Projet Fil Rouge

AJC Formation

L’équipe :

Vera Santos

Jérôme Casado

Isabelle Maze

Gwendoline Delestre

Introduction **4**
Architecture SOA **5**
Les données **5**
Transformations des données **8**
Visualisations **12**
1. Récupération des Données 12
2. Modélisation des Relations 12
3. Transformation des Données 12
4. Vérification des Données 12
5. Affichage du Rapport et Structure des Pages 12
6. Explication des Mesures DAX 14
Machine Learning **16**
Difficultés rencontrées **21**
Axes d'amélioration **21**
Conclusion **22**
Sources **24**

Introduction

Au premier semestre 2023, la France a enregistré 3 047 131 transactions frauduleuses par carte bancaire, représentant une part significative des 7,3 millions de fraudes recensées en Europe. (Source : Ouest France) Cette situation met en lumière l'ampleur des défis rencontrés par les institutions financières pour sécuriser les transactions électroniques.

À l'échelle mondiale, la fraude aux paiements est un phénomène en forte croissance. Selon Juniper Research, les pertes dues aux fraudes en ligne pourraient atteindre 206 milliards de dollars d'ici 2025. (Source : Stripe) Face à cette menace, les banques et les entreprises doivent adopter des solutions toujours plus performantes pour détecter et prévenir les fraudes en temps réel.

L'identification des transactions frauduleuses repose souvent sur des règles statiques qui ne permettent pas d'anticiper efficacement les nouvelles stratégies des fraudeurs. Pour améliorer la détection, il est essentiel de s'appuyer sur une analyse prédictive basée sur des données historiques.

Nous proposons de concevoir un modèle de machine learning capable de détecter les transactions suspectes en analysant les comportements des utilisateurs et en repérant des anomalies. Grâce à cette approche, il devient possible d'automatiser la détection des fraudes et de réduire les pertes financières, tout en limitant les faux positifs qui pénalisent les utilisateurs légitimes.

Architecture SOA

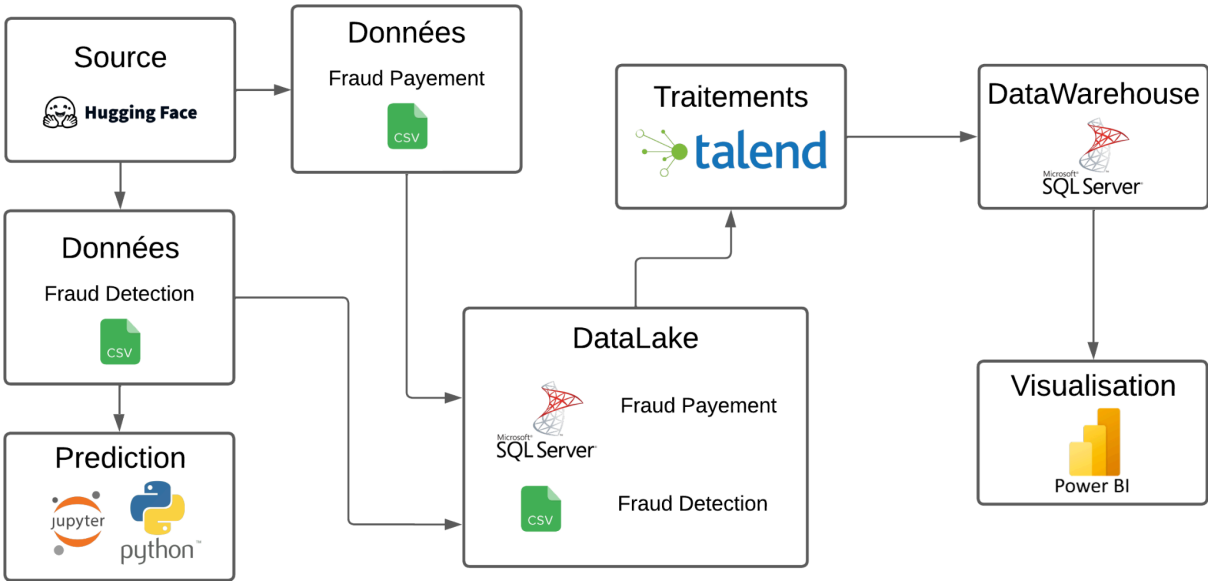


Image-Architecture SOA

Le diagramme présente l'architecture de notre projet de détection des fraudes ainsi que les technologies et les formats de fichiers que nous avons utilisés.

Les données

Nous avons recherché des données sur les transactions bancaires en ligne et avons décidé d'utiliser celles disponibles sur Hugging Face. Nous y avons trouvé deux jeux de données générés : "Fraud Payment" et "Fraud Detection", tous deux au format CSV, contenant des informations sur des transactions frauduleuses et non frauduleuses.

Pour être pertinent, le jeu de données devait inclure les informations suivantes :

- un identifiant de transaction
- le montant de la transaction
- la date et l'heure de la transaction
- des informations sur le titulaire du compte émetteur
- des informations sur la localisation de la transaction

Le jeu de données “Fraud Payement” contient à la base 1048575 lignes et les 13 colonnes suivantes :

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

Nous avons pour ce jeu de données, 1 variables quantitatives et 12 qualitatives avec au maximum, 14,39 pourcent de valeur manquantes dans les colonnes indiquées. 98% des transactions de se dataset sont légitimes et 2 % sont frauduleuses.

