Détection des fraudes bancaires

Jérôme, Vera, Isabelle, Gwendoline

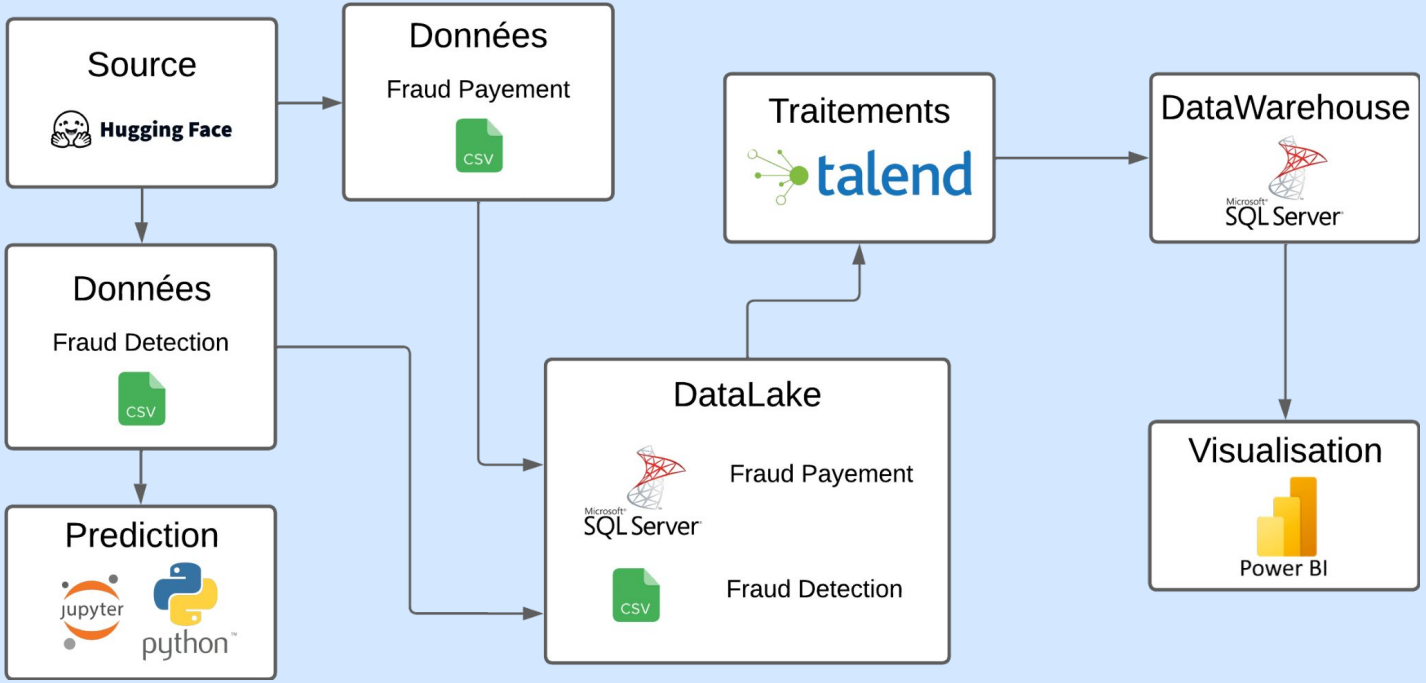
Sommaire

- Introduction
- Architecture SOA
- Collecte et traitement des données
- Visualisation
- Machine Learning
- Conclusion

Introduction

- 3 047 131 transactions frauduleuses au Q1 2023 en France
- Une étude de 2021 estimait une perte de 206 milliards de $ d’ici 2025
- La détection se repose sur des règles statiques
- Recherche de modèle de ML plus performant

Architecture SOA



Collecte et traitement des données

Fraud Payement

Nom de la variable	Type	Exemple
Time_step	datetime	3/15/2022 10:24
Transaction_Id	string	EXCHANGE-10115
Sender_Id	string	JPMC-CLIENT-10098
Sender_Account	string	ACCOUNT-10108
Sender_Country	string (NULL possible)	USA
Sender_Sector	float	35537.0
Sender_lob	string (NULL possible)	CCB
Bene_Id	string (NULL possible)	CLIENT-10100
Bene_Account	string (NULL possible)	ACCOUNT-10106
Bene_Country	string (NULL possible)	CANADA
USD_amount	float	558.43
Label	int (0 = non fraude, 1 = fraude)	0 ou 1
Transaction_Type	string	PAY-CHECK

