
# Hackathon

December 17th, 2023

https://giphy.com/gifs/pokemon-RRKLKJoeDX0IjTIXCj

Sirfetch'dSirfetch'd

# 1. Problem description

# Problem description

Using a sample from Farfetch's real data:

**The goal** is to forecast whether an order will lead to a genuine purchase or a fraudulent transaction based on the order's features and historical fraud patterns.

Classification problem

Binary target

# 2. Workflow

# 2.1 Data collection

# Data Collections

We used a dataset provided by Farfetch:

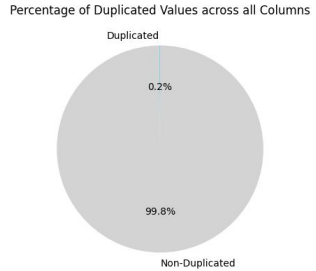
- CSV file
- a sample of a real dataset
  - 195401 transactions
  - 16 columns
- collected from August 2021 to November 2022.

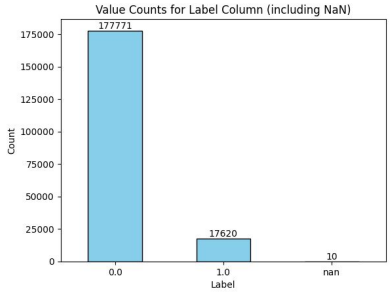
# 2.2 Data preparation

# Data preprocessing

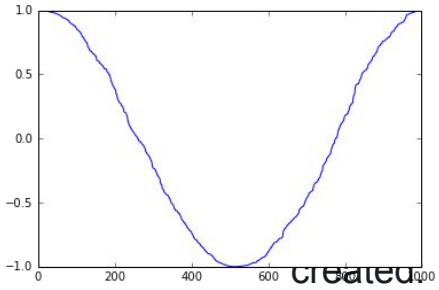
Transformations applied:

- Remove duplicates
- Remove null values in column Label
- In column ‘payment_method’, for payment_method_2, a ‘missing’ category was added in columns card_expiration_date, bin_brand,bin_type



# Data preprocessing

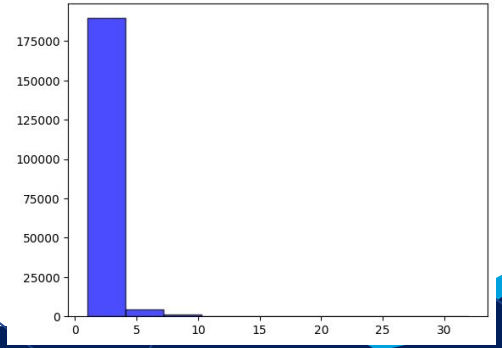



New columns were created:

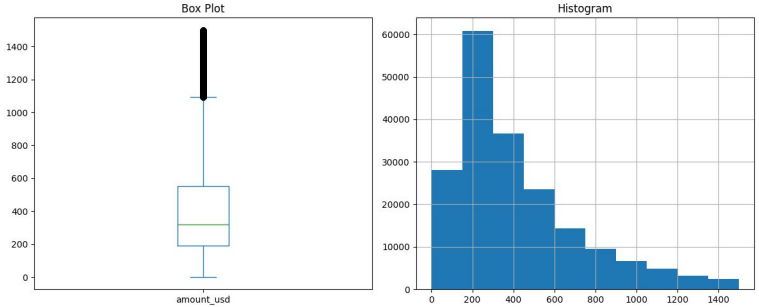
- **is_same_country**: check billing_address_country_code & shipping_address_country_code
- **avg_amount_spent_per_order_180d** : email_amount_spent_avg_180d/email_orders_count_180d
- **n_order_items** (shipping_method_type)
- **year** (from order_datetime )
- **month** (order_datetime )
- **day** (order_datetime )
- **time** (order_datetime) - cosine transformation applied
- **shipping_method_type**:

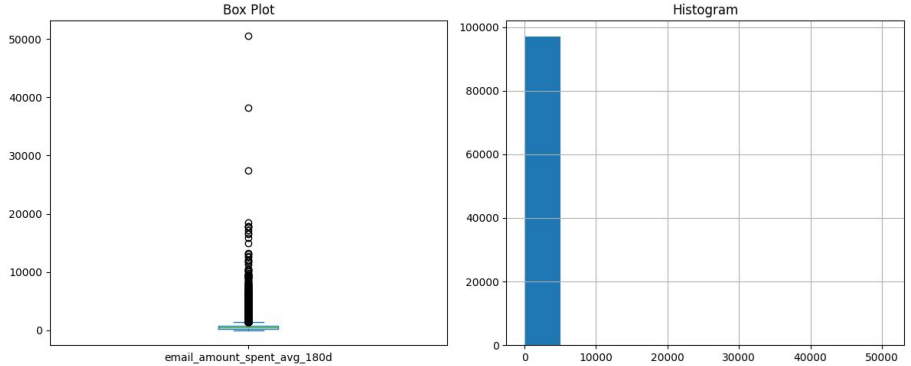
| | | | |
|---|---|---|---|
| - Count_9 | - Count_N | -9 | - N |
| - Count_C | - Count_V | -C | - S |
| - Count_D | - Count_unknown | -D | - Unkown |
| - Count_E | | -E | - V |

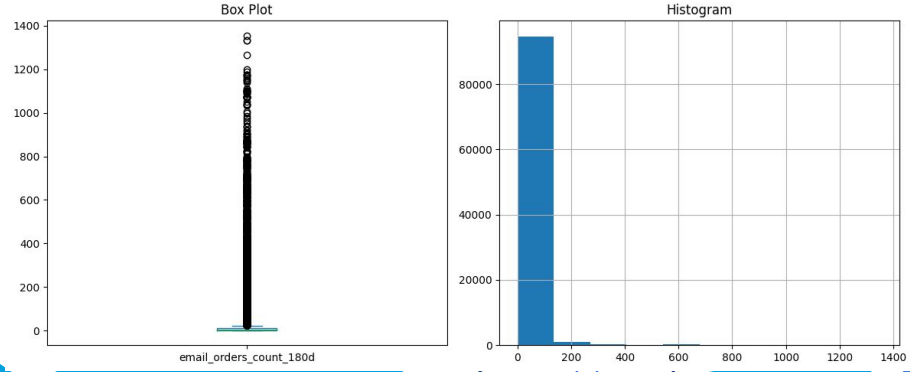
Items per Order



# Many outliers in our numerical features




Encoding for categorical features:

- **One-hot encoding**

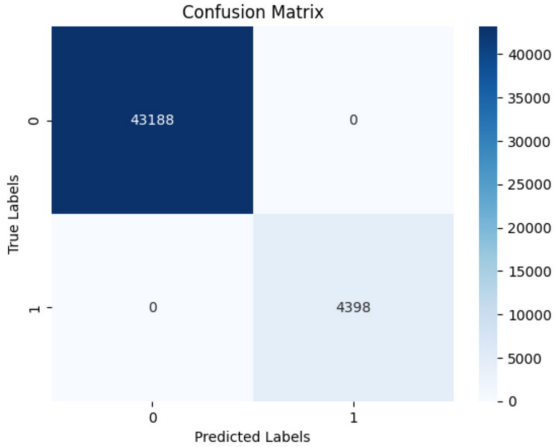
Scaling at numerical data:

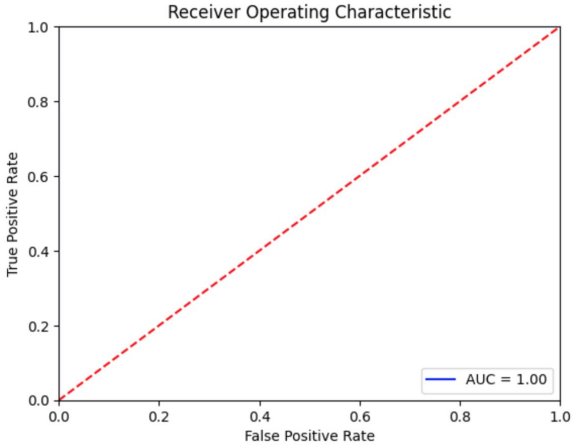
- **RobustScaler**

# 2.3 Model selection

# Model selection

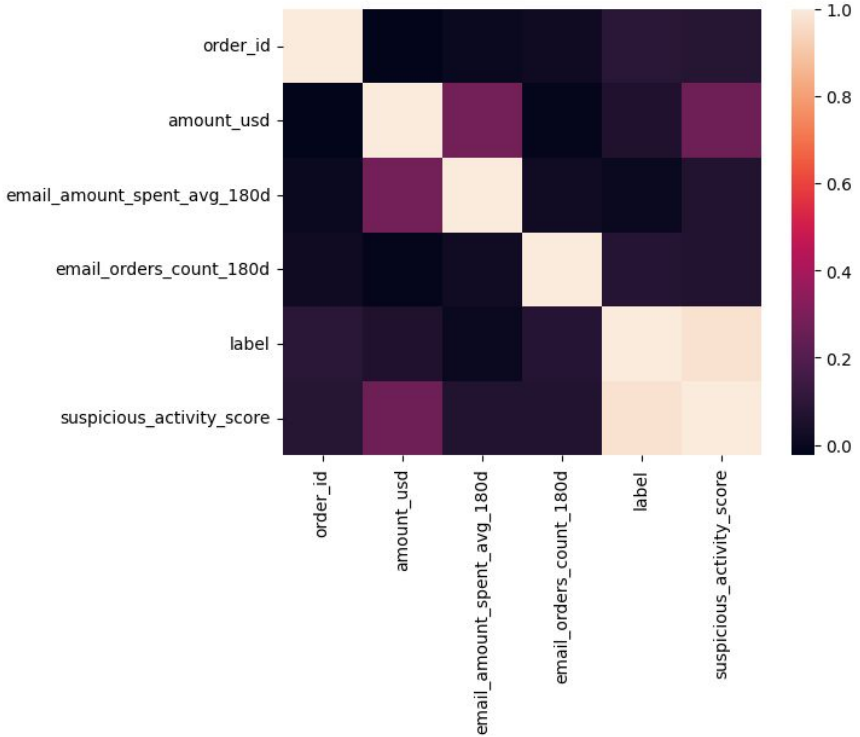
- Logistic Regression and Random Forest
- We split our data with a ratio of 70% training and 30% test



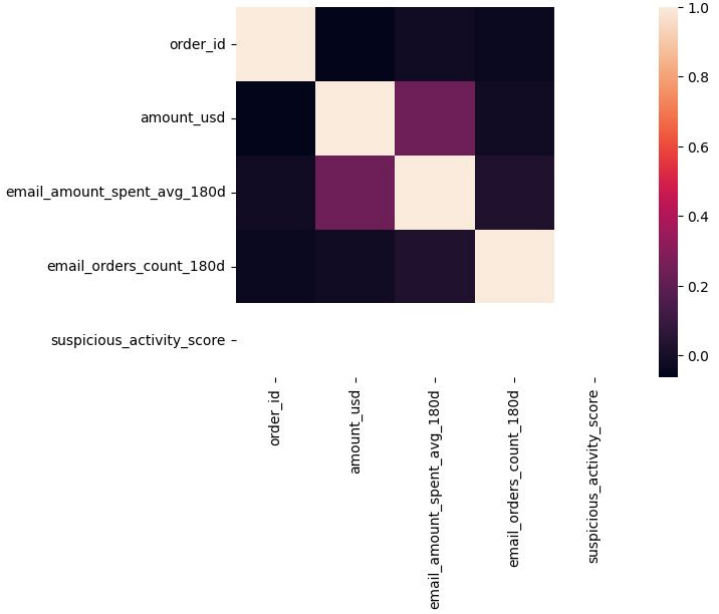




- In a perfect world, these results wouldn't arouse suspicion, but that's not the case...