Le jeu de données Fraud Detection contient 2646694 lignes et les 20 colonnes suivantes :

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

Nous avons 2 variables qualitatives et 18 variables quantitatives, 99.64% de transactions légitimes et 0.35% de transactions frauduleuses. En revanche, nous n'avons pas de valeur null dans ce jeu de données

Transformations des données

Au départ, nous avions décidé de stocker nos fichiers CSV dans un simple dossier, afin de pouvoir les manipuler facilement. Cependant, au fur et à mesure de notre progression dans la formation, nous avons dû insérer un fichier CSV dans une base de données SQL.

Nous avons donc intégré le fichier Fraud Detection dans une base de données SQL Server, nommée "PROJET_FIL_ROUGE".

La première étape a consisté en la création d'une table temporaire conçue pour accueillir l'ensemble des colonnes du fichier CSV, facilitant ainsi un chargement initial des données sans modification. Par la suite, nous avons défini plusieurs tables définitives afin d'optimiser la structuration des informations en les séparant de manière logique. Enfin, à l'aide de requêtes SQL, nous avons procédé à la répartition des données issues du fichier Fraud Detection dans ces différentes tables, en veillant à établir les relations appropriées entre elles pour assurer leur cohérence et leur exploitation optimale.

Pour le fichier Fraud Payement, nous avons pris le parti de supprimer les lignes pour lesquelles le pays du payeur n'était pas les États-Unis. Cela nous a permis de diminuer le nombre de valeurs null et nous trouvions cela plus pertinent du fait que le fichier Fraud Detection ne contient que des informations sur les États-Unis.

Notre architecture de stockage repose sur deux sources principales :

- Un fichier CSV (restant disponible pour une analyse brute ou des tests).
- Une base de données SQL Server (PROJET_FIL_ROUGE) avec un schéma structuré, facilitant les requêtes et l'analyse avancée des données.

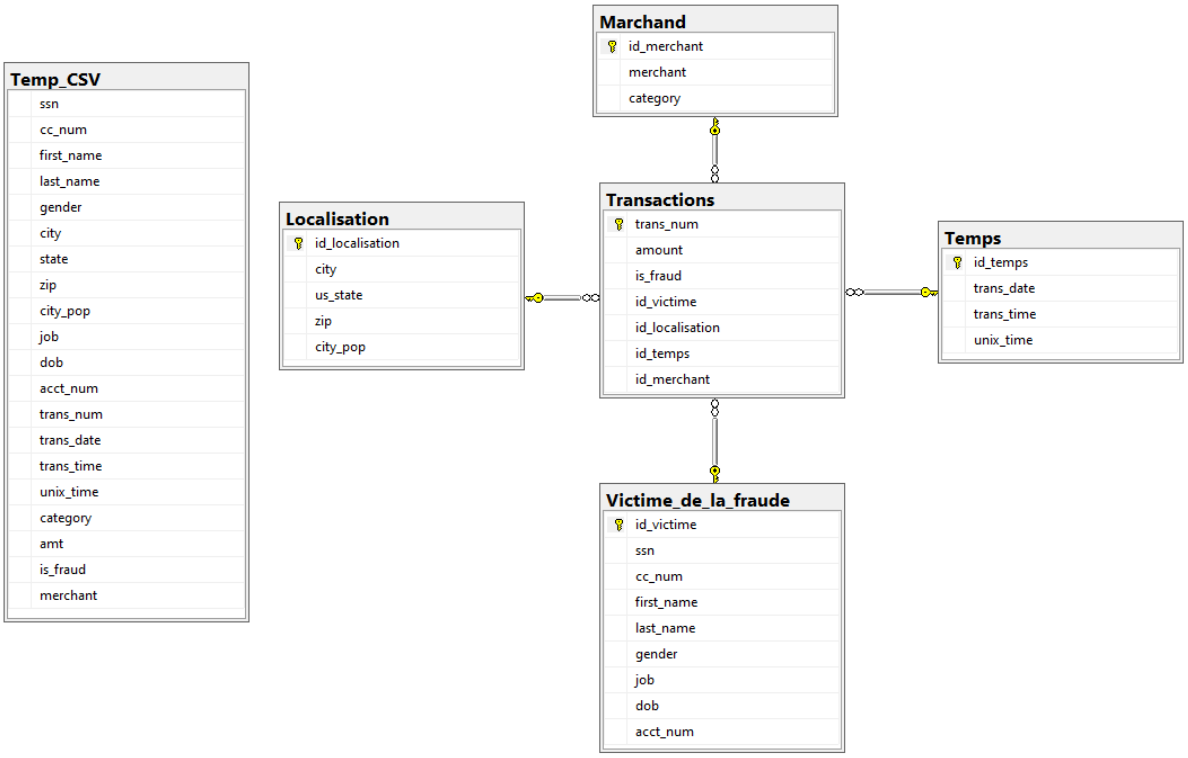
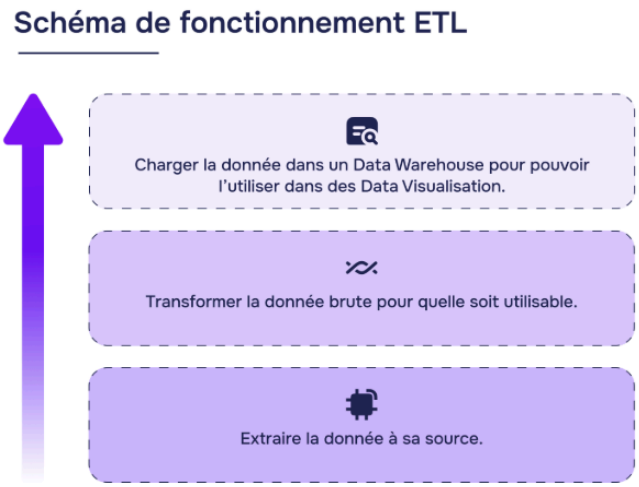
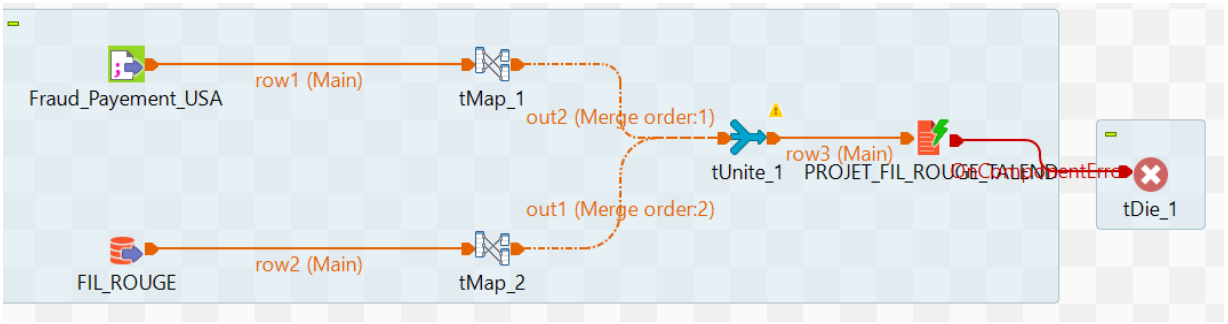


