



Поиск мошеннических операций в платежных транзакциях интернет-аукциона

[ВЛАДИМИР НИКИФОРОВ]

1. ПОСТАНОВКА ЗАДАЧИ

КАКАЯ ПРОБЛЕМА

В нашем мире постоянно растущей конкуренции на всех рынках качество и безопасность обслуживания, а значит и лояльность клиентов ставится на первое место среди приоритетов бизнеса.

В сфере электронных платежей для этого требуется быстрый и качественный анализ транзакций на фрод.

Необходимо с точностью не ниже 95% **классифицировать** транзакции на **легитимные** и **фрод**.

- При этом **классы** сильно **несбалансированные** (2% фрод-транзакций).
- Бизнесу важно, чтобы модель **минимум блокировала легитимные транзакции** и **максимум** блокировала **фрод**.
- И **максимально быстро** - это транзакции!

Результат работы модели: по вероятности класса «фрод» к каждой транзакции нужно применить одно из действий:

- **PASS** – транзакция не является подозрительной, пропускаем
- **ALERT_AGENT** – о транзакции следует сообщить наблюдателям
- **LOCK_USER and ALERT_AGENT** – транзакцию следует заблокировать и сообщить наблюдателям для анализа
- **LOCK_USER** – транзакцию строго блокируем, она является мошеннической



КАК РЕШАЛ ЗАДАЧУ

- Шаг 0 **Первичный анализ** задачи и **выдвижение гипотез** на основе EDA
- Шаг 1 **Feature Engineering**, **балансировка классов** для обучения
- Шаг 2 **Создание baseline модели** – логистической регрессии
- Шаг 3 **Создание более сложных моделей**, анализ прироста качества, подбор гиперпараметров
- Шаг 4 **Понижение размерности**
- Шаг 5 **Создание промышленной реализации** задачи в виде API, возвращающее необходимое «действие» над транзакцией для указанных признаков транзакции



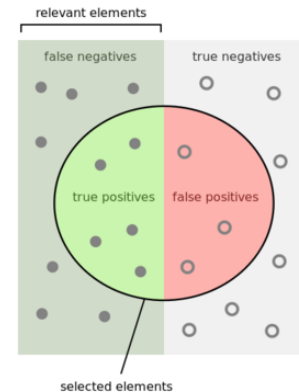


ЦЕЛЕВЫЕ МЕТРИКИ

- Метрика для **сравнения моделей и подбора гипер-параметров** (после балансировки классов) – **ROC–AUC**
- Метрики для **трансляции результатов бизнес-заказчикам** – **Precision** (доля срабатывания модели от всего потока транзакций) и **Recall** (доля пойманных мошеннических¹ транзакций от всех)

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$



How many selected items are relevant?

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

1. Мошенническими транзакциями будем считать оценки классификатора выше 50%

2. АНАЛИЗ

КАКИЕ ЕСТЬ АНАЛОГИ

Аналог-1

Detecting Credit Card Fraud Using Machine Learning^[1]

Недостатки:

- Отсутствует промышленное применение



Аналог-2

SAFE: A Neural Survival Analysis Model for Fraud Early Detection^[2].

Недостаток:

- Черный ящик для бизнеса

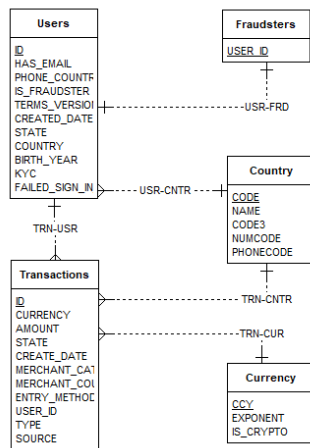


EDA

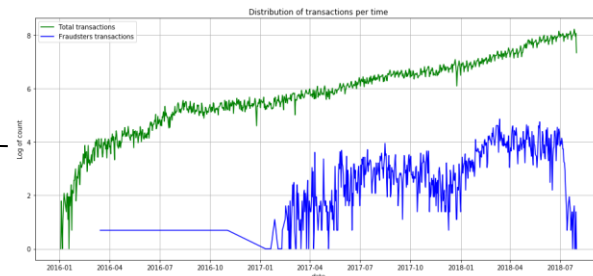
Исходные данные:

1. **Транзакции** пользователей (688 651, 11)
2. Справочник **пользователей** (9 944, 11)
3. Справочник стран (226, 5)
4. Справочник валют (184, 3)
5. Список злоумышленников (298, 1)

ERD



Log(count) всех
и фрод
транзакций



Созданы признаки:

1. **PROFILE_AGE** – возраст профиля на момент транзакции
2. **HOMELAND** – признак проведения транзакции в стране из профиля пользователя
3. **ENTRY_METHOD_*** - OneHotEncoding категориального признака ENTRY_METHOD
4. **TYPE_*** - OneHotEncoding категориального признака TYPE
5. **HOUR_OF_TRANSACTION** – час из времени проведения транзакции
6. **HOMELAND_PHONE** – признак соответствия страны профиля пользователя стране указанного в профиле телефона

Поиск мошеннических операций

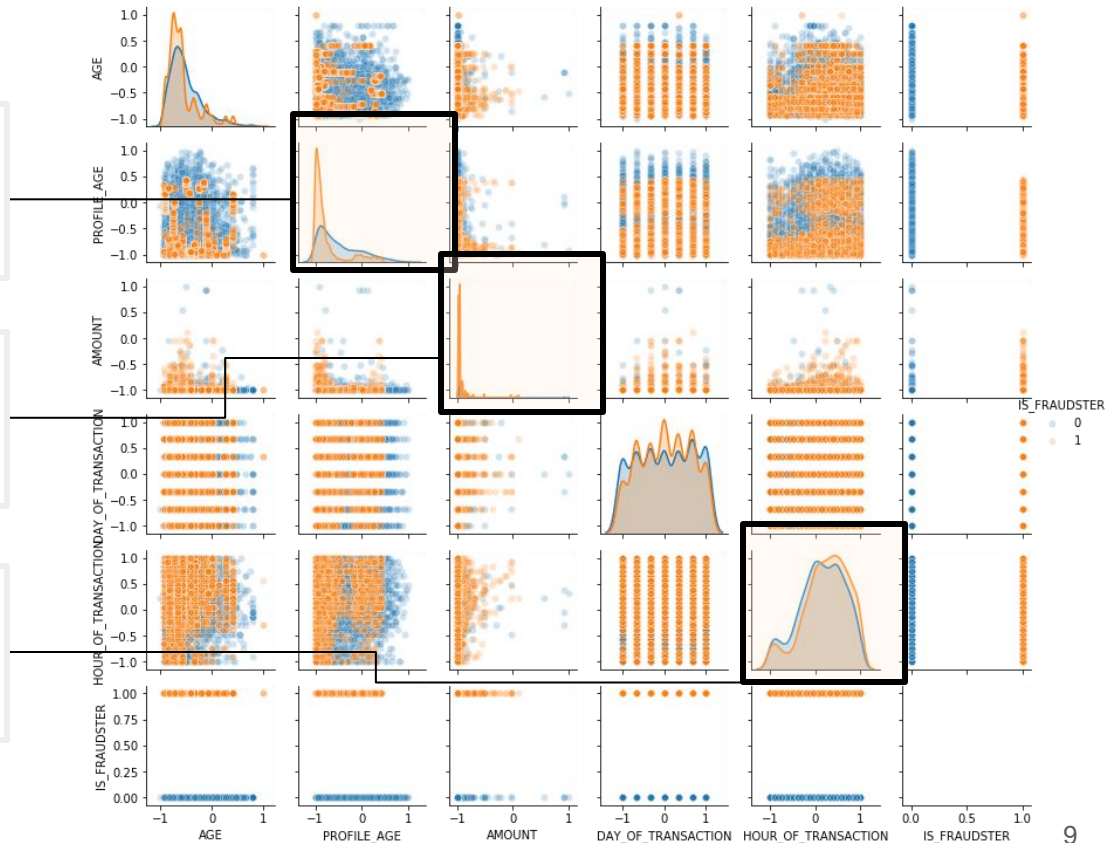
[ВЛАДИМИР НИКИФОРОВ]

EDA

Средний возраст профиля фрод-транзакции отличается от среднего возраста обычного пользователя

Средний объем фрод-транзакций отличается от среднего объема транзакции обычного пользователя

Средний час фрод-транзакции отличается от среднего часа транзакции обычного пользователя



