République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumédiène

Faculté d'Informatique
Département d'Intelligence Artificielle

Master 2 Systèmes Informatiques intelligents

Module : Data Mining

Rapport de projet, partie 4
Classification et Prédiction

Réalisé par :
BOUROUINA Rania, 181831052716
CHIBANE Ilies, 181831072041

Année universitaire : 2022 / 2023

Table des matières

Introduction Générale

Classification and Regression Trees ou CART en abrégé est un acronyme introduit par Leo Breiman pour désigner les algorithmes d'arbres décisionnels qui peuvent être utilisés pour les problèmes de modélisation prédictive de classification ou de régression.

Régression : La fonction de coût qui est minimisée pour choisir les points de séparation est la somme des erreurs au carré sur tous les échantillons d'apprentissage.

Classification : La fonction de coût est utilisée pour fournir une indication de la pureté des nœuds, la pureté des nœuds faisant référence à la mixité des données d'apprentissage attribuées à chaque nœud.

Dans cette partie de projet, nous allons nous concentrer sur l'utilisation de CART pour la classification.

Cette technique utilise des algorithmes mathématiques sophistiqués pour classer, diviser, segmenter l'ensemble des données, les pré-traiter si nécessaire et évaluer la possibilité d'événements futurs. La représentation du modèle CART est un arbre binaire. Il s'agit du même arbre binaire que celui des algorithmes et des structures de données. Une fois créé, un arbre peut être parcouru avec une nouvelle ligne de données suivant chaque branche avec les divisions jusqu'à ce qu'une prédiction finale soit faite.

Dans cette partie qui est consacrée à la classification et prédiction, nous allons commencé par créer des modèles personalisables (Arbre de Décision et Forêt aléatoire). Nous allons ensuite procédé à la prédiction en utilisant des données de test.

Prétraitement des données

Introduction Afin de pouvoir appliquer des modèles de classification sur notre jeu de données, il est impératif d'y opérer un prétraitement afin d'obtenir des résultats corrects et précis. C'est pour cela que nous allons appliquer les différentes méthodes de prétraitements vu dans la partie 2 du projet pour obtenir un dataset en parfaite condition pour y appliquer nos algorithmes d'apprentissage automatique.

1.1 Codification

La première étape consiste à codifier les données textuelle catégorique de notre dataset. Pour cela, nous optons simplement pour une codification ordinale de 0 à n, n étant le nombre de classes de nos données. Obtenant les résultats suivants :

Attrition : [Yes, No] → [0, 1]
Gender : [Male, Female] → [0, 1]
BusinessTravel : [Travel_Rarely, Travel_Frequently, Non_Travel] → [0, 1, 2]

1.2 Traitement des valeurs manquantes

Les valeurs manquantes sont très gênantes dans le cadre de la classification, c'est pour cela que nous nous devons d'y remédier. Notre jeu donné heureusement ne contient pas beaucoup de valeurs manquantes, 21 au total. Treize dans l'attribut EnvironmentSatisfaction qui étant une valeur catégorique, allons remplacer ces dites valeurs par le mode étant donné la moyenne et la médiane n'ont adapté aux valeurs catégoriques. Et sept dans l'attribut MonthlyIncome que nous allons remplacer par la médiane qui est en général la mesure de tendance la plus populaire en ce qui concerne le traitement des valeurs manquantes dans les attributs continue.

1.3 Traitement des valeurs aberrantes

Les valeurs aberrantes peuvent s'avérer gênante dans le cadre de la classification. Mais dans certains cas, elles représentent des données réelles pertinentes. Et il est préférable de les conserver.

Dans notre cas, nous avons identifié les valeurs aberrantes suivantes comme étant problématiques :

