وزارة التعليم العالي والبحث العلمي

Université 20 Aout 1955 de Skikda

Faculté des Sciences

Département de Mathématiques

جامعة 20 أوت 1955 ، سكيكدة

كلـــــــية العلـــــوم

قسم الرياضيات

N^0 : U.S/F.S/D.M/……/2023.

Faculté des Sciences
Département de Mathématiques

Mémoire

Présenté en vue de l'obtention du diplôme de
Master en Mathématiques

Implémentation de problème de classification en utilisant Naïve Bayes

Option :ANEDP

Par : Bourendous khawla

Encadré par : Mallem Khadidja M.C.B U. SKIKDA

Devant le jury :

Président : Karek Chafia M.C.B U. SKIKDA
Examinateur : Boulkeroua Fouzia M.C.B U. SKIKDA

بسم الله الرحمن الرحيم

﴿وَأَنْ لَيْسَ لِلْإِنْسَانِ إِلَّا مَا سَعَى * وَأَنَّ سَعْيَهُ سَوْفَ يُرَى﴾

[سورة النجم:39 40]

Remerciements

Je tiens tout d'abord à remercier Allah qui m'a èclairé le bon chemin et qui m'a permis de réaliser ce modeste travail.

Je tiens à exprimer mon remerciement le plus chaleureux à mon encadreur

Mallem khadidja, qui a dirigé ce travail pour ses précieux conseils et son suivi au cours de ce travail.

Aussi je remercie tout ce qui m'a aidée de près ou de loin, enseignants et étudiants, à la réalisation de ce mémoire.

Et enfin, je tiens à exprimer ma parfaite Considération aux membres de jury pour avoir bien voulu examiner et juger mon travail.

A tout le corps enseignant du département de Mathématiques pour l'enseignement reçu.

Dédicace

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que : je désire ce travail à :

Mes chers parents, pour tous leurs sacrifices, leurs amours, leurs tendresses, leurs soutiens et leurs prières au long de mes études.

Mes chers frères pour leur appui et leur encouragement.

Tous mes amis et mes collègues. Amis bien-aimées qui m'ont soutenu pendant toutes ces cinq années, et grâce à eux, mes années universitaires ont été les meilleures que j'ai eues :Boutheina, Asma et Manel.

Table des matières

Résumé

Toute expert des données a besoin d'apprendre les mathématiques du Machine Learning qui aident à sélectionner le bon algorithme. La compréhension des mathématiques nous permet donc de mieux comprendre le fonctionnement du modèle, notamment le choix du bon paramètre du modèle et les stratégies de validation. Il y a six domaines mathématiques constituent la base du Machine Learning : l'algèbre linéaire, géométrie analytique, décomposition matriciel, calcul vectoriel, probabilité et distributions et optimisation. Nous nous intéressons ici à l'apprentissage supervisé en consacrant aux problèmes de classification naïve bayésienne, Dans un premier temps nous aborderons les principaux outils d'un réseau bayésien pour pouvoir prendre en charge les problèmes comportant la notion d'incertitude en générale en donnant des définitions principales de probabilité sur plusieurs variables et en illustrant avec un exemple simplifié d'un réseau bayésien. Enfin nous avons étudié une forme très simplifiée de ces réseaux est appelée réseaux bayésiens naïfs qui est basé sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes. Nous nous intéressons ensuite au modèle de Régression Logistique, qui permet de résoudre des problèmes de classification binaires. qui consistent à prédire ou classer la valeur d'une variable discrète. Dans ce cas le modèle linéaire ne convient pas, on développe alors une nouvelle fonction, c'est la fonction logistique (sigma) qui a la particularité d'être toujours comprise entre 0 et 1. A partir de cette fonction, il est possible de définir une frontière de décision. Typiquement, on définit un seuil à 0.5. Lorsqu'on teste notre modèle sur le Dataset, celui-ci nous donne des erreurs. L'ensemble de ces erreurs, c'est ce qu'on appelle la Fonction Coût. Pour la régression linéaire, la Fonction Coût donnait une courbe convexe (qui présente un unique minima). C'est ce qui fait que l'algorithme de Gradient Descent fonctionne. En revanche, utiliser cette fonction pour le modèle Logistique ne donnera pas de courbe convexe (dû à la non-linéarité) et l'algorithme de Gradient Descent se bloquera au premier minima rencontré, sans trouver le minimum global. Il faut donc développer une nouvelle Fonction Coût spécialement pour la régression logistique. On utilise alors la fonction logarithme pour transformer la fonction sigma en fonction convexe . L'algorithme de Gradient Descent s'applique exactement de la même manière que pour la régression linéaire.L'idée centrale du Machine Learning, c'est de laisser la machine trouver quels sont les paramètres de notre modèle qui minimisent la Fonction Coût. Enfin, nous avons étudié l'algorithme de Naïve Bayes qui permet de résoudre des problèmes de classification à plusieurs classes de façon simple et très efficace. et nous avons réalisé le projet classification des maladies cardiaques en utilisant cet algorithme.

MOTS CLES : **Apprentissage automatique,Régression logistique, Réseau bayésien, réseaux bayésiens naïfs, théorème de Bayes, Python.**

Abstract

As machine learning becomes more ubiquitous and its software packages become easier to use, it is natural and desirable that the low-level technical details are abstracted away and hidden from the practitioner. However, this brings with it the danger that a practitioner becomes unaware of the design decisions and, hence, the limits of machine learning algorithms.

The enthusiastic practitioner who is interested to learn more about the magic behind successful machine learning algorithms currently faces a daunting set of pre-requisite knowledge of Mathematics and statistics and how machine learning builds on it

In machine learning we are often interested in selecting the best hypothesis given data. In a classification problem, our hypothesis may be the class to assign for a new data instance. One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge. This master's thesis is about of implementing logistic regression using Naive bayes using Python which help to create strong decision criteria for users to make the training dataset based on which machine can predict the proper output.

The naive Bayes algorithm is a powerful and widely-used machine learning algorithm that is particularly useful for classification tasks. this master's thesis explains the basic math behind the Naive Bayes algorithm and how it works for binary classification problems. Its simplicity and efficiency make it a popular choice for many data science applications. we have covered most concepts of the algorithm and how to implement it in Python.

Key words : **Machine learning, Logistic regression, Bayesian network, Naive Bayesian networks, Bayes theorem, Python.**

الملخص:

لقد كان لتعلم الآلة نصيب كبير من الاهتمام في السنوا ت الأخيرة، فلقد أحدث ثورة كبيرة في مختلف المجالات ولذلك ظهرت العديد من الخوارزميات أي برامج يمكنها التعلم دون تدخلات بشرية.
على غرار مذكرة كل من سارة، خوله وإكرام في الأعوام السابقة حيث طبقوا نظرية التصنيف على خوارزمية Knn، فنحن طبقنا هذه النظرية باستعمال Naïve Baise والتي تستخدم لبناء نموذج بخصائص منفصلة عن بعض وبشكل عام فان المصنف البايزي يعمل مع البيانات الفئوية بشكل أفضل من البيانات الرقمية، وهذا باستخدام لغة البرمجة بايثون.

الكلمات المفتاحية :
تعلم الآلة، الانحدار اللوجستيكي، الشبكة البايزية، الشبكة البايزية الساذجة، نظرية بايز، لغة البرمجة بايثون.

INTRODUCTION

En 2019, Le Machine Learning ou bien l'apprentissage automatique est tout autour de nous. Il intervient chaque fois que nous cherchons un mot dans Google, une série sur Netflix, une vidéo sur YouTube, un produit sur Amazon. Grâce au Machine Learning, des millions de cancers peuvent être diagnostiqués chaque année, des milliards de spams et de virus informatiques sont bloqués pour protéger nos ordinateurs.

Les fondations du Machine Learning

Comprendre pourquoi le Machine Learning est utilisé [9],[13] , [10], [1], [2], [15]
Nous, les êtres humains, sommes quotidiennement confronté à des problèmes que nous cherchons à résoudre. Par exemple : Comment construire un pont plus solide ? Comment augmenter nos bénéfices ? Comment éliminer le cancer ? Ou tout simplement quelle route emprunter pour aller au travail ? Pour nous aider dans nos recherches, nous avons inventé l'ordinateur, qui permet de résoudre en quelques minutes des calculs qui nous prendraient des millions d'années à effectuer. Mais il faut savoir qu'un ordinateur ne sait en réalité faire qu'une chose : résoudre les calculs qu'on lui donne.
À partir de là, 2 situations possibles :
1.On connait le calcul à effectuer pour résoudre notre problème.
Dans ce cas, facile ! On entre ce calcul dans l'ordinateur, c'est ce qu'on appelle la programmation, et l'ordinateur nous donne le résultat.
Exemple : Déterminer la structure d'un pon.
2.On ne connait pas le calcul qui résout notre problème
Dans ce cas... on est bloqué. Impossible de donner à un ordinateur un calcul que nous ne connaissons pas. C'est comme vouloir poster une lettre que nous n'aurions pas écrite.

Exemples : Reconnaître un visage sur une photo, prédire le cours de la Bourse, éliminer le cancer, composer de la musique, conduire une voiture ….

Le Machine Learning a justement été inventé pour venir débloquer la situation 2 (quand on ne connait pas le calcul) en utilisant une technique audacieuse,

Laisser la Machine apprendre à partir d'expériences

Le Machine Learning consiste à laisser l'ordinateur apprendre quel calcul effectuer, plutôt que de lui donner ce calcul (c'est-à-dire le programmer de façon explicite). On attribue généralement ses débuts à la création du test de Turing, en 1950. C'est le mathématicien britannique Alan Turing qui imagine cette épreuve, censée déterminer si une machine peut simuler la pensée humaine [17]. Dans ce contexte, de premiers programmes « intelligents » voient le jour. En 1959, c'est le mathématicien américain Arthur Samuel [14] qui utilise pour la première fois le terme « machine learning » qui a développé un programme pouvant apprendre tout seul comment jouer aux Dames.

Les deux méthodes d'apprentissage

Pour donner à un ordinateur la capacité d'apprendre, on utilise des **méthodes d'apprentissage** qui sont fortement inspirées de la façon dont nous, les êtres humains, apprenons è faire des choses. [4]. Parmi ces méthodes, on compte : l'apprentissage supervisé (Supervised Learning) et **l'apprentissage non supervisé** (Unsupervised Learning).

L'apprentissage supervisé

On parle de l'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples qu'elle doit étudier par exemple : lorsqu'on à apprendre le chinois, il faudra soit acheter un livre de traduction chinois-arabe, ou bien trouver un professeur de chinois. Le rôle du professeur ou du livre de traduction sera de superviser votre apprentissage en vous fournissant des exemples de traductions arabe-chinois que vous devrez mémoriser.

Les quatre notions de l'apprentissage supervisé

Il y' a quatre notions importantes dans l'apprentissage supervisé : le Dataset, le Modèle et ses paramètres , la Fonction Coût et l'Algorithme d'apprentissage.

Notion 1 : Apprendre à partir d'exemples (Dataset)

Pour apprendre la langue chinoise, on parle d'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples (x, y) dans le but de lui faire apprendre la relation qui relie x à y.

— La variable y porte le nom de **target** (la cible). C'est la valeur que l'on cherche à prédire.

— La variable x porte le nom de **feature** (facteur). Un facteur influence la valeur de y, et

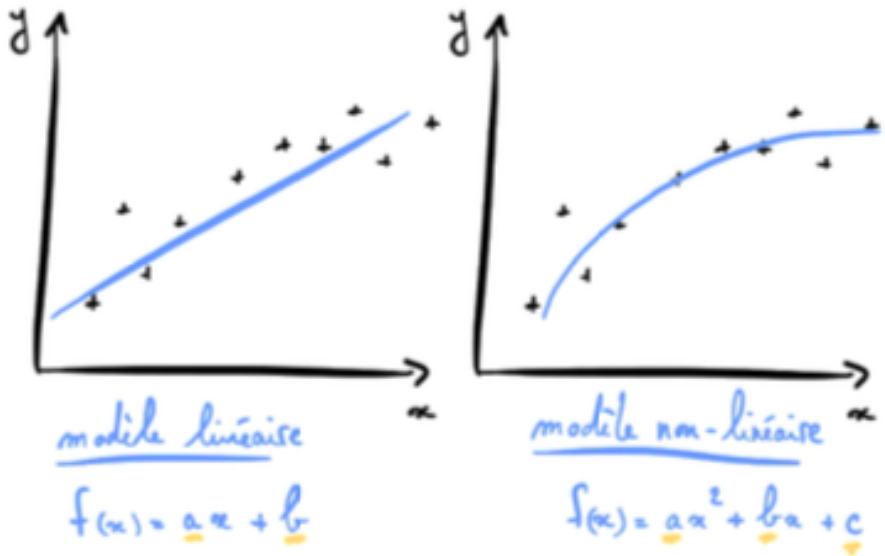
on a en général beaucoup de **features** (x_1,x_2,x_3 ...) dans notre Dataset que l'on regroupe dans une matrice X.

Ci-dessous, un Dataset qui regroupe des exemples d'appartements avec leur prix y ainsi que certaines de leurs caractéristiques (features).

