Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate (DiSTA)
Cours d'Intelligence Artificielle pour l'Astrophysique

Classification de Galaxies

Projet final
Lucas Barbier-Goy

Année Académique 2022-2023

"Uno dei miei giorni più produttivi è stato quando ho buttato via 1000 righe di codice."
- Ken Thompson (Scienziato informatico, sviluppatore del sistema operativo UNIX)

Table des matières

Table des figures

Liste des tableaux

1 Introduction

La quantité massive de données provenant de télescopes ou d'autres types de capteurs en physique rend aujourd'hui impossible la classification et le traitement manuel de ces données. Pour résoudre ce problème, de nombreuses techniques ont été développées, notamment l'apprentissage automatique et l'apprentissage profond, qui permettent entre autres la classification en masse de ces données ou la création de modèles prédictifs. Selon le type de données traitées, certaines méthodes seront plus appropriées que d'autres.

Dans le cadre de ce projet, nous nous intéressons au jeu de données "Galaxy Zoo", un projet scientifique participatif dans lequel des images de galaxies ont été analysées et classifiées par des citoyens volontaires. Le jeu de données se compose de deux parties : une partie "traitée" où l'on trouve certaines caractéristiques des galaxies telles que classifiées par les participants, et une partie brute qui présente les images telles qu'observées par les télescopes. L'objectif est de prédire si une galaxie est plutôt de type spirale ou elliptique.

Dans la première partie de ce projet (3.1), nous comparons les performances de différents estimateurs classiques, tandis que dans la seconde partie (3.3), nous utilisons un réseau neuronal convolutif pour faire des prédictions sur les images. Nous explorons également le jeu de données à l'aide de l'apprentissage non supervisé pour déterminer quels attributs du jeu de données ont le plus d'impact sur la décision des estimateurs. Cette partie (3.2), appelée Analyse en Composantes Principales (ACP), nous permettra de mieux comprendre le jeu de données étudié.

Enfin, l'objectif est de déterminer si la précision atteinte par ces méthodes est comparable à celle des humains. Autrement dit, est-ce qu'il serait possible d'utiliser

ces méthodes d'apprentissage automatique pour trier automatiquement les prochaines données de SDSS [1].

1. Il s'avère que ces méthodes sont déjà à l'emploi dans de nombreux cas pour le traitement d'images de télescopes.

2

Jeu de données

Dans cette étude, nous nous intéressons à la classification de galaxies de types spirale et elliptique à partir de données fournies par le projet scientifique citoyen « Galaxy Zoo ». Pour cela, nous avons utilisé deux jeux de données.

Le premier jeu de données est une table CSV contenant les classifications de galaxies effectuées par les participants de Galaxy Zoo [1]. Cette table contient des informations sur 667 944 galaxies et comporte seize colonnes. Les colonnes comprennent l'identifiant unique de la galaxie dans le catalogue SDSS, sa position dans le ciel (ascension droite et déclinaison), le nombre de votes obtenus pour cette galaxie, ainsi que les probabilités estimées par les participants pour six catégories différentes : elliptique, spirale avec rotation dans le sens horaire, spirale avec rotation dans le sens anti-horaire, galaxie avec bord flou, galaxie avec noyau dominant et galaxie avec plusieurs noyaux. De plus, la table contient des colonnes indiquant le type estimé de la galaxie, ainsi que des colonnes pour les probabilités corrigées pour les biais de classification (voir Tab. 2.1).

L'arbre de décision utilisé par Galaxy Zoo (voir Fig. 2.1) est basé sur une série de questions binaires qui permettent de distinguer différentes formes de galaxies. Ce processus commence par la distinction entre les galaxies elliptiques et les galaxies spirales, en fonction de la présence ou de l'absence de bras spiraux. Ensuite, les galaxies spirales sont classées en fonction de la direction de rotation de leurs bras, tandis que les galaxies elliptiques sont divisées en deux sous-catégories en fonction de leur symétrie.

Le deuxième jeu de données est un ensemble de fichiers d'images de galaxies correspondant aux classifications fournies dans la première table. L'idée est ici de démontrer

TABLE 2.1 – Description of Galaxy Zoo 1 catalog data

Column	Description
OBJID	Unique identifier of the galaxy in the SDSS catalog
RA	Right Ascension (in degrees) of the galaxy
DEC	Declination (in degrees) of the galaxy
NVOTE	Number of votes obtained for this galaxy
P_{EL}	Probability that the galaxy is elliptical
P_{CW}	Probability that the galaxy is a clockwise spiraled galaxy
P_{ACW}	Probability that the galaxy is an anticlockwise spiraled galaxy
P_{EDGE}	Probability that the galaxy is a galaxy with a blurred edge
P_{DK}	Probability that the galaxy is a galaxy with a dominant nucleus
P_{MG}	Probability that the galaxy is a galaxy with multiple nuclei
P_{CS}	Probability that the galaxy is a galaxy with strange features
$P_{EL_{DEBIASED}}$	Probability of the elliptical classification, corrected for bias effect
$P_{CS_{DEBIASED}}$	Probability of the classification with strange features, corrected for bias effect
SPIRAL	Number of votes for the "spiral" classification
ELLIPTICAL	Number of votes for the "elliptical" classification
UNCERTAIN	Number of votes for the "uncertain" classification

comment les méthodes d'apprentissage automatique peuvent gérer et faire des prédictions de haute précision sur les images brutes (ou traitées) des télescopes.

2.1 Traitement des données

Avant de commencer l'entraînement de modèle avec les données présentées ci-dessus, il convient de traiter le jeu de données. En effet, certaines caractéristiques présentes dans le jeu originel n'ont pas d'influence sur la classe de la galaxie. Notamment, la première colonne est un identifiant unique qui ne peut pas être une caractéristique pour notre modèle. Les deuxième et troisième colonnes représentent les positions des galaxies dans le ciel. Les modèles cosmologiques supposent que l'univers est homogène et isotrope, donc nous présumerons que la position dans le ciel de la galaxie n'est pas corrélée à nos classes/cibles. Une fois les attributs inutiles à la classifictaion abandonnés, il est important de vérifier la régularité de nos données. Par exemple, l'attribut "NVOTE" de notre ensemble de données a une échelle très différente et contient des valeurs aberrantes. Ces deux caractéristiques rendent difficile la visualisation des données et, plus important encore, elles peuvent altérer les performances prédictives de nombreux algorithmes d'apprentissage automatique. Des données non régularisés peuvent également ralentir ou même empêcher la convergence de nombreux estimateurs basés sur la descente de gradient. Une discussion des différents régularisateurs ou normaliseurs peut être trouvée ici : https://github.com/lukbrb/academic-physics/

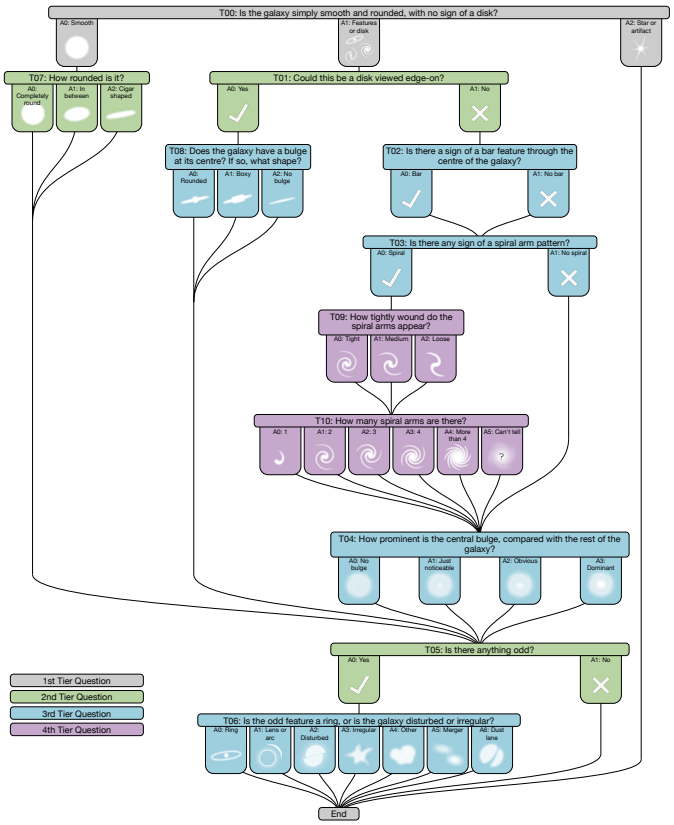

FIGURE 2.1 – Matrice de confusion pour l'algorithme du perceptron. L'algorithme a une forte tendance à confondre les galaxies spirales avec ...

