Introduction



[Machine Learning](http://www.journaldunet.com/solutions/dsi/machine-learning/)  en anglais, et « Apprentissage Automatique des machines » en Français, c’est quand les machines se mettent à apprendre comme les humains et non seulement exécuter les instructions contenues dans les algorithmes présents dans leurs processeurs.

Machine Learning est donc une discipline scientifique centrée sur le développement, l’analyse et l’implémentation de méthodes automatisables, qui offrent la possibilité à une machine d’évoluer grâce à un processus d’apprentissage.

La machine learning est une démarche empirique qui se base sur des données pour apporter une réponse à des problèmes, elle s’appuie souvent sur des algorithmes de data-mining connus depuis longtemps, souvent développés avant les années 2000. Hélas ! Leurs performances ont bien souvent été limitées par le manque de données disponibles ou de capacités informatiques pour traiter de larges volumes de données. C’est que certains problèmes de machine learning ne donnent parfois des résultats probants qu’à partir de plusieurs centaines de millions d’observations.

Aujourd’hui, la hausse des capacités de stockage et la profusion de données numériques qu’elles engendrent, couplées à des moyens de calcul informatique de plus en plus puissants, font revenir ces algorithmes sur le devant de la scène. Ayez donc conscience que les gros volumes de données ne sont pas un problème pour votre travail, mais au contraire une formidable opportunité d’y trouver des informations précieuses

Illustrons brièvement ce que peut faire la machine learning avec un cas simple, sans doute plus proche de notre quotidien : un filtre antispam. Dans un premier temps, on peut imaginer que la « machine » (votre service de messagerie) va « analyser » la façon dont vous allez classer vos mails entrants en spam ou pas. Grâce à cette période d’« apprentissage », la machine va déduire quelques grands critères de classification. Par exemple, la probabilité que la machine classe un mail en spam va augmenter si le mail contient des termes tels qu’« argent », « rencontre facile » ou « offre de rêve » et si l’expéditeur du mail n’est pas dans votre carnet d’adresses. A contrario, la probabilité de classement en spam va baisser si l’expéditeur est connu et que les mots du mail sont plus « classiques ».

Avec la machine learning, on passe d’une informatique impérative basée sur des hypothèses (une programmation impérative de type règle « if… then… else… »), À une informatique probabiliste basée sur des informations réelles.

**Champs d’applications**

Aujourd’hui on retrouve les algorithmes machine learning dans tous les types d’applications métiers partant de l’e-business, l’analyse prédictive d’un **panier d’achat** d’un consommateur, la **détection de fraude** dans les transactions bancaires, ou l’**estimation du risque**de non-remboursementd’un prêt en fonction du passé financier d’un demandeur de crédit. D’autre exemple plus complexe on retrouve les technologies de **reconnaissance de forme** comme la reconnaissance optique de caractères ([OCR](http://fr.wikipedia.org/wiki/Reconnaissance_optique_de_caract%C3%A8res)) ou celle des visages, **le traitement naturel du langage** qui permet la reconnaissance du langage naturel , la traduction automatique, ou l’[**analyse des sentiments**](http://en.wikipedia.org/wiki/Sentiment_analysis) qui cherche à classer des documents selon leur tonalité émotionnelle dominante.

Les principaux types de données

**D’où viennent les données ?**



La réponse est facile, elles viennent de partout ! C’est d’ailleurs pour cela qu’on observe de nos jours un tel engouement pour la data science. On distingue deux types de données :

* Les données privées accessibles dans une entreprise, comme les bases de données métiers, les documents numérisés, les fichiers de log, les objets IOT,…
* Les données publiques accessibles à tous comme les opens Data, les API des réseaux sociaux et le web scraping  « extraction du contenu de sites Web, via un script ou un programme »

Les algorithmes

**Sous les données, des liens… plus ou moins certains !**

Quel que soit l’algorithme qu’utilise la machine learning, son but est de découvrir les liens dans les données (on parle souvent de pattern). Dans le cadre de l’emploi de méthodes de machine learning, on suppose donc qu’il existe un lien au sein des données et que l’algorithme nous aide à le trouver

L’objectif de l’algorithme est de trouver une corrélation entre les données et le résultat, généralement un problème machine learning est représenté sous la forme suivante.

Hypothèse *h*

Valeur d’entrée X

Valeur de sortie Y

Les algorithmes ne sont pas tous destinés aux mêmes usages. On les classe usuellement selon deux composantes, le mode d’apprentissage et le type de problème à traiter :

On distingue deux modes d’apprentissage :

Un apprentissage supervisé, qui consiste à extraire la connaissance à partir d’un ensemble de données contenant des couples entrée-sortie. Ces couples sont déjà « connus », dans le sens où les sorties sont définies a priori, la valeur de sortie peut être une indication fournie par un expert : par exemple, des valeurs de vérité de type OUI/NON ou MALADE/SAIN. Ces algorithmes cherchent à définir une représentation compacte des associations entrée-sortie, par l’intermédiaire d’une fonction de prédiction.

