

Projet : Détection de Fraude 2023

9 FEVRIER

Chef de projet :

Aude BOUCHONNET

Equipe de développeurs :

Tetyana TERASENKO

Briand BAKOUZOU

Osman ARSLAN

Axel ARCIDIACO



SOMMAIRE

| | |
|------------------------------|---|
| 1- Contexte | 3 |
| 2- Planification | 4 |
| 3- Guide utilisateur | 6 |
| 4- Documentation technique | |
| 5- Difficultés rencontrées | |
| 6- Perspectives d'évolutions | |
| 7- Conclusion | |
| 8- Bilan de groupe | |
| 9- Bilans personnels | |

Contexte

Dans ce projet, nous travaillons au sein d'une société spécialisée dans la sécurité des systèmes bancaires et allons mener une étude approfondie dans le but de créer et proposer différents modèles d'approches qui détectent les activités suspectes qui sont révélatrices de fraude.

En effet, depuis des décennies l'utilisation des cartes de crédit pour réaliser des achats à augmenter de manière exponentielle. Le premier facteur fut la part de marché de plus en plus grande pour les achats en ligne.

Il a été développé une implémentation d'argent qui offre aux utilisateurs de téléphone la possibilité de transférer de l'argent entre eux en utilisant leurs mobiles comme un porte-monnaie électronique. Suite à la collecte suffisante de données nous pouvons désormais mettre en place un système efficace de prédiction de fraudes à la carte bancaire à l'aide d'un simulateur Mobile Money mis en place à travers cette étude.

Planification

1- Stratégie

Pour mener à bien cette étude, nous avons mis un place un planning prévisionnel qui nous permet de suivre, l'avancée en temps quasi-réel, du projet ainsi que les différentes tâches que nous nous sommes fixés préalablement. L'objectif de notre planification est de déterminer le coût, les ressources mobilisées et la meilleure manière d'ordonnancer toutes les tâches à effectuer. L'enjeu étant de recontextualiser l'étude afin d'avoir une vision claire de notre projet, de ses attentes et de choisir la manière de procéder la plus efficace pour le réaliser dans un minimum de temps (date butoir de 10 jours ici). Le tableau ci-dessous nous montre l'organisation que nous avons décider de suivre durant ce projet :

Rétroplanning :

| | Mercredi 01/02 | Jeudi 02/02 | Vendredi 03/02 | | Lundi 06/02 | Mardi 07/02 | Mercredi 08/02 | Jeudi 09/02 | Vendredi 10/02 |
|------------|------------------------------|----------------|-------------------|--|----------------|----------------------|--------------------|----------------|----------------------|
| Matin | Mise en place | Meeting | Meeting | | Meeting | Meeting | Meeting | Rapport | Finalisation du code |
| | | Début IA | IA | | IA | Début de l'interface | | | |
| | | | | | | | | | |
| | | | | | | | | | Présentation |
| | | | | | | | | | |
| Après-midi | Meeting | Meeting | Meeting | | Meeting | Meeting | Meeting | | Soutenance |
| | | IA | IA | | IA | | | | |
| | | | | | | | | | |
| | Exploration de données | | | | | | | | |
| | Fin de MERISE et BDD | | | | Fin IA | | Fin de l'interface | | |
| | Rapport partie Merise et BDD | Meeting | Meeting | | Meeting | Meeting | Meeting | Meeting | |

2- Mise en place des IDE

Après concertation par Brainstorming entre les différents développeurs et le chef de projet, nous avons choisi l'application web Trello comme support de planning du fait de sa simplicité et de ses outils plus que suffisants pour répondre à nos besoins



Par la suite nous avons procédé au découpage du projet en diverses tâches à accomplir aussi bien à plusieurs que seuls. En premier lieu nous avons pris en compte les envies de chacun pour pouvoir répartir les tâches en se basant sur la motivation de chaque membre du groupe. Un meeting collectif sera effectué à chaque début de demi-journée afin d'avoir une idée d'ensemble concrète de l'avancée du projet et pouvoir trouver une solution collectivement en cas de blocage ou d'éventuels retards car toutes les parties du projet sont liées.

Enfin, nous avons mis en différents moyens de communication pour nos échanges mutuelles et le partage des fichiers ou programmes. En effet, l'intérêt de tout ceci est d'avoir un gain de temps au niveau de la recherche de ces derniers et également palier aux éventuelles absences. Pour ce faire nous utiliserons Microsoft Teams comme base principale de stockage des fichiers et en cas d'imprévu, nous utiliserons l'application Discord pour les compléments d'informations.

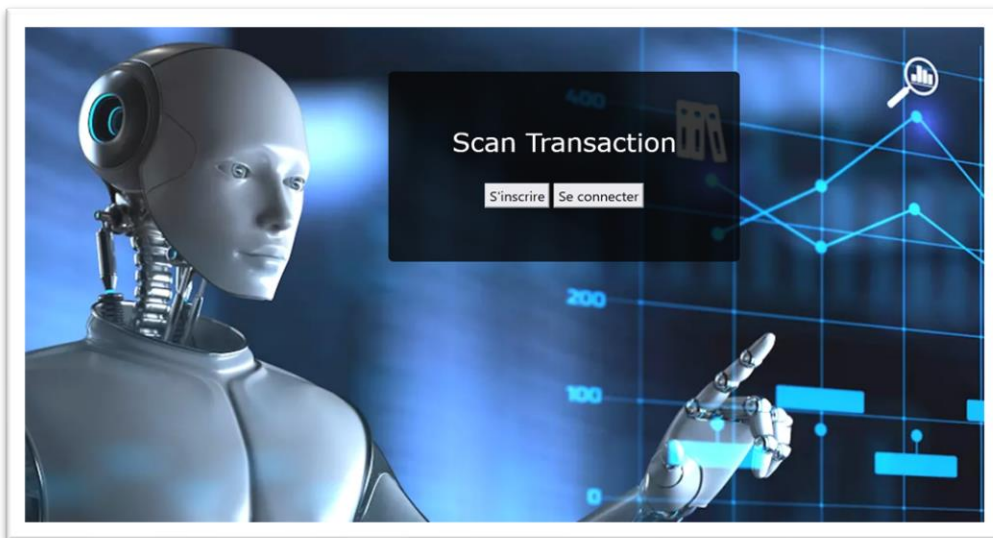


Guide utilisateur

1. Première page : BIENVENUE

Cette première permet d'offrir 2 fonctionnalités :

- S'inscrire : permet d'intégrer de nouveaux utilisateurs dans la base de données
- Se connecter : permet d'accéder à la page de login pour les utilisateurs appartenant déjà à la base de données.



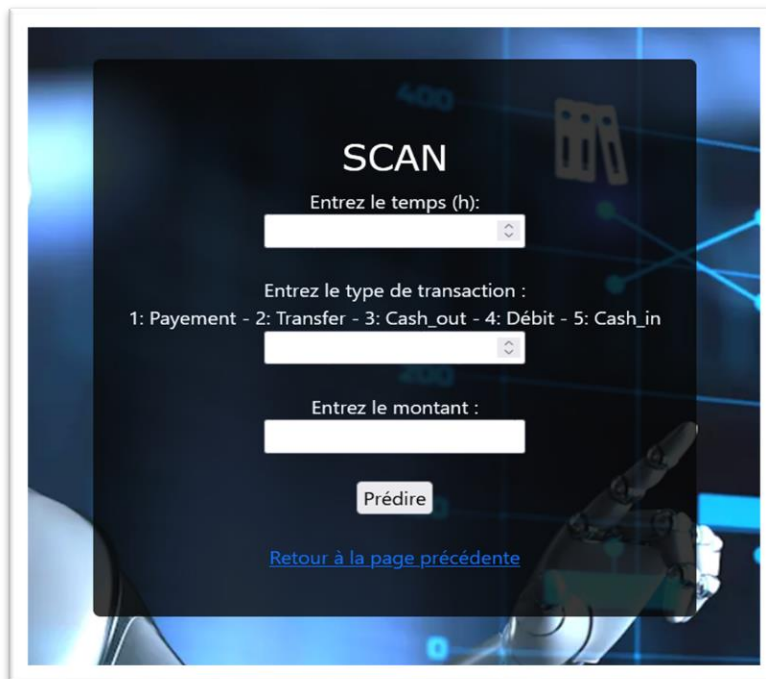
2. Deuxième page : LOGIN

Permet à l'utilisateur de s'identifier avec son identifiant et son mot de passe.



Troisième page : SCAN

L'utilisateur arrive ensuite sur la page où il sera invité à renseigner les différents paramètres nécessaires à la prédiction dans les emplacements dédiés puis les valide. Il peut bien évidemment revenir à la page précédente s'il le désire également.

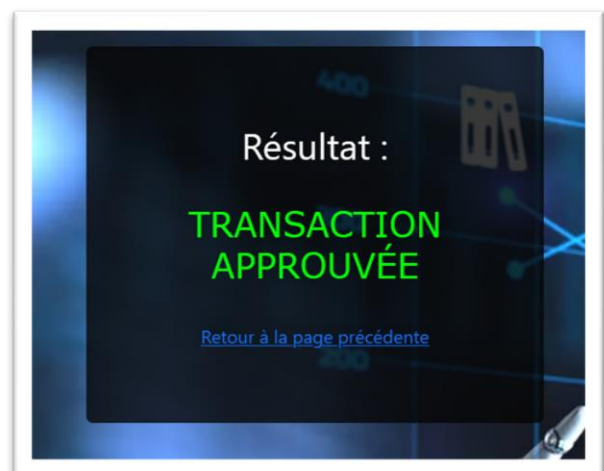
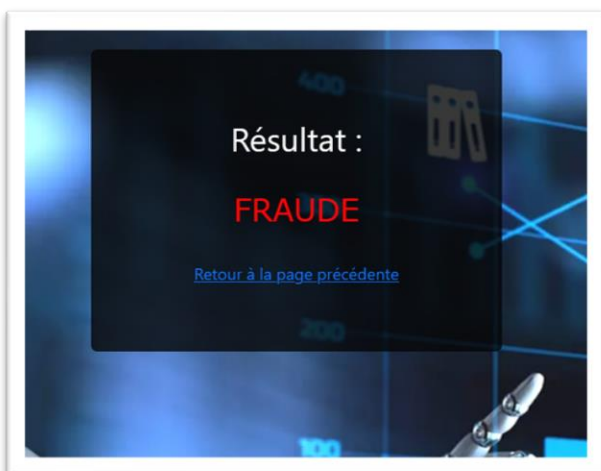


The screenshot shows a web form titled "SCAN" on a dark blue background with a futuristic, grid-like pattern. The form contains the following elements:

- A label "Entrez le temps (h):" followed by a white input field with a small downward arrow on the right.
- A label "Entrez le type de transaction :" followed by a list of options: "1: Payement - 2: Transfer - 3: Cash_out - 4: Débit - 5: Cash_in". Below this is a white dropdown menu with a downward arrow.
- A label "Entrez le montant :" followed by a white input field.
- A white button labeled "Prédire".
- A blue link at the bottom that says "Retour à la page précédente".

4. Quatrième page : PREDICTION

Après validation des données renseignées sur la page précédente, une page de prédiction apparaît avec le verdict que notre modèle générera en termes de prédiction, à savoir une situation de fraude ou non. De ce résultat, il pourra revenir aux pages précédentes.