blob/master/galaxy-classification/notebooks/00-Data-exploration.ipynb. On déduit dans cette analyse que le normalisateur 'Normalizer' de Scikit-learn est le meilleur choix, mais il s'avère en pratique l'échelonnage dit "min-max" donne les meilleurs résultats.

Finalement, il est important de vérifier que le jeu de données est bien équilibré pour éviter des biais lors de l'entraînement des modèles. Dans notre cas le jeu de données est largement déséquilibré (voir Fig. 2.2). Pour résoudre ce problème on échantillonne aléatoirement n galaxies de chaque classe, où n est le nombre de points pour la classe la moins représentée. Comme en général n est grand, on sélectionne pour les exemples suivant une valeur bien inférieure [1]. Un échantillon du jeu de données concrètement utilisé (avant normalisation) pour les entraînements est illustré Tab. 2.2.

Pour des questions de praticité avec l'API de Scikit-learn, nous avons également rassemblé les classes dans une unique colonne, et attribué à chacune d'entre-elles un chiffre :

1. La plupart des modèles montrés en exemple sont entraînés avec $n = 150$.

— Spirale : 0
— Elliptique : 1
— Incertaine : 2

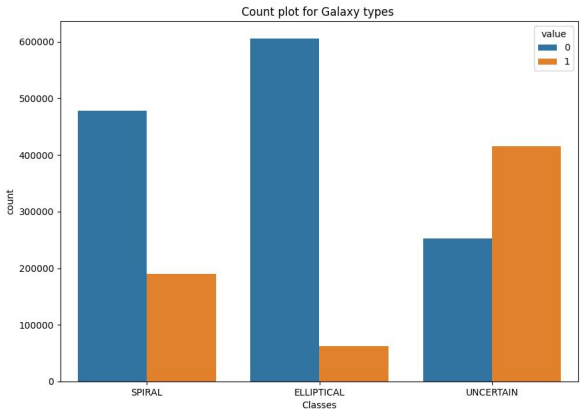

FIGURE 2.2 – Le nombre total de chaque galaxie est indiqué par la colonne orange. Il apparaît assez clairement que le jeu est déséquilibré en faveur de la classe "incertaines".

TABLE 2.2 – Reduced Galaxy Zoo1 Catalog Data used for the training.

NVOTE	**P_EL**	**P_CW**	**P_ACW**	**P_EDGE**	**P_DK**	**P_MG**	**P_CS**	**P_EL_DBS** [2]	**P_CS_DBS**	**CLASS**
59	0.6100	0.0340	0.0000	0.1530	0.1530	0.0510	0.1860	0.6100	0.1860	2
18	0.6110	0.0000	0.1670	0.2220	0.0000	0.0000	0.3890	0.2030	0.7970	0
68	0.7350	0.0290	0.0000	0.1470	0.0740	0.0150	0.1760	0.4320	0.4280	2
52	0.8850	0.0190	0.0000	0.0580	0.0190	0.0190	0.0770	0.8850	0.0770	1
59	0.7120	0.0000	0.0000	0.2200	0.0680	0.0000	0.2200	0.64	0.29	2

3

Aprentissage Automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui vise à doter les ordinateurs de la capacité à apprendre à partir de données. Cette discipline englobe plusieurs sous-parties, notamment l'apprentissage supervisé, l'apprentissage non-supervisé et l'apprentissage par renforcement.

Dans l'apprentissage supervisé, les algorithmes sont entraînés sur des données d'entraînement étiquetées, c'est-à-dire des données pour lesquelles la réponse correcte est connue. Les algorithmes apprennent à partir de ces données en ajustant leurs paramètres afin de minimiser l'erreur entre la réponse prédite et la réponse réelle. Lorsque l'apprentissage est terminé, l'algorithme peut être utilisé pour prédire la réponse pour de nouvelles données.

Dans cette analyse, nous nous concentrerons sur les techniques d'apprentissage supervisé pour la classification de galaxies. Nous utiliserons des modèles d'apprentissage supervisé classique pour traiter les données tabulaire, mais également un modèle d'apprentissage profond pour traiter les images de galaxies. L'apprentissage profond est une technique avancée d'apprentissage automatique qui utilise des réseaux de neurones pour apprendre à partir de données. Ces réseaux de neurones sont capables d'apprendre des représentations de plus en plus abstraites des données à mesure qu'ils avancent dans les couches de neurones. Nous introduirons brièvement une technique d'apprentissage non-supervisé dans la section 3.2.

Ainsi, notre analyse combinera des techniques d'apprentissage supervisé et d'apprentissage profond pour classifier les galaxies en utilisant les images et les données tabulaires.

3.1 Apprentissage supervisé

[1] L'apprentissage supervisé est l'une des sous-composantes de l'apprentissage automatique. Le domaine de l'apprentissage supervisé peut être lui-même être divisé en deux sous-catégories : la classification et la régression. La classification est utilisée lorsque la variable à prédire est discrète, telle que la classification d'une espèce d'iris en trois catégories. À l'opposé, lorsque la variable à prédire est de nature continue, comme le prix d'une maison, les méthodes de régression sont utilisées.

Dans notre cas, nous souhaitons déterminer la nature de la galaxie, à savoir si elle est spirale ou elliptique. Cette tâche est de nature discrète et relève donc de la classification. Nous nous concentrerons ainsi sur les méthodes de classification. Parmi les estimateurs possibles, les suivants seront étudiés :

— Les classificateurs naïfs bayésiens
— La méthode des k plus proches voisins
— Les arbres de décision
— Les forêts d'arbres de décision
— AdaBoost
— Les machines à vecteurs de support
— Le perceptron
— Le perceptron multicouche

Ces méthodes seront évaluées et comparées en fonction de leur performance en termes de précision et de rappel. À noter que le rappel mesure le nombre de vrais positifs détectés par le classificateur par rapport au nombre total de positifs dans l'ensemble de données. Il est souvent utilisé en conjonction avec la précision pour évaluer les performances globales d'un classificateur.

3.1.1 Classification naïve bayésienne

Les méthodes naïves bayésiennes sont un ensemble d'algorithmes d'apprentissage supervisé qui sont basés sur l'application du théorème de Bayes. Ce théorème permet de calculer la probabilité conditionnelle d'un événement A sachant un événement B, à partir de la probabilité conditionnelle de B sachant A et des probabilités marginales de A et B. Le théorème de Bayes s'écrit ainsi :

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \tag{3.1}$$

Les méthodes naïves bayésiennes font l'hypothèse d'indépendance conditionnelle entre chaque paire de caractéristiques, étant donné la valeur de la variable appartenant à la

1. Le code pour cette section est accessible à l'adresse suivante : https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/01-Classifiers.ipynb.

classe. Cela se traduit mathématiquement par la modification suivante du théorème de Bayes :

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)} \tag{3.2}$$

Cette formule est la base de tous les classificateurs naïfs bayésiens. La seule différence entre ces classificateurs réside dans la distribution $P(x_i|y)$ de (3.2), c'est-à-dire la probabilité que x_i correspondant à la classe y. Ce type de classificateur fonctionne bien pour la classification de documents ou le filtrage des pourriels, par exemple. Les classificateurs naïfs bayésiens ont de nombreux avantages tels que le peu de données d'entraînement nécessaire, la vitesse ou l'insensibilité au fléau de la dimension. Néanmoins, bien que ce soient de bons classificateurs, ils s'avèrent être de mauvais estimateurs [2].

