Welcome!

Hello and thank you for your application to our Machine learning engineer position!

The next step in our recruitment process is this technical challenge. Not only is this a way to make your machine learning skills shine, but it is also an opportunity to assess whether our daily work appeals to you.

About our team

Our role within the company is to ship products that incorporate Machine Learning so as to automate, streamline and sometimes even outperform manual operations needed throughout the company, from toxicity screening, to gamer support or transactional fraud detection.

We are a team of builders at heart: our motto is to be pragmatic and always go all the way to production. We strive to read the latest research papers, adapt them to our specific problems, build prototypes, and finally craft fully packaged products off of them.

Use case

Ubisoft faces a great growth on its e-commerce platform, selling main games as well as additional contents once the player entered the game. But such a growth triggered the rise of scams and credit card fraud.

To avoid the revenue loss, our team has built a real time scoring platform based on machine.

However, training machine learning models in the context of fraud detection can be a little tricky. Especially since you rarely have access to a fully labeled data set. The labels are usually obtained through the bank that returns a "fraud flag" once the stolen credit card has been identified. But the fraud management tool currently in place already blocks transactions and prevents you from getting the final label of a given order.

In this challenge, we offer you to try a semi-supervised learning approach to circumvent that problem.

Problem statement

You will find attached a dataset containing transactions made in January 2019 on Ubisoft in-game selling platform. This data has been anonymized. It contains both numerical and categorical values that describe a specific transaction per row.

Please find below a brief description of each column:

Column name	Description
order_id	Unique transaction identifier
user_id	Unique customer identifier. Please be aware that one customer can have multiple transactions in our dataset.
order_created_datetime	GMT timestamp at which the transaction was passed.
amount	Amount of the transaction (in €).

total_amount_14days	Amount spent in the last 14 days by the customer.
email_handle_length	Number of characters in the handle of the customer's email.
email_handle_dst_char	Number of distinct characters in the handle of the customer's email.
total_nb_orders_player	Total number of orders the customer made in the past.
player_seniority	Number of days since the creation of the player's account.
total_nb_play_sessions	Number of game sessions played by the customer.
geographic_distance_risk	Whether the difference between country of playing and the country of buying is risky.
transaction_status	Final status of the transaction. "LEGIT" means the transaction is good, "FRAUD" means it's fraudulent. The last status "BLOCKED" means that the existing fraud management tool has stopped the transaction and we do not have a final label for it.

1) Based on the pseudo-code of the following paper: http://homepages.rpi.edu/~bennek/kdd-KristinBennett1.pdf, implement the ASSEMBLE.AdaBoost model. You will consider that $\alpha=1$ and $\beta=0.9$. You can of course rely on sklearn for the weak learners.

2) Build a training pipeline with:

- training of your implemented algorithm
- observe the resulting confusion matrix on your test set

3) The financial output of a transaction depends on whether it was a fraud or not, and whether the transaction was blocked or not. Assume that the "gain" of the transaction, for Ubisoft, is summarized by the following matrix.

	Order is blocked	**Order is not blocked**
Order is legit	0	M
Order is fraud	0	-F

With:

- M > 0, is the price paid by the customer (amount variable).
- F > 0 is an input from the user and controls the fixed fraud fees.

Your model outputs, for each transaction, a probability p of it being a fraud and we want to maximize expected financial gain.

What would it mean to make the optimal decision (between blocking or not blocking) for a transaction? Write a method that returns the optimal decision given a fraud fee F, an amount M and a score p.

4) Create a simple API that serves your prediction through a "score" route and which will receive a json payload containing all the variables from the table. We will consider F, the fixed fraud fee, to be equal to 15€ for all transactions. The API should be wrapped in a docker container to be run on any machine.

The deliverable should contain:

- a jupyter notebook and/or python scripts of your model and training pipeline
- your code for the API