Target	features		
y	x_1	x_2	x_3
Prix	Surface m2	N chambres	Qualité
€313,000.00	124	3	1.5
€2,384,000.00	339	5	2.5
€342,000.00	179	3	2
€420,000.00	186	3	2.25
€550,000.00	180	4	2.5
€490,000.00	82	2	1
€335,000.00	125	2	2

Notion 2 : Développer un modèle à partir du Dataset

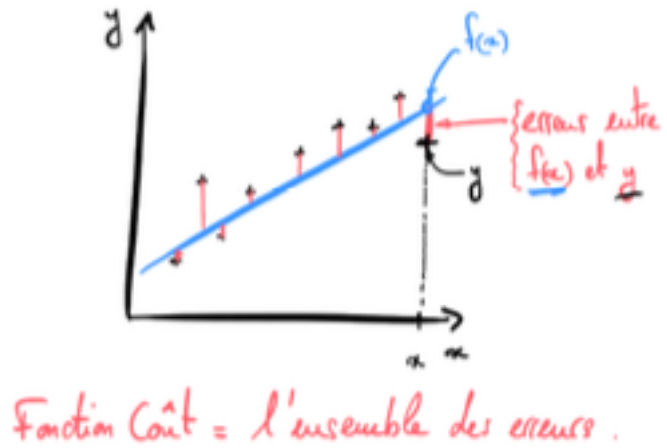
En Machine Learning, on développe un modèle à partir de ce Dataset. Il peut s'agir d'un modèle linéaire comme vous pouvez le voir à gauche, ou bien un modèle non-linéaire comme vous pouvez le voir à droite. Nous verrons dans ce livre comment choisir un modèle plutôt qu'un autre.



On définit a, b , c etc. comme étant les paramètres d'un modèle.

Notion 3 : Les erreurs de notre modèle - la Fonction-Coût

Autre chose à noter est qu'un modèle nous retourne des erreurs par rapport à notre Dataset. On appelle **Fonction Coût** l'ensemble de ces erreurs (le plus souvent on prend la moyenne quadratique des erreurs comme dans le chapitre 2).



Allons droit au but : Avoir un bon modèle, c'est avoir un modèle qui nous donne de petites erreurs, donc une petite **Fonction Coût**.

Notion 4 : Minimiser la Fonction Coût

L'objectif central en Supervised Learning, c'est de trouver les paramètres du modèle qui minimisent la Fonction Coût. Pour cela, on utilise un algorithme d'apprentissage, l'exemple le plus courant étant **l'algorithme de Gradient Descent**, (dans le deuxième et le troisième chapitre).

Les applications du Supervised Learning

Avec le Supervised Learning on peut développer des modèles pour résoudre deux types de problèmes :

— Les problèmes de **Régression**

— Les problèmes de **Classification**[16]

Dans les problèmes de régression, on cherche à prédire la valeur d'une variable **continue**, c'est-à-dire une variable qui peut prendre une **infinité** de valeurs. Par exemple :

— Prédire le prix d'un appartement y selon sa surface habitable x

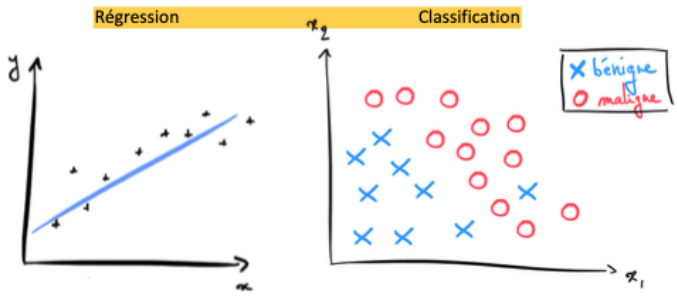
— Prédire la quantité d'essence consommée y selon la distance parcourue x

Dans un problème de classification, on cherche à classer un objet dans différentes classes, c'est-à-dire que l'on cherche à prédire la valeur d'une variable discrète (qui ne prend qu'un nombre fini de valeurs). Par exemple :

— Prédire si un email est un spam (classe $y = 1$) ou non (classe $y = 0$) selon le nombre de liens présent dans l'email

— Prédire si une tumeur est maligne ($y = 1$) ou bénigne ($y = 0$) selon la taille de la tumeur (x_1) et l'âge du patient (x_2)

Dans le cas d'un problème de classification, on représente souvent les classes par des symboles, plutôt que par leur valeur numérique $(0, 1, \dots)$



L'apprentissage non-supervisé

Supposons qu'on est , seul, en Chine, sans bouquin, sans traducteur, il existe tout de même une méthode pour apprendre le chinois. C'est l'apprentissage non-supervisé, dans ce mémoire on va pas introduire la notion de l'apprentissage non-supervisé .

Les quatre étapes essentielles pour l'apprentissage automatique ou bien en anglais " Machine Learning"

Le Dataset

En Machine Learning, tout démarre d'un Dataset qui contient nos données. Dans l'apprentissage supervisé, le Dataset contient les questions (x) et les réponses (y) au problème que la machine doit résoudre .

Le modèle et ses paramètres

A partir de ce Dataset, on crée un modèle, qui n'est autre qu'une fonction mathématique. Les coefficients de cette fonction sont les paramètres du modèle.

La Fonction Coût

Lorsqu'on teste notre modèle sur le Dataset, celui-ci nous donne des erreurs. L'ensemble de ces erreurs, c'est ce qu'on appelle la Fonction Coût.

L'Algorithme d'apprentissage

L'idée centrale du Machine Learning, c'est de laisser la machine trouver quels sont les paramètres de notre modèle qui minimisent la Fonction Coût.

Plan du travaille

Tout d'abord nous avons commencé par une introduction générale sur l'apprentissage automatique en expliquant ces fondations et ces quatre étapes essentielles.
Dans le premier chapitre nous aborderons les principaux outils d'un réseau bayésien pour pouvoir prendre en charge les problèmes comportant la notion d'incertitude en générale en donnant des définitions principales de probabilité sur plusieurs variables et en illustrant avec une exemple simplifié d'un réseau bayésien. Enfin nous avons étudié une forme très simplifiée de ces réseaux est appelée réseaux bayésiens naïfs qui est basé sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes. deuxième projet nous avons appliqué un modèle du corrélation avec python.
Dans le deuxième chapitre (resp. troisième chapitre) nous avons présenté le modèle du régression linéaire (resp. régression logistique).
Enfin, nous avons étudié l'algorithme de Naïve Bayes qui permet de résoudre des problèmes de classification à plusieurs classes de façon simple et très efficace. et nous avons réalisé le projet classification des maladies cardiaques en utilisant cet algorithme.

CHAPITRE 1

LE RÉSEAU BAYÉSIEN

Qu'est-ce qu'un réseau bayésien ?
Un réseau bayésien est un modèle graphique probabiliste simple. Très utilisé en machine learning, il s'agit d'un graphe (de type acyclique) composé de nœuds, chacun représentant une variable aléatoire. Les nœuds sont reliés par des liens de causalité.
A chaque nœud enfant, ou variable expliquée, est associé des fonctions de probabilité conditionnelle correspondant à chaque nœud parent (variable explicative). Une table de probabilités conditionnelles se traduit par un vecteur de probabilité différent pour chaque combinaison possible des valeurs des variables parents. **Un réseau bayésien combine ainsi le système des graphes avec la théorie des probabilités.**
Le domaine des réseaux bayésiens a comme particularité d'allier deux champs différents des mathématiques dans le but de représenter l'incertitude : la théorie des graphes, d'une part, qui fournit le cadre nécessaire pour une modélisation qualitative des connaissances ; et la théorie des probabilités, d'autre part, qui permet d'introduire une information quantitative dans ces connaissances.

1.1 Probabilités

La théorie des probabilités propose un cadre mathématique pour re- présenter quantitativement l'incertain. La présentation qui est faite ici est forcément tronquée puisque orientée vers son utilisation dans le domaine des réseaux bayésiens. En particulier, l'espace sur lequel seront définies les probabilités restera discret et fini. Ce n'est bien sûr pas le cas général mais c'est suffisant pour ce qui suit.

1.1.1 Définitions principales

Définition 1.1.1 (PROBABILITÉ)**.** Soit Ω un ensemble fini non vide, $(\Sigma, \cap, \cup)$ une algèbre sur Ω (Σ l'ensemble des parties de Ω). Soit $p : \Sigma \to [0,1]$ une fonction à valeurs réelles.
p est une probabilité sur (Ω, Σ) si et seulement si elle vérifie :

1. $\forall A \in \Sigma, 0 \leq p(A) \leq 1$;
2. $\forall A, B \in \Sigma, [A \cap B = \emptyset] \Longrightarrow p(A \cup B) = p(A) + p(B)$. A et B sont alors dits mutuellement exclusifs ;
3. $p(\Omega) = 1$ (et donc $p(\emptyset) = 0$.)

Tout élément (non nul) minimal au sens de l'inclusion de Σ est appelé un événement élémentaire sur Ω qu'on nomme souvent l'univers. Il est à noter qu'un événement sur Ω est une sous-partie de Ω. Un événement (modification de l'univers) est donc en fait représenté par l'ensemble des états de l'univers auxquels il peut mener. Ω est appelé l'événement certain. De même, on appellera $\emptyset$ l'événement impossible.

Définition 1.1.2 (VARIABLE ALÉATOIRE (V.A.))**.** Une variable aléatoire est une fonction X définie sur Ω :

$$X : \left| \begin{array}{ccc} \Omega & \longrightarrow & \mathcal{D}_X \\ \omega & \longmapsto & X(\omega) \end{array} \right.$$

Pour $x \in \mathcal{D}_X$, on note alors $X = x$ l'événement $\omega \in \Omega \,|X(\omega) = x$. $\mathcal{D}_X$ est le domaine de définition de X. Une variable aléatoire permet de caractériser des événements (qui sont des sous-ensembles d'événements élémentaires) par une simple valeur. Si le domaine de définition de la variable X est fini, alors X est une variable aléatoire discrète. Comme cette étude se restreint à un Ω fini, les variables aléatoires seront donc toujours considérées comme discrètes. De plus, on parle de variable aléatoire binaire lorsque le domaine de définition de la variable ne possède que deux éléments (« 0/1 », « oui/non », etc.).

Exemple 1.1.3. Pour étudier la distribution de probabilité de la somme du tirage de deux dés, il suffit de définir une variable aléatoire représentant cette somme, ce qui permet de manipuler beaucoup plus facilement les événements correspondants (1.1).

Pour la suite, on suivra la notation suivante : une variable aléatoire sera représentée par une majuscule ($A, B, \dots$). La valeur que prend cette variable aléatoire sera notée par la même lettre mais minuscule ($a \in \mathcal{D}_A, b \in \mathcal{D}_B, c \in \mathcal{D}_C, \dots$). Enfin, quand aucune ambiguïté ne sera possible, on simplifiera au maximum la notation un peu lourde de l'événement représenté par

$\mathcal{D}_X$	...1	2	3	4	5	6	7	8	9	10	11	12	13...
$\{X = x\}$	$\emptyset$	(1,1)	(1,2) (2,1)	(1,3) (2,2) (3,1)	(1,4) (2,3) (3,2) (4,1)	(1,5) (2,4) (3,3) (4,2) (5,1)	(1,6) (2,5) (3,4) (4,3) (5,2) (6,1)	(2,6) (3,5) (4,4) (5,3) (6,2)	(3,6) (4,5) (5,4) (6,3)	(4,6) (5,5) (6,4)	(5,6) (6,5)	(6,6)	$\emptyset$
$P(\{X = x\})$	0	$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{12}$	$\frac{1}{9}$	$\frac{5}{36}$	$\frac{1}{6}$	$\frac{5}{36}$	$\frac{1}{9}$	$\frac{1}{12}$	$\frac{2}{18}$	$\frac{1}{36}$	0

TAB. B.1 *Distribution des événements élémentaires en fonction d'une v.a.*

FIG. 1.1 – Tableau

$A = a$; de telle façon que : $p(A = a) = p(A = a) = p(a)$
. Pour terminer, il est certainement intéressant de noter la différence entre :

1. $p(A)$ qui est la probabilité associée à l'événement $A \subset \Omega$;
2. p(A = a) = p(A = a) = p(a) qui est la probabilité associée à l'événement $A = a$;
3. $p(A)$ qui est une fonction qui associe à tout élément $a \in D_A$ la valeur de probabilité de l'événement $p(A = a)$.

1.1.2 Probabilités sur plusieurs variables

Une variable aléatoire est donc un moyen pour condenser une information pertinente sur un univers. Cependant, il faut souvent plus d'une variable aléatoire pour caractériser précisément l'état de l'univers. L'étape suivante est bien sûr d'avoir le moyen de croiser ces différentes sources d'information.

Probabilités jointes

Soient A et B deux variables aléatoires sur le même univers Ω. On parle alors de probabilité pour la fonction définie sur $\mathcal{D}_A \times \mathcal{D}_B$ par :

$$
\begin{aligned}
p_{AB} :| \{D_A \times \mathcal{D}_B &\longrightarrow [0,1] \\
(a,b) &\longmapsto p_{AB}(a,b) = p(\{A=a\} \cap \{B=b\}) \\
&= p(\{\omega \in \Omega \mid A(\omega) = a \wedge B(\omega) = b\})
\end{aligned}
$$

Toutes ces probabilités jointes sont construites à partir de la même fonction de probabilité sur Ω : p. La liste des arguments d'une probabilité jointe est donc suffisante pour la caractériser. C'est pourquoi il est commun de les noter simplement p lorsqu'aucune ambiguïté n'est possible :

$$p_{ABCD}(a,b,c,d) = p(a,b,c,d)$$

Probabilités marginales

Réciproquement, la donnée d'une probabilité jointe d'ensemble de variables permet de retrouver la probabilité jointe de chacun de ses sous- ensembles. C'est ce qu'on appelle une probabilité marginale.

