Génie Logicielle et Intégration d'Application (GLIA)

Rapport de stage

Analyse des données astronomiques « Estimation du redshift et la catégorisation des supernovas par données photométriques »

Auteur :
Ousmane Issa

Encadreur :
Pr. Engelbert Mephu Nguifo
Tuteur pédagogique :
Dr. KERIVIN Hervé

2017-2018

Résumé

Pendant ce stage nous essayons de répondre à deux problématiques dans le domaine de l'astronomie sans utilisation des expérimentations couteuse du domaine physique (spectroscopie). Dans un premier temps nous tentons de réduire l'erreur d'estimation du redshift à partir des images du ciel prises directement. En effet l'estimation précise de ce composant physique sans expérimentations physiques, qui nous permet de connaitre beaucoup des paramètres cosmologiques tels que la distance de galaxies par rapport à la terre ou leurs vitesses de mouvement, reste toujours une grande question ou un défis à relever. Beaucoup d'approches ont été étudiés [13, 10, 5, 8, 23], nous essayons une approche dans laquelle nous estimons le redshift à partir des images directes du ciel en utilisant les techniques de l'apprentissage profond (ce choix est fait sur la base du succès de ces techniques en vision artificielle [35] et en détection des motives [29] ces dernières années) et quelques algorithmes de l'apprentissage automatique. Dans un second temps, nous participons à une compétition de classification des supernovas. Dans ce second travail nos données ne sont pas de nature figée comme les images mais des données sous forme de séries qui capturent la quantité lumineuse dans le temps (courbe lumineuse) lors de l'explosion d'une étoile (supernova). Nous appliquons un ensemble de techniques de la fouille de données pour les séries temporelles afin d'essayer de minimiser les faux catégorisés et obtenir une meilleure fonction de classification de supernovas.

Abstract

Remerciements

Table des matières

Table des figures

Liste des tableaux

Introduction

Ces dernières années plusieurs travaux ont essayé d'estimer le redshift en minimisant l'erreur d'estimation à travers différents ensembles de données, parmi ces travaux on peut citer [4, 34, 13, 10]. L'obtention du redshift à partir des expérimentations physique (spectroscopie) est la méthode la plus exacte pour estimer le redshift mais couteuse et consommateur de temps. Pendant plus dix sept ans, SDSS [7] a collectée plus de trois millions des images d'objets astronomiques dont un sous ensemble a également les redshift correspondants qui sont de la spectroscopie. SDSS opére dans cinq filtres différentes et a survolé un tiers du ciel en utilisant un télescope optique de 2.5 mètres de diamètre. Toutes ces données photométriques dont une partie est déjà étiquetée par son redshift, ouvre un grand horizon à l'analyse, à l'apprentissage supervisé et non supervisé pour mettre au point une fonction partant de l'espace de l'ensemble des ces données à l'ensemble de l'espace du redshift.

La précision sur l'estimation de l'allongement de la longueur d'onde (redshift) des corps cosmologiques est une condition nécessaire pour la réussite des missions astronomiques comme par exemple Euclid mission [1]. Ce composant physique nous permet non seulement de connaitre la vitesse de mouvement des corps célestes mais également la distance les séparant de la terre. L'obtention d'une valeur précise du redshift des corps célestes très éloignes (quazars) de nous, reste toujours un défit pour les scientifiques, en effet on reçoit moins de lumière de ces corps, ce qui rend la précision de leurs estimations très difficile. Dans le sens toujours de capturer profondément la lumière de corps eloignés, le projet du large télescope LSST est né. LSST opère dans 6 bandes de longueurs d'ondes différentes, son objectif principale est de faire 10 ans de survol du ciel pour générer jusqu'à 200 petabyte des images [2]. Ce qui va permettre aux astronautes et data scientistes une opportunité d'analyse profonde mais également c'est la naissance d'un nouveau chalenge d'analyse.

L'apprentissage profond a fait ses preuves surtout dans la vision artificielle[42, 17] ces quatre dernières années mais aussi dans la détection des motifs et la segmentions des images [29]. Le réseau convolutif a été testé également sur les images de la SDSS DR12 en le couplant avec la densité gaussienne mélangée pour estimer le redshift [10], et les auteurs ont obtenu des bons résultats. Nous

1. http ://sci.esa.int/euclid/
2. https ://www.lsst.org/

allons travailler avec les images de la version 13 de la base DR de SDSS dans une configuration différente des images du papier [10], en utilisant un réseau convolutif sans le coupler avec la densité gaussienne mélangée, mais d'autres algorithmes de l'apprentissage automatiques seront testés également.

Dans ce travail, nous essayons de répondre dans un premier temps à un problème d'estimation au quel beaucoup d'astronomes, data scientistes et statisticiens ont essayé de répondre : "Estimer l'augmentation de la longueur d'onde (redshift) de corps cosmologiques due à l'expansion de l'univers à partir des données photométriques". Dans le second temps, nous essayons de trouver un modèle ou un algorithme permettant de faire la classification des supernovas à partir des données photométriques en série temporelle provenant du LSST.

Ce papier est organisé comme suit :

* Une section dédiée à la présentation du laboratoire LIMOS au sein duquel j'ai effectué ce stage.
* Une première partie consacrée à l'estimation du redshift.
  - Le chapitre 1 est consacré à l'état de l'art concernant l'estimation du redshift.
  - Le chapitre 2 est consacré aux différents modèles utilisés.
  - Le chapitre 3 présente les résultats obtenus.
  - cette partie se termine par une conclusion
* Une seconde partie concernant la classification des supernovas. Après une présentation du contexte, nous continuons par les chapitres restants.
  - Le chapitre 4 est consacré aux données et les différents algorithmes d'analyse utilisés. Dans ce chapitre nous présentations également nos expérimentations et résultats.
  - La partie se termine par une conclusion.
* Le papier se termine par une conclusion et perspectives

Présentation du laboratoire LIMOS

Le Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) est une Unité Mixte de Recherche (UMR 6158) en informatique, et plus généralement en Sciences et Technologies de l'Information et de la Communication (STIC).

Le LIMOS est principalement rattachée à l'Institut des Sciences de l'Information et de leurs Interactions (INS2I) du CNRS, et de façon secondaire à l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS). Il a pour tutelles académiques l'Université l'université Clermont Auvergne et l'Ecole Nationale Supérieure des Mines de Saint-Etienne (EMSE), et comme établissement partenaire l'Institut Français de Mécanique Avancée (IFMA). Le LIMOS est membre des labex IMOBS3 et ClercVolc et de la fédération de recherche en Environnement FR 3467 (qui regroupe 17 laboratoires). Il est membre associé de la fédération MODMAD (MODélisation Mathématique et Aide à la Décision, FED 4169) portée par Université Jean Monnet de Saint-Etienne.

Ses principaux thèmes de recherche sont :

* Optimisation combinatoire et continue
* Recherche opérationnelle, Systèmes de production ; Logistique
* Algorithmique des graphes et des treillis
* Images et apprentissage
* Modélisation et simulation
* Grandes masses de données ; Fouille de données ; Interopérabilité des systèmes d'information
* Réseaux de Capteurs, confiance numérique

Pendant ce stage, j'étais intégré à un sous groupe appelé **miners** dont son thème de recherche est la fouille de données.

Première partie

Estimation du redshift photométrique par apprentissage profond et à partir des images

Dans cette première partie du mémoire, nous allons présenter le travail que nous avons fait sur l'estimation du redshift par les techniques de l'apprentissage automatique et plus précisément l'apprentissage profond et le boosting. Avant de présenter l'état de l'art et les algorithmes utilisés, nous présentons d'abord le contexte du travail et définir quelques termes.

Décalage vers le rouge (Redshift)

Le redshift est l'augmentation de la longueur d'onde d'un objet lors de l'application de radiation électromagnétique sur cet objet. Un observateur peut remarquer le déplacement de la lumière de cet objet vers le rouge, ce qui est un effet contraire au blueshift (décalage vers le bleu) où l'observateur remarque le déplacement de la lumière de l'objet vers le bleu.

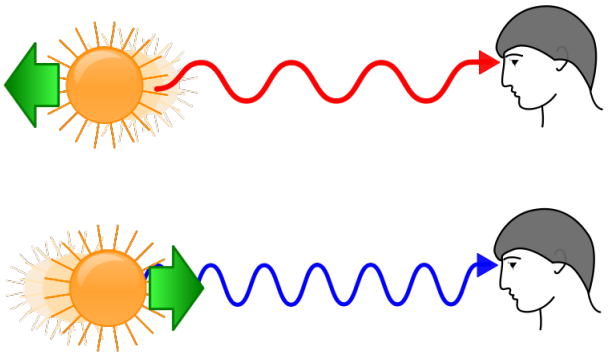

FIGURE 1 – illustration du redshift et blueshift

Le reshift est noté z et se manifeste quand l'objet en mouvement tout en s'éloignant de l'observateur (effet dopler) ou il est du à l'expansion de l'univers (redshift cosmologique) ou due à la gravitation (redshift gravitationel). Dans notre cas le redshit qu'on essaie d'estimer correspond au redshift cosmologique et il s'obtient de la manière suivante : $1 + z = \frac{\lambda_{observ}}{\lambda_{emis}}$ où λ_{observ} et λ_{observ} sont respectivement la longueur d'onde observée et celle émise. La loi de Huble établit une relation entre la distance et le redshift cosmologique comme suit $v = H_0 d = cz$ où d est la distance, v la vitesse, c la vitesse de la lumière, H_0 la constance de Huble et z le redshift cosmologique.

Spectroscopique

La méthode spectroscopique mesure le redshift par la dispersion de la lumière en comparant le décalage entre les lignes spectrales des éléments chimiques émises et celles absorbées. La spectroscopie donne des estimations précises du redshift, néanmoins elle est couteuse et prend beaucoup de temps. Ce qui encourage les scientifiques à chercher d'autres moyens moins couteux pour estimer le redshift.

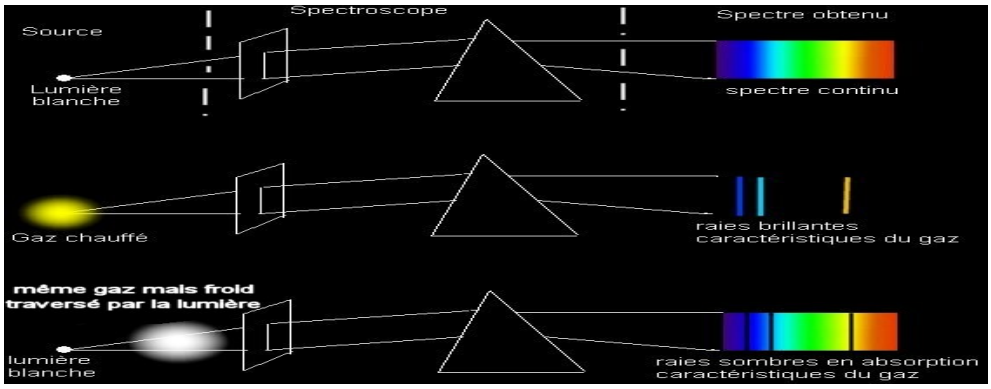

FIGURE 2 – Expériences spectroscopiques

Estimation par photométrie

La méthode basée sur la photométrie utilise un ensemble de filtres profonds dans des longueurs d'ondes spécifiques (5 bandes dans notre cas) pour capturer certaines caractéristiques de l'objet concerné. Puisque c'est une opération rapide, la photométrie donne une grande quantité d'observations en terme de nombre et de profondeurs de caractéristiques de chacune des observations. À partir des caractéristiques de l'objet observé, on tente de déterminer son décalage vers le rouge. Donc la précision du calcul du redshift par données photométries dépend de la profondeur du mesure de ces caractéristiques. SDSS a collecté plus de 300 millions des objets photométries. Le télescope LSST (Large Scale Survey Telescope) qui sera bientôt déployé donne plus de caractéristiques que ceux existants (utilise 6 bandes allant de l'ultra violet à l'infra-rouge), qui va rendre plus précis l'estimation du redshift par photométrie. Dans ce travail nous essayons d'estimer directement le redshift à partir des images.

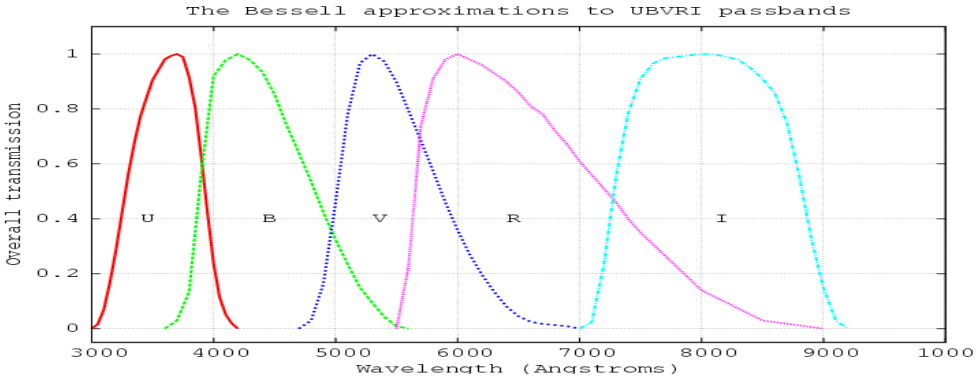

FIGURE 3 – Expériences photométries

Ces derniers années plusieurs travaux [16, 13, 10, 23, 34] ont été fait à partir des données photométries pour l'estimation du redshift. La méthode d'estimation du redshift par données photométries est divisée en deux sous méthodes à savoir celle basée modèle et celle basée sur les techniques de l'apprentissage automatique et statistiques dite empirique. Ces deux sous-méthodes seront détaillées dans le chapitre 1.

Qualité de prédiction du redshift

Il existe plusieurs paramètres pour définir la qualité de prédiction du redshift, ils sont tous basé sur une fonction de perte normalisée (erreur normalisée). L'objectif de ce travail est de trouver une fonction (f) qui prend en entrée des images superposées dans 5 bandes de longueurs d'onde différentes d'un corps cosmologique et retourne le redshift correspondant. Pour cela une fonction (Δz) dite fonction d'erreur est définie, elle permet de vérifier si la fonction f prédit bien en générale le redshift d'un corps en faisant la soustraction entre la valeur exacte du redshift (valeur spectroscopique) et celle prédite(estimée) $\Delta z = z_{phot} - z_{spec}$.
L'erreur normalisée est définie comme suit :
$\Delta z_{norm} = \frac{\Delta z}{z_{spec}+1}$
Dans ce travail nous utilisons principalement quatre paramètres de mesure de qualité de prédiction du redshift :

- Le biais qui est la moyenne de l'ensemble des erreurs notée $\mu(\Delta z_{norm})$ pour l'erreur normalisée.
  $\mu(\Delta z_{norm}) = \frac{\sum_{i=1}^{n} \Delta z_{norm_i}}{n}$
- Le standard déviation noté $\sigma(\Delta z_{norm}) = \sqrt{\frac{\sum_{i=1}^{n} (\Delta z_{norm_i} - \mu(\Delta z_{norm}))^2}{n}}$
- La proportionnalité des outliers/particuliers catastrophiques mesurée par $(\frac{card(|\Delta z_norm| \geq 0.15)}{n}) * 100$ ou $(\frac{card(|\Delta z_norm| \geq 3\sigma(\Delta z_norm))}{n}) * 100$ en pourcentage.
- Erreur moyenne quadratique (RMSE) qui est similaire à la déviation standard. $RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\Delta z_{norm_i})^2}{n}}$.
  Où n est le nombre total de l'échantillon de test, et $card$ est la fonction cardinale.

Chapitre 1

Travaux existants dans la littérature

1.1 Méthodes basées sur les modèles (template-based)

Ces méthodes sont basées sur de modèles prédéfinis utilisant la distribution spectral de l'énergie. Elles supposent un ensemble de modèles de galaxies et tire le modèle le plus adapté et ensuite deduire le redshift [5, 16, 23].
Cette méthode de mesure du redshift est biaisée (seulement les galaxies) et ainsi entraîne des erreurs dans la prédiction du redshift. À partir de template-based on peut déterminer d'autres composants physiques que le redshift [13]. Plusieurs tentatives de l'amélioration de la précision du redshift de galaxies ont été faites par cette méthode, ci-dessous un tableau des certains travaux intéressants de la littérature concernant cette méthode.

paper	database	type/SED TEMP	$\mu(Z_{norm})$	$std(Z_{norm})$	$\|Z_norm\| \geq 0.15(\%)$
[40]	PHAT	GALAXY/LP Flat Err	0.0013	0.0445	9.88
	CAPR	Galaxy/LP HDF Err	-0.0081	0.0490	7.65
[5]	HDF	GALAXY/BPZ	0.08		
		Galaxy/ML	0.10		
[4]	SDSS DR12	GALAXY	0.0000584	0.0205	4.11

TABLE 1.1 – Résultat concernant la méthode basée modèle

Dans [5] deux modèles probabilistes ont été utilisés : maximum likelihood (ML) et le modèle bayesian (BPZ).

Dans [40], l'analyse est faite directement sur les différentes bases de données sans les déplacer. Cela permet non seulement d'éviter un télé-chargement long des données et l'obligation d'avoir un très grand espace de stockage, mais également de profiter de la bonne structuration de bases de données relationnelles.

Ils ont également fait l'analyse sur multi-bandes variés, ce qui ne permet de faire que la méthode d'analyse basée modèle. Deux types de modèles de distribution l'énergie spectrale (SED) de galaxies ont été utilisés (BPZ et LP). Le modèle Bayesian a été sélectionné pour l'analyse.

[4] a introduit une méthode hybride, utilisant en même temps la régression (méthode empirique) et l'adaptation à un modèle de galaxies (méthode basée modèle). Cette méthode bat toutes les autres mais elle est biaisée car elle s'est focalisée uniquement sur les galaxies.

1.2 Méthodes empiriques

Les méthodes empiriques tentent de réduire l'erreur de prédiction du redshift par les méthodes statistiques et machines learning. Ces méthodes ont besoin des données étiquetées (chacun de vecteurs dans l'ensemble d'apprentissage doit être étiqueté par son redshift). Ce qui oblige à fusionner les données de la photométrie et spectroscopiques des mêmes objets (dans catalog [1], les objets avec les mêmes objid).La proportionnalité des images étiquetées par leur redshift est très petite par rapport à celles non étiquetées.

paper	database	type/method	$\mu(Z_{norm})$	$std(Z_{norm})$	$\|Z_norm\| \geq 0.15(\%)$
[34]	SDSS DR12	Galaxy/AdaBoost stacking	0.0008 (μ^{50})	0.0248 (σ_{68})	0.733
[18]	SDSS DR 10	Galaxy/DNNs	0.00	0.03 (σ_{68})	1.71
		AdaBoost	-0.001	0.03 (σ_{68})	1.56
[13]	SDSS DR 13	All type/DNN	-0.0255009	0.152743	9.83078
		All type/XGB	0.00149561	0.168866	10.7685
[10]	SDSS DR9	All type/DCMDN	-0.007	0.014	

TABLE 1.2 – Résultats concernant la méthode empirique

Dans [8], le réseau de neurones artificiel a été testé sur les données photométries sous forme de vecteurs (magnitudes et couleurs) de type galaxy provenant de SDSS et obtenir une précision rms (root mean square) de 0.0238.
Dans le papier [34] plusieurs modèles et architectures ont été testé et le meilleur parmi est le AdaBoost empilé. AdaBoost fait partir de la classe des algorithmes d'apprentissage de boosting qui fait l'apprentissage sur plusieurs modèles hiérarchiquement en augmentant à chaque niveau la distribution des éléments mal ajustés. Mais dans [34] on fait en plus du boosting mais également un empilement de modèles renforcés. Ces modèles renforcés sont hiérarchisés et chaque modèle suivant tient compte du redshift prédit du precedant.
Ben Hoyle, dans [18] utilise les réseaux de neurones profonds et AdaBoost pour estimer le redshift à partir des images provenant de la collecte SDSS DR 10. Dans [13], XGBoost et MLP a été utilisé pour estimer le redshift sans préclassification (Galaxies, étoiles et quasars confondus)et diminuer ainsi le biais d'estimation. [10] utilise la mixture de densité gaussian et le réseau convolutif.

1. http ://skyserver.sdss.org/dr13/en/help/browser/browser.aspx

À partir des réseaux convolutifs, un ensemble des paramètres de la distribution gaussian sont estimés, ainsi il obtient la meilleure deviation (0.014) standard et une erreur moyenne de -0.007 de l'estimation du redshift.

Chapitre 2

Modèles de l'apprentissage automatique

Ce chapitre est consacré aux différents modèles d'apprentissage profonds. Il se focalisera principalement sur les réseaux de neurones convolutifs puisqu'ils ont plus été utilisé dans ce travail que le reste, et il se terminera par un petit détail sur un des modèles renforcé (xgboost) .

2.1 Réseau de neurones artificiels

Le réseau de neurones artificiels est une fonction mathématique s'inspirant du réseau de neurones biologique. Il prend en entrée un vecteur de dimension d qu'il va par la suite transformer en un autre vecteur de dimension k en sortie. Pendant cette transformation, chaque élément de position i du vecteur d'entrée est multiplié par un réel w_{ij} pour donner le vecteur Sum_j de la sortie. L'élément j du vecteur de la sortie est obtenu en faisant la somme des éléments du vecteur Sum_j. Pour avoir une forme non linéaire, une fonction non linéaire est appliquée sur la sortie. Les w_{ij} sont appelés les poids de connexions entre les neurones. Chaque vecteur est appelé une couche, ainsi le vecteur d'entrée est appelé la couche d'entrée et celui de sortie la couche de sortie. Une couche intermédiaire est appelée une couche cachée.
Soit $I = (I_1, ..., I_d)$ le vecteur d'entrée et $S = (S_1, ..., S_k)$ celui de la sortie, W la matrice des poids et A la fonction d'activation. $\forall j = 1..k\ S_j = A(\sum_{i=1}^{d} W_{ij} * I_i)$.

L'apprentissage est l'étape de calibrage des paramètres, nous avons quatre modes d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage par renforcement. Notre travail est basé sur l'apprentissage supervisé. L'apprentissage supervisé suppose la connaissance au préalable de l'étiquette (valeur à estimer pour une entrée) de chaque individu de l'ensemble de l'échantillon d'entrée.À partir de cet échantillon étiqueté, un algorithme d'apprentissage essaie de trouver une corrélation

entre les données et leurs étiquettes en donnant des valeurs aux poids du réseau. Quant à l'apprentissage non supervisé, cette corrélation est obtenue à partir des données non étiquetées.

2.1.1 Perception multi-couches (MLP)

— Perception

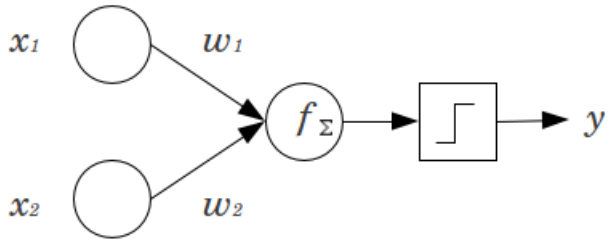

FIGURE 2.1 – Perceptron

Le perception est un réseau de neurone artificiel à une couche d'entrée et une de sortie sans couche cachée. Comme on peut voir un exemple sur la figure 2.1, le perception est le modèle de réseau de neurones le plus simple. Son défaut est son incapacité à estimer une fonction non linéaire.

— Perception multi-couches (MLP)

Le MLP(Multi-layer perceptron) est un réseau de neurones à au moins une couche cachée. On peut le considérer comme une séquence d'un ensemble de perceptrons. Cette manière de séquencer les perceptrons et d'activer chaque sortie de chaque percetron par une fonction non linéaire émerge un comportement non linéaire permettant ainsi d'estimer les fonctions non linéaire. Le nombre de séquence de perceptron défini la profondeur de l'architecture du réseau. Le MLP augmente le nombre de paramètres(poids) à calibrer, en fonction de l'augmentation de la profondeur. L'augmentation de la profondeur peut facilement amener le modèle à spécialiser le calibrage des poids sur les données d'entrée (sur-apprentissage), ce qui entraîne une mauvaise estimation d'un nouveau exemple.

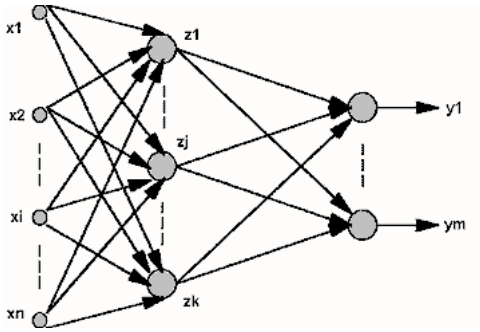

FIGURE 2.2 – MLP

2.1.2 Auto-Encoder

Un auto-encoder est un réseau de neurones dont sa phase d'apprentissage est non supervisée et chaque sortie du réseau est semblable à l'entrée. Le vecteur résultant est l'une des couches cachée appelée le code qui est un condensé de la représentation de l'entrée, il est comme l'analyse de la composante principale (PCA) mais il permet de plus une représentation non linéaire par rapport au PCA [39]. La partie de l'architecture réseau avant l'obtention du code est appelé codeur et celui après le décodeur, comme on le voir sur la figure 2.3.

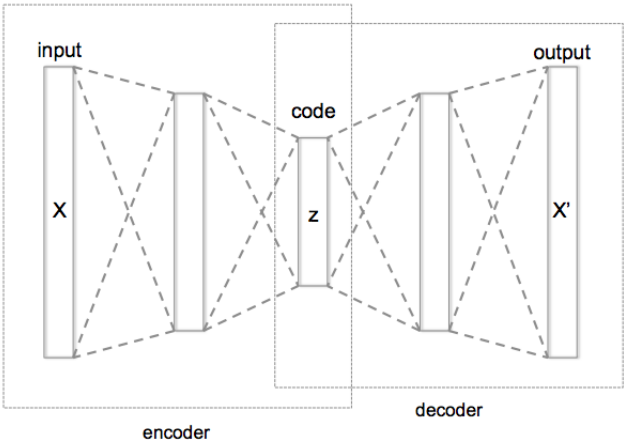

FIGURE 2.3 – Auto-encoder

2.2 Réseau de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs sont de type de réseau de neurone travaillant sur des images en se focalisant sur des caractéristiques locaux de l'image qu'il transforme d'une abstraction à une autre à travers les couches [35]. Il s'inspire du cortex visuel du chat, ce qui le rend plus approprié pour la vision artificielle.

Le réseau convolutif permet non seulement l'obtention d'une bonne précision par rapport au réseau de neurone classique mais aussi permet la diminution du nombre de paramètre du réseau à apprendre (les poids) à travers le partage des poids.

Noyau ou filtre(Kernel), Opération de convolution, Déplacement (strides), Padding
Le noyau est la matrice de paramètres qui se connecte sur la partie locale de l'image pour l'opération de convolution, donc cette partie locale doit avoir la même dimension que le noyau. Le noyau contient alors les paramètres (poids) à apprendre et il sera utilisé pour tout le reste des parties locales de même taille de l'image en faisant passer une fenêtre de déplacement sur l'image. L'opération de convolution est de multiplier case par case les deux matrices pour obtenir une matrice M, puis faire la somme des éléments de M pour l'obtention d'un scalaire qui est le résultat final de la convolution, c'est donc une fonction non inversible. Le déplacement est le nombre de case pour se déplacer vers la droite et en bas pour positionner le noyau sur une nouvelle partie locale de l'image pour une autre opération de convolution. Au coin supérieur gauche de l'image la position du déplacement est (0,0), puisque une opération complète de convolution donne comme résultat une image intermédiaire A qui est une abstraction de l'ancienne, la valeur de A(i,j) correspond au résultat de l'opération de convolution à la position du déplacement (i,j).À la position (p, q) de déplacement, la prochaine position à droite est (p+1, q) et en bas (p, q+1). On remarque rapidement la diminution de la dimension de l'image , le padding permet d'augmenter la taille de l'image par des zéros pour avoir la même dimension de sortie que l'entrée.

Fonctionnement
Le réseau convolutif divise la matrice de l'image en des petites matrices(de même taille que le noyau) ordonnées par leurs positions et ces petites matrices passe dans le même réseau de neurones (c'est-à-dire les mêmes poids : noyau) pour donner une nouvelle matrice en sortie qui sera à son tour une entrée d'une nouvelle couche et ainsi de suite jusqu'à la sortie. Dans mon explication ci-dessus certes j'ai fait plus d'abstraction de certains détails, il y a pas que de couches de convolution qui existe, il y à des couches de regroupement (pooling), de couches d'activation (ReLu (généralement utilisé), softMax, tanh, ...), de couche de simplification pour empecher de passer en surapprentissage (dropout, batch normalization, ...) [35]

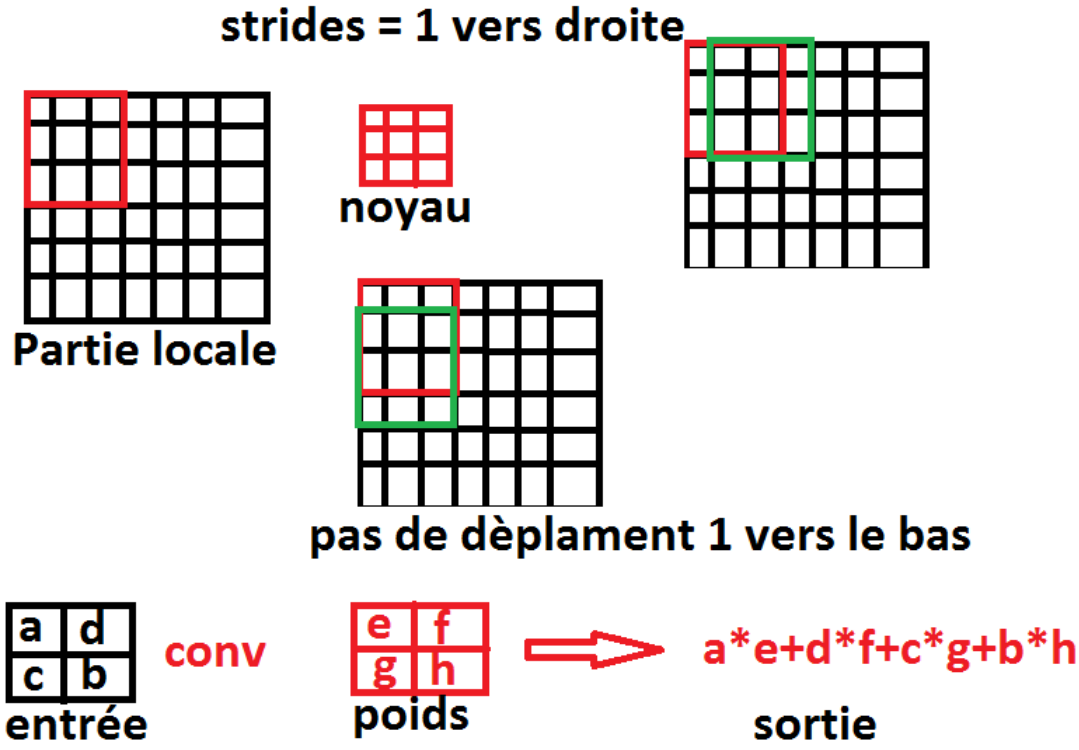

FIGURE 2.4 – fonctionnement du réseau de neurone convolutif

2.2.1 Couche convolutive

Plusieurs amélioration de la couche convolutive ont été proposées dans la littérature et ont donné des bons résultats surtout sur la base de données de Image-Net [42]Nous avons plusieurs de réseau convolutifs, mais nous avons présenter que quelques en détail.

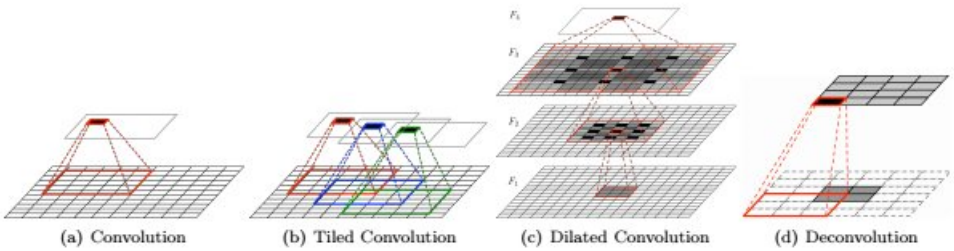

FIGURE 2.5 – Illustration de quelques type de CNN

Tiled convolutional network (TCN)

Le réseau convolutif d'origine n'utilise qu'un seul noyau qui est partagé entre toutes les parties locales de l'image. Il permet d'éviter des problèmes de translation mais pas d'autres dérangement dans l'image telle la rotation. Le TCN permet d'utiliser à chaque couche convolutive plusieurs noyaux, ainsi permettant plus d'abstractions[30, 47, 54].

Transposed convolution ou Deconvolution (TC)

Une couche convolutive de CNN prend une image (matrice) en entrée et retourne une autre image de dimension inférieur ou égale à l'image d'origine, le TC fait l'opération inverse. Le TC transforme chaque case de la matrice de l'image en une petite matrice. Ce n'est pas vraiment une opération inverse du CNN comme l'apparence nous fait croire mais c'est juste le type d'entrées et sorties qui sont inversé. Pour un CNN, les poids se fixent sur une localité et retourne un scalaire tandis pour TC on part d'un scalaire pour arriver à une matrice locale. Pour plus de détail on peut voir [51, 53, 52, 27, 46]

Dilated convolutional neural

Le dilated convolution neural élargit de plus le filtre en mettant autant des zéros que nécessaires (un hyper-paramètre à donner) entre une case et ses voisines de la matrice noyau. Pour plus de détail concernant le dilated convolutional le lecteur peut se référer [49, 24, 31, 38].

Network in network (NiN)

Le NiN[26] est une des améliorations la plus intéressante par rapport aux précédentes, elle permet dans chaque couche convolutive d'utiliser comme filtre un petit réseau de neurones profonds. C'est-à-dire sur la partie locale, est appliquée un petit réseau de neurones profond au lieu d'un perceptron comme dans le cas du CNN. C'est qui veut dire qu'une couche de NiN est constituée d'une couche de convolution de noyau (k, m) suivi par plusieurs couches de convolution de noyau (1,1) chacune.

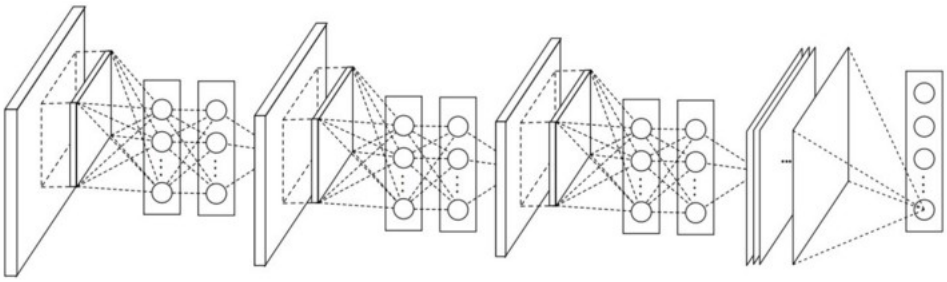

FIGURE 2.6 – Illustration NiN

Inception module

Inception module[42, 17, 43, 41] est une généralisation de NiN, dans chacune de ses couches on peut avoir plusieurs successions de couches convolutives de noyau de dimension (k_i,m_i) où les (k_i,m_i) ne sont pas nécessairement des couples de 1, les résultats de cet ensemble de couches successives sont fusionnés pour donner un ensemble d'images qui sont plus d'abstraction (profondeur). Un type de ce modèle a gagné le prix du challenge de la reconnaissance visuelle sur ImageNet en 2015 (GoogleNet [42]). Plusieurs amélioration de ce modèle ont été fait, il y a eu même une fusion avec le residual network qui donne plus de précision et permet de réduire le temps d'apprentissage également [42, 17, 43, 41]. Notre modèle s'inspire de ce type de réseau convolutif.

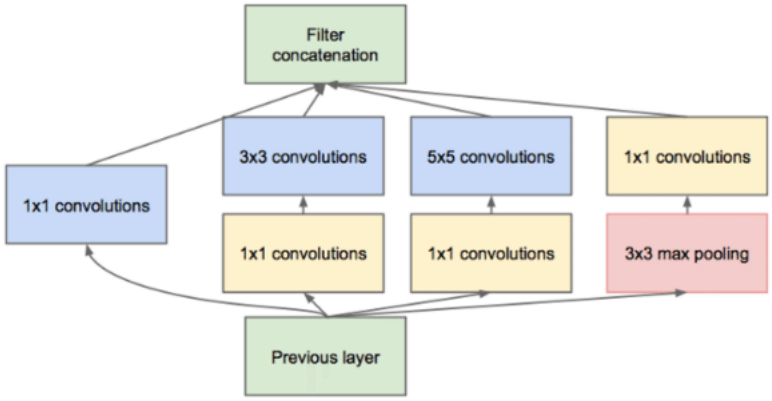

FIGURE 2.7 – Illustration Inception module

2.2.2 Couche d'agrégation (pooling)

La couche d'agrégation permet non seulement la réduction de la dimension mais également permet de capturer certaines abstractions statistiques locales [20, 48, 50, 37]. Plusieurs couches d'agrégation ont été définies parmi elles la couche maxpooling et celle de averagePooling sont les plus utilisées. La couche maxpooling permet de choisir la valeur maximale locale parmi les valeurs de la région considérée et le averagepooling fait la moyenne de la région locale considérée.

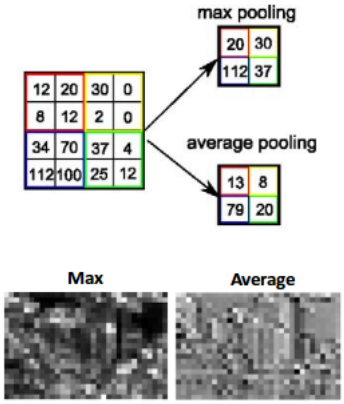

FIGURE 2.8 – maxpooling et averagepooling

2.3 Boosting

Le boosting est une technique d'apprentissage qui s'appuie sur des modèles légers appris pour construire de modèles sophistiqués. Il opère de manière séquentielle d'un modèle à un autre tout en corrigeant l'erreur du précèdent en augmentant la distribution de ceux qui sont mal prédits. L'ensemble des modèles légers n'est pas nécessairement homogène, on peut faire le boosting avec plusieurs modèles de familles différentes. Ce type de modèle ne cessent de nous surprendre dans certains domaine par rapport aux réseaux de neurones profonds comme dans notre étude actuelle.

Regularized gradient boosting tree

Un arbre de décision est une fonction/modèle permet d'estimer une valeur ou classifier en construisant un arbre dont les feuilles sont les classes ou les valeurs estimées. Formellement un arbre de décision est décrit comme suit :
Soit $x \in R^n$ l'objet pour lequel une valeur y doit être prédite, f la fonction correspondant à l'arbre de décision. $f(x) = W_{q(x)}$ où $W \in R^d$ est le vecteur correspondant aux valeurs prédites et $q : R^n -> \{1..d\}$.

gradient boosting tree est la version du boosting où le modèle léger est un arbre de décision et l'ensemble de ces modèles est obtenu en minimisant le gradient, du passage d'un modèle à un autre. Pour éviter le problème du

sur apprentissage, les techniques de régularisation sont appliquées. Supposons que nous avons k estimateurs (modèles légers appris), l'estimation d'une valeur correspondante à x est $\widehat{y} = \sum_{i=1}^{k} f_i$

Le problème d'optimisation se résume à minimiser l'expression suivante : $L(\Phi) = \sum_{i=1}^{n} l(y, \widehat{y}) + \sum_{j=1}^{k} \Omega(f_j)$ où $l(y, \widehat{y})$ est la fonction de perte, $\Omega(f_j)$ permet de contrôler la complexité du modèle pour empêcher le sur-apprentissage et Φ est l'ensemble de paramètres. $\Omega(f_j) = \frac{1}{2}\lambda \sum_{i=1}^{m} W_i^2 + \alpha \sum_{j=1}^{m} |W_j| + \gamma m$, où λ, α et γ sont des hyper-paramètres.
Son implantation est xbboost, pour plus de détails on peut se référer citeboost.

2.4 Apprentissage profond

Le problème d'apprentissage se résume à un problème d'optimisation, çà revient à adapter les paramètres du modèle pour une généralisation du monde concerné (population) à partir d'un échantillon des données. L'apprentissage profond est l'adaptions des paramètres d'un modèle construit à partir de la superposition de plusieurs couches de réseau de neurones. La Superposition des couches permet de capturer plus d'abstractions. Ces paramètres sont appelés les poids (weights) du réseau.

2.4.1 Apprentissage par rétro-propagation

La rétro-propagation permet de propager l'erreur en arrière sur toutes les couches pour ajuster les poids.

Optimisation

Trouver les poids adaptés se ramène à optimiser une certaine fonction. Cette fonction dépend du type d'architecture de réseaux de neurones. La prédiction d'un individu X par une architecture de n couches en mode feed foward ou une architecture où les signes sont propagés dans un seul sens de l'entrée à la sortie (comme MLP) est $\widehat{Y} = Y_n$ tel que $Y_1 = \sigma(W_1 * X + b_1)$, $Y_2 = \sigma(W_2 * Y_1 + b_2)$, $...Y_n = \sigma(W_{(n-1)} * X + b_{(n-1)})$ où W_i sont les poids du réseaux, b_i les biais et σ la fonction d'activation [36, 28, 12]. Pour l'obtention des poids adaptés d'une architecture feed forward, il revient à minimiser la fonction $\sum_{i=1}^{m} L(Y^i, \widehat{Y}^i)$ où Y^i est la vraie valeur correspondante à X, m est la taille de l'échantillon et L est la fonction de perte ainsi dans notre cas $L(Y, \widehat{Y}^i) = (\frac{Y - \widehat{Y}}{Y+1})^2$. Pour la minimisation, la descente du gradient est utilisée, pour chaque étape du gradient les paramètres et biais sont mis en jours. $W^i = W^{i-1} - \alpha \frac{\delta L}{\delta W} W^{i-1}$ où α est le taux d'apprentissage .

Le problème majeur du gradient est la difficulté d'obtention du minimum global, la rapidité de convergence et son coup de calcul. Plusieurs améliorations du gradient ont été effectuées concernant ces problèmes (SGD [15], AdaGrad [11], Adam [25], ...). Pour ce travail Adam et SGD ont été utilisés.

Régularisation

La régularisation permet de contrôler la complexité du modèle en pénalisant la fonction à optimiser d'une certaine valeur. Ce qui permet au modèle de généraliser mieux le monde concerné et éviter le sur apprentissage. Plusieurs méthodes de régularisations existent : **Batch normalisation [22]** qui normalise les données à l'entrée de la couche suivant la couche du batch, **L1 et L2** qui augmente sur la fonction de paramètres à optimiser la quantité respectivement $\lambda * \sum_{i=1}^{m} |w_i|$ et $\lambda * \sqrt{\sum_{i=1}^{m} w_i^2})$ où λ est un hyper-paramètre, et dropout qui supprime certains neurones d'une certaine proportionnalité α de la couche concernée. Dans ce travail batch normalisation, L2 et dropout ont été utilisés.

Chapitre 3

Résultats et discussion

3.1 Données

Dans cette section, nous allons décrire les pré-traitement sur les données (images) et mentionner leur source ainsi que le mode d'accès.

3.1.1 Source et nature de données

SDSS(Sloan Digital Sky Survey IV)

En termes simples, le Sloan Digital Sky Survey est l'étude astronomique la plus ambitieuse jamais entreprise. SDSS a observé en détail un quart de l'ensemble du ciel, déterminant les positions et la luminosité absolue de centaines de millions d'objets célestes[7]. Il mesure également les distances à plus d'un million de galaxies et quasars [1].

DR13 (database release version 13)

[2] Data Release 13 stocke les prèmieres données de la quatièmes phase de la collecte de SDSS. Elle inclut les données prises à partir du 25 juin 2015, et englobe plus que un tiers de la sphère céleste. Avec plusieurs mesures du ciel faites et dans plusieurs façons, 1231051050 objets (catalogues) ont été collectés [3]

1. http ://skyserver.sdss.org/dr13/en/sdss/sdsshome.aspx
2. http ://skyserver.sdss.org/dr13/en/help/browser/browser.aspx
3. http ://www.sdss.org/dr13/scope/

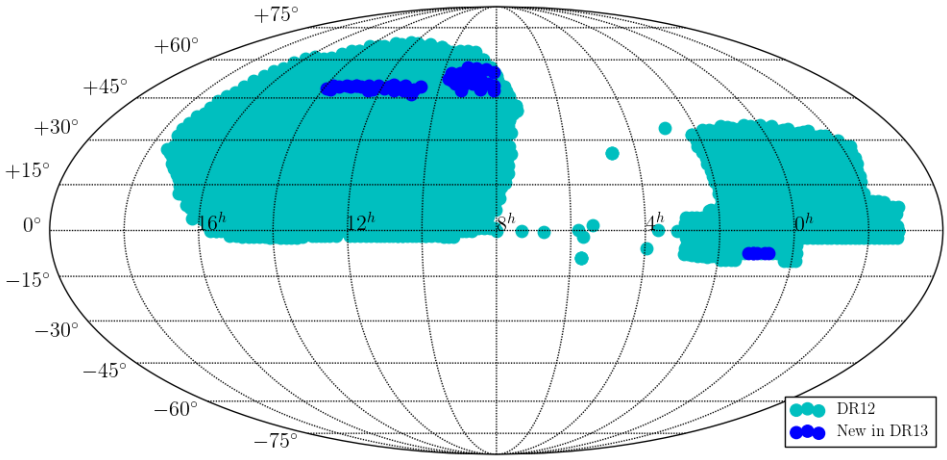

FIGURE 3.1 – eBOSS surveys

Toutes nos données proviennent de la collecte de données SDSS . Elles sont télé-chargées sous forme des vecteurs chacune dans 5 bandes différentes (griuz). À partir de ces données sous forme de vecteurs provenant des tables SpecPhoto et PhotoPrimary de DR13 de SDSS, les urls des images correspondant à chaque vecteur sont générés. À partir de ces urls, les images sont télé-chargées et sauvegardées localement.

Les données ne sont pas sélectionnées, elles sont chargées directement aleatoires et sont de distributions différentes.

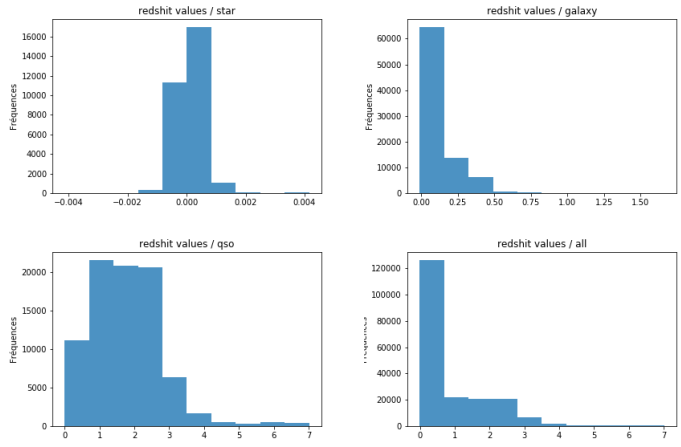

FIGURE 3.2 – Distribution du redshift selon les catégories des objets cosmologiques

3.1.2 Pré-traitement sur les images

Les images dans leur taille d'origine sont très grandes (1361*2048 pixels), et contiennent des quantités lumineuses très variées. La taille de chaque image est réduite à 32*32, cette réduction est faite en utilisant la moyenne des coordonnées centrales (rowc_ et colc_) des objets(images) de chaque bande. Pour normaliser, le minimum de la quantité lumineuse (pixel) de chaque image est soustraite de l'image et le logarithme est appliqué à chaque pixel de l'image.

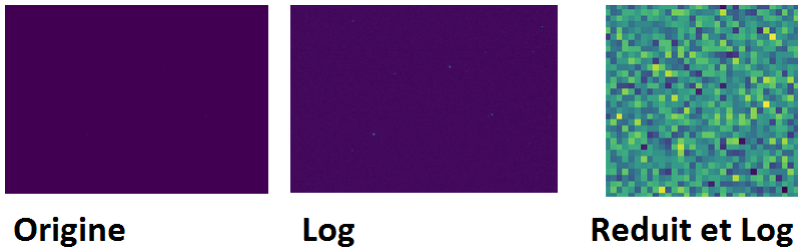

FIGURE 3.3 – pretement sur les images

3.2 Configurations techniques

Pendant toutes nos expérimentations, nous avons utilisés des machines hpc (haute performance computing) équipées d'une carte graphique graphique du type NVIDIA GP104GL et un processeurs de 10 cœurs physiques de type Intel(R) Xeon(R) CPU E5-2640 v4 avec une fréquence de 2.40GHz.

Pour la concrétisation de notre architecture du CNN, nous avons utilisé la bibliothèque Keras [6] et pour celle du boosting nous avons utilisé la bibliothèque xgboost [4] et scikit-learn [32].

3.3 Expérimentations

Dans ce travail, nous avons choisi d'utiliser deux principales modèles de l'apprentissage automatique à savoir boosting qui est un modèle ensembliste, et réseau de neurones convolutifs (apprentissage profonds). En effets le boosting et les réseaux convolutifs ont donnée des bons résultats non pas seulement sur l'estimation du redshift [13, 10] mais également sur d'autres problèmes de régression et classification [14, 35].

Le choix et l'adaptation des hyper-paramètres ont été fait manuellement pour les deux types de modèles en faisant des test sur un petit échantillon de données et quelques combinaisons des hyper-paramètres. Au niveau du CNN, l'obtention des bons hyper-paramètres est compliquée car le CNN possède beaucoup d'hyper-paramètres à adapter.

4. http ://xgboost.readthedocs.io

Dans le tableau tab. 3.1 est mentionnée la taille de données d'apprentissage et test de chaque type de corps cosmologiques.

Type_cosmologique	*#Apprentissage*	*#Test*
Galaxie	59887	25667
Quasar	58668	25144
Étoile	20994	8998
Tous	139550	59808

TABLE 3.1 – Nombre d'élément de chaque type cosmologique

3.3.1 CNN (type : inception module)

Notre modèle final du type de réseau de neurones, celui qui s'adapte bien à nos données c'est-à-dire qui réduit mieux l'erreur normalisée (Δz_{norm}) est un réseau convolutif de la famille des inceptions modules. Notre modèle est construit comme suit :
Notre couche principale (*cp*) fig.3.4 est constituée de la concaténation de trois sous couches suivies par une sous couche d'agrégation de filtre 2 par 2 et d'un pas de déplacement 2 (strides = 2). Les trois sous couches concaténées sont décrites comme suit :

- 1^{er}. Constitué de trois sous couches de convolution de filtre 1*1 et de pas déplacement 1*1 formées de manière séquentielle.
- 2^{e} Trois sous couches de convolutions formées de manière séquentielle de 3*3 comme filtre et de 1*1 comme pas déplacement.
- 3^{e} Trois sous couches de convolution formées de manière séquentielle de 5*5 comme filtre et de 1*1 comme pas déplacement.

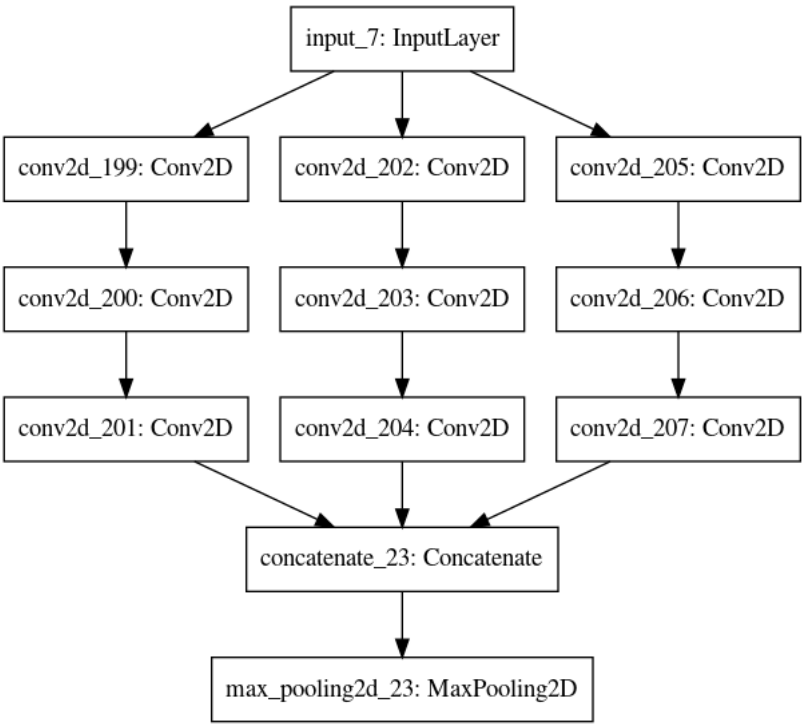

FIGURE 3.4 – La couche principale de notre architecture neurale

Dans chaque couche convolutive, la fonction d'activation est $relu(x) = max(0, x)$, pour éviter le sur-apprentissage une régularisation de type $l2$ est appliquée à chaque couche convolutive. Le padding zéro est appliqué sur toutes les couches convolutives. Notre architecture finale est constituée de 5 couches du même type que la *cp* avec de nombre des sorties différents. Pour l'apprentissage, *Adam* est utilisé entant qu'algorithme d'optimisation, à chaque deux reprises (epochs) de l'algorithme le taux d'apprentissage est divisé par deux, le early stoping est appliqué pour éviter le sur apprentissage de plus. Le early stoping est une manière d'arrêter rapidement l'apprentissage quand la précision sur les données d'apprentissage est sur le point de dépasser celle sur les données de validation. Le nombre de reprise(epochs) est paramétré à 30 et la taille de batch size est mise à 256.

Dans la suite du document, nous supposons les conventions suivantes : $out1 = |znorm| > 0.15(\%)$, $out2 = |znorm| > 3std(\%)$, $RZ = RMSE\_znorm$

type	RZ	μ_{norm}	σ_{norm}	RMSE	$out1$	$out2$
GXY/CNN	0.035056	0.035032	0.035032	0.046099	0.752123	1.310150
QSO/CNN	0.289760	0.276163	0.276163	1.065627	66.988385	0.441682
All/CNN	0.261338	0.252728	0.252728	0.906072	34.453168	1.325543

TABLE 3.2 – Résultats du CNN

Ces résultats s'éloignent, dans le sens négatif, des meilleurs résultats de l'état de l'art. Cette mauvaise estimation de redshift par notre architecture de réseau convolutif peut s'expliquer par plusieurs raisons dont le fait qu'on possède moins de données pour booster l'apprentissage, le fait que l'image est trop réduite (trop de perte d'information) sans essayer de la réduire par la convolution.

3.3.2 Boosting (type : xgboost)

Trois principaux hyper-paramètres (nombre d'estimateurs, taux d'apprentissage et la profondeur de l'arbre) ont été réglés et adapté et tout le reste des paramètres ont été laissé avec leurs valeurs par défaut. Les trois modèles configurés du boosting sont mentionnés dans le tableau 3.3.

Quelques paramètres commun

learning_rate = 0.1, colsample_bylevel=0.592, reg_alpha=0.651, reg_lambda=2.84

modèles	n_estimators	max_depth
model1	150	6
model3	150	5
model2	200	5

TABLE 3.3 – Paramètres de différents modèles de xgboost

3.3.3 Expérimentions avec XGBoost

model	type	RMSE_znorm	μ_{norm}	σ_{norm}	RMSE	$out1$	$out2$
$model1$	Galaxy	0.035728	0.000548	0.035723	0.046813	0.654537	1.137648
	QSO	0.318635	0.017468	0.318156	0.940743	64.03118	0.898823
	All	0.320223	0.015833	0.319832	0.801991	38.971375	2.543138
	Star	0.000429	0.000015	0.000428	0.000429	0.0	1.066904
$model3$	Galaxy	0.035938	0.000375	0.035936	0.04507	0.580512	1.133752
	QSO	0.318635	0.017468	0.318156	0.940743	64.03118	0.898823
	All	0.31818	0.010996	0.31799	0.799093	39.24057	2.539794
	Star	0.000432	0.000013	0.000432	0.000433	0.0	1.044677
$model2$	Galaxy	0.03556	0.000058	0.03556	0.046432	0.654537	1.211673
	QSO	0.317182	0.016874	0.316733	0.957897	64.528317	0.775533
	All	0.320004	0.013142	0.319734	0.798689	39.569957	2.523074
	Star	0.000423	0.000011	0.000423	0.000423	0.0	1.044677

TABLE 3.4 – Résultats du XGBoost

Les résultats obtenus par xgboost sont intéressants et s'approchent mieux des ceux existants dans l'état de l'art [13, 10, 4]. Nous remarquons que les modèles estiment bien les objets qui sont proches comme les étoiles puis les galaxies. Pour une estimation non biaisée tous les types de corps cosmologiques sont considérés et le modèle réduit mieux l'erreur sur l'ensemble de ces données fusionnées.

Conclusion

Dans cette première partie, nous avons essayé d'estimer le redshift à partir des modèles de l'apprentissage automatique tels que xgboost et le réseau de neurones convolutif qui est une architecture de l'apprentissage profond. Nous avons mis au point également une architecture de réseau convolutif qui s'adapte mieux à nos données pour l'estimation du redshift dont les hyper-paramètres ont été calibrés à la main. Certains hyper-paramètres du modèle de xgboost ont été adaptés en utilisant un petit échantillon de données pour calculer le score des validations croisées à partir de la librairie sciklearn (cross_val_score).

Deuxième partie

Classification des supernovas à partir des données photométriques en séries temporelles

Contexte

La plupart des sources observées dans le ciel nocturne évoluent pendant une très longue période de temps (des millions, voire des milliards d'années), ce qui les rend pratiquement statiques par rapport à l'échelle de temps d'une vie humaine. Cependant, ce n'est qu'une description partielle des événements possibles qui se déroulent dans l'Univers. Pendant longtemps, les astronomes ont enregistré des événements d'observation dont la durée de vie atteint des jours ou des mois. Plus récemment, l'avènement des télescopes modernes a élargi [2] notre capacité de détecter les événements astronomiques qui se produisent de quelques secondes à plusieurs années. Ces phénomènes sont appelés transitoires et peuvent concerner les chercheurs travaillant dans le domaine de l'astronomie temporelle [19]. C'est dans ce sens qu'une compétition de classification de supernova est née.

Dans cette deuxième partie du mémoire je présente ma participation à une compétition astro-physique concernant l'identification des transits (**supernova**) à partir des données de simulation de LSST sous forme de **série temporelle**. Il est demandé de mettre au point une fonction, algorithme ou modèle, qui, en prenant une série de données mesurée (courbe lumineuse), permet de donner son type de supernova. Pour une raison d'obtention d'un estimateur fiable, plusieurs contraintes ont été imposées sur les données.

Supernova

Un supernova est l'explosion cataclysmique d'une étoile qui, pendant un temps, peut briller plus vivement qu'une galaxie entière composée de centaines de milliards d'étoiles.

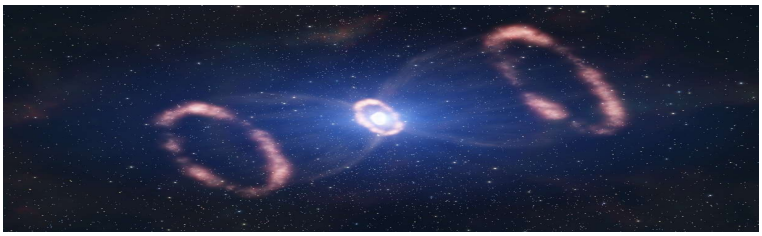

FIGURE 3.5 – Illustration de l'explosion d'une étoile (supernova)

Série temporelle

Une série temporelle est un ensemble ordonné de valeurs qui présente l'évolution d'un sujet au cours du temps. Ces valeurs sont séparées les unes des autres par la même valeur d'intervalle.

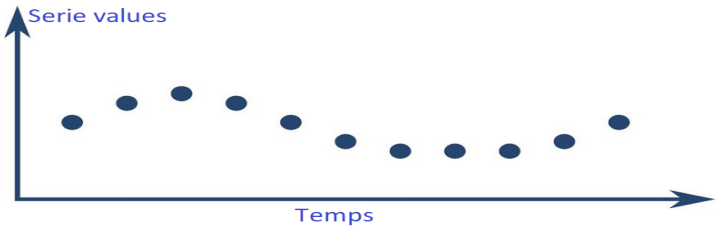

FIGURE 3.6 – Exemple d'une série temporelle

Références de mesure de bonnes prédictions

Nous allons utiliser dans cette deuxième partie deux principales références pour analyser l'exactitude et l'aptitude du modèle à bien prédire la bonne classe d'un supernova à partir de sa courbe lumineuse.

Accuracy

L'accuracy permet de donner la proportion des éléments bien catégorisés, elle est définie formellement comme suit : $accuracy = \frac{\sum_{i=1}^{n} fe(Y\_true_i, Y\_pred_i)}{\#Y\_true}$ où Y_true est les vraies classes des éléments , Y_pred aux classes prédites par le modèle et $\#Y\_pred$ correspond au nombre total d'éléments dans Y_pred, $fe(a, b) = 1$ si $a = b$ et 0 sinon.

rappel (Recall)

Le rappel donne la proportionnalité de la bonne prédiction des éléments d'une classe particulière. Soit c la classe à estimer son rappel, $rappel(c) = \frac{\sum_{i=1}^{k} f_c(Y\_pred_{c_i}, c)}{\#Y\_pred_c}$ où Y_true_c est la liste des éléments prédits juste pour la classe c, $f_c(x, c) = 1$ si $x == c$ et $f_c(x, c) = 0$ sinon.

Chapitre 4

Données et algorithmes utilisés

Dans ce chapitre nous allons décrire nos données et nos techniques de pré-traitement ainsi que les algorithmes d'analyse utilisés après ces pré-traitements pour la classification.

4.1 Données

Les données sont sous la forme des séries, dans quatre bandes de longueurs d'onde différentes donc de séries multivariées et de taille différentes. Ces séries ont des grilles temporelles très variées. Ce qui nous oblige à faire l'estimation de plusieurs valeurs manquantes ou d'estimer les valeurs de chaque série sur une grille spécifique. Pour cela, nous allons utiliser un ensemble de modèles estimateurs : le modèle paramétrique d'estimation de la courbe lumineuse (série) qui a été proposée dans [44], l'approximation polynomiale et l'estimation par processus gaussien. Parmi tout ces modèles le modèle proposé dans [44] est le plus intéressant et le plus proche.

La fonction paramètrique de Bazin et al. définie dans [44] est la suivante :

$$f(t) = A\frac{exp\left(-\frac{t-t_0}{t_{fall}}\right)}{1+exp\left(\frac{t-t_0}{t_{rise}}\right)} + B$$

Les cinq paramètres à estimer pour chaque courbe lumineuse dans chaque bande sont A, B, $t0$, t_{fall} et t_{rise}.

Dans un premier temps, nous avons essayé d'estimer les valeurs manquantes par le modèle de Bazin dans [44] et après utiliser ce même modèle pour définir une grille plus petite par rapport à la grille formée par l'estimation des valeurs manquantes. En effet l'estimation de valeurs manquantes entraine une grille temporelle de taille de plus de 600, ce qui peut rendre l'utilisation de certains algorithmes plus difficile. En plus nous essayé d'analyser les composants directement à partir des paramètres du modèle de Bazin.

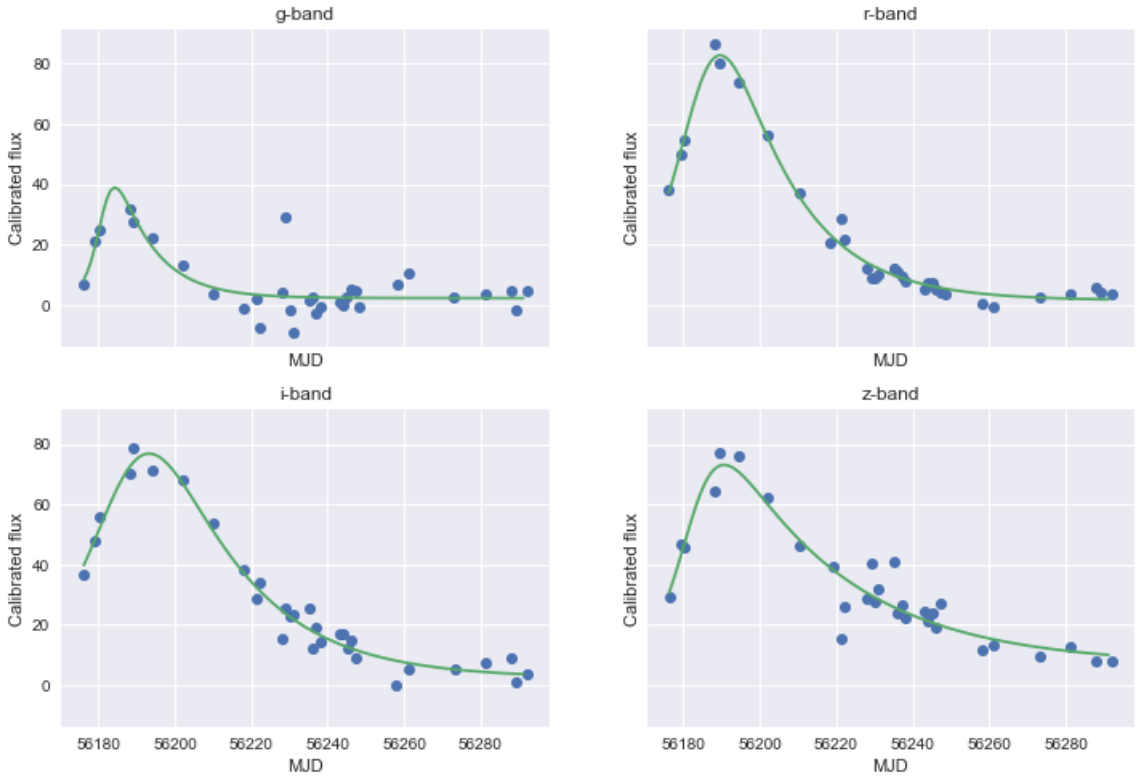

FIGURE 4.1 – Estimation des valeurs d'une série dans les quatre bandes

Plusieurs contraintes ont été imposées lors de la validation croisée c'est-à-dire : l'ensemble d'apprentissage est largement plus petit que l'ensemble de test (ensemble de test est 20 fois plus grand que celui de l'apprentissage) et il est également biaisé. Une classification binaire est faite entre la catégorie 0 de supernovas et les autres catégories.

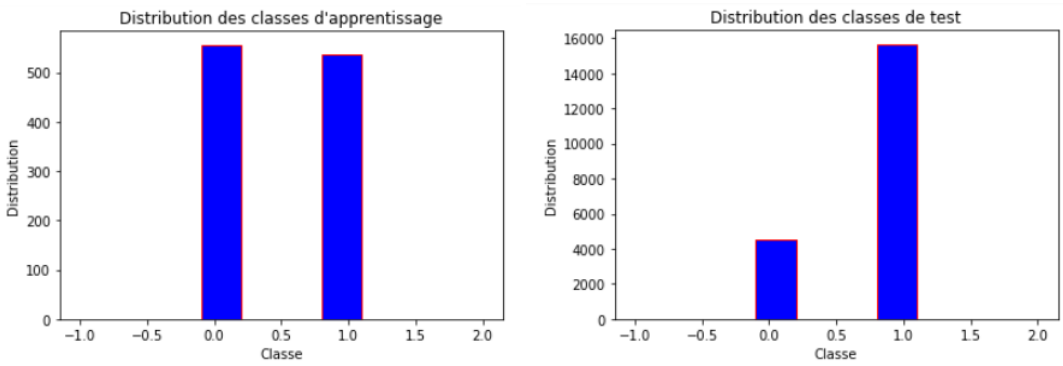

FIGURE 4.2 – Distribution de la classe de catégorie 0 par rapport à la distribution des autres catégories après transformation

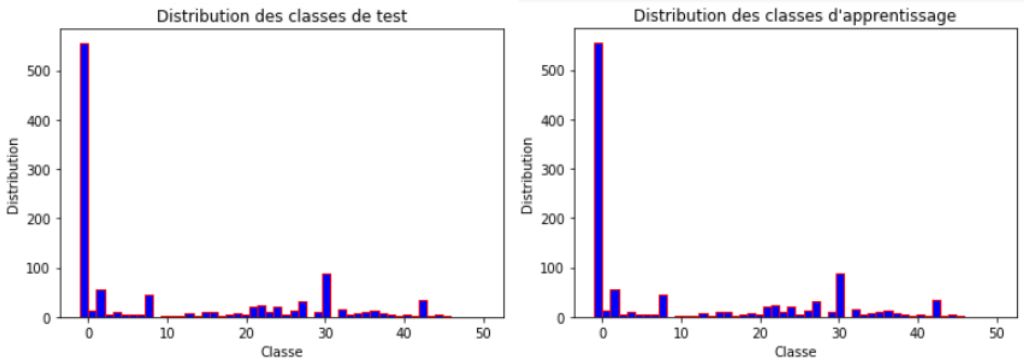

FIGURE 4.3 – Distribution des classes avant transformation

4.2 Différents algorithmes

Dans la littérature, il existe plusieurs méthodes de classification des données en série temporelle, dont la plupart cherche une distance de non-similarité ou de similarité pour séparer une catégorie des autres [45]. Une fois qu'une distance de similarité est définie, la classification est faite en considérant la distance comme caractéristiques. Il existe également d'autres techniques qui essayent d'extraire des caractéristiques directement à partir de la série elle même au lieu d'essayer de séparer globalement sur l'ensemble des séries pour en tirer une série ou une sous série qui divise mieux les classes.

Ces algorithmes de classification ont été catégorisés en six classes :

1. **Totalité de séries**, cette méthode calcule la similarité ou non similarité entre deux séries en considérant la taille totale de ces dernières et en les prenant comme deux vecteurs. Des techniques de calcul de similarité

comme la distance euclidienne peuvent ainsi être appliquées directement sur les séries. Cette méthode ne peut être utilisé dans un cas où les séries ne sont pas de même taille.

2. **Intervalle**, elle sélectionne un ensemble d'intervalles de la série, au lieu de considérer totalement la série, pour mesurer la similarité ou non similarité entre deux séries.
3. **Shapelets**, un shapelet est une sous série, dans cette famille de classification un shapelet qui sépare mieux les classes est cherché au long des série et ce shapelet est gardé comme référence pour le calcul de distance par rapport aux séries.
4. **Dictionnaire** Cette famille de méthode de classification de série se différencie de la famille des shapelets par le fait qu'on considère à ce niveau la fréquence ou la répétition d'une sous série comme caractéristiques au lieu de la présence ou non de cette sous série.
5. **Combinée** Combine un sous ensemble des méthodes ci-dessus pour former un seul classificateur.
6. **Basée modèle** Cette dernière méthode transforme les séries en des modèles et ces modèles sont comparés comme caractéristiques. C'est cette méthode qui a donné de meilleurs résultats parmi celles déjà testées.

Pour plus de détails sur ces méthodes, le lecteur se peut référer au papier [45].

En plus de ces algorithmes d'extraction des caractéristiques, les algorithmes de classification existants peuvent être utilisés. Nous avons alors testé six algorithmes de la fouille de données que nous décrivons ci-dessous.

* **Arbre de décision**, la méthode de classification basée sur un arbre de décision est un méthode qui, prenant un ensemble de données appelé l'ensemble d'apprentissage, construit une structure sous forme d'arbre dont les nœuds sont les attributs (caractéristiques ou features en anglais) des éléments de l'ensemble et les feuilles sont les étiquettes ou les différentes classes. Dans notre cas les caractéristiques sont la distance de chacune des séries dans les quatre bandes d'un supernova par rapport aux séries sélectionnées (selon la bande) comme meilleurs séparateurs ou les paramètres du modèle bazin et al. utilisés dans les quatre bandes quand cette dernière est utilisée comme modèle séparateur. Cet arbre est transformé en plusieurs règles qui ne sont que les différents chemins de la racine de l'arbre aux feuilles. Ces règles sont utilisées pour classifier un nouveau élément (individu). Plusieurs algorithmes pour construire cet arbre ont été étudiés [21].
* **Foret aléatoire**, comme boosting la méthode basée sur les forets aléatoires sont des méthodes ensemblistes. Contrairement aux boosting qui corrige l'erreur de manière séquentielle sur toutes les données pour former des petits modèles appris successivement, le foret aléatoire divise les données de l'ensemble de départ aléatoirement en de partitions de même distribution et sur chaque partition un arbre de décision est formée. Un nouveau élément est classifié selon la classe majoritaire prédite [33].
* **Boosting** voir la section 2.3 du chapitre 2

* **Réseau de neurones** voir la section 2.1 du chapitre 2
* **K plus proches voisins**, contrairement aux modèles ci dessous, la méthode de k plus proches voisins ne met au point une bonne fois pour tous un classificateur à partir des données d'apprentissage mais utilise ces données d'apprentissage lors de classification d'un élément x pour trouver k éléments qui ressemblent à x et associé à cet x la classe de ces k éléments [3].
* **Régression logistique**, la régression logistique [9] est un modèle linéaire qui, à partir des caractéristiques d'un élément (variables explicatives), forme une combinaison linéaire qui sera utilisée pour calculer une probabilité d'appartenance à chacune des classes (variable à expliquer). La classe ayant la probabilité la plus forte est choisie.

4.3 Expérimentations et résultats

Dans cette section nous présentons d'abord brièvement les algorithmes et techniques de classification utilisés puis nos résultats selon la technique utilisée. La taille de l'ensemble d'apprentissage est 1093 et celui du test est 20190, 556 sont d'une classe et 537 d'une autre parmi l'ensemble d'apprentissage.

4.3.1 SAX

Symbolic Aggregate Approximation (SAX) est une méthode de la famille Dictionnaire. La méthode Sax a été proposée dans [1] et eteind la méthode PAA (Piecewise Aggregate Approximation of time series). PAA résume une série de taille n en une autre de taille p ($p << n$) en divisant la série en p sous série d'à peu prés même taille et chaque sous série est remplacée par sa moyen. SAX améliore PAA en discrétisant la série obtenue par un ensemble fini de symboles qu'on appelle alphabet. Pendant nos expérimentations SAX a été testé après plusieurs transformations ou sans transformation des données. Cette méthode a l'avantage de la dimension réduite, ainsi elle rend l'application de différentes méthodes de fouilles plus légère, elle est beaucoup plus utilisée en biologie en fouille de texte. SAX a besoin de définir en plus de l'ensemble des alphabets, un ensemble ordonné de réels appelé les points de coupure. Cet ensemble permet de transformer chaque valeur de la série PAA en une lettre contenue dans l'ensemble des alphabets. Pour faciliter la mise au point de l'ensemble des points de coupure, les séries sont mises sous la forme normale de moyenne nulle et de déviation standard unitaire et ces points de coupure divise la courbe de données normalisées en de régions de probabilités égales et chaque région correspond à une lettre. Le SAX transformé obtenu est appelé mot (word en anglais). Quand la d'une série dépasse n alors une fenêtre de taille n traverse le long de la série et forme un ensemble de mots.

Soit $X = (x_1, ..., x_n)$ une série et $\bar{X} = (\bar{x_1}, ..., \bar{x_p})$ son PAA correspondant,

alors $\bar{x_i} = \frac{p}{n} \sum_{j=\frac{n}{m}(i-1)+1}^{\frac{n}{m}i} x_j$. Soit $B = \beta_1, ..., \beta_p$ l'ensemble ordonné des points de rupture, $\widehat{X} = (\widehat{x_1}, ..., \widehat{x_p})$ le SAX correspondant à la série X alors $\widehat{x_i} = alphabet_j$ et $\bar{x_i} \in [\beta_{j-1}, \beta_j)$. Soit $\widehat{A}$ et $\widehat{B}$ deux séries sous la forme SAX, la similarité entre ces deux séries est obtenue par la formule suivante :

$$Dist(\widehat{A}, \widehat{B}) = \sqrt{\frac{n}{p} \sum_{i=1}^{p} dist(\widehat{A}_i, \widehat{B}_i)} \qquad \text{où}$$

$$dist(a, b) = \begin{cases} 0 & si \quad |a - b| \leq 1 \\ B_{max(a,b)-1} - B_{min(a,b)-1} & sinon. \end{cases}$$

L'entropie est utilisée comme critère pour chercher la série qui sépare mieux les différentes classes après le calcul de la similarité entre cette série et le reste des séries. Après obtention de cette série la distance est utilisé comme critère de classification dans les quatre bandes en appliquant les algorithmes classique de la fouille de données.

Toutes les colonnes sont dans en pourcentage.

Application de SAX sans estimation des données manquantes sans hachage aléatoire

Algo	*Accuracy_Test*	*Rappel_Test_0*	*Rappel_Test_1*
AdaBoost	44.01261627285363	33.7636476000824	80.03861935795318
XGBoost	42.50507858441142	31.216095584700952	82.18682114409847
Decision Tree	46.34341922377846	38.99608597129712	72.16992517499396
Random Forest	42.55853736768952	31.772299663530866	80.47308713492637

SAX représentation sans estimation des données manquantes avec hachage aléatoire

Algo	*Accuracy_Test*	*Rappel_Test_0*	*Rappel_Test_1*
AdaBoost	48.85598203784882	45.58126759596237	60.36688390055516
XGBoost	49.395915748957556	46.57694156423814	59.30485155684286
Decision Tree	56.928258312840796	60.413376364760005	44.677769732078204
Random Forest	48.85063615952101	45.91087001304676	59.184166063239196

SAX représentation avec estimation des données manquantes avec hachage aléatoire

Chaque série est remplacée par les valeurs obtenues, en partant de son temps de début à son temps final et en se déplaçant à chaque fois d'une unité et ensuite soustraire ce temps de début à cet ensemble de temps ordonné obtenu, après

adaptation du modèle de [44] à cette série. Ce qui nous permet d'avoir les séries sur la même grille temporelle sans distorsion mais on a toujours des séries qui sont plus longues que d'autres.

Algo	*Accuracy_Test*	*Rappel_Test_0*	*Rappel_Test_1*
Decision Tree	64.1491679873217	73.47798340778557	31.822202565236623
XGboost	64.13431061806656	72.99936183790683	33.41441839893852
AdaBoost	65.18423137876387	74.71601786853861	32.15391419725785

4.3.2 La méthode basée modèle

Dans cette dernière expérimentation le modèle de [44] est utilisé en tant qu'un critère de séparation des classes et non pour une estimation des données manquantes. Puisque déjà les données de source contiennent des erreurs et qu'une estimation d'une valeur manquante par rapport aux autres entraine encore plus des erreurs. Après adaptation de ce modèle à une série, les paramètres $(A, B, t0, t_{fall}, t_{rise})$ sont gardés comme caractéristiques. Parmi ces cinq paramètres, il y a les quatre derniers de la bande i et ceux de la bande z qui séparent mieux les classes.

Algo	*Accuracy_Test*	*Rappel_Test_0*	*Rappel_Test_1*
XGboost	75.4135710747895	74.65162574651626	75.63341629970004
k plus proche voisin	66.20108964834075	79.27449679274497	62.42899993617972
Réseau de neurone	76.76572560673601	64.52112364521123	80.29867892016082
AdaBoost	73.49678058444775	78.7657597876576	71.97651413619248
Decision Tree	69.34621099554235	71.28953771289538	68.78550003191015
Random Forest	**76.3893016344725**	**79.03118779031188**	**75.62703427149148**
Régression logistique	69.172857850421	71.06834771068348	68.62594932669602

Conclusion

Dans cette deuxième partie du mémoire, nous avons présentons notre participation au challenge de classification des supernova. Nos meilleurs résultats ont été obtenus en appliquant la méthode basée modèle. Le modèle définie dans [44] pour estimer la courbe lumineuse des supernovas a été utilisée en tant que modèle de séparation des classes. D'autres méthodes ont été testées mais leurs résultats n'ont pas été très intéressants.

Conclusion et perspectifs

Appendix

Différentes requêtes

Pour les Galaxies

SELECT TOP 200000
p.objid, s.rowc_u, s.rowc_i, s.rowc_z, s.rowc_g, s.rowc_r, s.colc_u, s.colc_i, s.colc_z,s.colc_g, s.colc_r, p.rerun, p.run, p.camcol, p.field, p.z as redshift, p.class
FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid = p.objid
where p.class = 'GALAXIES'

Pour les Quazars

SELECT TOP 200000
p.objid, s.rowc_u, s.rowc_i, s.rowc_z, s.rowc_g, s.rowc_r, s.colc_u, s.colc_i, s.colc_z, s.colc_g, s.colc_r, p.rerun, p.run, p.camcol, p.field, p.z as redshift,p.class
FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid = p.objid
where p.class = 'QSO'

Pour les étoiles

SELECT TOP 200000
p.objid, s.rowc_u, s.rowc_i, s.rowc_z, s.rowc_g, s.rowc_r, s.colc_u, s.colc_i, s.colc_z, s.colc_g, s.colc_r, p.rerun, p.run, p.camcol, p.field, p.z as redshift, p.class
FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid = p.objid
where p.class = 'STAR'

Codes

https ://github.com/oussissa123/redshift_ estimation

Quelques figures

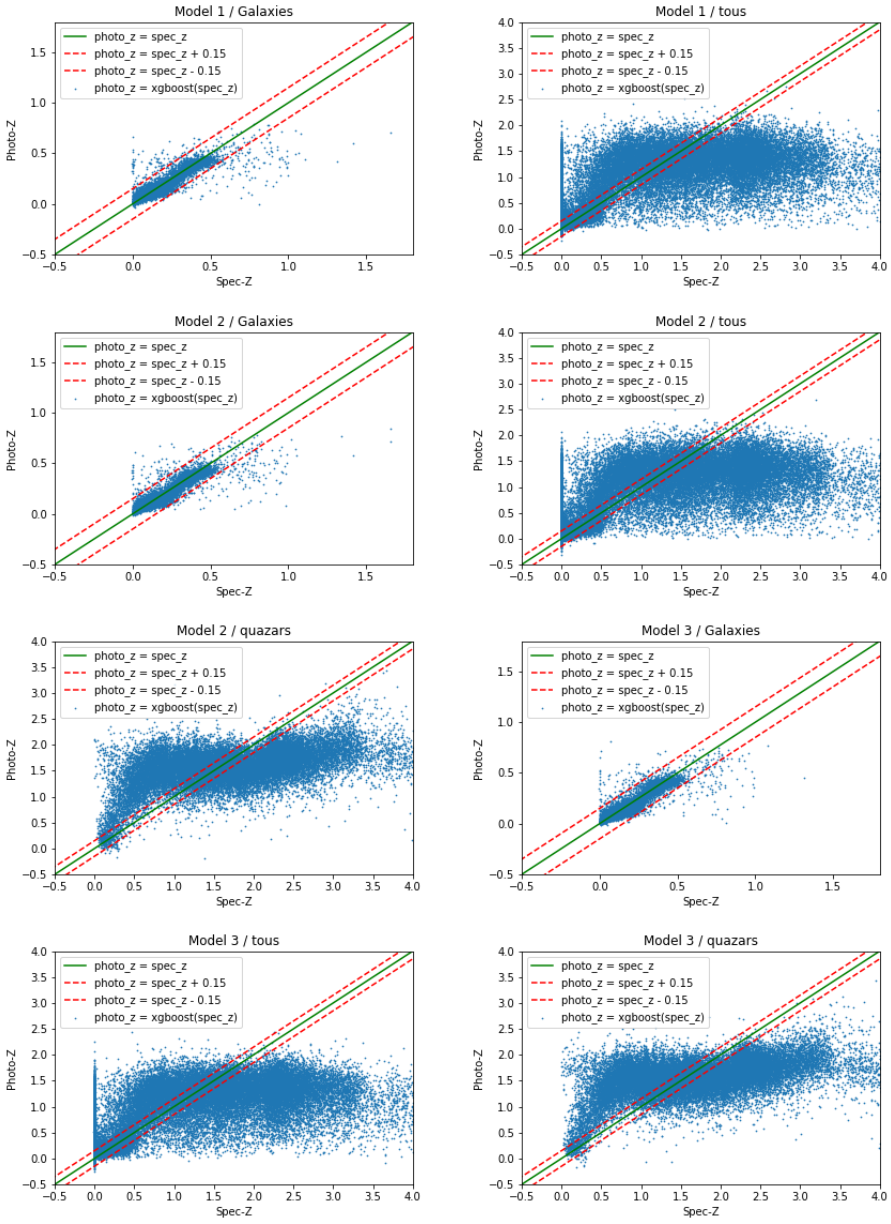

FIGURE 4.4 – Perturbation entre les valeurs spectroscopiques et photométriques à partir du modèle Boosting sur l'ensemble de test