One pager

Intent

In the last few months, FunPay registered a significant increase in the number of fraudulent credit card transactions. All these frauds were reported by the users as the company does not have any fraud detection system in place. In most cases, the company had to refund the money to the users. In the other cases, the users lost their money.

The company is looking for a solution to detect and prevent these fraudulent transactions. This will benefit both the company and its users financially and is expected to improve users' trust in the company.

Desired outcome

Being able to detect and prevent fraudulent credit card transactions is expected to increase the company's revenue in multiple ways. First, the company will not have to refund the money to the users – the cost of refunds to the users should decrease by at least 60%. Second, the company will be more credible to the users, and the number of new users, as well as the number of daily active users, should increase.

Essential metrics of the system are precision and recall. A balance needs to be found between those two. Allowing a fraudulent transaction is more costly than preventing it and having the user review it. Therefore, the system should be optimized for recall, which should be at least 90%, i.e. missing at most one fraudulent transaction out of ten. The precision should be at least 95%.

The fraud detection will happen in real-time. It is crucial not to noticeably slow down the payment process for the user. Therefore, the latency of the service should be below 800 ms.

Deliverable

The deliverable is a model that classifies whether a credit card transaction is fraudulent or not. It will be deployed as a web service in FunPay's private cloud, and it will be accessible only from the other company's systems. The existing payment processing system will be modified to call the fraud detection service before completing the credit card transaction.

The model will be trained on the historical data of credit card transactions. The data will be anonymized before training using PCA.

The model will be trained using the Pycaret library. It will be deployed using FastAPI and Docker.

Constraints

FunPay does not have a dedicated data science team. Therefore, the solution should be easy to develop and operate.

Security and data privacy are one of the fundamental values of FunPay. The system must be accessible only by other FunPay systems and must not use personally identifiable information.

Own solution is preferred to third-party solutions due to the ability to customize the solution later specifically for the company's needs.

Design document

Table of contents

1. Overview

FunPay is considering a new internal service that would help prevent credit card frauds. This document proposes a design of such a service in order to get feedback on the design.

2. Motivation

In the last few months, FunPay registered a significant increase in the number of fraudulent credit card transactions. All these frauds were reported by the users as the company does not have any fraud detection system in place. In most cases, the company had to refund the money to the users. In the other cases, the users lost their money. The last group of frauds are those caught neither by the company nor by the users. These are estimated to be around 10% of all fraudulent transactions.

The company is looking for a solution to detect and prevent these fraudulent transactions. This will benefit both the company and its users financially, and it will improve users' trust in the company.

Table 1: Shows the number of transactions and the number of frauds in the last three months. The last column shows the increase in the number of frauds compared to the previous month.

Month	Number of transactions	Number frauds	Percent of detected frauds	Increase in frauds
July	4,326,856	6,101	0.141%	–
August	4,386,752	6,887	0.157%	11.3%
September	4,272,105	7,305	0.171%	8.9%

Table 2: Shows who reported the frauds and the loss for the company and users caused by the frauds.

Month	User-reported frauds	Automatically detected frauds	Estimated undetected frauds	Number of frauds refunded by FunPay	Percent refunded by FunPay	Avg. transaction amount	FunPay loss	User loss
July	6,101	0	678	4,393	72%	89.7 EUR	394,052 EUR	214,024 EUR
Aug.	6,887	0	765	5,096	74%	87.2 EUR	444,371 EUR	222,883 EUR
Sept.	7,305	0	812	5,187	71%	88.3 EUR	458,012 EUR	258,719 EUR

3. Success metrics

According to the survey recently conducted by FunPay among existing users, credit card frauds were identified as a major concern by 60% of users. 20% of users had experienced credit card fraud while using FunPay credit cards. It is expected that the addition of automatic fraud detection will result in more user trust, thus resulting in increased FunPay credit card usage.

- Number of daily active users increases by at least 8%.
- Average number of credit card transactions per user per day increases by at least 10%.

Marketing this feature should also attract new users.

- Number of new users per month increases by at least 5%.

Most importantly, credit card frauds should be caught automatically, not relying on users reporting them after it is too late.

- At least 70% of credit card frauds should be caught automatically. I.e., the recall should be at least 70%.
- Number of user-reported frauds decreases by at least 80%.
- Cost of refunds to the users decreases by at least 60%.
- Monetary loss of the users decreases by at least 60%.