PROPRIÉTÉ (MARGINALISATION)

Soit U un ensemble fini, non vide de variables aléatoires, $V \subset U$ non vide et $V' = U \setminus V$ et $p(U)$ la probabilité jointe sur les variables de U ; on appelle alors marginalisation de p sur V la fonction :

$$\forall v \in \mathcal{D}_V, \quad p(v) = \sum_{v' \in \mathcal{D}_{V'}} p(v,v') \tag{1.1.1}$$

Cette fonction correspond à la probabilité jointe des variables de V, voir [8].

Probabilités conditionnelles

Définition 1.1.4 (LOI FONDAMENTALE)**.** Soient deux variables aléatoires A et B sur le même univers. Pour tout $a \in \mathcal{D}_A$ et $b \in \mathcal{D}_B$, la probabilité conditionnelle de $A = a$ étant donné $B = b$ est le nombre $p(a|b)$ vérifiant :

$$p(a,b) = p(a|b)p(b) \tag{1.1.2}$$

Définition 1.1.5 (LOI FONDAMENTALE GÉNÉRALISÉE)**.** Soit un ensemble de variables aléa-

toires $(A_i)_{i\in 1,\dots,n}$ sur le même univers,

$$p(a_1,\dots,a_n)=\prod_{i=1}^{n} p(a_i|a_1,\dots,a_{i-1})$$

On utilisera parfois la convention $p(X|\emptyset)=p(X)$

Théorème 1.1.6. *Si $p(b)$ est positive alors*

$$p(a|b)=\frac{p(b|a)p(a)}{p(b)}$$

Plus généralement,

$$p(a|b,c)=\frac{p(b|a,c)p(a|c)}{p(b|c)}$$

Indépendance conditionnelle

L'indépendance conditionnelle est un concept dont l'importance a été particulièrement soulignée par [5].

Définition 1.1.7 (INDÉPENDANCE CONDITIONNELLE)**.** Soient un univers Ω et un ensemble V de v.a. sur Ω. Soit $X,Y,Z\subset V$.
X est indépendant de Y conditionnellement à Z noté $(X \perp\!\!\!\perp Y|Z)$ si et seulement si ces ensembles vérifient :

$$X \perp\!\!\!\perp Y|Z=\begin{cases} p(X|Y,Z)=p(X|Z) \\ \text{et} \quad p(Y|X,Z)=p(Y|Z)\end{cases}$$

Définition 1.1.8 (INDÉPENDANCE MARGINALE)**.**

$$X \perp\!\!\!\perp Y=\begin{cases} \forall x\in\mathcal{D}_X, \quad p(Y|X=x)=p(Y) \\ \text{et} \quad \forall y\in\mathcal{D}_Y, \quad p(X|Y=y)=p(X)\end{cases}$$

Cette indépendance conditionnelle implique des relations entre les différentes probabilités :

$$\begin{aligned} X \perp\!\!\!\perp Y|Z &\iff p(X|Y,Z) = p(X|Z) \\ &\iff p(X,Y|Z) = p(X|Z)p(Y|Z) \\ &\iff p(X,Y,Z) = p(X|Z)p(Y|Z)p(Z) \end{aligned}$$

Plus généralement, supposons une loi jointe $p(X_1,\ldots,X_n)$. Cette loi jointe peut s'écrire par définition des probabilités conditionnelles (et sous réserve de positivité) :

$$p(X_1,\ldots,X_n) = \prod_{i=1}^{n} p(X_i|X_1,...,X_{i-1})$$

S'il est possible de simplifier chaque probabilité $p(X_i|X_1,\ldots,X_{i-1})$ grâce à des indépendances conditionnelles, la complexité du calcul de la loi jointe peut être grandement améliorée.

Théorème 1.1.9. *$\forall i, V_i \subset X_1,...,X_{i-1}$ tel que $X_i \perp\!\!\!\perp (X_1,\ldots,X_{i-1} \setminus V_i)|V_i$*

$$p(X_1,\ldots,X_n) = \prod_{i=1}^{n} p(X_i|V_i)$$

Propriétés 1.1.10. La relation ternaire d'indépendance conditionnelle vérifie les propriétés suivantes :

Si $X \perp\!\!\!\perp Y|Z$	*alors $Y \perp\!\!\!\perp X|Z$*
Si $X \perp\!\!\!\perp Y|Z$ et $X \perp\!\!\!\perp W|Y,Z$	*alors $X \perp\!\!\!\perp Y,W|Z$*
Si $X \perp\!\!\!\perp Y|Z,W$ et $X \perp\!\!\!\perp Z|Y,W$	*alors $X \perp\!\!\!\perp Y,Z|W$*

En particulier, la dernière proposition est invalide s'il existe une liaison déterministe entre Y et Z. Elle est vérifiée, par exemple, dans le cas où la loi $\mathrm{p}(X,Y,Z,W)$ est une loi strictement positive.

1.2 Réseau bayésien

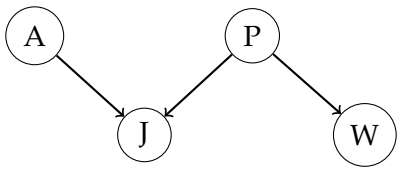
1.2.1 Une représentation graphique de la causalité

La représentation graphique la plus intuitive de l'influence d'un événement, d'un fait, ou d'une variable sur une autre, est probablement de représenter la causalité en reliant la cause à

l'effet par une flèche orientée.

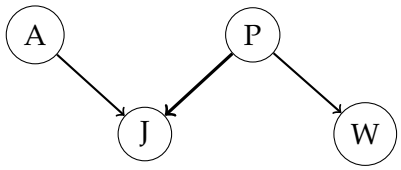
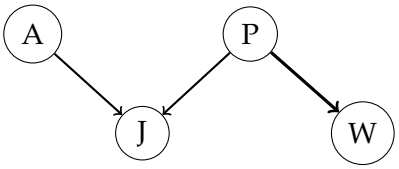
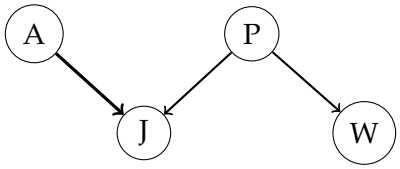
1.2.2 Circulation de l'information dans un graphe causal

Exemple 1.2.1. Pour cela, nous allons utiliser un exemple, extrêmement classique dans la littérature sur les réseaux bayésiens, initialement extrait de Pearl [11], et repris dans [8].
Ce matin-là, alors que le temps est clair et sec, M. Holmes sort de sa maison. **Il s'aperçoit que la pelouse de son jardin est humide**. Il se demande alors **s'il a plu pendant la nuit**, ou s'il a simplement **oublié de débrancher son arroseur automatique**. Il jette alors un coup d'œil à la pelouse de son voisin, M. Watson, et s'aperçoit qu'elle est également humide. Il en déduit alors qu'il a probablement plu, et il décide de partir au travail sans vérifier son arroseur automatique.
La représentation graphique du modèle causal utilisé par M. Holmes est la suivante :



A	J'ai oublié de débrancher mon arroseur automatique.
P	Il a plu pendant cette nuit.
J	L'herbe de mon jardin est humide.
W	L'herbe du jardin de M. Watson est humide.

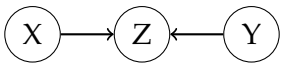
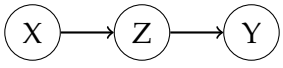
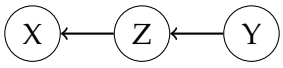
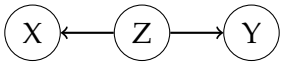
La lecture du graphe est bien conforme à l'intuition :

	S'il a plu pendant la nuit, l'herbe de mon jardin est humide.
	S'il a plu pendant la nuit, l'herbe du jardin de M. Watson est également humide.
	Si j'ai oublié de débrancher mon arroseur automatique, l'herbe de mon jardin est humide.

Dans cet exemple simple, on voit que l'information a circulé uniquement dans le sens effet $\rightarrow$ cause.

Le cas général

la circulation de l'information dans un graphe causal .

	Connexion convergente : X et Y causent Z.
 	Connexion en série : X cause Z, Z cause Y (ou le cas symétrique).
	Connexion divergente : Z cause X et Y.

1.2.3 D-séparation (blocage)

nous savons maintenant exactement dans quelles conditions une information peut circuler à l'intérieur d'un graphe. On voit qu'il ne s'agit pas de suivre le sens des flèches !

Supposons que nous disposions d'un graphe relativement complexe, pour lequel nous disposons déjà d'un certain nombre d'informations (i.e certaines variables sont déjà connues). Si nous apprenons maintenant une autre information, devons-nous réviser notre opinion sur

l'ensemble des autres nœuds de ce graphe ?
Pour répondre à cette question, nous pouvons essayer de synthétiser l'étude de ces circuits d'informations en une règle appelée d-séparation, qui décrit dans quelles conditions l'information entre un nœud X et un nœud Y est bloquée.
On dira que X et Y sont d-séparés par Z si pour tous les chemins entre X et Y, l'une au moins des deux conditions suivantes est vérifiée :

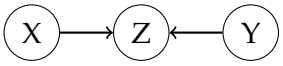
— Le chemin converge en un nœud W, tel que W = Z, et W n'est pas une cause directe de Z.

— Le chemin passe par Z, et est soit divergent, soit en série au nœud Z.

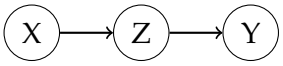
Essayons de comprendre intuitivement cette définition. Supposons que Z soit la seule information connue dans le graphe. Supposons maintenant que j'apprenne la valeur de X. Si X et Y sont d-séparés par Z, que se passe-t-il ?
Considérons un chemin entre X et Y. Soit ce chemin converge en un point W ($\rightarrow$ W $\leftarrow$), tel que W = Z, et W n'est pas une cause directe de Z. Donc, par hypothèse (Z est la seule information connue dans le graphe), aucune information n'est disponible sur W. D'après notre étude ci-dessus, ce chemin est donc bloqué.
Sinon, ce chemin passe par Z, et on a soit $\rightarrow$ Z $\rightarrow$, soit $\leftarrow$ Z $\rightarrow$. Toujours d'après notre étude, comme Z est connu, l'information ne peut circuler à travers Z. Tous ces chemins sont donc bloqués.
Donc si X et Y sont d-séparés par Z, et si Z est la seule information connue dans le graphe, une nouvelle information sur X ne modifie en rien mon opinion sur Y.

Exemple 1.2.2. (« X est d-séparé de Y par Z » est noté $\langle X \mid Z \mid Y \rangle$)

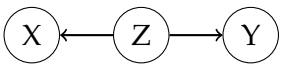
- X = tremblement de terre
  Y = cambriolage
  Z = alarme
  Le fait qu'il y ait eu un tremblement de terre dans le voisinage (X) n'a aucun lien a priori avec le fait que ma maison ait été cambriolée (Y). En revanche, si mon alarme s'est déclenchée (Z), j'ai tendance à croire que je viens d'être cambriolé (Y). Si maintenant j'apprends qu'il vient d'y avoir un tremblement de terre (X) dans le voisinage, je suis rassuré sur l'éventualité d'un cambriolage (Y).



L'information ne peut circuler de X à Y que si Z est connu.

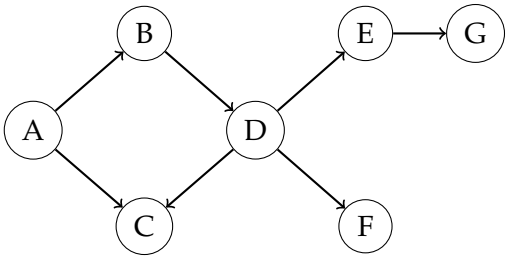
- X = ensoleillement
  Y = prix du blé
  Z = récolte
  Si la saison a été ensoleillée (X), la récolte sera abondante (Z). Si la récolte est abondante, le prix du blé est bas (Y). Si je sais déjà que la récolte a été abondante (Z), le fait de connaître l'ensoleillement (X) ne m'apprend plus rien sur le prix du blé (Y).
  

  L'information ne peut circuler de X à Y que si Z n'est pas connu.

- X = la pelouse de mon jardin est humide
  Y = la pelouse de mon voisin est humide
  Z = il a plu cette nuit
  Si la pelouse de mon jardin est humide (X), j'ai tendance à croire qu'il a plu cette nuit (Z), et donc que la pelouse de mon voisin sera aussi humide (Y). Si en revanche je sais qu'il a plu cette nuit (Z), je peux affirmer que la pelouse du jardin de mon voisin sera humide (Y), et l'information que je peux avoir sur l'état de ma propre pelouse (X) n'y change rien.
  

  L'information ne peut circuler de X à Y que si Z n'est pas connu.

- $\langle A \mid B \mid D \rangle$ Le chemin A-B-D est en série en B ($A \rightarrow B \rightarrow D$).
  Le chemin A-C-D est convergent en C ($A \rightarrow C \leftarrow D$).
  $\langle A \mid D \mid E \rangle$ Tous les chemins de A à E passent par D.
  Le chemin A-B-D-E est en série en D ($B \rightarrow D \rightarrow E$).
  Le chemin A-C-D-E est divergent en D ($C \leftarrow D \rightarrow E$).