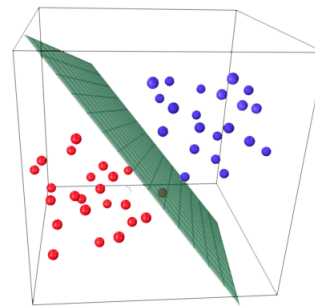
АЛГОРИТМЫ И ТЕХНИКИ

– Логистическая регрессия

`sklearn.linear_model.LogisticRegression`

– Случайный лес

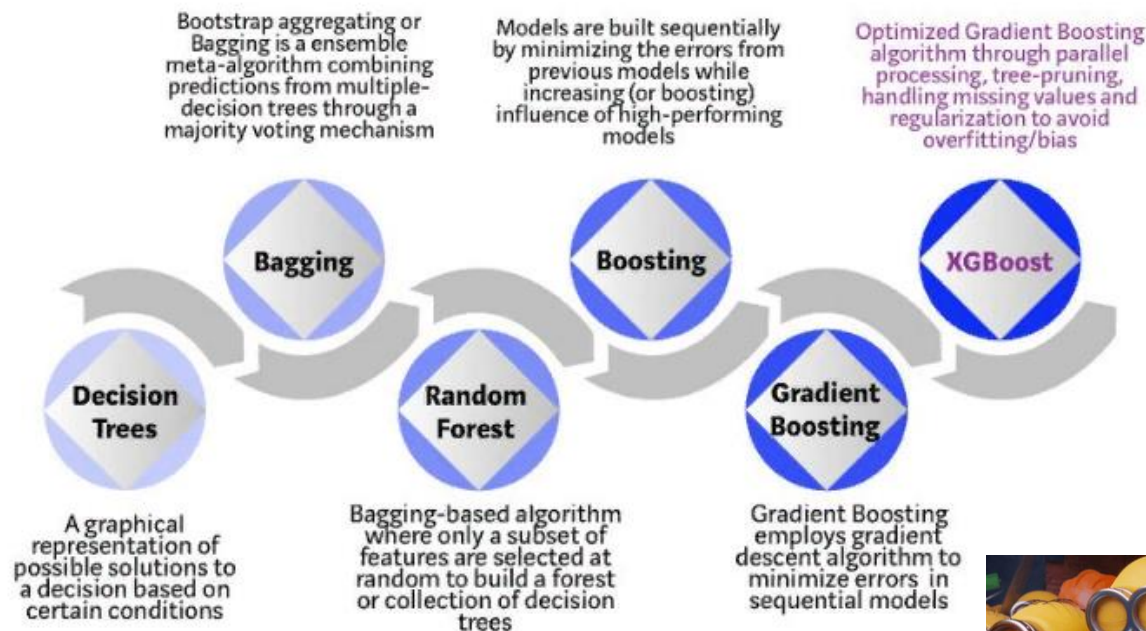
`sklearn.ensemble.RandomForestClassifier`



