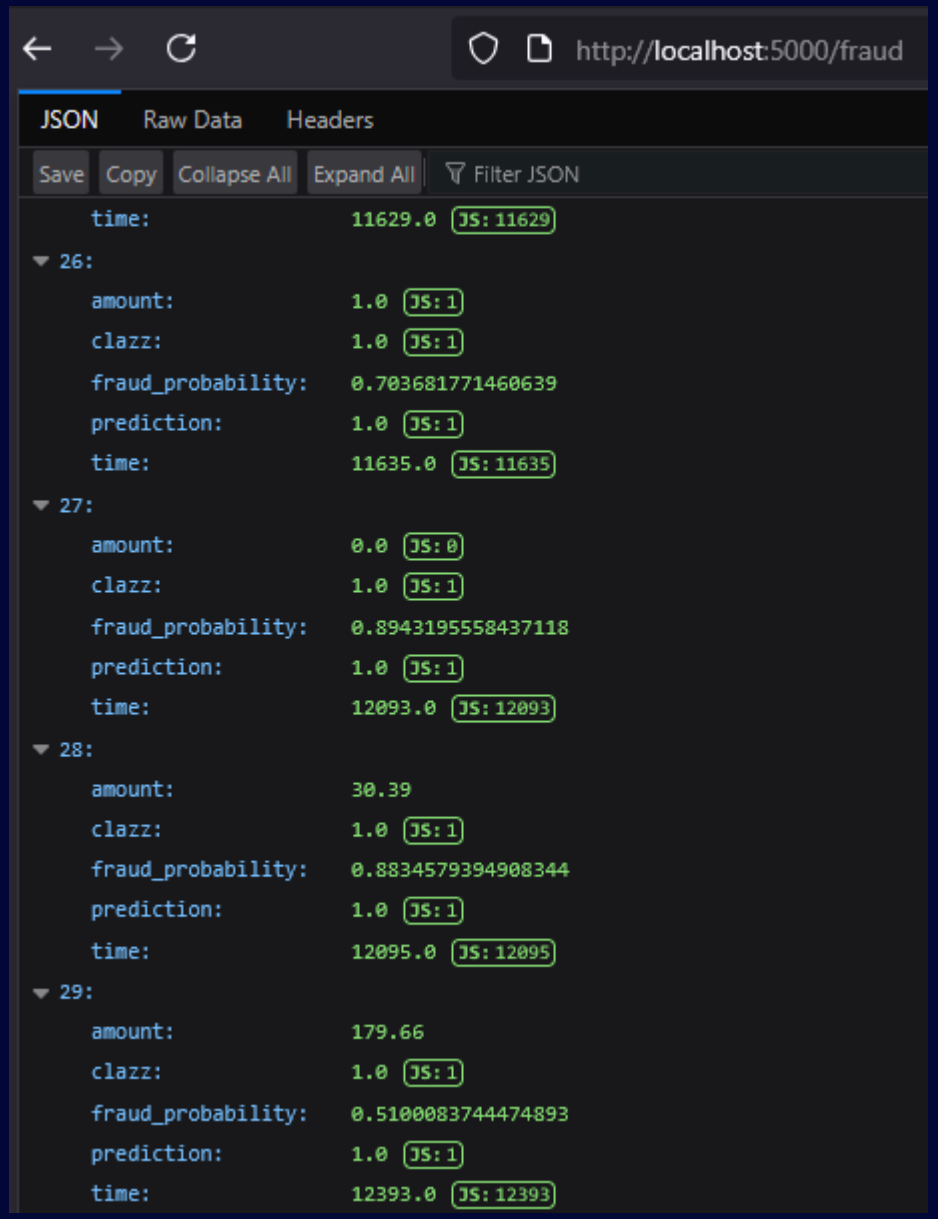
```
C:\Users\lunqa\Documents\fraud-analytic>docker exec -it fraud-analytic-processor-1 bash
root@9470a992183c:/app# python3 producer.py
  1%|                                                                                | 3421/284807 [00:05<07:11, 652.03it/s]
```

```
C:\Users\lunqa\Documents\fraud-analytic>docker exec -it fraud-analytic-processor-1 bash
root@9470a992183c:/app# ./sparkrun.sh consumer.py
+-------+------+-----------------+----------+-----+
|time   |amount|fraud_probability|prediction|clazz|
+-------+------+-----------------+----------+-----+
|47982.0|0.76  |0.8686631574080 54|1.0       |1.0  |
+-------+------+-----------------+----------+-----+
```

```
C:\Users\lunqa\Documents\fraud-analytic>docker exec -it fraud-analytic-cassandra-1 cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.9 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> SELECT * FROM fraud.alerts;

 time       | amount  | clazz | fraud_probability | prediction
------------+---------+-------+-------------------+------------
      41227 |  459.07 |     1 |          0.808634 |          1
  1.005e+05 |    0.76 |     1 |          0.653369 |          1
      20823 |       1 |     0 |          0.630376 |          1
  1.4618e+05 |    0.77 |     1 |          0.770868 |          1
  1.6066e+05 |    0.77 |     1 |          0.910239 |          1
```

