Projet de Sciences des Données

Explotation d'images satellites haute-résolution
pour la prévision d'indicateurs socio-économiques

Youcef - Kacer

22 Novembre 2016

Table des matières

Introduction

Dans ce document, nous présentons les premiers résultats de régression et de classification des communes françaises, à partir de leur histogramme de $NDVI$. Nous cherchons ainsi à prédire la densité de population de chaque commune.
Nous verrons comment ont été selectionnées les scènes *Landsat-8* pour le territoire français à partir desquelles on extrait le $NDVI$ de chaque commune.
Ensuite, nous présenterons d'abord le problème d'apprentissage comme un problème de régression où on cherche à prédire la densité comme valeur continue, à partir des histogrammes de $NDVI$ des communes. Puis, nous simplifierons le problème en catégorisant la densité, nous ramenant ainsi à un problème de classification.
Les techniques d'apprentissage utilisées seront très variées et on s'attachera à les comparer entre elles via un critère de performance.

Chapitre 1

Données Landsat-8 pour la France métropolitaine

Une scène *Landsat-8* correspond à une certaine zone de la Terre couverte périodiquement (16 jours) par le satellite. Elle est identifiée par un *path* (typiquement entre 192 et 204 pour la France) et un *row* (typiquement entre 023 et 032 pour la France).
Nous avons récupéré des scènes Landsat-8 couvrant la France métropolitaine entre le 15 Mai 2013 et le 15 Septembre 2013.
La figure 1.1 montre le polygone de sélection des scènes sur l'interface du site de l'$USGS$ [Sur16].