IMAGE-Schéma de la BDD PROJET_FIL_ROUGE

Nous avons collecté deux sources de données différentes, ce qui a posé un problème d'uniformisation. Pour résoudre ce problème, nous avons utilisé l'ETL Talend afin de structurer et de réunir ces données dans un référentiel unique.



IMAGE–Schéma de fonctionnement d'un processus ETL



IMAGE–Flux Talend

La technologie Talend nous a permis d'extraire les données d'un fichier CSV (Fraud_Payment_USA) contenant des transactions frauduleuses impliquant plusieurs victimes. La deuxième source, une base de données appelée PROJET_FIL_ROUGE, contient des données principales qui sont fusionnées avec celles du fichier CSV. L'extraction des données s'est faite via les composants *tFileInputDelimited* pour le fichier CSV et *tDBInput* pour la base de données. Une fois les données extraites, elles sont envoyées vers les composants *tMap* qui permettent de traiter chaque source séparément avant de les fusionner assurant ainsi une meilleure qualité des données, réduisant les erreurs de mapping et optimisant le chargement des lignes volumineuses.

Les colonnes du flux d'entrée *row1* et *row2* sont ensuite mappées vers le flux de sortie *out1* et *out2*. Nous avons conservé toutes les colonnes de la base PROJET_FIL_ROUGE et ajouté certaines colonnes issues du fichier CSV Fraud_Payment_USA. L'objectif est de rassembler toutes les informations nécessaires pour notre analyse. Dans le *tMap1*, nous avons ajouté les colonnes manquantes de PROJET_FIL_ROUGE et supprimé celles qui ne nous semblaient pas pertinentes ou dont nous ne comprenions pas l'utilité.

Lors de cette phase, nous avons réalisé plusieurs transformations afin d'assurer la qualité et la cohérence des données. Tout d'abord, nous avons converti certains formats, notamment les dates qui étaient initialement stockées sous forme de chaînes de caractères, en un format exploitable de type Date. Ensuite, nous avons normalisé les valeurs pour harmoniser les données et éviter les incohérences. Enfin, nous avons traité les valeurs manquantes en attribuant soit des valeurs par défaut, comme "UNKNOWN" pour certains champs, soit en conservant des valeurs null lorsque les informations n'étaient pas disponibles dans le dataset d'entrée.

Les données transformées sont ensuite combinées grâce au composant *tUnite* qui permet de fusionner plusieurs sources de données même en

l'absence d'une clé commune. Après cette fusion, le flux de données est dirigé vers PROJET_FIL_ROUGE_TALEND, notre nouvelle base de données qui nous sert de Data Warehouse.