Documentation Technique

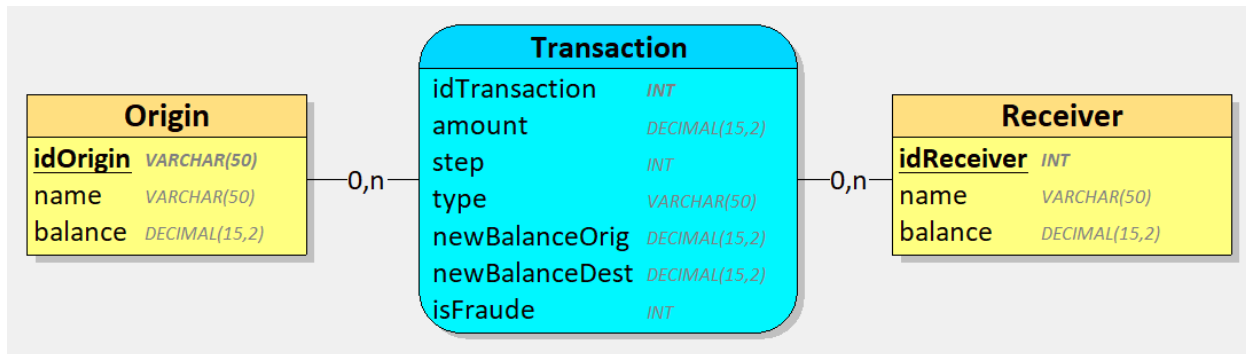
1- Etude MERISE du projet et BDD relationnelle

a- Dictionnaire de données

| Table | Variable | Type Variable | Description / Fonction |
|--------------------|----------------|---------------|---|
| Transaction | transactionId | Integer | Identifiant des transactions |
| | step | Integer | Cartographie une unité de temps dans le monde réel. Dans ce cas, 1 pas correspond à 1 heure de temps. |
| | type | Varchar | CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER. |
| | amount | Float | Montant de la transaction en devise locale. |
| | isFraud | Integer | Identifie si une transaction est une fraude (1) ou si elle ne l'est pas (0). |
| | newBalanceOrig | Float | Contient le nouveau solde du compte à l'origine de la transaction. |
| | newBalanceDest | Float | Contient le nouveau solde du compte destinataire de la transaction. |
| Origin | nameOrig | Varchar | Nom du compte qui a initiée la transaction. |
| | oldbalanceOrg | Float | Solde initial du compte qui est à l'origine de la transaction. |
| Receiver | nameDest | Varchar | Nom du compte qui est destinataire de la transaction. |
| | oldbalanceDest | Float | Solde initial du compte qui est destinataire de la transaction. |

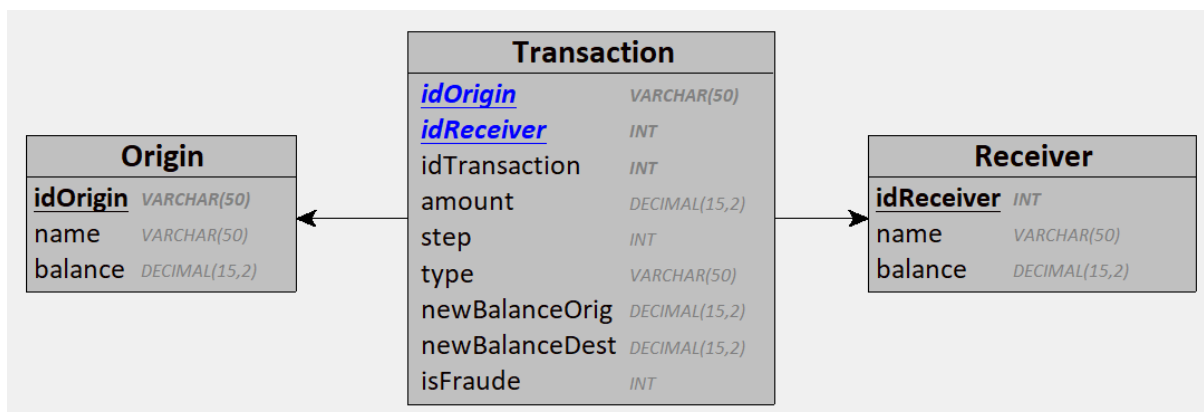
b- Modèle Conceptuel des Données (MCD)

Suite au dictionnaire de données présenté précédemment, nous avons pu mettre en place le MCD représentatif claire des données du système d'information à concevoir.



c- Modèle Logique de Données (MLD)

A l'aide du MCD réalisé précédemment, nous avons conçu le MLD associé à la représentation des données d'un système d'information.



d- BDD (MySQL ou SQLite)

Le modèle de base de données relationnelle MySQL que nous avons mis en place à l'aide de notre MCD et MLD nous permet de générer des tableaux qui permettent de retomber exactement sur la base de données initiale. Cela signifie que toutes les données traitées sur MySQL sont stockées dans des tableaux pouvant être reliés les uns aux autres via des clés et ainsi obtenir la même structure que les données fournis de départ. Cela signifie que si nous interrogeons notre BDD, il en ressortira les réponses correspondantes et inversement, si on souhaite lui rajouter de nouvelles données, ces dernières iront se placer dans l'emplacement correspondant. Vous trouverez en index un quelques aperçus de la BDD MySQL.

1- Intelligence Artificielle

a- Exploration des données

- Dans un premier temps, examinons la forme de nos données en sortant les différents types de variables qu'il contient :

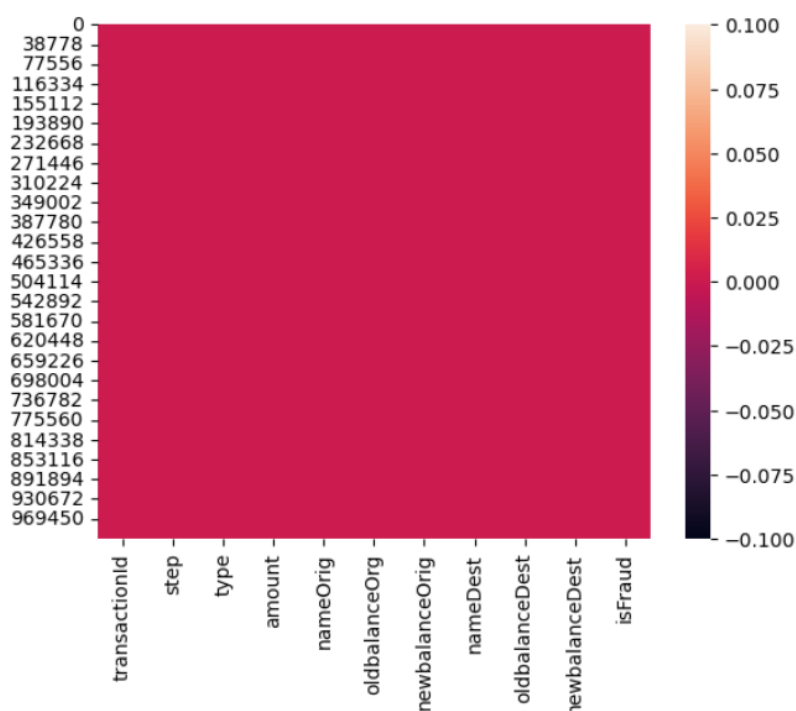
| # | Column | Non-Null Count | | Dtype |
|----|----------------|----------------|----------|---------|
| 0 | transactionId | 1008213 | non-null | int64 |
| 1 | step | 1008213 | non-null | int64 |
| 2 | type | 1008213 | non-null | object |
| 3 | amount | 1008213 | non-null | float64 |
| 4 | nameOrig | 1008213 | non-null | object |
| 5 | oldbalanceOrg | 1008213 | non-null | float64 |
| 6 | newbalanceOrig | 1008213 | non-null | float64 |
| 7 | nameDest | 1008213 | non-null | object |
| 8 | oldbalanceDest | 1008213 | non-null | float64 |
| 9 | newbalanceDest | 1008213 | non-null | float64 |
| 10 | isFraud | 1008213 | non-null | int64 |

On apprend par cette commande que dans l'ensemble des variables contenant nos données, on compte 8 variables quantitatives et 3 variables qualitatives (de type objet).

- On se lance ensuite dans l'éventuelle présence de valeurs manquantes dans nos données pour finalement n'en trouver aucune comme le montre le schéma ci-dessous :

```
sns.heatmap(df.isna())
```

<AxesSubplot:>



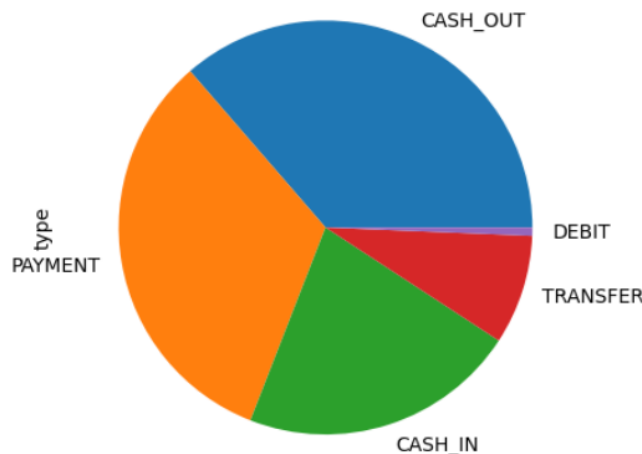
- Analyse de la signification des variables

TransactionId : Identifiant de la transaction.

Step : Temps de la transaction en heure.

Type : type de la transaction qui peut prendre 5 valeurs différentes PAYMENT, CASH_IN, CASH_OUT, DEBIT, TRANSFER.

```
: df['type'].value_counts().plot.pie()
: <AxesSubplot:ylabel='type'>
```



```
: df['type'].value_counts(normalize=True)
: CASH_OUT    0.363645
: PAYMENT     0.327180
: CASH_IN     0.217155
: TRANSFER    0.085598
: DEBIT       0.006422
: Name: type, dtype: float64
```

Amount : montant de la comme concernée par la transaction

NameOrig : identifiant du compte à l'origine de la transaction

OldbalanceOrg : solde avant la transaction du compte origine

NewbalanceOrig : solde après la transaction du compte origine

NameDest : identifiant du compte destinataire de la transaction

OldfbalanceDest : solde avant la transaction du compte destinataire

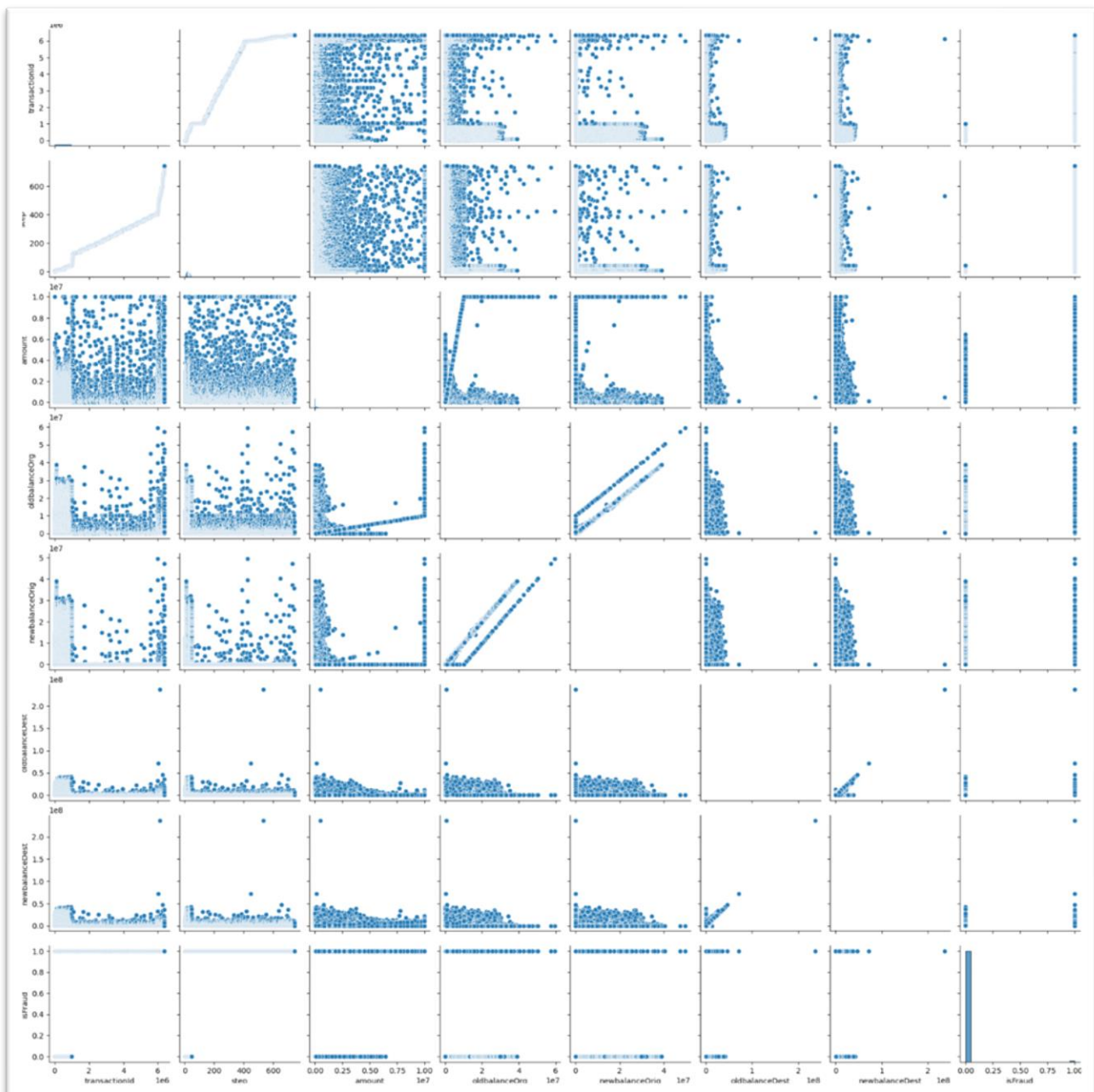
NewbalanceDest : solde après la transaction du compte destinataire