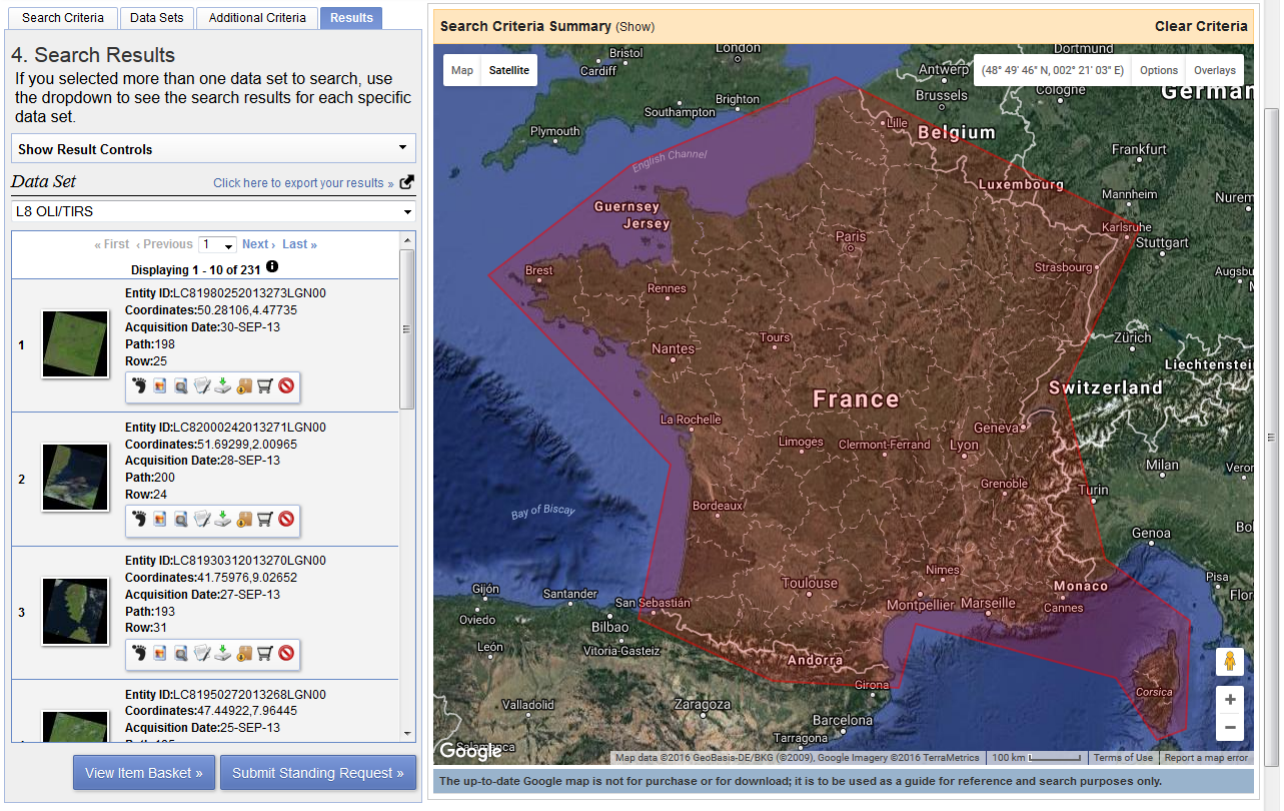

FIGURE 1.1 – Polygone de sélection des scènes *Landsat* – 8 pour la France sur l'interface du site de l'*USGS*

Cette période correspond à l'année des valeurs de densité de population à notre disposition pour l'ensemble des communes de France.
La période de Mai à Septembre est la plus courte permettant de couvrir tout le territoire métroplitain tout en conservant une occupation nuageuse inférieure à 20%. Nous obtenons ainsi 70 scènes Landsat-8 chacune correspondant à un couple $path$,row unique.
Les figures 1.2,1.3,1 et 1.4 montrent des miniatures couleurs des 4 scènes parmi les 70, ayant une couverture nuageuse supérieure à 10%

On voit donc que les scènes ayant une forte couverture nuageuse sont :

- soit des scènes contenant beaucoup de domaine maritime
- soit des scènes limitrophes de pays voisins.

La présence de nuage devrait donc avoir une faible impacte sur le $NDVI$ des communes françaises.

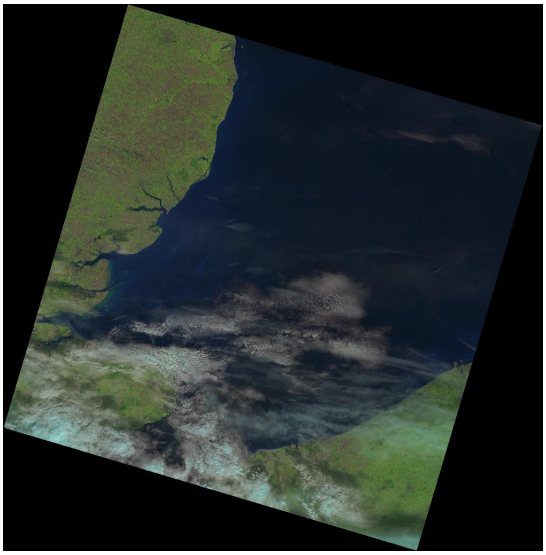

FIGURE 1.2 – Miniature couleur en projection Web Mercator (EPSG :3857) de la zone 200,024 (region Nord-Pas-de-Calais) - couverture nuageuse de 20.00%

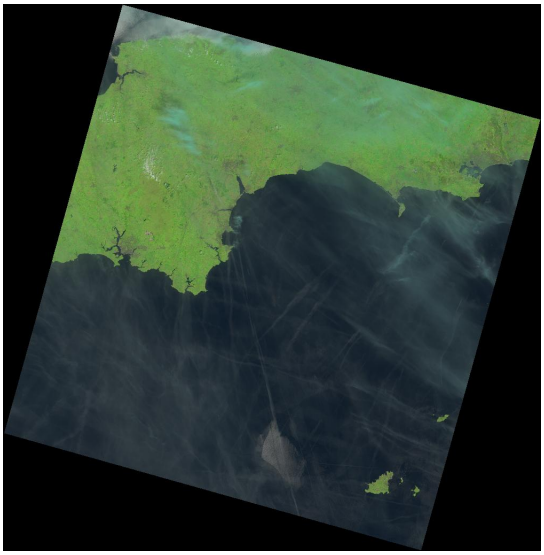

FIGURE 1.3 – Miniature couleur en projection Web Mercator (EPSG :3857) de la zone 203,025 (région des îles anglo-normandes) - couverture nuageuse de 18.30%

