Développement d'un modèle de détection des transactions bancaires frauduleuses à l'aide de la bibliothèque d'apprentissage automatique MLLib d'Apache Spark

Yanis Gherdane
25/07/2023

1 Introduction

Depuis plusieurs années maintenant le domaine de l'automatisation dans le secteur financier est en plein essor. En effet, ce domaine répondant à divers besoins des banques, particulièrement, la détection de fraude, les groupes bancaires y investissent de plus en plus. De nombreuses recherches ont été réalisées et plusieurs approches ont été utilisées.

Les transactions frauduleuses ne représentent qu'une infime partie des transactions bancaires, mais cela peut rapidement entraîner des pertes financières importantes et porter atteinte à la réputation d'une entreprise, laissant les clients facturés pour des biens qu'ils n'ont pas achetés. Pour éviter cela, les banques utilisent désormais la Data Science et le Machine Learning pour détecter et prévoir la fraude. Le but de mon travail est de développer un modèle de détection de fraude dans les transactions bancaires. Avec l'explosion des données que connaît le secteur de la banque, les modèles d'apprentissage peuvent nécessiter de longues heures de traitement. La technologie qui résout ce problème est la librairie MLlib de l'API Pyspark, une librairie d'apprentissage automatique qui fournit des outils pour l'analyse des données et permettant d'effectuer un traitement de larges volumes de données de manière distribuée. Elle fournit également une API pour créer et entraîner des modèles d'apprentissage automatique. En outre, MLlib prend en charge les pipelines d'apprentissage automatique, ce qui permet aux utilisateurs de développer un flux de travail cohérent pour transformer leurs données et exécuter différents algorithmes afin d'entraîner leurs modèles. Par ailleurs, il offre une API standardisée pour la gestion, l'entraînement et le déploiement des modèles d'apprentissage automatique.

2 Données

(https://www.kaggle.com/datasets/ealaxi/paysim1)
Le jeu de données utilisé est récupéré sur Kaggle. Il est généré à l'aide du simulateur PaySim qui simule des transactions mobiles d'argent sur la base d'un échantillon de transactions réelles extraites d'un mois de journaux financiers d'un service d'argent mobile mis en œuvre dans un pays africain.

Description des variables

- **'type'** type de transaction : payment, transfer, cash out, cash in, debit.
- **'amount'** montant de la transaction.
- **'nameOrig'** identifiant de l'émetteur.
- **'oldbalanceOrg'** solde de l'émetteur avant la transaction.
- **'newbalanceOrig'** solde de l'émetteur après la transaction.
- **'nameDest'** identifiant du destinataire.
- **'oldbalanceDest'** solde du destinataire avant la transaction.
- **'newbalanceDest'** solde du destinataire après la transaction.
- **'isFraud'** 1 si la transaction est frauduleuse, sinon 0.
- **'isFraggedFraud'** 1 si la transaction a été signalée frauduleuse, sinon 0.

Types des variables

```
root
 |-- step: string (nullable = true)
 |-- type: string (nullable = true)
 |-- amount: string (nullable = true)
 |-- nameOrig: string (nullable = true)
 |-- oldbalanceOrg: string (nullable = true)
 |-- newbalanceOrig: string (nullable = true)
 |-- nameDest: string (nullable = true)
 |-- oldbalanceDest: string (nullable = true)
 |-- newbalanceDest: string (nullable = true)
 |-- isFraud: string (nullable = true)
 |-- isFlaggedFraud: string (nullable = true)
```

3 Problématique

Comment tirer parti de ce jeu de données pour créer un projet de Machine Learning de bout en bout pour catégoriser les transactions en transactions frauduleuses et non frauduleuses ?

4 Objectif

Dans la résolution d'une problématique Data science, un Data scientist procède d'abord à la génération d'hypothèses, cela implique de comprendre le problème en détail en réfléchissant aux facteurs qui peuvent avoir un impact sur le résultat. Cela se fait en comprenant parfaitement le problème et avant d'examiner les données.

Ci-dessous certains des facteurs qui, à mon avis, peuvent fortement affecter la cible, c'est-à-dire une transaction frauduleuse :

Type de transaction ou mode de paiement, cela peut avoir un impact sur la transaction frauduleuse.

Montant nous pouvons trouver des activités inhabituelles grâce au montant de la transaction.

