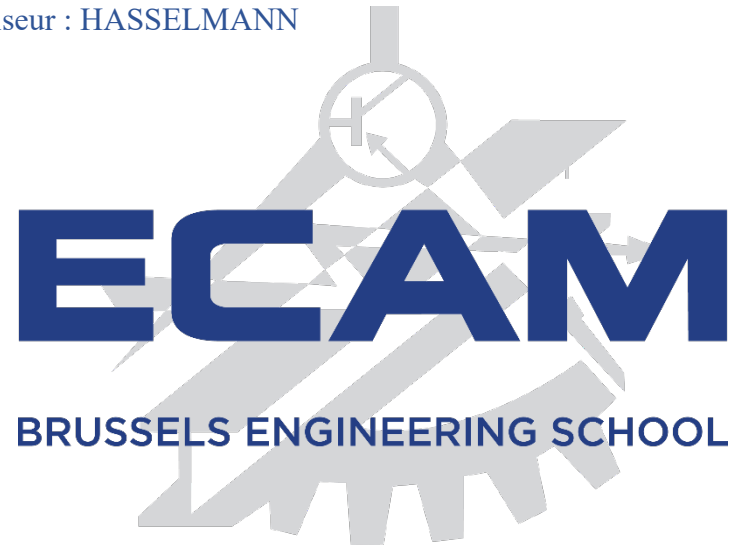
ECAM

Rapport AI : Détection de Fraudes

Superviseur : HASSELMANN



Dawid Krasowski 20194 & Matthias Léonard 20308
2022/2023

I. Introduction

Ce document a pour but de documenter nos démarches et réflexions dans le cadre des laboratoires d'intelligence artificielle, dispensée à l'ECAM en 2ème Master en ingénieur informatique, sous la supervision de Monsieur Ken HASSELMANN. Nous réalisons un projet de recherche de fraude à la carte bancaire.
Un jeu de données de transactions bancaires issues d'Ouganda nous est fourni. Ces données ont été étés publiés dans le cadre d'une compétition Xente de 2019. https://zindi.africa/competitions/xente-fraud-detection-challenge. Notre projet est disponible sur GitHub à l'adresse suivante : https://github.com/LeTouristeDeLECAM/Lab_AI_Fraud_Detection pour des questions de propriété et droit les jeux de données ne sont pas disponibles sur GitHub.

II. Données

Pour nous permettre de prédire si une transaction sera fraudeuse ou pas, il nous faut un modèle qui sera alimenté avec une quantité suffisante de données lui permettant d'apprendre les caractéristiques lui servant à effectuer la classification. Nous sommes en possession d'un ensemble de transactions passées sous forme de fichier CSV. Chacune des transactions se compose de 16 caractéristiques qui sont énumérées dans le tableau ci-dessous avec une courte description accompagnée de leur type respectif.

COLUMN NAME	DEFINITION	TYPE
TRANSACTIONID	Unique transaction identifier on platform	object
BATCHID	Unique number assigned to a batch of transactions for processing	object
ACCOUNTID	Unique number identifying the customer on platform	object
SUBSCRIPTIONID	Unique number identifying the customer subscription	object
CUSTOMERID	Unique identifier attached to Account	object
CURRENCYCODE	Country currency	object
COUNTRYCODE	Numerical geographical code of country	int64
PROVIDERID	Source provider of Item bought.	object
PRODUCTID	Item name being bought.	object
PRODUCTCATEGORY	ProductIds are organized into these broader product categories.	object
CHANNELID	Identifies if customer used web,Android, IOS, pay later or checkout	object
AMOUNT	Value of the transaction. Positive for debits from customer account and negative for credit into customer account	float64
VALUE	Absolute value of the amount	int64
TRANSACTIONSTARTTIME	Transaction start time	object
PRICINGSTRATEGY	Category of Xente's pricing structure for merchants	int64
FRAUDRESULT	Fraud status of transaction 1 -yes or 0-No	int64

Tableau 1: Description des données

Il s'agit de données qui sont brutes, notre but est de faire une série de manipulations dessus pour en tirer le meilleur parti avant de se lancer dans l'implémentation du classifieur. Ces manipulations seront expliquées brièvement dans la suite de ce document.