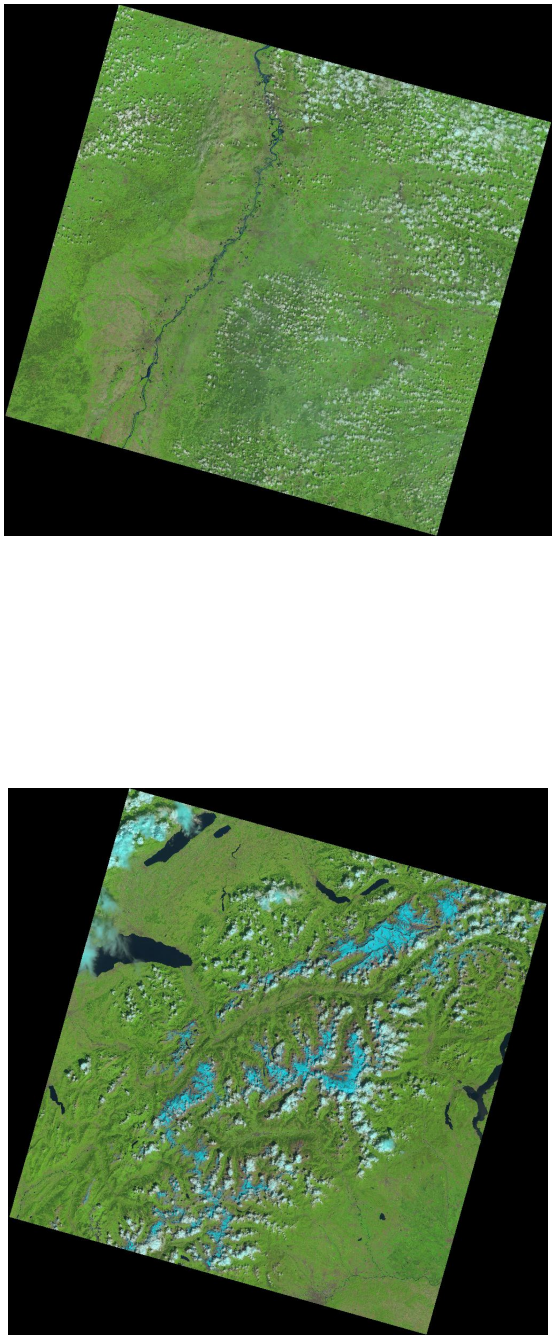

FIGURE 1.4 – Miniature couleur en projection Web Mercator (EPSG :3857) de la zone 203,028 (frontière franco-italo-suisse) - couverture nuageuse de 12.60%

la figure 1.5 présente toutes les scènes après projection en Web Mercator (EPSG :3857) (repère absolu).

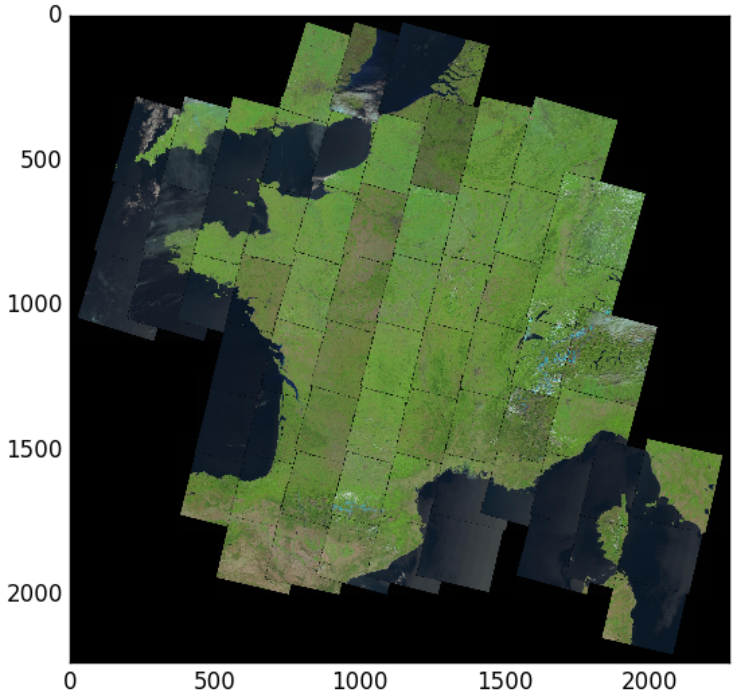

FIGURE 1.5 – Concaténation des 70 scènes Lansat-8 après projection en Web Mercator (EPSG :3857)