Un apprentissage non supervisés qui consiste à organiser les données en groupes. Chaque groupe doit comprendre des données similaires et les données différentes doivent se retrouver dans des groupes distincts. Dans ce cas, l’apprentissage ne se fait plus à partir d’une indication qui peut être préalablement fournie par un expert, mais uniquement à partir des fluctuations observables dans les données.

Mode d’apprentissage :

Le type de problème à traiter : (Régression et classification)

La distinction régression/classification se fait au sujet des algorithmes supervisés. On distingue deux types de valeurs de sorties. Dans le cadre d’un problème de régression, *Y* peut prendre une infinité de valeurs dans l’ensemble continu des réels (noté *Y* ∈ ℝ).

Ça peut être des températures, des tailles, des PIB, des taux de chômage, ou tout autre type de mesure n’ayant pas de valeurs finies a priori.

Dans le cadre d’un problème de classification, *Y* prend un nombre fini *k* de valeurs (*Y* = {1, …,

*k*}). On parle alors d’étiquettes attribuées aux valeurs d’entrée. C’est le cas des valeurs de vérité de type OUI/NON ou MALADE/SAIN évoqués précédemment.

Dans le tableau ci-dessous vous trouvez une classification des algorithmes les plus utilisés pour résoudre un problème machine learning par mode d’apprentissage et type de problème à traiter :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **algorithme** | **Mode d’apprentissage** | **Type de problème à traiter** |
| **Simple** | Regression lineaire univariee | Supervise | Regression |
| Regression lineaire multivariee | Supervise | Regression |
| Regression polynomiale | Supervise | Regression |
| Regression regularisee | Supervise | Regression |
| Naive Bayes | Supervise | Classification |
| Regression logistique | Supervise | Classification |
| Clustering hierarchique | Non supervise | - |
| Clustering non hierarchique | Non supervise | - |
| **Complexe** | Arbres de decision | Supervise | Regression ou classification |
| Random forest | Supervise | Regression ou classification |
| Gradient boosting | Supervise | Regression ou classification |
| Support Vector Machine | Supervise | Regression ou classification |
| Analyse en composantes principales | Non supervise | - |

Environnement de travail machine learning

Les outils de mise en place d’une solution Machine learning sont multiples, vous pouvez implémenter les algorithmes cités ci-dessous avec un langage de programmation de votre choix (JAVA, C++ ou autre langage) comme vous pouvez utiliser des librairies développées par un éditeur ou par la communauté.

On générale la communauté data scientist utilise un des deux langages de programmation :

* R qui est un langage de programmation et environnement statistique permet facilement la manipulation des fonctions mathématique et la représentation graphique des résultats, les exemples qui seront présentés dans la suite de l’article seront en R.
* Python pour sa simplicité et la disponibilité des bibliothèques qui implémentent les algorithmes les plus utilisés.

D’autres logiciels d’analyse des Big Data comme Sparck propose des bibliothèques d’implémentation des algorithmes machine learning MLib, des Framework sont implémentés au tour de l’écosystème Hadoop comme Weka, et Apache Mahout.

**Description sommaire de R**

Est un environnement intégré de manipulation de données, de calcul et de préparation de graphiques. Toutefois, ce n’est pas seulement un «autre» environnement statistique (comme SPSS ou SAS, par exemple), mais aussi un langage de programmation complet et autonome.

R est un langage principalement inspiré du S et de Scheme. Le S était à son tour inspiré de plusieurs langages, dont l’APL (autrefois un langage très prisé par les actuaires) et le Lisp.

Comme tous ces langages, le R est *interprété*, c’est-à-dire qu’il requiert un autre programme — l’*interprète* — pour que ses commandes soient exécutées.

R est un langage particulièrement puissant pour les applications mathématiques et statistiques (et donc actuarielles) puisque précisément développé dans ce but. Parmi ses caractéristiques particulièrement intéressantes, on note :

* Langage basé sur la notion de vecteur, ce qui simplifie les calculs mathématiques et réduit considérablement le recours aux structures itératives (boucles for, while, etc.)
* pas de typage ni de déclaration obligatoire des variables
* programmes courts, en général quelques lignes de code seulement
* temps de développement très court