1. Nettoyage

Il est important de commencer avec un jeu de données qui comporte aucune erreur, aucune cellule vide (Null) ou d'irrégularités. Dans notre cas, aucune transaction comporte de cellules vides ou d'erreurs. Cela dit, certaines caractéristiques ont une valeur numérique accompagnée d'une chaîne de caractères rendant impossible les opérations de calculs basiques. Nous avons retiré ces caractères avec la fonction split('_') et ne garder que les éléments se situant après le séparateur. Nous avons également choisi de mettre un nouvel index qui serait tout simplement le numéro de transaction. A présent, il nous est possible de passer à l'études des caractéristiques dans le chapitre suivant.

2. Features engineering

Le « feature engineering » reprends un ensemble de méthodes et stratégies visant à faire un choix des caractéristiques que nous estimons être les plus pertinentes et qui seront fournies à notre modèle dans la phase d'apprentissage et d'inférence.
Dans cette étape nous allons également manipuler des variables depuis notre ensemble de données initiales pour transformer et en créer des nouvelles. Le but est d'avoir des variables qui sont les plus pertinentes pour la tâche de modélisation. Il s'agit d'une étape cruciale car elle contribue grandement à la capacité d'un modèle à réaliser la classification correcte dans notre cas.

Une multitude de méthodes seront appliquées pour obtenir des caractéristiques pertinentes….
Nous avons commencé par retirer les factures que nous avons jugé inutiles de manière spontanée. Nous avons choisi de retirer 'CurrencyCode' et 'CountryCode', toutes les transactions se font dans la même devise et dans le même pays. 'Amount' est innutile car c'est la même chose que Value à l'exception du signe +/-. 'CustomerId' également car nous avons 'accountId' qui sera pris en compte dans l'entrainement.

Une fois que nous avons fait un premier filtrage, on réalise une transformation sur la valeur de 'TransactionStartTime'. La base de données à notre disposition nous permet observer la présence de la caractéristique horodatage sous format américain. Les données s'étendent du 15 novembre 2018 au 13 février 2019, nous avons décidé de nous intéresser à l'heure des transactions pour essayer d'identifier un comportement qui nous permettrai de crée une caractéristique plus pertinente.

Pour ce faire, nous allons observer la distribution en fonction de l'horaire, de toutes les transactions puis les transactions frauduleuses. Nous mettrons également ces graphiques en perspective en étudiant la proportion de transactions frauduleuses en fonction de l'horaire.

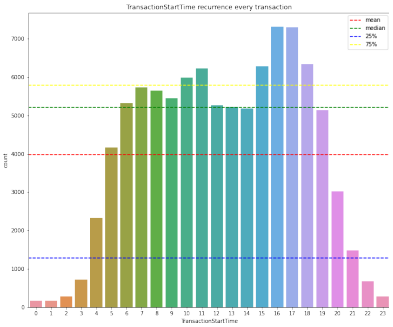


Figure 1: Distribution des transactions en fonction de l'heure

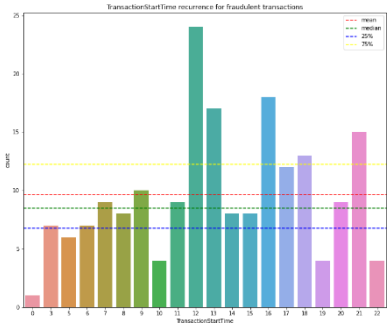


Figure 2: Distribution des transactions frauduleuses en fonction de l'heure

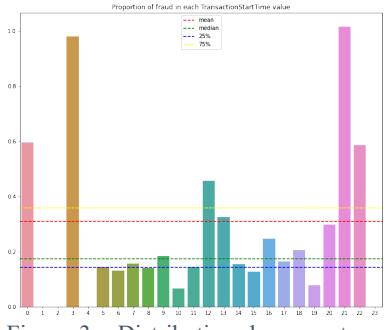


Figure 3: Distribution du rapport des transactions frauduleuses en fonction de l'heure

Nous pouvons observer que la distribution des transactions suit une courbe gaussienne avec une petite chute sur le temps de midi (Figure 1).