Parmi les lois de probabilités pour $P(x_i|y)$, on trouve notamment la loi normale, la loi de Bernoulli et la loi multinomiale. Parmi les classificateurs naïfs bayésiens on trouve :

— Classificateur naïf bayésien gaussien
— Classificateur naïf bayésien de Bernoulli
— Classificateur naïf bayésien multinomial
— Classificateur naïf bayésien complémentaire

En utilisant les méthodes naïves bayésiennes, nous pouvons donc construire des modèles de classification efficaces pour des problèmes avec des caractéristiques discrètes ou continues. Le choix de la distribution appropriée dépendra des caractéristiques des données et des hypothèses que nous faisons sur leur distribution. Pour déterminer quel classificateurs correspond le mieux à notre problème, nous les introduisons succinctement dans les parties suivantes.

Classificateur naïf bayésien gaussien

Dans cette version de l'algorithme, on suppose que la distribution de probabilité conditionnelle de chaque caractéristique pour chaque classe suit une loi normale :

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp -\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \tag{3.3}$$

où μ_y et σ_y^2 sont respectivement la moyenne et la variance de la caractéristique x_i pour la classe y. Ces paramètres sont estimés en utilisant le maximum de vraisemblance à partir des données d'apprentissage.

Le classificateur naïf bayésien gaussien fonctionne bien lorsque les données d'apprentissage suivent une distribution normale. Cependant, si ce n'est pas le cas, le classificateur peut être inexact. De plus, cette version du classificateur naïf bayésien est sensible aux valeurs aberrantes, ce qui peut également affecter la précision des prédictions. Néanmoins,

malgré ses limites, le classificateur naïf bayésien gaussien reste un choix populaire pour la classification de données en raison de sa simplicité et de sa rapidité d'exécution.

Classificateur naïf bayésien multinomial

Ce classificateur est souvent utilisé pour la classification de texte. La distribution est paramétrisée par des vecteurs $\boldsymbol{\theta}_y = (\theta_{y_1}, \ldots, \theta_{y_n})$ pour chaque classe y, où n est le nombre de caractéristiques et θ_{y_i} est la probabilité $P(x_i|y)$ que la caractéristique i apparaisse dans un échantillon de la classe y. Les paramètres $\boldsymbol{\theta}_y$ sont estimés par une version régularisée du maximum de vraisemblance. En particulier, la fréquence d'apparition relative $\hat{\theta}_{y_i}$ est calculée pour chaque caractéristique i dans chaque classe y :

$$\hat{\theta} y_i = \frac{N y_i + \alpha}{N_y + \alpha_n}$$

où N_{yi} est le nombre d'occurrences de la caractéristique i dans les échantillons de la classe y, N_y est le nombre total d'échantillons dans la classe y, α est un hyperparamètre de régularisation et $\alpha_n = n\alpha$ est la somme totale de l'hyperparamètre de régularisation pour toutes les caractéristiques. La valeur de α contrôle le degré de régularisation : plus α est grand, plus les $\boldsymbol{\theta}_y$ sont proches de valeurs uniformes, tandis que pour $\alpha = 0$, on retrouve l'estimation de maximum de vraisemblance non régularisée.

La distribution multinomiale est souvent utilisée pour modéliser des données qui comptent le nombre d'occurrences de chaque caractéristique dans un échantillon, comme les fréquences des mots dans un texte. Dans ce cas, les caractéristiques peuvent être considérées comme des comptages de mots, et θ_{y_i} est la probabilité que le mot i apparaisse dans un document de la classe y.

Classificateur naïf bayésien complémentaire

Le classificateur naïf bayésien complémentaire est une variante du classificateur naïf bayésien multinomial. Il a été conçu pour mieux gérer les jeux de données déséquilibrés. En effet, lorsque certaines classes sont sous-représentées, le classificateur naïf bayésien multinomial peut être biaisé en faveur des classes les plus représentées.

Le principe du classificateur naïf bayésien complémentaire est de considérer les compléments de chaque classe, c'est-à-dire les caractéristiques qui n'appartiennent pas à la classe. Ainsi, pour chaque classe y, on calcule les probabilités complémentaires $P(\neg x_i|y)$ pour chaque caractéristique i. Les paramètres du modèle sont ensuite déterminés en utilisant ces probabilités complémentaires pour calculer les pondérations du modèle.

Plus précisément, la probabilité $P(x_i|y)$ est calculée en utilisant la complémentarité : $P(x_i|y) = 1 - P(\neg x_i|y)$. Les paramètres du modèle sont alors estimés en utilisant la formule régularisée du maximum de vraisemblance, similaire à celle du classificateur naïf bayésien multinomial.

En résumé, le classificateur naïf bayésien complémentaire utilise les probabilités complémentaires pour mieux gérer les jeux de données déséquilibrés. Les paramètres du modèle sont estimés en utilisant une formule régularisée du maximum de vraisemblance, similaire à celle du classificateur naïf bayésien multinomial.

Choix et vérification du modèle

Les caractéristiques de notre jeu de données sont des variables continues prenant des valeurs dans l'intervalle [0, 1]. Les classificateurs de type Bernoulli ou catégoriques, qui reposent sur des lois de distributions de Bernoulli ou catégoriques, respectivement, utilisent des distributions de probabilité discrètes et ne semblent donc pas adaptés à notre problème. Le classificateur naïf bayésien multinomial, ainsi que sa version complémentaire, sont adaptés aux caractéristiques discrètes telles que des comptages et supposent une distribution multinomiale des caractéristiques.

Par conséquent, le classificateur naïf bayésien gaussien semble être le choix optimal pour notre tâche de classification, étant donné le jeu de données utilisé. De plus, il est raisonnable de supposer que nos variables sont distribuées selon une loi normale, puisque les variables sont des résultats de votes qui devraient suivre une loi normale. Tous ces classificateurs ont été testés à l'aide de la bibliothèque Python scikit-learn [3]. Nous avons échantillonné 150 éléments de chaque classe et divisé le jeu de données en un sous-ensemble d'entraînement et un sous-ensemble de validation, en utilisant une proportion de 70/30. Pour l'entraînement, nous avons utilisé un normaliseur "min-max", notamment pour normaliser la colonne "NVOTE". Comme attendu, le classificateur gaussien s'est avéré le plus performant, avec une précision de 90%.

Table 3.1 – Matrice de confusion pour le classificateur gaussien

	Prédit "Spirale"	Prédit "Elliptique"	Prédit "Incertain"
Réel "Spirale"	27	0	1
Réel "Elliptique"	0	32	1
Réel "Incertain"	1	6	22

Matrice de confusion

Une matrice de confusion est un tableau utilisé pour évaluer la qualité d'un système de classification. Elle montre le nombre de prédictions correctes et incorrectes effectuées par le modèle par rapport à un ensemble de données de test, en comparant les prédictions avec les vraies étiquettes de classe.

Il convient également de noter que le classificateur multinomial atteint tout de même une précision de 87%. Les résultats de précision pour chaque classificateur sont résumés

dans le tableau suivant :

Classificateur	**Précision (%)**
Gaussien	90
Multinomial	87
Bernoulli	57
Complement	69

TABLE 3.2 – Tableau de précision des classificateurs naïfs bayésiens.

