

Département de Mathématiques
Filière : Licence Sciences et Techniques

Mathématiques Appliquées

Mémoire de Projet de Fin d'Etudes

Présenté par :

EL BELGHITI Hamza et **MOUNADI** Ayoub

Modèle de prédiction du prix de location d'appartements sur une plateforme de marché Internet sous python

Soutenu le : 04 juillet 2022

Sous la direction de :

M. HARFAOUI Professeur à la Faculté des Sciences et Techniques Mohammedia.

Devant le jury composé de :

M. HARFAOUI Professeur à la Faculté des Sciences et Techniques Mohammedia.
M.R. AMATTOUCH Professeur a la Faculté des Sciences et Techniques Mohammedia.

Résumé

Parmi les défis auxquels les hôtes Airbnb sont confrontés est de déterminer le prix de location optimal par nuitée. Dans de nombreuses régions, les locataires potentiels verront une grande sélection d'annonces et pourront filtrer selon des critères tels que le prix, le nombre de chambres, la superficie, etc. Airbnb étant une place de marché, le montant qu'un hôte peut facturer par nuit est étroitement lié à la dynamique du marché. L'objectif de ce projet est de pouvoir prédire grâce à des techniques d'apprentissage automatique supervisé le prix idéal qu'un hôte pourrait facturer de tel sorte qu'il soit en accord avec les prix du marché. La question principale de ce projet est la suivante : Comment déterminer le prix de location idéal d'un appartement en se basant sur ses caractéristiques grâce à un modèle d'apprentissage automatique supervisé ? Pour répondre à cette question, nous avons utilisé un algorithme d'apprentissage supervisé appelé l'algorithme des K-plus proches voisins ou KNN, pour automatiser la stratégie appliquée par les hébergeurs pour déterminer le prix de location optimal. Plusieurs versions du modèle ont été implémentées, du modèle univarié au modèle multivarié afin de faire des comparaisons entre leurs différentes prédictions, et d'évaluer chacune d'elles pour trouver le modèle le plus performant de tous et le choisir comme solution à notre problème. Le modèle résultant de ce projet pourrait être intégré dans la partie back-end du site Airbnb, chose qui permettra aux propriétaires de déterminer plus facilement le prix idéal pour louer leur maison/appartement en fonction de ses caractéristiques.

Mots clés : Apprentissage automatique, prédiction du prix de location d'appartements, apprentissage supervisé, régression, K plus proches voisins

Abstract

One of the challenges that Airbnb hosts face is determining the optimal nightly rent price. In many areas, potential renters are presented with a good selection of listings and can filter by criteria like price, number of bedrooms, square footage, and more. Since Airbnb is a marketplace, the amount a host can make per night is closely tied to market dynamics. The main objective of this project is to be able to predict, using supervised machine learning techniques, the ideal price that a host can charge, in such a way that we do not overcharge and neither undercharge. The main question of this project is : How can we hit the sweet spot in the middle ? How do we predict the optimal rent price for an apartment based on its characteristics through a supervised machine learning model? To answer this question, we have used a supervised learning algorithm called the K-Nearest Neighbors or KNN for short, to automate the strategy applied by hosts to determine the optimal renting price. Several versions of the model have been implemented, from the univariate model to the multivariate model in order to make comparisons between their different predictions, and to evaluate each one of them to find the most efficient model of all as a solution to our problem. The result of this project could be integrated into the back-end part of the Airbnb website, which will make it easier for owners to determine the ideal price for renting their home/apartment based on its features.

Keywords : Machine learning, apartment rental price prediction, supervised learning, regression, K nearest neighbors

Table des matières

Introduction générale	8
1 Apprentissage automatique	12
1.1 Définition	12
1.1.1 Exemples	13
1.2 Catégories d'apprentissage automatique	15
1.2.1 Apprentissage en profondeur	15
1.2.2 Apprentissage par renforcement	17
1.2.3 Apprentissage non supervisé	18
1.2.4 Apprentissage supervisé	18
2 Algorithmes d'apprentissage automatique	23
2.1 Algorithmes d'apprentissage non supervisé	23
2.1.1 Analyse en composantes principales (ACP)	23
2.1.2 K moyennes clustering	28
2.2 Algorithmes d'apprentissage supervisé	30
2.2.1 Classification naïve de Bayes	30
2.2.2 Arbres de décision et forêts aléatoires	33
2.2.3 Régression linéaire	35
2.2.4 Régression logistique	36
2.2.5 K plus proches voisins (KNN)	38
3 Problématique et préparation de la base de données	42
3.1 Introduction	42
3.2 Importation de la base de données	43
3.3 Exploration de la base de données	44
4 Création et évaluation du modèle	46
4.1 Implementation du K-NN	46
4.1.1 Choix de la distance	47
4.1.2 Construction d'un modèle basique	48

4.1.3	Évaluation du modèle	51
4.2	Comparaison des modèles	53
4.2.1	Calcul de la distance euclidienne avec plusieurs caractéristiques	56
4.3	Création d'un modèle KNN multivarié	57
4.3.1	Création d'un modèle à l'aide de Scikit-learn	58
5	Conclusion	64
	Bibliographie	66

Table des figures

1	Organigramme fonctionnel de la faculté	11
1.1	Classification d'email en deux classes "spam" ou "pas spam"	13
1.2	Détection des signes et panneaux routiers par segmentation d'images.	13
1.3	Détection du cancer du sein à l'aide de l'apprentissage en profondeur	14
1.4	Prédiction du prix d'un bien immobilier	14
1.5	Classification d'une image par une réseau de neurones	15
1.6	L'équipe OG et l'équipe de développement OpenAI après le match historique.	16
1.7	Exemples d'images générées par le modèle DALL-E	16
1.8	Apprentissage par renforcement	17
1.9	Représentation des données sur un plan à deux caractéristiques (x,y)	19
1.10	Représentation visuelle de la classification des points grâce a une droite	20
1.11	Ensemble de points de données représentés sur un plan avec la droite de régression qui représente l'ensemble des points prédis	21
1.12	Le phénomène du décalage vers le rouge (redshift) observé à travers le télescope Hubble.	22
2.1	Les cinq premières observations de la base de données Iris de taille (150x4) .	24
2.2	Matrice de covariance de l'ensemble de données Iris	25
2.3	Valeurs propres de la matrice de covariance de l'ensemble de données de l'iris	25
2.4	Vecteurs propres de la matrice de covariance de l'ensemble de données de l'iris	26
2.5	Selection des 2 vecteurs propres dont les valeurs propres sont les plus élevées	26
2.6	Étapes visuelles des étapes appliquées	27
2.7	Tracé de données réduites en 2D	27
2.8	Groupement des points de données en K groupes	28
2.9	Données unidimensionnelles	29
2.10	Représentation visuelle des étapes de l'algorithme K-Moyennes	29
2.11	Le jeu de données décrivant le problème de jouer au golf selon la météo . . .	30
2.12	Classification d'objets grâce à un classifieur bayésien naïf	32
2.13	Arbre de décision pour identifier le type d'une fleur	33
2.14	Visualisation de l'algorithme forêts aléatoires	34

2.15 Fonction sigmoid	36
2.16 Matrice de confusion	37
2.17 Visualisation de l'algorithme KNN pour une nouvelle donnée non étiquetée à prédire pour K=3 voisins	39
2.18 Représentation graphique des points dans un plan 2D et leurs séparations en régions	40
3.1 L'expérience de recherche d'un logement sur la plateforme Airbnb	42
3.2 Importation de la base de données	43
3.3 Sélection des colonnes les plus importantes et affichage des 5 premières lignes	45
3.4 La somme des données manquantes pour chaque colonnes	45
3.5 Suppression des lignes contenant des données manquantes pour les colonnes bedrooms, bathrooms et beds	45
4.1 Premièrement, nous sélectionnons le nombre d'annonces similaires K	46
4.2 Deuxièmement, nous calculons la similarité de chaque annonce avec la nôtre en utilisant une métrique de similarité.	46
4.3 Troisièmement, nous classons chaque annonce à l'aide de notre métrique de similarité et on sélectionne les K premières annonces.	47
4.4 Enfin, nous calculons le prix moyen pour les K annonces similaires et l'utilisons comme prix d'annonce.	47
4.5 Calcul de la distance entre le nouveau point de donnée et la première observation de la base de données	48
4.6 Calcul de la distance avec tous les observations de la base de données	49
4.7 La randomisation de la base de données puis le tri des distances de façon croissante	49
4.8 L'affichage des 5 premières lignes ordonnées aléatoirement avec le prix associé	50
4.9 Suppression du symbole \$ et calcul du prix moyen	50
4.10 Séparation de l'ensemble de données en deux sous-ensembles : un pour l'entraînement et l'autre pour le test	51
4.11 Division de l'ensemble de données d'origine en deux sous-ensembles	51
4.12 Construction de la fonction KNN qui prédit un prix moyen d'une nouvelle observation	52
4.13 Prédiction des prix grâce à la fonction predict_price	52

4.14 Calcul de RMSE du modèle	53
4.15 Calcul de RMSE du modèle pour différentes caractéristiques	53
4.16 Les cinq premières lignes de la base de données normalisée	55
4.17 Division de l'ensemble de données en deux sous ensembles, un pour l'entraînement et un pour le test	55
4.18 Distance euclidienne entre les deux premières lignes dans la BDD normalisée	56
4.19 Distance euclidienne entre la 1ère et la 5ème ligne	57
4.20 Fonction K-NN de prédiction d'un modèle multivarié	57
4.21 Initialisation du modèle KNeighborsRegressor depuis la librairie Scikit-learn	59
4.22 Ajustement du modèle aux données de l'ensemble d'apprentissage et prédictions des valeurs de prix	60
4.23 Calcul de RMSE à l'aide de scikit-learn	61
4.24 Matrice de corrélation	62
4.25 Calcul du RMSE avec un modèle à 4 caractéristiques	63
4.26 Calcul du RMSE normalisé	63

Introduction générale

L'apprentissage automatique, appartient au champ de l'intelligence artificielle. On attribue généralement ses débuts aux travaux du mathématicien britannique Alan Turing qui imagine une épreuve, le test de Turing, censée déterminer si une machine peut simuler la pensée humaine. En 1959, l'informaticien américain Arthur Samuel utilise pour la première fois le terme « machine learning », pour son programme créé en 1952. Celui-ci est capable de jouer aux dames et d'apprendre au fur et à mesure de ses parties. Le début des années 1990 marque la renaissance du machine learning, et cela est surtout liée à l'augmentation de puissance de calcul des ordinateurs qui entraîne des progrès substantiels. De plus, l'essor d'Internet offre de nouvelles possibilités aux chercheurs. Ceux-ci peuvent désormais partager leurs résultats plus facilement, ce qui leur permet d'avancer plus vite. De même, ils ont alors accès à de nouvelles données, en plus grandes quantités, une variable incontournable de l'apprentissage automatique.

Contrairement à beaucoup d'autres programmes, un modèle d'apprentissage automatique n'est pas un système basé sur des règles dans lequel une série d'instructions "si/alors" sont utilisées pour déterminer les résultats. L'apprentissage automatique consiste à créer des modèles mathématiques pour aider à comprendre les données. «L'apprentissage» entre en jeu lorsque l'on donne à ces modèles des paramètres réglables et adaptables aux données observées; de cette manière, on peut considérer que le programme "apprend" à partir des données. Une fois que ces modèles ont été adaptés aux données précédemment observées, ils peuvent être utilisés pour prédire et comprendre les aspects des données nouvellement observées.

Problématique

Imaginons que nous ayons une maison que nous aimerais louer et que nous voudrions déterminer un prix de location approprié. Ce que nous pourrions faire est de regarder des maisons comparables à la nôtre, dans notre région et qui ont déjà été précédemment louées. Pour chaque maison, nous voudrions prendre en compte des facteurs tels que la taille de la maison, le nombre de chambres et de salles de bains dont elle dispose, sa distance par rapport aux commodités telles que les épiceries, etc.

Une fois que nous avons trouvé un certain nombre de maisons similaires, nous pouvons alors examiner leurs prix de location, faire la moyenne de ces prix et arriver à une prédition assez raisonnable du prix de notre propriété. Essayer de faire ce genre de prédictions à plus grande échelle, comme prédire le prix de n'importe quelle maison dans une ville serait incroyablement difficile pour un humain. C'est là où l'apprentissage automatique entre en jeu.

Dans ce projet de fin d'étude effectué au sein de la Faculté de Sciences et Techniques de Mohammedia, nous allons créer un modèle de recommandations à l'aide de Python, qui prédira le prix de location d'appartements sur Airbnb ; un service de plateforme communautaire payant de location de logements de particuliers. Les deux premiers chapitres discuteront l'apprentissage automatique, ses catégories et quelques-uns de ses algorithmes. Ensuite, pour les deux derniers chapitres nous implémenterons un modèle de recommandations basé sur l'algorithme K plus proche voisins (KNN), ainsi que nous évaluerons sa performance pour ses différentes variations.

Présentation da la faculté

La Faculté de Sciences et Techniques de Mohammedia (FSTM), composante de l'Université Hassan II de Casablanca, fait partie d'un réseau de huit FST à travers le Maroc dont la vocation est la formation universitaire dans les sciences et techniques, la FSTM, implantée dans une zone à forte activité industrielle, a toujours veillé à s'intégrer dans son environnement en disposant des cursus aboutissant à des profils qui répondent par excellence aux besoins socio-économiques. Elle œuvre depuis son inauguration, en 1994, à offrir une formation technique et en ingénierie de qualité, et s'est distinguée par la pertinence de sa recherche.

La FST de Mohammedia compte 186 enseignants chercheurs, 57 personnels administratifs et techniques et plus de 3000 étudiants dont 750 en dernière année de licence, 260 en master et 440 en filières d'ingénieurs. Elle délivre plus de 550 diplômés par an. C'est un établissement à accès régulé où l'entrée se fait après une sélection basée sur les notes au baccalauréat de matières définies en fonction du parcours choisi (MIP ou BCG). Les formations de la recherche au sein de la FSTM couvrent différents champs disciplinaires en Biologie, Mathématiques et Physiques. La FSTM propose une carte de formation riche et relativement complète en Licence, Master Sciences et Techniques et Filières d'ingénieurs.

Missions et valeurs

La Faculté des Sciences et Techniques de Mohammedia (FSTM) regroupe des structures qui s'investissent pleinement dans les domaines de la formation et de la recherche scientifique pour contribuer au développement du Maroc. Ces structures sont engagées pour :

- Assurer des formations scientifiques et techniques de qualité en adéquation avec le marché de l'emploi.
- Dispenser des formations continues pour répondre aux besoins du secteur socio-économique.
- Développer la recherche scientifique fondamentale et appliquée pour contribuer à la production du savoir, répondre au besoin des entreprises et encourager l'innovation.
- Participer au progrès scientifique, technique, professionnel et économique du pays.

La FSTM véhicule des valeurs basées sur l'intégrité intellectuelle et morale :

- La FSTM s'engage à développer le professionnalisme, le sens de la responsabilité, la créativité et le respect d'autrui.
- La FSTM s'engage à respecter les principes d'équité, d'égalité des chances et de la pluralité culturelle.
- La FSTM s'engage à émuler l'excellence, la créativité et l'innovation.

Département Mathématique

La FSTM compte 8 départements, 13 laboratoires et un centre d'études doctorales constitué de 2 formations doctorales couvrant différents champs disciplinaires en Biologie, Chimie, Génie des procédés, Génie électrique, Physique, Informatique et Mathématiques.

Le département de Mathématiques compte 29 professeurs et 2 laboratoires ; le laboratoire de mathématiques, cryptographie, mécanique et analyse numérique (LMCMAN) et le laboratoire de mathématiques, informatique et applications (LMCSA). Parmi ces axes stratégiques, il vise à valoriser les compétences acquises par l'étudiant durant le cursus licence en développant le lien avec le milieu socio-économique, assurer la cohérence des parcours de formation avec les recherches développées dans les deux laboratoires qui lui sont associés ; ainsi que la favorisation de toute action visant à améliorer l'insertion professionnelle des étudiants.