1.3 Exemple d'un réseaux bayésiens

Définition 1.3.1. Un réseau bayésien est défini par :

- un graphe orienté sans circuit (DAG) $G = (V, E)$, où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G ;

- un espace probabilisé fini (Ω, Z, p) ;

- un ensemble de variables aléatoires associées aux nœuds du graphe et définies sur (Ω, Z, p), tel que :

$$p(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} p(V_i | C(V_i)) \tag{1.3.1}$$

où $C(Vi)$ est l'ensemble des causes (parents) de V_i dans le graphe G.

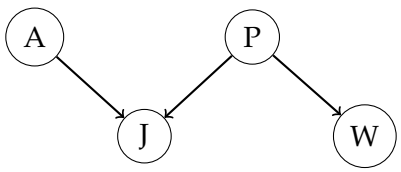
1.3.1 Propriétés

Un réseau bayésien est donc un graphe causal auquel on peut associé une représentation probabiliste sous-jacente. Comme on l'a vu, cette représentation permet de rendre quantitatifs les raisonnements sur les causalités que l'on peut faire à l'intérieur du graphe. On va également évoqué très rapidement le lien entre d-séparation et indépendance. En réalité un résultat très important existe, qui affirme que « si X et Y sont d-séparés par Z, alors X et Y sont indépendants sachant Z ». Ce résultat, démontré par Verma et Pearl en 1988 [18], constitue la propriété fondamentale des réseaux bayésiens.

$$<X|Z|Y> \Longrightarrow p(X \mid Y, Z) = p(X \mid Z) \tag{1.3.2}$$

1.3.2 Exemple [graphe causal avec tableau de probabilité]

Reprenons l'exemple précédent de M. Holmes :



A	J'ai oublié de débrancher mon arroseur automatique.
P	Il a plu pendant cette nuit.
J	L'herbe de mon jardin est humide.
W	L'herbe du jardin de M. Watson est humide.

Chaque valeur peut avoir deux valeurs (vrai ou faux) (1 ou 0) par exemple : J L'herbe de mon jardin est humide et $\overline{J}$ L'herbe de mon jardin n 'est pas humide.
On peut évaluer les probabilités de certains événements, soit marginales, soit conditionnellement à un autre événement.

Probabilités a priori

Evénement	Probabilité	Commentaire
$A = V$	0.4	M. Holmes oublie assez souvent de débrancher son arroseur automatique.
$A = F$	0.6	
$P = V$	0.4	La région est relativement pluvieuse
$P = F$	0.6	

Probabilités conditionnelles(à posteriori)

La table ci-après exprime la connaissance selon laquelle l'herbe de mon jardin est humide si, et seulement si, il a plu, ou si j'ai oublié de débrancher mon arroseur automatique. $p(J/A, P)$

	A=V		A=F	
	P=V	P=F	P=V	P=F
J=V	1	1	1	0
J=F	0	0	0	1

Enfin, la table ci-après exprime la connaissance selon laquelle l'herbe du jardin de mon voisin M. Watson est humide si, et seulement si, il a plu. $p(W/P)$

	P=V	P=F
W=V	1	0
W=F	0	1

Indépendances

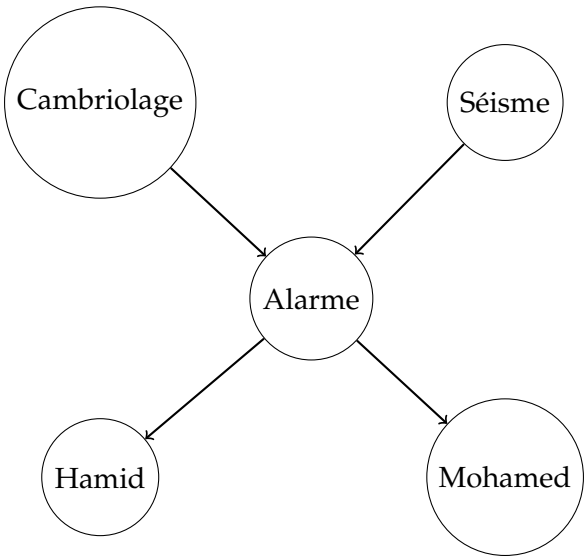
Les variables A et P sont indépendantes

Exemple 1.3.2. • Je suis au travail et mon voisin Mohamed et Hamid m'ont promis de m'appeler chaque fois que **mon alarme** sonne.

- **Mon voisin Hamid m'appelle** pour me dire que mon alarme sonne, parfois il confond l'alarme avec la sonnerie du téléphone.
- Par contre **mon voisin Mohamed, ne m'appelle pas toujours** parfois il met la musique trop fort.
- Parfois mon alarme se met à sonner lorsque il' y a de léger **séisme**
- Comment conclure qu'il y a eu un **cambriolage** chez moi ou non .

Les relations de causalité- les dépendances conditionnelles

- Un cambriolage peut déclencher l'alarme.
- Un séisme peut déclencher l'alarme
- L'alarme peut inciter Mohamed à appeler
- l'alarme peut Hamid à appeler.



Les tables de probabilité On prend les notations suivantes :

C	Cambriolage
S	Séisme
A	Alarme
H	Hamid
M	Mohamed

$p(C) = 0.001$

$p(S) = 0.002$

C	S	$p(A/C,S)$
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	$p(H/A)$
T	0.9
F	0.05

M	$p(M/A)$
T	0.7
F	0.01

Remarque 1.3.3.

- Les variables C et S n'ont pas des parents dans ce cas on parle de probabilité a priori.
- Pour les autres variables on parle de la probabilité à postériori (conditionnelles).
- Il y a autres appellation de réseau Bayésiens : les réseaux de croyances et modèle graphique dirigé acyclique.

Calcul de probabilité jointes

On applique (1.3.1)

$$p(C,S,A,H,M) = p(C)p(S)p(A/C,S)p(H/A)p(M/A)$$

Une démonstration est possible en appliquant le théorème de bayes (1.1.2) et la propriété (1.3.2) :

$$\begin{aligned}
p(H,M,A,C,S) &= p(H/M,A,C,S)p(M,A,C,S) \\
&= p(H/M,A,C,S)p(M/A,C,S)p(A,C,S) \\
&= p(H/M,A,C,S)p(M/A,C,S)p(A/C,S)p(C,S) \\
&= p(H/M,A,C,S)p(M/A,C,S)p(A/C,S)p(C/S)p(S) \\
&= p(H/A)p(M/A)p(A/C,S)p(C)p(S) \\
&= p(C)p(S)p(A/C,S)p(H/A)p(M/A) \\
&= p(C,S,A,H,M)
\end{aligned}$$

Calcul de $p(H,M,A,\overline{C},\overline{S})$.

D'après les tables de probabilités et (1.1.2) on obtient :

$$\begin{aligned}
p(H,M,A,\overline{C},\overline{S}) &= p(H/A)p(M/A)p(A/C,S)p(\overline{C})p(\overline{S}) \\
&= 0.9 * 0.7 * 0.001 * 0.999 * 0.998 \\
&= 0.000628
\end{aligned}$$

Calcul de probabilité marjinale $p(\overline{C}, A)$

On applique la propriété (1.1.1)

$$\begin{aligned} p(\overline{C}, A) &= \sum_{M,A,S,\overline{C}} p(H, M, A, \overline{C}, A) \\ &= \sum_{s} p(A|\overline{C}, S)p(\overline{C})p(S) \sum_{H} p(H|A) \sum_{M} p(M|A) \\ &= \sum_{s} p(A|\overline{C}, S)p(\overline{C})p(S) \\ &= p(A|\overline{C}, S)p(\overline{C})p(S) + p(A|\overline{C}, \overline{S})p(\overline{C})p(\overline{S}) \\ &= 0.29 \times 0.999 \times 0.002 + 0.001 \times 0.999 \times 0.998 \\ &= 0.000579 + 0.00099 \end{aligned}$$

Remarque 1.3.4. Dans les probabilités marjinales, on peut ignorer les noeuds dont les descendants ne sont pas observés.

Dans l'exemple précédent $p(\overline{C}, A)$:

- H et M et leur descendants ne sont pas observé, on peut les ignorer.
- S est un ancêtre de A, donc on doit le marjinaliser explicitement .

1.4 Modèle bayésien naïf (Naive Bayes classifier) [retour au machine learning]

Soit un Dataset (X, y) feature et target tel que : $X = (x_1, x_2, x_3, \ldots, x_n)$ on appelle au théorème de Bayes :

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

pour l'appliquer au Dataset

$$\begin{aligned} p(y|X) &= \frac{p(X|y)p(y)}{p(X)} \\ &= \frac{p(x_1, x_2, \ldots, x_n|y)p(y)}{p(x_1, x_2, \ldots, x_n)} \end{aligned}$$

En pratique, seul le numérateur nous intéresse, puisque le dénominateur ne dépend pas de y et les valeurs des features x_i sont données. Le dénominateur est donc en réalité constant. Le numérateur est soumis à la loi de probabilité à plusieurs variables et factorisé de la façon

suivante, en utilisant plusieurs fois la définition de la probabilité conditionnelle :

$$\begin{aligned} p(y)p(x_1, x_2, \ldots, x_n|y) &= p(y)p(x_1|y)p(x_2, \ldots, x_n|y, x_1) \\ &= p(y)p(x_1|y)p(x_2|y, x_1)p(x_3, \ldots, x_n|y, x_1, x_2) \\ &= p(y)p(x_1|y)p(x_2|y, x_1)p(x_n, \ldots, x_n|y, x_1, x_2, \ldots, x_{n-1}) \end{aligned}$$

C'est là que nous faisons intervenir l'hypothèse naïve : si chaque x_i est indépendant à des autres features $x_{j\neq i}$, conditionnellement à y alors

$$p(x_j|y, x_j) = p(x_i|y)$$

pour tout $j \neq i$, par conséquent la probabilité conditionnelle peut s'écrire

$$\begin{aligned} p(x_1, x_2, \ldots, x_n|y) &= p(x_1|y)p(x_2|y)p(x_3|y)\ldots p(x_n|y) \\ &= \prod_{i=1}^{n} p(x_i|y) \end{aligned}$$

Par conséquent, en tenant compte de l'hypothèse d'indépendance ci-dessus, on obtient :

$$p(y|x_1, x_2, \ldots, x_n) = \frac{1}{z}\prod_{i=1}^{n} p(x_i|y)p(y)$$

où z (appelé « évidence ») est un facteur d'échelle qui dépend uniquement de $x_1, \ldots, x_n$, à savoir une constante dans la mesure où les valeurs des variables caractéristiques sont connues.

1.4.1 Construire un classifieur à partir du modèle de probabilités

Jusqu'à présent nous avons établi le modèle à caractéristiques indépendantes, à savoir le modèle de probabilités bayésien naïf. Le classifieur bayésien naïf couple ce modèle avec une règle de décision. Une règle couramment employée consiste à choisir l'hypothèse la plus probable. Il s'agit de la règle du maximum a posteriori ou MAP. Le classifieur correspondant à cette règle est la fonction classifieur suivante :

$$\text{classifieur}(x_1, \ldots, x_n) = argmax_y p(y)\prod_{i=1}^{n} p(x_i|y)$$

Exemple 1.4.1. On prend le famous Dataset for playing Tennis or not :

Outlook				
	Yes	No	p(yes)	p(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature				
	Yes	No	p(yes)	p(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cold	3	1	3/9	1/5
Total	9	5	100%	100%

Play		
		p(yes) and p(no)
yes	9	9/14
no	5	5/14
Total	14	100%

Supposons qu'aujourd'hui est Sunny et Hot. On pose today(Sunny, Hot) on cherche la probabilité de $p(yes|today)$ en appliquant le théorème de Bayes on obtient :

$$p(yes|today) = \frac{p(Sunny|yes)p(Hot|yes)p(yes)}{p(today)}$$

$$p(Sunny|yes)p(Hot|yes)p(yes) = 2/9 * 2/9 * 9/14 = 0.031$$

$$p(Sunny|no)p(Hot|no)p(no) = 3/5 * 2/5 * 5/14 = 0.0875$$

$$p(today) = 0.031 + 0.0875 = 0.1185$$

$$p(yes|today) = \frac{0.031}{0.031 + 0.0875} = 0.27$$

$$p(no|today) = 1 - 0.27 = 0.73$$

$$\text{classifieur}(x_1, \ldots, x_n) = argmax_y p(y) \prod_{i=1}^{n} p(x_i|y) = 0.73$$

1.4.2 Naive Bayes Gaussian GNB

Dans ce mémoire on prend la loi normale (lois gaussienne) une loi de probabilité qui dépend de deux paramètres : son espérance, un nombre réel noté μ, et son écart type, un nombre réel positif noté σ. La densité de probabilité de la loi normale d'espérance μ, et d'écart type σ est donnée par :

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

CHAPITRE 2

LA RÉGRESSION LINÉAIRE

La régression linéaire est une modélisation linéaire qui permet d'établir des estimations dans le futur à partir d'informations provenant du passé, par exemple on cherche à prédire le cours de la bourse, le prix d'un appartement, ou bien l'évolution de la température sur Terre, donc on cherche en fait à résoudre un problème de régression.
On dispose d'un Dataset (x,y) donc on peut utiliser l'apprentissage supervisé pour développer un modèle de régression. Dans ce chapitre on va montrer comment développer notre premier modèle de Machine Learning ! [6], [3]