It is important that the service does not consider valid transactions as fraudulent. The number of false positives should be kept low.

- The precision should be at least 95%.

Because the fraud detection will happen in real-time, the service should not noticeably slow down the payment process for the user.

- The latency of the service should be less than 800 ms.

4. Requirements & constraints

4.1 Functional requirements

- Credit card transactions that are found to be fraudulent are declined.
- Credit card transactions that are found to be valid are processed as usual.
- The recall is at least 75%.
- The precision is at least 95%.

4.2 Non-functional requirements

- Performance
  - The throughput of the service is at least 100 requests per second.
  - The service handles peaks of up to 1000 requests per second.
  - The P99 latency of the service is less than 800 ms.
  - The error rate of the service is less than 0.1%.
- Security & data privacy
  - The service is accessible only from the other company's systems.
  - The service does not store any data about the users or their transactions.
  - The service processes only numerical input variables, which are the result of a PCA transformation, and the transaction amount.
- Costs
  - The cost of the service is less than 0.01 EUR per transaction.

4.3 What is in-scope & out-of-scope?

The underlying model will be trained on a fixed set of data and a fixed number of features. When new data and/or features are obtained, the model will be redeployed manually. This process will not be automated. There are two main reasons for this decision:

1. The service is in its early stages, and it is necessary to get user feedback first before allocating more resources to the development of a more complicated solution.
2. Each new model needs to be thoroughly evaluated before being put into production.

In the future, the model could be re-trained continuously as new data is obtained. The new data will consist of:

- Transactions reported by users as valid that were considered fraudulent by the system.
- Transactions reported by users as fraudulent that were considered valid by the system.
- Transactions that were considered valid by the system and were not reported by users as fraudulent.
- Transactions that were considered fraudulent by the system and were not reported by users as valid.

Another feature that is out of scope is a predefined list of merchants that are considered fraudulent. In the first stage, the service will work only with credit card transaction data, without external data sources.

Lastly, the service threshold on when to consider a transaction as fraudulent will be fixed. In the future, it could depend on the transaction amount, i.e., high amount transactions could be considered fraudulent more easily. This is called *Example-Dependent Cost-Sensitive Classification*.

4.4 What are our assumptions?

With around 57,000 users making 5 transactions per day on average, the average system load would be about 8 requests per second. It is also assumed that the occasional peaks are at most 100 requests per second.

5. Methodology

5.1. Problem statement

FunPay has a dataset of previous transactions that contains labels on whether a given transaction was fraudulent or not. Therefore, a supervised approach will be used. The fraud detection will be framed as a binary classification task with the classes being fraudulent and non-fraudulent transaction.

Having high recall is more important than having high precision. FunPay wants to maximize caught fraud at the cost of more false alarms.

5.2. Data

The dataset that will be used to train and evaluate the model is a collection of labeled transactions that FunPay processed in a span of two days in September 2023. It contains 284,807 transactions in total, 492 of which are frauds (i.e., only 0.173%). This makes it a highly imbalanced dataset.

To comply with user data privacy, the original features have been transformed using PCA into 28 numerical floating point features, named V1,..., V28. Those are the principal components obtained with PCA.

The data also contain untransformed transaction amount (in EUR; floating point number) and the label (1 meaning fraudulent, 0 meaning valid). These are named "Amount" and "Class", respectively. The minimal amount is 0 (transactions that validate if a credit card is valid), and the maximum is 25,691.16 EUR.

5.3. Techniques

The data science team has received the data already pre-processed from the payment processing team. The dataset does not have any missing values. Therefore, no imputation techniques need to be used. The dataset may contain some outliers. These will be removed using the Isolation Forest method. The dataset contains 29 features and the target variable. No further features need to be created.

Two baseline models should be included for comparison with more complicated models – one that classifies each transaction as non-fraudulent and the second one that uses Naive Bayes. Preliminary analysis has been conducted,

and machine learning models such as Random Forest Classifier and Logistic Regression give good results. These models should be explored further. Lastly, it would be good to try solving the problem using neural networks.