Organisation générale

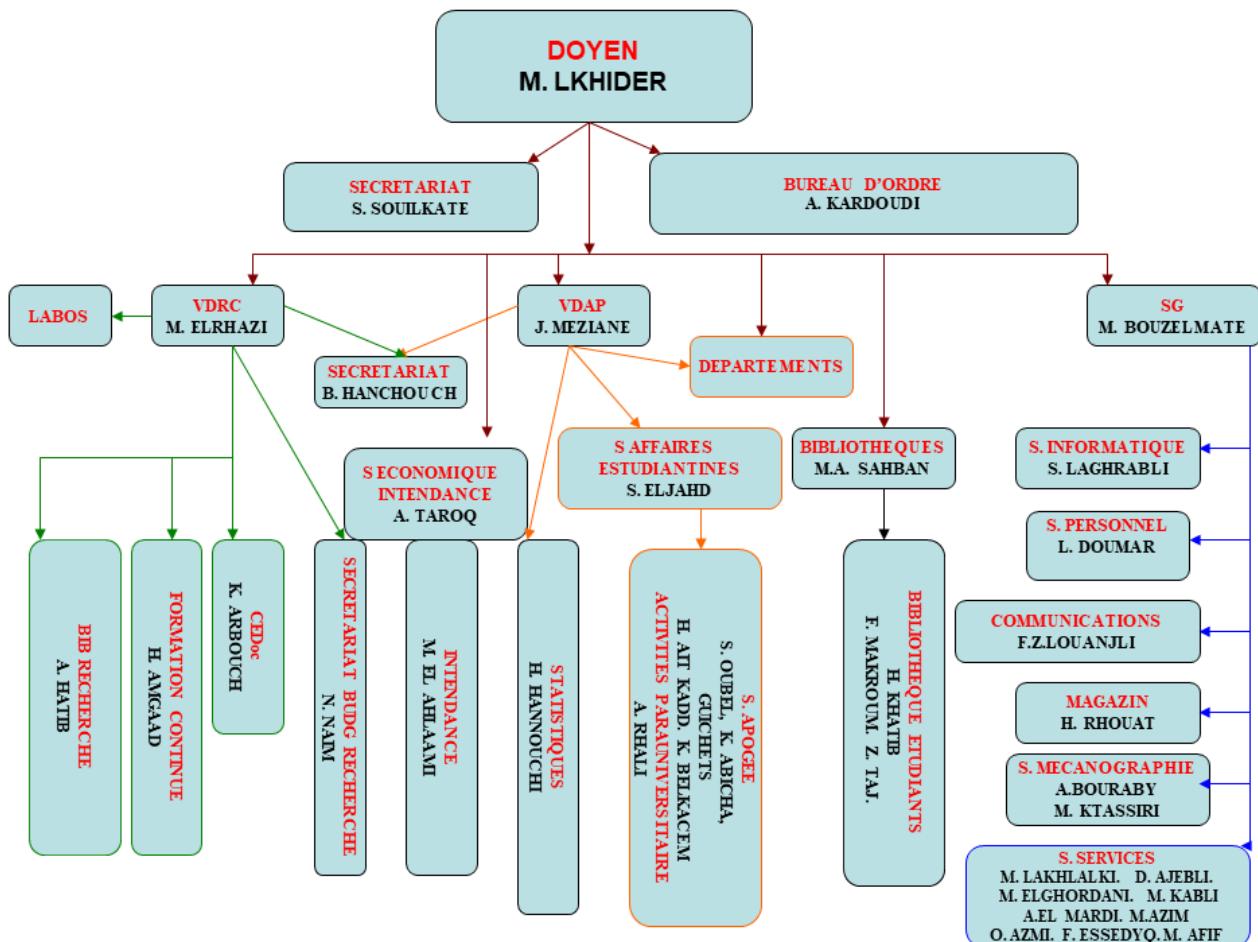


FIGURE 1 – Organigramme fonctionnel de la faculté

Apprentissage automatique

1.1 Définition

L'apprentissage automatique (*machine learning*) ou apprentissage statistique est une forme d'intelligence artificielle qui est axée sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'« apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. L'apprentissage automatique et l'intelligence artificielle sont souvent abordés ensemble et ces termes sont parfois utilisés de manière interchangeable bien qu'ils ne renvoient pas exactement au même concept. Une distinction importante est que, même si l'intégralité du machine learning repose sur l'intelligence artificielle, cette dernière ne se limite pas au machine learning.

D'après IBM, l'une des entreprises pionnières de cette industrie, la définition de l'apprentissage automatique est : "L'apprentissage automatique, également appelé apprentissage machine ou apprentissage artificiel et en anglais machine learning, est une forme d'intelligence artificielle (IA) qui permet à un système d'apprendre à partir des données et non à l'aide d'une programmation explicite. Cependant, l'apprentissage automatique n'est pas un processus simple. Au fur et à mesure que les algorithmes ingèrent les données de formation, il devient possible de créer des modèles plus précis basés sur ces données. Un modèle de machine learning est le résultat généré lorsque vous entraînez votre algorithme d'apprentissage automatique avec des données. Après la formation, lorsque vous fournissez des données en entrée à un modèle, vous recevez un résultat en sortie. Par exemple, un algorithme prédictif crée un modèle prédictif. Ensuite, lorsque vous fournissez des données au modèle prédictif, vous recevez une prévision qui est déterminée par les données qui ont servi à former le modèle." [1]

Aujourd’hui, nous utilisons l’apprentissage automatique dans tous les domaines. Lorsque nous interagissons avec les banques, achetons en ligne ou en utilisant les réseaux sociaux, des algorithmes d’apprentissage automatique entrent en jeu pour optimiser, fluidifier et sécuriser notre expérience. L’apprentissage automatique et la technologie qui l’entoure se développent rapidement, et nous commençons seulement à entrevoir ses capacités.

1.1.1 Exemples

Pour illustrer l’utilité de l’apprentissage automatique, considérons les exemples suivants :

Exemple 1 : Supposons qu’on dispose d’un nombre important de mails dans votre boîte mail. Comment classer ces emails comme étant “spam” ou “pas spam” ?

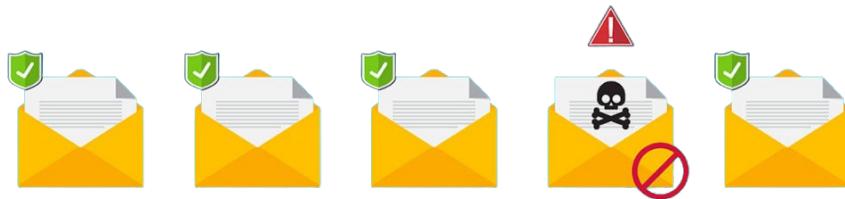


FIGURE 1.1 – Classification d'email en deux classes "spam" ou "pas spam"

Exemple 2 : Les développeurs de voitures autonomes utilisent de grandes quantités de données provenant de systèmes de reconnaissance d’images, ainsi que des méthodes d’apprentissage automatique, pour créer des systèmes capables de conduire de manière autonome.



FIGURE 1.2 – Détection des signes et panneaux routiers par segmentation d’images.

Exemple 3 : L'apprentissage automatique est utilisé pour analyser des ensembles de données de santé massifs afin d'accélérer la découverte de traitements et de remèdes, d'améliorer les résultats pour les patients et d'automatiser les processus de routine afin d'éviter les erreurs humaines.

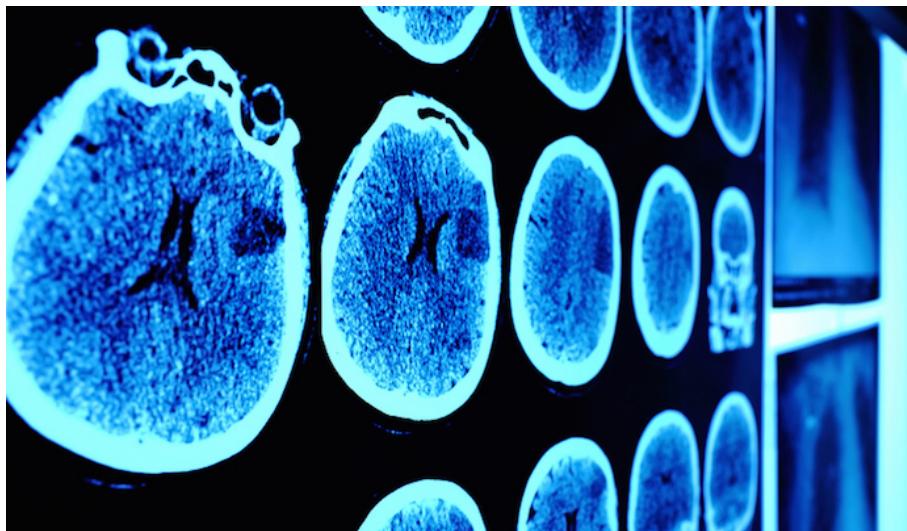


FIGURE 1.3 – Détection du cancer du sein à l'aide de l'apprentissage en profondeur

Exemple 4 : Supposons que l'on dispose d'une base de données regroupant les caractéristiques d'appartements dans une ville : superficie, quartier, étage, année de construction, nombre de chambres et le prix. Comment prédire le prix d'un appartement à partir des autres caractéristiques pour un appartement qui n'appartiendra pas à cette base ?



FIGURE 1.4 – Prédiction du prix d'un bien immobilier

1.2 Catégories d'apprentissage automatique

Quatre grandes approches relèvent de l'apprentissage automatique : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage par renforcement et l'apprentissage en profondeur. Selon la nature du problème traité, il existe différentes approches qui varient selon le type et le volume des données. Dans cette section, nous discuterons les catégories d'apprentissage automatique. [4]

Tout d'abord, concentrons nous sur la notion de données, puisqu'elle est le cœur de cette nouvelle technologie qui est l'apprentissage automatique. La machine se base sur les données aussi appelées échantillons, observations ou exemples, pour aboutir à des résultats intéressants. Concrètement, nous disposons d'un ensemble de données ou aussi appelé dataset de taille qu'on notera M , les observations sont alors (x_m) pour $1 \leq m \leq M$.

On trouve généralement deux types d'ensemble de données :

- **Données étiquetées** : chaque observation (x_m) admet une étiquette (label) notée (y_m) .
- **Données non-étiquetées** : les données n'admettent aucune étiquette.

1.2.1 Apprentissage en profondeur

L'apprentissage en profondeur est une méthode spécifique d'apprentissage automatique qui intègre des réseaux neuronaux en couches successives afin d'apprendre des données de manière itérative. L'apprentissage en profondeur est particulièrement utile lorsque vous tentez de détecter des tendances à partir de données non structurées. Les réseaux neuronaux et l'apprentissage en profondeur sont souvent utilisés dans les applications de reconnaissance d'image, de communication orale et de vision numérique.

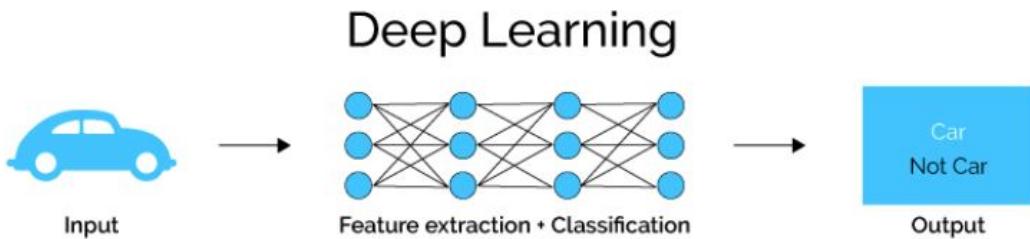


FIGURE 1.5 – Classification d'une image par un réseau de neurones

Parmi les derniers résultats de l'apprentissage en profondeur :

- En 2019, le modèle OpenAI Five [7] composée de cinq réseaux de neurones bat l'équipe championne du monde OG sur Dota 2 (jeu vidéo de type arène de bataille multijoueur) devenant la première IA à battre les champions du monde dans un jeu d'e-sport.



FIGURE 1.6 – L'équipe OG et l'équipe de développement OpenAI après le match historique.

- Révélé par OpenAI le 5 janvier 2021, DALL-E [8] est un modèle d'apprentissage profond capable de créer des images à partir de descriptions textuelles. Il utilise une version à 12 milliards de paramètres du modèle de langage GPT-3 pour interpréter les entrées en langage naturel. [6]



FIGURE 1.7 – Exemples d'images générées par le modèle DALL-E

1.2.2 Apprentissage par renforcement

Le scénario général de l'apprentissage par renforcement est illustré par la figure 1.8. Contrairement au scénario de l'apprentissage supervisé, ici, l'apprenant ne reçoit pas passivement un ensemble de données étiquetées. Au lieu de cela, il recueille des informations par le biais d'un cours d'actions en interagissant avec l'environnement. En réponse à une action, l'apprenant ou l'agent reçoit deux types d'informations : son état actuel dans l'environnement et une récompense à valeur réelle, spécifique à la tâche et à son objectif correspondant. L'objectif de l'agent est de maximiser sa récompense et donc de déterminer le meilleur plan d'action, ou la meilleure politique, pour atteindre cet objectif. Cependant, les informations qu'il reçoit de l'environnement ne sont que la récompense immédiate liée à l'action qu'il vient de mener. Un aspect important de l'apprentissage par renforcement consiste à envisager des récompenses ou des pénalités différées. L'agent est confronté à un dilemme entre l'exploration d'états et d'actions inconnues pour obtenir plus d'informations sur l'environnement et les récompenses, et l'exploitation des informations déjà recueillies pour optimiser sa récompense. C'est ce que l'on appelle le compromis entre l'exploration et l'exploitation, qui est lié à l'apprentissage par renforcement.

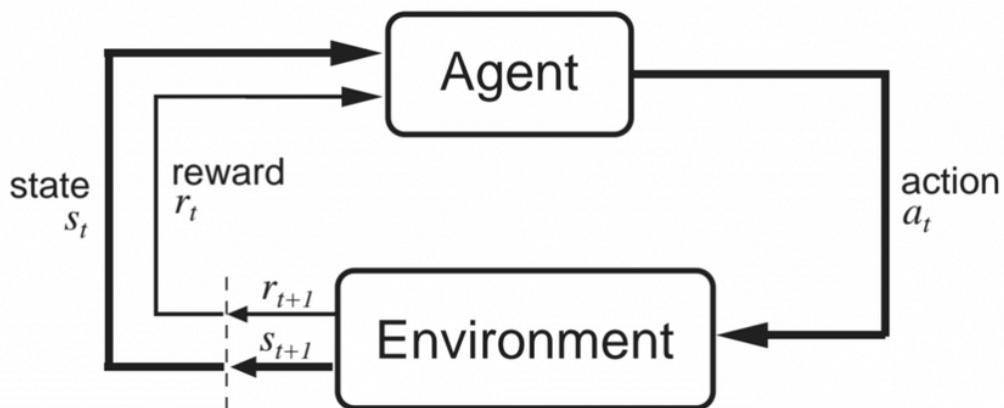


FIGURE 1.8 – Apprentissage par renforcement

Le programme AlphaGo de DeepMind [9] est un bon exemple d'apprentissage par renforcement : il a fait la une des journaux en mars 2016 lorsqu'il a battu le champion du monde Lee Sedol au jeu de Go. Il a appris sa politique gagnante en analysant des millions de parties, puis en jouant de nombreuses parties contre lui-même.

1.2.3 Apprentissage non supervisé

L'apprentissage non supervisé traite des données non étiquetées. L'objectif est d'identifier automatiquement des caractéristiques communes aux observations. Par exemple, les applications de réseaux sociaux, telles que Twitter, Instagram et Snapchat, exploitent toutes de très grandes quantités de données non étiquetées. Pour comprendre le sens de ces données, il est nécessaire d'utiliser des algorithmes qui classifient les données en fonction des tendances ou des groupes *clusters* qu'ils décèlent. L'apprentissage non supervisé mène un processus itératif, analysant les données sans intervention humaine. Il est utilisé avec la technologie de détection de spam envoyé par e-mail. Ces modèles incluent des tâches telles que le regroupement (clustering) et la réduction de la dimensionnalité (dimensionality reduction). Les algorithmes de clustering identifient des groupes de données distincts, tandis que les algorithmes de réduction de dimensionnalité recherchent des représentations plus succinctes des données. Nous verrons des exemples de ces deux types d'apprentissage non supervisé dans le chapitre suivant.

1.2.4 Apprentissage supervisé

L'apprentissage supervisé s'intéresse aux données étiquetées. L'objectif est de prédire l'étiquette inconnue associée à une nouvelle observation x , à partir de la connaissance fournie par les M observations étiquetées du jeu de données (x_n, y_n) . Les méthodes les plus utilisées sont des algorithmes qui cherchent une fonction de prédiction f_ω qui s'écrit en fonction des X variables d'entrées, avec ω un vecteur qui représente un ensemble de paramètres.

Cet apprentissage supervisé se fait en deux phases, une phase d'apprentissage et une d'entraînement où ω est adapté de manière à optimiser les performances de prédiction sur la base des observations étiquetées, appelée ici base d'apprentissage ou d'entraînement. L'apprentissage est basé sur la mesure de l'écart entre les «vraies» étiquettes y_m et les étiquettes prédites $f_\omega(x_m)$. Le problème à résoudre est de mesurer l'écart entre la vraie étiquette et l'étiquette prédite et le choix du critère d'optimisation de ω .

On distingue deux grandes familles de problèmes dans l'apprentissage supervisé : classification supervisée (y désigne une classe) et régression (y est une valeur scalaire ou vectorielle).

1.2.4.1 Classification

La majorité des problèmes qu'on rencontre en apprentissage automatique sont des problèmes de classification. Lorsque les étiquettes y_m prennent leurs valeurs dans un ensemble fini dont les éléments correspondent à des catégories (ou classes) à identifier, on parle de classification supervisée. La fonction f_ω associe alors une nouvelle observation x à une des classes. Elle définit donc une partition de l'ensemble des observations : chaque élément de cet ensemble sera affecté à une des classes. Les frontières entre classes sont appelées frontières de séparation. La détection des emails spam dans l'exemple 1 est un problème de classification. Imaginons que nous ayons les données présentées dans cette figure : [2]

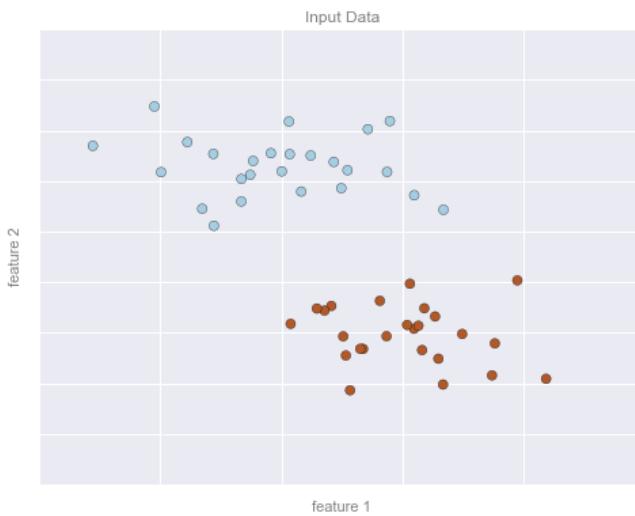


FIGURE 1.9 – Représentation des données sur un plan à deux caractéristiques (x,y)

Ici, nous avons des données bidimensionnelles : c'est-à-dire que nous avons deux caractéristiques pour chaque point, représentées par les positions (x, y) des points sur le plan. De plus, nous avons l'une des deux étiquettes de classe pour chaque point, ici représentées par les couleurs des points. À partir de ces caractéristiques et étiquettes, nous aimeraisons créer un modèle qui nous permettra de décider si un nouveau point doit être étiqueté "bleu" ou "rouge".