Comme pour la base PROJET_FIL_ROUGE, nous avons créé une table temporaire qui stocke toutes nos données puis à l'aide de requêtes SQL nous avons séparés les données pour les mettre dans les tables suivantes :

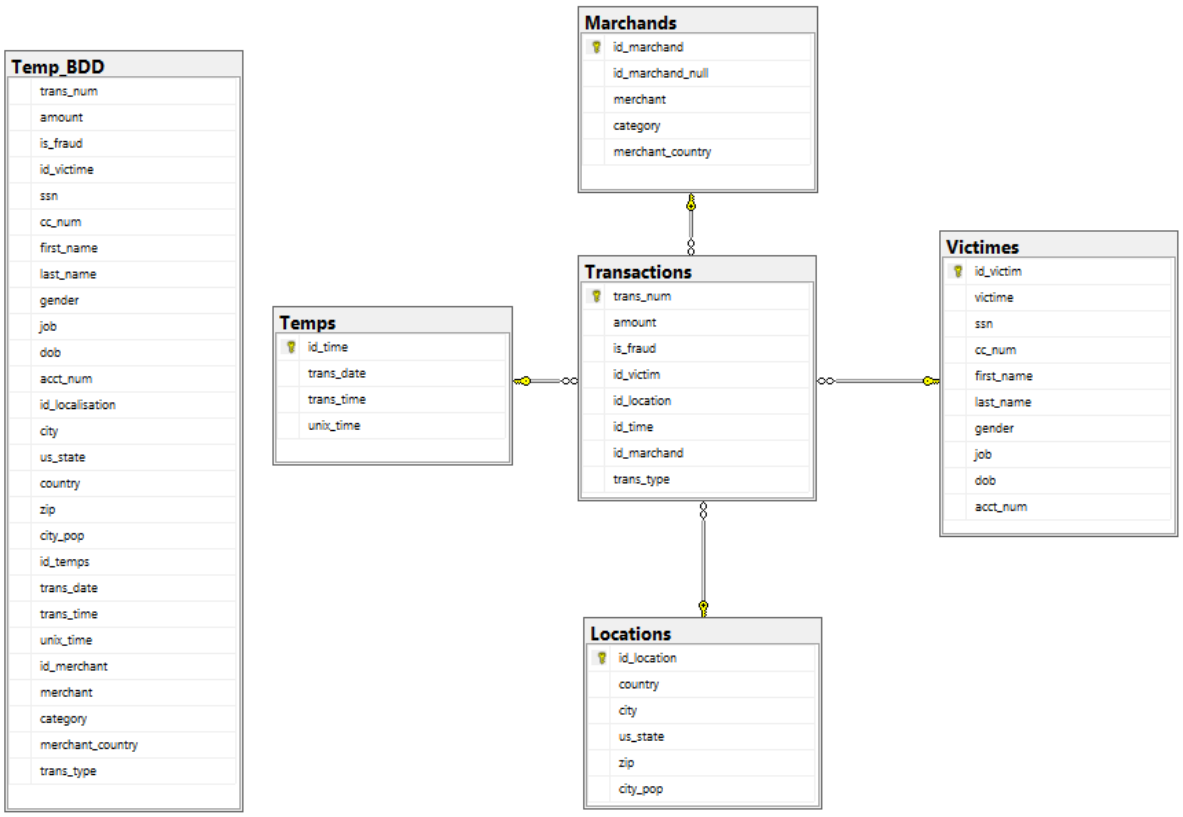


IMAGE-Schéma de la BDD PROJET_FIL_ROUGE_TALEND

Enfin, le composant *tDie* gère la gestion des erreurs afin d'éviter d'envoyer des données erronées et facilite le diagnostic ainsi que la correction des erreurs.

Visualisations

1. Récupération des Données

Les données utilisées pour ce projet ont été extraites de la base de données PROJET_FIL_ROUGE_TALEND, hébergée sur Microsoft SQL Server. Ces données incluent les transactions, les informations sur les victimes, les marchands, la localisation et les périodes temporelles associées.

2. Modélisation des Relations

Dans la vue Modèle de Power BI, des relations ont été établies entre les différentes tables afin d'assurer une analyse efficace des fraudes. Les relations clés comprennent :

- Transactions connectée à Victimes via id_victime
- Transactions connectée à Marchands via id_marchand
- Transactions connectée à Localisation via id_localisation
- Transactions connectée à Temps via id_temps

3. Transformation des Données

Dans Power Query, plusieurs transformations ont été appliquées afin d'enrichir les données. Dans la table Victimes, une nouvelle colonne a été ajoutée pour calculer l'âge des individus à partir de leur date de naissance (dob). De même, dans la table Temps, plusieurs colonnes ont été créées afin d'extraire des informations détaillées sur les transactions, notamment le nom du mois, l'année, ainsi que l'heure à laquelle elles ont eu lieu.

4. Vérification des Données

Une vérification approfondie a été effectuée afin d'assurer la qualité des données. Cela a impliqué l'identification et le traitement des valeurs nulles, la détection des doublons ainsi que la validation des relations entre les tables.

5. Affichage du Rapport et Structure des Pages

Le rapport Power BI est organisé en cinq pages principales. Chaque page comprend une barre de navigation ainsi que des filtres permettant de

sélectionner une période, une localisation, un genre (F/M), une catégorie de transaction et un marchand. Une présentation générale des principaux indicateurs de performance est également intégrée.

La page d'accueil propose une visualisation en donut illustrant la répartition des fraudes selon le type de transaction.

La page dédiée aux victimes met en avant un classement des personnes ayant subi les montants fraudés les plus élevés, ainsi qu'un classement des victimes ayant rencontré le plus grand nombre de transactions frauduleuses. Ces classements sont affichés grâce à une mesure DAX combinée à un composant HTML pour une meilleure lisibilité.