5.4. Experimentation & validation

5.4.1 Before deployment

The data will be split into two parts. 70% will be used for training, and 30% will be used for the final validation of the chosen model. To compare various models and their hyperparameters, the training data will be further split using stratified k-fold with 10 folds. This will result in multiple average cross-validated scores. The most relevant metrics are recall and precision. Because of the high-class imbalance, accuracy is not a good metric. The area under the precision-recall curve (AUPRC) will be used instead.

5.4.2 After deployment

An A/B test will be conducted. The treatment and control groups will be customer-based and will be assigned randomly. 50% of users will be assigned to the treatment group and 50% to the control group. The treatment group will have fraud detection enabled, while the control group will not. The following metrics will be monitored for both groups:

- Number of daily active users
- Average number of credit card transactions per user per day
- Number of user-reported frauds
- Cost of refunds to the users
- Monetary loss of the users

In addition to these metrics, the following metrics will be monitored for the treatment group:

- True positives – when the system detected a transaction as fraudulent and the user did not raise a complaint
- True negatives – when the system detected a transaction as valid and the user did not raise a complaint
- False positives – when the user had to manually allow a transaction that was considered fraudulent by the system
- False negatives – when the system detected a transaction as valid and the user later raised a complaint on the transaction being fraudulent

This test will run for 30 days. At the end, recall and precision will be calculated, and a decision will be made on whether the system meets the expectations defined in section 3. Besides meeting these success metrics, the company's guardrail metrics will be monitored, too. If the average transaction processing time exceeds 2000 ms, the experiment will be considered a failure.

5.5. Human-in-the-loop

Users will be able to set a maximum amount, up to which the transactions will be automatically accepted. This amount cannot be larger than 100 EUR.

When a credit card transaction is considered fraudulent, users are notified. They can then review the transaction, and if they find that it was a false positive, they can still allow it. Optionally, they can choose to consider all future transactions from the given merchant as valid. This will minimize the inconvenience for the users.

6. Implementation

6.1. High-level design

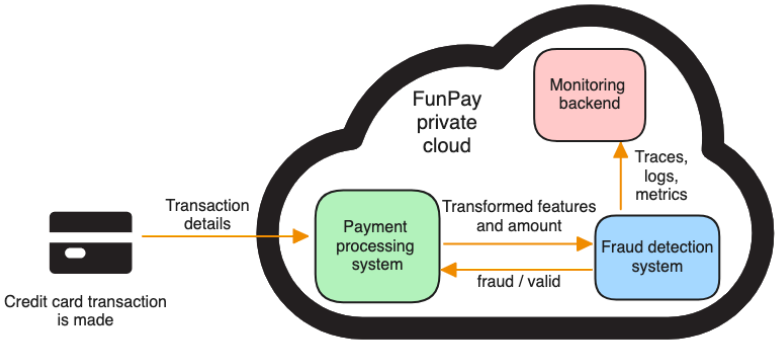

Figure 1: High-level design

The model uses data from the product team that are already pre-processed and, therefore, does not require any data preparation pipeline. All features are available. No new features need to be created. The service does not store any data.

The model will be trained and deployed manually. It is expected that this process will be automated in the future. However, this is out of scope for now.

6.2. Technological choices

For training the model, Python low-code machine learning library Pycaret will be used. FunPay does not have a dedicated data science team. Therefore, it is important to use a tool that is easy to use for so-called *citizen data scientists*. Pycaret is a good choice for this use case.

The trained model will then be deployed in a Docker container as a web service using FastAPI.

6.3. Infrastructure & scalability

FunPay operates its private cloud, where most of the company's services are deployed. Deploying the fraud detection service there will make the deployment easier, scalable and will also result in lower costs compared to using a public cloud. Furthermore, the service will be accessible only from the other company's systems, ensuring security.

The training will be done on virtual machines in this private cloud.

6.4. Performance

The cloud infrastructure will allow for easy horizontal and/or vertical scaling, should the need arise – either due to more users having fraud detection enabled or due to more complex models being deployed.

The system will be automatically horizontally scaled during request peaks.

Having the fraud detection service deployed in the same environment as the payment processing system will minimize the communication latency.

A model trained in Pycaret can be transpiler to other languages, such as C or Java. This will result in faster inference speed.

6.5. Security

The service will be accessible only from the other company's systems, which are already secured from attacks such as denial-of-service.