Fraud Detection

Nom de la variable	Type	Exemple
ssn	datetime	367-85-9826
cc_num	int	4361337605230458
first	string	Kristie
last	string	Davis
gender	string	F
city	stirng	Chandler
state	string	OK
zip	int	74834
city_pop	int	7590
job	string	Chief Strategy Officer
date_of_birth	date	1987-06-12
acct_num	int	349734538563
trans_num	string	c036244703adb9d5392f4027d9d4b38d
trans_date	date	2021-07-31
trans_time	time	02:30:01
unix_time	int	1627678801
category	string	grocery_pos
amt	float	337.54
is_fraud	int (0 = non fraude, 1 = fraude)	0 ou 1
merchant	string	fraud_Kovacek

Collecte et traitement des données

Table temporaire

```
use PROJET_FIL_ROUGE

CREATE TABLE Temp_CSV (
    ssn VARCHAR(50),
    cc_num varchar(50),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    gender VARCHAR(50),
    city VARCHAR(50),
    state VARCHAR(50),
    zip INT,
    city_pop INT,
    job VARCHAR(50),
    dob DATE,
    acct_num varchar(50),
    trans_num VARCHAR(50),
    trans_date DATE,
    trans_time TIME,
    unix_time INT,
    category VARCHAR(50),
    amt DECIMAL(10,2),
    is_fraud BIT,
    merchant VARCHAR(50)
);


BULK INSERT Temp_CSV
FROM 'D:\Download\cc_fraud_detection.csv' -- changer le chemin
WITH (
     FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '0x0A',
    TABLOCK
);
```

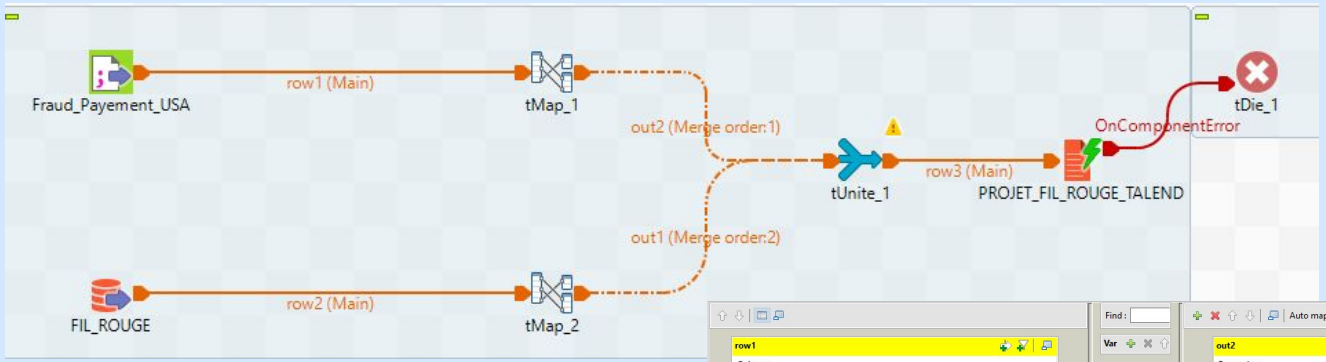
Tables utilisées

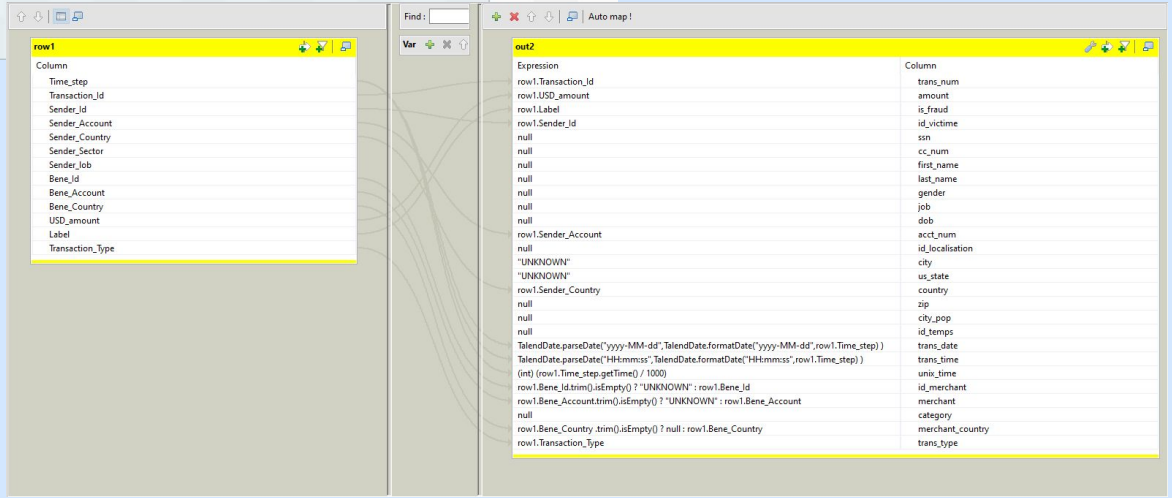
```
/*-------------------------------------------------------------
-- Table: Victime_de_la_fraude
-------------------------------------------------------------*/

CREATE TABLE Victime_de_la_fraude (
    id_victime      INT IDENTITY(1,1) NOT NULL,
    ssn             VARCHAR(50) NOT NULL,
    cc_num          VARCHAR(50) NOT NULL,
    first_name      VARCHAR(50) NOT NULL,
    last_name       VARCHAR(50) NOT NULL,
    gender          VARCHAR(50) NOT NULL,
    job             VARCHAR(50) NOT NULL,
    dob             DATE NOT NULL,
    acct_num        VARCHAR(50) NOT NULL,
    CONSTRAINT Victime_de_la_fraude_PK PRIMARY KEY (id_victime)
);

/*-------------------------------------------------------------
-- Table: Localisation
-------------------------------------------------------------*/
CREATE TABLE Localisation (
    id_localisation INT IDENTITY(1,1) NOT NULL,
    city            VARCHAR(50) NOT NULL,
    us_state        VARCHAR(50) NOT NULL,
    zip             INT NOT NULL,
    city_pop        INT NOT NULL,
    CONSTRAINT Localisation_PK PRIMARY KEY (id_localisation)
);

/*-------------------------------------------------------------
-- Table: Temps
-------------------------------------------------------------*/
CREATE TABLE Temps (
    id_temps     INT IDENTITY(1,1) NOT NULL,
    trans_date   DATE NOT NULL,
    trans_time   TIME NOT NULL,
    unix_time    INT NOT NULL,
    CONSTRAINT Temps_PK PRIMARY KEY (id_temps)
);
```