Nous ferons l'hypothèse que les deux groupes peuvent être séparés en traçant une ligne droite passant par le plan qui les sépare, de sorte que les points de chaque côté de la ligne appartiennent au même groupe. Ici, le modèle est une version quantitative de l'énoncé "une ligne droite sépare les classes", tandis que les paramètres du modèle sont les nombres particuliers décrivant l'emplacement et l'orientation de cette ligne pour nos données.

Les valeurs optimales pour ces paramètres de modèle sont apprises à partir des données, ce qui est souvent appelé apprentissage du modèle. La figure suivante montre une représentation visuelle de ce à quoi ressemble le modèle entraîné pour ces données :

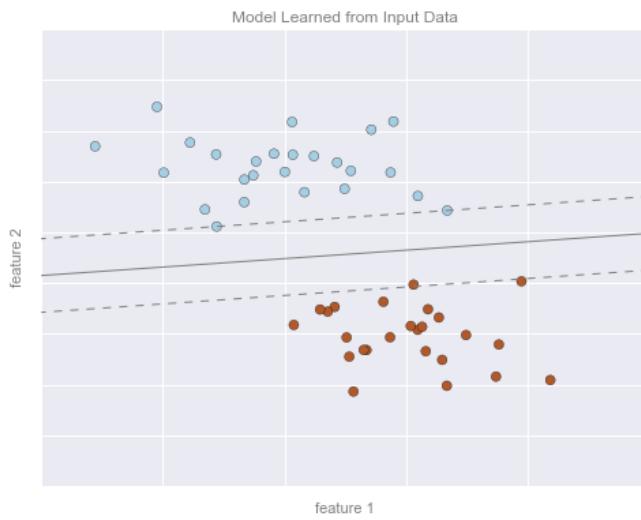


FIGURE 1.10 – Représentation visuelle de la classification des points grâce à une droite

Maintenant que ce modèle a été formé, il peut être généralisé à de nouvelles données non étiquetées. En d'autres termes, nous pouvons prendre un nouvel ensemble de données, y tracer cette ligne de modèle et attribuer des étiquettes aux nouveaux points en fonction de ce modèle. Cette étape est généralement appelée prédition.

La classification peut être utiliser pour la tâche de détection automatisée des spams pour les e-mails ; dans ce cas, nous pourrions utiliser les fonctionnalités et étiquettes suivantes :

- **Caractéristique 1, caractéristique 2, etc.** → nombre normalisé de mots ou de phrases importantes ("Iphone gratuit", "prince nigérian", etc.)
- **L'étiquette** → "spam" ou "pas spam"

Pour l'ensemble d'entraînement, ces étiquettes peuvent être déterminées par l'inspection individuelle d'un petit échantillon représentatif d'e-mails ; pour les e-mails restants, l'étiquette serait déterminée à l'aide du modèle.

Il existe plusieurs types d'algorithme de classification, on en cite : la classification naïve bayésienne, le classificateur de forêt aléatoire et la régression logistique que nous détaillerons dans le prochain chapitre.

1.2.4.2 Régression

Lorsque les étiquettes y_n prennent des valeurs scalaires ou vectorielles, on parle de problème de régression. L'exemple 4 est un problème de régression, car on cherche à prédire le prix d'appartements qui est une grandeur scalaire. Dans l'apprentissage automatique, le but de la régression est d'estimer une valeur (numérique) de sortie à partir des valeurs d'un ensemble de caractéristiques en entrée. Par exemple, estimer le prix d'une maison en se basant sur sa surface, le nombre d'étages, son emplacement, etc. Donc, le problème revient à estimer une fonction de calcul en se basant sur les données d'entraînement.

$$\hat{y}(x) = f(x_1, x_2, \dots, x_m) \quad (1.1)$$

où \hat{y} est la sortie estimée (résultat) et les x_m sont les caractéristiques d'entrée.

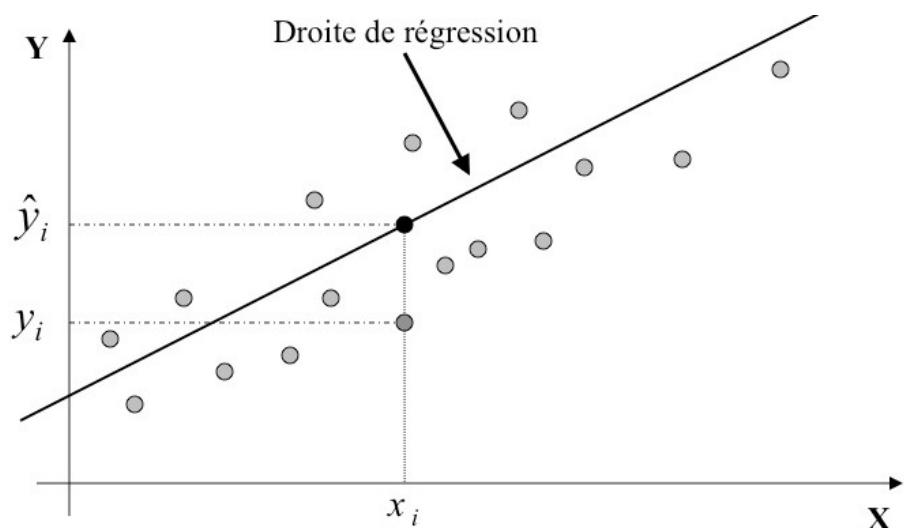


FIGURE 1.11 – Ensemble de points de données représentés sur un plan avec la droite de régression qui représente l'ensemble des points prédicts

Par exemple, cela est similaire à la tâche de calculer la distance aux galaxies observées à travers un télescope. Dans ce cas, nous pourrions utiliser les caractéristiques et étiquettes suivantes :

- **Caractéristique 1, caractéristique 2, etc.** → luminosité de chaque galaxie à l'une des nombreuses longueurs d'onde ou couleurs
- **L'étiquette** → distance ou redshift de la galaxie

Les distances pour un petit nombre de ces galaxies pourraient être déterminées par un ensemble indépendant d'observations (généralement plus coûteuses). Les distances aux galaxies restantes pourraient alors être estimées à l'aide d'un modèle de régression approprié, sans qu'il soit nécessaire d'employer l'observation la plus coûteuse sur l'ensemble de l'ensemble. Dans les cercles d'astronomie, c'est ce qu'on appelle le problème du "décalage photométrique vers le rouge".

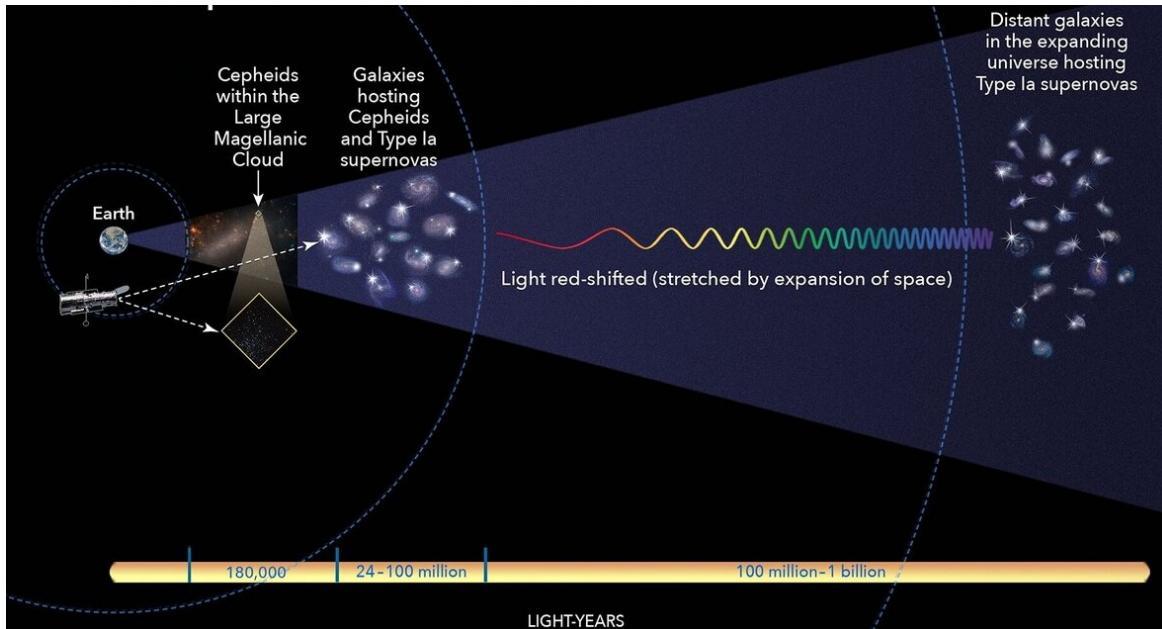


FIGURE 1.12 – Le phénomène du décalage vers le rouge (redshift) observé à travers le télescope Hubble.

Il existe plusieurs algorithmes de régression, on en cite :

- Régression linéaire
- Régression polynomial
- Régression quantile
- Régression logistique

Dans le prochain chapitre, nous nous intéresserons principalement aux algorithmes d'apprentissage automatique supervisé et non supervisé.

CHAPITRE 2

Algorithmes d'apprentissage automatique

2.1 Algorithmes d'apprentissage non supervisé

2.1.1 Analyse en composantes principales (ACP)

Explorons ce qui est peut-être l'un des algorithmes non supervisés les plus largement utilisés, l'analyse en composantes principales (ACP). L'ACP est fondamentalement un algorithme de réduction de dimensionnalité, mais il peut également être utile comme outil de visualisation, de filtrage du bruit, d'extraction et d'ingénierie de caractéristiques, et bien plus encore. L'analyse en composantes principales est une méthode non supervisée rapide et flexible pour la réduction de la dimensionnalité des données.

L'utilisation de l'ACP pour la réduction de dimensionnalité implique la mise à zéro d'une ou plusieurs des plus petites composantes principales, ce qui entraîne une projection de dimension inférieure des données qui préserve la variance maximale des données. Pourquoi devons-nous réduire les dimensions dans l'ensemble de données ? N'est-ce pas une perte d'informations ? Oui, nous perdons des informations lorsque nous supprimons certaines des dimensions de nos données. Cependant, dans certains cas, nos données peuvent avoir de nombreuses fonctionnalités ou variables pour appliquer une technique d'apprentissage automatique pour effectuer une classification ou un regroupement. Par exemple pour AmazonVideo, Youtube ou Netflix, ils peuvent être en millions de dimensions où chaque vidéo est une variable, et multipliez-le par le nombre d'utilisateurs qu'ils ont lorsque vous devez extraire des similitudes entre les utilisateurs ou les vidéos et produire des recommandations.

Pour analyser et construire un nouvel ensemble de données de dimensions réduites à partir de l'original par l'ACP, les étapes suivantes sont généralement utilisées : [13]

1. Importer l'ensemble de données
2. Calculer la matrice de covariance des données
3. Calculer les valeurs propres et les vecteurs propres sur la matrice de covariance
4. Choisir les composants principaux
5. Construire un nouvel ensemble de données vedette à partir des composants choisis

Appliquons chaque étape une par une à l'ensemble de données Iris [12].

1. Importation de l'ensemble de données

L'ensemble de données Iris se compose de 4 caractéristiques ou variables ; ou 4 dimensions en algèbre linéaire, et 1 vecteur cible qui montre le type de fleur dépendant des 4 caractéristiques. Donc, le problème est en 4 dimensions. Nous essaierons de réduire l'ensemble de données à 2 dimensions pour illustrer l'ACP.

	Petal length	Petal Width	Sepal Length	Sepal Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

FIGURE 2.1 – Les cinq premières observations de la base de données Iris de taille (150x4)

2. Calcul de la matrice de covariance des données

La covariance est la variance de 2 caractéristiques ; en d'autres termes, comment 2 caractéristiques varient l'une de l'autre. Il s'agit d'une information très utile lorsqu'on doit extraire de nouveaux modèles ou fonctionnalités à partir de fonctionnalités existantes.

$$var(x) = \frac{1}{N} \sum_{i=0}^n (x_i - \bar{x})^2 \quad (2.1)$$

$$cov(x, y) = \frac{1}{N} \sum_{i=0}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (2.2)$$

La matrice de covariance est une matrice carrée contenant l'ensemble des covariances entre variables prises 2 à 2.

$$M = \begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) \\ \text{cov}(y, x) & \text{cov}(y, y) \end{bmatrix} \quad (2.3)$$

En deuxième étape, nous devons calculer la matrice de covariance de notre ensemble de données. Puisqu'il y a 4 caractéristiques dans les données, nous avons 4 variances à calculer et 6 covariances. Ci-dessous, nous avons la matrice de covariance des résultats. Les valeurs diagonales sont des variances de chaque caractéristique mais elles ne nous intéressent pas car nous essayons de trouver de nouvelles caractéristiques parmi les fonctionnalités existantes. Par conséquent, d'autres entrées, à l'exception de celles en diagonale, sont essentielles pour l'ACP.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	0.685694	-0.039268	1.273682	0.516904
sepal width (cm)	-0.039268	0.188004	-0.321713	-0.117981
petal length (cm)	1.273682	-0.321713	3.113179	1.296387
petal width (cm)	0.516904	-0.117981	1.296387	0.582414

FIGURE 2.2 – Matrice de covariance de l'ensemble de données Iris

3. Calcul des valeurs propres et des vecteurs propres

Puisque nous recherchons de nouvelles fonctionnalités pour réduire la dimensionnalité de nos données, les vecteurs propres de la matrice de covariance des données sont calculés pour trouver des modèles (vecteurs propres) avec leur signification (valeurs propres). Les vecteurs propres de la matrice de covariance représenteront de nouvelles fonctionnalités et nous en choisirons certaines en fonction de leur puissance ou de leur impact de valeur propre. Faisons cela sur notre exemple d'iris.

[4.22484077, 0.24224357, 0.07852391, 0.02368303]

FIGURE 2.3 – Valeurs propres de la matrice de covariance de l'ensemble de données de l'iris

```
[ 0.36158968, -0.65653988, -0.58099728,  0.31725455]
[-0.08226889, -0.72971237,  0.59641809, -0.32409435]
[ 0.85657211,  0.1757674 ,  0.07252408, -0.47971899]
[ 0.35884393,  0.07470647,  0.54906091,  0.75112056]
```

FIGURE 2.4 – Vecteurs propres de la matrice de covariance de l’ensemble de données de l’iris

4. Choix des composants principaux

Comme nous l’avons vu précédemment, les valeurs propres représentent l’impact ou la puissance d’un vecteur, nous devons donc choisir les vecteurs propres dont la valeur propre est la plus élevée. Dans ce cas, puisque nous voulons réduire la dimensionnalité des données de l’iris à 2, nous choisirons les premiers vecteurs propres car leurs valeurs propres sont les 2 les plus élevées dans les résultats. Les vecteurs propres de valeur la plus élevée sélectionnés seront nos principaux composants pour construire un nouvel ensemble de données en vedette et réduit. Et, nous appellerons cette matrice comme nouveau vecteur de caractéristiques.

```
[ 0.36158968, -0.65653988]
[-0.08226889, -0.72971237]
[ 0.85657211,  0.1757674 ]
[ 0.35884393,  0.07470647]
```

FIGURE 2.5 – Selection des 2 vecteurs propres dont les valeurs propres sont les plus élevées

En supprimant certains composants ou valeurs propres/vecteurs propres, nous perdrions certaines informations. Cependant, puisque nous choisissons les composants ayant les valeurs ou l’importance les plus élevées, cette perte sera raisonnable. En laissant tomber, nous aurons des données dans moins de dimension pour travailler.

5. Construction du nouvel ensemble de données réduit

Maintenant, nous sommes prêts à construire de nouvelles données réduites à partir des composants principaux sélectionnés de l’étape précédente. Pour créer un nouvel ensemble de données, nous devons multiplier la transposition de la matrice d’origine à gauche de la transposition du nouveau vecteur de caractéristiques (composantes principales sélectionnées).

$$\text{nouveauEnsembleDonnees}^t = \text{vecteurCaracteristique}^t \cdot \text{ensembleDonnees}^t$$

La raison pour laquelle nous avons multiplié la transposition de l'ensemble de données d'origine et des composants principaux est d'obtenir de nouvelles données en termes de vecteurs propres que nous avons choisis.

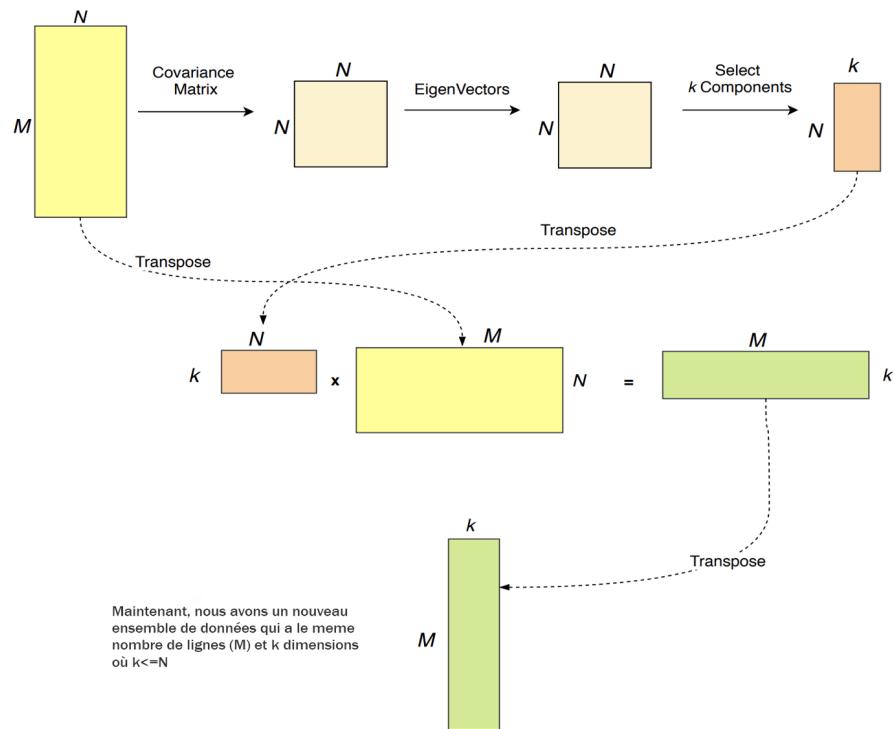


FIGURE 2.6 – Étapes visuelles des étapes appliquées

Au début, notre jeu de données avait 4 dimensions et il était impossible de tracer, mais avec le nouvel ensemble de données, les données 2D sont facile à tracer.