Chapitre 2

Extraction de l'histogramme de NDVI

Nous avons à notre disposition un fichier de 36700 comunes françaises contenant entre autres caractéristiques, les latitude et longitude en degrés, la surface em km^2, la densité de population tel que recensée par l'INSEE en 2013 [ins16].
Afin d'obtenir des longitudes et latitudes précises, nous les avons corrigés en utilisant une API *Python* de géolocalisation : *geopy* [geo06].

La méthode d'extraction d'information du NDVI pour chaque commune se fait comme suit :

- Pour une commune donnée, on projete ses latitude et longitude en Web Mercator. Les positions x,y obtenues permettent d'aller récupérer la scéne *Landsat-8* dont le centre est le plus proche de la commune. Prendre la scène la plus proche de cette manière, permet d'éviter que la commune échoue sur un bord non couvert par une scène.
- Puis, on découpe un carré de centre les coordonnées de la commune, et d'aire égale à la surface de la commune.
- On crée alors l'image de *NDVI* correspondante
- On calcule l'histogramme de l'image de *NDVI* en prenant 256 bins uniformément répartis dans l'intervalle [-1 1].
- On obtient ainsi un vecteur descripteur pour la commune
- On réitère le procédé pour chacune des communes

Nous présentons le procédé à travers l'exemple ci-après 2.1 :

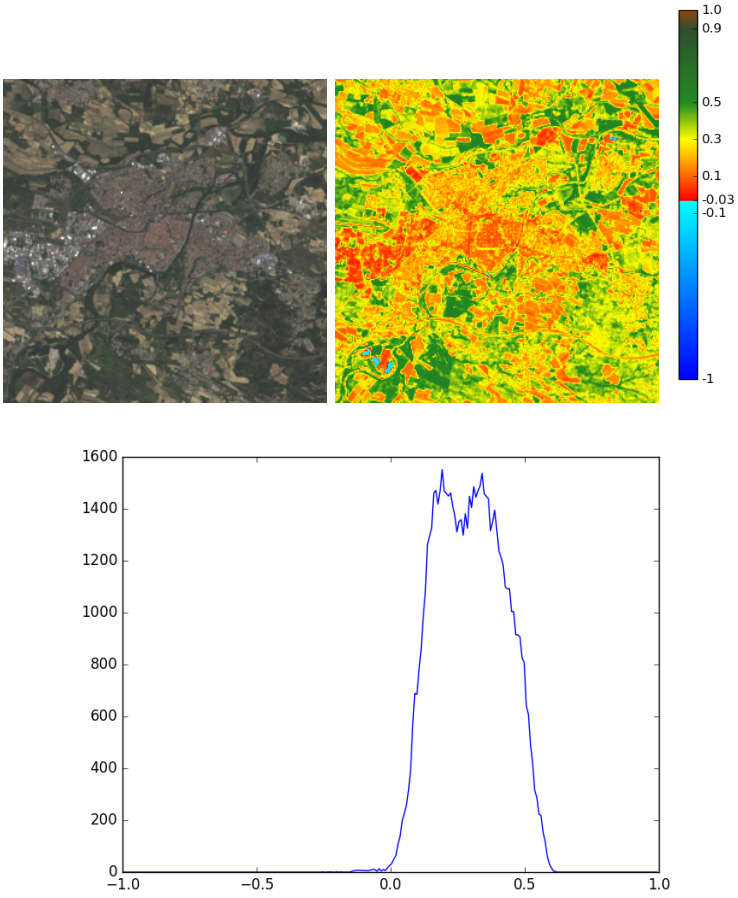

FIGURE 2.1 – Image couleur, image $NDVI$ et histogramme de $NDVI$ pour la commune de *Carcassonne* sur un périmètre de 65.08km^2 au mois de Mai 2013