АЛГОРИТМЫ И ТЕХНИКИ

XGBoost

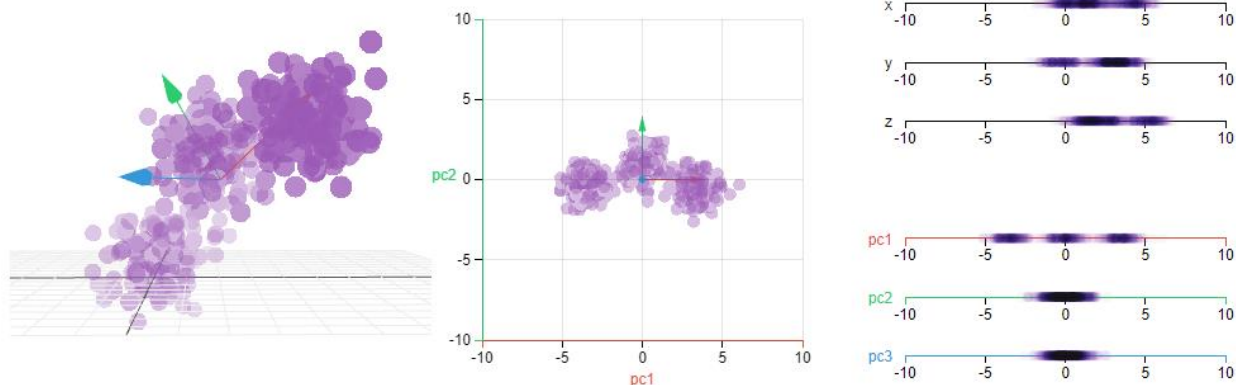
xgboost.XGBClassifier



АЛГОРИТМЫ И ТЕХНИКИ

Метод снижения размерности с помощью поиска
главных компонент

`sklearn.decomposition.PCA`



3. МЕТОДИКА РЕШЕНИЯ

РЕАЛИЗАЦИЯ

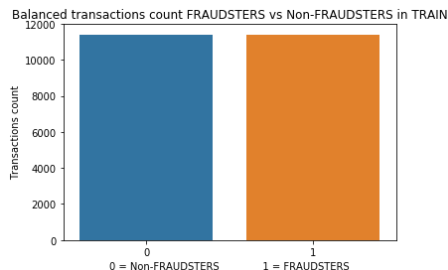
1. Будем **обучаться на 70% транзакций до даты 2018-05-24**, на остальных 30% будем тестировать нашу модель
2. **Случайным образом отбираем легитимные транзакции** в обучающую выборку в кол-ве мошеннических транзакций
3. Строим **baseline модель – LogisticRegression**
4. Тестируем другие модели
5. Тестируем лучшую модель с понижением размерности
6. **Создаем функцию определения «действия» и API** к ней

- В **тренировочном** датасете уникальных пользователей - **267 мошенников / 5873 легитимных**
- В **тестовом** датасете уникальных пользователей - **95 мошенников / 5153 легитимных**



ПРЕДОБРАБОТКА

- Сбалансированные классы необходимы для корректной оценки качества модели



	Логистическая регрессия
Несбалансированная выборка	ROC AUC = 0.66
Сбалансированная выборка	ROC AUC = 0.87

- Выбросы удалить нельзя, т.к. в них более 26% фрод-транзакций

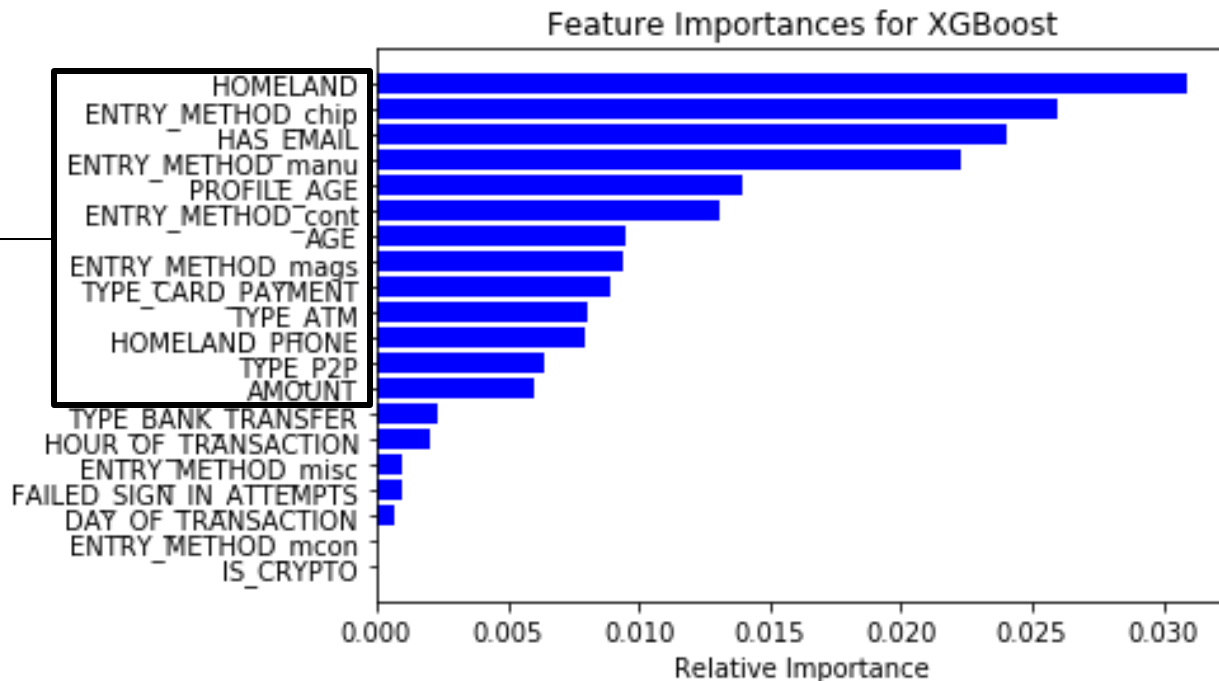
ИТОГОВАЯ МОДЕЛЬ

XGBoostClassifier на основе уменьшенного количества признаков с помощью **PCA** с подобранными гиперпараметрами на основе кросс-валидации Kfold кратности 3:

- Количество признаков снижено с 196 до 47;
- Максимальная глубина деревьев: 7
- Количество деревьев: 200



ВАЖНОСТЬ ПЕРЕМЕННЫХ



ДЕЙСТВИЯ НАД ТРАНЗАКЦИЯМИ

1. Модель возвращает для каждой транзакции вероятность мошенничества
2. Вероятность можно разделить на группы действий требуемые с точки зрения бизнеса процесса

Вероятность	Действие	Вес
> 50%	ALERT	1
> 75%	LOCK and ALERT	2
> 90%	LOCK	3



По каждому пользователю выбираем
наиболее весомое действие



```
1 def check_alert(y_predicted):
2     # Return the most important (heaviest by weight) flag as a result
3     ...
4     Rules:
5     If percent is more than first level (50% for example) we need ALERT (weight=1) because it's suspicious transaction.
6     If percent is more than second level (75% for example) we need LOCK and ALERT (weight=2) because it's very suspicious transaction and it's better to lock user and send alert signal to work with this user.
7     If percent is more than max level (90% for example) we need LOCK (weight=3) because it's fraudster.
8     ...
9
10    # dictionary of alerts
11    dict_of_alerts = {0: ['PASS'],
12                      1: ['ALERT_AGENT'],
13                      2: ['LOCK_USER', 'ALERT_AGENT'],
14                      3: ['LOCK_USER']}
15
16    # for each prediction in y_prediction check the rules, get the max weight and apply dictionary to get the alert
17    return dict_of_alerts[max([
18        y >= .9: 3,
19        .75 <= y < .9: 2,
20        .5 <= y < .75: 1,
21        y < .5: 0][True for y in y_predicted])]
```



4. РЕЗУЛЬТАТЫ



ОЦЕНКА И ВАЛИДАЦИЯ

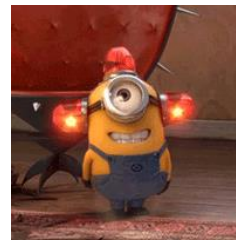
Метрика	LR	RF	XGB
Precision	0.58	0.94	0.96
Recall	0.87	0.94	0.96
ROC-AUC	0.87	0.98	0.99

Валидация функции определения действия

Проверка на «хороших» пользователях из отложенной выборки – из вероятности отнесения к классу фрода рождается необходимое над транзакцией действие:

```
1 test_user_df['PATROL_SOLUTION'].value_counts()

[PASS]                                73
[LOCK_USER]                           6
[ALERT_AGENT]                          3
[LOCK_USER, ALERT_AGENT]               2
Name: PATROL_SOLUTION, dtype: int64
```



ВНЕДРЕНИЕ

Модель анализирует каждую транзакцию



```
1 X = df[df['TRN_ID']=='961f9451-2d7d-4c62-8593-bf44d15d38b0'].drop(parameters.id_features + [parameters.target_feature],axis=1)
2 y_pred = rf.predict_proba(X[:,1])[0]
3 print(f"Probability of class IS_FRAUDSTER=1 = {y_pred}")
```

Probability of class IS_FRAUDSTER=1 = 1.0

And model predicted this transaction is by fraudster

И определяет необходимое действие



над пользователем по его транзакциям

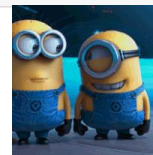
```
1 def patrol(user_id, rf=None):
2     if rf is None:
3         # Load our pretrained model
4         from sklearn.externals import joblib
5         #rf = joblib.Load(parameters.model_pkl)
6         rf = pickle.load(open(parameters.model_pkl,'rb'))
7     # Load data for user_id
8     X = get_user_data(user_id = user_id, asserting=False)
9     # get prediction
10    if len(X) > 0:
11        # if we have transactions for this user => act on them
12        y_pred = rf.predict_proba(X[:,1])
13        return check_alert(y_pred)
14    # if we have no transactions for this user => pass him
15    return ['PASS']
```

```
1 patrol(user_id='fb23710b-609a-49bf-8a9a-be49c59ce6de')
['LOCK_USER']
```

```
1 # Load our pretrained model
2 #model = joblib.Load(parameters.model_pkl)
3 model = pickle.load(open(parameters.model_pkl,'rb'))
4
5 # check FRAUDSTER-users from users' dataset
6 d = {u: patrol(user_id=u, rf=model) for u in user_df[user_df['IS_FRAUDSTER']==True]['ID'].values}
7 print('Patrol-function actions on FRAUDSTERS:')
8 pd.DataFrame(list(d.items()), columns=['USER_ID','ACTION'])['ACTION'].value_counts()
```