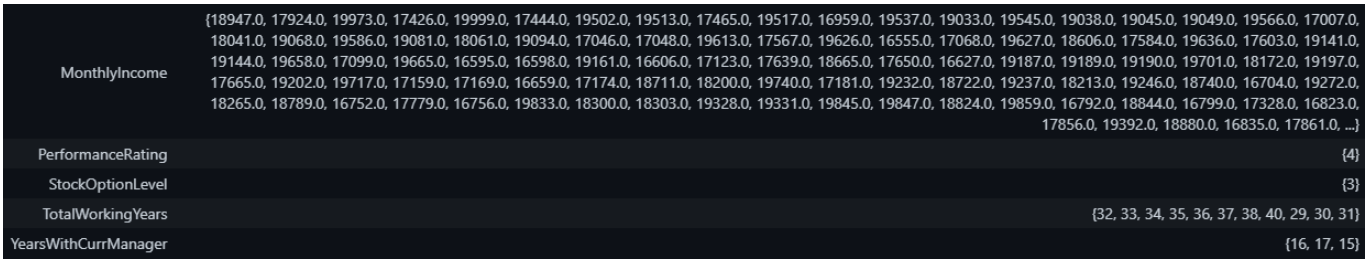


FIGURE 1.1 – Valeurs aberrantes à traiter

Il existe de nombreuses manières de traiter les outliers et nous allons appliquer plusieurs d'entre elle au cas par cas. Pour commencer, les attributs PerformanceRating et StockOptionLevel ne contenant qu'un outlier respectivement, nous les remplacerons simplement par la médiane. En ce qui l'attribut YearsWithCurrentManager, nous appliquerons un remplacement des valeurs par celle de l'IQR. Enfin, les attrbuts avec le plus de valeurs aberrantes MonthlyIncome et TotalWorkingYears ont été traités via la discrétisation en utilisant la formule suivante :

$$new\_val = \frac{(max - min + degre\_de\_prcision)}{k}$$

avec $K = 1\frac{10}{3} * log(n, base = 10)$

Une fois fait, le problème des valeurs aberrantes est réglé.

1.4 Discrétisation Lissage des données

La discrétisation est très utile dans notre cas, car elle nous permettra de voir les instances à supprimer lors de la réduction de la dimensionnalité H. Pour cela, nous procédant de la manière suivante :

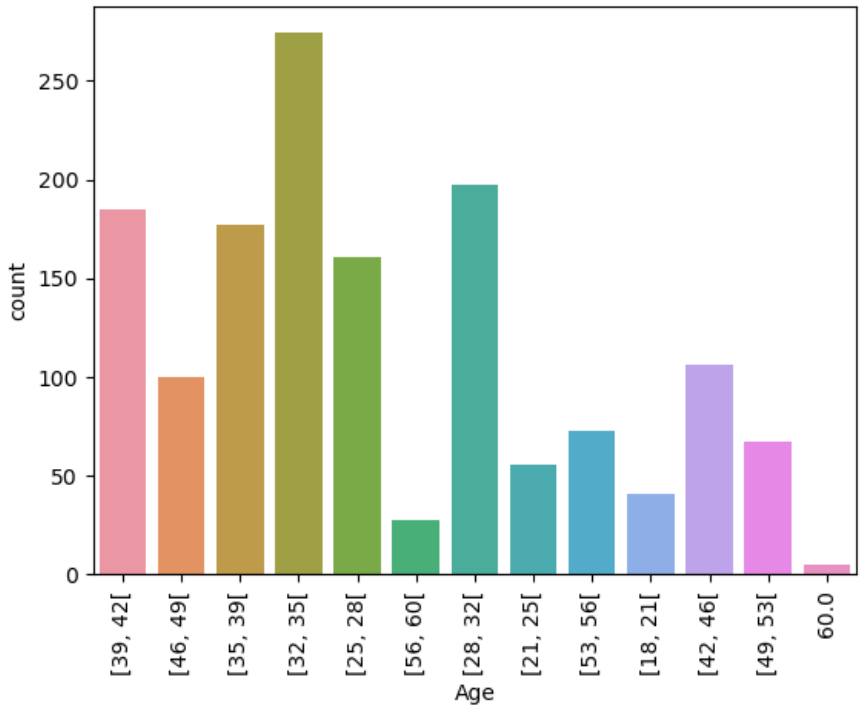
$$w = \frac{(max - min + degre\_de\_prcision)}{k}$$