Exemple ci-dessous:

La mesure 9_HTML_Victimes permet de créer un composant visuel personnalisé en HTML pour présenter un classement des victimes en fonction de plusieurs indicateurs issus des données de fraude. Voici une capture d'écran de cette mesure:

```
9_HTML_Victimes =
VAR Top5Victimes =
    TOPN(5,
        ADDCOLUMNS(
            'Victimes',
            "Nb_Fraud", [3_NB_trans_fraud],
            "Classement", [8_Classement_Victimes]
        ),
        [Classement], ASC
    )
RETURN
"<div class='slider-container'>
    <div class='slider'>
        " &
        CONCATENATEX(
            Top5Victimes,
            "<div class='card'>
                <h3>" & Victimes[acct_num] & "</h3>
                <p>Nombre de transactions non frauduleuses : " & [5_NB_trans_non_fraud] & "</p>
                <p>Nombre de Fraudes : " & [Nb_Fraud] & "</p>
                <p>Montant détourné : " & [2_Montant_Total_Fraud] & " $</p>
                <p>Montant max détourné : " & [10_Montant_Max_Fraud] & " $</p>
                <p>Part des fraudes : " & FORMAT([11_Pourcentage_Fraude], "###.0#") & " %</p>
                <p class='classement' style='color: #FFD700;'><strong>Classement : " &
                FORMAT([Classement], "0") & "</strong></p>
            </div>",
            "",
            [Classement]
        )
        & "
    </div>
</div>
```

Ce code DAX construit une mesure qui génère dynamiquement un composant HTML présentant un slider vertical avec les informations des 5 victimes les mieux classées. L'approche combine le traitement des données (via ADDCOLUMNS et TOPN) et la génération de code HTML/CSS (via CONCATENATEX et RETURN d'une chaîne complète) pour offrir une visualisation personnalisée dans Power BI. Les indicateurs affichés incluent le nombre de transactions non frauduleuses, le nombre de fraudes, les montants détournés et la part des fraudes, le tout enrichi par une animation qui dynamise l'affichage.

La page dédiée aux marchands présente un classement des fraudes selon les catégories de marchands ainsi qu'un classement des fraudes par marchands individuels. Ces classements sont générés à l'aide d'une mesure DAX combinée à un composant HTML pour une meilleure lisibilité.

La page de localisation propose une carte interactive affichant les montants des fraudes en fonction de leur emplacement géographique. Un graphique en barres met également en évidence les cinq villes où les fraudes ont généré les montants les plus élevés.

Enfin, la page consacrée aux périodes comprend un graphique en courbe illustrant l'évolution mensuelle du nombre de fraudes ainsi qu'un graphique en barres représentant la répartition des fraudes par heure.

6. Explication des Mesures DAX

Dans le cadre du projet, plusieurs mesures DAX ont été élaborées afin d'établir une analyse approfondie et une visualisation dynamique des données de fraude. Ces mesures permettent d'extraire, d'agréger et de présenter des indicateurs clés tels que le nombre de transactions frauduleuses, les montants détournés, le nombre de fraudes par victime ou par marchand, ainsi que des analyses spécifiques selon la localisation et les périodes.

Exemple d'une mesure calculant le nombre de fraudes:

```
3_NB_trans_fraud = CALCULATE(
    COUNT(Transactions[amount]),
    Transactions[is_fraud] = TRUE()
)
```

La mesure 3_NB_trans_fraud utilise la fonction CALCULATE pour compter le nombre d'enregistrements dans la colonne *amount* de la table *Transactions*, en appliquant un filtre qui ne prend en compte que les transactions frauduleuses. Concrètement, la condition utilisée est :
Transactions[is_fraud] = TRUE()
Cela signifie que la mesure renvoie le nombre total de transactions où la colonne *is_fraud* est vraie, c'est-à-dire le nombre total de transactions frauduleuses.

La mesure 12_HTML_Marchand associe des calculs DAX et du code HTML/CSS pour générer un slider interactif mettant en avant les trois marchands les plus impactés par la fraude selon différents indicateurs.

Tout d'abord, la sélection des données est réalisée grâce à la fonction TOPN, combinée avec ADDCOLUMNS, afin d'extraire les trois marchands les mieux classés à partir de la table correspondante. À cette étape, les informations essentielles telles que le nombre de fraudes (3_NB_trans_fraud) et le classement (13_Classement_Marchand) sont ajoutées. Le tri est ensuite effectué en ordre croissant sur le classement.

Ensuite, la génération du code HTML repose sur la fonction CONCATENATEX, qui assemble pour chaque marchand une carte affichant :

- Le nom du marchand
- Le nombre de transactions non frauduleuses
- Le nombre de fraudes
- Le montant total détourné
- Le montant maximum détourné
- Le classement, affiché en jaune après soustraction de 1

Enfin, l'ensemble est encapsulé dans un conteneur stylisé avec du CSS. Ce dernier définit notamment la taille, un fond en dégradé, des bordures arrondies et une animation de défilement vertical via @keyframes slideVertical.

Ainsi, cette mesure permet d'intégrer un slider dynamique directement dans Power BI, offrant une visualisation fluide et optimisée des marchands les plus concernés par les fraudes.

Machine Learning

Comme solution à notre problématique, nous avons pour objectif principal de rechercher un modèle de Machine Learning capable de détecter de façon fiable les transactions frauduleuses à partir d'un jeu de données financières.

La fraude bancaire représente en effet un enjeu majeur pour les institutions financières, tant du point de vue économique qu'en matière de confiance client. Ainsi, l'enjeu est double : d'une part, maximiser la détection des fraudes afin d'éviter les pertes financières, et d'autre part, minimiser les faux positifs, c'est-à-dire éviter autant que possible de bloquer inutilement des transactions légitimes.