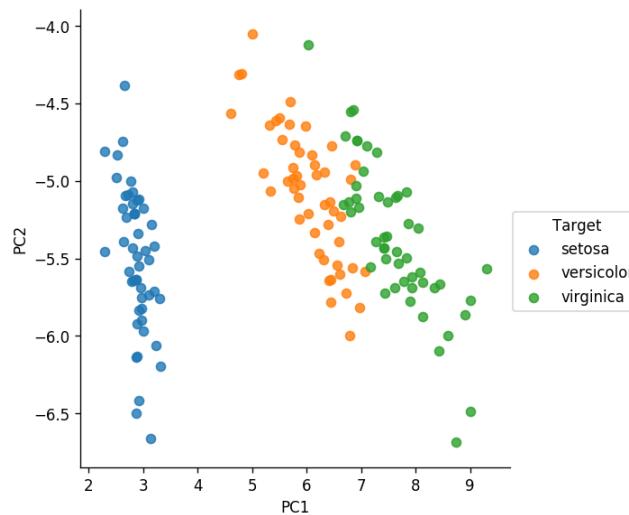


FIGURE 2.7 – Tracé de données réduites en 2D

À partir du graphique ci-dessus, il sera maintenant facile de classer ou de regrouper les échantillons sur 2 composants principaux. Bien que nous ayons perdu certaines informations en supprimant d'autres composants mineurs, nous avons maintenant beaucoup plus de données interprétables entre nos mains.

2.1.2 K moyennes clustering

Dans la section précédente, nous avons exploré une catégorie de modèles d'apprentissage automatique non supervisé : la réduction de la dimensionnalité. Ici, nous allons passer à une autre classe de modèles d'apprentissage automatique non supervisés : les algorithmes de clustering. Les algorithmes de clustering cherchent à apprendre, à partir des propriétés des données, une division optimale ou un étiquetage discret de groupes de points. [5]

Le clustering K-Moyennes (K-Means) est une approche simple et élégante pour partitionner un ensemble de données en K clusters distincts et non superposés. Pour effectuer des K-means clustering, il faut d'abord préciser le nombre de clusters K souhaité ; puis l'algorithme des K-moyennes va affecter chaque observation à exactement l'un des K groupes :

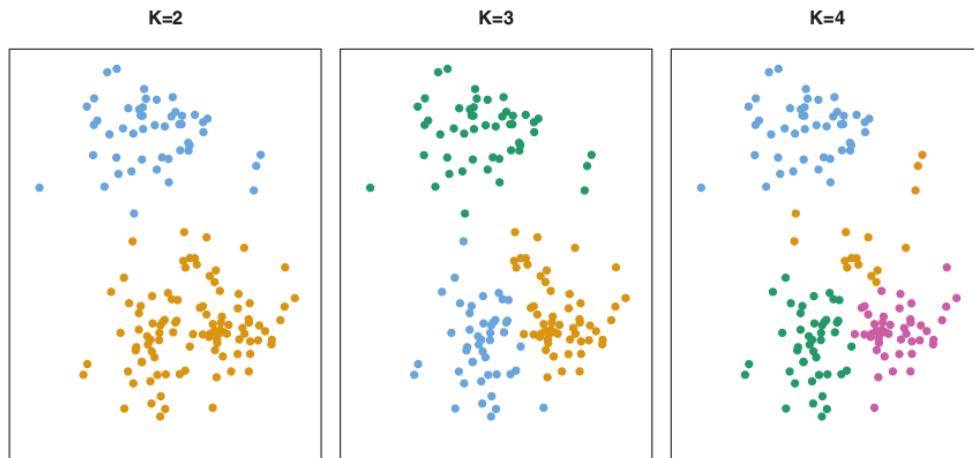


FIGURE 2.8 – Groupement des points de données en K groupes

L'algorithme K-means regroupe les données en essayant de séparer les échantillons en N groupes de variance égale, en minimisant un critère connu sous le nom d'inertie ou somme des carrés intra-cluster. L'algorithme K-means vise à choisir le centre de gravité qui minimise l'inertie, ou critère de somme des carrés intra-cluster : [15]

$$\sum_{i=0}^n \min(\|x_i - \mu_j\|^2)$$

Algorithme des K moyennes clustering

Algorithm 1 Algorithme K-Moyennes Clustering

1. Déterminer la valeur K, où K représente le nombre de clusters
 2. Sélectionner au hasard K centroïdes distincts
 3. Mesurer la distance (distance euclidienne) entre chaque point et le centroïde
 4. Attribuer chaque point au cluster le plus proche
 5. Calculer la moyenne de chaque cluster en tant que nouveau centroïde
 6. Répéter les étapes 3 à 5 avec le nouveau centre du cluster
 7. Calculer la variance de chaque cluster
 8. Répétez les étapes 2 à 7 jusqu'à obtenir la somme de variance la plus faible
-

Considérons les données fournies ci-dessous et appliquons l'algorithme :



FIGURE 2.9 – Données unidimensionnelles

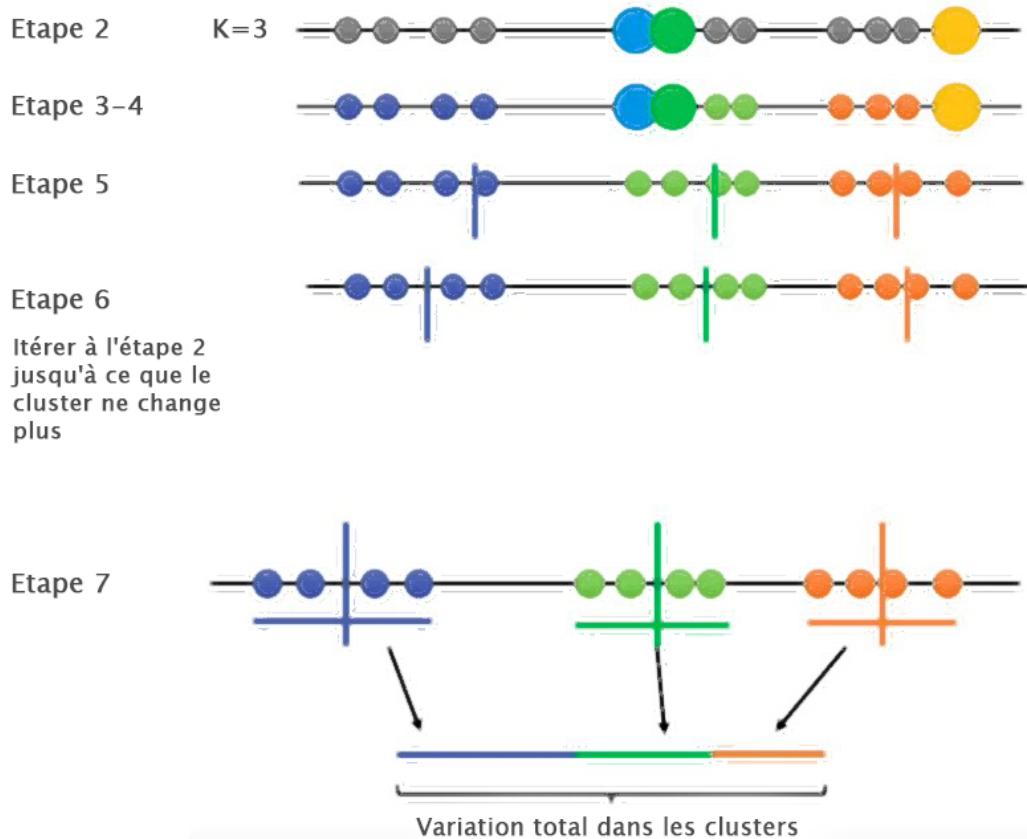


FIGURE 2.10 – Représentation visuelle des étapes de l'algorithme K-Moyennes

2.2 Algorithmes d'apprentissage supervisé

2.2.1 Classification naïve de Bayes

Les modèles Naive Bayes sont un groupe d'algorithmes de classification extrêmement rapides et simples qui conviennent souvent aux ensembles de données de très grande dimension. Parce qu'ils sont si rapides et ont si peu de paramètres réglables, ils finissent par être très utiles comme base rapide pour un problème de classification.

Les classificateurs naïfs de Bayes reposent sur le théorème de Bayes, qui est une équation décrivant la relation des probabilités conditionnelles de quantités statistiques. Le théorème de Bayes nous indique comment exprimer cela en termes de quantités que nous pouvons calculer plus directement :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.4)$$

En utilisant le théorème de Bayes, nous pouvons trouver la probabilité que A se produise, étant donné que B s'est produit. Ici, B est la preuve et A est l'hypothèse. L'hypothèse faite ici est que les prédicteurs/caractéristiques sont indépendants. La présence d'une caractéristique particulière n'affecte pas l'autre. C'est là qu'intervient le "naïf" dans "naïf Bayes" : si nous faisons des hypothèses très naïves sur le modèle génératif pour chaque étiquette, nous pouvons trouver une approximation grossière du modèle génératif pour chaque classe, puis procéder à la classification bayésienne. Prenons un exemple pour avoir une meilleure intuition. Considérez le problème de jouer au golf. Le jeu de données est représenté comme ci-dessous : [14]

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
:	:	:	:	:	:
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

FIGURE 2.11 – Le jeu de données décrivant le problème de jouer au golf selon la météo

Nous classons si la journée est appropriée pour pratiquer du golf, en prenant en compte la météo du jour. Les colonnes représentent ces caractéristiques et les lignes représentent les entrées individuelles. Si nous prenons la première ligne de l'ensemble de données, nous pouvons observer que ce n'est pas une bonne journée pour jouer au golf si la journée est pluvieuse, la température est chaude, l'humidité est élevée et il n'y a pas de vent. Nous faisons ici deux hypothèses, l'une comme indiqué ci-dessus nous considérons que ces prédicteurs sont indépendants. Une autre hypothèse faite ici est que tous les prédicteurs ont un effet égal sur le résultat. Autrement dit, le jour venteux n'a pas plus d'importance dans la décision de jouer au golf ou non. Selon cet exemple, le théorème de Bayes peut être réécrit comme suit :

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (2.5)$$

La variable y est la variable de classe (jouer au golf), qui représente s'il convient de jouer au golf ou non compte tenu des conditions. La variable X représente les paramètres, X est donné par : $X = (x_1, x_2, x_3, \dots, x_n)$. Ici x_1, x_2, \dots, x_n représentent les caractéristiques, c'est-à-dire qu'elles peuvent être mappées aux perspectives, à la température, à l'humidité et au vent. En remplaçant X et en développant à l'aide de la règle de la chaîne, nous obtenons :

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)} \quad (2.6)$$

Maintenant, on peut obtenir les valeurs de chacun en examinant l'ensemble de données et les substituer dans l'équation. Pour toutes les entrées du jeu de données, le dénominateur ne change pas, il reste statique. Par conséquent, le dénominateur peut être supprimé et une proportionnalité peut être introduite :

$$P(y|x_1, \dots, x_n) \approx P(y) \prod_{i=1}^n P(x_i|y) \quad (2.7)$$

Dans notre cas, la variable de classe (y) n'a que deux résultats, oui ou non. Il pourrait y avoir des cas où la classification pourrait être multivariée. Par conséquent, nous devons trouver la classe y avec une probabilité maximale :

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y) \quad (2.8)$$

En utilisant la fonction ci-dessus, nous pouvons obtenir la classe, compte tenu des prédicteurs.

Les types de classifieurs naïf de Bayes qui existent sont :

- **Bayes Naïf Multinomial** : Principalement utilisé pour les problèmes de classification de documents, c'est-à-dire si un document appartient à la catégorie des sports, de la politique, de la technologie, etc. Les caractéristiques/prédicteurs utilisés par le classifieur sont la fréquence des mots présents dans le document.
- **Bayes Naïf Bernoulli** : Similaire aux bayes naïfs multinomiaux mais les prédicteurs sont des variables booléennes. Les paramètres que nous utilisons pour prédire la variable de classe ne prennent que des valeurs oui ou non, par exemple si un mot apparaît dans le texte ou non.
- **Bayes Naïf Gaussien** : Lorsque les prédicteurs prennent une valeur continue et ne sont pas discrets, nous supposons que ces valeurs sont échantillonnées à partir d'une distribution gaussienne.

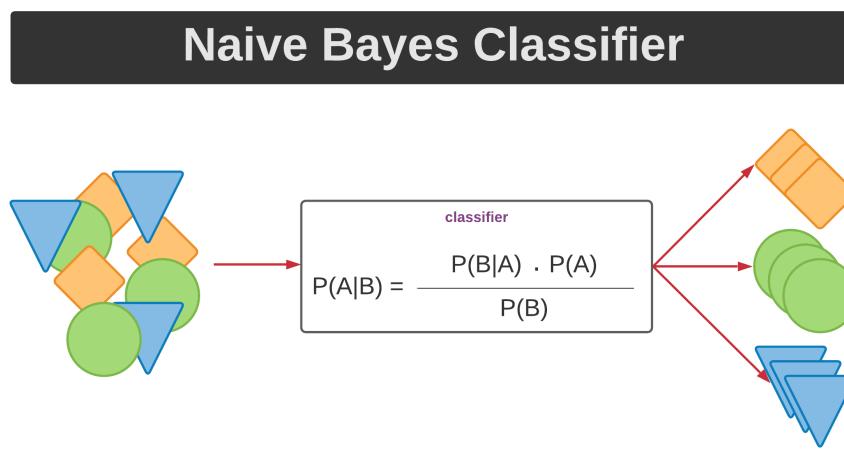


FIGURE 2.12 – Classification d'objets grâce à un classifieur bayésien naïf

En conclusion, les algorithmes Naïve Bayes sont principalement utilisés dans l'analyse des sentiments, le filtrage du spam, les systèmes de recommandation, etc. Ils sont rapides et faciles à mettre en oeuvre, mais leur plus grand inconvénient est l'exigence d'indépendance des prédicteurs. Dans la plupart des cas réels, les prédicteurs sont dépendants, ce qui nuit aux performances du classifieur.

2.2.2 Arbres de décision et forêts aléatoires

Précédemment, nous avons détaillé un classificateur génératif simple. Ici, nous allons examiner la motivation d'un autre algorithme puissant, un algorithme non paramétrique appelé forêts aléatoires. Les forêts aléatoires sont un exemple de méthode d'ensemble, ce qui signifie qu'elle repose sur l'agrégation des résultats d'un ensemble d'estimateurs plus simples. Le résultat quelque peu surprenant avec de telles méthodes d'ensemble est que la somme peut être supérieure aux parties : c'est-à-dire qu'un vote majoritaire parmi un certain nombre d'estimateurs peut finir par être meilleur que n'importe lequel des estimateurs individuels faisant un vote.

Les forêts aléatoires (random forest) sont un exemple d'apprenant d'ensemble construit sur des arbres de décision. Commencerons par discuter des arbres de décision eux-mêmes. Les arbres de décision sont des moyens extrêmement intuitifs de classer ou d'étiqueter des objets : il suffit de poser une série de questions conçues pour cibler la classification. Par exemple, si on décide de construire un arbre de décision pour classer une fleur qu'on a croisé lors d'une promenade, on pourrait construire l'arbre de décision ci-dessous :

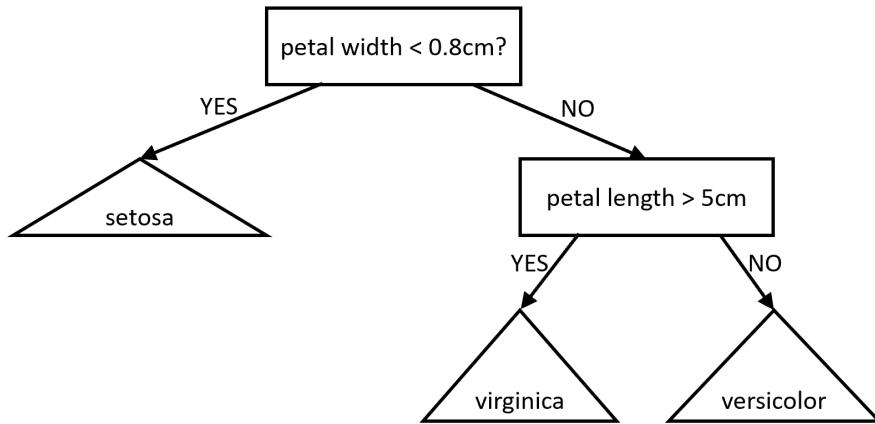


FIGURE 2.13 – Arbre de décision pour identifier le type d'une fleur

Le découpage binaire rend cela extrêmement efficace : dans un arbre bien construit, chaque question réduira le nombre d'options d'environ la moitié, réduisant très rapidement les options même parmi un grand nombre de classes. Dans les implémentations d'arbres de décision d'apprentissage automatique, les questions prennent généralement la forme de divisions alignées sur les axes dans les données : c'est-à-dire que chaque nœud de l'arbre divise les données en deux groupes en utilisant une valeur limite dans l'une des caractéristiques.