En observant la distribution des transactions frauduleuses, il semble que la distribution suit une gaussienne comme précédemment mais avec un écart-type plus grand (Figure 2).
En observant la distribution de la proportion des transactions frauduleuses (Figure 3), nous pouvons identifier diverses tranches horaires : nuit, matinée, midi et après-midi. Les transactions frauduleuses semblent être plus importantes durant la nuit et le temps de midi.

Nous avons donc divisé en tranche horaire de la manière suivante :

- 1: Morning 04h00 - 11h59
- 2: Lunch 12h00 - 13h59
- 3: Afternoon 12h00 - 19h59
- 4: Night 20h00 - 03h59

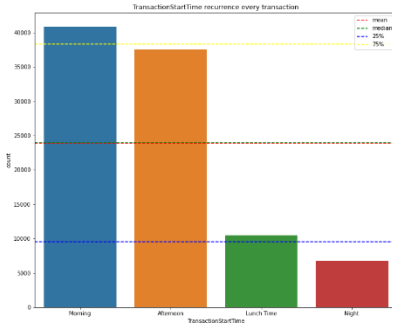


Figure 4: Distribution des transactions en fonction de l'heure

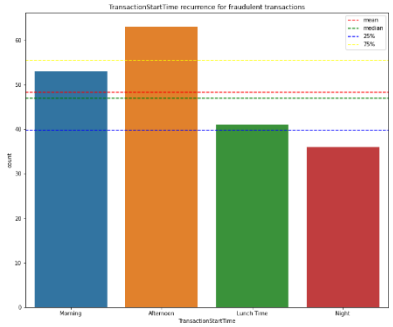


Figure 5: Distribution des transactions frauduleuses en fonction de l'heure

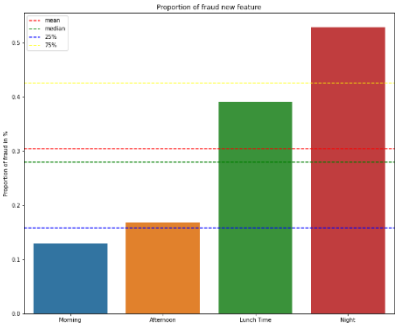


Figure 6: Distribution de la proportion des transactions frauduleuses en fonction de l'heure

Nous réalisons les mêmes graphiques pour les données issues de notre nouvelle caractéristique, nous observons que la proportion des transactions frauduleuses est sensiblement plus élevée durant la nuit et le temps de midi.

Par ailleurs, nous passons par un encodeur de type one hot pour passer de données catégorielles vers ordinales. Ainsi chaque catégorie existante au sein d'une caractéristique en une nouvelle colonne ayant une valeur binaire. Cela permet d'augmenter le nombre de caractéristiques.

A présent, nous sommes arrivés à l'étape ou nous allons utiliser la régression logistique. Il s'agit d'un type d'algorithme qui permet de faire la classification binaire et multinomial. Lorsque nous mettons en place un modèle de régression logistique, nous pouvons ainsi évaluer les poids de chaque caractéristique et ainsi retirer celles qui sont moins importantes. Ainsi, nous pouvons former un modèle avec une complicité moindre et par conséquent plus précis. Sur la figure 7, nous voyons les coefficients de chaque caractéristique, plus il est grand, plus son importance est élevée. Il est important de mentionner que les données en entrée du modèle décrit doivent être impérativement normalisées. A l'issue de cette étape, nous avons ateint 32 caractéristiques dont les 15 les plus pertinentes sont :