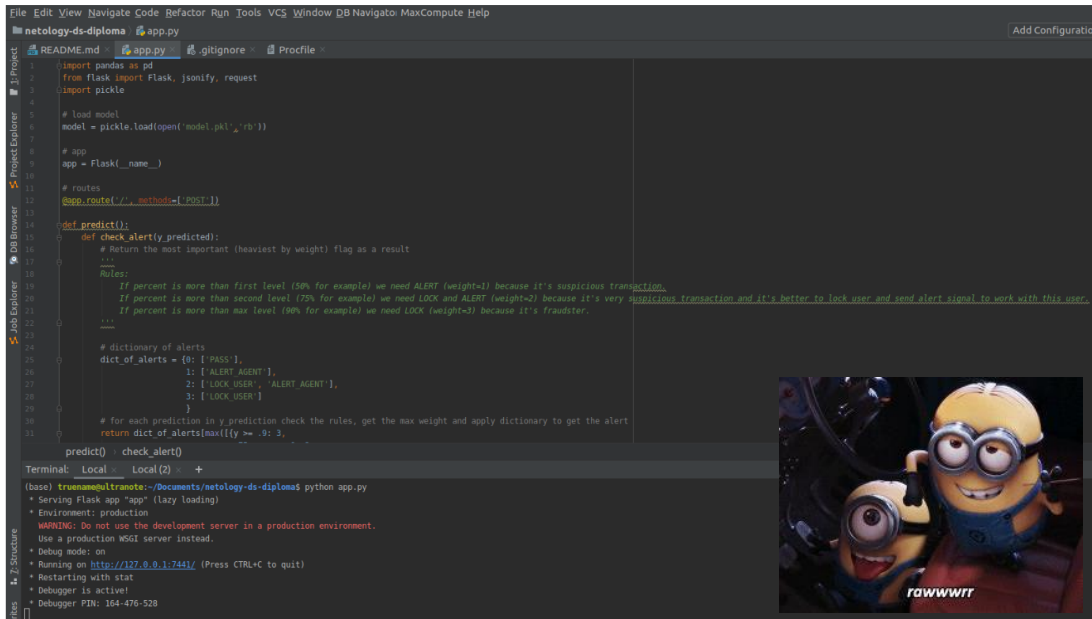
Patrol-function actions on FRAUDSTERS:

```
[LOCK_USER]      294
[PASS]            2
[LOCK_USER, ALERT_AGENT]  1
[ALERT_AGENT]     1
Name: ACTION, dtype: int64
```



ВНЕДРЕНИЕ

- Создан API с помощью фреймворка Flask для эмуляции работы с моделью в production-системе
- В API подается JSON с предикторами транзакции, в результате получаем необходимое «действие» над транзакцией



```
File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigato MaxCompute Help
netology-ds-diploma app.py
Project Explorer
Project README.md app.py .gitignore Procfile
DB Browser
JOS Explorer
1 import pandas as pd
2 from flask import Flask, jsonify, request
3 import pickle
4
5 # Load model
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 # app
9 app = Flask(__name__)
10
11 # routes
12 @app.route('/', methods=['POST'])
13
14 def predict():
15     # check alert(y predicted):
16     # Return the most important (heaviest by weight) flag as a result
17     ...
18     Rules:
19     If percent is more than first level (50% for example) we need ALERT (weight=1) because it's suspicious transaction.
20     If percent is more than second level (75% for example) we need LOCK and ALERT (weight=2) because it's very suspicious transaction and it's better to lock user and send alert signal to work with this user.
21     If percent is more than max level (90% for example) we need LOCK (weight=3) because it's fraudster.
22     ...
23     # dictionary of alerts
24     dict of alerts = {0: ['PASS'],
25                      1: ['ALERT_USER'],
26                      2: ['LOCK_USER', 'ALERT_USER'],
27                      3: ['LOCK_USER']}
28
29     # for each prediction in y prediction check the rules, get the max weight and apply dictionary to get the alert
30     return dict of alerts[max(i.y >= .5: 3,
31
32 predict() check_alert()
Terminal: Local Local (2) +
(base) truenam@ultranote:~/Documents/netology-ds-diploma$ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:7441/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 164-476-528
```