Au final, nos données se présente sous la forme d'un tableau contenant à chaque ligne, l'histogramme d'une commune et sa densité. Avec un total de 36139 communes métropolitaines.2.1.

nom	bin-1	bin-2	...	bin-135	bin-136	...	bin-255	bin-256	**densité (habs/km^2)**
Ozan	0	0	...	1	5	...	0	0	**93.0**
Cormoranche-sur-saone	0	0	...	1	4	...	0	0	**107.0**
Paris	0	0	...	1953	1815	...	0	0	**21288.0**
Lyon	0	0	...	1099	1032	...	0	0	**460.0**
Tours	0	0	...	268	238	...	0	0	**3888.0**
Besancon	0	0	...	97	122	...	0	0	**1797.0**
...	...	...	...	...	...	...	...	...	**...**

TABLE 2.1 – Variables explicatives (histogramme de $NDVI$) et variable à prédire (densité) par régression, sous forme de tableau

Nous avons donc à présent des données sous forme de tableau prêt à être utiliser pour l'apprentissage supervisée.

Chapitre 3

Régression pour la prédiction de densité de population en fonction du NDVI

3.1 Erreur de généralisation d'un modèle de régression

Nous avons testé plusieurs algorithmes (ou modèles) de régression, du plus simple (régression linéaire) au plus élaboré (forêt aléatoire d'arbres régressifs). Chacun de ces algorithmes est évalué par cross-validation.
C'est-à-dire qu'on compose plusieurs sets à partir des échantillons de départ. Chaque set contient un sous ensemble aléatoire des échantillons pour l'entraînement de l'algorithme (70% du total), le reste des echantillons (30%) constitue l'ensemble de test et est utilisé pour calculer l'erreur commise par l'algorithme entrainé.
Nous avons donc après cross-validation, un nombre d'erreurs égale au nombre de sets, dont on considère la moyenne et la variance.
Cette moyenne est alors l'erreur d'ajustement aux données commise par l'algorithme. La variance mesure elle, la sensibilité de cette erreur à la fluctuation des données et donc la généralisation du modèle.
Ces deux valeurs ne peuvent être simultanément aussi basses qu'on le souhaite : un modèle qui s'ajuste parfaitement aux données aura toujours du mal à généraliser cette ajustement à des données nouvelles. Inversement, un modèle comportant un léger biais saura s'adapter aux données nouvelles.

Pour calculer l'erreur sur le sous-ensemble de test, nous utilisons le coefficient de détermination (souvent noté R^2) comme suit :

$$Erreur = 1 - R^2 = \frac{\sum_{i=1}^{n_{test}} (y_i - p_i)^2}{\sum_{i=1}^{n_{test}} (y_i - \widehat{y})^2} \tag{3.1}$$

avec :

- y_i, l'ensemble des valeurs à prédire pour les échantillons de test.
- p_i, l'ensemble des prédictions associées et calculées via le modèle entrainé.
- $\widehat{y} = \sum_{i=1}^{n_{test}} y_i$, la moyenne des échantillons de test

Une telle erreur peut-être potentiellement supérieur à 1, auquel cas le modèle s'ajuste très mal aux données (il fait pire que l'estimateur constant de moyenne). A l'opposé, plus l'erreur est proche de 0, mieux il s'ajuste aux données.

3.2 Comparatif de plusieurs algorithmes régressifs

Nous avons utilisé la librairie *Scikit-learn*[PVG+11] sur *Python* pour l'implémentation de différents modèles régressifs. Le module *GridSearchCV* permet de lancer la cross-validation d'un modèle de manière très simple.
A noter que plus un algorithme est élaboré, plus il comporte d'hyperparamètres à régler. A titre d'exemple, la régression pénalisée (Ridge) ne comporte qu'un seul hyperparamètre, le coefficient de pénalisation. Alors que la forêt d'arbres aléatoires en comporte plus d'une dizaine (nombre d'arbres, profondeur des arbres, nombre de variables sélectionnées,...).
L'erreur de généralisation que nous présentons pour chaque modèle est celle qui utilise la meilleure combinaison d'hyperparamètres.
Le nombre de sets pour la cross-validation a été fixé 5.