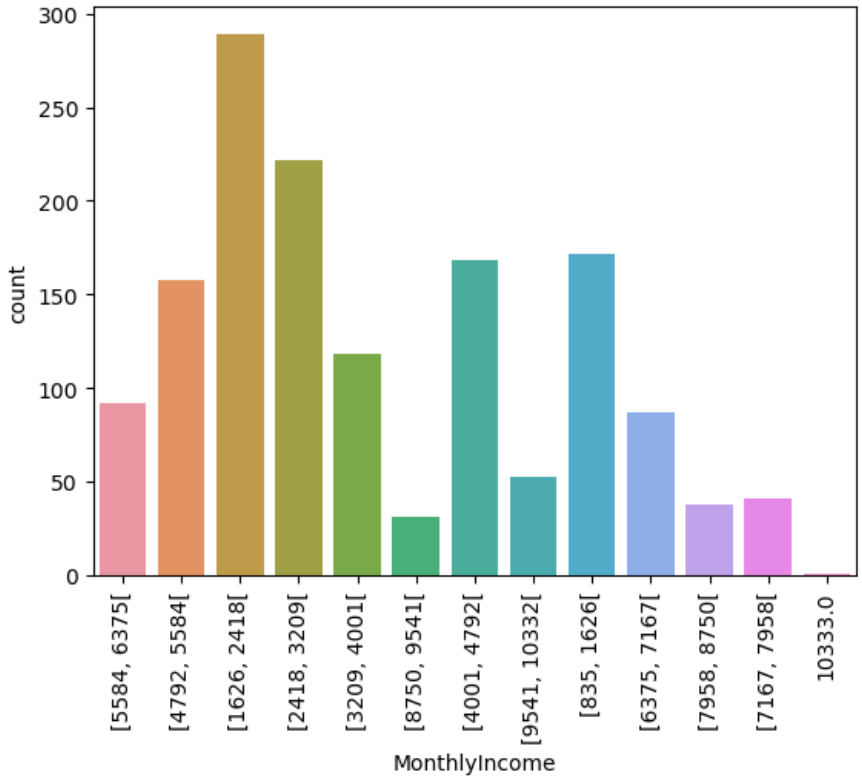
avec $K = 1\frac{10}{3} * log(n, base = 10)$

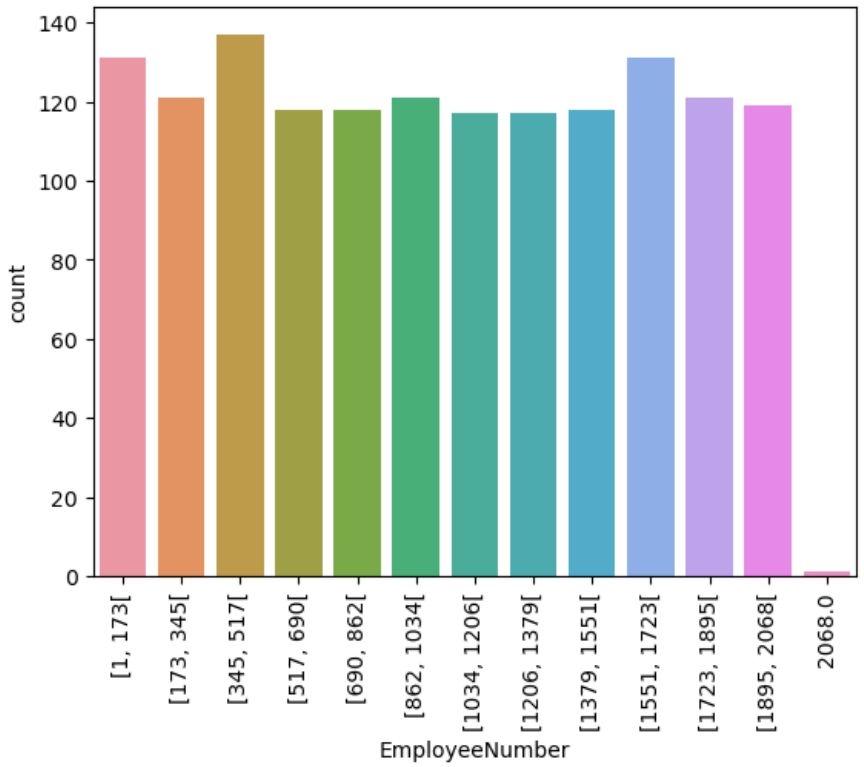
ce qui nous permettra d'obtenir les intervalles suivants :

$$[min, min + w[, [min + w, min + 2w[, \ldots, [min + (k-1) * w, min + k * w[.$$

Une fois les intervalles créée, nous pouvons repérer certaines anomalies lors de la visualisation, quelques exemples montrés ci-dessous :