Séparation des données dans les tables correspondantes

```
INSERT INTO Victime_de_la_fraude (ssn, cc_num, first_name, last_name, gender, job, dob, acct_num)
SELECT DISTINCT ssn, cc_num, first_name, last_name, gender, job, dob, acct_num
FROM Temp_CSV;

INSERT INTO Localisation (city, us_state, zip, city_pop)
SELECT DISTINCT city, state, zip, city_pop
FROM Temp_CSV;

INSERT INTO Temps (trans_date, trans_time, unix_time)
SELECT DISTINCT trans_date, trans_time, unix_time
FROM Temp_CSV;

INSERT INTO Marchand (merchant, category)
SELECT DISTINCT merchant, category
FROM Temp_CSV;


INSERT INTO Transactions (trans_num, amount, is_fraud, id_victime, id_localisation, id_temps, id_merchant)
SELECT
    t.trans_num,
    t.amt,
    t.is_fraud,
    v.id_victime,
    l.id_localisation,
    tp.id_temps,
    m.id_merchant
FROM Temp_CSV t
JOIN Victime_de_la_fraude v ON t.ssn = v.ssn AND t.cc_num = v.cc_num
JOIN Localisation l ON t.city = l.city AND t.state = l.us_state AND t.zip = l.zip
JOIN Temps tp ON t.trans_date = tp.trans_date AND t.trans_time = tp.trans_time AND t.unix_time = tp.unix_time
JOIN Marchand m ON t.merchant = m.merchant AND t.category = m.category;
```

Collecte et traitements des données





Collecte et traitement des données

Table temporaire

```sql
create database PROJET_FIL_ROUGE_TALEND;
use PROJET_FIL_ROUGE_TALEND;

-- si probleme pour inserer avec talend en utilisant bulk
-- GRANT ADMINISTER BULK OPERATIONS TO [nom_utilisateur];

-- table temporaire
CREATE TABLE Temp_BDD (
    trans_num VARCHAR(50),
    amount DECIMAL(10,2),
    is_fraud INTEGER,
    id_victime VARCHAR(50),
    ssn VARCHAR(50),
    cc_num varchar(50),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    gender VARCHAR(50),
    job VARCHAR(50),
    dob DATE,
    acct_num varchar(50),
    id_localisation VARCHAR(50),
    city VARCHAR(50),
    us_state VARCHAR(50),
    country VARCHAR(50),
    zip INT,
    city_pop INT,
    id_temps VARCHAR(50),
    trans_date DATE,
    trans_time TIME,
    unix_time INT,
    id_merchant VARCHAR(50),
    merchant VARCHAR(50),
    category VARCHAR(50),
    merchant_country VARCHAR(50),
    trans_type VARCHAR(50),
);
```