- Visualisation de la répartition de nos données Target et on constate qu'il y a très peu de données classé fraude dans l'ensemble, on a donc à faire à des données déséquilibrées.

```
: df['isFraud'].value_counts(normalize=True)
: 0    0.991854
: 1    0.008146
: Name: isFraud, dtype: float64
```

- Relation variables/Target

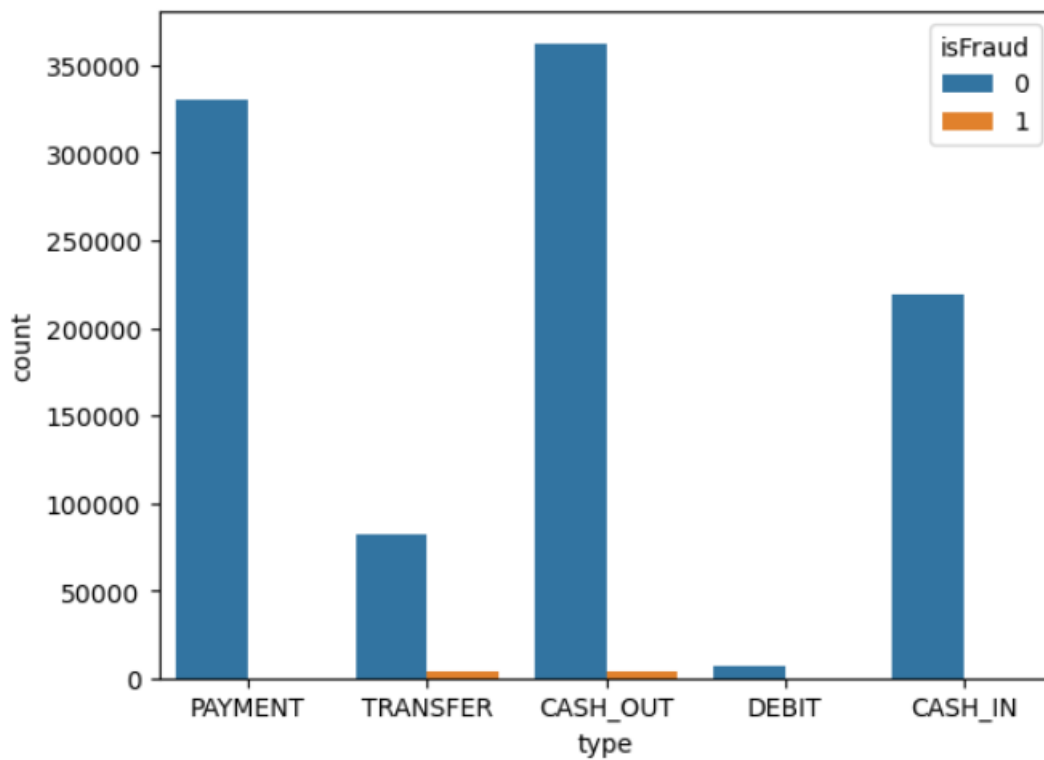
Ci-dessous l'ensemble des graphiques possibles avec le dataset généré avec Seaborn :



- Observons d'abord la relation avec le type de transaction. On constate que les données de type fraude sont présentes que dans deux types de transaction, à savoir les transactions de « TRANSFER » et de « CASH-OUT ».

```
sns.countplot(x='type', hue='isFraud', data=df)
```

```
<AxesSubplot:xlabel='type', ylabel='count'>
```



b- Modification globale des données et préparatifs pour les modèles

Dans l'ensemble de cette démarche nous avons enlevé l'attribut TransactionId car de façon raisonnable on peut considérer qu'il n'est pas corrélé à la fraude car ils sont générés par le système informatique.

Les attributs nameOrig et nameDest représentent les noms des comptes pour lesquels on suppose qu'ils n'auront pas d'influence sur la fraude.

La matrice de confusion est un outil de mesure de la performance des modèles de classification à 2 classes ou plus. Dans le cas binaire, la matrice de confusion est un tableau à 4 valeurs représentant les différentes combinaisons de valeurs réelles et valeurs prédites comme dans la figure ci-dessous.

| Confusion matrix | | Reality | |
|------------------|--------------|---------------------|---------------------|
| | | Negative : 0 | Positive : 1 |
| Prediction | Negative : 0 | True Negative : TN | False Negative : FN |
| | Positive : 1 | False Positive : FP | True Positive : TP |

Figure 1 Matrice de confusion

True Negative : nombre de réelle prédiction négative que le modèle a trouvé.

False Negative : nombre de fausse prédiction négative que le modèle a trouvé.

False positive : nombre de fausse prédiction positive que le modèle a trouvée.

True positive : nombre de réelle prédiction positive que le modèle a trouvée.

La matrice de confusion est le socle incontournable sur lequel s'appuient toutes les métriques de classification : accuracy, F1- score, precision, recall...

Définition :

La precision et le recall sont deux métriques pour évaluer la performance des modèles de classification à 2 classes ou plus. Ces métriques sont basées sur la matrice de confusion. La precision et le recall sont deux métriques qui se concentrent sur la performance du modèle concernant les individus positifs :

- accuracy : L'accuracy permet de décrire la performance du modèle sur les individus positifs et négatifs de façon symétrique. Elle mesure le taux de prédictions correctes sur l'ensemble des individus.

- precision : La précision est également appelée Positive Predictive Value. Elle correspond au taux de prédictions correctes parmi les prédictions positives. Elle mesure la capacité du modèle à ne pas faire d'erreur lors d'une prédiction positive.

- recall : Le recall correspond au taux d'individus positifs détectés par le modèle: Il mesure la capacité du modèle à détecter l'ensemble des individus positifs.
- F1-score : Le F1-score permet de résumer les valeurs de la precision et du recall en une seule métrique. Le F1-score est la moyenne harmonique de la precision et du recall.

c- Premiers modèles de classification : La régression logistique

L'algorithme de régression ne traite pas les données catégorielles entant que string par conséquent nous avons transformé les catégories PAYMENT, CASH_IN, CASH_OUT, DEBIT, TRANSFER en chiffre de 1 à 5.

Pour l'ensemble des modèles créés avec l'algorithme de regression logistique la séparation des données se fait de la façon suivante : data d'entraînement 70% et un data de test 30%

Pour chaque modèle il sera précisé les attributs conservés.

- Premier modèle

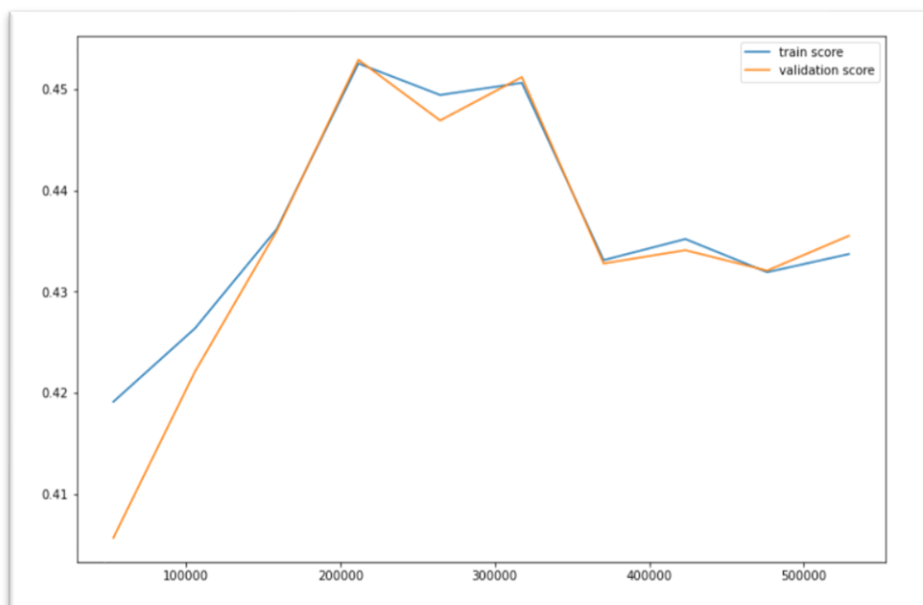
Les colonnes concernées sont : step, type, amount, oldbalanceOrig, OldbalanceDest, newbalanceOrig, newbalanceDest,

On entraine le modèle avec les hyperparamètres par défaut.

On obtient la matrice de confusion suivante :

| | |
|------------|----------|
| [[297782 | 2317 |
| [1063 | 1302]] |

On obtient la courbe de d'apprentissage suivante :



Ci-dessous le tableau d'évaluation

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.99 | 0.99 | 300099 |
| 1 | 0.36 | 0.55 | 0.44 | 2365 |
| accuracy | | | 0.99 | 302464 |
| macro avg | 0.68 | 0.77 | 0.71 | 302464 |
| weighted avg | 0.99 | 0.99 | 0.99 | 302464 |

- Deuxième modèle

Si les zéros représentent des données manquantes et qu'on décide de les enlever du dataset.

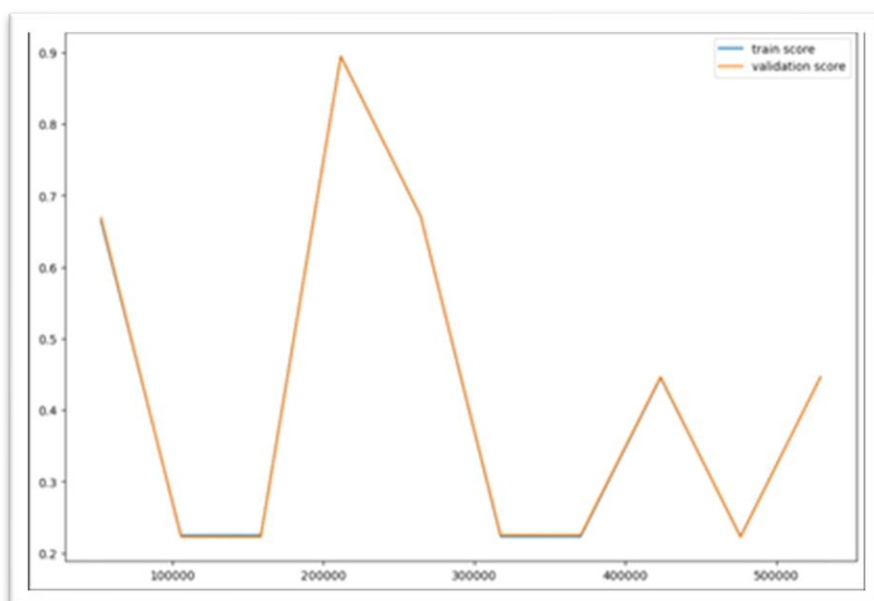
Attributs conservés : step, type, amount

Les résultats semblaient meilleurs, augmentation des VP et VN et diminution des FN et FP.