Le choix d'utiliser du Machine Learning pour ce type de problématique se justifie par les limites des méthodes traditionnelles de détection basées sur des règles fixes. Ces règles sont certes faciles à interpréter et rapides à exécuter, mais elles ne s'adaptent pas aisément à l'évolution constante des comportements frauduleux. À l'inverse, le Machine Learning permet d'identifier automatiquement des motifs complexes, parfois subtils, et d'apprendre en continu, tout en nécessitant une préparation rigoureuse des données pour éviter un nombre trop élevé de faux positifs.

Le jeu de donnée Fraud Detection utilisé dans notre solution comporte environ 2,6 millions de transactions financières, chacune associée à différentes informations telles que le profil utilisateur (âge, genre, profession, ville, etc.), les détails spécifiques à la transaction (montant, type, date, heure, etc.), ainsi que des informations sur les commerçants concernés (nom, catégorie, pays). Chaque transaction est également associée à un label indiquant clairement s'il s'agit d'une fraude (1) ou d'une transaction légitime (0). Le principal défi rencontré ici est le très fort déséquilibre dans les données, puisque seulement 0.35% des transactions totales sont frauduleuses, ce qui rend leur détection particulièrement complexe.

Avant la phase d'entraînement des modèles, un soin particulier a été apporté à la préparation des données. Dans un premier temps, un nettoyage rigoureux a été réalisé afin de vérifier l'absence d'anomalies majeures, sans toutefois supprimer les valeurs extrêmes, ces dernières pouvant être légitimement frauduleuses. Par la suite, un encodage spécifique, via la méthode OneHotEncoding, a été appliqué aux variables catégoriques pour les rendre exploitables par les modèles de Machine Learning. Les variables

numériques, quant à elles, ont été standardisées afin d'éviter les biais potentiels liés aux échelles de valeurs très différentes entre les variables.

Compte tenu du fort déséquilibre entre la classe majoritaire (transactions normales) et la classe minoritaire (fraudes représentant seulement 0.35% du total), une stratégie de rééquilibrage a été mise en place. Un downsampling de la classe majoritaire a permis de réduire artificiellement le nombre de transactions normales afin que les deux classes soient plus équilibrées en nombre. Cette approche vise à améliorer la capacité du modèle à détecter les fraudes, plutôt que de favoriser la prédiction systématique de la classe majoritaire.

Les données ont ensuite été divisées en deux ensembles : un ensemble d'entraînement (70% des données équilibrées) servant à construire et optimiser les modèles, et un ensemble de test (30% des données équilibrées) permettant d'évaluer objectivement leurs performances. En parallèle, un échantillon de 20 transactions supplémentaires (10 fraudes et 10 non-fraudes) a été isolé pour effectuer des cas pratiques, afin de mieux apprécier la capacité prédictive réelle du modèle dans des scénarios concrets.

Parmi les différents modèles testés, trois se sont distingués par leur performance relative : tout d'abord, le modèle Support Vector Machine (SVM), qui malgré des tests approfondis en cross-validation, a produit un score insuffisant d'environ 0.6, confirmant son inadéquation pour ce type de problème.

Ensuite, le modèle RandomForest, bien qu'offrant des résultats honorables, s'est révélé moins performant qu'espéré sur notre jeu de données, notamment en raison de difficultés à généraliser correctement.

Enfin, XGBoost s'est nettement distingué en atteignant un score AUC particulièrement élevé de 0.98, indiquant ainsi un excellent équilibre entre précision et rappel, tout en étant capable de gérer efficacement le déséquilibre des classes.

Le modèle retenu, XGBoost, a bénéficié d'une optimisation rigoureuse de ses hyperparamètres grâce à l'utilisation de GridSearchCV. Après une recherche approfondie, les meilleurs paramètres identifiés sont : un nombre total d'arbres (n_estimators) fixé à 500, une profondeur maximale des arbres (max_depth) de 10, un taux d'apprentissage (learning_rate) à 0.1, et un ajustement automatique du paramètre scale_pos_weight.

Le choix de 500 arbres permet au modèle de saisir des relations complexes dans les données sans tomber dans le surapprentissage. Un nombre trop faible pourrait empêcher une modélisation efficace, tandis qu'un nombre trop élevé pourrait nuire à la généralisation.

La profondeur maximale des arbres, fixée à 10, permet au modèle de capter efficacement des relations subtiles sans surapprentissage. Une profondeur trop faible limiterait la performance, tandis qu'une profondeur trop élevée risquerait d'induire un surapprentissage.

Le taux d'apprentissage de 0.1 assure un apprentissage progressif et stable. Ce taux modéré est idéal pour garantir une bonne convergence tout en évitant les erreurs dues à un apprentissage trop rapide ou trop lent.

L'ajustement automatique du paramètre scale_pos_weight est essentiel en raison du déséquilibre entre transactions frauduleuses et normales. Ce paramètre permet de renforcer l'importance des cas minoritaires, améliorant ainsi la détection des fraudes tout en limitant les faux négatifs.