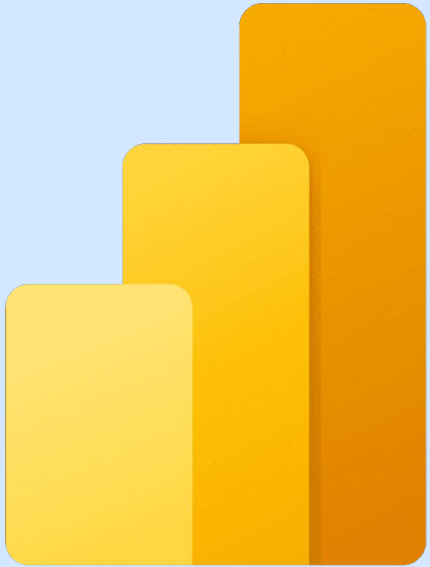
Tables utilisées

```sql
/*-------------------------------------------------------------
-- Table: Victime_de_la_fraude
-------------------------------------------------------------*/
CREATE TABLE Victimes (
    id_victim       INT IDENTITY(1,1) PRIMARY KEY,
    victime         VARCHAR(50),
    ssn             VARCHAR(50),
    cc_num          VARCHAR(50),
    first_name      VARCHAR(50),
    last_name       VARCHAR(50),
    gender          VARCHAR(50),
    job             VARCHAR(50),
    dob             DATE,
    acct_num        VARCHAR(50)
);

/*-------------------------------------------------------------
-- Table: Localisation
-------------------------------------------------------------*/
CREATE TABLE Locations (
    id_location     INT IDENTITY(1,1) PRIMARY KEY,
    country         VARCHAR(50),
    city            VARCHAR(50),
    us_state        VARCHAR(50),
    zip             INT,
    city_pop        INT
);

/*-------------------------------------------------------------
-- Table: Temps
-------------------------------------------------------------*/
CREATE TABLE Temps (
    id_time    INT IDENTITY(1,1) PRIMARY KEY,
    trans_date   DATE NOT NULL,
    trans_time   TIME NOT NULL,
    unix_time    INT
);
```

Séparation des données dans les tables correspondantes

```sql
-- Insérer les victimes
INSERT INTO Victimes (victime, ssn, cc_num, first_name, last_name, gender, job, dob, acct_num)
SELECT DISTINCT t.id_victime, t.ssn, t.cc_num, t.first_name, t.last_name, t.gender, t.job, t.dob, t.acct_num
FROM Temp_BDD t; --38437

-- Insérer les localisations
INSERT INTO Locations (country, city, us_state, zip, city_pop)
SELECT DISTINCT country, city, us_state, zip, city_pop
FROM Temp_BDD; --943

-- Insérer les informations temporelles
INSERT INTO Temps (trans_date, trans_time, unix_time)
SELECT DISTINCT trans_date, trans_time, unix_time
FROM Temp_BDD; --2939348

-- Insérer les marchands
INSERT INTO Marchands (id_marchand_null, merchant, category, merchant_country)
SELECT DISTINCT
    CASE
        WHEN id_merchant IS NULL THEN 'UNKNOWN'
        ELSE id_merchant
    END AS id_marchand,
    merchant,
    category,
    merchant_country
FROM Temp_BDD; -- 187904

select count(*) from Temp_BDD -- 3344959

select count(*) from Temp_BDD where id_merchant = 'UNKNOWN' -- 133999

-- insertion transation
WITH RankedTransactions AS (
    SELECT
        t.trans_num,
        t.amount,
        t.is_fraud,
        v.id_victim,
        l.id_location,
        tp.id_time,
        m.id_marchand,
        t.trans_type,
        ROW_NUMBER() OVER (PARTITION BY t.trans_num ORDER BY tp.id_time) AS row_num
    FROM Temp_BDD t
    JOIN Victimes v ON t.id_victime = v.victime
    JOIN Locations l ON t.city = l.city AND t.us_state = l.us_state AND t.country = l.country
    JOIN Temps tp
    ON t.trans_date = tp.trans_date
    AND t.trans_time = tp.trans_time
    AND (t.unix_time = tp.unix_time OR (t.unix_time IS NULL AND tp.unix_time IS NULL))
    LEFT JOIN Marchands m ON t.merchant = m.merchant AND t.category = m.category
)
INSERT INTO Transactions (trans_num, amount, is_fraud, id_victim, id_location, id_time, id_marchand, trans_type)
SELECT trans_num, amount, is_fraud, id_victim, id_location, id_time, id_marchand, trans_type
FROM RankedTransactions
WHERE row_num = 1; -- On garde seulement la première ligne pour chaque transaction
```

Visualisation



Power BI

Machine Learning

Conclusion

- XGBoost est le modèle le plus adapté

Difficultés rencontrées :

- Données bancaires réelles difficiles à obtenir
- Données générées ne colle pas tout à fait à la réalité
- Absence de certaines informations
- Manque d’information sur les données

Améliorations possibles :

- Nouvelles sources de données
- Nouvelles variables
- Archivages des données
- Utilisation d’index