Compte-rendu de projet de SY09

Aïda El Mernissi, Gabriel Souza e Silva, Gautier Daures

12 juin 2020

Étude des données de la classification d'images satellites

Résumé

Cette étude a pour but de discuter et de conclure sur le meilleur modèle permettant de classifier les données d'images satellites provenant du dépôt de données numériques de l'UCI [2]. Une comparaison de plusieurs modèles de décision est présentée ainsi que quelques méthodes de sélection de variables dans l'optique d'améliorer la précision des modèles étudiés.

1 Introduction

1.1 Problématique et cadre de l'étude

Le but du projet est d'appliquer des méthodes de classification automatique sur notre jeu de données, d'analyser et de comprendre leurs performances. Les données dont on dispose sont des mesures dérivées d'images satellites provenant de deux sources : d'une part des images satellites temporelles, collectées entre 2014 et 2015, d'autre part des données terrestres open-source provenant d'OpenStreetMap [1]. Ces données ont été collectées par l'UCI [2] et sont disponibles sur leur site. Différents résultats seront exposés à partir des traitements effectués et un approfondissement sera fait concernant la sélection des variables. Nous pouvons donc aboutir à la problématique suivante :

Quel modèle de classification semble le plus adapté dans le cadre de notre jeu de données et en quoi appliquer une sélection des variables permettrait d'améliorer la précision des prédictions ?

1.2 Les données

Les données sont réparties en deux fichiers :

— un fichier *training.csv* de 10545 lignes et 29 colonnes. Le fichier est décrit comme contenant beaucoup de bruit et est destiné à servir d'ensemble d'apprentissage ;

— un fichier *testing.csv* de 300 lignes et 29 colonnes. Il ne contient pas de bruit et est destiné à servir d'ensemble de test.

Les individus sont donc les images satellites et les variables les décrivant sont de type quantitatif. Elles correspondent à des mesures d'un indice de végétation appelé NDVI[3] réalisées à différentes dates. Chaque variable a un nom au format *<date de la mesure>_N*. De plus, nous disposons d'une variable *max_ndvi* qui correspond au maximum des NDVI mesurés. Nous trouvons enfin la colonne des étiquettes, nommée *class*. La période de mesure s'étend du 01/01/2014 au 20/07/2015, à des intervalles de temps irréguliers. De manière générale, le bruit correspond à des imprécisions, des défauts, des approximations dans les mesures ou encore des données manquantes car non mesurées. Notre jeu de données ne comporte pas de valeurs manquantes, mais le bruit peut être provoqué, par exemple, par le passage de nuages faussant ainsi la valeur du NDVI. Notons que le NDVI est habituellement compris entre -1 et 1 mais que les données fournies ont une amplitude plus large, de l'ordre de -10^3 à 10^3.

2 Démarche

2.1 Analyse exploratoire

Dans un premier temps, nous avons effectué quelques analyses sur l'ensemble d'apprentissage, afin de mieux comprendre nos données. Nous avons vérifié la structure des données pour voir si elles étaient conformes à la documentation. Nous avons également vérifié le type des données et leur intégrité pour voir s'il n'y avait pas de valeurs manquantes. Après avoir constaté la bon format des données et l'absence de valeurs manquantes, nous sommes passés à l'analyse des variables. Tout d'abord, le comptage des classes, représenté sur la figure 1, montre que plus de 70% des valeurs sont étiquetées comme *Forest* et nous avons très peu de données d'autres classes. Ce déséquilibre peut poser problème

par la suite, car il sera plus difficile pour un modèle de prédire une classe qui apparaît peu dans l'ensemble de test.

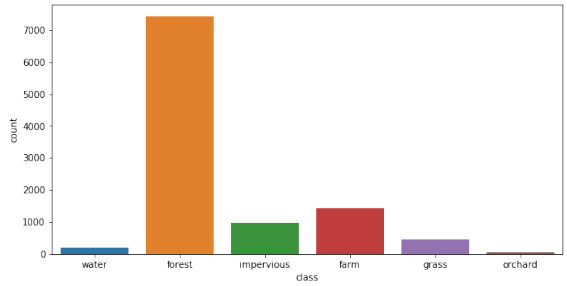

FIGURE 1 – Distribution des classes

De plus, la figure 2 confirme que les variables comportent beaucoup de bruit. Les fractiles sont très différents selon les variables, alors que celles ci mesurent le même indice. Les données extrêmes sont présentes en très grand nombre, comme cela a été décrit dans la documentation du jeu de données.

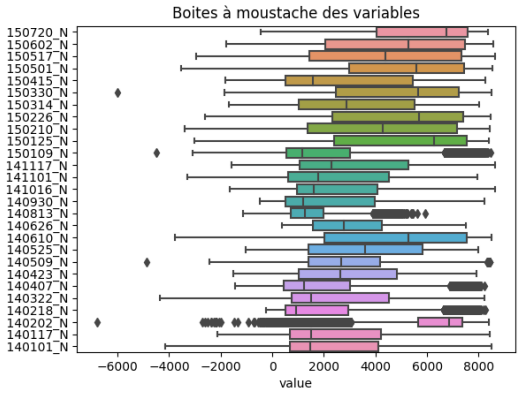

FIGURE 2 – Diagrammes en boite des variables

Nous avons également cherché des relations entre la variable *class* et les autres variables. Nous savons, d'après nos recherches sur le *ndvi*, que les nuages sont la source du bruit dans ce type de données et que le bruit a tendance à diminuer la valeur du *ndvi*. Nous faisons donc l'hypothèse que chaque classe est fortement corrélée aux valeurs de *max_ndvi*. Cela semble raisonnable, étant donné que le bruit réduit le *ndvi* et que, par conséquent, les valeurs maximales du *ndvi* caractérisent plus précisément le type de végétation en question.