L'algorithme des forêts aléatoires comporte généralement les étapes suivantes :

Algorithm 2 Algorithme Forêts Aléatoires

Début

1. Sélectionner au hasard les fonctionnalités K parmi les fonctionnalités m totales où $k < m$
2. Parmi les K caractéristiques, calculer le nœud d en utilisant le meilleur point de partage
3. Diviser le nœud en nœuds filles en utilisant la meilleure division
4. Répétez les étapes 1 à 3 jusqu'à ce que le nombre 1 de nœuds soit atteint
5. Construire une forêt en répétant les étapes 1 à 4 pour un nombre n de fois pour créer un nombre n d'arbres
6. Chaque arbre de décision générera un résultat. Le résultat final est considéré sur la base du vote majoritaire pour la classification ou de la moyenne pour la régression

Fin

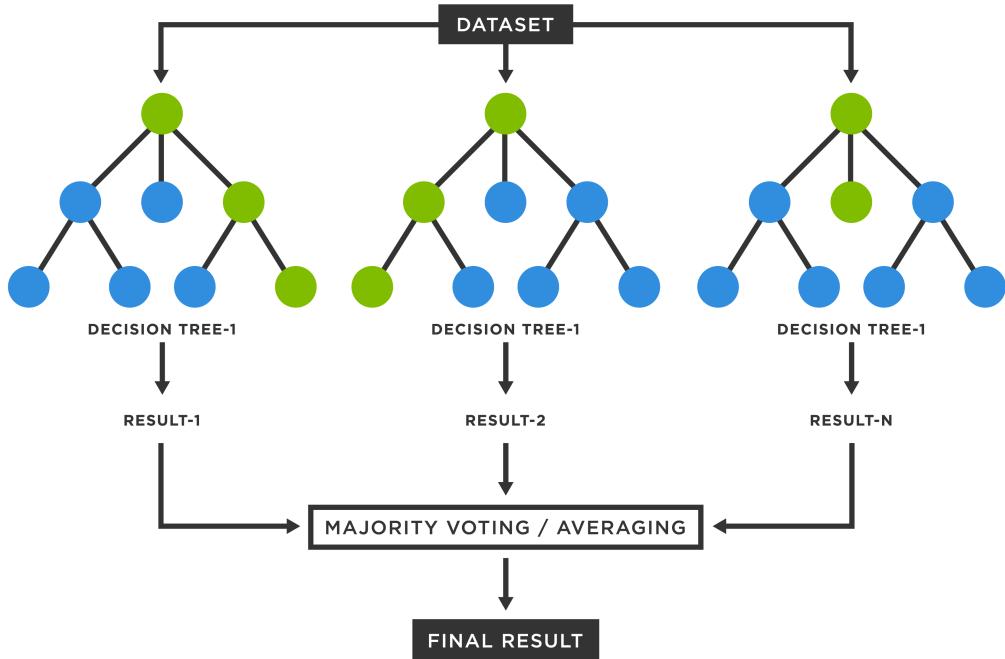


FIGURE 2.14 – Visualisation de l'algorithme forêts aléatoires

Les forêts aléatoires sont une méthode puissante avec plusieurs avantages dont la rapidité de la formation et la prédiction en raison de la simplicité des arbres de décision sous-jacents. De plus, les deux tâches peuvent être directement parallélisées, car les arbres individuels sont des entités entièrement indépendantes ainsi que le modèle non paramétrique est extrêmement flexible et peut donc bien fonctionner sur des tâches qui sont sous-ajustées par d'autres estimateurs.

2.2.3 Régression linéaire

Tout comme Bayes naïf qui est un bon point de départ pour les tâches de classification, les modèles de régression linéaire sont un bon point de départ pour les tâches de régression. De tels modèles sont populaires car ils peuvent être ajustés très rapidement et sont très interprétables. Vous connaissez probablement la forme la plus simple d'un modèle de régression linéaire (c'est-à-dire l'ajustement d'une ligne droite aux données), mais ces modèles peuvent être étendus pour modéliser un comportement de données plus complexe.

Régression linéaire simple

Nous commencerons par la régression linéaire la plus familière, un ajustement linéaire aux données. La régression linéaire simple porte bien son nom : il s'agit d'une approche linéaire très simple permettant de prédire une réponse quantitative y sur la base d'une seule variable prédictive x . Elle suppose qu'il existe approximativement une relation linéaire entre x et y . Mathématiquement, nous pouvons écrire cette relation linéaire comme suit :

$$y = ax + b \quad (2.9)$$

où a est communément appelé la pente et b est appelé l'ordonnée à l'origine.

L'objectif d'un modèle de régression linéaire est de trouver une relation entre une ou plusieurs caractéristiques (variables indépendantes) et une variable cible continue (variable dépendante). Lorsqu'il n'y a qu'une caractéristique, cela s'appelle une régression linéaire univariée et s'il y a plusieurs caractéristiques, cela s'appelle une régression linéaire multiple.

Régression linéaire multiple

Le modèle de régression linéaire multiple peut être représenté par l'équation suivante :

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n \quad (2.10)$$

où y est la valeur prédictive, β_0 est le terme de biais, β_1, \dots, β_n sont les paramètres du modèle et X_1, \dots, X_n sont les valeurs des caractéristiques.

2.2.4 Régression logistique

Les techniques de classification sont une partie essentielle des applications d'apprentissage automatique et d'exploration de données. La régression logistique est l'une des méthodes les plus courantes pour la résolution des problèmes de classification binaire, elle peut être utilisée pour divers problèmes de classification tels que la détection de spam, prédiction du diabète, si un client donné achètera un produit particulier, si l'utilisateur cliquera sur un lien publicitaire donné ou non, et de nombreux autres exemples. La régression logistique décrit et estime la relation entre une variable binaire dépendante et des variables indépendantes.

Il s'agit d'un cas particulier de régression linéaire où la variable cible est de nature catégorielle. Il utilise un journal des cotés comme variable dépendante. La régression logistique prédit la probabilité d'occurrence d'un événement binaire à l'aide d'une fonction logit. L'équation de la régression linéaire s'écrit :

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_n X_n \quad (2.11)$$

Où y est une variable dépendante de X_1, X_2, \dots et les X_n sont des variables explicatives.

On introduit la fonction sigmoid, qu'on applique à l'équation de régression linéaire ce qui nous permet d'obtenir une valeur de $0 \leq y \leq 1$:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \quad (2.12)$$

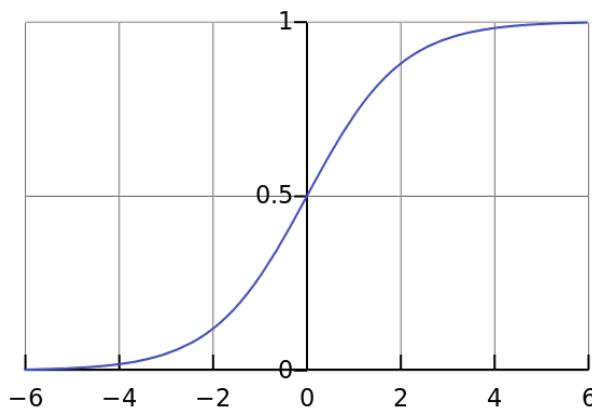


FIGURE 2.15 – Fonction sigmoid

La régression linéaire nous donne une sortie continue, mais la régression logistique fournit une sortie constante. Un exemple de sortie continue est le prix de l'immobilier et le prix des actions. Un exemple de la sortie discrète prédit si un patient a un cancer ou non, prédit si le client va se désabonner. Pour évaluer les performances d'un modèle de régression logistique, la matrice de confusion est utilisée à la place des calculs de la somme des carrés. Voici un exemple de matrice de confusion d'un algorithme qui prédit la résiliation de contrat des clients d'une assurance :

		Données prédites par l'algorithme	
		Résilié prédit	Abonné prédit
Données réelles	Résilié	Vrai positif	Faux négatif
	Abonné	Faux positif	Vrai négatif

FIGURE 2.16 – Matrice de confusion

La précision du modèle est donnée par :

$$\frac{VraiPositif + VraiNegatif}{VraiPositif + VraiNegatif + FauxPositif + FauxNegatif} \quad (2.13)$$

Il existe trois types de régression logistique :

- **Régression logistique binaire** : la variable cible n'a que deux résultats possibles, tels que Spam ou Pas de spam, Cancer ou Pas de cancer.
- **Régression logistique multinomiale** : la variable cible a trois catégories nominales ou plus, telles que la prédiction des panneaux routiers.
- **Régression logistique ordinaire** : la variable cible a trois catégories ordinaires ou plus telles que la note du restaurant ou du produit de 1 à 5.

La régression logistique est largement utilisée car elle est extrêmement efficace et ne requiert pas d'énormes quantités de ressources informatiques. Elle peut être interprétée facilement et ne nécessite pas de mise à l'échelle des caractéristiques d'entrée. Par contre, elle ne peut pas être utilisée pour résoudre des problèmes non linéaires.

2.2.5 K plus proches voisins (KNN)

Le K plus proches voisins (KNN) est un algorithme d'apprentissage automatique supervisé, très simple et polyvalent. Il est utilisé dans une variété d'applications telles que la finance, la santé, les sciences politiques, la détection d'écriture manuscrite, la reconnaissance d'images et la reconnaissance vidéo. Dans les cotes de crédit, les instituts financiers prédisent la cote de crédit des clients. Lors du décaissement du prêt, les instituts bancaires prédisent si le prêt est sûr ou risqué. L'algorithme KNN est utilisé pour des problèmes de classification et de régression. L'algorithme est basé sur l'approche de similarité des caractéristiques.

KNN est un algorithme d'apprentissage non paramétrique et paresseux (lazy learning algorithm). Non paramétrique signifie qu'il n'y a aucune hypothèse pour la distribution des données. En d'autres termes, la structure du modèle est déterminée à partir de l'ensemble de données. Cela sera très utile dans la pratique où la plupart des ensembles de données du monde réel ne suivent pas les hypothèses théoriques mathématiques. L'algorithme paresseux signifie qu'il n'a pas besoin de points de données d'apprentissage pour la génération du modèle. Toutes les données de formation utilisées dans la phase de test. Cela rend la formation plus rapide et la phase de test plus lente et plus coûteuse. Dans le pire des cas, KNN a besoin de plus de temps pour balayer tous les points de données et balayer tous les points de données nécessitera plus de mémoire pour stocker les données de formation. Puisque cet algorithme est basé sur la distance, la normalisation peut améliorer sa précision.

L'algorithme du K plus proches voisins comporte les étapes de base suivantes :

Données en entrée :

- Un ensemble de données \mathbf{D}
- Une fonction de définition de la distance d
- Un nombre entier K

Pour une nouvelle observation \mathbf{X} dont on veut prédire sa variable de sortie y on execute l'algorithme suivant :

Algorithm 3 Algorithme KNN

Début

1. Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D
2. Retenir les K observations de l'ensemble de données D les proches de X en utilisation le fonction de calcul de distance d
3. Prendre les valeurs de y des K observations retenues :
 - Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenues
 - Si on effectue une classification , calculer le mode de y retenues
4. Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X .

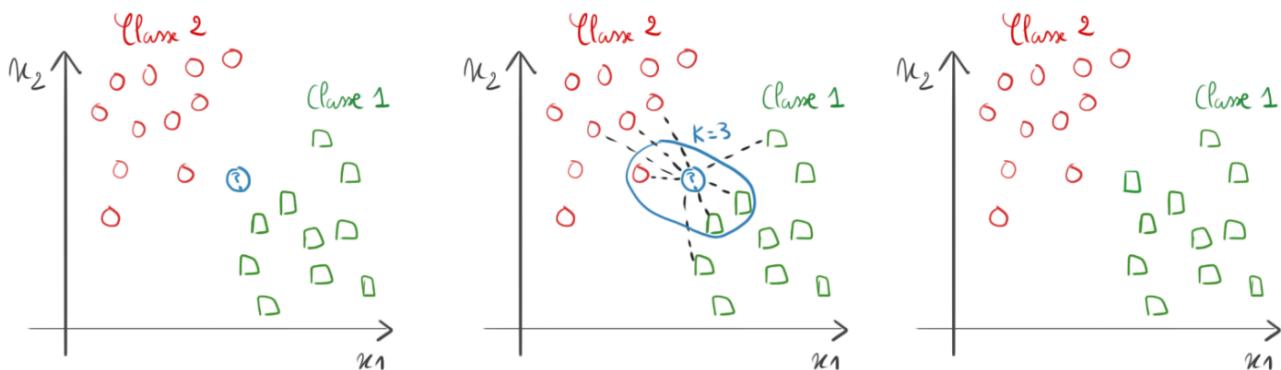
Fin

FIGURE 2.17 – Visualisation de l'algorithme KNN pour une nouvelle donnée non étiquetée à prédire pour $K=3$ voisins

Dans KNN, K est le nombre de plus proches voisins. Le nombre de voisins est le principal facteur décisif. K est généralement un nombre impair si le nombre de classes est de 2. Supposons que P est le point pour lequel l'étiquette doit être prédite. Tout d'abord, nous cherchons les K points les plus proches de P , puis on classe les points par vote majoritaire de ses K voisins. Chaque objet vote pour sa classe et la classe avec le plus de votes est prise comme prédiction. Pour trouver les points similaires les plus proches, on cherche la distance entre les points en utilisant des mesures de distance telles que la distance euclidienne, la distance de Hamming, la distance de Manhattan et la distance de Minkowski.

Généralement, pour l'algorithme KNN lorsque le nombre de caractéristique augmente, cela nécessite plus de données. Pour éviter le sur-ajustement, les données nécessaires devront croître de manière exponentielle à mesure que vous augmentez le nombre de dimensions. Ce problème de dimension est connu sous le nom de malédiction de la dimensionnalité.

Choix du K

Le choix de la valeur K à utiliser pour effectuer une prédiction avec KNN, varie en fonction du jeu de données. En règle générale, moins on utilisera de voisins plus on sera sujet au sous-apprentissage (underfitting), un modèle généraliste qui s'adapte mal à l'ensemble d'entraînement et qui est incapable de fournir des prédictions précises. Par ailleurs, plus on utilise de voisins plus on sera fiable dans notre prédiction. Toutefois, si on utilise K nombre de voisins avec $K=N$ et N étant le nombre d'observations, on risque d'avoir du sur-apprentissage (overfitting), un modèle trop spécialisé sur les données de l'ensemble d'entraînement et qui se généralise mal sur les nouvelles observations.

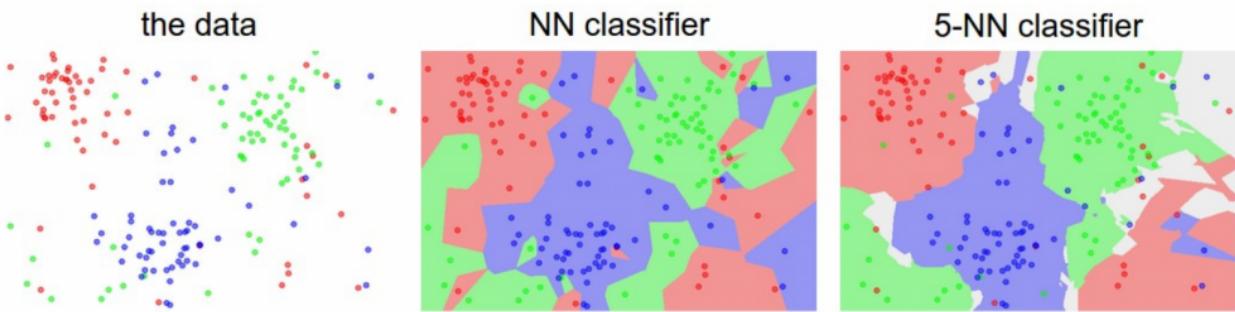


FIGURE 2.18 – Représentation graphique des points dans un plan 2D et leurs séparations en régions

Pour le 5-NN classifieur, les limites entre chaque région sont assez lisses et régulières. Quant au N-NN classifieur, on remarque que les limites sont "chaotiques" et irrégulières. Cette dernière provient du fait que l'algorithme tente de faire rentrer tous les points bleus dans les régions bleues, les rouges avec les rouges etc... c'est un cas d'overfitting. Pour cet exemple, on préférera le 5-NN classifieur sur le NN Classifieur. [16]

Distance

Il existe plusieurs fonctions de calcul de distance, notamment, la distance euclidienne, la distance de Manhattan, la distance de Minkowski, la distance de Hamming etc... On choisit la fonction de distance en fonction des types de données qu'on manipule. Ainsi pour les données quantitatives (exemple : poids, salaires, taille etc...) et du même type, la distance euclidienne est un bon candidat. Quant à la distance de Manhattan, elle est une bonne mesure à utiliser quand les données ne sont pas du même type (exemple : age, sexe, poids etc...)

- **Distance Euclidienne :**

Il s'agit de la fonction de similarité la plus intuitive puisqu'elle formalise l'idée de distance : la distance en ligne droite qui sépare deux points dans l'espace. Elle calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points :

$$D(x, y) = \sqrt{\sum_{j=1}^n (x_j - y_j)^2} \quad (2.14)$$

- **Distance Manhattan :**

La distance de Manhattan correspond à un déplacement à un angle droit sur un damier. Elle calcule la somme des valeurs absolues des différences entre les coordonnées de deux points :

$$D(x, y) = \sum_{i=1}^k |x_i - y_i| \quad (2.15)$$

La distance de Manhattan peut être intéressante pour des données qui ne sont pas du même type (c'est-à-dire des données qui n'ont pas été mis sur la même échelle).