3.1.2 Les K plus proches voisins

La méthode des k plus proches voisins (K-NN) est un algorithme supervisé, faisant parti de la catégorie des classificateurs à voisinage. Il consiste à attribuer une étiquette de classe à un exemple non étiqueté en recherchant les k exemples les plus similaires dans l'ensemble d'apprentissage, appelés les k plus proches voisins, et en attribuant la classe la plus fréquente parmi ces k voisins à l'exemple non étiqueté. En d'autres termes, si $k = 5$, et que pour un point considéré, 4 des 5 voisins les plus proches de ce point représentent des galaxies spirales, alors l'algorithme déduira que le point considéré représente une galaxie spirale. À noter que cet algorithme peut également être utilisé pour la régression.

Pour déterminer la similarité entre deux exemples, l'algorithme utilise une métrique telle que la distance euclidienne ou la distance de Manhattan. Le choix de la mesure de distance peut dépendre du type de données. La distance euclidienne reste la métrique la plus utilisée.

De par son caractère non paramétrique, cet algorithme est souvent très efficace lorsqu'il s'agit de classifier des situations pour lesquelles la frontière de décision n'est pas nette. Les seuls paramètres que prend cet algorithme sont le nombre de voisins k et la métrique. Ces paramètres dépendent naturellement du jeu de données. On essaye souvent de choisir un nombre impair pour éviter les égalités de votes lors d'une classification binaire. Une valeur de k trop petite peut conduire à une sur-adaptation, tandis qu'une valeur de k trop grande peut entraîner une sous-adaptation. Lors de l'entraînement nous essaierons plusieurs valeurs de k afin de trouver celle qui donne la meilleure précision sur le jeu de données de validation.

Algorithmes

Bien que la méthode des K-NN reste la même, il existe différents algorithmes qui permettent de l'implémenter. Chacun dépend du jeu de données, et notamment du nombre N d'échantillons, et de la dimension D des caractéristiques. Pour un petit jeu de données,

on peut utiliser la méthode de la force brute, qui consiste à calculer les distances entre chaque échantillons, en D dimensions. Bien que plus rapide pour N petit, cet algorithme a une complexité qui croît en $\mathcal{O}(D.N)$. Il devient donc inutilisable pour de grands jeux de données.

Une solution à ce problème est de construire un arbre K-dimensionnel ; une généralisation en K-dimensions des arbres quaternaires et octaux en dimensions 2 et 3. Cette structure de données évite de calculer les distances entre tous les points. Si deux points ne sont pas sur le même noeud ou la même branche, leur distance n'est pas calculée. Une fois l'arbre construit, la complexité de l'algorithme croît en $\mathcal{O}(\log N)$. L'arbre K- dimensionnel est très efficace pour une dimension $D < 20$, mais souffre ensuite du fléau de la dimension. Une solution à ce problème est l'arbre à bille (ou arbre métrique) [4].

Sachant que notre jeu de données contient seulement 10 caractéristiques, la force brute ou l'arbre K-dimensionnel seront utilisés, selon le nombre d'échantillons sélectionnés en chargeant le jeu de données.

Entraînements et résultats

Lors de l'étape d'entraînement, nous avons utilisé la même configuration que dans l'exemple précédent, que nous avons conservée pour la suite de l'analyse. Bien que la normalisation des données n'ait pas eu d'effet sur les classificateurs naïfs bayésiens, elle a un impact significatif sur les performances de l'algorithme des K plus proches voisins. En effet, sans normalisation, la précision obtenue est de seulement 64%, tandis qu'elle atteint 84% avec une normalisation min-max. Contre notre attente, normaliser les données pour avoir des vecteurs unité donne une précision légèrement plus faible (83%).

La recherche par grille indique que le nombre de voisins donnant le meilleur résultat est 9, en utilisant une métrique dite de "Manhattan". Il est important de noter que ces résultats varient en fonction du nombre de points utilisés pour l'entraînement.

Finalement nous avons évalué les performances de l'algorithme K-NN sur un ensemble de données de test séparé. Nous avons obtenu une matrice de confusion qui montre les résultats de la classification pour chaque classe. La matrice de confusion est présentée sur la Fig. 3.1.

3.1.3 Machine à vecteurs de support

Les machines à vecteurs de support (SVM) sont une famille d'algorithmes d'apprentissage supervisé utilisée pour la classification et la régression. L'objectif des SVM est de trouver une frontière de décision optimale, appelée hyperplan, qui sépare les différentes classes d'échantillons de données en maximisant la marge, c'est-à-dire la distance entre les échantillons les plus proches de chaque classe (voir Fig. 3.2). Cette approche est particulièrement utile pour les jeux de données avec des caractéristiques complexes et non

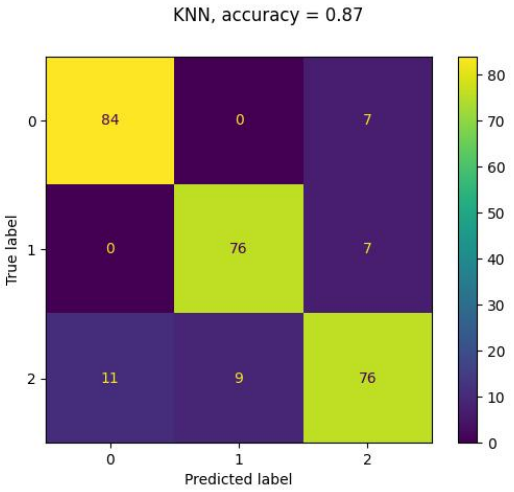

FIGURE 3.1 – Matrice de confusion pour l'algorithme K-NN. On remarque par exemple que l'algorithme a confondu 11 galaxies "incertaines" avec des galaxies spirales. À noter que le score apparaît plus élevé car plus de points ont été utilisés. Pour un très grand nombre de point la précision semble converger vers 87%.

linéaires, où une simple frontière de décision linéaire ne suffit pas à séparer efficacement les classes. Les SVM peuvent également être utilisées pour des jeux de données de petite et moyenne taille, mais ne sont pas recommandées pour des jeux de données très volumineux car elles peuvent être assez gourmandes en ressources.

Les SVM disposent de plusieurs paramètres importants qui doivent être ajustés pour obtenir les meilleures performances pour un certain jeu de données. Parmi ces paramètres, on peut citer le choix de la fonction noyau (par exemple, linéaire, polynomial ou fonction de base radiale (RBF)), la régularisation C et le gamma. La fonction noyau est utilisée pour transformer les données d'entrée en un espace de dimension supérieure où les classes sont plus facilement séparables. Les noyaux les plus couramment utilisés sont les noyaux linéaires, polynomiaux et radiaux [5].

Le paramètre de régularisation C contrôle le compromis entre la maximisation de la marge et la minimisation de l'erreur. Les valeurs de C plus élevées entraînent des marges plus étroites et une classification plus précise des points de données d'entraînement, mais peuvent également entraîner un sur-ajustement. Les valeurs de C plus faibles permettent des marges plus larges et une classification moins précise des points de données d'entraînement, mais peuvent également réduire le sur-ajustement.