count
250
200
150
100
50
0
[39, 42[
[46, 49[
[35, 39[
[32, 35[
[25, 28[
[56, 60[
[28, 32[
[21, 25[
[53, 56[
[18, 21[
[42, 46[
[49, 53[
60.0
Age


count
300
250
200
150
100
50
0
[5584, 6375[
[4792, 5584[
[1626, 2418[
[2418, 3209[
[3209, 4001[
[8750, 9541[
[4001, 4792[
[9541, 10332[
[835, 1626[
[6375, 7167[
[7958, 8750[
[7167, 7958[
10333.0
MonthlyIncome



Comme on peut facilement le remarquer, il existe des valeurs externes aux intervalles. Les instances comportant ces valeurs seront supprimer lors de la réduction des données.

1.5 Réduction de la dimensionnalité

Grace aux observations de la partie 1,du projet, nous pouvons aisément détecter certains attributs supprimables de note jeu de données tel que EmployeeCount, Over18, StandardHours, et PerformanceRating (même valeur partout) et EmployeeNumber étant un attribut non pertinent. Tous ces attributs sont supprimés ainsi que les instances contenant des anomalies détectées lors de la discrétisation.

1.6 Normalisation

La normalisation est essentielle afin d'avoir tous nos attribue continue sur la même échelle et pour cela, nous utilisons la formule Min-Max suivante :

$$X' = Min_{new} + \frac{(X - Min_{new})(Max_{new} - Min_{new})}{Max_{old} - Min_{old}}$$

Une fois appliqué, nous pouvons apercevoir les résultats sur un échantillion des attributs sur lesquelles la normalisation fut effectuée :

Age	DailyRate	DistanceFromHome	MonthlyIncome	MonthlyRate
0.56	0.72	0.00	0.54	0.70
0.76	0.13	0.26	0.45	0.92
0.46	0.91	0.04	0.13	0.01
0.37	0.92	0.07	0.22	0.85
0.34	0.65	0.04	0.24	0.39
...	...	...	...	...
0.20	0.76	0.15	0.22	0.77
0.44	0.56	0.81	0.18	0.41
0.51	0.37	0.19	0.96	0.78
0.22	0.04	0.11	0.56	0.12
0.76	0.66	0.04	0.48	0.45

On constate que les valeurs sont toutes dans l'intervalle [0, 1], nos attributs sont donc tous sur la même échelle.

Après ça, nous avons fini le prétraitement des données et nous nous retrouvons avec un dataset parfaitement optimisé afin de réaliser la classification en utilisant des modèles de machine learning plus précisément Décisions trees et Random Forest.

Notions et choix

Introduction La création d'un arbre de décision est un processus de division de l'espace d'entrée. L'approche utilisée est la division binaire récursive. Il s'agit d'une procédure numérique dans laquelle toutes les valeurs sont alignées et différents points de division sont testés à l'aide d'une fonction de coût.
Le fractionnement se poursuit jusqu'à ce que les nœuds contiennent un nombre minimal d'exemples d'apprentissage ou qu'une profondeur maximale de l'arbre soit atteinte.
La foret aléatoire est ensuite générée à partir de plusieurs variants de l'arbre de décision.
C'est pour celà que dans ce chapitre, nous allons découvrir le processus de création de ses modèles en introduisant les nouvelles notions et justifiant à chaque fois les choix.

2.1 Arbre de décision

Lors de l'apprentissage d'un modèle sur les relations caractéristique-cible, un arbre est développé à partir d'un nœud racine (parent) (toutes les données contenant des relations caractéristique-cible), qui est ensuite divisé de manière récursive en nœuds enfants (sous-ensemble de l'ensemble des données) de manière binaire.