Поиск мошеннических операций

[ВЛАДИМИР НИКИФОРОВ]

ВНЕДРЕНИЕ

Варианты вызова API: из JupyterNotebook или через обычный cURL:

```
test_json = ml_predict_catching_fraud_data('961f9451-2d7d-4c62-8593-bf44d15d38b0')
test_json
```

```
'[{"pc0":-132.4537088991,"pc1":23.9069752413,"pc2":-4.3130240921,"pc3":-4.3929295845,"pc4":-1.8210378034,"pc5":1.3298811371,"pc6":0.191096318,"pc7":-0.6116878408,"pc8":-0.1968860537,"pc9":0.7151061651,"pc10":-0.3549136297,"pc11":-0.1460732538,"pc12":0.2854419859,"pc13":-0.0351717793,"pc14":0.0621614226,"pc15":0.1294242791,"pc16":-0.2472887561,"pc17":-0.1678900575,"pc18":-0.320031598,"pc19":0.0609354221,"pc20":-0.3986248723,"pc21":-0.0779848935,"pc22":0.0600801355,"pc23":-0.1383980383,"pc24":-0.4205490863,"pc25":-0.2671511684,"pc26":0.0106348906,"pc27":-0.0071147181,"pc28":0.2045036641,"pc29":0.0209013339,"pc30":0.0307195189,"pc31":0.0152274207,"pc32":0.033238404,"pc33":0.0239170012,"pc34":0.0277473495,"pc35":0.0052902787,"pc36":0.0165091078}]'
```

Call API from the python

```
ml_predict_catching_fraud_api(p_json_data=test_json)
```

```
{'results': {'ACTION': ['LOCK_USER']}}
```

Call API thru cURL

```
curl -s -XPOST 'http://127.0.0.1:7441/' -H 'accept-content: application/json' -d '{"pc0":-132.4537088991,"pc1":23.9069752413,"pc2":-4.3130240921,"pc3":-4.3929295845,"pc4":-1.8210378034,"pc5":1.3298811371,"pc6":0.191096318,"pc7":-0.6116878408,"pc8":-0.1968860537,"pc9":0.7151061651,"pc10":-0.3549136297,"pc11":-0.1460732538,"pc12":0.2854419859,"pc13":-0.0351717793,"pc14":0.0621614226,"pc15":0.1294242791,"pc16":-0.2472887561,"pc17":-0.1678900575,"pc18":-0.320031598,"pc19":0.0609354221,"pc20":-0.3986248723,"pc21":-0.0779848935,"pc22":0.0600801355,"pc23":-0.1383980383,"pc24":-0.4205490863,"pc25":-0.2671511684,"pc26":0.0106348906,"pc27":-0.0071147181,"pc28":0.2045036641,"pc29":0.0209013339,"pc30":0.0307195189,"pc31":0.0152274207,"pc32":0.033238404,"pc33":0.0239170012,"pc34":0.0277473495,"pc35":0.0052902787,"pc36":0.0165091078}'
```



5. ЗАКЛЮЧЕНИЕ

ВЫВОДЫ

1. Лучшая модель – **XGBoost**
2. Самые важные признаки:
 - HOMELAND – признак транзакции из страны регистрации профиля
 - ENTRY_METHOD – способ оплаты
 - HAS_EMAIL – указан ли email в профиле
 - PROFILE_AGE - Возраст профиля
 - AGE - возраст клиента
3. Модель классифицирует транзакции в условиях, приближенных к реальным, и готова к замерам производительности и расчетам необходимой производительности виртуальной машины для ее хостинга

КУДА ДАЛЬШЕ

- ✓ Требуется обогащение клиентского профиля для повышения качества модели
- ✓ Замер скорости обработки транзакций пользователей с учетом расчёта дополнительных параметров транзакций
- ✓ Если скорость удовлетворяет разрешенным пределам, то интеграция в production-систему



Спасибо за внимание!
[ВЛАДИМИР НИКИФОРОВ]