Solde initial et solde restant en comparant les soldes des comptes avant la transaction et après la transaction, il est possible de déterminer s'il y a une activité inhabituelle.

Le problème est un problème de classification binaire.

5 EDA des données

Lorsque l'on est confronté à un cas d'usage en Data science, la première étape est de procéder à une analyse exploratoire des données. Il s'agit principalement de calculer des mesures récapitulatives, telles que la moyenne, la valeur maximale, la valeur minimale etc. Et de réaliser des représentations graphiques, des diagrammes pour visualiser la répartition des données. Cela, permet d'obtenir les informations les plus pertinentes sur les données et d'avoir une vision plus claire sur les données et les interactions entre les différentes variables.

5.1 Statistique descriptive et visualisations

On calcule les mesures récapitulatives pour le jeu de données.
On peut observer les intervalles [min, max] des chacune des variables, les moyennes et les écarttypes.

	summary	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	count	3181807	3181807	3181807	3181807	3181807	3181807	3181807
1	mean	243.33075419093615	180376.4924033053	834950.8743586149	856168.4637992503	1101763.094531916	1226409.6227083874	0.0012734901896940952
2	stddev	142.39545491604187	613321.2650007016	2891187.841932541	2927031.285909358	3369890.924303856	3654159.749004548	0.035663269790674586
3	min	1.0	0.0	0.0	0.0	0.0	0.0	0
4	max	743.0	6.98867313E7	5.958504037E7	4.958504037E7	3.5601588935E8	3.5617927892E8	1

```
+-------+-------+
|isFraud|  count|
+-------+-------+
|      1|   4052|
|      0|3177755|
+-------+-------+
```

Dans les graphes suivants, on observe que les transactions frauduleuses représentent une partie très infime de l'ensemble de transactions (on a que 8 213 (0,13 %) sur un jeu de données de 3 millions de transactions).