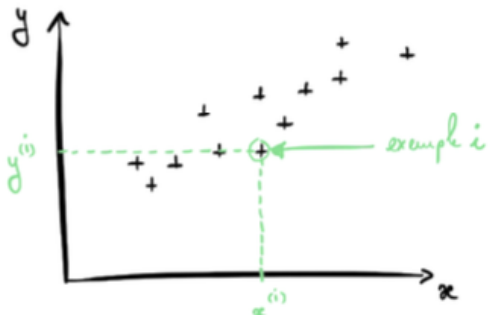
2.1 Récolter les données

Imaginons que plusieurs agences immobilières nous aient fourni des données sur des appartements à vendre, notamment le prix de l'appartement (y) et la surface habitable (x). En Machine Learning, on dit qu'on dispose de m exemples d'appartements.
On désigne :

— $x^{(i)}$ la surface habitable de l'exemple i

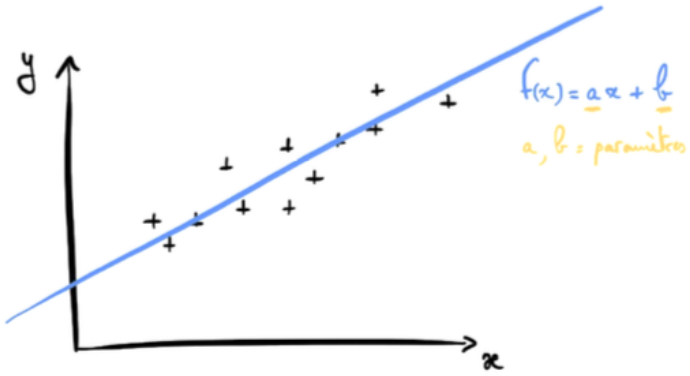
— $y^{(i)}$ le prix de l'exemple i

En visualisant notre Dataset, on obtient le nuage de points suivant :



2.2 Créer un modèle linéaire

A partir de ces données, on développe un modèle linéaire $f(x) = ax + b$ où a et b sont les paramètres du modèle.
Un bon modèle donne de petites erreurs entre ses prédictions $f(x)$ et les exemples (y) du Dataset. Nous ne connaissons pas les valeurs des paramètres a et b, ce sera le rôle de la machine de les trouver, de sorte à tracer un modèle qui s'insère bien dans notre nuage de point comme ci-dessous :



2.3 Définir La Fonction Coût

Pour la régression linéaire, on utilise le **squared error** pour mesurer les erreurs entre $f(x)$ et (y) . Concrètement, voici la formule pour exprimer l'erreur i entre le prix $y^{(i)}$ et la prédiction faites en utilisant la surface $x^{(i)}$

$$erreur^{(i)} = (f(x^{(i)}) - y^{(i)})^2$$

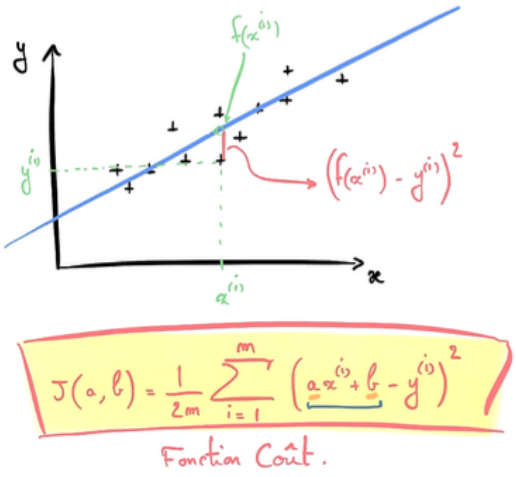
Par exemple, imaginons que le 10 ième exemple de notre Dataset soit un appartement de $x^{10} = 80\ m^2$ dont le prix s'élève à $y^{(10)} = 100,000\ €$ et que notre modèle prédise un prix de $f(x^{(10)}) = 100,002\ €$. L'erreur pour cette exemple est donc :

$$erreur^{(10)} = (f(x^{(10)}) - y^{(10)})^2 = (100,002 - 100,000)^2 = (2)^2 = 4$$

Chaque prédiction s'accompagne d'une erreur, on a donc m erreurs.
On définit la **Fonction Coût** $J(a,b)$ comme étant la moyenne de toutes les erreurs :

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m} erreur^{(i)} = \frac{1}{2m}\sum_{i=1}^{m}(f(x^{(i)}) - y^{(i)})^2$$

Note : En français, cette fonction a un nom : c'est l'erreur **quadratique moyenne** (Mean Squared Error)



2.4 Trouver les paramètres qui minimisent la Fonction Coût "Gradient Descent"

La prochaine étape est l'étape la plus excitante, il s'agit de laisser la machine apprendre quels sont les paramètres qui **minimisent** la Fonction Coût, c'est-à-dire les paramètres qui nous donnent le meilleur modèle. Pour trouver le minimum, on utilise un algorithme d'optimisation qui s'appelle **Gradient Descent** (la descente de gradient).

Comprendre le Gradient Descent (la descente de gradient)

Imaginons nous, perdu en montagne. notre but est de rejoindre le refuge qui se trouve au point

le plus bas de la vallée. nous n'avons pas pris de carte avec nous donc nous ne connaissons pas les coordonnées de ce refuge, nous devons le trouver tout seul.
Pour nous en sortir, voici une stratégie à adopter :

1. Depuis notre position actuelle, nous partons en direction de là où la pente descend la plus forte.

2. nous avançons une certaine distance en suivant cette direction coûte que coûte (même si ça implique de remonter une pente)

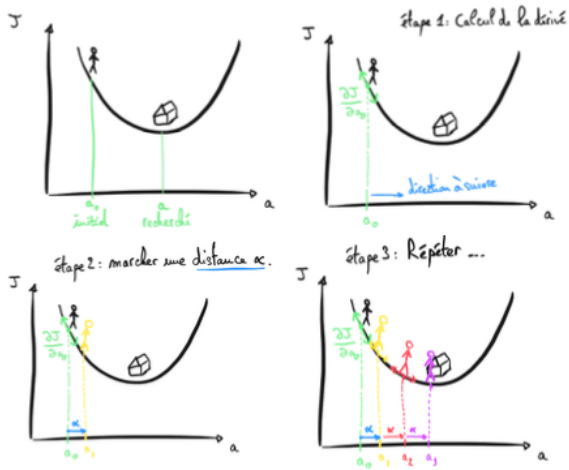
3. Une fois cette distance parcourue, nous répétons les 2 premières opérations en boucle, jusqu'à atteindre le point le plus bas de la vallée.

Les étapes 1, 2 et 3 forment ce qu'on appelle l'algorithme de **Gradient Descent**.
Cet algorithme vous permet de trouver le **minimum** de la Fonction Coût $J(a, b)$ (le point le plus bas de la montagne) en partant de coordonnées a et b **aléatoires** (votre position initiale dans la montagne) :

1. Calculer la **pente** de la Fonction Coût, c'est-à-dire la **dérivée** de $J(a, b)$.

2. **Evoluer** d'une certaine **distance** $\propto$ dans la direction de la pente la plus forte. Cela a pour résultat de modifier les paramètres a et b
   3. Recommencer les étapes 1 et 2 jusqu'à atteindre le minimum de $J(a, b)$.

Pour illustrer l'algorithme, voir le dessin ci-dessous, où on montre la recherche du paramètre idéal (la même chose s'applique au paramètre b)



Comment utiliser l'algorithme de Gradient Descent

Pour rappel, nous avons jusqu'à présent créé un Dataset, développé un modèle aux paramètres inconnus, et exprimé la **Fonction Coût** $J(a, b)$ associée à ce modèle.

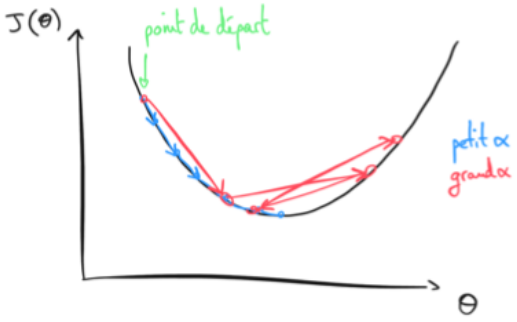
Notre objectif final : Trouver les paramètres a et b qui minimisent $J(a, b)$. Pour cela, nous allons choisir a et b au **hasard** (nous allons nous perdre en montagne) puis allons utiliser en **boucle** la descente de gradient pour mettre à jour nos paramètres dans la direction de la Fonction Coût **la plus faible**.

Répeter en boucle :

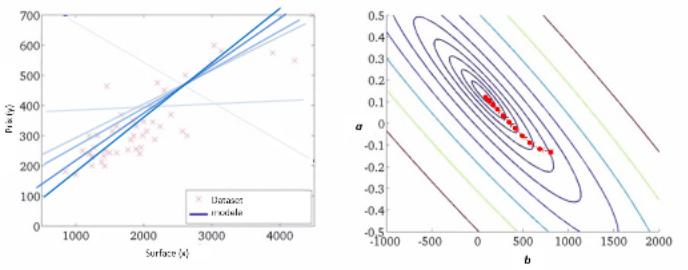
$$a = a - \propto \frac{\partial J(a,b)}{\partial a}$$
$$b = b - \propto \frac{\partial J(a,b)}{\partial b}$$

À chaque itération de cette boucle, les paramàtres a et b sont mis à jour en **soustrayant** leur propre valeur à la valeur de la **pente** $\frac{\partial J(a,b)}{\partial ...}$ multipliée par la distance à parcourir $\propto$. On appelle $\propto$ la vitesse d'apprentissage (**Learning rate)**.

Si la vitesse est trop lente, le modèle peut mettre longtemps à être entraîné, mais si la vitesse est trop grande, alors la distance parcourue est trop longue et le modèle peut ne jamais converger. Il est important de trouver un juste milieu. Voir le dessin ci-dessous .



Une fois cet algorithme programmé, on va voir notre première intelligence artificielle apprendre à prédire le prix d'un appartement selon sa surface habitable. on voit comme ci-dessous que l'algorithme arrive à minimiser la Fonction Coût avec le nombre d'itérations.



A partir de là, c'est la porte ouverte aux algorithmes qui automatisent les transactions immobiliàres, et le même concept que celui que vous venez d'apprendre sera appliqué pour apprendre à une machine comment reconnaitre un visage sur une photo, comment prédire le cours de la bourse, etc. Mais avant de voir la magie s'opérer, il faut avoir préalablement calculer les **dérivées partielles** de la Fonction Coût.

Calcul des dérivées partielles

Pour implémenter l'algorithme de Gradient Descent, il faut donc calculer les **dérivées partielles** de la Fonction Coût. On rappelle que , la dérivée d'une fonction en un point nous donne la valeur de sa pente en ce point.

Fonction Coût :

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})^2$$

Dérivée selon le paramètre a :

$$\frac{\partial J(a,b)}{\partial a} = \frac{1}{m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)}) \times x^{(i)}$$

Dérivée selon le paramètre b :

$$\frac{\partial J(a,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})$$

Note :
la dérivée d'une fonction composée :

$$(g \circ f)' = f' \times g' \circ f$$

Dans notre cas : $f = ax + b - y$ et $g = (f)^2$
En dérivant, le carré tombe et se simplifie avec la fraction $\frac{1}{2m}$ pour devenir $\frac{1}{m}$ et $x^{(i)}$ apparait en facteur pour la dérivée par rapport à a.

2.5 La régression linéaire à plusieurs variables-utilisation des matrices et des vecteurs

Algèbre linéaire [19] et [7].
Dans la pratique, on exprime notre Dataset et nos paramètres sous forme **matricielle**, ce qui simplifie beaucoup les calculs. On créer ainsi un **vecteur** $\theta = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{R}^{n+1}$ qui contient tous les paramètres pour notre modèle, un **vecteur** $y \in \mathbb{R}^m$ et une **matrice** $X \in \mathbb{R}^{m\times(n+1)}$ qui inclut toutes les features n. Dans la régression linéaire à une seule variable, $n = 1$.

2.6 Résumé des étapes pour développer un programme de Régression Linéaire

1. Récolter des données (X, y) avec $X \in \mathbb{R}^{m\times(n+1)}$, $y \in \mathbb{R}^m$
2. Donner à la machine un modèle linéaire $F(X) = X.\theta$ où $\theta = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{R}^{n+1}$
3. Créer la Fonction Coût $J(\theta) = \frac{1}{2m}\sum(F(X) - y)^2$
4. Calculer le gradient et utiliser l'algorithme de **Gradient Descent**.

Répeter en boucle :

$$\theta = \theta - a \times \frac{\partial J(\theta)}{\partial \theta}$$

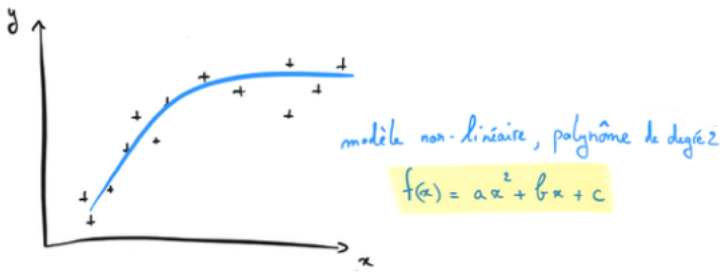
Gradient $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T.(F(X) - Y)$

Le learning rate a prend le nom **d'hyper-paramètre** de par son influence sur la performance finale du modèle (s'il est trop grand où trop petit, la fonction le Gradient Descent ne converge pas).