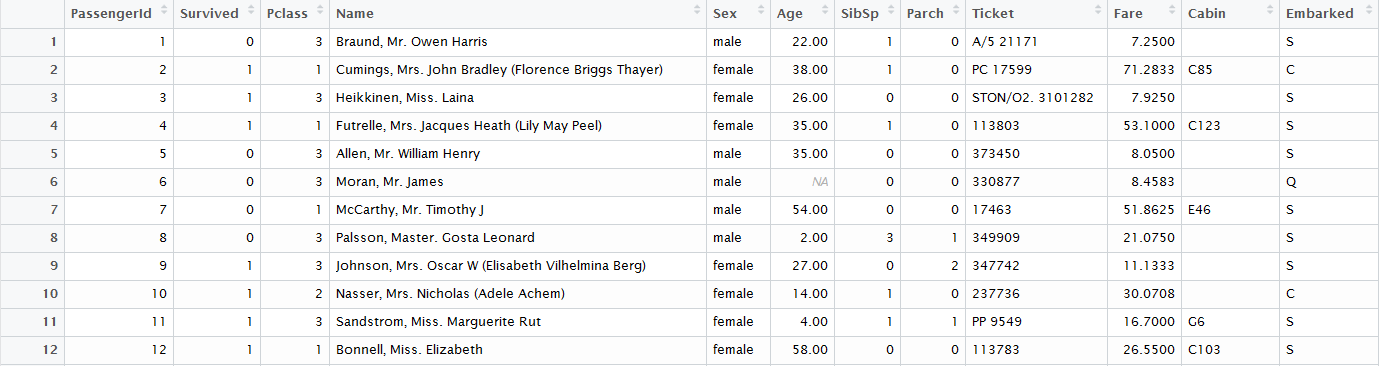


**Machine learning  cas d’étude : Prédire les survivants du Titanic**



Pour mieux comprendre comment les algorithmes machines learning arrivent à trouver la relation entre les données, nous allons prendre un exemple classique qu’on retrouve dans tous les livres et les formations machine learning : Comment prédire les survivants du Titanic.

Pour commencer nous devons récupérer les données d’apprentissage, le fichier train.csv (https://github.com/ylasmak/ML-R-Titanic/blob/master/train.csv) qui contient la base d’apprentissage supervisé

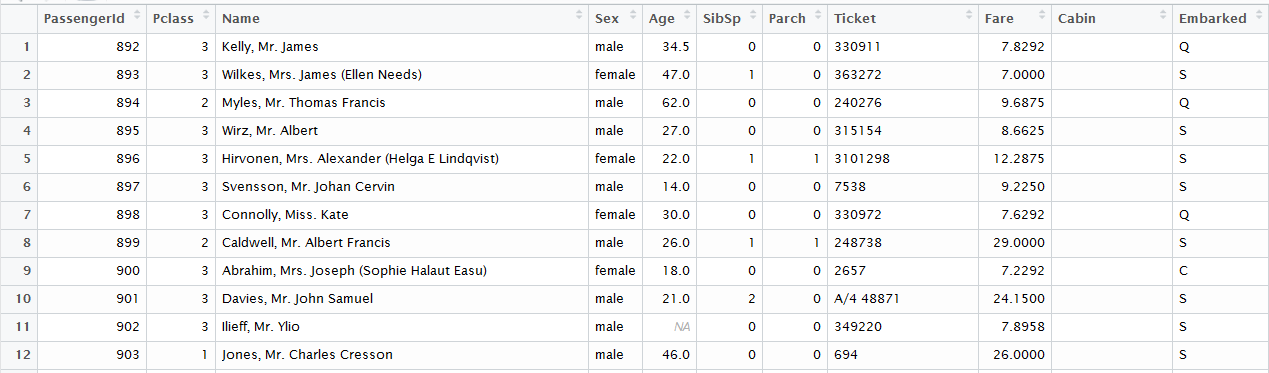


Chaque ligne de la dataSet train correspond à un passager (une observation ou un vecteur dans le jargon des data sentist), les colonnes correspondent à des variables.

De point de vue métier voici la signification de ces variables :

|  |  |
| --- | --- |
| **Variable** | **Signification métier** |
| Survived | Indique la mort ou survie du passager ,1 pour survie, 0 pour mort |
| Pclass | La classe des chambre du navire ,1 étant la meilleure, et 3 la classe économique |
| Name | Le nom du passager |
| Sex | Le sexe du passager Male et Femele |
| Age | Age du passager |
| SibSp | le nombre des membres de famille du passager de type frère, sœur, femme, marie |
| Parch | le nombre des membres de famille du passager de type parents, fils, filles, |
| Ticket | Le numéro du ticket |
| Fare | Le prix du ticket |
| Cabin | Le numéro de la cabine |
| Embarked | Le port d’embarquement |

Le fichier test.csv (https://github.com/ylasmak/ML-R-Titanic/blob/master /test.sv) contient un jeu de test qui va nous permettre par la suite de réaliser des prédictions, on retrouve les mêmes colonnes que celles du dataset d’apprentissage à part la colonne survived (colonne résultat)



**Préparation des données**

Avant de commencer la manipulation des algorithmes, il faut passer par une phase très importante qui consiste à analyser et traiter les données pour mieux préparer le dataset d’apprentissage

Etape 1 : Analyse des variables

Une première analyse des dataset nous permet de distinguer 3 types de variables :

Variable continue : variable de type numérique continue () : Age, SibSp ,Parch, Fare

Variable catégorielle : les valeurs de ces variables font partie d’un sous ensemble bien déterminé, (une énumération) : Pclass, Sex, embraked

Variable textuelle : la valeur de ce type de variable est un texte : Name, ticket