Le tableau 3.1 résume les résultats obtenus.

On constate que le NDVI n'explique pas du tout la densité de manière linéaire, au vu des très mauvais scores obtenus pour la régression linéaire.
Les Machines à Support de Vecteurs font aussi bien que l'estimateur de moyenne.
Les arbres de décision permettent de diminuer l'erreur mais ne premettent toujours pas d'expliquer la densité.
Comme nous le verrons dans la prochaine partie, la distribution de la densité comporte un petit nombre de valeurs très hautes, il serait donc intéressant de chercher à prédire le logarithme de la densité d : $log(\alpha + d)$ avec α, une constante strictement positive à régler.
Cependant, nous allons dans la prochaine partie laisser de côté le problème de la régression et simplifier le problème en nous rapportant à un problème de classification.

Régression	
Modèle	Erreur de généralisation
Régression	> 1
Régression pénalisée	> 1
Support Vector Régression à noyau gaussien	1
Réseau de neurones régressif	0.822
Forêt d'arbres aléatoires régressifs	0.800
Boosting d'arbres aléatoires régressifs	0.739

TABLE 3.1 – Erreur de généralisation pour différents modèles régressifs

Chapitre 4

Classification pour la prédiction de densité de population en fonction du NDVI

Les résultats de régression présentés au chapître précédent sont très perfectibles malgré l'utilisation d'algorithmes très puissants qui ont déjà fait leur preuve sur un certain nombre de challenges ($Kaggle$,$Hackathon$). Dès lors, on peut se demander si une simplification du problème permettrait de mieux lier le $NDVI$ à la densité. La régression ayant eu du mal à s'ajuster au données continue de densité, nous allons catégoriser celle-ci afin de nous ramener à un problème de classification.

4.1 Catégorisation

Il s'agit de découper l'intervalle de densité en une partition d'intervalles. La multi-binarisation d'Otsu permet de créer une telle partition en minimisant la variance au sein de chaque partition, tout en maximisant la variance inter-partition.
La figure 4.1 présente l'histogramme de densité de nos échantillons :