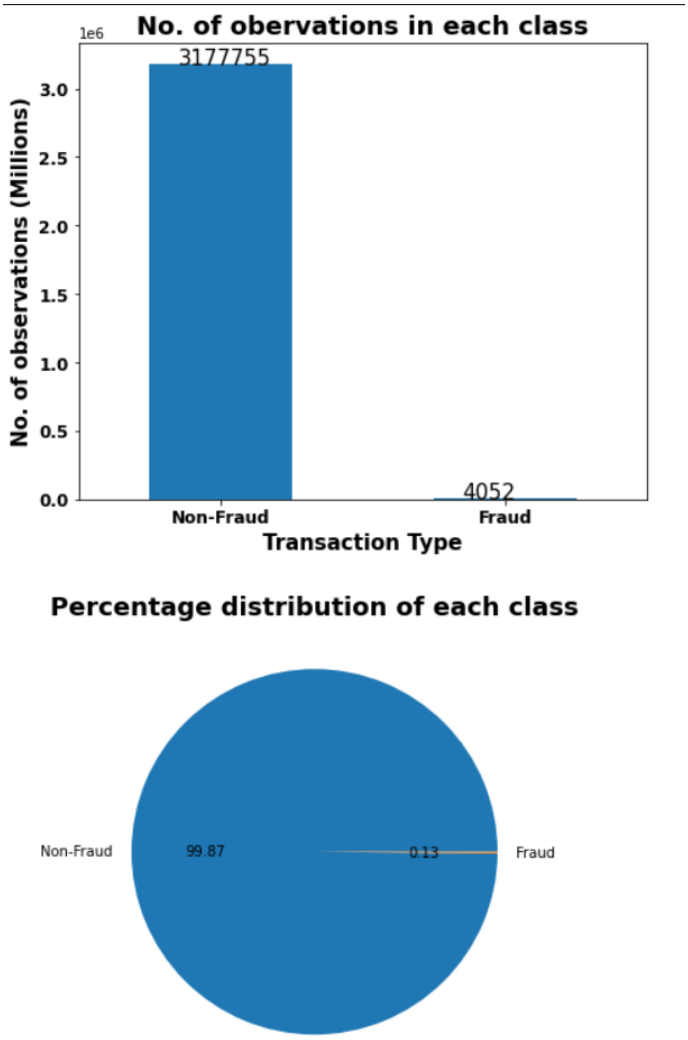
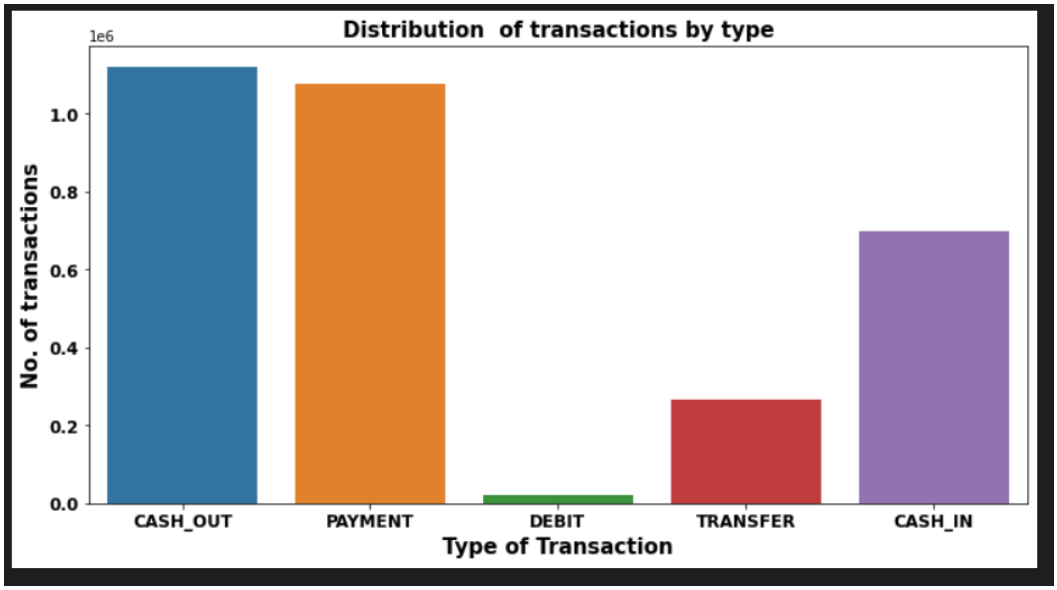
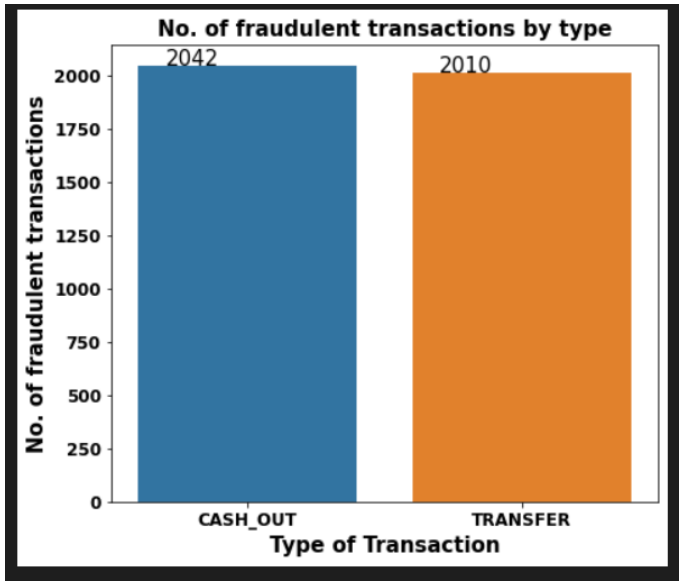


Figure 1: Distribution des transactions par type de transactions

On observe ci-dessous que la majorité des transactions sont de type CASH_OUT, PAYMENT et CASH_IN.