- **Distance Cosinus :**

La distance cosinus est principalement utilisée pour calculer la similarité entre deux vecteurs. Elle est mesuré par le cosinus de l'angle entre deux vecteurs et détermine si deux vecteurs pointent dans la même direction. Lorsqu'elle est utilisée avec KNN, cette distance nous donne une nouvelle perspective sur un problème commercial et nous permet de trouver des informations cachées dans les données que nous n'avons pas vues en utilisant les deux métriques de distance ci-dessus.

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|} \quad (2.16)$$

En utilisant cette distance, nous obtenons des valeurs comprises entre 0 et 1, où 0 signifie que les vecteurs sont exactement similaires et 1 signifie qu'ils ne sont pas du tout similaires.

CHAPITRE 3

Problématique et préparation de la base de données

3.1 Introduction

Airbnb est une plateforme de marché Internet pour les locations de maisons et d'appartements à court terme. La plateforme permet à un utilisateur, par exemple, de louer son appartement pour une semaine pendant son absence, ou de louer sa chambre d'amis à des voyageurs. La société elle-même a connu une croissance rapide depuis sa création en 2008 jusqu'à une valorisation de plus de 70 milliards de dollars américains en 2022. [17]

L'un des défis auxquels les hôtes Airbnb sont confrontés est de déterminer le prix de location optimal par nuit. Les locataires potentiels peuvent filtrer les appartements selon des critères tels que le prix, le nombre de chambres, le type de chambre, etc. Étant donné que Airbnb est une place de marché, le montant qu'un hôte peut facturer par nuit est étroitement lié à la dynamique de la place de marché.

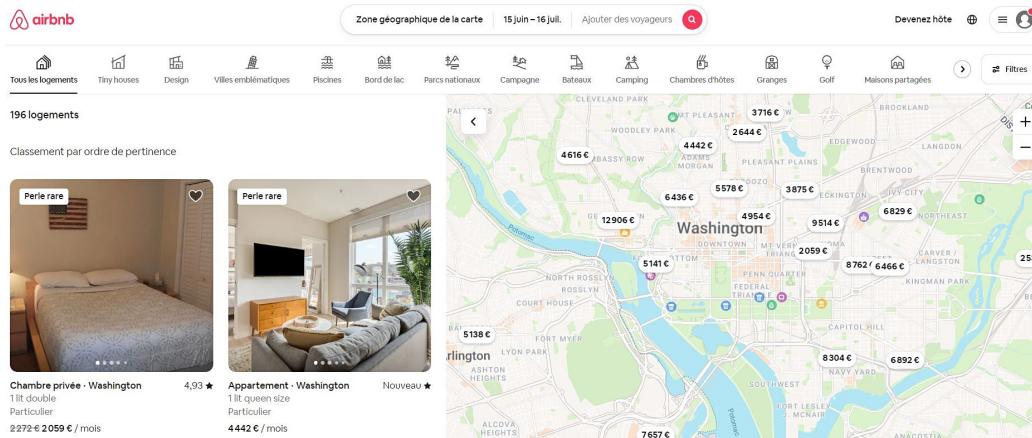


FIGURE 3.1 – L'expérience de recherche d'un logement sur la plateforme Airbnb

Imaginons que nous ayons un appartement que nous aimerais louer sur Airbnb. En tant qu'hôtes, si nous essayons de facturer au-dessus du prix du marché, les locataires choisiront des alternatives plus abordables et nous ne gagnerons pas d'argent. D'un autre côté, si nous fixons un prix de location par nuit trop bas, nous manquerons de revenus potentiels. Comment alors pouvons-nous déterminer le prix idéal ? Une stratégie que nous pourrions utiliser consiste à :

1. Trouver les annonces similaires à la nôtre
2. Faire la moyenne du prix de ceux qui sont le plus proches à la nôtre
3. Définir notre prix proposé aux locataires selon le prix moyen calculé

Faire ce travail manuellement prendrait beaucoup de temps, c'est pourquoi au lieu de le faire manuellement, nous allons créer un modèle d'apprentissage automatique pour automatiser ce processus en utilisant l'algorithme des K plus proches voisins.

3.2 Importation de la base de données

Airbnb ne publie aucune donnée sur les annonces publiées sur sa plateforme, mais un groupe distinct nommé **Inside Airbnb** [18] a extrait des données sur un échantillon d'annonces pour de nombreuses grandes villes sur la plateforme. Dans ce projet, nous allons travailler avec l'ensemble de données du 3 octobre 2015 sur les listes de Washington, D.C., la capitale des États-Unis. Commençons tout d'abord, par lire notre base de données grâce à la librairie pandas [19], en affichant sa taille et ses premières lignes.



```

● ● ●
1 import pandas as pd
2 dc_listings = pd.read_csv('dc_airbnb.csv')
3 print(dc_listings.shape)
4 dc_listings.head()

```

(3723, 92)																
	id	listing_url	scrape_id	last_scraped	name	summary	space	description	neighborhood_overview	...	review_scores_value	requires_license	license	jurisdiction_names	reviews_per_month	
0	7087327	https://www.airbnb.com/rooms/7087327	20151002231825	2015-10-03	Historic DC Condo-Walk to Capitol!	Professional pictures coming soon! Welcome to —	NaN	Professional pictures coming soon!	Welcome to —	NaN	...	NaN	f	NaN	NaN	NaN
1	975833	https://www.airbnb.com/rooms/975833	20151002231825	2015-10-03	Spacious Capitol Hill Townhouse	Beautifully renovated Capitol Hill townhouse.	NaN	Beautifully renovated Capitol Hill townhouse.	...	NaN	...	NaN	f	NaN	DISTRICT OF COLUMBIA, WASHINGTON	2.11
2	8249488	https://www.airbnb.com/rooms/8249488	20151002231825	2015-10-03	Spacious/private room for single traveler th...	This is an ideal room for a single traveler th...	NaN	This is an ideal room for a single traveler th...	Silver Spring is booming. You can walk to a n...	—	NaN	...	f	NaN	NaN	1.00

FIGURE 3.2 – Importation de la base de données

Chaque ligne de la base de données correspond à une annonce de location d'un appartement ou d'une maison sur Airbnb dans la région de Washington, DC. En termes d'apprentissage automatique, chaque ligne est une observation. Les colonnes décrivent différentes caractéristiques de chaque annonce.

3.3 Exploration de la base de données

Passons maintenant à l'exploration de notre base de données. Elle contient 3723 lignes (observations) avec 92 colonnes (caractéristiques). Pour notre cas, nous nous intéresserons seulement aux colonnes les plus importantes, car ce sont toutes les caractéristiques qu'un locataire pourrait utiliser pour évaluer quelle annonce choisir, selon ses besoins :

- **accommodates** : le nombre de personnes que la location peut accueillir
- **bedrooms** : nombre de chambres comprises dans la location
- **bathrooms** : nombre de salles de bain comprises dans la location
- **beds** : nombre de lits compris dans la location
- **minimum nights** : nombre minimum de nuits qu'un client peut rester
- **maximum nights** : nombre maximum de nuits qu'un client peut rester
- **number of reviews** : nombre d'avis laissés par les clients précédents
- **price** : prix de la nuitée

Idéalement, notre ensemble de données doit être "nettoyé". En d'autres termes, nous devons faire attention aux colonnes qui ne fonctionnent pas bien avec l'équation de distance qui est essentielle dans l'algorithme KNN. Cela inclut les colonnes contenant :

- **Valeurs non numériques** : par exemple, ville ou état
- **Valeurs non ordinaires** : par exemple latitude ou longitude
- **Valeurs manquantes**

Sélection des caractéristiques

Au lieu d'éliminer toutes les colonnes qui ne remplissent pas nos conditions, gardons ceux qui nous intéressent en créant un nouvel ensemble de donnée :



```

1 features = ["accommodates", "bedrooms", "bathrooms", "beds", "minimum_nights", "maximum_nights", "number_of_reviews", "price"]
2 dc_listings = dc_listings[features]
3 dc_listings.head()

```

	accommodates	bedrooms	bathrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
0	4	1.0	1.0	2.0	1	1125	0	\$160.00
1	6	3.0	3.0	3.0	2	30	65	\$350.00
2	1	1.0	2.0	1.0	2	1125	1	\$50.00
3	2	1.0	1.0	1.0	1	1125	0	\$95.00
4	4	1.0	1.0	1.0	7	1125	0	\$50.00

FIGURE 3.3 – Sélection des colonnes les plus importantes et affichage des 5 premières lignes

Valeurs manquantes

Vérifions si notre ensemble de données contient des valeurs manquantes :



```

1 dc_listings.isna().sum()

```

	accommodates	maximum_nights	0
bedrooms	21	number_of_reviews	0
bathrooms	27	price	0
beds	11		
minimum_nights	0	dtype:	int64

FIGURE 3.4 – La somme des données manquantes pour chaque colonnes

Étant donné que le nombre de lignes contenant une ou plusieurs valeurs manquantes pour ces 3 colonnes est faible, nous pouvons sélectionner et supprimer ces lignes sans perdre beaucoup d'informations.



```

1 dc_listings.dropna(subset=['bedrooms', 'bathrooms', 'beds'], axis=0, how='any', inplace=True)

```

FIGURE 3.5 – Suppression des lignes contenant des données manquantes pour les colonnes bedrooms, bathrooms et beds

CHAPITRE 4

Création et évaluation du modèle

4.1 Implementation du K-NN

Nous implémenterons l'algorithme des K plus proches voisins et l'utiliserons pour suggérer un prix pour une nouvelle annonce sur la plateforme. Pour cette première partie du projet, nous allons utiliser une valeur K fixe de 5, après on utilisera un algorithme K-NN depuis la librairie Scikit-Learn [20]. L'algorithme K plus proches voisins (KNN) fonctionne de manière similaire au processus en trois étapes que nous avons décrit précédemment : [10]

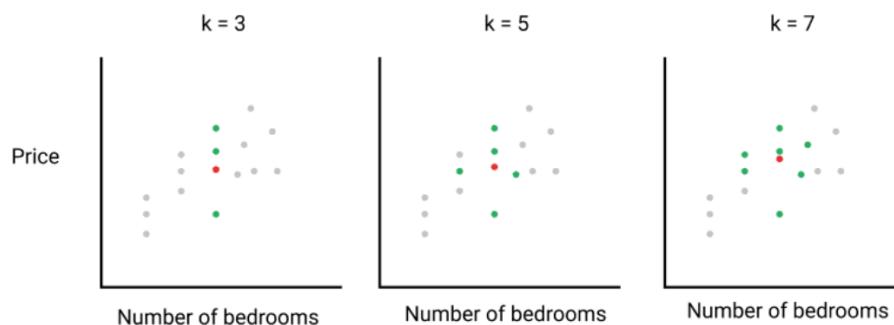


FIGURE 4.1 – Premièrement, nous sélectionnons le nombre d'annonces similaires K



FIGURE 4.2 – Deuxièmement, nous calculons la similarité de chaque annonce avec la nôtre en utilisant une métrique de similarité.

dataset (ordered by similarity)			our unpriced listing	
bedrooms	price	similarity	bedrooms	price
1	160	0		
1	60	0		
1	95	0		
1	50	0		
3	350	2		

FIGURE 4.3 – Troisièmement, nous classons chaque annonce à l'aide de notre métrique de similarité et on sélectionne les K premières annonces.



FIGURE 4.4 – Enfin, nous calculons le prix moyen pour les K annonces similaires et l'utilisons comme prix d'annonce.

Et bien que les images ci-dessus n'utilisent que deux caractéristiques (chambres et prix) par observation à comparer avec notre annonce par simplicité, l'apprentissage automatique nous permet de faire des comparaisons beaucoup plus complexes en évaluant beaucoup plus de caractéristiques pour chaque observation.

4.1.1 Choix de la distance

Lorsqu'on essaye de prédire une valeur continue comme le prix, la principale métrique de similarité utilisée est la distance euclidienne. Voici la formule générale de la distance euclidienne :

$$D_e = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \quad (4.1)$$

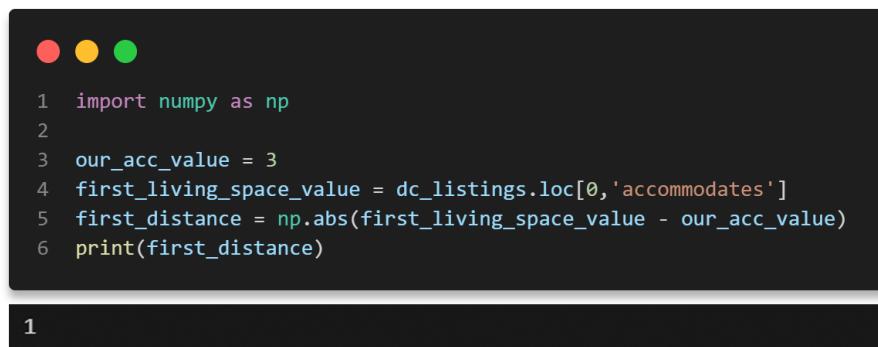
Où $(q_1 \dots q_n)$ représente les valeurs de caractéristique pour une observation et $(p_1 \dots p_n)$ représente les valeurs de caractéristique pour l'autre observation.

4.1.2 Construction d'un modèle basique

Commençons par décomposer les choses en examinant une seule colonne de notre ensemble de données. La formule de la distance pour une seule caractéristique (feature) devient alors : $D_e = \sqrt{(q_1 - p_1)^2}$ qui devient $D_e = |q_1 - p_1|$ par simplification de la racine par la puissance carrée.

Nous voudrions trouver la valeur absolue de la différence entre l'observation et le point de données que nous voulons prédire pour la caractéristique que nous utilisons. Pour ce premier modèle, supposons que l'appartement que nous souhaitons louer peut accueillir trois personnes.

Nous allons d'abord calculer la distance, en utilisant uniquement la colonne **accommodation**, entre le premier appartement de la base de données et le nôtre. Nous pouvons utiliser la fonction NumPy [21] `np.abs()` pour obtenir la valeur absolue.



```

● ● ●
1 import numpy as np
2
3 our_acc_value = 3
4 first_living_space_value = dc_listings.loc[0, 'accommodates']
5 first_distance = np.abs(first_living_space_value - our_acc_value)
6 print(first_distance)

1

```

FIGURE 4.5 – Calcul de la distance entre le nouveau point de donnée et la première observation de la base de données

La plus petite distance euclidienne possible est zéro, ce qui signifierait que l'observation à laquelle nous comparons est identique à la nôtre, donc le résultat que nous avons obtenu ici est logique. Cependant, la valeur ne signifie pas grand-chose à moins que nous ne sachions comment elle se compare à d'autres valeurs. Calculons la distance euclidienne pour chaque observation dans notre ensemble de données et examinons la plage de valeurs que nous avons en utilisant `pd.value_counts()` :



```

1 dc_listings['distance'] = np.abs(dc_listings.accommodates - nv_accomodation)
2 dc_listings.distance.value_counts().sort_index()

```

0	461
1	2294
2	503
3	279
4	35
5	73
6	17
7	22
8	7
9	12
10	2
11	4
12	6
13	8

Name: distance, dtype: int64

FIGURE 4.6 – Calcul de la distance avec tous les observations de la base de données

On observe qu'il existe 461 annonces qui ont une distance de 0, et accueillent donc le même nombre de personnes que notre appartement. Ces annonces pourraient être un bon point de départ. Si nous utilisons simplement les cinq premières valeurs avec une distance de 0, nos prédictions seraient biaisées par rapport à l'ordre existant de l'ensemble de données. Au lieu de cela, nous allons randomiser l'ordre des observations, puis sélectionner les cinq premières lignes avec une distance de 0. Nous allons utiliser DataFrame.sample() pour randomiser les lignes. Cette méthode est généralement utilisée pour sélectionner une fraction aléatoire du DataFrame, mais nous lui dirons de sélectionner au hasard 100%, ce qui va mélanger les lignes au hasard pour nous. Nous utiliserons également le paramètre random_state égale à 0 qui nous donnera un ordre aléatoire reproductible.



```

1 dc_listings = dc_listings.sample(frac=1,random_state=0)
2 dc_listings = dc_listings.sort_values('distance')
3 dc_listings.price.head()

```

FIGURE 4.7 – La randomisation de la base de données puis le tri des distances de façon croissante

```

2645    $75.00
2825    $120.00
2145    $90.00
2541    $50.00
3349    $105.00
Name: price, dtype: object

```

FIGURE 4.8 – L'affichage des 5 premières lignes ordonnées aléatoirement avec le prix associé

Avant de pouvoir prendre la moyenne de nos prix, on remarque que notre colonne de prix a le type d'objet, en raison du fait que les prix ont des signes dollar et des virgules (notre exemple ci-dessus ne montre pas les virgules car toutes les valeurs sont moins de 1 000 \$).

Nettoyons cette colonne en supprimant ces caractères et en la convertissant en type flottant, avant de calculer la moyenne des cinq premières valeurs. Nous utiliserons le Series.str.replace() de pandas pour supprimer les caractères parasites.

```

● ● ●
1 dc_listings['price'] = dc_listings.price.str.replace("\$|,",'').astype(float)
2 mean_price = dc_listings.price.iloc[:5].mean()
3 print(mean_price)

```

```
88.0
```

FIGURE 4.9 – Suppression du symbole \$ et calcul du prix moyen

Nous avons maintenant fait notre première prédiction, notre modèle KNN nous a dit que lorsque nous n'utilisons que la colonne **accommodates** pour trouver un prix approprié pour notre annonce pour trois personnes, nous devrions louer notre appartement pour 88 \$.