INDEX	ATTRIBUTE	IMPORTANCE
4	Value	0.669972
20	ProviderId_3	0.604002
22	ProviderId_5	0.505009
18	ProviderId_1	0.371593
13	ProductCategory_utility_bill	0.284851
7	ProductCategory_financial_services	0.273339
16	ChannelId_3	0.229452
30	Hour_3	0.157197
31	Hour_4	0.127462
14	ChannelId_1	0.090762
27	PricingStrategy_4	0.087083
2	SubscriptionId	0.048229
11	ProductCategory_transport	0.040076
1	AccountId	0.030601
24	PricingStrategy_0	0.007077

Tableau 2 : Importance des caractéristiques

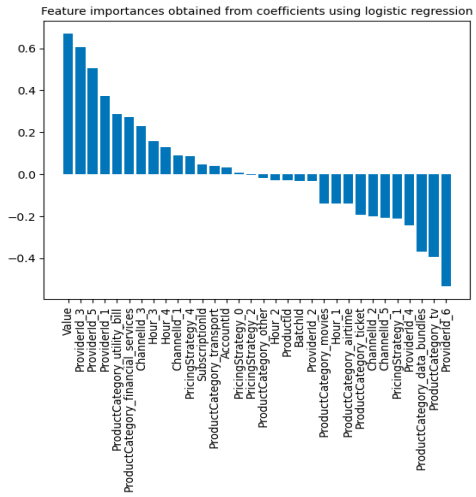


Figure 7 : importance des caractéristiques

3. Fractionnement des données et validation croisée

Le fractionnement des données est une technique courante utilisée dans l'apprentissage automatique pour évaluer les performances d'un modèle sur des données qui n'ont pas été utilisées pour l'entraînement. Le fractionnement des données consiste à diviser l'ensemble de données en deux parties distinctes : un ensemble d'entraînement et un ensemble de test. Cette pratique est fortement employée lorsque notre ensemble de données est réduit et permet de diminuer le risque de surajustement (over-fitting).

Dans le scénario de détection de transactions frauduleuses, le fractionnement des données est particulièrement important car il permet de s'assurer que le modèle est capable de généraliser à de nouvelles données, c'est-à-dire de prédire avec précision la classe de transactions frauduleuses et non frauduleuses qui ne font pas partie de l'ensemble d'entraînement.

Il est important de s'assurer que les deux ensembles sont mutuellement exclusifs et que les classes sont équitablement représentées dans les deux ensembles. Ainsi, il nous faudra faire de rééchantillonnage de nos sous-ensembles pour assurer cette propriété.

4. Déséquilibrage

Notre ensemble de donnée est sujet à un souci récurent dans le domaine de la classification. Il s'agit des ensembles de données qui sont déséquilibrés. Cela veut dire que nous avons une des classes (transactions frauduleuses dans notre cas concret) qui est sous représentée du fait de son petit nombre comparé aux transactions non-frauduleuses.

C'est un problème dans le monde de l'intelligence artificielle car une grande partie des algorithmes d'apprentissage sont pour atteindre une haute précision en minimisant l'erreur globale. Ainsi, lorsqu'un ensemble est déséquilibré, l'algorithme peut avoir un biais vers la classe considérée comme étant majoritaire. Cela compromet inévitablement la compétence du modèle à réaliser une classification correcte et performante.
Le ratio entre la classe frauduleuse, dite « rare » et la classe majoritaire et de 193 pour 95469, soit 0,2017520018% des transactions sont frauduleuses. Il s'agit d'un déséquilibre fort. Notre rôle sera de faire une série de manipulations pour remédier au souci que nous venons d'exposer.