2.7 Régression Polynômiale à plusieurs variables

Si on achète un stylo à 10 (DA), combien on coûteront 100 stylos ? 1000 (DA) ? Faux !
Nous vivons dans un monde réagit par des lois souvent non-linéaires et une infinité de facteurs peuvent influencer nos résultats.
Par exemple, si on achète 100 stylos, on aura peut-être une réduction à 900 (DA). Si en revanche il y a une pénurie de stylos, ce même stylo qui coûtait 10(DA) pourrait valoir 15 (DA).
C'est là qu'Excel ne pourra plus rien pour nous et que le Machine Learning trouve son utilité dans le monde réel.
Problème non-linéaire : Un problème plus compliqué ?
Pour le nuage de point ci-dessous, il semblerait judicieux de développer un modèle polynômial de degré 2.

$$f(x) = ax^2 + bx + c$$



Nous pouvons améliorer nos features et la forme de notre fonction de prédiction $f(x)$ pour avoir un bon modèle c'est à dire avoir des petites erreurs. La fonction de prédiction $f(x)$ a besoin d'être non linéaire si elle ne correspond pas bien aux données en anglais on dit : it does not fit the data well.

CHAPITRE 3

LA RÉGRESSION LOGISTIQUE/ CLASSIFICATION

Dans l'apprentissage supervisé, il y a deux type de problèmes :

— Les régressions linéaire

— Les régressions logistique (classifications)

Dans ce chapitre, on va découvrir le modèle de Régression Logistique, qui permet de résoudre des problèmes de classification binaires.

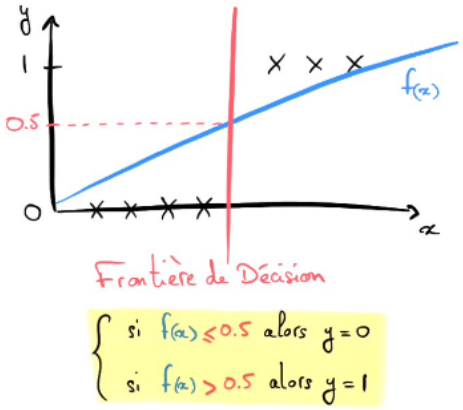
3.1 Les problèmes de Classification

Jusqu'à présent, nous avons appris comment résoudre des problèmes de régression. Au cours de l'introduction, on a parlé des problèmes de classification, qui consistent par exemple à classer un email en tant que 'spam' ou 'non spam'.
Dans ce genre de problème, on aura un Dataset contenant une variable target y pouvant prendre 2 valeurs seulement, par exemple 0 ou 1

— si $y = 0$, alors l'email n'est pas un spam

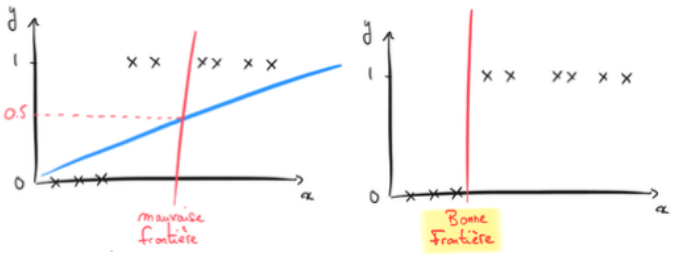
— si $y = 1$, alors l'email est un spam

On dit également que l'on a **2 classes**, c'est une classification **binaire**
Pour ces problèmes, on ajoute au modèle une frontière de décision qui permet de classer un email dans la classe 0 ou la classe 1.

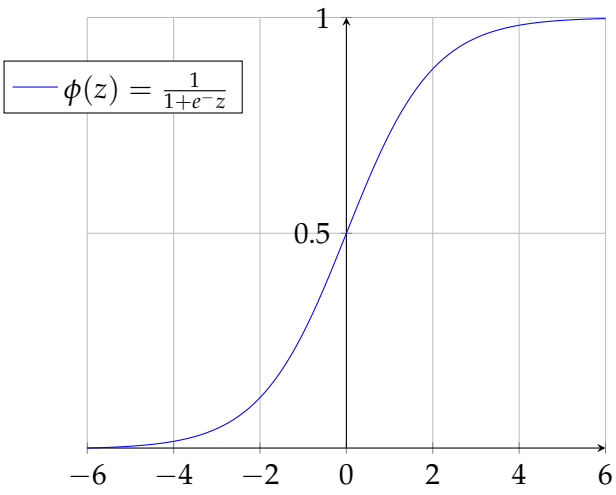


3.2 Le modèle de Régression logistique

Pour les problèmes de classification binaire, un modèle linéaire $F = X.\theta$, comme on l'a tracé sur la figure précédente, ne convient pas. on voit plutôt le résultat que l'on obtient avec un tel modèle pour le Dataset suivant :



On développe alors une nouvelle fonction pour les problèmes de classification binaire, c'est la fonction logistique (aussi appelé fonction sigmoïde ou tout simplement sigma σ). Cette fonction a la particularité d'être toujours comprise en 0 et 1.

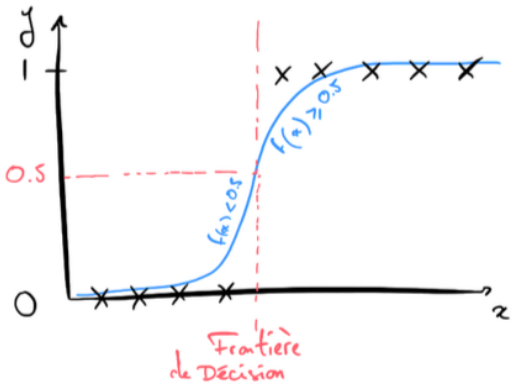


Pour coller la fonction logistique sur un Dataset X, y on y fait passer le produit matriciel $X.\theta$ ce qui nous donne le modèle de Logistic Regression :

$$\sigma(X.\theta) = \frac{1}{1 + e^{-X.\theta}}$$

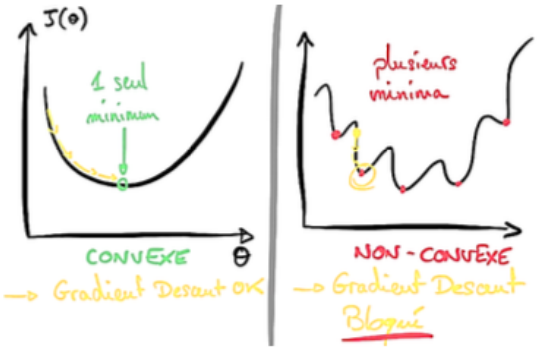
A partir de cette fonction, il est possible de définir une frontière de décision. Typiquement, on définit un seuil à 0.5 comme ceci :

$$y = 0 \quad \text{si} \quad \sigma(X.\theta) < 0.5$$
$$y = 1 \quad \text{si} \quad \sigma(X.\theta) \geq 0.5$$

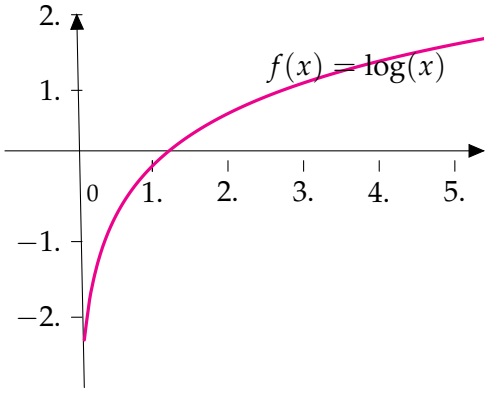


3.3 Fonction Coût associée à la Régression Logistique

Pour la régression linéaire, la Fonction Coût $J(\theta) = \frac{1}{2m}\sum(X.\theta - Y)^2$ donnait une courbe **convexe** (qui présente un unique minima). C'est ce qui fait que l'algorithme de Gradient Descent fonctionne.
En revanche, utiliser cette fonction pour le modèle Logistique ne donnera pas de courbe convexe (dû à la non-linéarité) et l'algorithme de Gradient Descent se **bloquera** au premier minima rencontré, sans trouver le minimum **global**.



Il faut donc développer une nouvelle Fonction Coût spécialement pour la régression logistique. On utilise alors la fonction **logarithme** pour transformer la fonction sigma en fonction convexe en séparant les cas où $y = 1$ des cas où $y = 0$

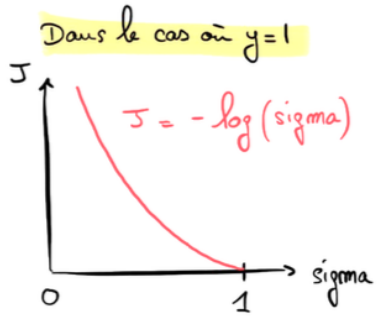


3.3.1 Fonction Coût dans les cas où $y = 1$

Voici la Fonction Coût que l'on utilise dans les cas où $y = 1$:

$$J(\theta) = -\log(\sigma(X.\theta))$$

Explications :
Si notre modèle prédit $\sigma(x) = 0$ alors que $y = 1$, on doit pénaliser la machine par une grande erreur (un grand coût). La fonction logarithme permet de tracer cette courbe avec une propriété convexe, ce qui poussera le Gradient Descent à trouver les paramètres θ pour un coût qui tend vers 0.

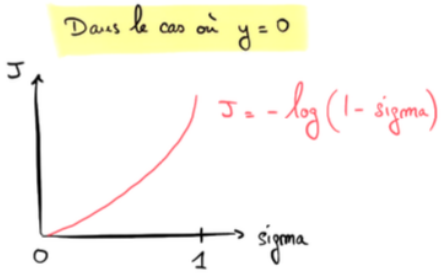


3.3.2 Fonction Coût dans les cas où $y = 0$

Cette fois la Fonction Coût devient :

$$J(\theta) = -\log(1 - \sigma(X.\theta))$$

Explications :
Si notre modèle prédit $\sigma(x) = 1$ alors que $y = 0$, on doit pénaliser la machine par une grande erreur (un grand coût). Cette fois $-log(1 - 0)$ donne la même courbe, inversée sur l'axe vertical.



3.3.3 Fonction Coût complète

Pour écrire la Fonction Coût en une seule équation, on utilise l'astuce de séparer les cas $y = 0$ et $y = 1$ avec une annulation :

$$J(\theta) = \frac{-1}{m} \sum y \times \log(\sigma(X.\theta)) + (1-y) \times \log(1 - \sigma(X.\theta))$$

Dans le cas où $y = 0$, il nous reste :

$$J(\theta) = \frac{-1}{m} \sum \cancel{0 \times \log(\sigma(X.\theta))} + 1 \times \log(1 - \sigma(X.\theta))$$

Dans le cas où $y = 1$:

$$J(\theta) = \frac{-1}{m} \sum 1 \times \log(\sigma(X.\theta)) + \cancel{0 \times \log(1 - \sigma(X.\theta))}$$

3.4 Gradient Descent pour la Régression Logistique

L'algorithme de Gradient Descent s'applique exactement de la même manière que pour la régression linéaire. En plus, la dérivée de la Fonction Coût est la même aussi ! On a :

$$\begin{array}{ll} \text{Gradient :} & \frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} \sum (\sigma(X.\theta) - y).X \\ \text{Gradient Descent :} & \theta = \theta - \alpha \times \frac{\partial J(\theta)}{\partial \theta} \end{array}$$

3.5 Résumé de la Régression Logistique

Modèle	$\sigma(X.\theta) = \frac{1}{1+e^{-X.\theta}}$
Fonction Coût	$\frac{-1}{m} \sum y \times \log(\sigma(X.\theta)) + (1-y) \times \log(1 - \sigma(X.\theta))$
Gradient	$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T.(\sigma(X.\theta) - y)$
Gradient Descent	$\theta = \theta - \alpha \times \frac{\partial J(\theta)}{\partial \theta}$

CHAPITRE 4

PROGRAMMATION

4.1 L'algorithme de Naïve Bayes

Les deux dernière dernière années nous avons abordé la structure générale pour développer des modèle de régression Linéaire et Polynômiale avec SKlearn [12] et nous avons étudié l'algorithme de Nearest Neighbour qui permet de résoudre des problèmes de classification à plusieurs classes de façon simple et très efficace en réalisant le projet de prédiction de survie du Titanic.
Cette année nous nous intéressons à l'algorithme de Naïve Bayes, encore pour résoudre les problèmes de classification, en réalisant un projet de classification des maladies cardiaques (heart diseas classificaton project)

4.2 Introduction

Le Naîve Bayes Classifier (classifieur bayésien naïf ou classifieur naïf de Bayes) est un modèle d'apprentissage automatique de type supervisé permettant la classification en se basant sur le théorème de Bayes avec une forte indépendance des hypothèses (c'est-à-dire pas de corrélation entre les caractéristiques d'un ensemble de données). Cette classification appartient à la catégorie des classifieurs linéaires.