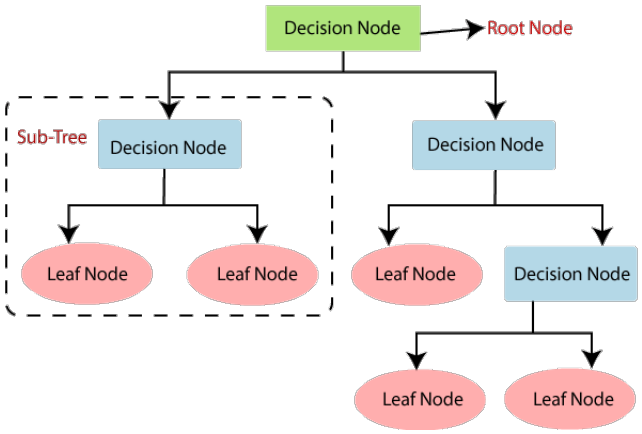


FIGURE 2.1 – Arbre de décision

Chaque division est effectuée sur une seule caractéristique du nœud parent, à une valeur seuil

souhaitée de la caractéristique. Par exemple, lors de chaque division du nœud parent, nous passons au nœud de gauche (avec le sous-ensemble de données correspondant) si une caractéristique est inférieure au seuil, et au nœud de droite sinon. Mais comment décider de la division ? Nous utilisons une fonction de cout.

2.1.1 La fonction de cout

Comment chaque critère trouve-t-il la répartition optimale ? Et quelles sont les différences entre ces deux critères ? Dans cette partie, nous allons répondre à ces questions. Tout d'abord, nous expliquons les critères de gini et d'entropie et leurs différences, puis nous présentons un exemple pratique qui compare les deux critères.

Gini Impurity

L'impureté gini est calculée à l'aide de la formule suivante :

$$GiniIndex = 1 - \sum_j p_j^2$$

Où p_j est la probabilité de la classe j. L'impureté de Gini mesure la fréquence à laquelle un élément de l'ensemble de données sera mal étiqueté lorsqu'il est étiqueté de manière aléatoire.
La valeur minimale de l'indice de Gini est de 0. Cela se produit lorsque le nœud est pur, ce qui signifie que tous les éléments contenus dans le nœud sont d'une classe unique. Par conséquent, ce nœud ne sera pas divisé à nouveau. Ainsi, la division optimale est choisie par les caractéristiques ayant le plus faible indice de Gini. De plus, il obtient la valeur maximale lorsque la probabilité des deux classes est la même.

$$Gini_{min} = 1 - (1^2) = 0$$

$$Gini_{max} = 1 - (0.5^2 + 0.5^2) = 0.5$$

Entropy

L'entropie est calculée à l'aide de la formule suivante :

$$Entropy = -\sum_j p_j \cdot log_2 \cdot p_j$$

Où, comme précédemment, p_j est la probabilité de la classe j. L'entropie est une mesure d'information qui indique le désordre des caractéristiques par rapport à la cible. Comme pour l'indice de Gini, la répartition optimale est choisie par la caractéristique ayant la plus faible entropie. Elle obtient sa valeur maximale lorsque la probabilité des deux classes est la même et un nœud est pur

lorsque l'entropie a sa valeur minimale, qui est 0 :

$$Entropy_{min} = -1 \cdot log_2(1) = 0$$

$$Entropy_{max} = -0.5 \cdot log_2(0.5) - 0.5 \cdot log_2(0.5) = 1$$

Gini vs Entropy

L'indice de Gini et l'entropie présentent deux différences principales :
– L'indice de Gini a des valeurs comprises dans l'intervalle [0, 0.5] alors que l'intervalle de l'entropie est [0, 1]. Dans la figure suivante, les deux sont représentés. L'indice de Gini a également été représenté multiplié par deux pour voir concrètement les différences entre eux.

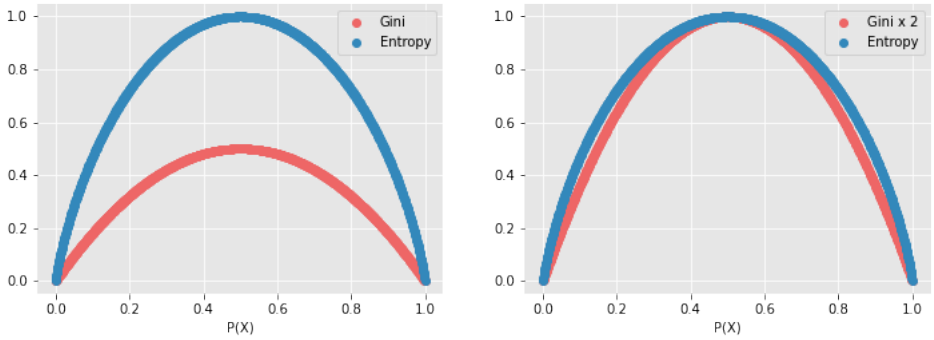


FIGURE 2.2 – Gini vs Entropy

–Du point de vue computationnel, l'entropie est plus complexe puisqu'elle fait appel à des logarithmes et, par conséquent, le calcul de l'indice de Gini sera plus rapide.
NB : Nous allons voir la différence concrètement dans l'étude experimentale.