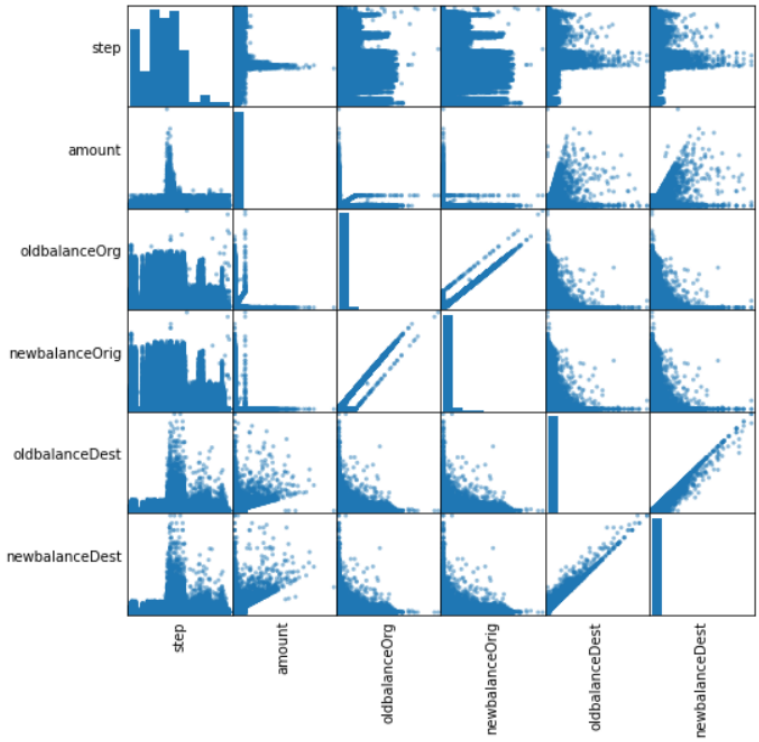
On observe également que les transactions frauduleuses sont uniquement des retraits CASH_OUT et des virements TRANSFER avec des pourcentages de fraude de 50%. Pour les autres types, il n y'a pas de fraude.

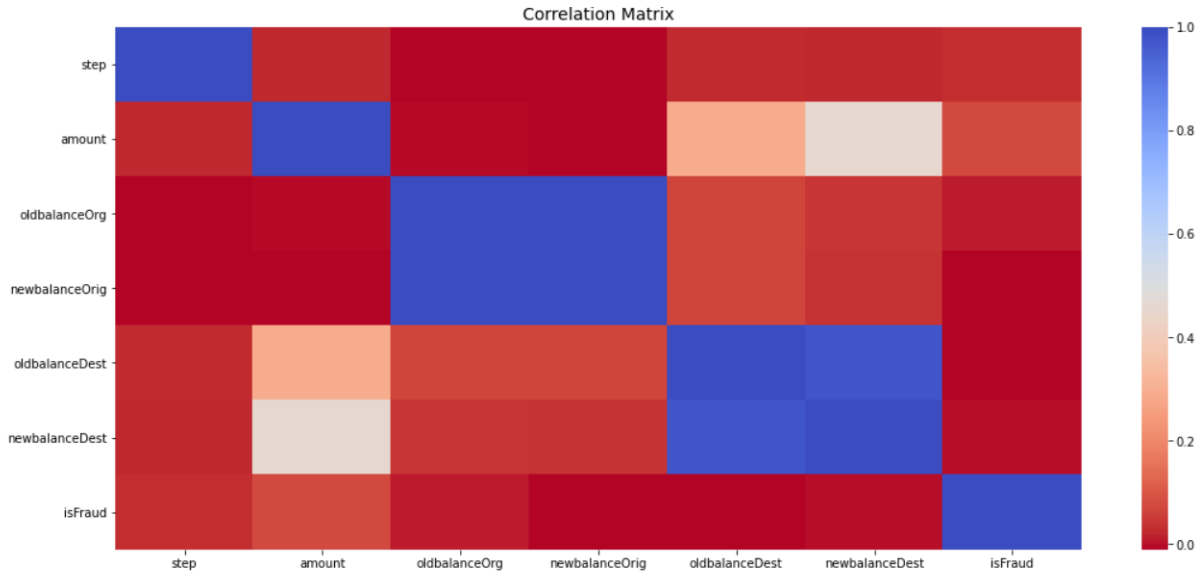




5.2 Analyse de corrélation

L'analyse de corrélation permet d'identifier des indépendances entre variables. Il est important de noter que si deux variables sont très corrélées, alors c'est qu'elles apportent des informations similaires. Ainsi, il y aurait de la redondance durant l'apprentissage du modèle.
La figure suivant montre la corrélation entre chacun des variables du jeu de données.





Ici, on peut observer qu'il y a des variables corrélées entre elles, d'où une redondance dans les données.
- La variable "oldbalanceDest" et "newbalanceDest" sont corrélées, on doit donc supprimer une d'entre elles. Pour décider laquelle supprimer, on doit vérifier leurs corrélations avec la variable dépendante "isFraud", "oldbalanceDest" est fortement corrélé négativement avec elle que "newbalanceDest". Donc, je dois supprimer la variable "newbalanceDest".

- De même " oldbalanceOrig" et "newbalanceOrig" sont également corrélés entre elles et newbalanceOrig est fortement corrélé négativement avec "isFraud" que "oldbalanceOrg". Donc, je dois supprimer la variable "oldbalanceOrig".

6 Pré traitement des données

Le pré traitement des données est un processus crucial dans le traitement des données. Il implique l'identification et la suppression des données incorrectes, manquantes ou inutiles, afin de garantir que les données restantes sont valides et prêtes à être analysées. Il faut s'assurer que les ensembles de données soient cohérents avant de pouvoir les analyser. De plus, le nettoyage des données peut faciliter l'accès aux informations pertinentes et accroître la qualité des résultats obtenus. Les outils d'analyse automatisée peuvent également être utilisés pour faciliter le processus de nettoyage des données. Ainsi, les données seront plus compréhensibles par le framwork Apache Spark.

L'ensemble de données contient 6362620 lignes et 11 colonnes. La dernière colonne désigne la classe (1 si transaction frauduleuse, 0 sinon). Du fait de la puissance de calcul de ma machine est pas forte, je traite la moitié de cet ensemble de données.

```
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+-------------+
|step|    type|  amount|   nameOrig|oldbalanceOrg|newbalanceOrig|   nameDest|oldbalanceDest|newbalanceDest|isFraud|isFlaggedFraud|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+-------------+
|   1| PAYMENT| 9839.64|C1231006815|     170136.0|     160296.36|M1979787155|           0.0|           0.0|      0|            0|
|   1| PAYMENT| 1864.28|C1666544295|      21249.0|      19384.72|M2044282225|           0.0|           0.0|      0|            0|
|   1|TRANSFER|   181.0|C1305486145|        181.0|           0.0| C553264065|           0.0|           0.0|      1|            0|
|   1|CASH_OUT|   181.0| C840083671|        181.0|           0.0|  C38997010|       21182.0|           0.0|      1|            0|
|   1| PAYMENT|11668.14|C2048537720|      41554.0|      29885.86|M1230701703|           0.0|           0.0|      0|            0|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+-------------+
only showing top 5 rows
```

6.1 Vérification des données manquantes

Il n'y a pas de valeurs manquantes dans ce jeu de données.

```
+----+----+------+--------+-------------+--------------+--------+--------------+--------------+-------+-------------+
|step|type|amount|nameOrig|oldbalanceOrg|newbalanceOrig|nameDest|oldbalanceDest|newbalanceDest|isFraud|isFlaggedFraud|
+----+----+------+--------+-------------+--------------+--------+--------------+--------------+-------+-------------+
|   0|   0|     0|       0|            0|             0|       0|             0|             0|      0|            0|
+----+----+------+--------+-------------+--------------+--------+--------------+--------------+-------+-------------+
```

6.2 Transformation de données

Afin d'assurer la cohérence des données avec les modèles qui seront utilisés, il est important de vérifier les types de données.

```
root
 |-- step: string (nullable = true)
 |-- type: string (nullable = true)
 |-- amount: string (nullable = true)
 |-- nameOrig: string (nullable = true)
 |-- oldbalanceOrg: string (nullable = true)
 |-- newbalanceOrig: string (nullable = true)
 |-- nameDest: string (nullable = true)
 |-- oldbalanceDest: string (nullable = true)
 |-- newbalanceDest: string (nullable = true)
 |-- isFraud: string (nullable = true)
 |-- isFlaggedFraud: string (nullable = true)
```

On voit bien que toutes les variables sont de type String comme. Il est important de noter que Pyspark ne peut travailler que sur des variables numériques. D'où la nécessité de transformer les variables catégorielles et numériques présentés en String en des variables numériques. Pour cela on utilise la fonction cast et le StringIndexer pour transformer en valeurs numériques indexées les variables type, nameOrig et nameDest, qui contiennent des données catégorielles.

Pour les variables 'nameOrig' et 'nameDest', on doit séparer le premier caractère de la chaîne ('C' ou 'M') du reste afin de pouvoir les transformer en valeurs numérique binaire (0 ou 1).
Pour les colonnes résultantes 'str_orig' et 'str_est' contenant les valeurs catégorielles et la colonne 'type', j'utilise le OneHotEncoder pour encoder les String en numérique.
On peut procéder maintenant la résolution du problème de classification binaire pour détecter si une transaction est frauduleuse (classe 1) ou non (classe 0).