Le paramètre gamma est le coefficient du noyau RBF. Ce paramètre affecte la forme des frontières de décision. Les valeurs de gamma plus élevées entraînent des frontières de décision plus complexes, qui peuvent conduire à un sur-ajustement. Les valeurs de gamma plus faibles entraînent des frontières de décision plus lisses, qui peuvent conduire

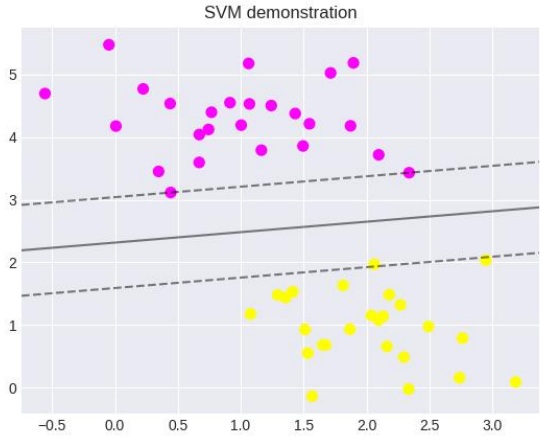

FIGURE 3.2 – Illustration des frontières crées par l'algorithme sur un jeu de données jouet en deux dimensions.

à un sous-ajustement.

Entraînements et résultats

Après avoir exploré les données à l'aide d'une PCA (voir section 3.2), il semble que les échantillons soient linéairement séparables. Ceci laisse suggérer que la méthode des SVM avec un noyau linéaire conviendrait bien pour résoudre notre problème de classification.

La recherche par grille indique effectivement que les paramètres produisant la meilleure précision sont un noyau linéaire avec une valeur de C égale à 1000. Cette configuration permet d'atteindre une précision de 92%, dépassant ainsi les classificateurs naïfs bayésiens et les k plus proches voisins.

3.1.4 Perceptron

Le perceptron est un algorithme de classification binaire linéaire qui crée un hyperplan séparateur pour séparer les données en deux classes. Il est basé sur une fonction d'activation en escalier qui prend la somme pondérée des entrées et retourne une sortie binaire. Le modèle est entraîné en ajustant les poids des entrées pour minimiser l'erreur de classification. Le perceptron est le plus efficace lorsque les données sont linéairement séparables, c'est-à-dire lorsque les deux classes peuvent être séparées par un hyperplan. Il a été l'un des premiers algorithmes d'apprentissage automatique à être développé [6] et est souvent utilisé comme base pour des modèles plus complexes comme les réseaux de neurones. Il est également utilisé pour la classification en temps réel et l'apprentissage

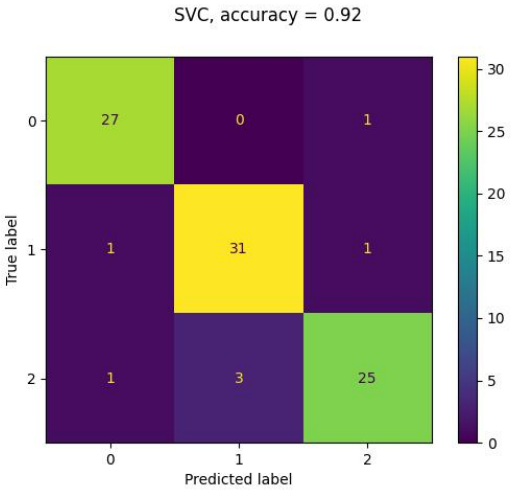

FIGURE 3.3 – Matrice de confusion pour l'algorithme SVM. L'algorithme fait peu de confusions entre les différentes classes.

en ligne. Le perceptron est à la base un algorithme binaire, donc il est plus naturellement utilisé pour des problèmes de classification binaire. Cependant, il est possible de l'étendre à plusieurs classes en utilisant des techniques de "one-vs-all" ou "one-vs-one". Dans le premier cas, on entraine K perceptrons binaires, chacun distinguant une classe des autres, et dans le deuxième cas, on entraine $K(K-1)/2$ perceptrons binaires, chacun distinguant deux classes à la fois.

Entraînement et résultats

Les paramètres que l'on peut passer au perceptron concernent principalement les méthodes de pénalisation. Les plus couramment utilisées pour le perceptron sont la pénalité L1 et la pénalité L2. La pénalité L1 ajoute une contrainte à la fonction de coût qui oblige les coefficients à être proches de zéro. Cela permet une sélection de fonctionnalités automatique, car les coefficients correspondant aux fonctionnalités moins importantes sont réduits à zéro. La pénalité L2, elle, ajoute une contrainte qui oblige les coefficients à être petits et proches de zéro, mais sans les réduire exactement à zéro. Cela permet de réduire le sur-ajustement en réduisant les coefficients de fonctionnalités qui ont une faible contribution à la prédiction. Il existe également d'autres méthodes de pénalisation pour le perceptron, comme la pénalité "ElasticNet" qui combine la pénalité L1 et L2.

La recherche par grille nous indique que la pénalité L1 donne de meilleurs résultats, avec un taux d'apprentissage de 10^{-4}. La précision atteinte par le perceptron est seulement de 74%, confirmant que cet estimateur n'était pas le plus adapté à notre problème.

3.1.5 Arbre décisionnel

En tant que méthode de classification et de régression non paramétrique, l'arbre de décision [7] s'illustre comme un outil incontournable de l'apprentissage supervisé. Son but est de créer un modèle capable de prédire la valeur d'une variable cible en apprenant des règles de décision simples à partir des caractéristiques des données. Il convient de noter qu'un arbre peut être perçu comme une approximation constante par morceaux. Plus l'arbre est profond, plus les règles de décision sont complexes et plus le modèle s'ajuste aux données. Les avantages de cette méthode sont nombreux : sa simplicité de compréhension et d'interprétation, sa visualisation aisée, sa faible exigence en termes de préparation des données, son coût logarithmique $\mathcal{O}(\log N)$ pour la prédiction des données, sa capacité à gérer à la fois les données numériques et catégorielles, ainsi que les problèmes à sorties multiples. Cependant, l'arbre de décision peut présenter certains inconvénients, tels que la création d'arbres trop complexes qui ne généralisent pas bien, l'instabilité ou encore la création d'arbres biaisés si une classe domine les données.

L'arbre de décision est un choix populaire lorsque les données sont complexes et hétérogènes, ce qui en fait un choix adapté pour les jeux de données astronomiques. Cependant, il est important de noter que l'arbre de décision est plus adapté pour des ensembles de données plus petits. Étant donné que le jeu de données que nous étudions provient d'un arbre décisionnel (voir Fig. 2.1), il est naturel de considérer l'utilisation d'un arbre décisionnel comme une bonne option pour la classification.

Entraînement et résultats

En ce qui concerne les paramètres, ils incluent la profondeur de l'arbre, le nombre minimum d'exemples requis pour scinder un nœud, le nombre minimum d'exemples requis pour un nœud de feuille, le critère de mesure de qualité de la scission (gini, entropie ou erreur de classification) et le seuil de la mesure de qualité. Les paramètres spécifiques à la régression incluent la perte de la fonction d'impureté (erreur quadratique moyenne ou erreur absolue moyenne).

La recherche de grille sur les paramètres de l'arbre décisionnel pour la classification des galaxies a permis d'identifier les valeurs optimales du critère de scission et de profondeur maximale de l'arbre, respectivement égales à l'entropie et à 18. Les résultats obtenus à partir de ces paramètres sont satisfaisants, avec une précision maximale de 86%. Le résultat est satisfaisant bien qu'en dessous de ce que nous pouvions espérer.

Le fait que les noeuds soient crées de manière aléatoire rend la recherche des paramètres optimaux quelque peu compliqué, et ceux trouvés ici pourraient ne pas être ceux donnant le meilleur résultat possible. L'arbre donnant les meilleurs résultats est présenté sur la Fig. 3.4. Il est intéressant de constater que la décision se fait tout d'abord sur la colonne "P_CS_DEBIASED" , et que l'arbre sépare assez distinctement dès le premier

noeud les galaxies spirales et elliptiques.