Ci-dessous le tableau d'évaluation

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 300099 |
| 1 | 0.98 | 0.82 | 0.89 | 2365 |
| accuracy | | | 1.00 | 302464 |
| macro avg | 0.99 | 0.91 | 0.95 | 302464 |
| weighted avg | 1.00 | 1.00 | 1.00 | 302464 |

En générant la courbe d'apprentissage on obtient le graphique ci-dessous et on se rend compte que l'apprentissage ne se réalise pas le modèle en overfitting.



- Troisième modèle avec Gridsearch

On test avec Gridsearch avec les mêmes attributs que le 2ème essai pour optimiser les hyperparamètres entre les types de pénalité (L1 et L2) et de maximum d'itération (max_iterint) pour le solveur.

Les types de pénalités sont les suivantes :

'l2' : ajouter un terme de pénalité L2 et c'est le choix par défaut ;

'l1' : ajouter un terme de pénalité L1 ;

max_iterint est le nombre maximal d'itérations nécessaires pour que les solveurs convergent.

Ci-dessous le tableau d'évaluation :

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 1.00 | 1.00 | 300099 |
| 1 | 0.98 | 0.82 | 0.89 | 2365 |
| accuracy | | | 1.00 | 302464 |
| macro avg | 0.99 | 0.91 | 0.95 | 302464 |
| weighted avg | 1.00 | 1.00 | 1.00 | 302464 |

On obtient les mêmes résultats car les mêmes résultats que dans le 2ème essais les paramètres. Les meilleurs paramètres établis par le gridsearch correspondent aux hyperparamètres par défauts de l'algorithme.

- Quatrième modèle

Le 4ème modèle on a appliqué SMOTE afin d'augmenter la catégorie minoritaire, ici les non fraude, avec les 3 attributs step, type, amount.

Les résultats obtenus sont très mauvais tous les l'ensemble des éléments du X_test sont reconnus comme des fraudes.

Ci-dessous le tableau d'évaluation :

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.00 | 0.00 | 0.00 | 300099 |
| 1 | 0.01 | 1.00 | 0.02 | 2365 |
| accuracy | | | 0.01 | 302464 |
| macro avg | 0.00 | 0.50 | 0.01 | 302464 |
| weighted avg | 0.00 | 0.01 | 0.00 | 302464 |

- Cinquième modèle

Le 5ème essai consiste à appliquer SMOTE comme dans le 4ème essai mais cette fois si avec les 7 attributs step, type, amount, oldbalanceOrig, OldbalanceDest, newbalanceOrig, newbalanceDest.

Les résultats obtenus sont moins mauvais que dans l'essai 4 mais reste peu concluant.

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.73 | 0.85 | 300099 |
| 1 | 0.03 | 0.99 | 0.06 | 2365 |
| accuracy | | | 0.74 | 302464 |
| macro avg | 0.51 | 0.86 | 0.45 | 302464 |
| weighted avg | 0.99 | 0.74 | 0.84 | 302464 |

La régression logistique ne semble pas appropriée pour répondre à la problématique d'identifications des fraudes compte tenu des résultats obtenus par les diverses méthodes essayer ci-dessus.

d- Deuxième modèle de classification : L'arbre de décision (ID3)

Dans un premier temps, l'étude de données a confirmé la présence de données aussi bien quantitatives que qualitatives dans notre Dataset. En effet, avec la présence de ces données se trouvent dans la colonne « type », la colonne « nameOrig » et « nameDest » comme le montre l'extrait du tableau ci-dessous.

| | transactionId | step | type | amount | nameOrig | oldbalanceOrig | newbalanceOrig | nameDest | oldbalanceDest | newbalanceDest | isFraud |
|---|---------------|------|----------|----------|-------------|----------------|----------------|-------------|----------------|----------------|---------|
| 0 | 0 | 1 | PAYMENT | 9839.64 | C1231006815 | 170136.00 | 160296.36 | M1979787155 | 0.00 | 0.00 | 0 |
| 1 | 1 | 1 | PAYMENT | 1864.28 | C1666544295 | 21249.00 | 19384.72 | M2044282225 | 0.00 | 0.00 | 0 |
| 2 | 2 | 1 | TRANSFER | 181.00 | C1305486145 | 181.00 | 0.00 | C553264065 | 0.00 | 0.00 | 1 |
| 3 | 3 | 1 | CASH_OUT | 181.00 | C840083671 | 181.00 | 0.00 | C38997010 | 21182.00 | 0.00 | 1 |
| 4 | 4 | 1 | PAYMENT | 11668.14 | C2048537720 | 41554.00 | 29885.86 | M1230701703 | 0.00 | 0.00 | 0 |

Il a donc fallu dans un premier temps les adapter de manière à pouvoir être pris en compte dans notre modèle. D'abord on remplace dans les colonnes « nameOrig » et « nameDest » les noms de compte par la lettre C ou M selon le compte qu'il possède pour pouvoir intervenir dessus plus facilement comme le montre l'aperçu ci-dessous :

| transactionId | step | | type | amount | nameOrig | oldbalanceOrig | newbalanceOrig | nameDest | oldbalanceDest | newbalanceDest | isFraud |
|---------------|------|---|----------|----------|----------|----------------|----------------|----------|----------------|----------------|---------|
| 0 | 0 | 1 | PAYMENT | 9839.64 | C | 170136.00 | 160296.36 | M | 0.00 | 0.00 | 0 |
| 1 | 1 | 1 | PAYMENT | 1864.28 | C | 21249.00 | 19384.72 | M | 0.00 | 0.00 | 0 |
| 2 | 2 | 1 | TRANSFER | 181.00 | C | 181.00 | 0.00 | C | 0.00 | 0.00 | 0 |
| 3 | 3 | 1 | CASH_OUT | 181.00 | C | 181.00 | 0.00 | C | 21182.00 | 0.00 | 0 |
| 4 | 4 | 1 | PAYMENT | 11668.14 | C | 41554.00 | 29885.86 | M | 0.00 | 0.00 | 0 |

Ensuite nous modifierons ces mêmes valeurs que nous avons remplacées précédemment en même temps que les 5 classe de la colonne « type » (« PAYMENT », « TRANSFER », « CASH-OUT », « CASH-IN », « DEBIT ») pour les afficher selon un code numérique que nous aurons prédéfini :

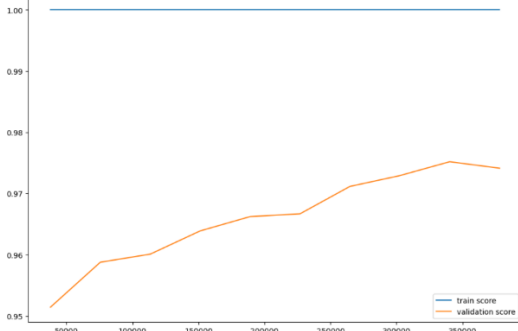
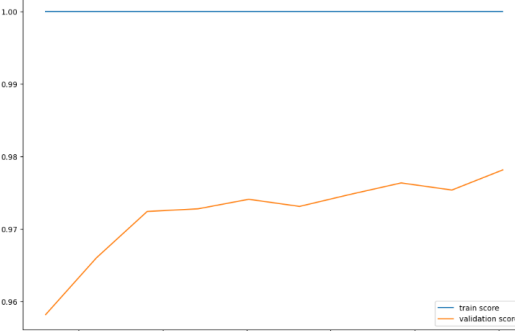
| transactionId | step | type | type | amount | nameOrig | oldbalanceOrig | newbalanceOrig | nameDest | oldbalanceDest | newbalanceDest | isFraud |
|---------------|------|------|------|----------|----------|----------------|----------------|----------|----------------|----------------|---------|
| 0 | 0 | 1 | 1 | 9839.64 | 1 | 170136.00 | 160296.36 | 2 | 0.00 | 0.00 | 0 |
| 1 | 1 | 1 | 1 | 1864.28 | 1 | 21249.00 | 19384.72 | 2 | 0.00 | 0.00 | 0 |
| 2 | 2 | 1 | 2 | 181.00 | 1 | 181.00 | 0.00 | 1 | 0.00 | 0.00 | 1 |
| 3 | 3 | 1 | 3 | 181.00 | 1 | 181.00 | 0.00 | 1 | 21182.00 | 0.00 | 1 |
| 4 | 4 | 1 | 1 | 11668.14 | 1 | 41554.00 | 29885.86 | 2 | 0.00 | 0.00 | 0 |

Désormais, notre Dataset ne se compose plus que de variables numériques qui n'empêcherons pas l'avancement aux étapes de tests modèle sur l'ensemble de nos données.

```
int64      6
float64    5
dtype: int64
```

Nous pouvons désormais commencer avec une première conception de données sans ajustement ou rééquilibrage. On découpe le Dataset avec 20% des données mis à l'écart pour la partie test et le restant pour l'entraînement.

Voici les premiers modèles sans réajustement de nos données déséquilibrées :

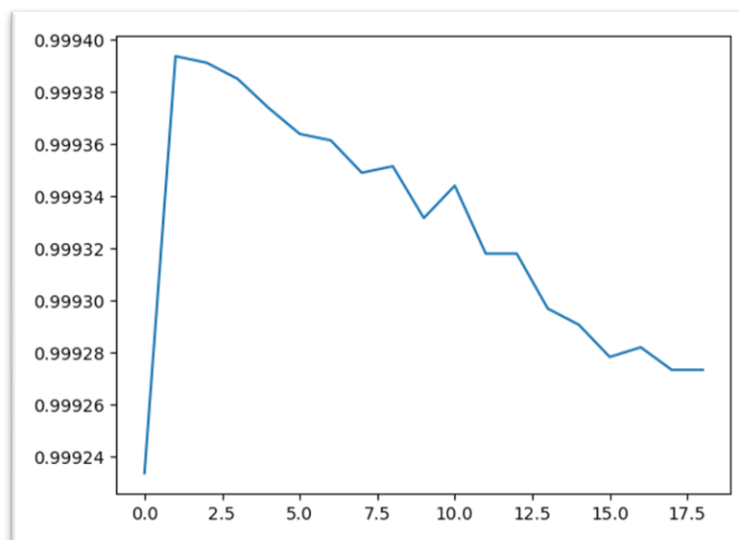
| Modèle 1 Données brutes | | | | | Modèle 2 Données brutes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------------------|---|--------|----------|---------|---|-----------|--------|----------|---------|---|------|------|------|--------|---|------|------|------|------|----------|--|--|------|--------|-----------|------|------|------|--------|--------------|------|------|------|--------|--|--|--|--|--|-----------|--------|----------|---------|---|------|------|------|--------|---|------|------|------|------|----------|--|--|------|--------|-----------|------|------|------|--------|--------------|------|------|------|--------|
| Factures | 9 | | | | 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Target | isFraud | | | | Is Fraud | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Nombre de Target | Non-fraude=1000000 Fraude=8213 | | | | Non-fraude=1000000 Fraude=8213 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Paramètres | <div>X_train=50%</div> <div>X_test, y_test = preprocessing(testset)</div> <div>0 500059 1 4048 Name: isFraud, dtype: int64</div> <div>y_train=50%</div> <div>X_train, y_train = preprocessing(trainset)</div> <div>0 499941 1 4165 Name: isFraud, dtype: int64</div> | | | | <div>X_train = 80%</div> <div>0 799970 1 6600 Name: isFraud, dtype: int64</div> <div>Y_train = 20%</div> <div>0 200030 1 1613 Name: isFraud, dtype: int64</div> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Hyperparamètres | CV = 4, scoring=f1, train_sizes=0,1, 1, 10 | | | | CV = 4, scoring=f1, train_sizes=0,1, 1, 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Train score & Validation score |  | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Evaluation | <div>[[499954 105] [86 3962]]</div> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>500059</td></tr><tr><td>1</td><td>0.97</td><td>0.98</td><td>0.98</td><td>4048</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>504107</td></tr><tr><td>macro avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>504107</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>504107</td></tr></tbody></table> | | | | | precision | recall | f1-score | support | 0 | 1.00 | 1.00 | 1.00 | 500059 | 1 | 0.97 | 0.98 | 0.98 | 4048 | accuracy | | | 1.00 | 504107 | macro avg | 0.99 | 0.99 | 0.99 | 504107 | weighted avg | 1.00 | 1.00 | 1.00 | 504107 | <div>[[200001 29] [29 1584]]</div> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>1.00</td><td>1.00</td><td>1.00</td><td>200030</td></tr><tr><td>1</td><td>0.98</td><td>0.98</td><td>0.98</td><td>1613</td></tr><tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>201643</td></tr><tr><td>macro avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>201643</td></tr><tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>201643</td></tr></tbody></table> | | | | | precision | recall | f1-score | support | 0 | 1.00 | 1.00 | 1.00 | 200030 | 1 | 0.98 | 0.98 | 0.98 | 1613 | accuracy | | | 1.00 | 201643 | macro avg | 0.99 | 0.99 | 0.99 | 201643 | weighted avg | 1.00 | 1.00 | 1.00 | 201643 |
| | precision | recall | f1-score | support | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1.00 | 1.00 | 1.00 | 500059 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0.97 | 0.98 | 0.98 | 4048 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| accuracy | | | 1.00 | 504107 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| macro avg | 0.99 | 0.99 | 0.99 | 504107 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| weighted avg | 1.00 | 1.00 | 1.00 | 504107 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | precision | recall | f1-score | support | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1.00 | 1.00 | 1.00 | 200030 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0.98 | 0.98 | 0.98 | 1613 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| accuracy | | | 1.00 | 201643 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| macro avg | 0.99 | 0.99 | 0.99 | 201643 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| weighted avg | 1.00 | 1.00 | 1.00 | 201643 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Conclusion | Pas fiable | | | | Pas fiable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