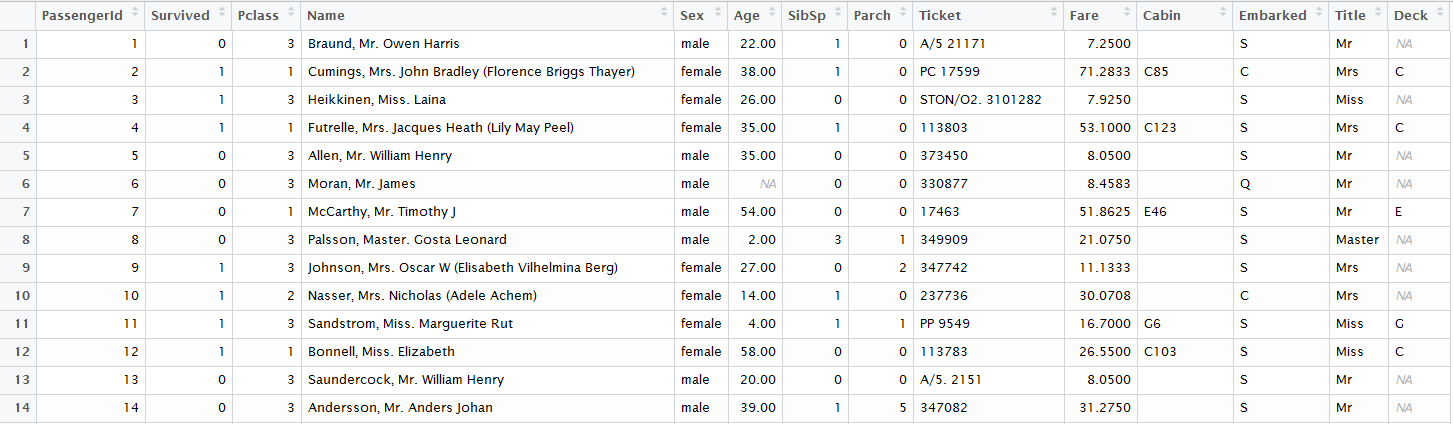
Les variables de type texte sont peu ou difficilement exploitables, mais il est possible d’en extraire des informations utiles

Etape 2 : Déduction des nouvelles variables

Comme vu dans la première étape la variable Name est de type chaine de caractère, nous remarquons que les noms ont toujours la même structure [nom1, titre, nom2, nom3 (nom3 étant optionnel)], nous pourrions alors à l’aide d’une expression régulière, extraire le titre est créé une nouvelle variable catégorielle Title

De même pour la variable cabine est sous forme [Pont][numéro de la cabine] nous pouvons déduire le Pont où se trouve la cabine du passager qui est une variable catégorielle.

Vous trouvez ci-dessous la nouvelle base d’apprentissage avec les nouvelles colonnes :



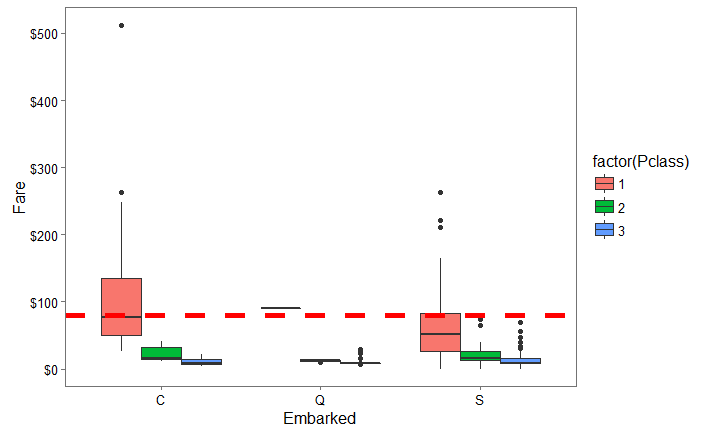
Etape 3 : correction des données manquantes

Une des problématiques qu’on peut rencontrer dans notre démarche machine learning est les valeurs manquantes qui empêchent le modèle d’apprentissage. Il est très fréquent de se retrouver avec des observations dont ils manquent des variables .

Il existe plusieurs solutions pour corriger la base d’apprentissage, la plus simple est d’ignorer toutes les observations dont ils manquent des variables, cette technique est efficace si le pourcentage des observations incomplètes est inférieure à 5%.

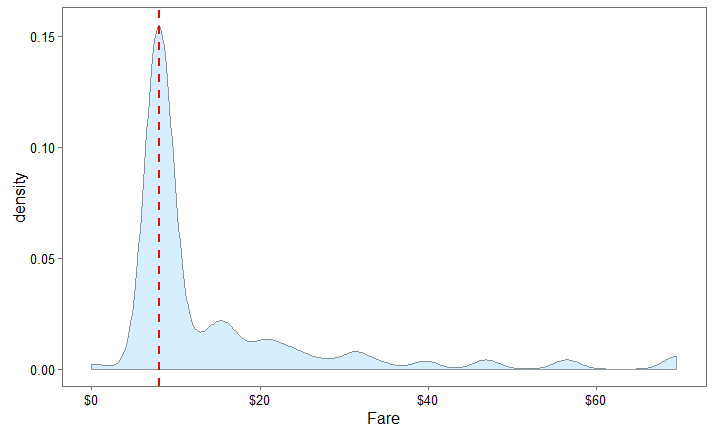
Une autre technique consiste à déduire l’information qui manque à partir des données existantes, prenant l’exemple des lignes 62 et 830 dont ils leurs manquent le port d’embarquement qui est une variable catégorielle, nous allons essayer de déduire leurs valeurs à partir des informations existantes (la classe et prix du ticket).