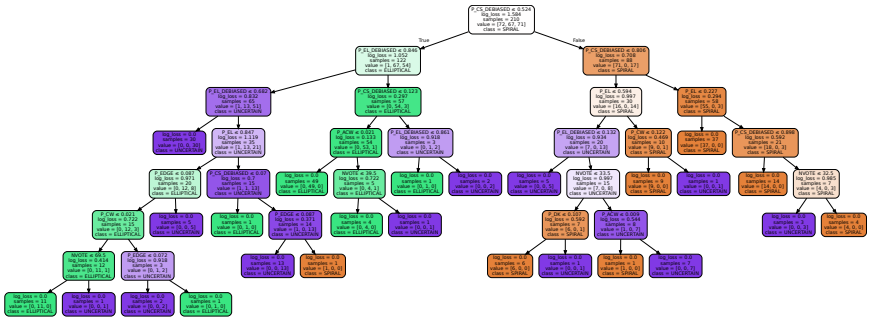

FIGURE 3.4 – Structure interne de l'arbre de décision crée par l'algorithme. Pour une meilleure visibilité : https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/plots/arbre_decision.pdf.

3.1.6 Méthodes d'ensembles

Bien que l'arbre décisionnel soit un outil très puissant, il peut parfois présenter des limites. En particulier, si l'on utilise un seul arbre de décision pour classifier des données, il est possible que celui-ci soit biaisé ou ne généralise pas bien les données, menant à un sur- apprentissage. C'est pour cela que l'on peut avoir intérêt à utiliser des méthodes d'ensemble qui combinent plusieurs modèles d'arbres de décision pour obtenir une classification plus robuste et plus précise. Les méthodes d'ensemble les plus courantes sont les forêts d'arbres décisionnels [8] et le AdaBoost [9]. Dans les deux cas, on combine plusieurs modèles d'arbres de décision (ou perceptron) pour obtenir une classification plus précise.

Les forêts d'arbres décisionnels sont constituées de plusieurs arbres de décision qui sont construits de manière aléatoire. Les paramètres importants sont le nombre d'arbres et la profondeur de chaque arbre. En général, plus il y a d'arbres dans la forêt, plus la classification est précise. Cependant, cela augmente également le temps de calcul. Dans notre cas, les forêts d'arbres décisionnels peuvent être particulièrement efficaces car elles permettent de gérer des données complexes avec plusieurs caractéristiques et classes. Le AdaBoost, quant à lui, consiste à entraîner des modèles d'arbres de décision itérativement. À chaque itération, le modèle se concentre sur les échantillons mal classés de l'itération précédente. Les paramètres importants sont le nombre d'itérations et la profondeur de chaque arbre. Le AdaBoost peut être très efficace pour des jeux de données difficiles, en particulier lorsque les classes sont déséquilibrées. Cependant, il peut également être sensible au bruit dans les données.

En somme, les méthodes d'ensemble sont une approche efficace pour la classifica-

tion de données complexes. Dans notre cas, les forêts d'arbres décisionnels peuvent être particulièrement utiles pour classifier des données de galaxies avec de nombreuses caractéristiques. Le AdaBoost peut également être efficace si les classes sont déséquilibrées.

Entraînement et résultats

Les résultats de la recherche de grille montrent que pour la forêt aléatoire et le AdaBoost, le nombre d'estimateurs optimal est respectivement de 20 et 250. Pour la forêt aléatoire, le critère de qualité utilisé pour diviser les nœuds de l'arbre est l'entropie, et la profondeur maximale est de 34. Pour le AdaBoost, l'estimateur optimal est un arbre de décision avec pour critère de qualité l'entropie, une profondeur maximale de 3, et un taux d'apprentissage de 0.1.

La précision obtenue pour la forêt aléatoire est de 93% et de 81% pour le AdaBoost. Ces résultats montrent que les méthodes d'ensemble peuvent être très efficaces pour la classification de galaxies. Notamment, cela confirme la redoutable efficacité des forêts aléatoires.

3.1.7 Le perceptron multicouche

Le perceptron multicouche (MLP) [10] est un type de réseau de neurones artificiels (voir 3.5) qui peut être utilisé pour la classification et la régression. Il est constitué de plusieurs couches de neurones, où chaque couche est connectée à la précédente et à la suivante. Les couches intermédiaires sont appelées couches cachées, car leurs sorties ne sont pas directement observées. Le MLP est entraîné en minimisant une fonction de coût, qui mesure la différence entre les prédictions du modèle et les valeurs réelles. Cette fonction de coût est minimisée en utilisant une technique d'optimisation telle que la descente de gradient. Pour implémenter le MLP nous avons besoin de spécifier plusieurs hyperparamètres, tels que le nombre de couches cachées, le nombre de neurones dans chaque couche, la fonction d'activation utilisée pour calculer les sorties des neurones, la fonction de coût utilisée pour entraîner le modèle, ainsi que les hyperparamètres spécifiques à l'optimisation, tels que le taux d'apprentissage.

Détails mathématiques

Un réseau neuronal est une fonction mathématique qui peut être représentée sous la forme d'un graphe de neurones interconnectés. Mathématiquement, cette fonction peut être décrite comme une série de fonctions non-linéaires imbriquées les unes dans les autres. La sortie de chaque fonction alimente l'entrée de la fonction suivante, et ainsi de suite, jusqu'à ce que la sortie finale du réseau soit produite :

$$\hat{y}(x) = w_L^T \phi_L \left(w_{L-1}^T \phi_{L-1} \left(\cdots \phi_2 \left(w_2^T \phi_1 \left(w_1^T x + b_1 \right) + b_2 \right) \cdots \right) + b_{L-1} \right) + b_L \tag{3.4}$$

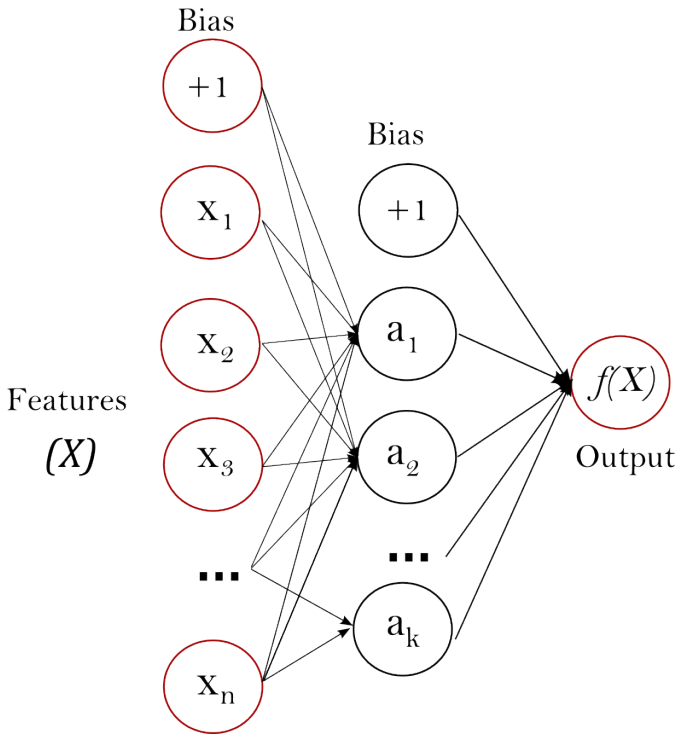

FIGURE 3.5 – Structure d'un perceptron avec une couche cachée. Référence : https://scikit-learn.org/stable/modules/neural_networks_supervised.html

où $\mathbf{x}$ est l'entrée, $\hat{y}(\mathbf{x})$ est la sortie prédite, $\mathbf{w}_l$ et $\mathbf{b}_l$ sont les poids et les biais de la couche l, respectivement, et $\phi_l(\cdot)$ est la fonction d'activation de la couche l. En d'autres termes, chaque couche cachée est calculée en appliquant une transformation linéaire $\mathbf{w}_l^T \cdot$ à la sortie de la couche précédente, suivie d'une fonction d'activation non-linéaire $\phi_l(\cdot)$. La sortie finale est calculée en appliquant une dernière transformation linéaire $\mathbf{w}_L^T \cdot$ à la sortie de la dernière couche cachée, suivie d'un biais $\mathbf{b}_L$.

Cela permet au MLP d'apprendre des représentations hiérarchiques des données d'entrée, où chaque couche cachée extrait des caractéristiques de plus en plus abstraites des données en entrée.

Typiquement, les fonctions d'activation sont des fonctions non linéaires telles que la fonction sigmoïde ou la fonction ReLU (Unité Linéaire Rectifiée). Ces fonctions sont importantes car elles permettent au réseau de modéliser des relations non linéaires entre les entrées et les sorties.

Enfin, pour entraîner un réseau neuronal, on utilise un algorithme de rétropropagation du gradient [11] pour ajuster les poids et les biais du réseau de manière à minimiser une fonction de coût. Cette fonction de coût mesure l'écart entre les sorties réelles du réseau et les sorties attendues, et l'algorithme de rétropropagation du gradient utilise cette me-

sure pour ajuster les poids et les biais de manière itérative jusqu'à ce que l'erreur soit minimisée.

Entraînement et résultats

En utilisant la recherche par grille, nous avons identifié les paramètres optimaux pour notre modèle, à savoir : une fonction d'activation tangente hyperbolique et une architecture comprenant deux couches cachées et, 50 neurones par couche. Pour la fonction de coût, nous avons utilisée la fonction d'entropie croisée, qui est couramment utilisée pour les tâches de classification. Notre modèle obtient un score de précision de 91%.

3.2 Apprentissage non-supervisé

L'apprentissage non supervisé est une approche de l'apprentissage automatique qui vise à découvrir des structures cachées dans les données sans utiliser d'étiquettes ou de réponses préalables. Cette approche est particulièrement utile lorsque nous ne savons pas exactement quelles caractéristiques des données sont pertinentes ou intéressantes.

L'une des méthodes les plus couramment utilisées pour l'apprentissage non supervisé est la réduction de dimension, qui permet de projeter des données à haute dimension dans un espace de dimension inférieure tout en préservant les caractéristiques les plus importantes. L'objectif de cette méthode est de trouver des combinaisons linéaires de variables qui réduisent la dimensionnalité des données tout en préservant leur variabilité. Un algorithme populaire en apprentissage non supervisé est l'analyse en composantes principales (PCA), qui consiste à trouver les axes de variabilité maximale dans les données et à projeter les données sur ces axes pour réduire la dimensionnalité. Enfin, les algorithmes de clustering sont également largement utilisés en apprentissage non supervisé. Les algorithmes de clustering cherchent à regrouper des données similaires en groupes distincts, appelés clusters (agrégat en bon français). Les méthodes de clustering les plus courantes sont le clustering hiérarchique et le K-means. Le clustering hiérarchique regroupe les données en utilisant une approche ascendante ou descendante, tandis que le K-means partitionne les données en K clusters en minimisant la distance entre chaque point et son centroïde de cluster correspondant. En résumé, l'apprentissage non supervisé est une approche de l'apprentissage automatique qui permet de découvrir des structures cachées dans les données sans utiliser d'étiquettes ou de réponses préalables.

3.2.1 Analyse en composantes principales

Nous nous intéressons ici uniquement à la méthode d'analyse en composantes principales (PCA), car elle nous permet de mieux comprendre notre jeu de données et d'analyser les relations internes entre les différentes variables. Comme décrit, cette technique

nous permet de réduire la complexité de nos données en les projetant sur un nouvel espace de dimension inférieure, tout en préservant au maximum l'information contenue dans les données initiales. Les composantes principales sont déterminées en diagonalisant la matrice de covariance des données. De cette façon, nous pouvons identifier les variables les plus importantes et comprendre comment elles contribuent aux différentes dimensions principales.

Résultats

Nous avons effectué deux analyses en composantes principales (PCA) sur notre jeu de données de galaxies, en conservant ou en excluant les données labellisées « incertaines ». En utilisant la PCA, nous avons réduit notre jeu de données de 10 dimensions à 2 dimensions, ce qui nous a permis de visualiser les données sur un graphique en deux dimensions en fonction de leurs composantes principales.

Les résultats de la PCA ont montré que notre jeu de données de galaxies est linéairement séparable en deux dimensions, notamment si l'on exclue les données classifiées incertaines (voir Fig. 3.6). Plus précisément, nous observons que les deux premières composantes principales expliquent une grande partie de la variance des données (voir Fig. 3.7).

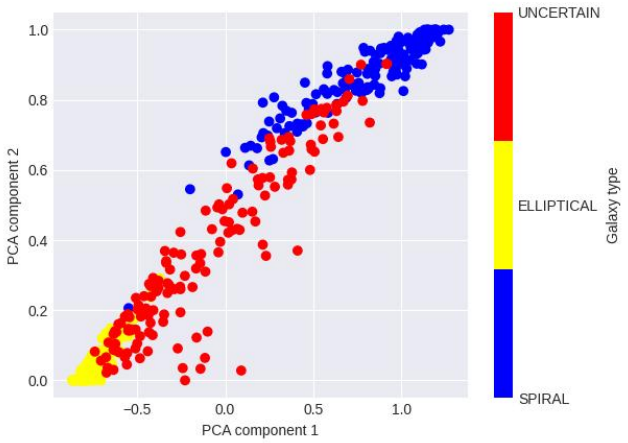

FIGURE 3.6 – Les points sont distribués dans le plan formé par les composantes principales 1 et 2. Celles-ci sont données par les combinaisons linéaires suivantes : CP1 = (-0.028 x NVOTE - 0.440 x P_EL + 0.145 x P_CW + 0.096 x P_ACW + 0.215 x P_EDGE - 0.016 x P_DK - 0.002 x P_MG + 0.457 x P_CS - 0.503 x P_EL_DEBIASED + 0.517 x P_CS_DEBIASED) et CP2 = (0.396 x NVOTE - 0.094 x P_EL + 0.675 x P_CW + 0.013 x P_ACW - 0.605 x P_EDGE - 0.018 x P_DK + 0.044 x P_MG + 0.083 x P_CS + 0.036 x P_EL_DEBIASED - 0.037 x P_CS_DEBIASED)

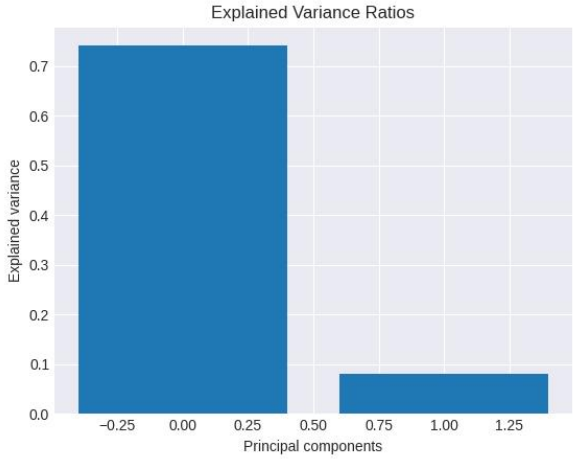

FIGURE 3.7 – Ratios d'explication de la variance. Sur la gauche la première composante, sur la droite la seconde. La première composante contribue à plus de 74% à la variance, contre seulement 8% pour la seconde.

Nous avons également observé que la première composante principale est fortement corrélée aux caractéristiques liées à la forme des galaxies voir (Fig. 3.8), tandis que la deuxième composante principale est fortement corrélée aux caractéristiques liées à la couleur des galaxies.

En utilisant la PCA, nous avons pu mieux cerner la structure interne de notre jeu de données et identifier les caractéristiques les plus importantes pour la séparation des variables. Nous aurions pu commencer notre analyse en utilisant la PCA pour constater cette séparation linéaire (dans un autre espace) et ainsi choisir un classificateur adapté en conséquence.

3.3 Apprentissage profond

Les données de Galaxy Zoo utilisées jusqu'ici sont basées sur les probabilités de classification attribuées par un grand nombre de participants, plutôt que sur des mesures directes des caractéristiques physiques des galaxies. Nous avons créé des classificateurs basés sur les données numériques qui pourraient éventuellement déterminer les classes des galaxies toujours classifiées comme "incertaines". Bien que ces probabilités soient utiles pour prédire le type d'une galaxie, elles demandent un traitement humain conséquent en amont. Nous pourrions accélérer le processus de classification en utilisant un modèle qui permettrait de classifier les galaxies en utilisant directement les images du télescope.

Nous implémenterons ici un réseau de neurones convolutionnel (CNN) pour classer les images des galaxies. Le CNN est un choix populaire pour la classification d'images

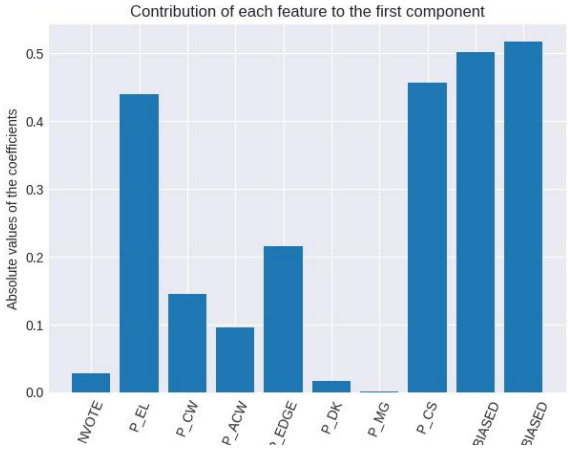

FIGURE 3.8 – Valeurs absolues des coefficients de la première composante de la PCA. On observe que similairement à l'arbre décisionnel, la caractéristique "P_CS_DEBIASED" est prépondérante.

en raison de sa capacité à extraire automatiquement les caractéristiques importantes des images en utilisant des filtres de convolution [12].

3.3.1 Réseau de neurones convolutionnel

Le fonctionnement d'un CNN est basé sur plusieurs couches de traitement d'informations. La première couche traite l'image brute et applique des filtres de convolution pour extraire des caractéristiques visuelles telles que les bords, les coins, les textures, etc. La deuxième couche applique de nouveaux filtres pour extraire des caractéristiques plus complexes à partir des résultats de la première couche. Les couches suivantes appliquent des filtres encore plus complexes pour extraire des caractéristiques plus abstraites et hiérarchisées. Après que les caractéristiques aient été extraites, les résultats sont aplatis en une seule dimension et envoyés à travers une ou plusieurs couches de neurones denses pour la classification.

3.3.2 Entraînement du réseau

Le modèle se compose de deux couches de convolution, chacune suivie d'une couche de pooling pour réduire la taille de la représentation de l'image. La première couche de convolution a 32 filtres de taille 3x3, tandis que la deuxième couche a 64 filtres de la même taille. Les couches de pooling utilisent une fenêtre de 2x2. Après les couches de convolution et de pooling, l'image est aplatie dans un vecteur unidimensionnel, qui est ensuite connecté à une couche dense de sortie avec une fonction d'activation « softmax » pour

classer l'image comme une galaxie spirale ou elliptique. Le modèle a été entraîné sur un ensemble de données contenant 10 000 images de galaxies spirales et 10 000 images de galaxies elliptiques. Le jeu de données a été divisé en ensembles d'entraînement et de validation, avec un ratio de 80/20. Le modèle a été entraîné avec une méthode de descente de gradient stochastique (SGD) avec un taux d'apprentissage de 0,01 et une taille de lot de 32 images. La précision du modèle a ensuite été évaluée sur un ensemble de test comptant 5000 images de chaque type de galaxie. La précision atteinte est de 97,6%. Le code peut être trouvé à l'adresse suivante : https://github.com/lukbrb/academic-physics/blob/master/galaxy-classification/notebooks/03-Deep-learning.ipynb.

4 Conclusion

Dans cette étude, nous avons comparé plusieurs classificateurs (voir 4.1) pour prédire la nature de galaxies à partir de données sur leur forme, ou dans le cas du CNN à partir des images traitées d'un télescope.

Classificateur	**Précision**
Perceptron	74%
K plus proches voisins	84%
Naïf Bayes	90%
SVM	92%
Arbre de décision	86%
Forêt d'arbres aléatoires	93%
AdaBoost	81%
Perceptron multicouche	91%
Réseau de neurones convolutif	98%

TABLE 4.1 – Résumé des résultats des différents classificateurs.

Les résultats montrent que la précision varie significativement selon les classificateurs. La précision la plus élevée a été obtenue avec le réseau convolutionnel, entraîné sur des images de galaxies spirales et elliptiques, avec un taux de précision de 98%. Parmi les autres classificateurs, le SVM et la forêt d'arbres aléatoires ont également montré des performances remarquables, avec des taux de précision de 92% et 93% respectivement.

Cependant, il est important de noter que certains estimateurs nécessitent plus ou

moins de données pour atteindre leur performance optimale. Par exemple, le MLP a été entraîné avec plus de données que les autres estimateurs. Nous avons également remarqué que les classificateurs ont des temps de calculs différents, ce qui peut être important dans certaines applications pratiques. De plus, le CNN a été entraîné en utilisant uniquement des images de galaxies spirales ou elliptiques. La PCA ou le partitionnement en k-moyennes nous a permis de voir que la frontière entre les données de galaxies elliptiques et spirales est très claire, mais que les points correspondant aux galaxies "incertaines" sont éparpillés entre les deux. Ainsi, en excluant les galaxies incertaines lors de l'essai du CNN, nous avons largement facilité sa tâche, et donc la précision finale.

Il serait intéressant de reproduire cette analyse en excluant les données incertaines de l'entraînement des tous les classificateurs. On pourrait ensuite utiliser notre meilleur classificateur pour déterminer la vraie nature des galaxies classées "incertaines". Alternativement, on pourrait utiliser notre meilleur classificateur actuel et utiliser les probabilités de classification donnée par l'estimateur. Enfin, pour une analyse tabulaire plus poussée, le jeu de données de Hart et al. [13] est plus précis et détaillé. Pour un tel jeu de données, comprenant davantage de caractéristiques (environ 230), les techniques de PCA évoquées dans la section 3.2 peuvent se révéler indispensables.

Bibliographie

[1] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, *et al.*, "Galaxy zoo 2 : detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 435, no. 4, pp. 2835–2860, 2013.

[2] H. Zhang, "Exploring conditions for the optimality of naive bayes," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 02, pp. 183–198, 2005.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn : Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[5] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[6] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[7] F. Breiman, "Olshen, and stone," *Classification and Regression trees*, 1984.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[9] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[10] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[11] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks : Tricks of the trade*, pp. 9–50, Springer, 2002.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] R. E. Hart, S. P. Bamford, K. W. Willett, K. L. Masters, C. Cardamone, C. J. Lintott, R. J. Mackay, R. C. Nichol, C. K. Rosslowe, B. D. Simmons, *et al.*, "Galaxy zoo : comparing the demographics of spiral arm number and a new method for correcting redshift bias," *Monthly Notices of the Royal Astronomical Society*, vol. 461, no. 4, pp. 3663–3682, 2016.