Définition 4.2.1. Nous définirons dans ce qui suit des notions nécessaires à l'apprentissage et la maîtrise du classifieur Naîf Bayesien. Tout d'abord, le théorème de Bayes décrit la probabilité d'une caractéristique en se basant au préalable sur des situations liées à celle-ci. Par exemple, si la probabilité qu'une personne soit diabétique est liée à son âge, alors en utilisant ce théorème, on peut utiliser l'âge pour prédire avec plus de précision la probabilité de souffrir

de cette maladie chronique. Le mot naïf dans l'intitulé de l'algorithme implique que chaque paire de caractéristiques dans l'ensemble des données en question est indépendante de l'autre. C'est-à-dire que la valeur d'une caractéristique particulière est indépendante de la valeur de toute autre caractéristique pour une classe donnée. Par exemple, un fruit est classé comme une pomme s'il est rond, s'il a un diamètre d'environ 13 cm et s'il est de couleur rouge. Avec le Naîve Bayes Classifier, chacune des trois caractéristiques : forme, taille, et couleur contribuent indépendamment à la probabilité que ce fruit soit une pomme. On suppose, en plus, qu'il n'y a pas de corrélation entre la forme, la taille et la couleur. Dans la majorité des cas, vous allez faire face à des caractéristiques continues. Dans ce cas, l'algorithme en question suppose que les valeurs continues associées à chaque classe sont distribuées selon une distribution normale.

4.3 Module Scikit-learn

La bibliothèque Scikit-learn de Python destinée à l'apprentissage automatique fourni le module sklearn.naîve_bayes qui contient plusieurs classificateurs Bayes Naïfs, dont chacun performe mieux sur un certain type de donnée.

4.4 Etapes essentiels d'implémentation avec SKlearn algorithme de Naîve Bayes

- model= GaussianNB() :pour sélectionner l'estimateur et préciser ses hyperparamètres,
- model.fit(x, y) : pour entraîner notre modèle
- model.score(x, y) : pour évaluer notre modèle
- model.predict(x) :pour générer des prédictions.

Les types de classificateurs que la bibliothèque contient sont les suivants :

- Gaussian Naîve Bayes ;
- Multinomial Naîve Bayes ;
- Complement Naîve Bayes ;
- Bernoulli Naîve Bayes ;
- Categorical Naîve Bayes.

Nous nous intéresserons lors de ce mémoire au classificateur Gaussien

4.5 Gaussian Naîve Bayes

Afin de découvrir concrètement le fonctionnement de l'algorithme Gaussian Naîve Bayes, nous l'appliquerons sur un ensemble de données contenant trois caractéristiques : Glucose, pression artérielle et diabète afin de classifier si une personne est atteinte de diabète ou non (1 pour oui 0 sinon). Alors, nous importons dans un premier temps les éléments nécessaires, à savoir : pandas, train_test_split, GaussianNB et accuracy_score. Ensuite, nous chargeons le fichier csv contenant les données et nous affichons les résultats à l'aide de la fonction head(). Ainsi, nous afficherons uniquement les cinq premiers enregistrements pour avoir une idée sur l'ensemble de données.
code

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naîve_bayes import GaussianNB
from sklearn.metrics import accuracy_score
df = pd.read_csv('C:/Users/LENOVO/Desktop/coursGratuit/diabete.csv')
df.head()
```

- Résultat de l'exécution :

```
Entrée [14]: import pandas as pd
             from sklearn.model_selection import train_test_split
             from sklearn.naive_bayes import GaussianNB
             from sklearn.metrics import accuracy_score

             df = pd.read_csv( 'C:/Users/LENOVO/Desktop/coursGratuit/diabete.csv')
             df.head()
```

Out[14]:

	glucose	pressionArterielle	diabete
0	40	85	0
1	40	92	0
2	45	63	1
3	45	80	0
4	40	73	1

On définit ensuite deux variables : x et y représentant respectivement les valeurs de caractéristiques et la valeur cible. Puis, on utilise la fonction train_test_split pour fractionner l'ensemble des données en données de train et de test au pourcentage 80 % pour les données d'entraine-

ment et 20 % pour les données de test. Ensuite, on crée une instance de l'objet GaussianNB et on l'entraine à l'aide de la méthode fit() qu'on lui passe en paramètre x_train et y_train. On fait une prédiction des valeurs de test à l'aide de la méthode predict() pour comparer ensuite le résultat avec les valeurs y_test afin de calculer la précision du modèle.
code

```python
a = df['glucose']
b = df['pressionArterielle']
x = list(zip(a, b))
y = df['diabete']
#fractionner dataset (train-test)
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.20)
#instanciation
model_Gaussian = GaussianNB()
#training
model_Gaussian.fit(x_train, y_train)
#prédiction
prediction = model_Gaussian.predict(x_test)
print(prediction)
#evaluation du modèle
precision = accuracy_score(y_test, prediction)*100
print(precision)
```

Résultat de l'exécution :

Entrée [9]:
```
a = df['glucose']
b = df['pressionArterielle']
x = list(zip(a, b))
y = df['diabete']

#fractionner dataset (train-test)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)

#instanciation
model_Gaussian = GaussianNB()

#training
model_Gaussian.fit(x_train, y_train)

#prédiction
prediction = model_Gaussian.predict(x_test)
print(prediction)

#evaluation du modèle
precision = accuracy_score(y_test, prediction)*100
print(precision)
```

```
[0 1 1 0 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0
 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 1 0 0 0 1
 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 0 1 0 1
 0 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 1
 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0
 1 0 0 1 0 1 1 1 0 1 1 1 0 1]
92.96482412060301
```

4.6 Projet : heart disease classification(classification des maladies cardiaques)

Dans ce projet, le dataset est un échantillon des gens dans des régions en Europe.

L'objectif de ce projet est de construire un modèle de classification pour prédire quels patients sont les plus susceptibles de souffrir d'une maladie cardiaque dans un avenir proche en utilisant les caractéristiques fournies et en appliquant l'algorithme de Naîve Bayes.
Ce dataset indique pour chaque patient les informations suivantes :

- **age** : l'âge d'un patient
- **sex**(1 = male ; 0 = female) : le sexe d'un patient (1 = masculin ; 0 = féminin)
- **cp** (chest pain type) : type de douleur thoracique
- **trestbps** (resting blood pressure (in mm Hg on admission to the hospital) : tension artérielle au repos (en mmHg à l'admission à l'hôpital)
- **chol** (serum cholestoral in mg/dl) : cholestérol sérique en mg/dl

- **fbs** (fasting blood sugar > 120 mg/dl (1 = true ; 0 = false)) : glycémie à jeun > 120 mg/dl) (1 = vrai ; 0 = faux)

- **restecg** (resting electrocardiographic results) : résultats électrocardiographiques au repos

- **thalach** (maximum heart rate achieved) : fréquence cardiaque maximale atteinte

- **exang** (exercise induced angina (1 = yes ; 0 = no)) : angine induite par l'exercice (1 = oui ; 0 = non)

- **oldpeak**(ST depression induced by exercise relative to rest) : Dépression ST induite par l'exercice par rapport au repos

- **slope**(the slope of the peak exercise ST segment) : la pente du segment ST d'exercice maximal

- **ca**(number of major vessels (0-3) colored by flourosopy) : nombre de vaisseaux principaux (0-3) colorés par fluoroscopie

- **thal**(thal - 3 = normal ; 6 = fixed defect ; 7 = reversable defect) : tal - 3 = normal ; 6 = défaut corrigé ; 7 = défaut réversible
  target : (Displays whether the individual is suffering from heart disease or not : 1 = yes 0 = no) : affiche si le patient souffre ou non d'une maladie cardiaque : 1 = oui 0 = non.

Voilà le projet réalisé sur Notebook Jupiter.

heart23

March 3, 2023

1 Préparation et Traitement des données

1.1 Importer les bibliothèques et le dataset

[6]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

1.2 Afficher les 5 premier lignes DataFrame.

[7]:
```
df=pd.read_csv("heart.csv")
df.head()
```

[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

1.3 Data contains;

age - age in years

sex - (1 = male; 0 = female)

cp - chest pain type

trestbps - resting blood pressure (in mm Hg on admission to the hospital)

chol - serum cholestoral in mg/dl

fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

restecg - resting electrocardiographic results

thalach - maximum heart rate achieved

exang - exercise induced angina (1 = yes; 0 = no)

oldpeak - ST depression induced by exercise relative to rest

slope - the slope of the peak exercise ST segment

ca - number of major vessels (0-3) colored by flourosopy

thal - 3 = normal; 6 = fixed defect; 7 = reversable defect

target - have disease or not (1=yes, 0=no)

1.4 Afficher la dimensien de DataFrame

[8]:
```
df.shape
```

[8]:
```
(1025, 14)
```

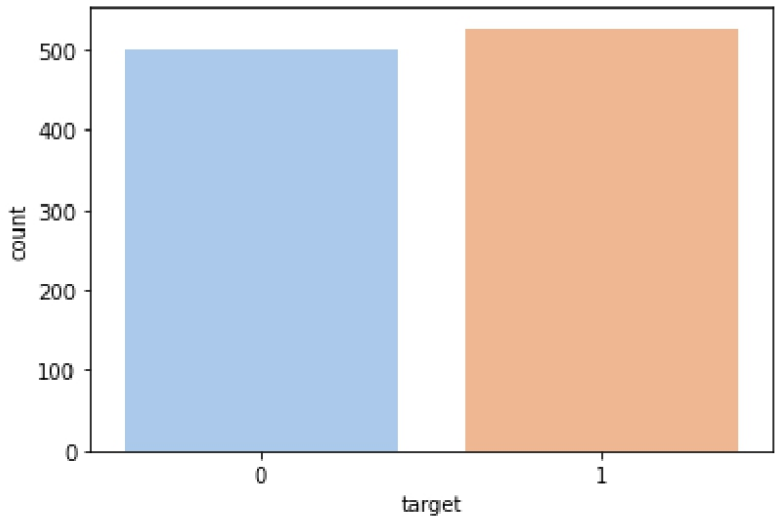
1.5 Compter les fréquences d'éléments dans la collone target

[9]:
```
df.target.value_counts()
```

[9]:
```
1    526
0    499
Name: target, dtype: int64
```

1.6 Afficher ces derniers resultats dans un diagramme à barre

[10]:
```
sns.countplot(x='target',data=df,palette='pastel')
plt.show()
```



[11]:
```
print("Percentage of patients with Heart disease:{:.2f}%".format(len(df[df.
↪target==1])*100/len(df.target)))

print("Percentage of patients with no Heart disease:{:.2f}%".format(len(df[df.
↪target==0])*100/len(df.target)))
```

```
Percentage of patients with Heart disease:51.32%
Percentage of patients with no Heart disease:48.68%
```

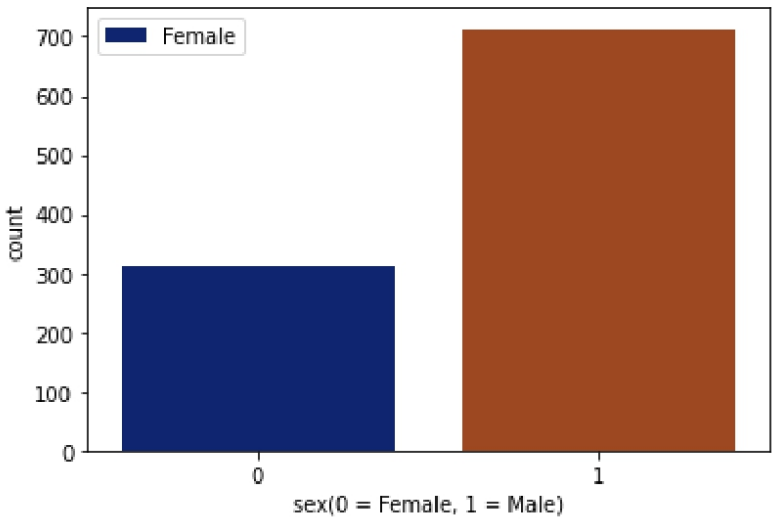
1.7 Compter les fréquences d'éléments dans la collone sex

[12]:
```
df.sex.value_counts()
```

[12]:
```
1    713
0    312
Name: sex, dtype: int64
```

[13]:
```
sns.countplot(x='sex',data=df,palette="dark")
plt.legend(["Female","Male"])
plt.xlabel('sex(0 = Female, 1 = Male)')
```

```
plt.show()
```



[14]:
```
print("Percentage of Female Patients:{:.2f}%".format(len(df[df.sex==0])*100/
 ↪len(df.sex)))
print("Percentage of Male Patients:{:.2f}%".format(len(df[df.sex==1])*100/
 ↪len(df.sex)))
```