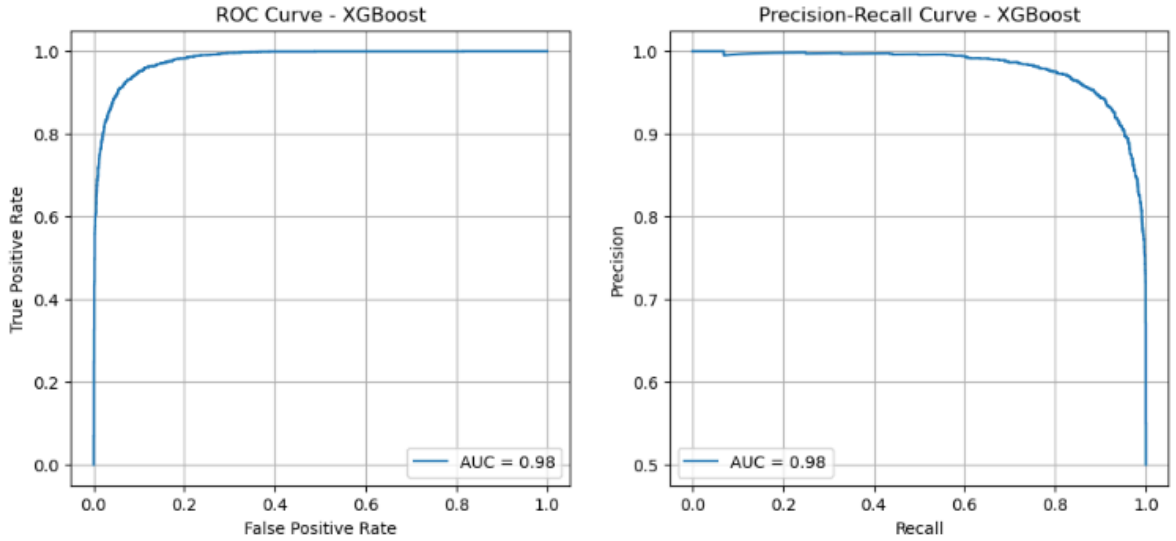
Ces hyperparamètres optimisés assurent ainsi un équilibre optimal entre précision et généralisation du modèle, particulièrement adapté à la détection des fraudes.

```
Évaluation du modèle : XGBoost
Seuil optimal choisi : 0.47
Confusion Matrix :
[[2613  211]
 [ 195 2629]]

Classification Report :
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      2824
           1       0.93      0.93      0.93      2824

    accuracy                           0.93      5648
   macro avg       0.93      0.93      0.93      5648
weighted avg       0.93      0.93      0.93      5648
```



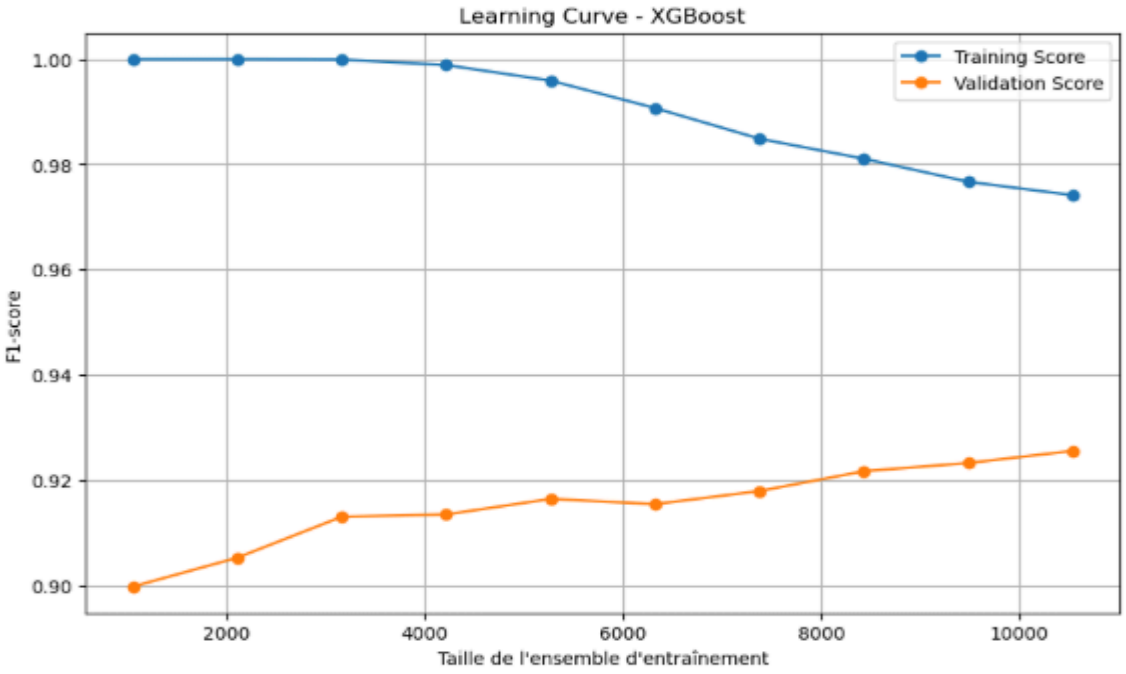
Concernant les features temporelles, malgré leur apparente pertinence initiale (jour de la semaine, heure des transactions), les tests réalisés ont clairement démontré que leur ajout n'apportait pas d'amélioration significative aux performances du modèle. Ces variables temporelles ont donc été volontairement exclues afin de privilégier une simplicité efficace.