The service does not require authentication because all incoming requests have been previously authenticated by the payment processing system.

Safety will be used to analyze dependencies for security vulnerabilities.

Bandit will be used to analyze the code for security issues.

6.6. Data privacy

FunPay is a European company. As such, it is subject to the General Data Protection Regulation (GDPR). However, the fraud detection service will not store any data about the users or their transactions. The service will be trained on and later process only numerical input variables, which are the result of a PCA transformation, and the transaction amount.

6.7. Monitoring & alarms

The service will be instrumented using OpenTelemetry. The OpenTelemetry data will be sent to a centralized Prometheus backend and displayed using Grafana. Alerting rules will be set up in Prometheus to notify the on-call engineer when a metric breaches a threshold.

The metrics that will be monitored and their thresholds for alerts are:

- Server-side latency – > 800 ms
- Throughput – < 100 requests per second
- Error rate – > 0.1%

6.8. Cost

Two engineers will be allocated to this project. The cost of their work is 10,000 EUR per month per person. The initial deployment of this service is expected to take 2 months. The operational cost of the infrastructure where the model will be trained is 1,000 EUR per month. In total, the upfront costs are 42,000 EUR.

After the initial deployment, the same two engineers will continue to work on the service, applying insights from A/B testing and improving the model. The operational cost of the infrastructure where the model will be deployed is 1,000 EUR per month. Because the company's SRE team will be responsible for the maintenance of the deployed application, there are few additional labor costs associated with the operation of the service.

Assuming 4,000,000 transactions per month, the cost of the service is 0.005 EUR per transaction.

6.9. Integration points

The payment processing system will send a request to the fraud detection service when a credit card transaction is being processed. The request will contain the transaction amount and the principal components obtained with PCA. The service will respond with a boolean value indicating whether the transaction is considered fraudulent or not.

Sample request:

```
{
  "Amount": 149.62,
  "V1": -1.36,
  "V2": -0.07,
  ...,
  "V28": 2.54
}
```

Sample response:

```
{
  "fraudulent": false
}
```

6.10. Risks & uncertainties

The collected dataset might not be representative enough and the model might not generalize well to new data.

The model might not be able to detect new types of fraud that were not present in the training data.

Some information might be lost during the PCA transformation.

7. Appendix

7.1. Alternatives

Google Vertex AI

Google Vertex AI platform has been considered as an alternative. It can be used both for model training and model deployment.

Training the model requires uploading the data to Google Cloud. The Google Cloud is well-secured, and the data are anonymized through PCA. Therefore, from the security and data privacy point of view, this does not pose any risks. The model can then be trained using either *AutoML* or newer *AutoML on Pipelines*. At the time of writing, using *AutoML on Pipelines* resulted in multiple issues and the training could not be completed. Training using *AutoML* was successful.

The model can be deployed as a web service – an *endpoint* in Google's terminology. This endpoint can be configured to be private only, which is an important requirement of the system. However, testing of the endpoint showed that the inference speed was very slow and would not satisfy the latency requirement of 800 ms.

Lastly, the usage of Google Vertex AI for model training and model deployment resulted in high costs. Leveraging the existing infrastructure of FunPay to train and deploy the models will reduce the costs significantly.

7.2. Experiment Results

Experiments show that precision of 95% and recall of 79% are achievable using a Random Forest Classifier. The respective AUPRC was 0.85.

7.3. Milestones & timeline

- January 2024: Start of the project
- March 2024: Service is deployed as an A/B test
- April 2024: A/B test is finished, the service is evaluated and potentially deployed to all users

7.4. References

The template for this design document is based on:

- Yan, Ziyou. (Feb 2021). How to Write Better with The Why, What, How Framework. eugeneyan.com. https://eugeneyan.com/writing/writing-docs-why-what-how/.
- Yan, Ziyou. (Mar 2021). How to Write Design Docs for Machine Learning Systems. eugeneyan.com. https://eugeneyan.com/writing/ml-design-docs/.
- Yan, Ziyou. (Mar 2023). ml-design-docs https://github.com/eugeneyan/ml-design-docs.

The content of the document is inspired by the *Credit Card Fraud Detection* dataset by Machine Learning Group of ULB (Université Libre de Bruxelles) retrieved from https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data in October 2023.

The high-level design diagram was created using these icons: