Projet Random Forest

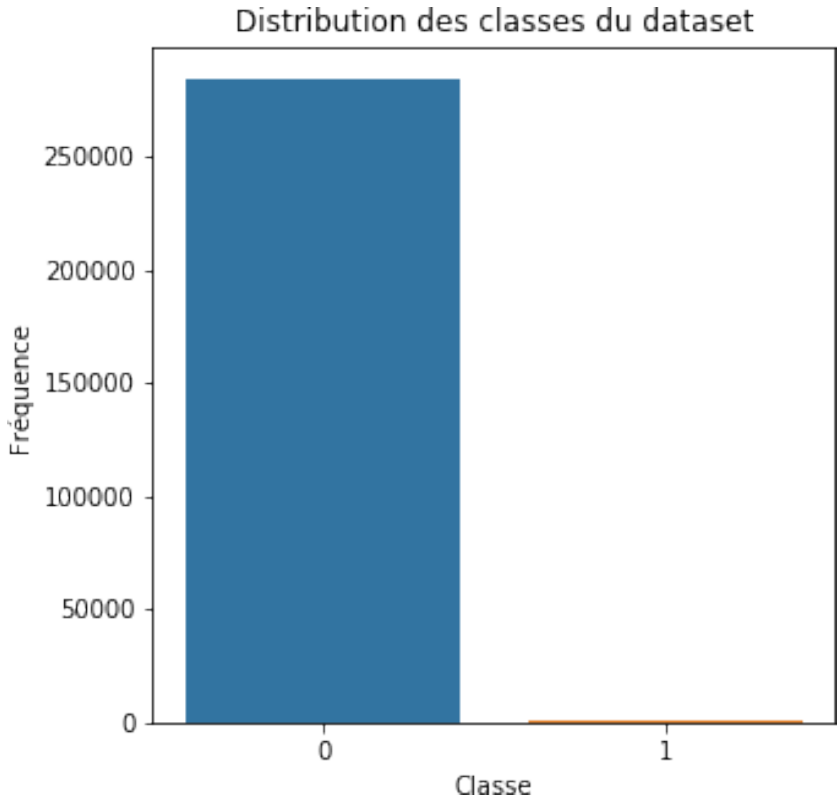
Rebecca Leygonie *Nemanja Kostadinovic*

Exploration des données :

Les jeux de données contiennent des transactions effectuées par des titulaires de carte bancaire européennes. Cet ensemble de données présente les transactions qui ont eu lieu en deux jours, il y a moins de 500 fraudes sur presque 285.000 transactions.

L'ensemble de données est très déséquilibré, la classe positive (fraude) représente 0,2% de toutes les transactions.

Confirmé par le graphique suivant représentant de la distribution des classes du jeu de données :



Application naive de l'algorithme Random Forest :

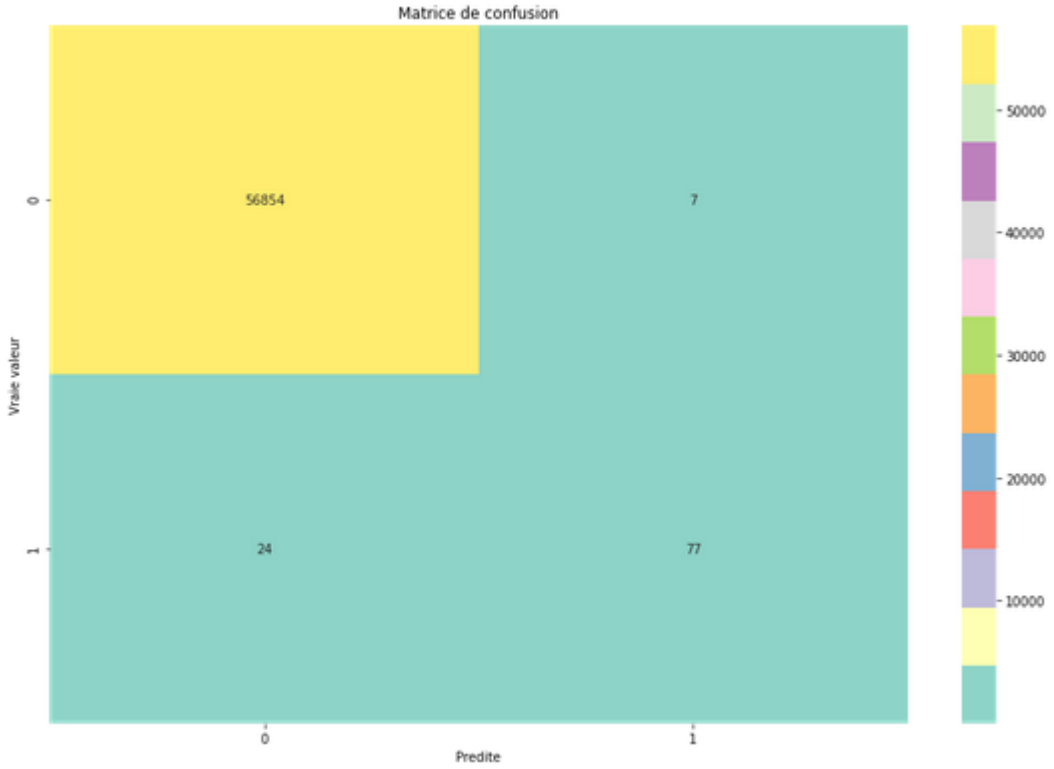
Nous avons entrainé un classifieur Random Forest sur les données d'origines en utilisant RandomizedSearchCV pour trouver les meilleurs hyper-paramètres :

```
Les paramètres du meilleur estimateur obtenu :
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=9, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=5,
                       min_weight_fraction_leaf=0.0, n_estimators=80,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Pour évaluer un modèle, il est d'usage de calculer la proportion de données bien classées (**accuracy**) ou le **taux d'erreur**, nombre de données mal classées (1-accuracy)

Ici, l'accuracy = 0.9995303824968728 et taux d'erreur = 0.0004696175031272398
On remarque que notre modèle classifie donc très bien les données.

Pour visualiser plus simplement le résultat de la classification, nous avons affiché la matrice de confusion du modèle.



On se rend compte qu'effectivement, 24 des 101 opérations frauduleuses (classe 1) issues du groupe test ont été classées comme des opérations non frauduleuses (classe 0).
Ainsi, à peu près 24% des opérations frauduleuse n'ont pas été détectées ce qui confirme que le modèle a du mal à généraliser. L'hypothèse de sur-apprentissage est donc confirmée.

Dans le contexte d'une banque, il est intéressant de réduire le nombre de cas réellement frauduleux classés non frauduleux.

En effet, comme dit dans le sujet du projet, il est important que les sociétés de cartes de crédit soient en mesure de reconnaître les transactions frauduleuses afin que les clients ne soient pas facturés pour les articles qu'ils n'ont pas achetés.

Ainsi, détecter un cas comme frauduleux alors qu'il ne l'est pas n'a pas le même impact que de détecter un cas non frauduleux qui en réalité en est un.

Pour cela, le plus adapté est d'utiliser la mesure **recall-score** qui représentent 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 : **miss_rate =**1 - recall-score.

Ici, recall-score = 0.7623762376237624 et miss_rate = 0.2376237623762376

Apprentissage de l'algorithme Random Forest sur des données sous échantillonné:

Nous avons sous échantillonner les données (*under-sampling* grâce au package *imblearn*) pour entrainer un nouveau Random Forest.

Dans la « vraie vie », les cas de fraudes ne sont pas aussi nombreux que les cas de « non fraudes ».
C'est pourquoi, pour la suite des questions, nous prenons les résultats du modèle Random Forest entrainé (fit) sur des données sous échantillonnées (984 lignes aux total) mais testé sur des données avec la répartition d'origine qui permet de capturer plus fidèlement la répartition des classes et d'éviter le sur-apprentissage.

Pour la suite des questions, nous évaluerons les différents arbres issus de ce modèle avec ces quatre mesures (accuracy, taux d'erreur, recall-score, miss_rate)

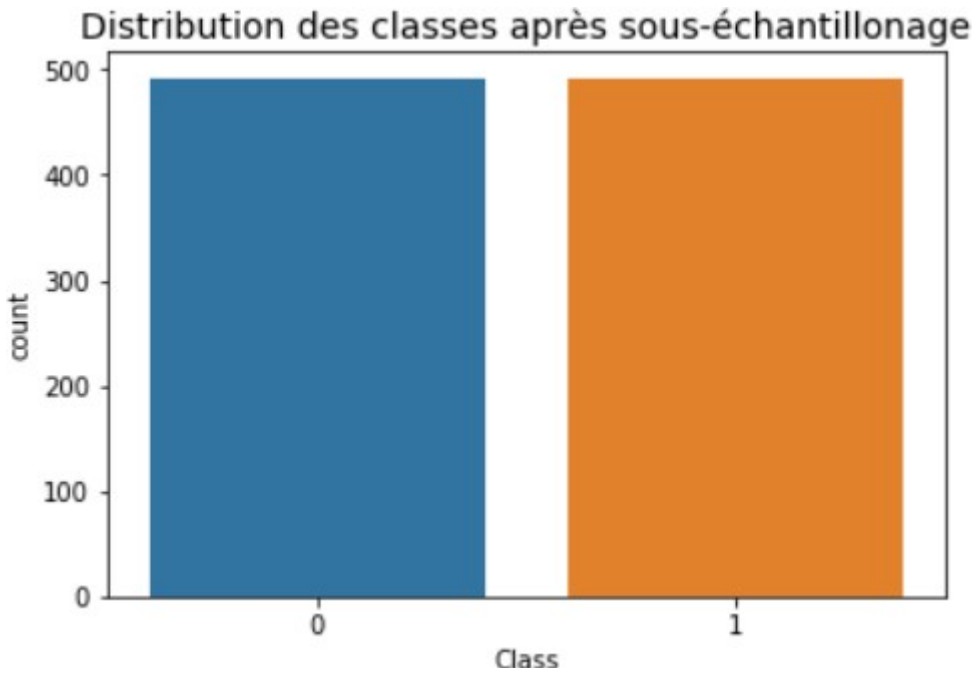
Attention : Pour ce projet, notre objectif est de minimiser la valeur de miss_rate (nombre de faux négatifs = cas frauduleux classés non frauduleux).
Nous ne nous basons pas sur l'accuracy ou le taux d'erreur pour le GridSearch. Ainsi, notre meilleur modèle, calculé grâce au recall-score, aura une moins bonne accuracy que si on avait fait un GridSearch basé sur l'accuracy.

Question 1 :

Afin de pouvoir utiliser l'algorithme Random Forest pour la classification des transactions en deux catégories, frauduleuse (1) et non frauduleuse (0) il faut avoir un peu près la même répartition des classes 0 et 1. Comme remarqué dans l'analyse, le jeu de données est mal distribué.

En effet, la classe 0 étant trop importante, si l'algorithme apprend des données telles quelle sont, il y aura un phénomène de sur-apprentissage (*over-fitting*) qui prédira toujours la classe 0 et jamais la classe 1.

Pour résoudre ce problème, nous appliquerons une fonction qui prend aléatoirement les données dans la classe 1 et 0 afin qu'il y en ait le même nombre. C'est ce qu'on appelle un sous-échantillonnage.



Les pré-traitements obligatoires avant d'appliquer l'algorithme de Random Forest sont de séparer les données en deux groupes, un groupe contenant toutes les colonnes (DataFrame : train) sauf la colonne Class que nous séparerons dans un autre groupe (DataFrame: y) car c'est la variable cible que nous voulons prédire.

Puis nous avons utilisé la fonction *train_test_split* de *Sklearn* pour avoir nos groupes de données d'entrainement et de test, pour évaluer la classification.

Question 2 :

a) Nous avons utilisé GridSearch pour optimiser les paramètres afin d'avoir le meilleur recall-score et donc le taux miss_rate minimum sur le classificateur Random Forest.

b) Les paramètres optimaux sont :

```
best_clf2 = RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                criterion='entropy', max_depth=9, max_features='auto',
                max_leaf_nodes=None, max_samples=None,
                min_impurity_decrease=0.0, min_impurity_split=None,
                min_samples_leaf=3, min_samples_split=5,
                min_weight_fraction_leaf=0.0, n_estimators=10,
                n_jobs=None, oob_score=False, random_state=None,
                verbose=0, warm_start=False)
```

c) Les paramètres que nous avons testés pour le GridSearch sont :

Le nombre d'arbres générés : n_estimators de 10 à 100

La profondeur maximale de l'abre : max_depth de 1 à 10, avec un pas de 2 en plus du None.

Le nombre minimum d'exemples requis pour diviser un nœud : min_samples_split = de 1 à 10 avec un pas de 2, en plus du None (valeur par défaut) .

Le nombre minimum d'exemples dans un feuille : min_samples_leaf de 1) 10 avec un pas de 2, en plus du None (valeur par défaut).

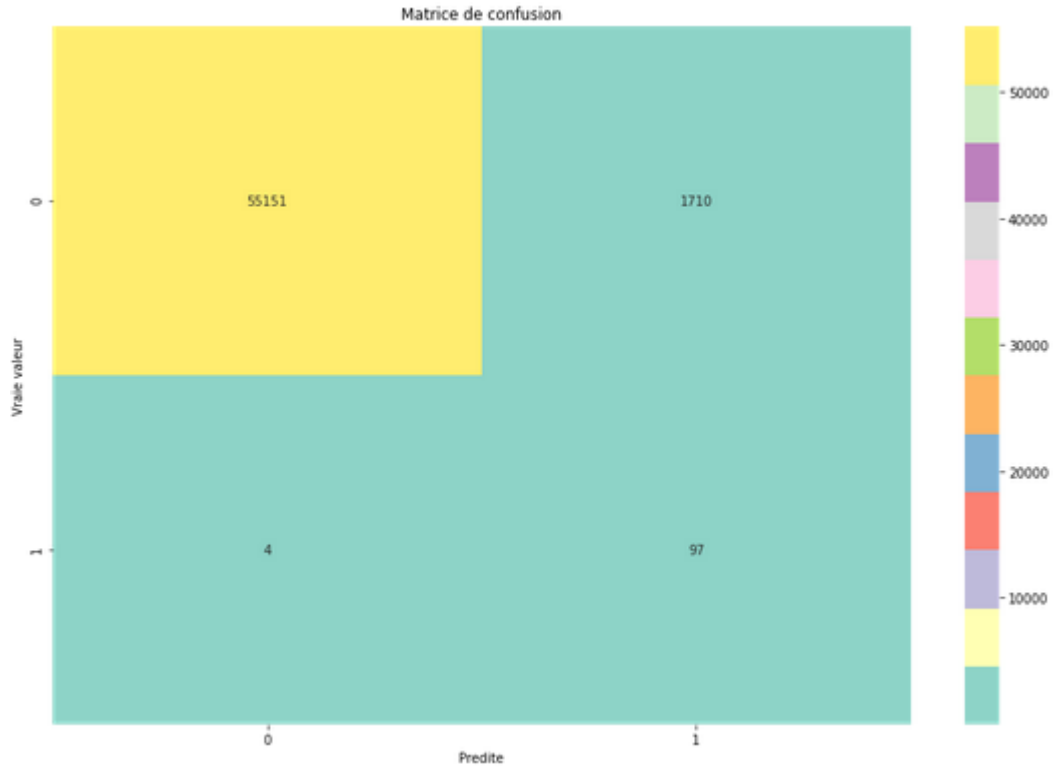
Le fait que des échantillons de bootstrap soient utilisés pour la construction d'arbres. Si bootstrap=False, l'ensemble du jeu de données est utilisé pour construire chaque arbre.: bootstrap = [True, False]

La mesure de qualité d'une division. Les critères sont "gini" pour l'impureté de Gini et l'entropie" pour le gain d'information : criterion = ['gini', 'entropy']

Question 3 :

Pour Random Forest avec les meilleurs paramètres, entrainé sur un sous-échantillon des données et testé sur les données non échantillonné, les résultats d'évaluations sont :

Accuracy: 0.9699097644043397
Taux d'erreur : 0.030090235595660264
Recall score : 0.9603960396039604
miss-rate : 0.03960396039603964



Il y seulement 4 opérations frauduleuses sur 101 qui ont était classé comme non frauduleuses.
Notre modèle entrainé sur les données sous-échantillonnées reconnait beaucoup mieux (4% contre 24%) les opérations frauduleuses.

Question 4:

Le meilleur classifier de notre modèle est le : 2e selon le score de rappel (recall score)

Son évaluation est :
Accuracy : 0.9138021839120818
Taux d'erreur : 0.08619781608791821

Recall score: 0.9801980198019802
Miss-rate: 0.01980198019801982

Question 5:

```
Estimator n° : 0 ,accuracy score : 0.9401004178224079 , recall sore :
0.9207920792079208 , taux-d'erreur : 0.05989958217759206 , miss-rate
0.07920792079207917

Estimator n° : 1 ,accuracy score : 0.9181208524981567 , recall sore :
0.9603960396039604 , taux-d'erreur : 0.08187914750184333 , miss-rate
0.03960396039603964

Estimator n° : 2 ,accuracy score : 0.9138021839120818 , recall sore :
0.9801980198019802 , taux-d'erreur : 0.08619781608791821 , miss-rate
0.01980198019801982

Estimator n° : 3 ,accuracy score : 0.9336575260700116 , recall sore :
0.9405940594059405 , taux-d'erreur : 0.06634247392998838 , miss-rate
0.05940594059405946

Estimator n° : 4 ,accuracy score : 0.9038657350514377 , recall sore :
0.9702970297029703 , taux-d'erreur : 0.09613426494856225 , miss-rate
0.02970297029702973

Estimator n° : 5 ,accuracy score : 0.9294617464274428 , recall sore :
0.9306930693069307 , taux-d'erreur : 0.07053825357255716 , miss-rate
0.06930693069306926

Estimator n° : 6 ,accuracy score : 0.9243881886169727 , recall sore :
0.9306930693069307 , taux-d'erreur :  0.0756118113830273 , miss-rate
0.06930693069306926

Estimator n° : 7 ,accuracy score : 0.9159439626417611 , recall sore :
0.9306930693069307 , taux-d'erreur : 0.08405603735823886 , miss-rate
0.06930693069306926

Estimator n° : 8 ,accuracy score : 0.8914715073206699 , recall sore :
0.9207920792079208 , taux-d'erreur : 0.10852849267933007 , miss-rate
0.07920792079207917

Estimator n° : 9 ,accuracy score : 0.87651416733963 , recall sore :
0.9702970297029703 , taux-d'erreur :  0.12348583266037005 , miss-rate
0.02970297029702973
```

Question 6 :

Le moins bon classifier de notre modèle est le 0 selon le score de rappel (recall score)

Son évaluation est :
Accuracy : 0.9401004178224079
Taux d'erreur : 0.05989958217759206
Recall score : 0.9207920792079208
Miss-rate : 0.07920792079207917

Le meilleur classifier de notre modèle est le : 2e selon le score de rappel (recall score)

Son évaluation est :
Accuracy : 0.9138021839120818
Taux d'erreur : 0.08619781608791821
Recall score: 0.9801980198019802
Miss-rate: 0.01980198019801982

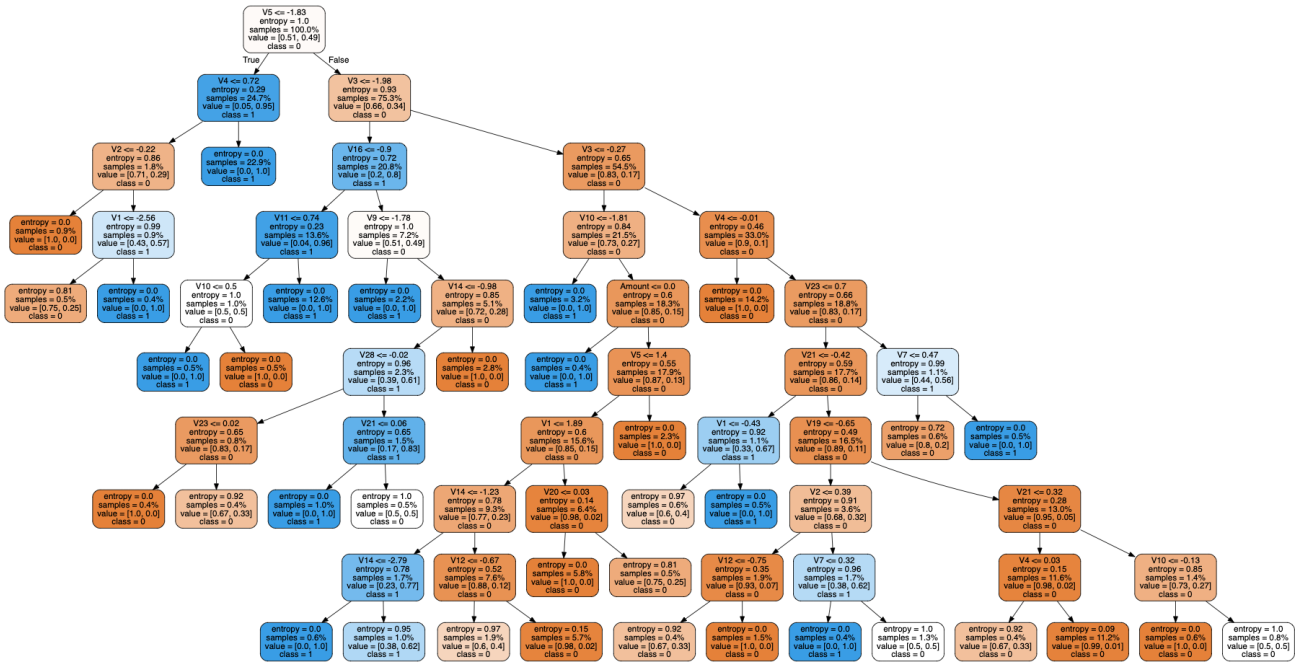
Question 7 :

moy = (0.07920792079207917+0.01980198019801982)/2
diff = moy - 0.01980198019801982

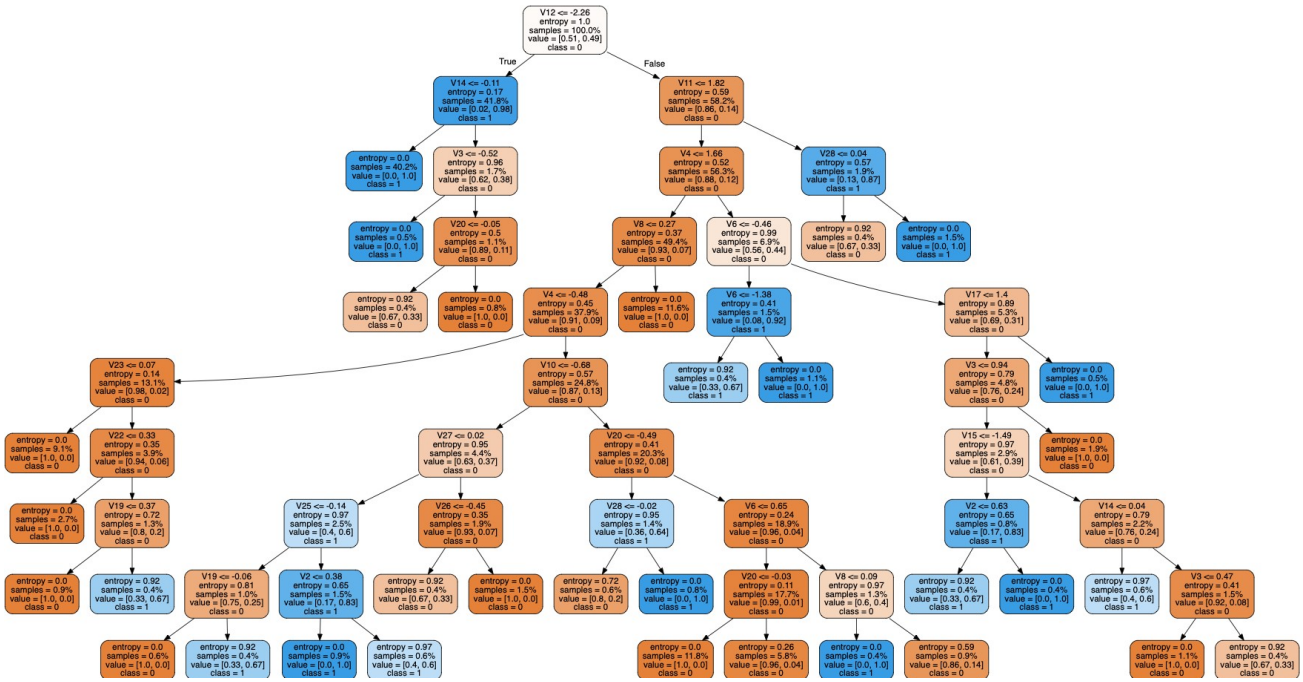
La différence est : 0.029702970297029674

Question 8 :

Estimateur n°0, le plus mauvais du Random Forest :



Estimateur n°2, le meilleur de notre modèle Random Forest :



Question 9 :

Pour le moins performant :

```
1 ) V5 <= -1.83  & V4 <= 0.72  & V2 <= -0.22  --> 0 ( 100 % )
2 ) V5 <= -1.83  & V4 <= 0.72  & V2 > -0.22 & V1 <= -2.56  --> 0 ( 75 % )
3 ) V5 <= -1.83  & V4 <= 0.72  & V2 > -0.22 & V1 > -2.56 --> 1 ( 100.0 % )
4 ) V5 <= -1.83  & V4 > 0.72 --> 1 ( 100 % )
5 ) V5 > -1.83 & V3 <= -1.98 & V16 <= -0.90  & V11 <= 0.74  & V10 <= 0.50
--> 1 ( 100 % )
6 ) V5 > -1.83 & V3 <= -1.98  & V16 <= -0.90  & V11 <= 0.74  & V10 > 0.50
--> 0 ( 100 % )
7 ) V5 > -1.83 & V3 <= -1.98  & V16 <= -0.90  & V11 > 0.74 --> 1 ( 100 % )
8 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 <= -1.78  --> 1 ( 100 % )
9 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 > -1.78 & V14 <= -0.98 & V2
8 <= -0.02  & V23 <= 0.02  --> 0 ( 100 % )
10 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 > -1.78 & V14 <= -0.98  &
V28 <= -0.02  & V23 > 0.02 --> 0 ( 67 % )
11 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 > -1.78 & V14 <= -0.98  &
V28 > -0.02 & V21 <= 0.06  --> 1 ( 100 % )
12 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 > -1.78 & V14 <= -0.98  &
V28 > -0.02 & V21 > 0.06 --> 0 ( 50 % )
13 ) V5 > -1.83 & V3 <= -1.98  & V16 > -0.90 & V9 > -1.78 & V14 > -0.98 -->
0 ( 100 % )
14 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 <= -1.81  --> 1 ( 100 % )
15 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount <= 0.0
 --> 1 ( 100 % )
16 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 <= 1.89  & V14 <= -1.23  & V14 <= -2.79  --> 1 ( 100 % )
17 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 <= 1.89  & V14 <= -1.23  & V14 > -2.79 --> 1 ( 62 % )
```

```
18 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 <= 1.89  & V14 > -1.23 & V12 <= -0.67  --> 0 ( 60 % )
19 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 <= 1.89  & V14 > -1.23 & V12 > -0.67 --> 0 ( 98 % )
20 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 > 1.89 & V20 <= 0.03  --> 0 ( 100 % )
21 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 <= 1.4  & V1 > 1.89 & V20 > 0.03 --> 0 ( 75 % )
22 ) V5 > -1.83 & V3 > -1.98 & V3 <= -0.27  & V10 > -1.81 & Amount > 0.0  &
V5 > 1.4 --> 0 ( 100 % )
23 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 <= -0.01  --> 0 ( 100 % )
24 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 <
= -0.42  & V1 <= -0.43  --> 0 ( 60 % )
25 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 <
= -0.42  & V1 > -0.43 --> 1 ( 100 % )
26 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 <= -0.65  & V2 <= 0.39  & V12 <= -0.75  --> 0 ( 67 % )
27 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 <= -0.65  & V2 <= 0.39  & V12 > -0.75 --> 0 ( 100 % )
28 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 <= -0.65  & V2 > 0.39 & V7 <= 0.32  --> 1 ( 100 % )
29 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 <= -0.65  & V2 > 0.39 & V7 > 0.32 --> 0 ( 50 % )
30 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 > -0.65 & V21 <= 0.32  & V4 <= 0.03  --> 0 ( 67% % )
31 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 > -0.65 & V21 <= 0.32  & V4 > 0.03 --> 0 ( 99 % )
32 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 > -0.65 & V21 > 0.32 & V10 <= -0.13  --> 0 ( 100 % )
33 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 <= 0.7  & V21 >
 -0.42 & V19 > -0.65 & V21 > 0.32 & V10 > -0.13 --> 0 ( 50 % )
34 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 > 0.7  & V7 <=
0.47  --> 0 ( 80 % )
35 ) V5 > -1.83 & V3 > -1.98 & V3 > -0.27 & V4 > -0.01 & V23 > 0.7  & V7 > 0
.47 --> 1 ( 100 % )
```

Pour le plus performant :