C'est un bon début, mais le problème est que nous n'avons aucun moyen de savoir à quel point ce modèle est précis, ce qui le rend impossible à optimiser et à améliorer. Ce n'est pas une bonne idée de suivre les prédictions d'un modèle d'apprentissage automatique sans évaluer sa précision

4.1.3 Évaluation du modèle

Un moyen simple de tester la qualité de notre modèle consiste à :

1. Diviser l'ensemble de données en 2 partitions :
 - Un ensemble d'entraînement : contient la majorité des lignes (75%)
 - Un ensemble de test : contient la minorité restante des lignes (25%)
2. Utiliser les lignes de l'ensemble d'apprentissage pour prédire la valeur de prix des lignes de l'ensemble de test
3. Comparer les valeurs prédites avec les valeurs de prix réelles dans l'ensemble de test pour voir la précision des valeurs prédites

Nous allons suivre cette approche et diviser les 3 723 lignes de notre ensemble de données en deux parties : train_df et test_df dans une répartition de 75% à 25%.

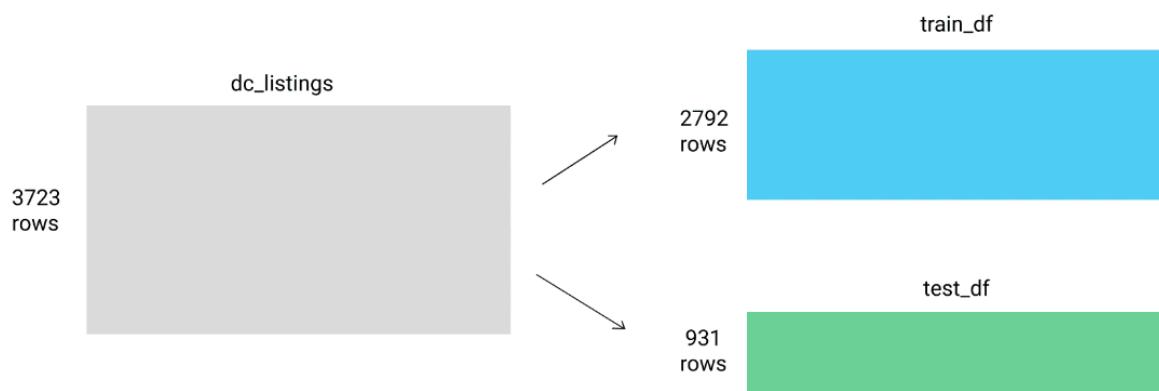


FIGURE 4.10 – Séparation de l'ensemble de données en deux sous-ensembles : un pour l'entraînement et l'autre pour le test

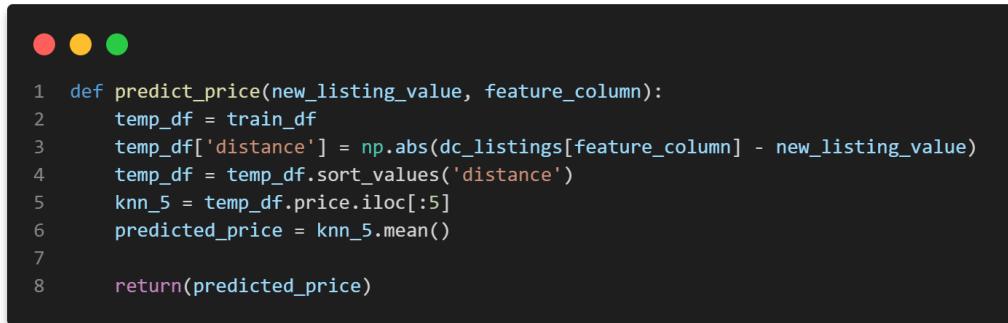
```

● ● ●
1 train_df = dc_listings.copy().iloc[:2792] # 75%
2 test_df = dc_listings.copy().iloc[2792:] # 25%

```

FIGURE 4.11 – Division de l'ensemble de données d'origine en deux sous-ensembles

Pour nous faciliter la tâche pendant que nous examinons les métriques, nous allons combiner le modèle que nous avons créé précédemment dans une fonction. Nous n'aurons pas à nous soucier de la randomisation des lignes, car elles sont toujours randomisées depuis le début.



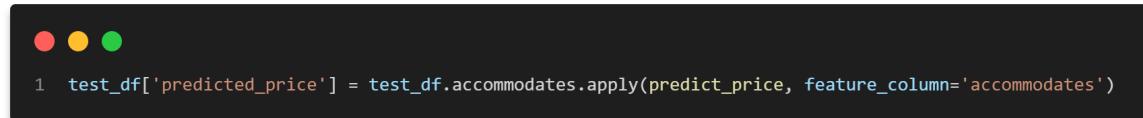
```

1 def predict_price(new_listing_value, feature_column):
2     temp_df = train_df
3     temp_df['distance'] = np.abs(dc_listings[feature_column] - new_listing_value)
4     temp_df = temp_df.sort_values('distance')
5     knn_5 = temp_df.price.iloc[:5]
6     predicted_price = knn_5.mean()
7
8     return(predicted_price)

```

FIGURE 4.12 – Construction de la fonction KNN qui prédit un prix moyen d'une nouvelle observation

Nous pouvons maintenant utiliser cette fonction pour prédire les prix de notre ensemble de données de test en se basant seulement sur la colonne accommodates comme caractéristique.



```

1 test_df['predicted_price'] = test_df.accommodates.apply(predict_price, feature_column='accommodates')

```

FIGURE 4.13 – Prédiction des prix grâce à la fonction predict_price

Évaluation du modèle en utilisant le RMSE

Pour de nombreuses tâches de prédiction, nous voulons pénaliser beaucoup plus les valeurs prédites qui sont plus éloignées de la valeur réelle que celles qui sont plus proches de la valeur réelle. C'est pourquoi nous avons choisi d'utiliser le RMSE ou la racine de l'erreur quadratique moyenne, qui est donnée par cette équation :

$$RMSE = \sqrt{\frac{(actuelle_1 - predite_1)^2 + (actuelle_2 - predite_2)^2 + \dots + (actuelle_n - predite_n)^2}{n}} \quad (4.2)$$

Où n représente le nombre de lignes dans l'ensemble de test.

Maintenant, calculons la valeur RMSE pour les prédictions que nous avons faites sur l'ensemble de test :

```
● ● ●
1 test_df['squared_error'] = (test_df['predicted_price'] - test_df['price'])**2
2 mse = test_df['squared_error'].mean()
3 rmse = mse ** (1/2)
4 print(rmse)
```

```
212.9892796705153
```

FIGURE 4.14 – Calcul de RMSE du modèle

L'une des choses pratiques à propos de RMSE est que, parce que nous mettons au carré puis prenons la racine carrée, les unités de RMSE sont les mêmes que la valeur que nous prédisons, ce qui facilite la compréhension de l'échelle de notre erreur. Nous avons obtenu une RMSE d'environ 212\$. Dans ce cas, cette échelle est assez grande par rapport à la fourchette de prix de notre ensemble de données. Nous sommes encore assez loin d'avoir une prédiction précise.

4.2 Comparaison des modèles

Maintenant que nous avons une métrique d'erreur que nous pouvons utiliser pour voir la précision de notre modèle, créons d'autres modèle qui utilisent différentes colonnes/caractéristiques et regardons comment notre erreur varie :

```
● ● ●
1 for feature in ['accommodates','bedrooms','bathrooms','number_of_reviews']:
2     test_df['predicted_price'] = test_df.accommodates.apply(predict_price,feature_column=feature)
3     test_df['squared_error'] = (test_df['predicted_price'] - test_df['price'])**2
4     mse = test_df['squared_error'].mean()
5     rmse = mse ** (1/2)
6
7     print("RMSE for the {} column: {}".format(feature,rmse))
```

```
RMSE for the accommodates column: 212.9892796705153
RMSE for the bedrooms column: 216.49048609414766
RMSE for the bathrooms column: 216.89419042215704
RMSE for the number_of_reviews column: 240.2152831433485
```

FIGURE 4.15 – Calcul de RMSE du modèle pour différentes caractéristiques

Nous pouvons constater que le meilleur modèle des quatre que nous avons formés est celui utilisant la colonne/caractéristique **accommodates**. Cependant, les taux d'erreur que nous obtenons sont assez élevés par rapport à la fourchette de prix des annonces dans notre ensemble de données. Les prévisions avec ce genre de taux d'erreur ne seraient pas très utiles. Jusqu'à présent, nous avons formé notre modèle avec une seule caractéristique, connue sous le nom de modèle univarié. Pour plus de précision, nous pouvons lui faire évaluer plusieurs caractéristiques en même temps, ce qui est connu sous le nom de modèle multivarié.

Nous remarquons que même si les colonnes `accommodates`, `bedrooms`, `bathrooms`, `beds`, et `minimum_nights` se trouvent dans un intervalle entre 0 et 7 (au moins dans les premières lignes), les valeurs des colonnes `maximum_nights` et `number_of_reviews` couvrent des intervalles beaucoup plus larges. Par exemple, la colonne `maximum_nights` a des valeurs aussi basses que 30 et hautes que 1125, dans les premières lignes. Si nous utilisons ces 2 colonnes dans le cadre d'un modèle k-plus proches voisins, ces attributs pourraient finir par avoir un effet démesuré sur les calculs de distance en raison de la taille des valeurs.

Par exemple, 2 appartements pourraient être identiques pour chaque attribut mais être très différents uniquement sur la colonne `maximum.nights`. Si une liste avait une valeur `maximum_nights` de 1125 et l'autre une valeur `maximum_nights` de 30, en raison de la façon dont la distance euclidienne est calculée, ces listes seraient considérées comme très éloignées en raison de l'effet démesuré que la grandeur des valeurs avait sur l'ensemble des distances euclidiennes. Pour éviter qu'une seule colonne ait un impact trop important sur la distance, nous pouvons normaliser toutes les colonnes pour avoir une moyenne de 0 et un écart type de 1.

Pour normaliser les valeurs d'une colonne par rapport à la distribution normale standard, on doit soustraire de chaque valeur la moyenne de la colonne, puis diviser chaque valeur par l'écart type de la colonne :

$$z = \frac{x - \mu}{\sigma} \quad (4.3)$$

où x est une valeur dans une colonne spécifique, μ est la moyenne de toutes les valeurs de la colonne et σ est l'écart type de toutes les valeurs de la colonne.

Procédons maintenant à la normalisation de notre ensemble de données, en appliquant l'équation ci-dessus :



```
1 normalized_listings = (dc_listings - dc_listings.mean()) / (dc_listings.std())
2 normalized_listings['price'] = dc_listings['price']
3 normalized_listings.head()
```

	accommodates	bedrooms	bathrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
0	0.401366	-0.249467	-0.439151	0.297345	-0.341375	-0.016573	-0.516709	160.0
1	1.399275	2.129218	2.969147	1.141549	-0.065038	-0.016603	1.706535	350.0
2	-1.095499	-0.249467	1.264998	-0.546858	-0.065038	-0.016573	-0.482505	50.0
3	-0.596544	-0.249467	-0.439151	-0.546858	-0.341375	-0.016573	-0.516709	95.0
4	0.401366	-0.249467	-0.439151	-0.546858	1.316644	-0.016573	-0.516709	50.0

FIGURE 4.16 – Les cinqs premières lignes de la base de données normalisée

Nous allons ensuite randomisés les lignes de notre ensemble de données et le divise en deux sous ensembles l'un d'entraînement contenant 75% des lignes et l'un de test contenant les 25% restantes :



```
1 normalized_listings = normalized_listings.sample(frac=1,random_state=0)
2 # normalized_listings.shape : (3671, 8)
3 norm_train_df = normalized_listings.copy().iloc[0:2753] # 75%
4 norm_test_df = normalized_listings.copy().iloc[2753:] # 25 %
```

FIGURE 4.17 – Division de l'ensemble de données en deux sous ensembles, un pour l'entraînement et un pour le test

4.2.1 Calcul de la distance euclidienne avec plusieurs caractéristiques

Rappelons-nous à quoi ressemblait l'équation de distance euclidienne originale :

$$d = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \quad (4.4)$$

Nous allons commencer par construire un modèle qui utilise les caractéristiques accommodates et bathrooms. Dans ce cas, notre équation euclidienne ressemblerait à :

$$d = \sqrt{(accommodates_1 - accommodates_2)^2 + (bathrooms_1 - bathrooms_2)^2} \quad (4.5)$$

Pour trouver la distance entre deux appartements, nous devons calculer la différence au carré entre les deux valeurs de accommodates, la différence au carré entre les valeurs de bathrooms, les additionner, puis prendre la racine carrée de la somme résultante. Voici à quoi ressemble la distance euclidienne entre les deux premières lignes dans normalized_listings :

accommodates	bathrooms
-0.596544	-0.439151
-0.596544	0.412923

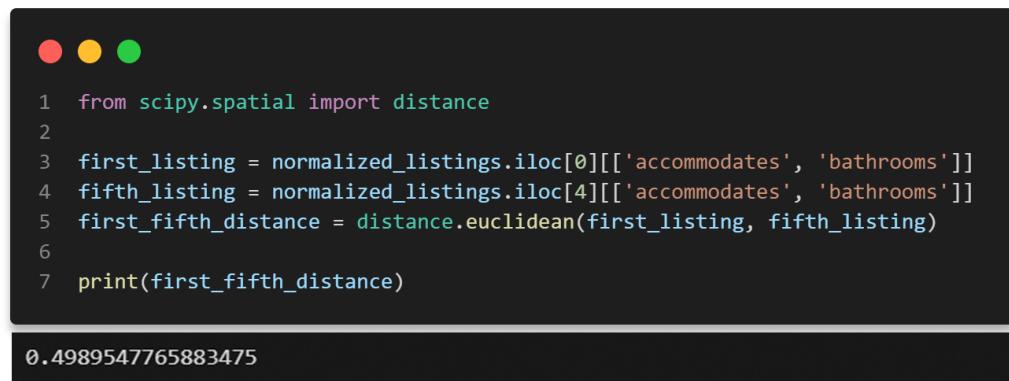
$(q_1 - p_1) + (q_2 - p_2) + \dots + (q_n - p_n)$	differences	$(-0.596544 + 0.596544) + (-0.439151 - 0.412923)$
$(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2$	squared differences	$(0)^2 + (-0.852074)^2$
$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2}$	Euclidean distance	$= \sqrt{0 + 0.72603}$ $= 0.852074$

FIGURE 4.18 – Distance euclidienne entre les deux premières lignes dans la BDD normalisée

Jusqu'à présent, nous avons calculé nous-mêmes la distance euclidienne en écrivant nous-mêmes la logique de l'équation. Nous pouvons à la place utiliser la fonction `distance.euclidean()` de `scipy.spatial` [22], qui prend deux vecteurs comme paramètres et calcule la distance euclidienne entre eux. La fonction `euclidean()` attend que :

- Les deux vecteurs doivent être représentés à l'aide d'un objet de type liste
- Les deux vecteurs doivent être unidimensionnels et avoir le même nombre d'éléments

Utilisons la fonction euclidean() pour calculer la distance euclidienne entre la première et la cinquième ligne de notre ensemble de données :



```

● ● ●

1 from scipy.spatial import distance
2
3 first_listing = normalized_listings.iloc[0][['accommodates', 'bathrooms']]
4 fifth_listing = normalized_listings.iloc[4][['accommodates', 'bathrooms']]
5 first_fifth_distance = distance.euclidean(first_listing, fifth_listing)
6
7 print(first_fifth_distance)

```

0.4989547765883475

FIGURE 4.19 – Distance euclidienne entre la 1ère et la 5ème ligne

4.3 Création d'un modèle KNN multivarié

Nous pouvons étendre notre fonction précédente pour utiliser deux fonctionnalités et l'ensemble de notre ensemble de données. Au lieu de distance.euclidean(), nous allons utiliser distance.cdist(), car cela nous permet de passer plusieurs lignes à la fois. La méthode cdist() peut être utilisée pour calculer la distance à l'aide de diverses méthodes, mais elle est par défaut euclidienne.



```

● ● ●

1 def predict_price_multivariate(new_listing_value, feature_columns):
2     temp_df = norm_train_df
3     temp_df['distance'] = distance.cdist(temp_df[feature_columns], [new_listing_value[feature_columns]])
4     temp_df = temp_df.sort_values('distance')
5     knn_5 = temp_df.price.iloc[:5]
6     predicted_price = knn_5.mean()
7     return(predicted_price)
8
9 cols = ['accommodates', 'bathrooms']
10
11 norm_test_df['predicted_price'] = norm_test_df[cols].apply(predict_price_multivariate, feature_columns=cols, axis=1)
12 norm_test_df['squared_error'] = (norm_test_df['predicted_price'] - norm_test_df['price'])**2
13 mse = norm_test_df['squared_error'].mean()
14 rmse = mse ** (1/2)
15 print(rmse)

```

118.92

FIGURE 4.20 – Fonction K-NN de prédiction d'un modèle multivarié

On remarque que notre RMSE est passé de 212 à 118 lors de l'utilisation des deux caractéristiques : accommodates et bathrooms.

4.3.1 Creation d'un modele a l'aide de Scikit-learn

Jusqu' prsent, nous avons crit manuellement les fonctions  partir de zro pour former nos modles K plus proches voisins. Maintenant, nous pouvons travailler beaucoup plus rapidement et efficacement en utilisant la bibliothque scikit-learn [20] de Python.

Scikit-learn est la bibliothque d'apprentissage automatique la plus populaire en Python. Elle possde des fonctions intgres pour tous les principaux algorithmes d'apprentissage automatique et un flux de travail simple et unifi. Ces deux proprits permettent aux data scientists d'tre incroyablement productifs lors de la formation et du test de diffrents modles sur un nouvel ensemble de donnes. [3]

Le flux de travail scikit-learn se compose de quatre tapes principales :

1. Instancier le modle d'apprentissage automatique spcifique qu'on souhaite utiliser.
2. Ajuster le modle aux donnes d'entranement
3. Utiliser le modle pour faire des prdictions
4. valuer l'exactitude des prdictions

Chaque modle dans scikit-learn est implment en tant que classe distincte et la premre tape consiste  identifier la classe dont nous voulons crer une instance. Tout modle qui nous aide  prdire des valeurs numriques telles que les prix d'inscription dans notre modle est appel modle de rgression. L'autre classe principale de modle d'apprentissage automatique est appele classification. Les modles de classification sont utiliss lorsque nous essayons de prdire une tiquette  partir d'un ensemble fixe d'tiquettes (par exemple, groupe sanguin ou sexe).