A l'aide de la méthode des k plus proche voisin, on obtient pour les paramètres suivants, un best score d'environ 0.999393 obtenue pour les best paramètres qui suivent.

```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'metric': ['euclidean', 'manhattan'],
                         'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,
 8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19])})
```

```
{'metric': 'euclidean', 'n_neighbors': 2}
```

Ce résultat est confirmé par la courbe ou on remarque une chute graduelle d'efficacité au-delà de deux plus proches voisins :



Cependant la matrice de confusion obtenue ci-contre n'est toujours pas à la hauteur de nos espérances.

```
array([[200023,    7],
       [    85,  1528]], dtype=int64)
```

- Pour les modèles qui suivent, nous ajusteront les données en augmentant les variables minoritaires dans un premier temps tout en ne gardant que les colonnes nécessaires à la détection (step, type, amount et isFraud). On monte à un total de 2 millions de données avec le rééquilibrage soit 50% de données en situation de fraude et 50% en non-fraude.

| | step | type | amount | isFraud |
|--------------------------|------|------|------------|----------|
| | 0 | 1 | 1 | 9839.64 |
| | 1 | 1 | 1 | 1864.28 |
| | 4 | 1 | 1 | 11668.14 |
| | 5 | 1 | 1 | 7817.71 |
| | 6 | 1 | 1 | 7107.77 |
| ... | ... | ... | ... | ... |
| 1005553 | 499 | 2 | 116530.20 | 1 |
| 1004648 | 412 | 3 | 2425164.58 | 1 |
| 1002126 | 187 | 2 | 856386.66 | 1 |
| 1002547 | 227 | 2 | 1162155.11 | 1 |
| 1007092 | 643 | 3 | 352179.76 | 1 |
| 1000000 rows x 4 columns | | | | |

| | |
|-----------------------------|---------|
| 0 | 1000000 |
| 1 | 1000000 |
| Name: isFraud, dtype: int64 | |

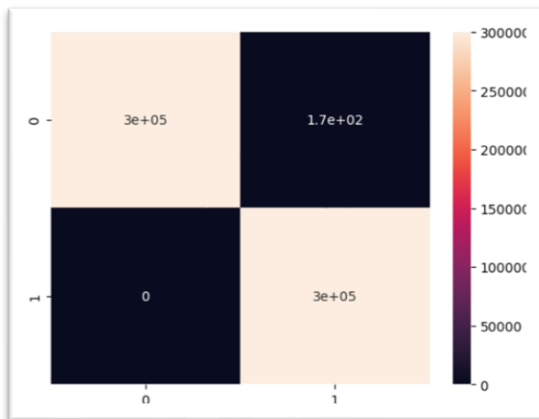
Notre trainset contiendra ces données en guise de données fraude représenté par 1 et données non-fraude représentées par 0 :

| | |
|-----------------------------|--------|
| 1 | 700248 |
| 0 | 699752 |
| Name: isFraud, dtype: int64 | |

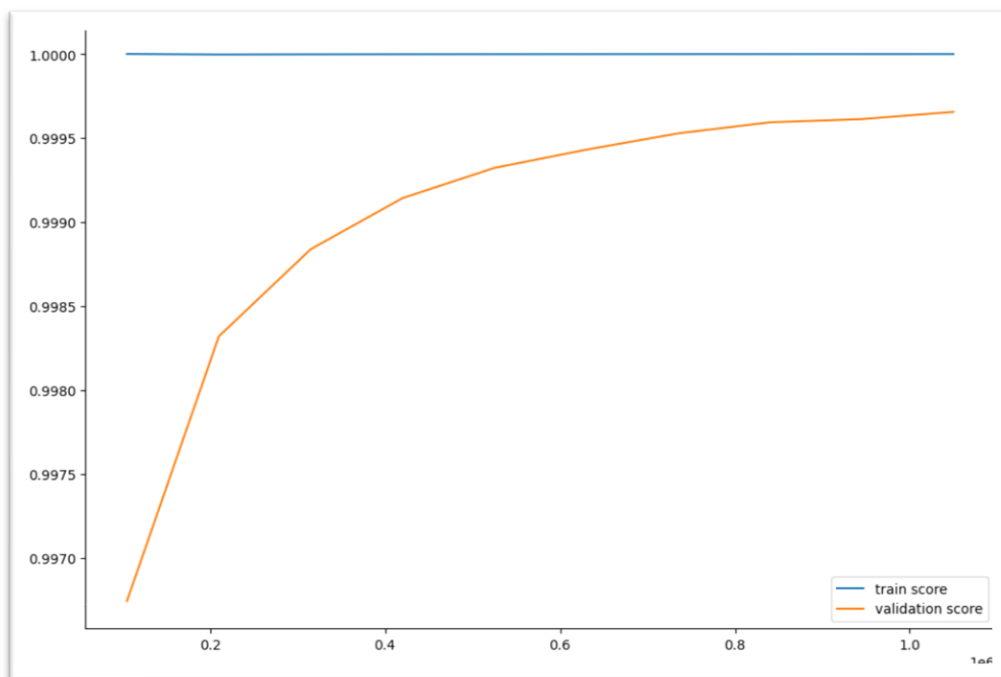
Le testset lui en revanche contiendra les données suivantes :

| | |
|-----------------------------|--------|
| 0 | 300248 |
| 1 | 299752 |
| Name: isFraud, dtype: int64 | |

L'entraînement du modèle avec les hyperparamètre de base donnent :



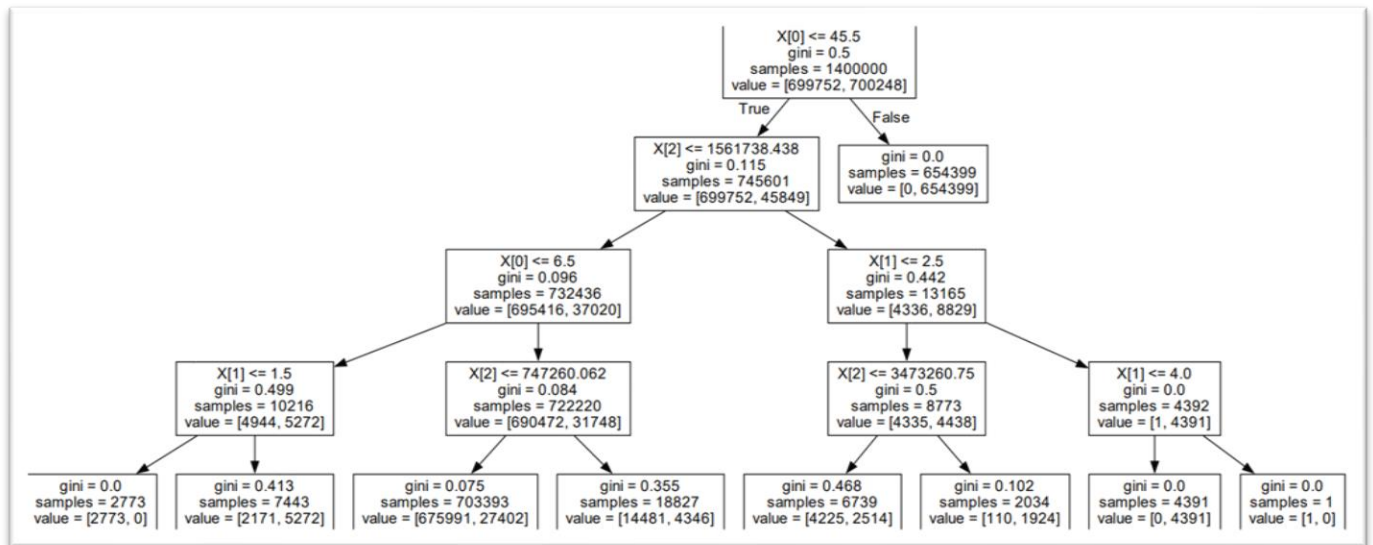
| | | | | | |
|--|--------------|-----------|--------|----------|---------|
| | | precision | recall | f1-score | support |
| | 0 | 1.00 | 1.00 | 1.00 | 300248 |
| | 1 | 1.00 | 1.00 | 1.00 | 299752 |
| | accuracy | | | 1.00 | 600000 |
| | macro avg | 1.00 | 1.00 | 1.00 | 600000 |
| | weighted avg | 1.00 | 1.00 | 1.00 | 600000 |



Les résultats obtenus sont plutôt satisfaisants bien notre train score donne l'impression d'être en overfitting, il ne l'est pas. Désormais si l'on teste les données du testset dans notre modèle, on obtient un score de :

0.9744416666666667

On génère son arbre de décision pour obtenir le modèle qui suit sur une profondeur maximum de 4 :



- Le modèle précédent semble raver, on décidera tout de même de procéder à la création d'un autre modèle en réduisant les données de la classe majoritaire afin de pouvoir comparer les résultats sur les mêmes features.