Les deux passagers 62 et 830 ont acheté leurs tickets à 80 $ en première classe, une analyse de la médiane des prix des tickets par classe nous permet de conclure que le port d’embarquement des deux passagers est le port C.



On remarque que le passager 1044 lui manque le prix du ticket (variable continue), que nous allons essayer de le déduire.

Le passager 1044 a embarqué du port S et il était en troisième classe, la technique consiste à calculer la médiane de la densité des prix des autres passagers qui ont embarqué du port S et qui étaient en troisième classe, nous pouvons déduire que le prix du ticket du passager 1044 est de 8,05$.



Il existe des bibliothèques en R comme *Nice* qui permettent de réaliser les imputations des données manquantes (vous pouvez consulter la documentation de de la bibliothèque Nice sur le lien suivant

**Phase apprentissage :**

Une fois notre base de donnée est prête, nous passons à la phase d’apprentissage, le but de cette phase est de trouver la fonction h qui permet de prédire la valeur de sortie dans le cas de notre exemple survived = 0, ou 1, en fonction des informations des passagers.

Il existe une dizaine d’algorithmes applicables à notre problème, nous allons voir dans la suite de ce paragraphe trois d’entre eux classés par ordre de complexité : les arbres de décision, le Random Forest et le Gradient Boosting

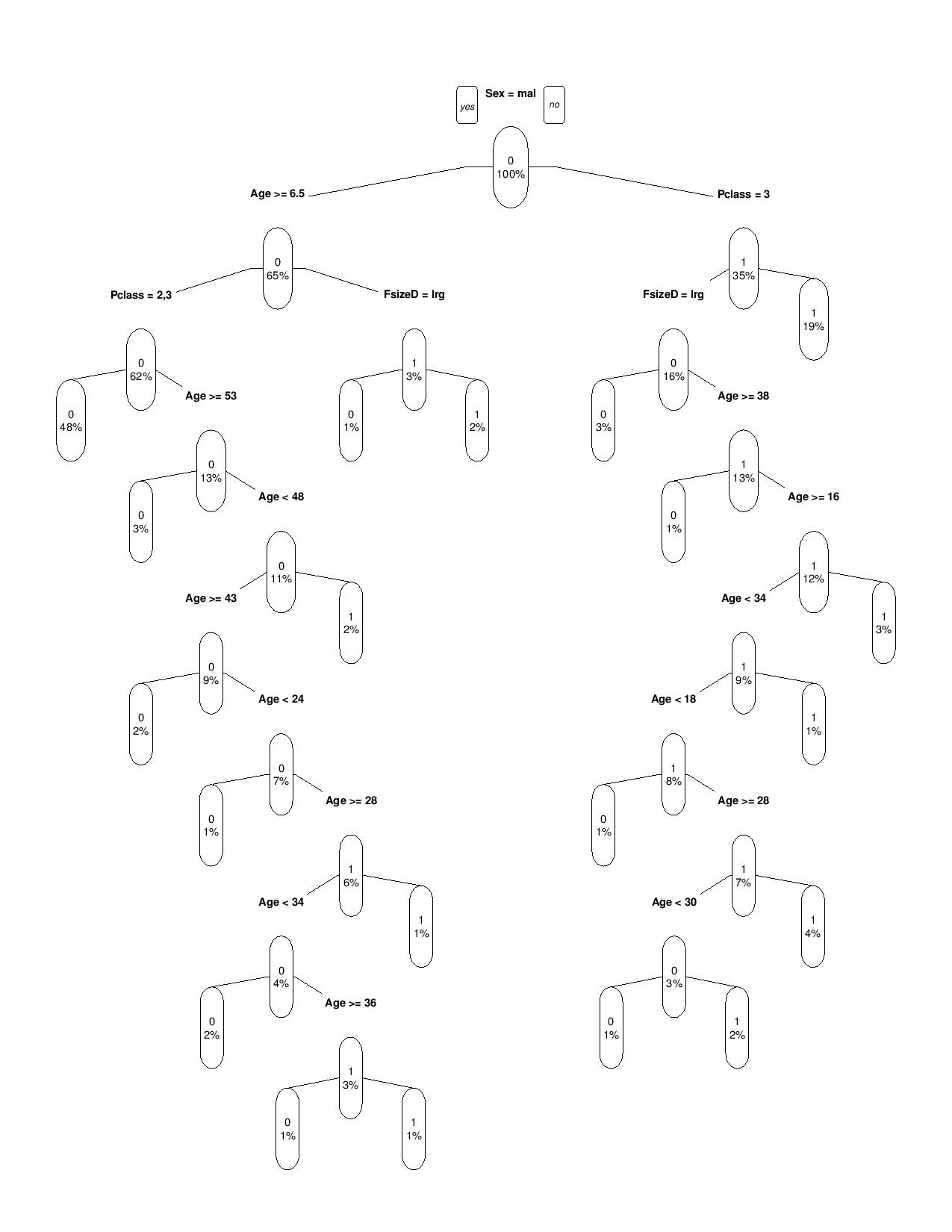
**Les arbres de décision :**

Les arbres de décision sont des arbres au sens informatique, leurs but est d’expliquer une valeur à partir d’une série de variables, on est donc dans le cas classique d’une matrice X avec m observations et n variables associées à une valeur de sortie Y à expliquer.