2.2 Foret Aléatoire

La Foret Aléatoire, est une collection d'arbres décisionnels.
L'algorithme Random Forest est construit sur l'idée du vote par des apprenants "faibles" (arbres décisionnels), ce qui donne l'analogie avec les arbres qui composent une forêt. L'élément aléatoire comporte quelques aspects :

–Chaque arbre est ajusté sur un sous-ensemble de l'ensemble des données, donc chaque arbre sera développé différemment, avec des règles différentes.
–Si l'ensemble des données est amorcé de façon aléatoire avec remplacement, chaque arbre aura toujours une distribution légèrement différente des données, et donc sera développé différemment,

avec des règles différentes.
–Même si l'ensemble des données est amorcé sans remplacement, chaque arbre peut encore être développé différemment, en raison de l'ordre aléatoire ou du sous-ensemble des caractéristiques prises en compte pour une division optimale dans l'arbre de décision.

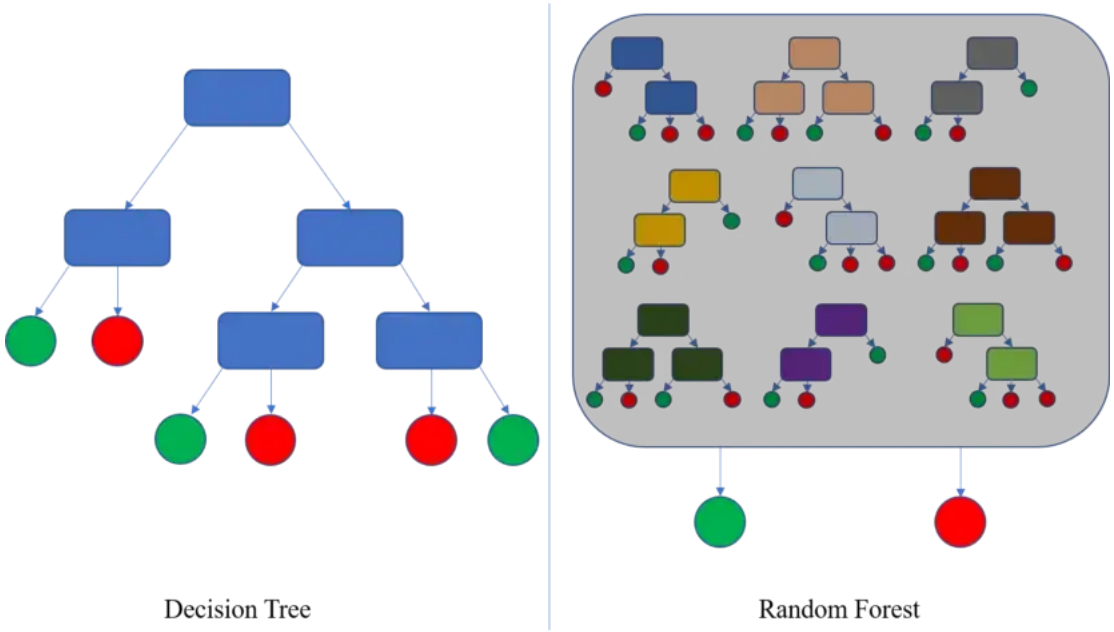


FIGURE 2.3 – Decision Tree vs Random Forest

En résumé,

–Pendant l'entrainement, N ensembles de données choisies aléatoirement sont obtenus séquentiellement à partir de l'ensemble des données.
–Chaque arbre de décision (N arbres de décision au total) est construit à partir de chaque ensemble de données choisies.
–Pendant l'inférence, une prédiction est faite par chaque arbre de décision, et la prédiction finale par la forêt aléatoire est retournée comme un vote majoritaire (majority vote).

Etude Experimentale

Introduction Dans ce chapitre, nous allons procéder à l'évaluation de nos modèles avec différentes métriques, une comparaison entre les hyperparamètres, des modèles sur plusieurs itérations et en fin, une comparaison avec les modèles de Sklearn.