```
1 ) V12 <= -2.26  & V14 <= -0.11  --> 1 ( 100 % )
2 ) V12 <= -2.26  & V14 > -0.11 & V3 <= -0.52  --> 1 ( 100 % )
3 ) V12 <= -2.26  & V14 > -0.11 & V3 > -0.52 & V20 <= -0.05 --> 0 ( 67 % )
4 ) V12 <= -2.26  & V14 > -0.11 & V3 > -0.52 & V20 > -0.05 --> 0 ( 100 % )
5 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 <= -0.48  &
V23 <= 0.07  --> 0 ( 100 % )
6 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 <= -0.48  &
V23 > 0.07 & V22 <= 0.33  --> 0 ( 100 % )
7 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 <= -0.48  &
V23 > 0.068 & V22 > 0.33 & V19 <= 0.37  --> 0 ( 100 % )
8 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 <= -0.48  &
V23 > 0.068 & V22 > 0.33 & V19 > 0.37 --> 1 ( 67 % )
9 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48 & V1
0 <= -0.68  & V27 <= 0.02  & V25 <= -0.14  & V19 <= -0.06  --> 0 ( 100 % )
10 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 <= -0.68  & V27 <= 0.02  & V25 <= -0.14  & V19 > -0.06 --> 1 ( 67 % )
11 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 <= -0.68  & V27 <= 0.02  & V25 > -0.14 & V2 <= 0.38  --> 1 ( 100 % )
12 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 <= -0.68  & V27 <= 0.02  & V25 > -0.14 & V2 > 0.38 --> 1 ( 60 % )
```

```
13 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 <= -0.68  & V27 > 0.02 & V26 <= -0.45  --> 0 ( 67 % )
14 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 <= -0.68  & V27 > 0.02 & V26 > -0.45 --> 0 ( 100 % )
15 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66& V8 <= 0.27    & V4 > -0.48  &
V10 > -0.68 & V20 <= -0.49  & V28 <= -0.02  --> 0 ( 80 % )
16 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 > -0.68 & V20 <= -0.49  & V28 > -0.02 --> 1 ( 100 % )
17 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 > -0.68 & V20 > -0.49 & V6 <= 0.65  & V20 <= -0.03  --> 0 ( 100 % )
18 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 > -0.68 & V20 > -0.49 & V6 <= 0.65  & V20 > -0.03 --> 0 ( 96 % )
19 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 > -0.68 & V20 > -0.49 & V6 > 0.65 & V8 <= 0.09  --> 1 ( 100 % )
20 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 <= 0.27  & V4 > -0.48  &
V10 > -0.68 & V20 > -0.49 & V6 > 0.65 & V8 > 0.09 --> 0 ( 86 % )
21 ) V12 > -2.26 & V11 <= 1.82  & V4 <= 1.66  & V8 > 0.27 --> 0 ( 100 % )
22 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 <= -0.46  & V6 <= -1.38  --
> 1 ( 67 % )
23 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 <= -0.46  & V6 > -1.38 -->
1 (100 % )
24 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
<= 0.94  & V15 <= -1.49  & V2 <= 0.63  --> 1 ( 67 % )
25 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
<= 0.94  & V15 <= -1.49  & V2 > 0.63 --> 1 ( 100 % )
26 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
<= 0.94  & V15 > -1.49 & V14 <= 0.04  --> 1 ( 60 % )
27 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
<= 0.94  & V15 > -1.49 & V14 > 0.04 & V3 <= 0.47  --> 0 ( 100 % )
28 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
<= 0.94  & V15 > -1.49 & V14 > 0.04 & V3 > 0.47 --> 0 ( 67 % )
29 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 <= 1.4  & V3
> 0.94 --> 0 ( 100 % )
30 ) V12 > -2.26 & V11 <= 1.82  & V4 > 1.66 & V6 > -0.46 & V17 > 1.4  --> 1
( 100 % )
31 ) V12 > -2.26 & V11 > 1.82 & V28 <= 0.04  --> 0 ( 67 % )
32 ) V12 > -2.26 & V11 > 1.82 & V28 > 0.04 --> 1 ( 100 % )
```

Question 10 :

Le mauvais estimateur devrait changer de nœud racine (V5 à V12), les autres règles découlent de ce nœud. De plus, le meilleur estimateur a des règles que le pire n'a pas (V8,V22,V27,V25,V26,V6,V17 et V15)

Apprentissage de l'algorithme Random Forest sur des données sous échantillonné qui maximise l'accuracy: (joint dans le jupyter notebook : best_accuracy.ipynb)

Si malgré tout, nous voulons maximiser l'accuracy et donc avoir le taux d'erreur le plus bas voici les paramètres identifiés avec GridSearch.

Les paramètres du meilleur classifieur qui maximise l'accuracy sont :

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=9, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, n_estimators=99,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Son évaluation est :
Accuracy : 0.9881148836066149
Taux d'erreur : 0.011885116393385076
Recall score : 0.9603960396039604
Miss-rate : 0.03960396039603964