Après applications des modifications, nos données ne se composent plus que de 16426 enregistrements :

16426 rows × 4 columns

Dont 50% de données sont des fraudes et 50% des non-fraudes :

```

1    0.5
0    0.5
Name: isFraud, dtype: float64
  
```

Pour ce modèle, on découpera nos données en gardant 30% pour le testset et le restant pour l'entraînement :

Trainset :

```

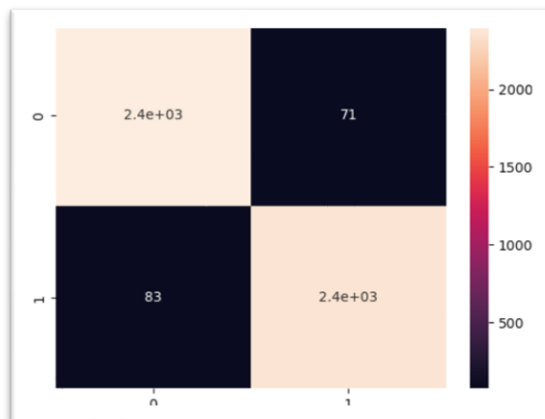
1    5775
0    5723
Name: isFraud, dtype: int64
  
```

Testset :

```

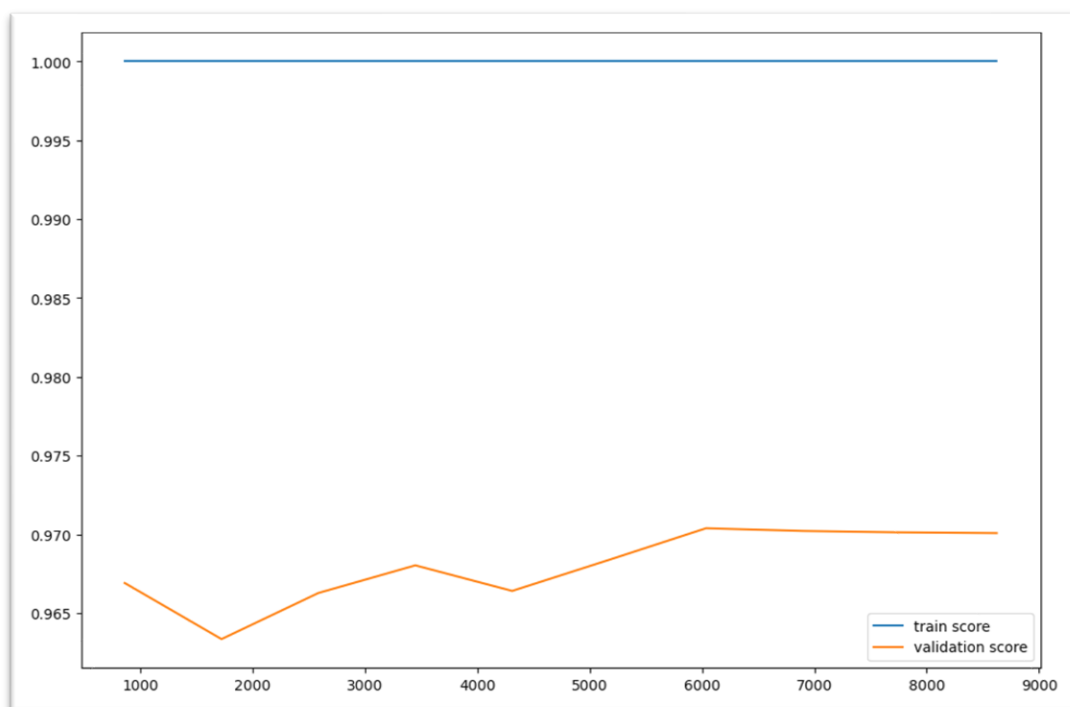
0    2490
1    2438
Name: isFraud, dtype: int64
  
```

On obtient après compilation du modèles, la table d'évaluation et sa matrice comme suit :



| | | | | | |
|--|--------------|-----------|--------|----------|---------|
| | [[2419 71] | | | | |
| | [83 2355]] | | | | |
| | | precision | recall | f1-score | support |
| | 0 | 0.97 | 0.97 | 0.97 | 2490 |
| | 1 | 0.97 | 0.97 | 0.97 | 2438 |
| | accuracy | | | 0.97 | 4928 |
| | macro avg | 0.97 | 0.97 | 0.97 | 4928 |
| | weighted avg | 0.97 | 0.97 | 0.97 | 4928 |

Sa courbe d'évolution :



Il apparait clairement ici que dans ce modèle donne malgré ses performances, des cas ou il ne détecterait pas des fraudes qu'il n'aurait pas dû laisser passer, ce qui serait embêtant pour le concerner ainsi.

- En gardant le même ajustement de données et découpage comme ci-dessus, on va tenter de construire un modèle qui en concevant beaucoup plus de features que passant de 4 à 10 (un drop que sur la colonne transactionID), à savoir step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest et isFraud :

| | step | type | amount | nameOrig | oldbalanceOrg | newbalanceOrig | nameDest | oldbalanceDest | newbalanceDest | isFraud |
|---|------|------|----------|----------|---------------|----------------|----------|----------------|----------------|---------|
| 0 | 1 | 1 | 9839.64 | C | 170136.00 | 160296.36 | M | 0.00 | 0.00 | 0 |
| 1 | 1 | 1 | 1864.28 | C | 21249.00 | 19384.72 | M | 0.00 | 0.00 | 0 |
| 2 | 1 | 2 | 181.00 | C | 181.00 | 0.00 | C | 0.00 | 0.00 | 1 |
| 3 | 1 | 3 | 181.00 | C | 181.00 | 0.00 | C | 21182.00 | 0.00 | 1 |
| 4 | 1 | 1 | 11668.14 | C | 41554.00 | 29885.86 | M | 0.00 | 0.00 | 0 |

Avec l'entraînement du modèle contenant les mêmes paramètres que précédemment, on obtient la table d'évaluation et sa courbe suivante :

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.99 | 0.99 | 0.99 | 2449 |
| 1 | 0.99 | 0.99 | 0.99 | 2479 |
| accuracy | | | 0.99 | 4928 |
| macro avg | 0.99 | 0.99 | 0.99 | 4928 |
| weighted avg | 0.99 | 0.99 | 0.99 | 4928 |



Le modèle ici nous prouve encore une fois que les modèles générés par l'arbre de décision sont les plus performants, cependant, ce dernier obtenu ne se place pas parmi les meilleurs de sa catégorie du fait du nombre de faux positifs dans les données de fraude qu'il ne traite pas correctement.

e- Troisième modèle de classification : Le Random Forest

L'algorithme de classification Random Forest est l'un des algorithmes de Machine Learning que nous avons retenu dans le but de déterminer lequel serait le plus adapter pour répondre à notre problème de détection de fraudes bancaires.

La première étape de notre évaluation du modèle Random Forest pour la recherche de fraudes bancaires a été de l'adapter aux données de notre fichier csv sans que celle-ci ne soit modifiées ou altérer. A la suite de ce test pour lequel nous avons demandé à l'ordinateur de prendre 30% des données du dataset pour tester le modèle et 70% des données pour entrainer le modèle, nous avons obtenu la matrice de confusion et le rapport de classification suivant :

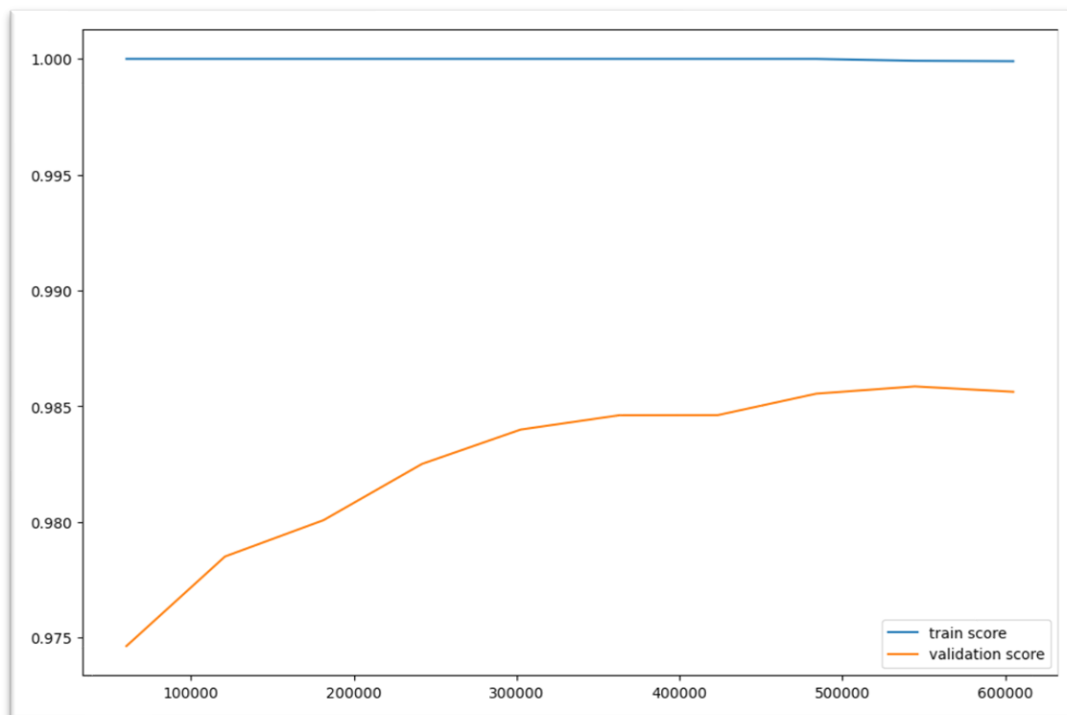
```
[[200059      2]
 [    44  1538]]
```

| | | precision | recall | f1-score | support |
|--------------|---|-----------|--------|----------|---------|
| | 0 | 1.00 | 1.00 | 1.00 | 200061 |
| | 1 | 1.00 | 0.97 | 0.99 | 1582 |
| accuracy | | | | 1.00 | 201643 |
| macro avg | | 1.00 | 0.99 | 0.99 | 201643 |
| weighted avg | | 1.00 | 1.00 | 1.00 | 201643 |

On pourrait penser en lisant le rapport de classification ci-dessus que notre modèle est surentrainé car il possède une précision de 1 ou 100% aussi bien lors de la détection des fraudes que des non-fraudes ainsi qu'un rappel de 100% des non-fraudes mais cela n'est pas le cas lorsque l'on analyse les résultats de la fonction `cross_val_score` de la bibliothèque `scikit-learn` car on constate que l'ordinateur a arrondi les résultats à deux chiffres après la virgule et donc à 1 car 0.999 est plus proche de 1 que 0.99.

```
Scores: [0.99970928 0.99976914 0.99983754 0.99965798 0.99976914]
Accuracy: 0.99975 (+/- 0.00012)
```

Finalement, lorsque l'on affiche la courbe d'apprentissage, qui est basé sur le f1-score (agrégat de la précision et du rappel), du modèle Random Forest, on peut voir que celle-ci est une parabole qui tend vers 1 sans pour autant l'atteindre ce qui nous indique bien que notre modèle n'est pas surentrainé.



Sur la courbe ci-dessus, l'axe des abscises représente le nombre de lignes, ou entrées, dans notre dataset et l'axe des ordonnées représente la probabilité que l'algorithme les identifie correctement comme des fraudes ou des non-fraudes sur une échelle allant de zéro à un.

La droite bleue représente les résultats attendus de l'algorithme selon les données d'entraînement qui lui ont été donné tandis que la courbe orange représente les résultats obtenus lors de la phase de test du modèle. Ces deux courbes se basent sur le f1-score qui l'agrégat de la précision, la capacité du modèle à prédire correctement les étiquettes des données d'entraînement qui sont dans notre cas fraude et non-fraude, et le rappel, la capacité du modèle à détecter correctement tous les cas positifs dans un jeu de données.