La définition du seuil optimal d'identification des fraudes constitue une étape critique du projet. Plusieurs seuils, comme 50%, 72%, 76%, ou même des seuils beaucoup plus bas (0.13, 0.15, 0.20) ont été testés.

Seuil	Faux positifs (transactions normales classées en fraude)	Faux négatifs (fraudes ratées)
50%	1	1
76%	0	2
80%	0	2
72%	1	2
73%	1	2
0.13	1	1
0.15	1	1
0.20	1	1

Après analyse approfondie, un seuil classique de 50% s'est imposé comme étant optimal, puisqu'il offre le meilleur compromis possible entre précision et rappel, limitant à la fois le nombre de faux positifs et celui de fraudes non détectées. À l'inverse, des seuils plus bas ont montré des lacunes en laissant passer un nombre inacceptable de fraudes.

L'analyse détaillée des learning curves a également été réalisée pour comparer les capacités de généralisation des différents modèles testés. RandomForest a présenté un surapprentissage significatif, avec un très bon score sur l'entraînement mais des performances relativement faibles en validation, indiquant une faible généralisation aux nouvelles données.



À l'inverse, XGBoost a montré un apprentissage efficace, équilibrant correctement performance et généralisation, avec un écart réduit entre les résultats en entraînement et en validation. Il en ressort que XGBoost représente clairement le meilleur compromis performance/généralisation pour notre contexte spécifique.

Les résultats obtenus avec le modèle XGBoost optimisé sont très satisfaisants : il atteint une précision et un rappel élevés, tous deux de 93%, ainsi qu'un excellent F1-score de 93%. Ce résultat valide pleinement l'intérêt de ce modèle pour une mise en application réelle, tout en confirmant sa robustesse face à un dataset fortement déséquilibré.

Suite à nos recherches, on peut en conclure que le Machine Learning, et plus particulièrement le modèle XGBoost, permet d'apporter une réponse flexible et robuste à une problématique complexe et évolutive comme la détection de fraudes financières.

Difficultés rencontrées

L'un des principaux défis de ce projet a été l'accès aux données. Les informations bancaires étant souvent protégées et difficiles à obtenir, il a été nécessaire de travailler avec des jeux de données générés. Cependant, ces données synthétiques ne reflètent pas toujours parfaitement la réalité, ce qui peut limiter la fiabilité des analyses.

Un autre obstacle important réside dans l'absence de certaines informations clés, comme la localisation exacte des fraudes dans les deux datasets. Cette lacune complique l'identification des schémas frauduleux et peut introduire des biais dans le modèle d'analyse. De plus, la compréhension de l'utilité de chaque variable des données brutes s'est révélée complexe en raison du manque de précision dans leur description par la source, rendant l'exploitation et l'interprétation des données plus délicates.

Axes d'amélioration

Pour optimiser le flux ETL sous Talend, il serait pertinent d'explorer l'intégration de nouvelles sources de données afin d'enrichir les analyses futures. L'optimisation du stockage en base de données constitue également un levier d'amélioration : l'archivage des anciennes transactions permettrait de réduire la charge en mémoire et d'améliorer les performances globales.

De plus, l'implémentation d'index sur les colonnes les plus sollicitées accélérerait les requêtes et fluidifierait le traitement des données.

Concernant le Machine Learning, plusieurs pistes peuvent être envisagées pour renforcer l'efficacité du modèle. L'exploitation de nouvelles variables issues de l'historique des utilisateurs, l'expérimentation de modèles plus avancés comme les réseaux de neurones ou encore l'enrichissement progressif du jeu de données avec des transactions réelles pourraient significativement améliorer la détection des fraudes.

Conclusion

En plus des traitements sous Talend qui nous ont permis de lier nos deux sources de données, d'autres ajustements ont été faits pour structurer le dashboard et faciliter l'analyse.

Les transformations incluent l'enrichissement des tables, le traitement des valeurs nulles et des doublons, ainsi que des mesures DAX avancées. Certaines intègrent du HTML/CSS pour des visualisations personnalisées. La cohérence des résultats a été vérifiée via des requêtes SQL.

Notre projet nous a aussi permis de démontrer l'efficacité des méthodes de Machine Learning dans la détection de fraudes bancaires. Grâce à une approche rigoureuse de préparation et de transformation des données, nous avons pu structurer un pipeline d'analyse performant, en intégrant des jeux de données variés et en optimisant leur exploitation via Talend et SQL Server.

L'entraînement de plusieurs modèles nous a permis d'évaluer leurs performances respectives et d'identifier XGBoost comme la solution la plus adaptée. Ce modèle a offert un excellent compromis entre précision et rappel, tout en limitant les faux positifs et en détectant efficacement les transactions frauduleuses.

Néanmoins, des pistes d'amélioration restent envisageables. Une approche basée sur le Deep Learning pourrait être explorée pour capturer des patterns encore plus complexes dans les données. De plus, l'intégration de flux de données en temps réel permettrait d'adapter le modèle en continu face aux nouvelles stratégies des fraudeurs.

En conclusion, notre projet met en lumière l'importance des technologies de data science dans la lutte contre la fraude bancaire et souligne le potentiel

des algorithmes prédictifs pour renforcer la sécurité des transactions financières.

Sources

Fraud Detection :
https://huggingface.co/datasets/Nooha/cc_fraud_detection_dataset

Fraud Payement :
https://huggingface.co/datasets/saifhmb/FraudPaymentData