Dans notre cas, nous voulons utiliser la classe **KNeighborsRegressor**. Le mot rgresseur du nom de classe KNeighborsRegressor fait rfrence  la classe de modle de rgression dont nous venons de parler, et "KNeighbors" vient du modle k-plus proches voisins que nous construisons.

Scikit-learn utilise un style orienté objet similaire à Matplotlib. Nous devons instancier un modèle vide avant de faire quoi que ce soit d'autre en appelant le constructeur. Si nous nous référons à la documentation, nous remarquons que par défaut :

- **n_neighbors** : le nombre de voisins, est défini sur 5
- **algorithm** : pour le calcul des voisins les plus proches, est défini sur auto
- **p** : mis à 2, correspondant à la distance euclidienne

Définissons le paramètre **algorithm** sur "brute" et laissons la valeur **n_neighbors** à 5, ce qui correspond à l'implémentation manuelle que nous avons construite auparavant.



```
1 from sklearn.neighbors import KNeighborsRegressor  
2 knn = KNeighborsRegressor(algorithm='brute')
```

FIGURE 4.21 – Initialisation du modèle KNeighborsRegressor depuis la librairie Scikit-learn

Ajustement du modèle et prédictions

Maintenant, nous pouvons ajuster le modèle aux données en utilisant la méthode d'ajustement. Pour tous les modèles, la méthode d'ajustement prend en compte deux paramètres obligatoires :

- Un objet de type matrice, contenant les colonnes de fonctionnalités que nous voulons utiliser à partir de l'ensemble d'apprentissage.
- Un objet de type liste, contenant les valeurs cibles correctes.

Un "objet de type matrice" signifie que la méthode est flexible et peut accepter soit un pandas DataFrame, soit un tableau NumPy 2D. Cela signifie que nous pouvons sélectionner les colonnes que nous voulons utiliser à partir du DataFrame et les utiliser comme premier paramètre pour la méthode d'ajustement. Pour le deuxième paramètre, rappelez-vous du précédent, tous les éléments suivants sont des objets de type liste acceptables comme un tableau NumPy, une liste Python ou encore un objet de la série pandas.

Lorsque la méthode fit() est appelée, scikit-learn stocke les données d'apprentissage que nous avons spécifiées dans l'instance KNearestNeighbors (KNN). Si nous essayons de transmettre des données contenant des valeurs manquantes ou des valeurs non numériques dans la méthode fit, scikit-learn renverra une erreur. C'est l'une des choses merveilleuses à propos de cette bibliothèque - elle contient de nombreuses fonctionnalités comme celle-ci pour nous empêcher de faire des erreurs.

Maintenant que nous avons spécifié les données d'apprentissage que nous voulons utiliser pour faire des prédictions, nous pouvons utiliser la méthode de prédiction pour faire des prédictions sur l'ensemble de test. La méthode de prédiction n'a qu'un seul paramètre obligatoire : Un objet de type matrice, contenant les colonnes de caractéristiques de l'ensemble de données sur lequel nous voulons faire des prédictions.

La méthode predict() renvoie un tableau NumPy contenant les valeurs de prix prévues pour l'ensemble de test.



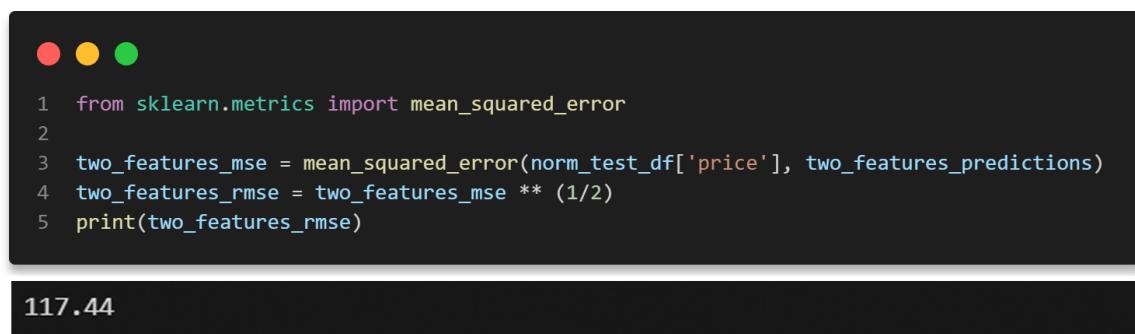
```
1 knn.fit(norm_train_df[cols], norm_train_df['price'])
2 two_features_predictions = knn.predict(norm_test_df[cols])
```

FIGURE 4.22 – Ajustement du modèle aux données de l'ensemblle d'apprentissage et prédictions des valeurs de prix

Calcul de RMSE à l'aide de Scikit-Learn

Jusqu'à présent, nous calculions manuellement les valeurs RMSE, en utilisant les fonctions NumPy et SciPy. Alternativement, nous pouvons utiliser à la place la fonction `sklearn.metrics.mean_squared_error()`. La fonction `mean_squared_error()` prend deux entrées :

- Un objet de type liste, représentant les vraies valeurs de votre jeu de test.
- Un deuxième objet de type liste, représentant les valeurs prédites générées par le modèle.



```
1 from sklearn.metrics import mean_squared_error
2
3 two_features_mse = mean_squared_error(norm_test_df['price'], two_features_predictions)
4 two_features_rmse = two_features_mse ** (1/2)
5 print(two_features_rmse)
```

117.44

FIGURE 4.23 – Calcul de RMSE à l'aide de scikit-learn

Non seulement c'est beaucoup plus simple du point de vue de la syntaxe, mais cela prend également moins de temps pour que le modèle s'exécute car scikit-learn a été fortement optimisé pour la vitesse. Nous observons que notre RMSE est un peu différent de notre algorithme implémenté manuellement, cela est probablement dû à la fois aux différences de randomisation et à de légères différences d'implémentation entre notre algorithme KNN "manuel" et la version scikit-learn.

Utilisation de plusieurs caractéristiques

Essayons de créer un modèle qui utilise quatre caractéristiques au lieu de deux et voyons si cela améliore nos résultats. Pour nous aider à sélectionner les caractéristiques les plus influentes sur le prix, nous pouvons utiliser la matrice de corrélation. Une matrice de corrélation est utilisée pour évaluer la dépendance entre plusieurs variables en même temps. Le résultat est une table contenant les coefficients de corrélation entre chaque variable et les autres. Il existe différentes méthodes de tests de corrélation : le test de corrélation de Pearson, la corrélation de Kendall et celle de Spearman qui sont des tests basés sur le rang.

Voilà à quoi ressemble la matrice de corrélation pour notre ensemble de données, en utilisant seaborn ; une bibliothèque Python de visualisation de données basée sur matplotlib. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs.

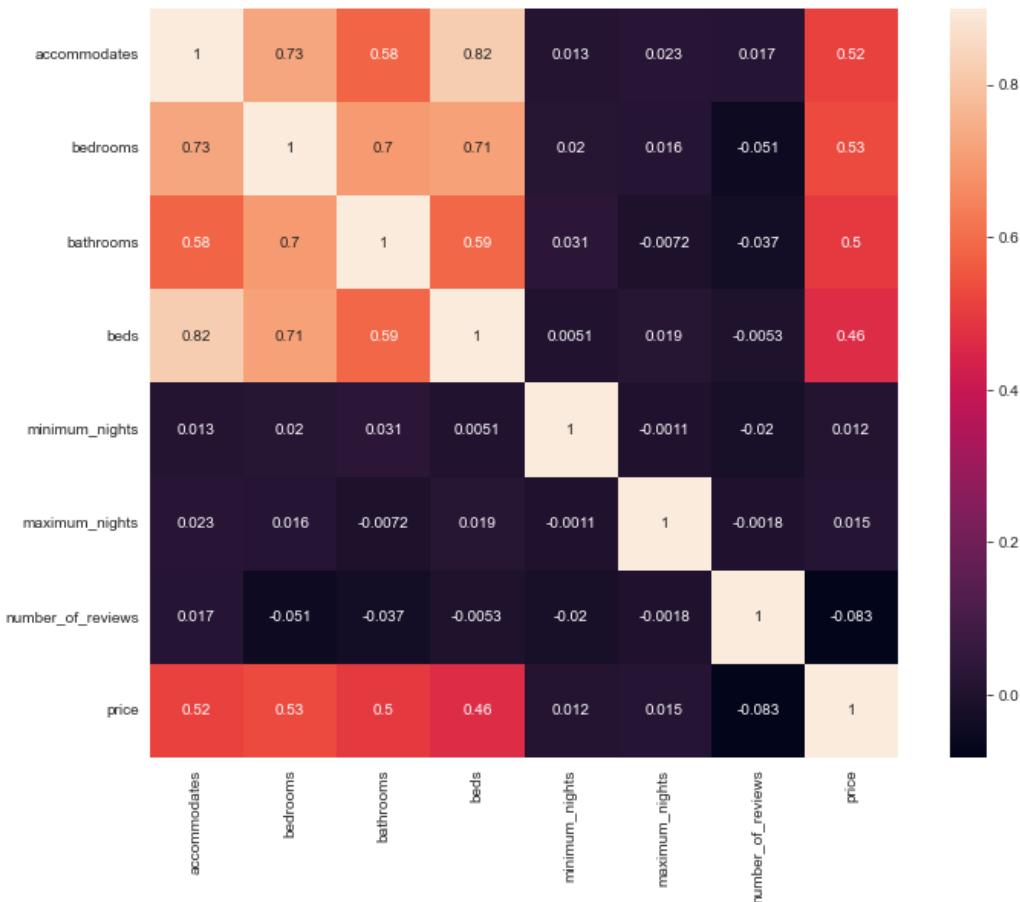


FIGURE 4.24 – Matrice de corrélation

Nous remarquons que les caractéristiques `accommodates`, `bedrooms`, `bathrooms` et `beds` sont les plus corrélées avec le prix d'après la matrice. Cependant, les variables `accommodates` et `bedrooms` sont aussi étroitement corrélées, car le nombre de personnes qu'un appartement peut accueillir est étroitement lié au nombre de chambres de cet appartement, par conséquent, nous n'avons besoin que d'une de ces variables. On pourrait dire la même chose pour les variables `beds` et `bedrooms`. Mais pour raison d'expérimentation, créons un modèle qui utilise les quatres caractéristiques :



```

1 knn = KNeighborsRegressor(algorithm='brute')
2 cols = ["accommodates", "bathrooms", "beds", "bedrooms"]
3 knn.fit(norm_train_df[cols], norm_train_df['price'])
4 four_features_predictions = knn.predict(norm_test_df[cols])
5 four_features_mse = mean_squared_error(norm_test_df['price'], four_features_predictions)
6 four_features_rmse = four_features_mse ** (1/2)
7 print("%.2f" % four_features_rmse)

```

118.68

FIGURE 4.25 – Calcul du RMSE avec un modèle à 4 caractéristiques

Comme prévu, l'ajout de caractéristiques supplémentaires ne produit pas nécessairement un modèle plus précis. Pour notre cas, l'utilisation de quatres caractéristiques ne fait pas de mal, mais imaginons le cas des entreprises comme Facebook ou Amazon qui ont des ensembles de données avec un nombre gigantesque de caractéristiques, il est impossible de tous les utiliser, c'est pourquoi il faut soigneusement choisir les caractéristiques les plus significatives. Une façon de mieux comprendre si une certaine valeur RMSE est "bonne" est de la normaliser à l'aide de la formule suivante :

$$RMSE_n = \frac{RMSE}{y_{max} - y_{min}} \quad (4.6)$$

$y_{max} - y_{min}$ étant la différence entre le prix maximum et le prix minimum. Cela produit une valeur comprise entre 0 et 1, où les valeurs plus proches de 0 représentent des modèles mieux ajustés.



```

1 normRMSE = four_features_rmse / (dc_listings['price'].max() - dc_listings['price'].min())
2 print("%.3f" % normRMSE)

```

0.042

FIGURE 4.26 – Calcul du RMSE normalisé

Pour notre cas nous obtenons une RMSE normalisé de 0.04 qui très proche de 0, donc on peut conclure que le modèle principal à retenir comme solution à notre problème est le modèle qui utilisent les deux caractéristiques : accommodates et bathrooms, puisque la distance moyenne entre les prix prédicts et les prix réels est la plus faible parmi tous les modèles précédents, et il est le plus économique en terme de calcul et de temps.

Conclusion

Nous nous sommes intéressés dans ce projet à la tâche de prédiction du prix de location de logements de particuliers sur une plateforme communautaire payante de location de logements de particuliers Airbnb, grâce à un modèle d'apprentissage automatique. Le résultat de ce projet, nous a permis d'automatiser la tâche d'affectation de prix idéal de location à chaque logement selon ses caractéristiques comme le nombre de chambres, le nombre de salles de bains etc... Celà a été possible grâce à l'exploitation des milliers de données d'annonces générées par des hôtes sur la plateforme, une tâche qui prendra énormément de temps et d'efforts pour les futures hôtes désirant mettre leur logement en location.

Dans le premier chapitre, nous avons initié la notion de l'apprentissage automatique ainsi que quelques-unes de ces applications dans différents domaines qui impactent nos vies quotidiennes. Puis on a enchaîné avec les quatre types majeurs de l'apprentissage automatique : supervisé, non supervisé, en profondeur et par renforcement avec des exemples pour chacuns d'entre eux. Nous avons également défini la classification et la régression, les deux types de problèmes les plus rencontrés dans l'apprentissage automatique.

Dans le chapitre qui suit, nous nous sommes approfondis dans quelques-uns des algorithmes les plus utilisés dans l'apprentissage non supervisé tel que l'analyse en composantes principales et le K moyennes clustering, et supervisé tel que la classification naïve de Bayes, la régression linéaire, les forêts aléatoires et le K plus proches voisins. Nous avons défini ces derniers et nous avons traité quelques exemples pour bien comprendre le fonctionnement de chacun de ces algorithmes.

Dans le chapitre trois, nous avons encerclé et détaillé le problème traité dans ce projet, avant d'entamer la phase de la préparation de l'ensemble de données qui comporte les étapes de bases telles que : l'importation et l'exploration de la base de données. Ces étapes nous fluidifient et facilitent la manipulation des données plus tard dans la phase de modélisation.

Dans le quatrième chapitre, nous avons entamé la construction de notre modèle basique basé sur l'algorithme des K plus proches voisins, qui utilise dans un premier temps qu'une seule caractéristique. Après l'évaluation du modèle, nous avons créé un modèle multivarié qui admet plusieurs caractéristiques et nous avons comparé les résultats avec le modèle à une seule variable. Et pour terminer, nous avons eu recours à la librairie Scikit-learn, grâce à laquelle nous avons créé un modèle KNN avec plusieurs caractéristiques qui a eu la meilleure précision parmi tous et qui a été choisi comme modèle final.

En guise de conclusion, le modèle que nous avons créé peut être potentiellement intégré dans la partie backend de la plateforme Airbnb, où il générera automatiquement des recommandations de prix de location du logement dès que l'annonceur fait entrer les caractéristiques de ce dernier, chose qui offrira une agréable expérience à l'utilisateur puisqu'il lui économisera du temps et d'efforts. Bien que l'intelligence artificielle ait participé à la réalisation d'énormes progrès scientifiques au niveau de la santé, l'astronomie et d'autres disciplines, cette technologie a un impact sur l'avenir de pratiquement toutes les industries et de tous les êtres humains. L'intelligence artificielle a été le principal moteur des technologies émergentes telles que la bigdata, la robotique et l'IoT, et elle continuera d'agir en tant qu'innovation technologique dans l'avenir.

Bibliographie

- [1] Judith Hurwitz, Daniel Kirsch, IBM, Machine learning for dummies, Le machine learning et la science des données
- [2] Jake Vanderplas, Python Data Science Handbook
- [3] Aurélien Géron, Machine Learning avec Scikit-Learn - 2e édition
- [4] Andreas Mueller, Sarah Guido, Introduction to Machine Learning with Python, 2016
- [5] James. Gareth, Witten. Daniela, Hastie. Trevor, Tibshirani. Robert, Intoduction to Statistical Learning with Applications in R.
- [6] Luca Massaron, John Paul Mueller, Machine Learning for Dummies, 2016
- [7] <https://openai.com/blog/openai-five/>
- [8] <https://openai.com/blog/dall-e/>
- [9] <https://www.deepmind.com/research/highlighted-research/alphago>
- [10] Aurélien Géron, Hands-on Machine Learning with Sckit-learn, Keras & Tensorflow, 2nd Edition 2019
- [11] <https://drdk.me/analyse-en-composante-principale.html>
- [12] <https://archive.ics.uci.edu/ml/datasets/iris>
- [13] Kadir Yasar, PCA : Principal Component Analysis, Medium
- [14] Rohith Gandhi, Naive Bayes Classifier, Medium
- [15] Scikit-learn documentation, Clustering, K-Means
- [16] <https://mrmint.fr/introduction-k-nearest-neighbors>
- [17] Celeste Grupman, Python Machine Predicting Airbnb Prices, Dataquest
- [18] <http://insideairbnb.com/get-the-data/>
- [19] <https://pandas.pydata.org/docs/>
- [20] <https://scikit-learn.org/stable/>
- [21] <https://numpy.org/doc/stable/>
- [22] <https://docs.scipy.org/doc/scipy/>