Ils sont très utilisés en statistique classique, sous leurs formes simple, ils sont très rarement utilisés en machine learning, par contre la compréhension de leurs principe et leurs méthodes de construction est un prérequis pour l’étude des algorithmes avancés comme le Random Forest ou le Gradient Boosting.

Les nœuds internes de l’arbre de décision sont appelés des nœuds de décision, Un tel nœud est étiqueté par un *test* qui peut être appliqué à toute description d'un individu de la population. En général, chaque test examine la valeur d'un unique attribut (sexe, Age, Port d’embarquement,…) des variables de notre problème. Les réponses possibles au test, correspondent aux labels des arcs issus de ce nœud, on passe par la suite au nœud suivant , le processus se répète jusqu’au nœud terminal (feuille), une fois arrivé à la feuille nous obtenons notre prédiction (vivant ou mort)

Ci-dessous le résultat de l’exécution de l’algorithme de l’arbre de décision sur notre base d’apprentissage des passagers du Titanic, la fonction utilisé est Rpart de la bibliothèque Part en R (le code est sur le lien suivant)



Prenons l’exemple du passager 892 et essayant de prédire son sort

Mr Kelly James est un homme, on navigue alors sur l’arc gauche du premier nœud, il est âgé de 34,5 ans donc supérieure à 6,5, on navigue alors sur l’arc gauche, il était en 3ieme classe, on navigue encore sur l’arc gauche, à la fin on se retrouve sur un nœud terminal (une feuille), on déduit que Mr Kelly James est mort au naufrage du Titanic, et voilà comment une prédiction est réalisée en utilisant les arbres de décision.

La question que vous vous posez maintenant est comment on choisit les critères et les valeurs de division ? Pourquoi on commence avec le sexe du passager et non pas d’une autre variable ? , et bien c’est l’intelligence de l’algorithme qui le fait pour vous, le principe est simple : il faut commencer par les variables les plus discriminantes aux variables les moins discriminantes, c’est-à-dire il faut créer le plus de désordre possible, ce principe est modélisé mathématiquement par la fonction d’entropie.

Dans la pratique, les arbres de décision sont de moins en moins utilisés du fait de leur très forte propension au sur-apprentissage (1), ils sont plus tôt utilisés en tant que classificateur faible à la base des méthodes ensemblistes

1. Le sur-apprentissage ou sur-ajustement (en anglais « overfitting ») est un problème pouvant survenir dans les méthodes mathématiques et informatiques d'**apprentissage** automatique supervisé. Il est en général provoqué par un mauvais dimensionnement de la structure utilisée pour classifier les données.

**Random Forest**

L’algorithme Random Forest est la star des algorithmes machine learning, il est rapide à entrainer et implémenté dans la plus part des outils machine learning.

Les méthodes ensemblistes dont fait partie le Random Forest partage une idée simple et intuitive : *si un médecin vous annonce que vous avez un problème qui nécessite une intervention chirurgicale que feriez-vous ? , il est fort probable que vous demandez un deuxième ou un troisième avis*.

Les méthodes ensemblistes fonctionnent avec le même principe, au lieu d’avoir un classificateur complexe censé tous faire, on en construit plusieurs de moindre qualité individuelle, comme chaque estimateur a ainsi une vision parcellaire du problème, il fait de son mieux pour le résoudre avec les données dont il dispose, ensuite ces multiples estimateurs sont réunis pour fournir une vision globale du problème.

Dans le cas du Random Forest, l’estimation finale d’un problème de classification est réalisée par vote .

Le Random Forest est une association des deux algorithmes tree bagging et feature sampling:

**Tree bagging**

Le Tree bagging apporte une amélioration significative de performance pour les arbres de décision

Prenons le cas d’un problème de classification binaire, nous disposons d’une matrice d’apprentissage X de m exemples d’apprentissage, chacun décrit par n variables et d’un vecteur binaire Y de dimension m.

La construction de B arbres de décision se fera comme suit :

* Tirer aléatoirement et avec remplacement B échantillons de (X,Y), ce qui nous permet de créer (Xbi,Ybi)
* Trainer un arbre de décision sur chaque couple (Xbi,Ybi)

Pour chaque donnée, on applique chacun des B arbres, il suffit ensuite de prendre la majorité parmi les B réponse.

**Feature sampling**

En plus du tirage aléatoire sur les lignes, le Feature sampling consiste à appliquer un tirage aléatoire sur les variables à utiliser, par défaut  variable pour un problème à n variables.