# Correlation Matrixes

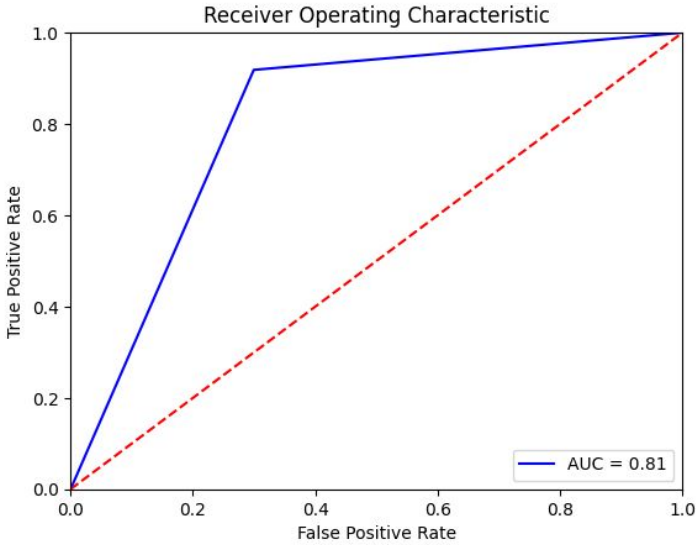


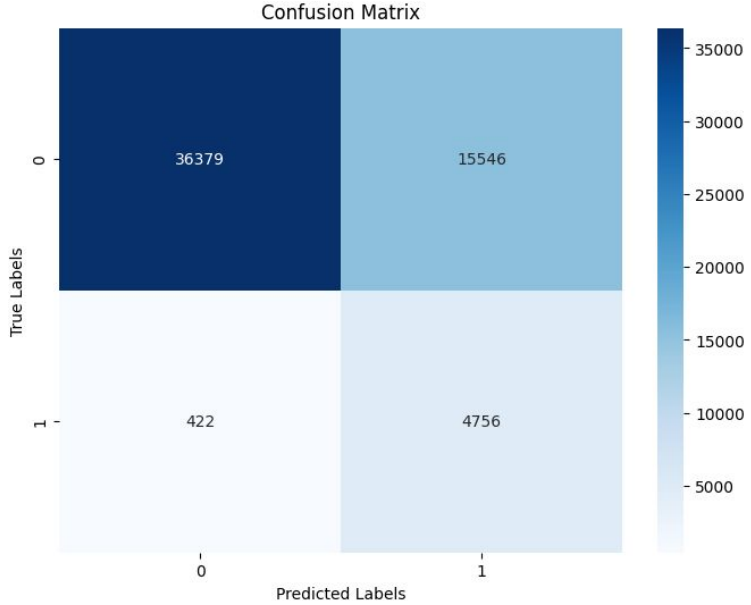

**suspicious_activity_score is data leakage!!**



# 2.4 Results and discussion

# Random Forest




**Percentage of correct predictions for label 0: 70.06%**
**Percentage of correct predictions for label 1: 91.85%**

# Results and discussion

- AUC : 81% (space for improvement)
- This is a fraud detection exercise and so capturing more false positive than false negatives is a good thing!
- It wouldn’t be the case if this was an exercise where the output sends someone to prison…

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.70 | 0.82 | 51925 |
| 1.0 | 0.23 | 0.92 | 0.37 | 5178 |
| accuracy | | | 0.72 | 57103 |
| macro avg | 0.61 | 0.81 | 0.60 | 57103 |
| weighted avg | 0.92 | 0.72 | 0.78 | 57103 |

# 3. Future Work

# Future work

- Try random search and Bayesian search for Hyperparameter tuning
- Test XGBoost models
- Get feature importance (we can use shap.Explainer) and train a model with only the features that have the most information for the prediction.

# The End!

# Pun!