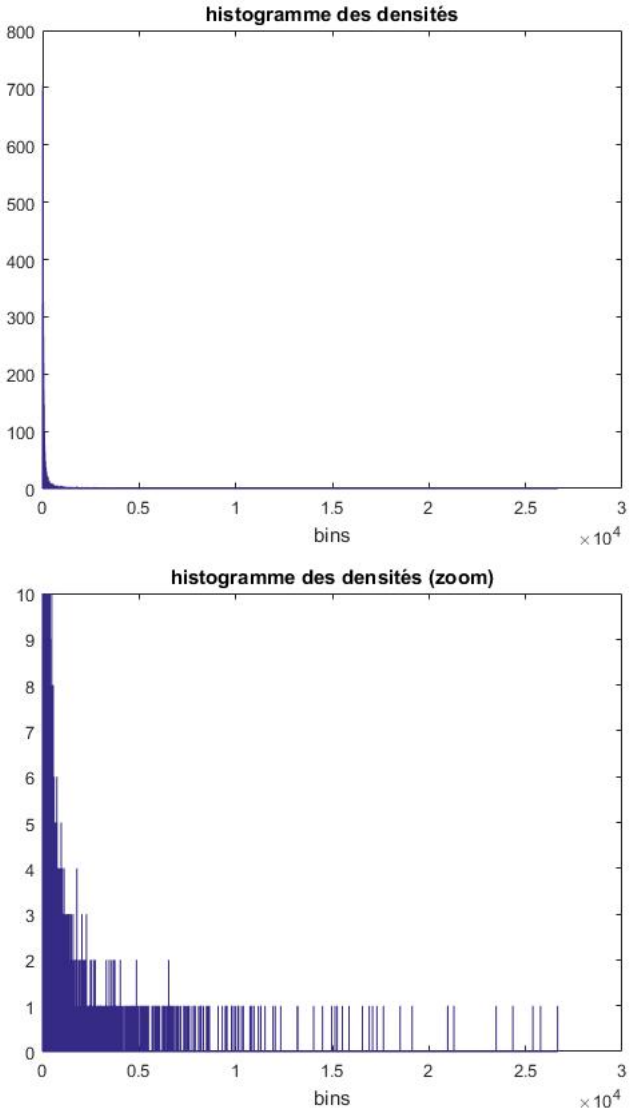

FIGURE 4.1 – Histogramme des densités et son zoom

On applique alors la méthode d'Otsu (*Matlab*) sur l'histogramme pour différentes valeurs en nombre de catégories (ou nombre de clusters si on voit la méthode comme une méthode de segmentation). On obtient à chaque fois un score reflétant la compacité des catégories créées. La figure 4.2 montre l'évolution du score pour différentes nombre de catégories :

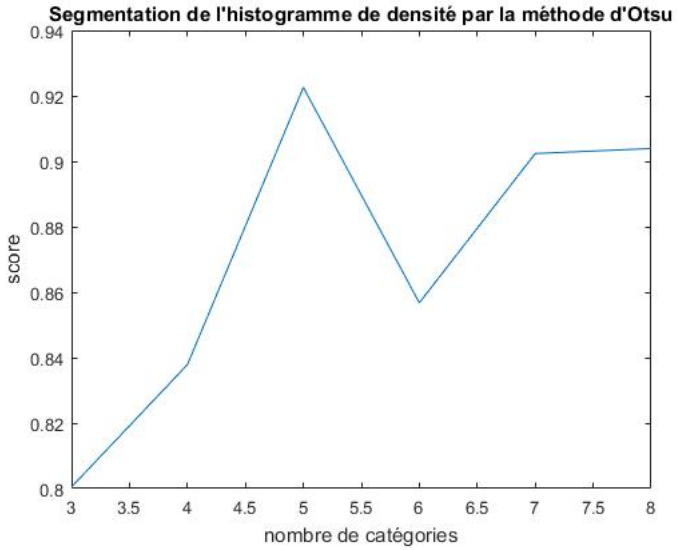

FIGURE 4.2 – Score de catégorisation par Otsu pour différents nombre de catégories

On remarque que le score est maximal pour un nombre de catégories de 5.
La figure 4.3 montre les 4 seuils obtenus pour la segmentation à 5 catégories.