7 Développement de l'application de détection de fraude en utilisant MLlib d'Apache Spark

Afin d'effectuer la résolution du problème de classification binaire, on dispose de données prêtes à l'utilisation. Il reste à choisir un algorithme approprié. Dans ce cas, on testera plusieurs algorithmes de classification afin de déterminer le plus efficace. Le choix peut se porter sur un algorithme de forêt aléatoire, d'arbre binaire, de Naive Bayes, de GBT classifier ou un algorithme de régression logistique. Une fois l'algorithme sélectionné, il est possible d'entraîner le modèle et d'utiliser les résultats pour prédire les classes. Enfin, une évaluation approfondie devrait être effectuée afin de vérifier la précision des modèles et ses capacités à généraliser à des données de test non vues auparavant.

Avant d'appliques les algorithmes choisis, on doit transformer les données à l'aide de VectorAssembler pour avoir des représentations vectorielles des données en entrée des modèles. L'étape suivante consiste à deviser le jeu de données en données d'entraînement et données de test (80% de training set et 20% de test set).

Par ailleurs, il faut créer un modèle d'apprentissage pour l'entraîner sur les données d'entraînement, puis effectuer des prédictions sur les données de test. Enfin, faire une évaluation des prédictions des modèles.
Ci-dessous, je décris une implémention de l'algorithme forêt aléatoire, et ce sera la même démarche pour les autres algorithmes.
Instancier le modèle en lui passant comme paramètres le label de la colonne "isFraud" et le vecteur des variables résultant de VectorAssembler, puis faire l'entraînement avec la méthode .fit() sur les données d'entraînement :

```
def rf_train(train):
    # Train a RandomForest model.
    rf = RandomForestClassifier(labelCol="isFraud", featuresCol="features", numTrees=10)
    rf_model = rf.fit(train)
    return rf_model
```

8 Évaluation des prédictions des modèles

On peut maintenant utiliser les métriques d'évaluation pour évaluer la précision du modèle. On vérifie si les prédictions correspondent aux valeurs réelles et déterminer si le modèle est capable de générer des prédictions fiables. On peut également mesurer la disposition des données et le nombre d'erreurs commises par le modèle. Il est important de comprendre les métriques qui seront utilisées pour l'évaluation. Une fois que les métriques sont calculées, On peut être sûrs du modèle qui fournit de meilleurs prédictions.

Les métriques d'évaluation utilisées sont Recall, Precision, F1 Score, Area Under ROC et Area Under PR :

Recall représente le taux de vrais positifs (classe 1 bien classés = cas frauduleux classés frauduleux). Grâce à cette mesure nous pouvons donc calculer le taux de faux négatifs = cas frauduleux classés non frauduleux.

Precision représente le taux d'identifications positives (classe 1 prédite par le modèle) qui étaient réellement correctes. Un modèle qui ne produit aucun faux positif a une précision de 1,0.

Score F1 une combinaison de Precision et de Recall. Un modèle parfait obtient un score F1 de 1,0.

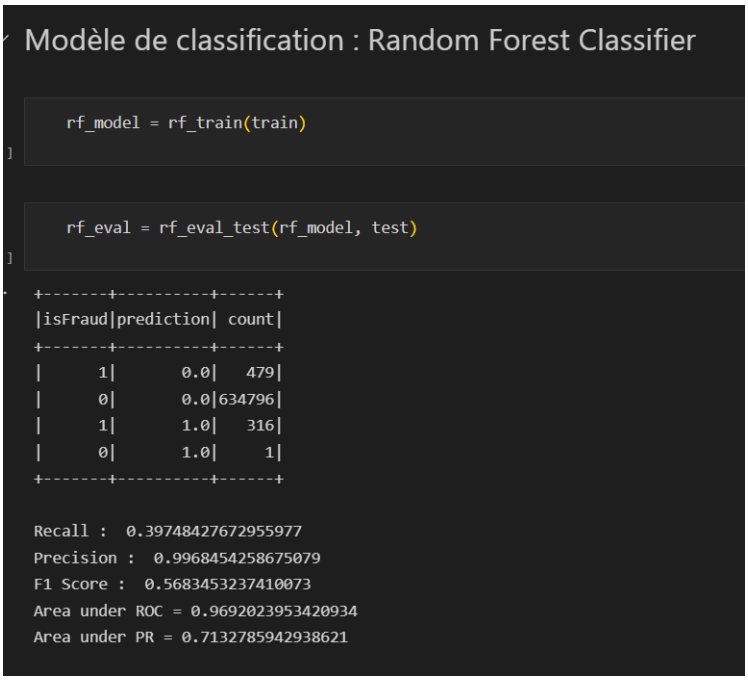
Area Under ROC (AUC) aire sous la courbe ROC. Un modèle parfait obtient un score de 1,0.