La seconde étape de notre évaluation du modèle Random Forest pour la recherche de fraudes bancaires a été d'appliquer la méthode SMOTE à nos données afin de rééquilibrer le nombre de fraudes et de non-fraudes dans notre Dataset afin d'améliorer les résultats de notre modèle.

```
[[419691 138]
 [ 368 419803]]
      precision    recall  f1-score   support

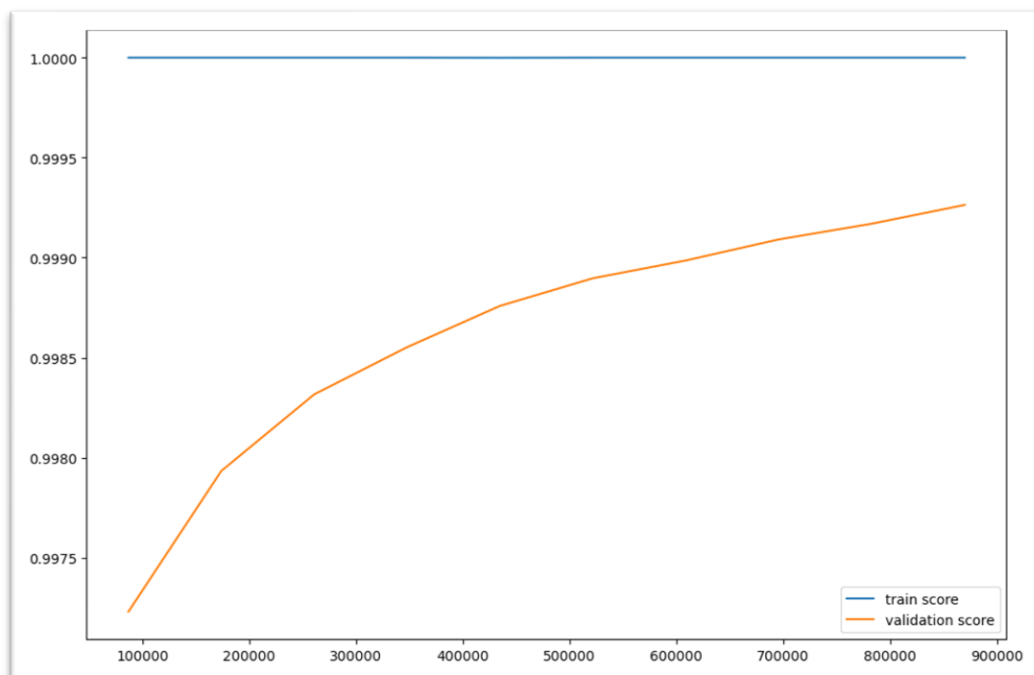
      0       1.00      1.00      1.00     419829
      1       1.00      1.00      1.00     420171

 accuracy          1.00          1.00          1.00     840000
 macro avg          1.00          1.00          1.00     840000
weighted avg          1.00          1.00          1.00     840000
```

Comme avec la première étape de notre évaluation, on obtient un rapport de classification où les valeurs de la précision, du rappel, et du f1-score (agrégat de la précision et du rappel) qui sont égal à 1 ce qui peut encore une fois faire penser que le modèle est surentrainé mais lorsque l'on effectue une validation croisée, on peut constater que l'ordinateur a arrondi les valeurs à deux chiffres après la virgule lors de l'affichage du rapport de classification.

```
Scores: [0.99935776 0.99937069 0.999375 0.9992069 0.99930172]
Accuracy: 1.00 (+/- 0.00)
```

Finalement, lorsque l'on affiche la courbe d'apprentissage du modèle Random Forest, on peut voir que celle-ci est une parabole qui tend vers 1 sans pour autant l'atteindre ce qui nous indique bien que notre modèle n'est pas surentrainé.



Comme pour la courbe précédente, l'axe des abscisses représente le nombre de lignes, ou entrées, dans notre dataset et l'axe des ordonnées représente la probabilité que l'algorithme identifie correctement les données de fraudes ou des non-fraudes sur une échelle allant de zéro à un.

On peut constater en analysant cette courbe que l'utilisation de la méthode SMOTE a permis à notre modèle Random Forest de gagner en performance au niveau de la détection des fraudes comparé à nos précédents tests ce qui le rend plus fiable.

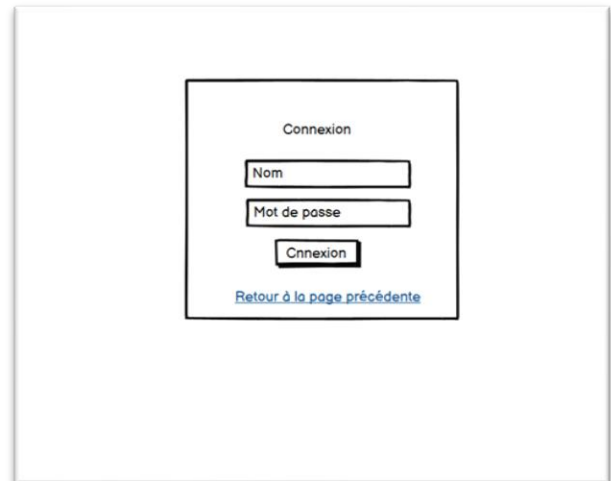
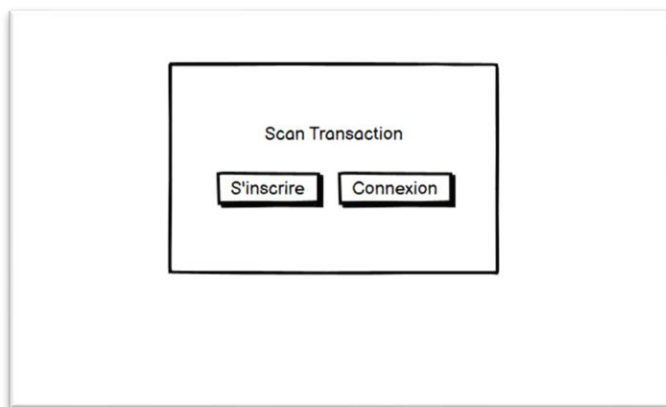
| Test / Essai de détection de fraudes | Modèle de Machine Learning utilisé | Nombre de features utilisées | Features utilisées | Target(s) retenue(s) | Nombre d'éléments dans le dataset utilisé | SMOTE ? | Vrai Positif | Faux Négatif | Vrai Négatif | Faux Positif | accuracy | recall | f1-score | final accuracy |
|--------------------------------------|------------------------------------|------------------------------|---|----------------------|--|---------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|----------------|
| 1 | Random Forest | 8 | transactionId, step, type, amount, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest | isFraud | Non-fraudes = 1000000 Fraudes = 8213 | non | 420032 | 5 | 3321 | 92 | 1,00 1,00 | 1,00 0,97 | 1,00 0,99 | 0,99932 |
| 2 | Random Forest | 8 | transactionId, step, type, amount, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest | isFraud | Non-fraudes = 8213 Fraudes = 8213 | non | 1625 | 5 | 1636 | 20 | 0,99 1,00 | 1,00 0,99 | 0,99 0,99 | 0,99224 |
| 3 | Random Forest | 3 | step, type, amount | isFraud | Non-fraudes = 8213 Fraudes = 8213 | non | 1621 | 28 | 1597 | 40 | 0,98 0,98 | 0,98 0,98 | 0,98 0,98 | 0,97511 |
| 4 | Random Forest | 8 | transactionId, step, type, amount, oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest | isFraud | Non-fraudes = 1000000 Fraudes = 1000000 | oui | 419820 | 126 | 419723 | 331 | 1,00 1,00 | 1,00 1,00 | 1,00 1,00 | 0,99975 |
| 5 | Random Forest | 3 | step, type, amount | isFraud | Non-fraudes = 1000000 Fraudes = 8213 | non | 419954 | 64 | 3257 | 175 | 1,00 0,98 | 1,00 0,95 | 1,00 0,96 | 0,99946 |
| 6 | Random Forest | 3 | step, type, amount | isFraud | Non-fraudes = 1000000 Fraudes = 1000000 | oui | 417664 | 2160 | 418633 | 1543 | 1,00 0,99 | 1,00 1,00 | 1,00 1,00 | 0,99528 |

En conclusion, on peut dire que l'algorithme de classification Random Forest peut être utilisé pour faire de la détection de fraudes bancaire, qu'il est capable de les détecter avec une grande fiabilité aussi bien lorsque les données sont déséquilibrées que lorsque celle-ci sont équilibrées, et qu'il obtient, après comparaison des différents entraînements et tests présentant dans le tableau (annexe 1) ci-dessus, les meilleurs résultats lorsque les données ont été rééquilibrées en utilisant la méthode SMOTE.

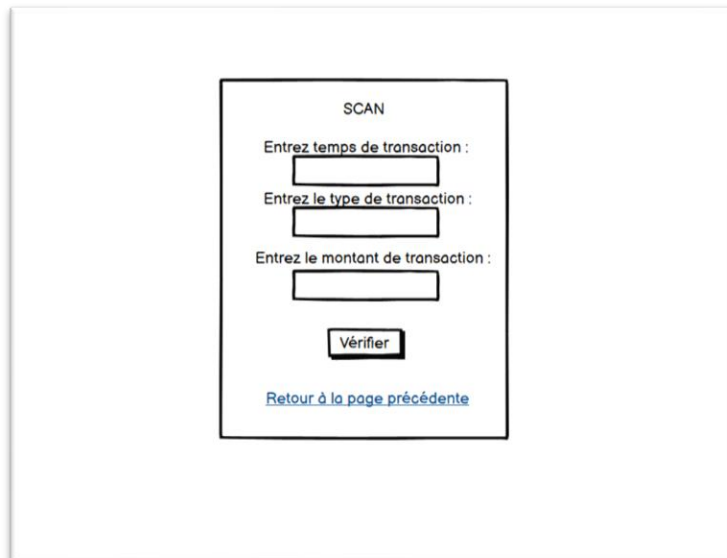
2- Interface

a. Maquette

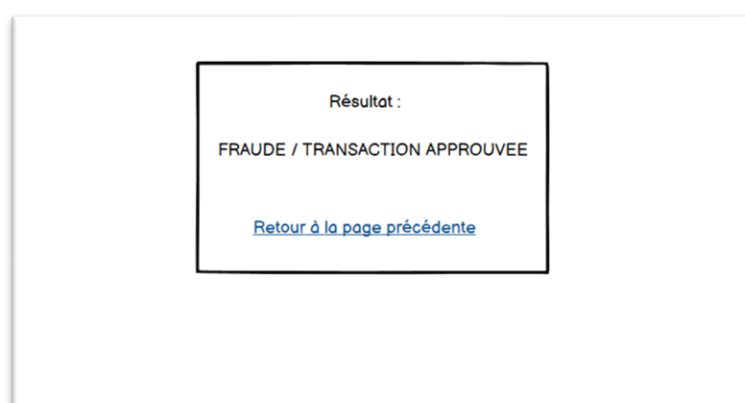
Pour la réalisation de la maquette nous avons utilisé Wireframe Balsamiq. On s'est appuyé sur le cahier de charge pour déterminer les points essentiels qui sont nécessaire d'être affichés dans notre application. Donc sur la page d'accueil on peut trouver le moyen de s'inscrire et/ou de se connecter si nos identifiants sont déjà dans la base de données de l'application.



Si l'utilisateur est reconnu par l'application, il sera redirigé vers la page de Scan, qui permet de rentrer les valeurs nécessaires pour vérifier une transaction en question. Après avoir appuyé sur le bouton "Vérifier" on est redirigé vers la dernière de prédiction, qui affichera si la transaction saisie est une fraude ou pas.



The screenshot shows a web form titled "SCAN". It contains three input fields with labels: "Entrez temps de transaction :", "Entrez le type de transaction :", and "Entrez le montant de transaction :". Below these fields is a button labeled "Vérifier". At the bottom of the form is a blue link that says "Retour à la page précédente".