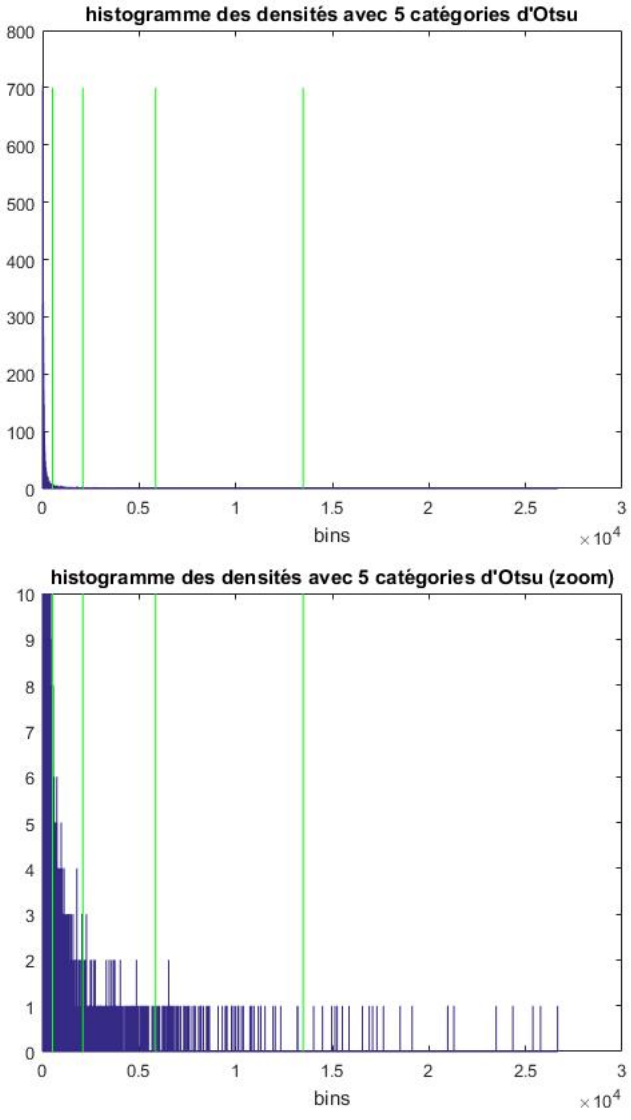

FIGURE 4.3 – Histogramme des densités et son zoom avec seuils d'Otsu pour 5 catégories

Les 4 seuils obtenus sont par ordre croissant : 523,2091,5855 et 13487.
Ainsi, en arrondissant ces valeurs, on créé 5 catégories pour nos échantillons :

catégorie 1 : densité comprise entre 0 et 500
catégorie 2 : densité comprise entre 500 et 2000
catégorie 3 : densité comprise entre 2000 et 5000
catégorie 4 : densité comprise entre 5000 et 13000
catégorie 5 : densité supérieure à 13000

Un tel découpage donne lieu à une distribution avec une grande disproportion entre les catégories, ce qui peut potentiellement perturber certains algorithmes de classification :

catégorie 1 : 20 échantillons

catégorie 2 : 102 échantillons

catégorie 3 : 286 échantillons

catégorie 4 : 1287 échantillons

catégorie 5 : 34443 échantillons

D'où le nouveau tableau obtenu en tenant compte cette fois de la catégorie comme variable à prédire 4.1 :

nom	bin-1	bin-2	...	bin-135	bin-136	...	bin-255	bin-256	**densité (catégorie)**
Ozan	0	0	...	1	5	...	0	0	**1**
Cormoranche-sur-saone	0	0	...	1	4	...	0	0	**1**
Paris	0	0	...	1953	1815	...	0	0	**5**
Lyon	0	0	...	1099	1032	...	0	0	**1**
Tours	0	0	...	268	238	...	0	0	**3**
Besancon	0	0	...	97	122	...	0	0	**2**
...	...	...	...	...	...	...	...	...	**...**

TABLE 4.1 – Variables explicatives (histogramme de $NDVI$) et variable à prédire (densité) par classification, sous forme de tableau

4.2 Erreur de généralisation d'un modèle de classification

L'idée est exactement la même que pour la régression à ceci près que la formule de calcul de l'erreur sur le sous-ensemble de test d'un set donné, est différente. En effet, les valeurs à prédire étant à présent catégorisées, on utilise l'erreur de précision dont la formule est décrite ci-après :

$$Erreur = \frac{\sum_{i=1}^{n_{test}} (y_i \neq p_i)}{n_{test}} \tag{4.1}$$

avec :

- y_i, l'ensemble des valeurs à prédire pour les échantillons de test.
- p_i, l'ensemble des prédictions associées et calculées via le modèle entrainé.

4.3 Comparatif de plusieurs algorithmes régressifs

Ici encore, nous présentons les erreurs de classification pour différents modèles, chaque modèle étant utilisé avec sa meilleure combinaison d'hyperparamètres.

Le tableau 4.2 résume les résultats obtenus.
Le boosting d'arbres aléatoires suffit à très bien expliquer la relation entre le NDVI et la densité catégorisée (4,3% d'erreur en généralisation)
.

Classification	
Modèle	Erreur de généralisation
Boosting d'arbres aléatoires	0.043

TABLE 4.2 – Erreur de généralisation pour différents modèles de classification

Liste des tableaux

Table des figures

Bibliographie

[geo06] geopy is a python 2 and 3 client for several popular geocoding web services., 2006.

[ins16] Données harmonisées des recensements de la population à partir de 1968, 2016.

[PVG+11] 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. https ://pypi.python.org/pypi/utm Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[Sur16] U.S. Geological Survey. `http://eros.usgs.gov/`, 2016.