Pour remédier à ce problème, il nous est possible de mettre en place plusieurs stratégies.
Premièrement, il existe la génération de données synthétiques. Nous avons également le rééchantillonnage qui mettras plus en valeur la classe minoritaire. Les méthodes dites d'ensemble, visent à combiner plusieurs modèles qui seront entrainés sur des sous-ensembles de données. Cela est fort intéressant car une amélioration de la classification est observable tout en diminuant la sur-adaptation du modèle qui est récurrente. Nous pouvons mentionner des méthodes telles que le « bagging » et le « boosting » qui seront décrites plus en détail. Il est égaiement pertinent de mettre en place des métriques qui seront sensibles à la classe rare. Les métriques alternatives à l'exactitude sont la précision, le rappel (recall en anglais) ainsi que le score F1.
La combinaison de ces solutions peut aider à contrer le problème mais il reste important d'expérimenter pour s'assurer de leur impact positif sur nos modèles. Ces solutions ne sont pas garanties de fonctionner pour tous les cas de données déséquilibrées. Une multitude d'autres solutions existent. Dès lors, il est important de cerner le cas dans lequel nous travaillons et effectuer des recherches des implémentations déjà existantes.

III. Modèle

Comme présenté précédemment le jeu de données est déséquilibré nous devons donc mettre en place des stratégies pour équilibrer le jeu de données et permettre aux algorithmes de classification de machine Learning d'opéré ceux-ci étant sensibles aux déséquilibrage des données.

Nous mettrons en pratique deux méthodes de rééchantillonnage parmi celles qui ont été énumérée pour palier à notre souci de déséquilibrage :

- Sous-échantillonnage : Consiste à réduire la taille du jeu de données en supprimant les éléments de la classe majoritaire. Il existe des stratégies pour réduire la taille du jeu de données de manière intelligente comme Edit Nearest Neighbours.
- Génération de données synthétiques : Consiste à augmenter artificiellement le nombre de données issue de la classe minoritaire. Soit en les recopiant au risque de réaliser un sur-apprentissage lors de la phase d'entrainement du modèle, ou en utilisant de méthode intelligente comme SMOTE qui essaye de crée des données entre deux points.

Il est également possible d'utiliser des algorithmes qui sont plus résistants à ce problème en utilisant des méthodes :

- Bagging : Consiste à diviser le jeu de donnée en pleins de jeu de données par une méthode de sous-échantillonnage et classifier sur les petits jeux donnés pour réaliser une 'moyenne' sur l'ensemble des classifieurs entrainé.
- Boostrampping : Consiste à diviser le jeu de donnée en pleins de jeu de données par une méthode de under sampling, Le classifieur classifie le jeu de donnée en pondérant de manière plus importante les données où il a échoué, chaque nouveaux classifieur recevra ainsi un jeu de donnée pondéré par le précèdent.

Pour évaluer notre performance il est aussi nécessaire d'utiliser une autre méthode que la précision, nous allons utiliser le score f1 qui est la moyenne harmonique de la précision et du recall :

$$f1\ score = \frac{(2 * Précision * recall)}{recall + précision}$$

Avec :

$$recall\ = \frac{True\ positives}{True\ Positive\ + False\ Negatives}\ Précision = \frac{True\ positives}{True\ positives + False\ Positives}$$

1. Easy Ensemble Model

L'Easy Ensemble Model est un modèle utilisant du boostrampping combiné à un sous-échantillonnage aléatoire. Il nous est possible de choisir le classifieur utilisé et régler plusieurs paramètres tels que la proportion de données minoritaire introduite dans le jeu de donnée et le nombre de jeu de données.
Nous avons décidé de réaliser l'entrainement sur plusieurs classifieurs :

- AdaBoostClassifier : L'expérimentation nous a amenée à sélectionner les paramètres suivants, 10 jeux de données (n_estimators), et une sampling stratégie de 1% ce qui signifie que les données de fraude représentent 1% du jeu de donnée sur lequel le modèle s'entraine.

- GradientBoostingClassifier : L'expérimentation nous a amenée à sélectionner les paramètres suivants, 5 jeux de données (n_estimators), et une sampling stratégie de 1% ce qui signifie que les données de fraude représentent 1% du jeu de donnée sur lequel le modèle s'entraine.