3.1 Evaluation des modèles

3.1.1 Gini vs Entropy

Comme nous l'avons mentionnée auparavant, il existe deux choix de fonction de cout pour créer un arbre de décision (criterion), nous allons donc analyser les différents métriques avec chaque fonction de cout.

```
recall  :0.6926406926406926
 precision :0.764928738950027
 fpr  : 0.3073593073593074
 tnr  / Specificity  : 0.6926406926406926
 tpr  / Sensitivity : 0.764928738950027
fnr  : 0.23507126104997295
 F-score  : 0.7203389830508474
 Accuracy   : 0.44696969696969696
Execution time : 0.8073136806488037
```

FIGURE 3.1 – Gini cost function Résultats

```
recall  :0.5930735930735931
 precision :0.6390358612580835
 fpr  : 0.4069264069264069
 tnr  / Specificity  : 0.5930735930735931
 tpr  / Sensitivity : 0.6390358612580835
fnr  : 0.3609641387419165
 F-score  : 0.6080637599624942
 Accuracy   : 0.42803030303030304
Execution time : 1.1728205680847168
```

FIGURE 3.2 – Entropy cost function Résultats

La fonction Gini a donné les meilleurs résultats puisque c'est une fonction au calculs simples et adaptés à notre problèmatique.

(a) Decision Tree confusion Matrix (b) Random Forest confusion Matrix

FIGURE 3.3 – Les Matrices de Confusion de nos modèles

Dans cette exécution de nos modèles, la foret aléatoires a obtenu de meilleurs résultats (Nombre de TPs).

3.1.2 Arbre de décision vs Foret Aléatoire

A titre d'exemple, nous allons essayer les deux modèles avec les memes hyperparametres pour effectuer une comparaison :
Puisque la fonction de cout Gini a donnée de meilleurs résultats auparavant, nous allons la choisir pour entrainer n0s deux modèles. Les autres paramètres tels que La pronfondeur maximale sera mise à 5, et le nombre de caractéristiques utilisés va etre notre dataset au complet.

Modèle	Métrique	Valeur
Decision Tree		
	Accuracy	0.44696969696969696
	Précision/TPR/Sensitivity	0.764928738950027
	Recall/TNR/Specificity	0.6926406926406926
	FPR	0.3073593073593074
	FNR	0.23507126104997295
	FScore	0.7203389830508474
	Temps d'execution(s)	1.2717814445495605
Random Forest		
	Accuracy	0.42992424242424243
	Précision/TPR/Sensitivity	0.6549586776859504
	Recall/TNR/Specificity	0.6082251082251082
	FPR	0.3917748917748918
	FNR	0.3450413223140496
	FScore	0.624524312896406
	Temps d'execution(s)	6.4531309604644775

Théoriquement,
Les arbres de décision nécessitent peu de calculs, ce qui réduit le temps de mise en œuvre, tout en offrant une faible précision. Cependant les forets aléatoires consomment plus de calculs. Le processus de génération et d'analyse prend beaucoup de temps. Cette hypothèse est confirmée par notre étude expérimentale.
Par ailleurs, il est connu que les arbres de décision ne sont pas aussi précis que les forets aléatoires. En revanche, dans notre étude, le modèle de foret aléatoire est moins précis. Ceci est du au choix des hyperparamètres ainsi que le hasardisme du choix de l'échantillon de données.

Conclusion Générale

Dans cette partie du projet, nous nous sommes familiarisés avec le concept de classification, son importance dans le domaine du datamining, et le fonctionnement de certain modèle de classification. Tel que les arbres de décisions (decision tress) et l forêt d'arbres décisionnels (Random forest). Nous avons aussi pu constater l'importance du prétraitement des données vu lors de la partie deux, car sans elle notre classification n'aurait pas été des plus performantes.
Cependant, l'apport de la classification au datamining restant indéniable, elle reste limitée par le fait que les classes doivent être prédéfinies à l'avance. Classes qui ne peuvent pas toujours être fournis. C'est pourquoi il existe un autre sous domaine du machine learning appelé apprentissage non supervisé qui peut remédier à ce problème et que nous verrons plus en détail dans la prochaine et dernière partie de ce projet.