Area Under PR aire sous la courbe Recall-Precision.

Évaluer les prédictions du modèle Random Forest:

```
def rf_eval_test(rf_model, test):
    # Make predictions.
    prediction_RF = rf_model.transform(test)
    prediction_RF.groupBy("isFraud", "prediction").count().show()

    tp = prediction_RF[(prediction_RF.isFraud == 1) & (prediction_RF.prediction == 1)].count()
    tn = prediction_RF[(prediction_RF.isFraud == 0) & (prediction_RF.prediction == 0)].count()
    fp = prediction_RF[(prediction_RF.isFraud == 0) & (prediction_RF.prediction == 1)].count()
    fn = prediction_RF[(prediction_RF.isFraud == 1) & (prediction_RF.prediction == 0)].count()
    recall_RF = tp/(tp+fn)
    precision_RF = tp/(tp+fp)
    f1_score_RF = 2*(recall_RF*precision_RF)/(recall_RF+precision_RF)
    print("Recall : ",recall_RF)
    print("Precision : ", precision_RF)
    print("F1 Score : ", f1_score_RF)
    evaluator = BinaryClassificationEvaluator(labelCol="isFraud")
    # Area under ROC curve
    areaUnderROC_RF = evaluator.evaluate(prediction_RF, {evaluator.metricName: "areaUnderROC"})
    print("Area under ROC = %s" % areaUnderROC_RF)
    # Area under precision-recall curve
    areaUnderPR_RF = evaluator.evaluate(prediction_RF, {evaluator.metricName: "areaUnderPR"})
    print("Area under PR = %s" % areaUnderPR_RF)
    results = {}
    results['Random Forest Classifier'] = [recall_RF, precision_RF, f1_score_RF, areaUnderROC_RF, areaUnderPR_RF]
    return results
```



La même démarche est utilisée pour le reste des modèles.

9 Résultats et comparaisons des performances des modèles

Il est possible maintenant de choisir le meilleur modèle selon les critères dont on a besoin. Si on veut le modèle le plus précis on choisit celui qui a la meilleure précision.
Dans ce cas, les meilleurs algorithmes sont : Random Forest avec une précision de 99%, et le Gradient Boosted Tree avec 97%
Selon le F1 Score, le Gradient Boosted Tree classifier est le meilleur avec un F1 Score de 81%.
Selon le Area Under ROC c'est la regression logistique avec 99% et Gradient Boosted Tree classifier avec 98%.
Et enfin Selon Area Under Precision-Recall, c'est le Gradient Boosted Tree classifier avec 78%.

'**Observation** Le GRadient Boost Classifier est plus performant que les autres modèles si on considère toutes les métriques de l'évaluation. Donc c'est celui qui sera utilisé pour classer les nouvelles transactions.
L'algorithme Naïve Bayes, donne les résultats les moins bons, on peut expliquer cela par le déséquilibre qu'on a entre les deux classes (0.13% de la classe 1 et 99.87% non-fraude).

10 Conclusion

Travailler sur ce projet m'a permis de se familiariser avec les données de transactions bancaires, l'examination de ces données et la génération d'hypothèses et réfléchissant à des solutions en prenant en considération l'impact des differentes variables sur les résultats.
De plus, la réalisation de mon travail avec les modèles MLLib de Apache Spark m'a permis de découvrir qu'ils sont extrêmement puissants et peuvent aider les Data scientists à découvrir des insights précieux et profonds a partir de leurs données. Ils permettent un traitement rapide de grandes quantites de données, ce qui rend leur utilisation encore plus attrayante. Les modèles MLLib sont adaptables et peuvent aider les entreprises a prendre de meilleures décisions en temps réel. De plus, ils sont faciles à utiliser et offrent des fonctionnalités avancées qui peuvent être personnalisées pour satisfaire les besoins des utilisateurs.