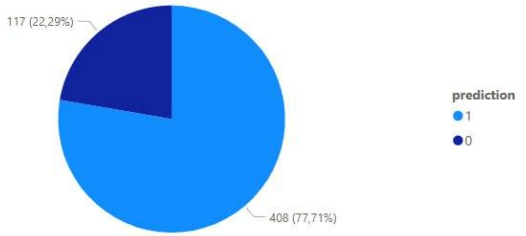
Передача данных



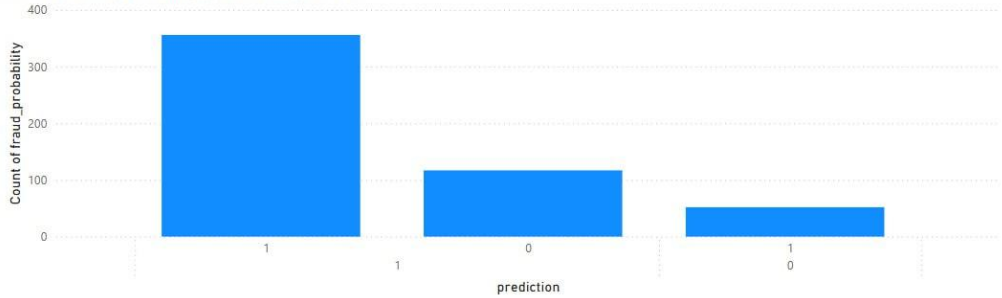
Визуализация в Power BI

Count of clazz by prediction

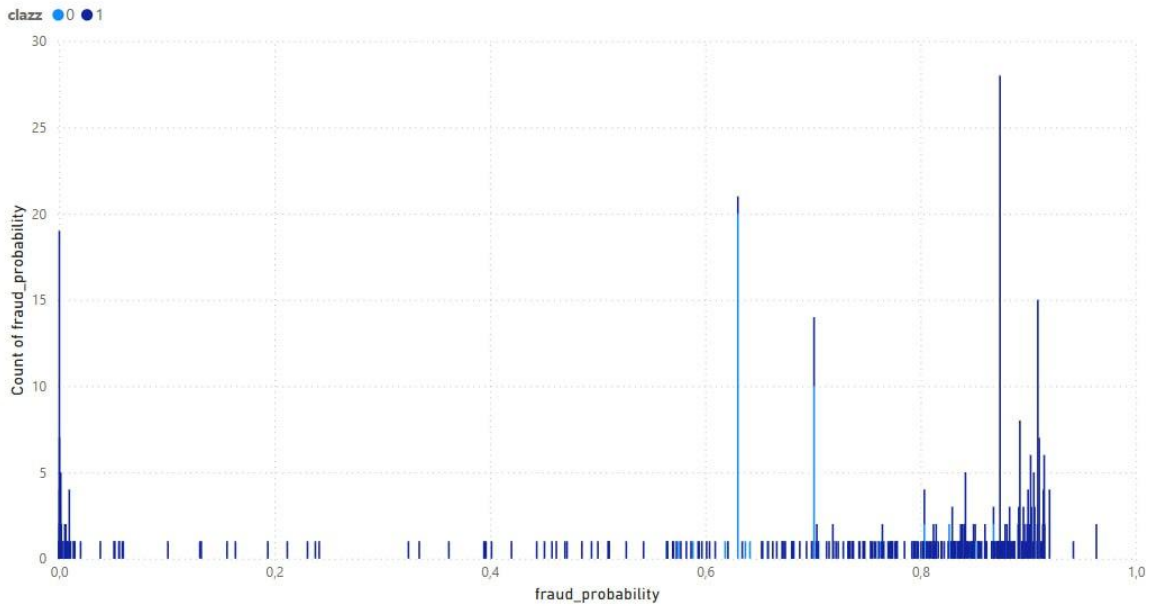
clazz	0	1	Total
0		52	52
1	117	356	473
Total	117	408	525