- RandomForestClassifier : L'expérimentation nous a amenée à sélectionner les paramètres suivants, 2 jeux de données (n_estimators), et une sampling stratégie de 1% ce qui signifie que les données de fraude représentent 1% du jeu de donnée sur lequel le modèle s'entraine.

2. sous-échantillonnage et génération de données synthétiques

Dans un souci de comparaison nous avons également décidé d'implémenter modèle reprenant une stratégie plus conventionnelle pour la gestion du déséquilibrage. Notre choix c'est porté par la combinaison d'un un sous-échantillonnage et la génération de données synthétiques.

Dans un premier temps nous réalisons un sous-échantillonnage aléatoire, où les données minoritaires représentent 4% du nouveau jeu de donné.
La deuxième étape consiste à augmenter artificiellement le nombre de données minoritaire dans le jeu de données, Pour ce faire notre choix se porte sur la méthode SMOTENC qui est une méthode SMOTE où il nécessaire de préciser l'index des données catégorielles.
Suite à ces deux opérations, nous obtenons un jeu de donnée équilibré de 7850 données prêtes à être utilisées pour entrainer un classifieur.

Nous implémentons un RandomForestClasifier avec un n_ estimators de 30 arbres, l'expérimentation nous a conduit à choisir ces paramètres.

IV. Comparaison des modèles sur les données de validation

MODELES	F1 LOCAL	F1 PUBLIC	F1 PRIVE
Easy ensemble model avec adaboost	0,88	0.5714	0,6176
Easy ensemble model avec gradientboosting	0,96	0.6774	0,6929
Easy ensemble model avec randomforest	0,88	0.6667	0,6887
Random under sampling + smote radomforest	0,55	0.2785	0,2864

Tableau 3: F1 score des modèles

Comme nous pouvons l'observer, les résultats des modèles utilisant du boosting grâce à Easy Ensemble Model sont grandement plus performant que le modèle Random Forest Classifier combiné à un sous-échantillonnage de la classe majoritaire et la génération de données synthétiques pour la classe minoritaire. Cette différence de performance avec les solutions EEM est liée à l'utilisation d'une stratégie de boostrampping, qui semble mieux répondre à ce scénario.

Les points que nous pourrions essayer de travailler pour améliorer notre sans changer nos modèles sont les caractéristiques des données, nous pourrions poursuivre l'étude de la distribution des transactions sur les jours de la semaine, de même sur la distribution hebdomadaire sur le mois. Il est éventuellement envisageable d'introduire le calendrier des jours fériés et le calendrier chrétien.
Il est aussi envisageable de calculer une nouvelle caractéristique qui reprend le nombre de compte par utilisateur.

V. Conclusion :

En suivant la structure que nous avons décrit dans ce document, nous avons obtenu des données nettoyée et traitées, plusieurs modèles dont nous avons testé les performances et comparés entre eux.
Avec le choix des caractéristiques actuelles nous avons un résultat F1 qui est acceptable (0,6929).

Cela dit, il est encore possible d'améliorer ce dernier. Pour augmenter ce score, il serait pertinent de faire une analyse des composantes principales afin de réduire davantage la dimensionnalité de l'ensemble des données. Cela rendrait le modèle moins complexe en trouvant des nouvelles variables. Nous avons réalisé diverses méthodes pour identifier les relations entre les variables mais les résultats n'étaient pas satisfaisants à notre gout. Nous pourrions essayer de re faire un entrainement avec moins de features car il y a également une part d'expérimentation dans ce genre de projets. Nous avons choisi de garder 15 features sur les 32 que nous avons obtenus (Tableau 2).
Si le temps n'était pas une contrainte, nous aurions aimé mettre en place un réseau neuronal pour faire davantage d'essais et potentiellement trouver une meilleure solution.

Nous avons été agréablement surpris par la facilité à mettre en place les techniques de rééchantillonnage. L'entrainement de nos modèles n'a pas nécessité la puissance de calculs dont dispose Kaggle.

Ce travail nous a permis également de mettre en pratique les acquis et s'assurer de la compréhension de notions propres au Machine Learning.