```
Percentage of Female Patients:30.44%
Percentage of Male Patients:69.56%
```

1.8 Regrouper les collonnes selon la collone target en utilsant la moyenne

[15]:
```
df.groupby('target').mean()
```

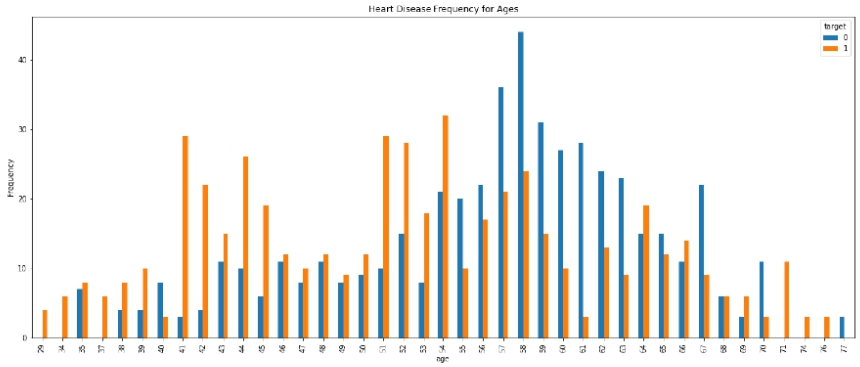
[15]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	56.569138	0.827655	0.482966	134.106212	251.292585	0.164329	0.456914	139.130261	0.549098	1.600200	1.166333	1.158317	2.539078
1	52.408745	0.570342	1.378327	129.245247	240.979087	0.134981	0.598859	158.585551	0.134981	0.569962	1.593156	0.370722	2.119772

[16]:
```
df.groupby('target').mean()
```

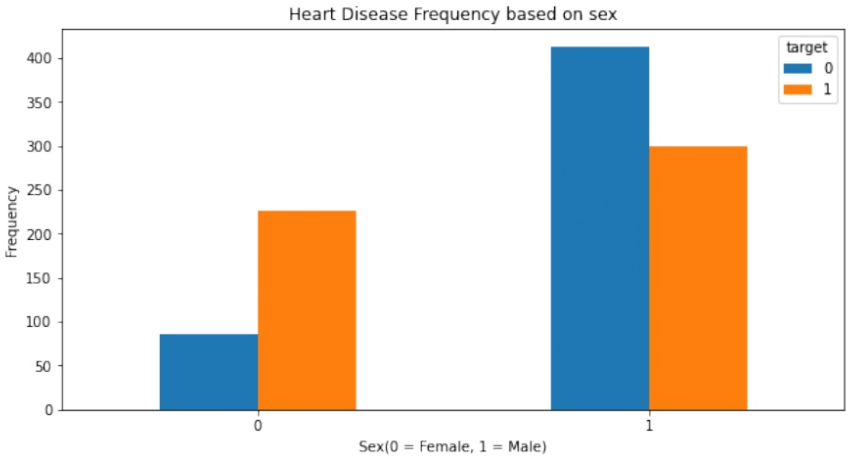
[16]:
```
                age       sex        cp    trestbps        chol       fbs  \
target
0         56.569138  0.827655  0.482966  134.106212  251.292585  0.164329
1         52.408745  0.570342  1.378327  129.245247  240.979087  0.134981

           restecg     thalach     exang   oldpeak     slope        ca      thal
target
0         0.456914  139.130261  0.549098  1.600200  1.166333  1.158317  2.539078
1         0.598859  158.585551  0.134981  0.569962  1.593156  0.370722  2.119772
```

[17]:
```
pd.crosstab(df.age,df.target).plot(kind='bar',figsize = (20, 8))
plt.title('Heart Disease Frequency for Ages')
plt.ylabel('Frequency')
plt.show()
```

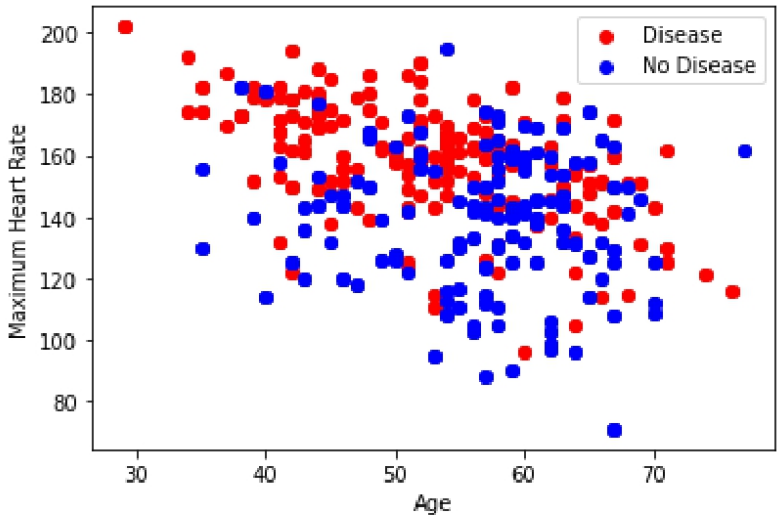


1.9 Fréquence des maladies cardiaques basée sur le sexe

[17]:
```
pd.crosstab(df.sex,df.target).plot(kind='bar',figsize = (10, 5))
plt.title('Heart Disease Frequency based on sex')
plt.xticks(rotation=0)
plt.xlabel('Sex(0 = Female, 1 = Male)')
plt.ylabel('Frequency')
plt.show()
```

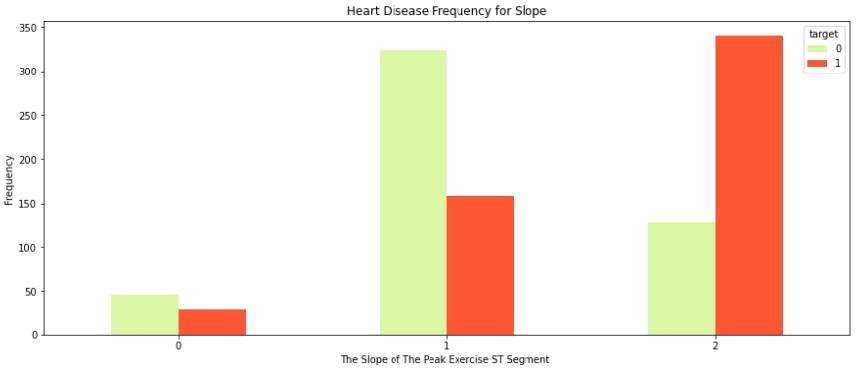


[18]:
```
plt.figure(figsize=(6,4))
plt.scatter(x=df.age[df.target==1],y=df.thalach[(df.target==1)],c='red')
plt.scatter(x=df.age[df.target==0],y=df.thalach[(df.target==0)],c='blue')
plt.legend(['Disease','No Disease'])
plt.xlabel('Age')
plt.ylabel('Maximum Heart Rate')
plt.show()
```

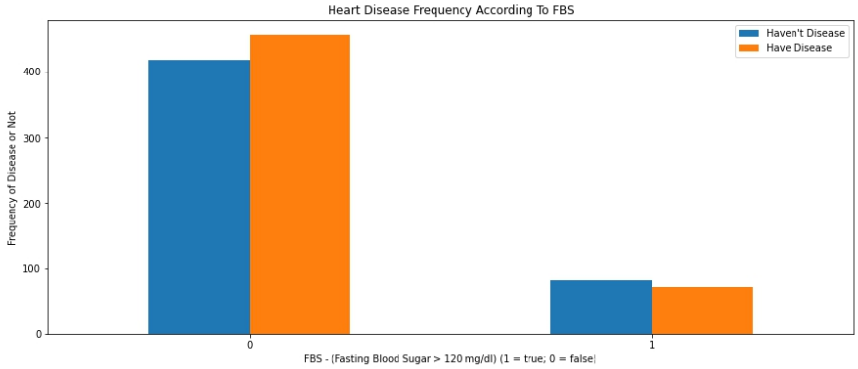


1.10 Fréquence des maladies cardiaques pour le slope

[19]:
```
pd.crosstab(df.slope,df.target).
 ↪plot(kind="bar",figsize=(15,6),color=['#DAF7A6','#FF5733' ])
plt.title('Heart Disease Frequency for Slope')
plt.xlabel('The Slope of The Peak Exercise ST Segment ')
plt.xticks(rotation = 0)
plt.ylabel('Frequency')
plt.show()
```

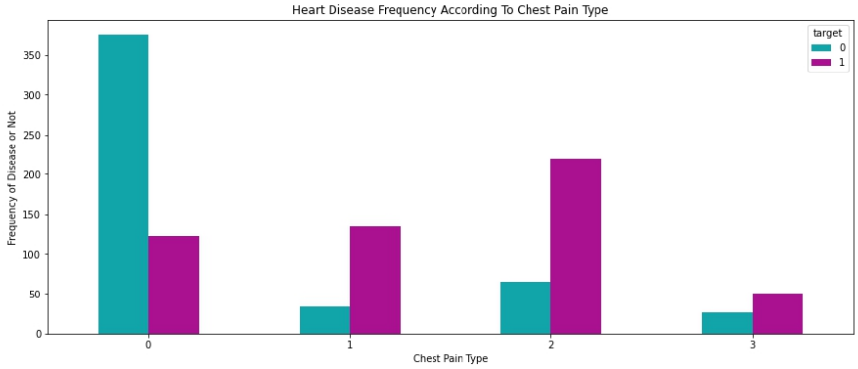


[21]:
```
pd.crosstab(df.fbs,df.target).plot(kind="bar",figsize=(15,6))
plt.title('Heart Disease Frequency According To FBS')
plt.xlabel('FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
plt.xticks(rotation = 0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency of Disease or Not')
plt.show()
```



1.11 Fréquence des maladies cardiaques selon To Chest Pain Type

[20]:
```
pd.crosstab(df.cp,df.target).
↪plot(kind="bar",figsize=(15,6),color=['#11A5AA','#AA1190' ])
plt.title('Heart Disease Frequency According To Chest Pain Type')
plt.xlabel('Chest Pain Type')
plt.xticks(rotation = 0)
plt.ylabel('Frequency of Disease or Not')
plt.show()
```



1.12 Classification desmaladies cardiaques

[21]:
```
y = df.target.values
x = df.drop(['target'], axis = 1)
```

[22]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.
↪3,random_state=0)
```

2 Logistic Regression

[23]:
```
accuracies = {}

lr = LogisticRegression()
lr.fit(x_train,y_train)
acc = lr.score(x_test,y_test)*100

accuracies['Logistic Regression'] = acc
print("Test Accuracy {:.2f}%".format(acc))
```

```
Test Accuracy 87.01%
```

3 K-nearest Neighbour

[24]:
```
# KNN Model
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 2)  # n_neighbors means k
knn.fit(x_train, y_train)
prediction = knn.predict(x_test)
acc=knn.score(x_test, y_test)*100
accuracies['KNN'] = acc
print("{} NN Score: {:.2f}%".format(2, acc))
```

```
2 NN Score: 92.53%
```

4 Naive Bayes Algorithm

[25]:
```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)

acc = nb.score(x_test,y_test)*100
accuracies['Naive Bayes'] = acc
print("Accuracy of Naive Bayes: {:.2f}%".format(acc))
```

```
Accuracy of Naive Bayes: 84.42%
```

Bibliographie

[1] S. V. ALEX SMOLA, *Introduction to Machine Learning*, Cambridge University Press (2008).

[2] D. BARBER, *Bayesian Reasoning and Machine Learning*, Cambridge University Press, 2011.

[3] C. M. BISHOP, *Pattern Recognition and Machine Learning*, Springer, 2006.

[4] L. BREIMAN, *Statistical Modeling : The Two Cultures (with comments and a rejoinder by the author)*, Statistical Science, 16 (2001), pp. 199 – 231.

[5] A. P. DAWID, *Conditionnal independence in statistical theory.*, Journal of the Royal Statistical Society,, (1979), pp. 1–39.

[6] T. HASTIE, R. TIBSHIRANI, AND J. FRIEDMAN, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[7] J. HEFFERON, *Linear Algebra*, SBN 978-1-944325-11-4, OCLC 1178900366, OL 30872051M, 2020.

[8] F. V. JENSEN, *An introduction to Bayesian networks*, 1996.

[9] A. NG, *Machine Learning Yearning*, Online Draft, 2017.

[10] N. J. NILSSON, *Introduction to machine learning : An early draft of a proposed textbook. pages 175-188. http ://robotics.stanford.edu/people/nilsson/mlbook.ht ml*, 1996.

[11] J. PEARL, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[12] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT, AND E. DUCHESNAY, *Scikit-learn : Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[13] G. SAINT-CIRGUE, *Machine Learning*, machinelearnia, 2019.

[14] A. L. SAMUEL, *Some studies in machine learning using the game of checkers*, IBM J. Res. Dev., 3 (1959), pp. 210–229.

[15] S. SHALEV-SHWARTZ AND S. BEN-DAVID, *Understanding Machine Learning - From Theory to Algorithms.*, Cambridge University Press, 2014.

[16] O. SIMEONE, *A brief introduction to machine learning for engineers*, ArXiv, abs/1709.02840 (2018).

[17] A. M. TURING, *I.—COMPUTING MACHINERY AND INTELLIGENCE*, Mind, LIX (1950), pp. 433–460.

[18] T. VERMA AND J. PEARL, *Causal networks : Semantics and expressiveness* *this work was partially supported by the national science foundation grant iri-8610155. "graphoids : A computer representation for dependencies and relevance in automated reasoning (computer information science)."*, in Uncertainty in Artificial Intelligence, R. D. SHACHTER, T. S. LEVITT, L. N. KANAL, and J. F. LEMMER, eds., vol. 9 of Machine Intelligence and Pattern Recognition, North-Holland, 1990, pp. 69–76.

[19] B. WISE AND N. GALLAGHER, *An introduction to linear algebra*, Critical Reviews in Analytical Chemistry, 28 (1998), pp. 1–19.