L’avantage de l’algorithme Random Forest est l’indépendance entre les arbres à entrainer, ce qui permet de paralléliser les traitements et améliorer les performances de l’apprentissage.

**Performance de l’algorithme**

La performance d’un algorithme (la capacité de prédire les bons résultats) est mesurée à partir de la base d’apprentissage, l’idée est de découper cette dernière en deux sous matrices S1 et S2, S1 contient 80% des observations, S2 contient les 20% restantes.

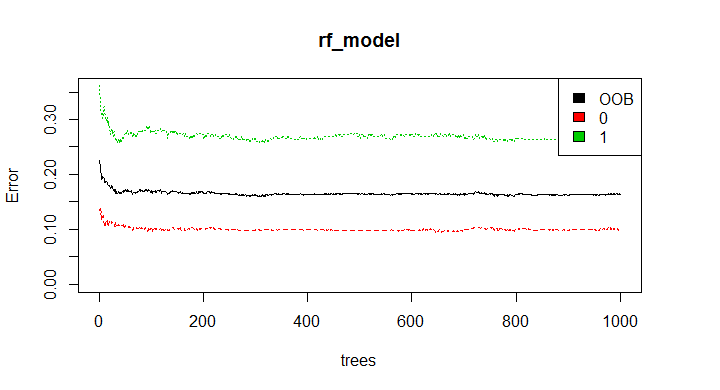
L’algorithme sera entrainé sur la sous matrice S1, puis on réalise des prédictions sur la matrice S2, enfin on compare les résultats de la prédiction aux valeurs réelles, la performance de l’algorithme est mesuré alors par le pourcentage des bonnes prédictions.

Le graphe ci-dessous est le résultat de l’exécution de l’algorithme Random Forest (Le code source en R est disponible sur XXXXXXXXXX) sur la base d’apprentissage du Titanic.

La courbe en noire représente la moyenne des fausses prédictions. On constate que l’erreur se stabilise à 18% après 400 entrainements (entrainement de 400 sous arbres tirés aléatoirement).

Il est important de spécifier un bon nombre d’arbres (appeler aussi nombre des itérations) pour l’algorithme d’apprentissage, une sous-estimation des nombres des itérations augmente le risque des erreurs (fausse prédiction), une surestimation des nombres des itérations impacte les performances de l’apprentissage (temps nécessaire pour exécuter l’algorithme), et augmente le phénomène du sur-apprentissage.

La courbe verte représente les faux positifs, et la courbe rouge représente les faux négatifs.

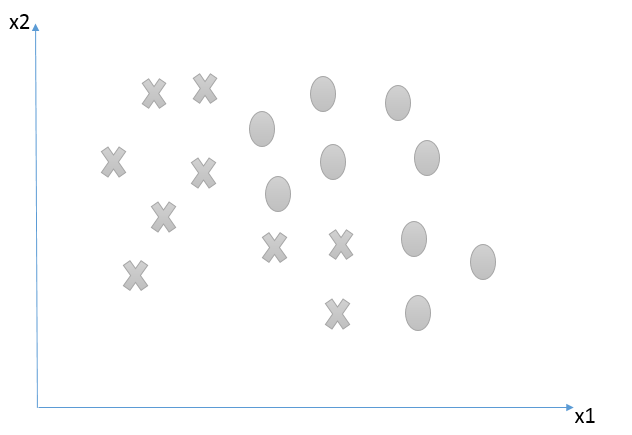


**Gradient Boosting**

Le Gradient Boosting repose sur un principe très simple, comme pour les humains, le Gradient Boosting apprend de ses erreurs.

L’idée principale est d’agréger plusieurs mini-classifieurs ensembles, mais en les créant itérativement. Ces mini-classifieurs  sont généralement des fonctions simples et paramétrées, le plus souvent des arbres de décision dont chaque paramètre est le critère de split des branches. Le super-classifieur final est une pondération de ces mini-classifieurs.

Prenons l’exemple d’un problème à deux dimensions (x1,x2), nous cherchons à classifier les observations (X ou O) représentées dans la figure ci-dessous



Dans un premier temps, nous trouvons la fonction h1 (mini-classifieur), dont l’erreur est représenté par le fond en rouge, cette erreur sera prise en compte par le deuxième mini-classifieur h2, que lui-même va générer des erreurs, qui seront à leurs tours pris en compte par le mini-classifieur h3.