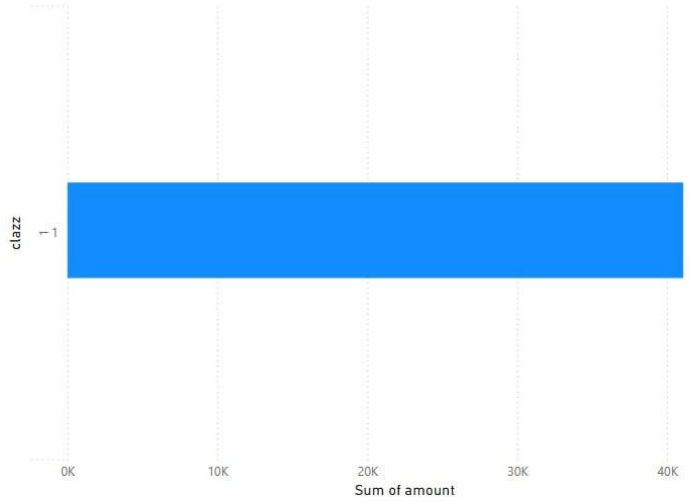
Count of fraud_probability by clazz and prediction



Count of fraud_probability by fraud_probability and clazz



Sum of amount by prediction and clazz



Визуализация в дашборде

http://localhost:5000

Дашборд мошеннических транзакций

Обновлено: 9:55:33 PM

Мошеннические транзакции	Ср. вероятность	Сред. сумма	Макс. сумма	Последняя операция
317	0.81	103.53	1809.68	121238

График: динамика вероятности мошенничества по времени

Последние мошеннические транзакции

Время	Сумма	Вероятность
121238	144.62	0.874
118603	105.89	0.877
118532	105.89	0.848
117109	119.99	0.772
116195	19.99	0.763
116067	65	0.778
115691	17.06	0.816
110552	80.22	0.860
110547	0.76	0.876
109298	186.13	0.914
109297	0	0.904
103808	2.27	0.663
103455	1.52	0.850
102676	119.74	0.905

Выводы и заключение

Точность обнаружения

Модель RandomForest показала AUC ≈ 0.97 на отложенной выборке.

Обработка в реальном времени

Spark Structured Streaming обрабатывает пакеты с минимальной задержкой, обеспечивая своевременное оповещение о рисках.

Масштабируемость и надёжность

Комбинация Kafka и Cassandra выдерживает нагрузку, а также осуществляет автоматическую балансировку и резервирование.

Разработанная система готова к промышленному внедрению: она оперативно обнаруживает мошенничество, легко масштабируется и может быть расширена новыми алгоритмами и источниками данных.

Спасибо за внимание!

Овод
Алексей Михайлович
Усачев
Илья Витальевич