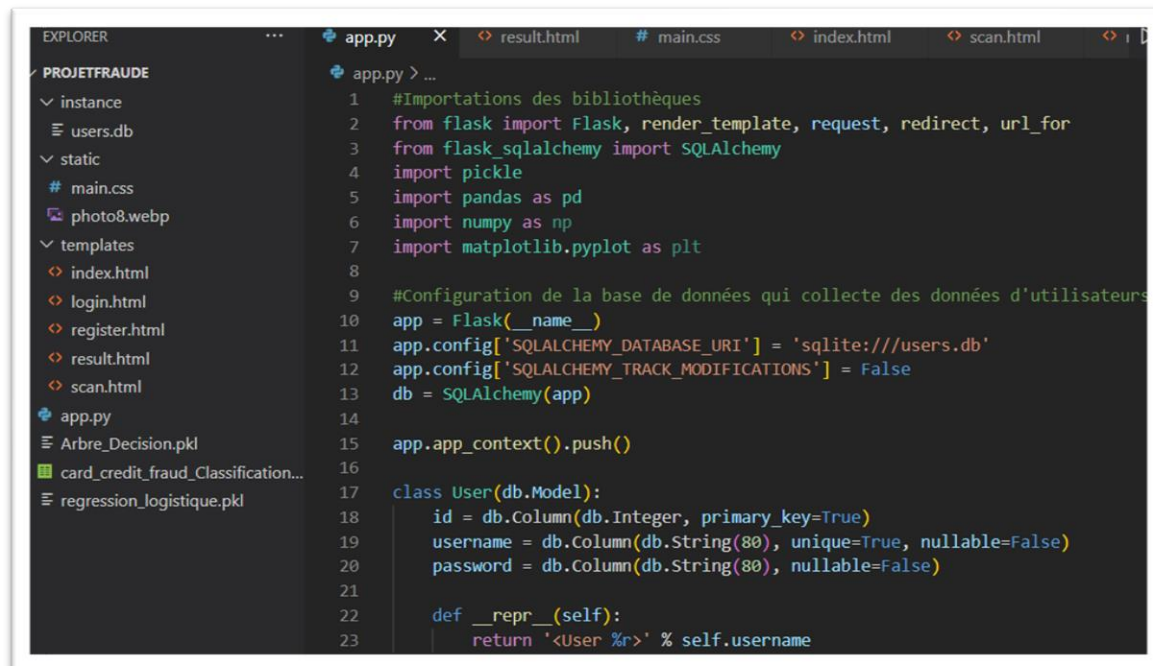
The screenshot shows a web page with a box containing the text "Résultat : FRAUDE / TRANSACTION APPROUVEE". Below this box is a blue link that says "Retour à la page précédente".

b. Interface Graphique pour la prédiction

Pour la réalisation de l'interface de l'application nous avons utilisé le micro-Framework Flask, Python, HTML et CSS.

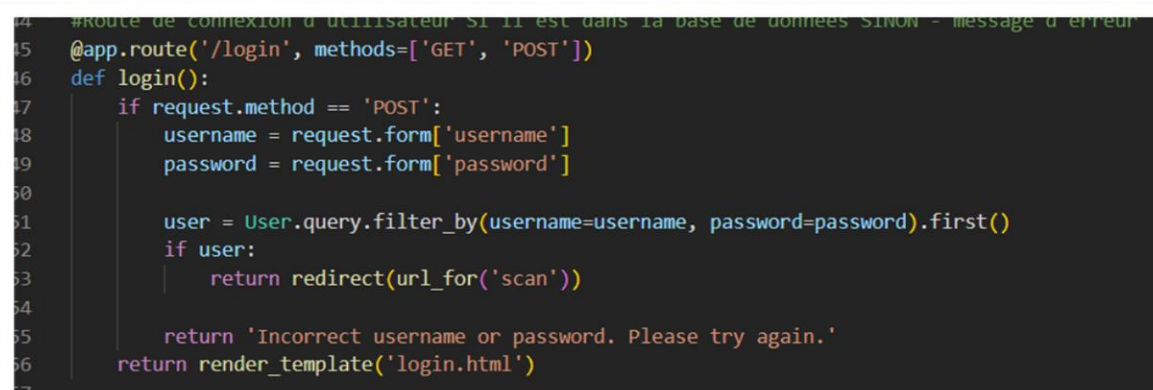
Arborescence de nos fichiers du projet on a commencé par la création du fichier app.py, le dossier templates qui contient tous les fichiers d'affichage des pages en .html, puis on a créé le dossier static qui contient un fichier main.css pour ajuster notre application selon la maquette réalisée.

Dès la première inscription d'utilisateur, un dossier instance a été créé avec SQLAlchemy – qui contient les données d'utilisateur inscrits.



```
1 #Importations des bibliothèques
2 from flask import Flask, render_template, request, redirect, url_for
3 from flask_sqlalchemy import SQLAlchemy
4 import pickle
5 import pandas as pd
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 #Configuration de la base de données qui collecte des données d'utilisateurs
10 app = Flask(__name__)
11 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
12 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
13 db = SQLAlchemy(app)
14
15 app.app_context().push()
16
17 class User(db.Model):
18     id = db.Column(db.Integer, primary_key=True)
19     username = db.Column(db.String(80), unique=True, nullable=False)
20     password = db.Column(db.String(80), nullable=False)
21
22     def __repr__(self):
23         return '<User %r>' % self.username
```

Dans notre code Flask, les méthodes GET et POST sont spécifiées dans la déclaration de la fonction de route pour indiquer les types de requêtes HTTP acceptées par cette fonction. Par exemple, dans le code suivant, la fonction login accepte les requêtes GET et POST



```
44 #route de connexion d'utilisateur SI il est dans la base de données SINON - message d'erreur
45 @app.route('/login', methods=['GET', 'POST'])
46 def login():
47     if request.method == 'POST':
48         username = request.form['username']
49         password = request.form['password']
50
51         user = User.query.filter_by(username=username, password=password).first()
52         if user:
53             return redirect(url_for('scan'))
54
55         return 'Incorrect username or password. Please try again.'
56     return render_template('login.html')
```

Ci-dessus nous pouvons voir que si utilisateur est reconnu par notre base de données il sera redirigé vers la page Scan :

if user:
Return redirect(url_for('scan'))

Pour le chargement du modèle IA on a utilisé le format Pickle .pkl, en utilisant la méthode **POST**. L'application ira chercher les données rentrées par utilisateur qui seront ensuite évaluées par notre modèle pour afficher la prédiction :

```
1 # Chargement du modèle IA en .pkl - liason avec le formulaire SCAN
2 model = pickle.load(open("Arbre_Decision.pkl", "rb"))
3
4 @app.route("/scan", methods=["POST"])
5 def scanForm():
6     temps = request.form["input1"]
7     transaction = request.form["input2"]
8     montant = request.form["input3"]
9     data_predict = pd.DataFrame(columns=["step", "type", "amount"], data=[[temps, transaction, montant]])
10
```

Pour faciliter le stockage et la manipulation des données entre le modèle et le formulaire nous avons utilisée Data.Frame.
Notre modèle retourne la valeur 0 en cas de non fraude et 1 en cas de fraude. Afin d'avoir un résultat plus parlant pour l'utilisateur on utilise une boucle if/else pour afficher les messages souhaités.

```
1 # Utiliser les données collectées pour faire des prédictions avec le modèle
2 prediction = model.predict(data_predict)
3
4 if prediction == 1:
5     message = "Fraude"
6     css_class = "fraude"
7 else:
8     message = "Transaction approuvée"
9     css_class = "non-fraude"
10
11 return render_template("result.html", message=message, css_class=css_class)
```

L'utilisateur est redirigé vers la page de résultat :

```
1 <div class="card" style="width: 25rem;">
2   <div class="card-body">
3     <div class="result">
4       <h2>Résultat :</h2><br/>
5       <h1 class="{{ css_class }}">{{ message }}</h1><br/>
6
7       <a href="javascript:history.back()">Retour à la page précédente </a><br/><br/><br/>
8     </div>
9   </div>
10 </div>
```

Il faut noter, que l'utilisateur de l'application a la possibilité de retourner vers la page précédente grâce à un lien de retour à la page précédente. Ce que lui permettra de réaliser une autre prédiction.

Difficultés rencontrées

- Distinguer un bon entraînement d'un overfitting
- Le manque de puissance de calcul pour pouvoir optimiser les modèles avec la fonction `GridSearchCV` de la bibliothèque `scikit-learn` (`sklearn`) de python. (Exemple : ~114 minutes d'attente pour avoir des résultats avec Random Forest)
- Comprendre les hyperparamètres des algorithmes.
- Faire interagir le modèle de prédiction avec le formulaire créé pour utilisateur

Perspectives d'évolution

- Mise en place d'une base de données des recherche effectuées
- Mettre des inputs en mode radio pour limiter les valeurs qui peuvent être entrées
- Tester d'autres algorithmes avec d'autres hyperparamètres
- Optimiser les algorithmes avec les hyperparamètres récupérer à l'aide de la fonction GridSearchCV

Conclusion

Nous avons effectué une étude merise du projet et une BDD relationnelle, créé un dictionnaire de données, un modèle conceptuel de données (MCD/MLD), implémenté l'IA en utilisant différents algorithmes de classification tels que la régression logistique, l'arbre de décision et RandomForest, effectué des évaluations en utilisant GridSearch, prédit les fraudes à l'aide du meilleur modèle et enfin développé une interface et une maquette ainsi qu'une interface graphique pour la prédiction.

Notre travail a abouti à la création d'une application fonctionnelle qui peut détecter efficacement les fraudes de transactions en utilisant des techniques d'IA avancées. Nous avons évalué et comparés différents algorithmes de classification. Nous avons optimisé les hyperparamètres en utilisant GridSearch.

Nous avons mis en place d'une interface graphique conviviales pour la prédiction de fraudes, intuitive et facile à utiliser.
En somme, nous avons mené à bien ce projet et développé une application fonctionnelle et utile pour détecter les fraudes de transactions.

Bilan de groupe

Ce projet nous a permis de conjuguer modèle d'intelligence artificielle et application web. L'ensemble des membres de l'équipe a pu apprendre et évoluer techniquement.

On a manqué de rigueur dans la structuration du suivi de nos résultats pour la recherche du meilleur algorithme IA.

Bonne entente et bonne collaboration entre les membres de l'équipe.

Bilans personnels

Aude BOUCHONNET

J'ai pu mettre un peu plus en application fask sur la partie liaison entre le modèle et l'interface. J'ai pu tester différents modèles. Je pense avoir pu progresser et mieux comprendre qu'est-ce que je devais faire pour m'améliorer. Etre plus stratégique, augmenter l'automatisation.

Tetyana TARASENKO

Dans le cadre du projet de prédiction des fraudes sur les transactions, j'ai participé en tant que concepteur d'interface et de maquette. Mon objectif principal était de créer une interface conviviale pour l'utilisateur final, qui permettrait de visualiser les prédictions de fraude de manière claire et efficace. Pour ce faire, j'ai travaillé en étroite collaboration avec l'équipe pour déterminer les fonctionnalités nécessaires et les critères d'interface les plus pertinents pour ce type d'application. J'ai également utilisé des techniques de design d'interface telles que la mise en page, la sélection de couleurs et de polices pour créer une expérience visuelle cohérente pour l'utilisateur.

Le résultat final était une interface intuitive qui permettait aux utilisateurs de voir les prédictions de fraude. Les commentaires reçus jusqu'à présent ont été positifs et l'interface a été appréciée pour sa simplicité et sa facilité d'utilisation.

J'ai également fait un essai pour la création du modèle IA de prédiction, notamment celle de la Régression Logistique, mais les résultats ont montré qu'elle n'est pas très pertinente dans ses prédictions.

En général, j'ai été très heureuse de participer à ce projet et de contribuer à la création d'une application utile pour les entreprises pour lutter contre la fraude sur les transactions.

Briand BAKOUZOU

Ce projet d'études m'a permis

Osman ARSLAN

J'ai pu mettre en application le modèle ID3 sur le dataset des fraudes bancaires. J'ai pu tester différents modèles. J'ai appris que pour avoir un bon modèle il faut équilibrer les données avec différentes méthodes. Je dois continuer à explorer les ressources pour m'améliorer sur ce sujet, il y'a pas mal de chose à apprendre sur l'algorithme ID3.

Axel ARCIDIACO

J'ai pu mettre en application le modèle Random Forest sur le dataset des fraudes bancaires et ainsi voir s'il était capable de bien identifier les fraudes bancaires ainsi que comment la taille du dataset et la proportion de fraudes parmi les non-fraudes affectait les résultats de l'algorithme en utilisant la méthode SMOTE pour augmenter le nombre de fraudes pour qu'il y en est autant que de non-fraudes ou en réduisant le nombre de non-fraudes (1000000) dans le dataset pour qu'il soit égal à celui des fraudes (8213).