Si on s’arrête à la troisième itération, le Gradient Boosting réalise la pondération des trois fonctions et crée un super classifieur H.

|  |  |
| --- | --- |
|  |  |
|  |  |

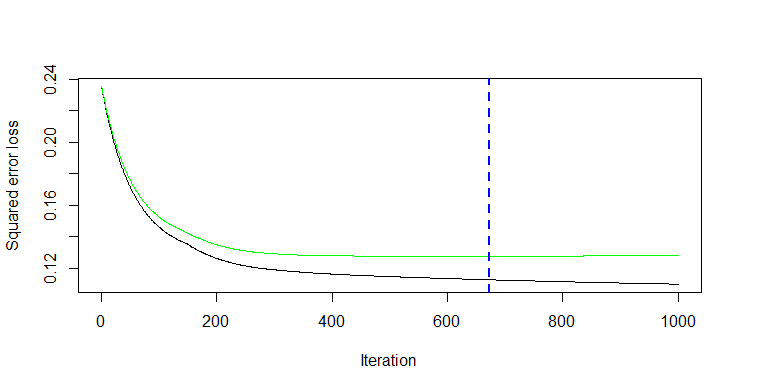
Nous venons de voir le principe du boosting , qui consiste à une tendance à réduire l’erreur, bien que dans notre exemple, nous avons abouti à un classifieur parfait, on réalité ce n’est pas le cas , on aura toujours un pourcentage de l’erreur qui il faut réduire tout au long des itérations de l’algorithme .

L’erreur dans les algorithmes ensembliste est représentée par une fonction de Cout, dont il faut trouver le minimum, c’est le rôle de la fonction de la descente du gradient , le Gradient Boosting est une association des algorithmes du boosting et la descente du gradient.

Revenant à notre exemple de la catastrophe du Titanic, nous entrainons la dataSet préparée dans le chapitre 2, en utilisant la bibliothèque **gmb** du langage **R** , on lance l’apprentissage avec l’algorithme Gradient Boosting (Le code source en R est sur le lient XXXXXXX) , si vous exécutez le code vous remarquez que l’exécution de la méthode du training *gbm* prend un temps considérable , contrairement à celle du Randon Forest qui est très rapide , cette lenteur est dûe au fait que le Gradient Boosting est un algorithme séquentiel qui ne peut être parallélisable.

La fonction gbm.perf de la bibliothèque permet de mesurer l’optimisation de la fonction du cout, ci-dessous l’évolution de la fonction du coût dans notre cas c’est une fonction gaussienne.

En analysant ce graphe on remarque que l’erreur ‘la courbe verte’ se stabilise à partir de l’itération 673.



**Machine learning dans le cloud**

Plusieurs plateforme propose des machines learning sur le cloud, l’avantage de cette solution est de disposer des performances du cloud, dans la pratique, exécuter un apprentissage sur une base de connaissance de plusieurs millions d’observation demande des ressources machine assez importantes, dans ce cas le cloud est très intéressant.

Autre point fort des machine learning dans le cloud, est l’encapsulation des complexités des algorithmes, vous ne devez pas être un expert dans le domaine pour exécuter un apprentissage (un simple tutoriel vous permet de lancer votre premier apprentissage), il suffit de charger la base d’apprentissage, votre fournisseur de service s’occupe du reste pour vous, il s’occupe de l’étape préparation des données, corrections des données et aussi le choix du bon algorithme, en réalité il exécute plusieurs algorithmes et propose le résultat du meilleur d’entre eux.

Vous trouvez dans la vidéo ci-dessous une très bonne présentation de Amazon machine learning

<https://youtu.be/PAHU8tPA7xs>

Il faut noter que pour Amazon comme pour Microsoft ou Google, vous n’aurez jamais accès au model, les trois plateformes proposent des interfaces pour la réalisation des prédictions et aussi des web service qui peuvent être appelés par vos applications métiers, bien les prédictions sont payantes comme l’entrainement de l’algorithme, à titre d’exemple pour un problème de classification d’un catalogue des produit selon les informations utilisateur et les descriptions des produits. Dans ce cas, vous avez utilisé 20 heures de temps de calcul pour générer vos modèles, le modèle qui en résulte avait une taille de 100 Mo et vous avez obtenu 890 000 prédictions en temps réel sur une période d'un mois.

Frais mensuels applicables aux prédictions

Le tarif mensuel applicable aux prédictions en temps réel et de 0,0001 USD par prédiction

Frais mensuels applicables aux prédictions = (0,0001 USD) \* 890 000 = 89,00 USD

Les prédictions en temps réel sont également soumises à des frais en ce qui concerne les capacités réservées, qui s'élèvent à 0,001 USD/10 Mo par heure

Frais applicables aux capacités réservées = (0,001/10 Mo par heure) \* (24 heures/jour) \* (31 jours/mois)\*(100 Mo/10) = 7,44 USD

Total des frais de prédiction = 89,00 USD + 7,44 USD = 96,44 USD

Total des frais mensuels

Frais de calcul + frais de prédiction = 8,40 USD + 96,44 USD = 104,84 USD

**Conclusion**

Nous avons vu tout au long de cet article le principe des machine learning ainsi que quelques algorithmes de classification pour l’apprentissage supervisé, il existe bien sûr d’autre algorithme plus complexe et plus performant comme SVM (Super Vector Machine) qui est un classificateur fortement non linéaire.

La compréhension des algorithmes utilisés dans un problème machine learning reste essentielle pour leurs bonnes utilisations, heureusement vous ne serez pas amenés à implémenter ces algorithmes vous-même, il existe des bibliothèques dans les langages les plus utilisé par la communauté (R , Python , Java et Matlab ) qui les implémentent, mais le paramétrage de ces bibliothèque nécessite une meilleure connaissance de ces algorithmes.