Pour explorer cette hypothèse, nous avons tout d'abord regardé la distribution des *max_ndvi* par classe, comme présenté sur la figure 3. Nous remarquons que les courbes de distribution se comportent selon la définition de l'indicateur *ndvi*. En effet, nous constatons que les classes représentant un type de végétation vert, c'est à dire *Forest*, *Grass*, *Orchard* et *Farm* sont situées autour de valeurs élevées de *ndvi*, alors que les classes *Water* et *Impervious*, qui représentent des régions peu vertes, sont situées autour de valeurs plus faibles. Cette hypothèse est par conséquent intéressante pour notre étude. Elle doit être gardée en tête pour la classification.

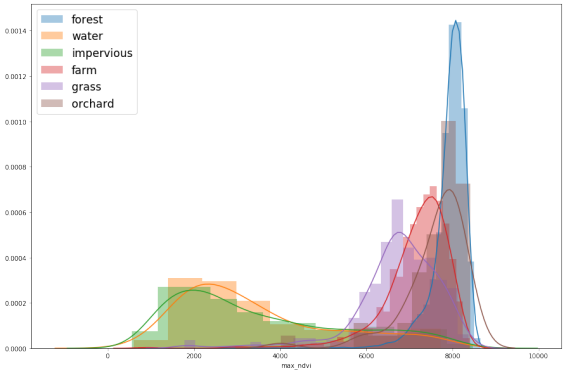

FIGURE 3 – Distribution de max_ndvi par classe

Il est également intéressant d'étudier la corrélation entre les variables, pour vérifier leur indépendance. En effet, certaines variables pourraient contenir des informations redondantes. La figure 4 permet de détecter une éventuelle dépendance entre variables. Cependant, nous constatons qu'aucune paire de variable ne présente une relation de corrélation évidente.

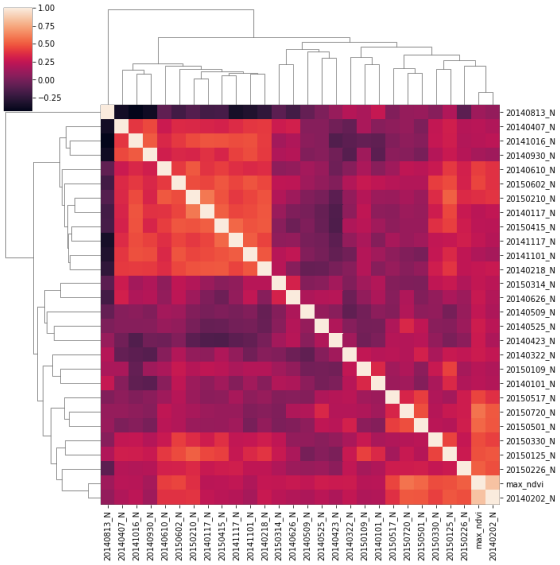

FIGURE 4 – Distribution de max_ndvi par classe

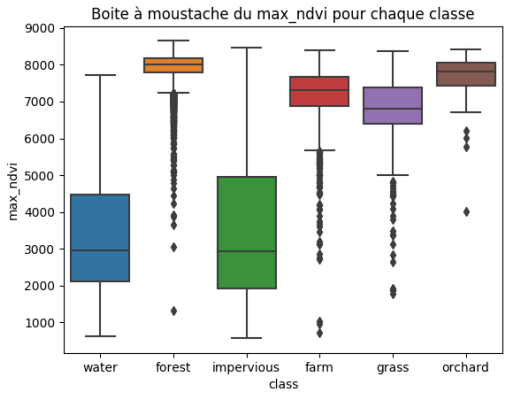

FIGURE 5 – Diagramme en boîte de la répartition du max_ndvi suivant la classe

Pour terminer cette analyse exploratoire, nous avons trouvé intéressant d'appliquer un test de Student à chaque paire de classes pour vérifier si les moyennes de *max_ndvi* étaient significativement différentes les unes des autres. D'après le test, nous constatons effectivement que les moyennes pour chaque classes sont significativement différentes, excepté entre les classes *Water* et *Impervious*.

2.1.1 Corrélation entre les classes *Impervious* et *Water*

Grâce aux figures 3, 7 et 5, nous voyons assez clairement que les classes *Impervious* et *Water* ont des distributions très similaires (même médiane et étendue interquartiles qui s'emboîtent l'une dans l'autre). Nous voulions donc savoir si le fait de fusionner ces deux classes aurait un effet significatif sur la performance d'un algorithme de clusterisation tel que le KMeans.

Nous donc effectué deux tests du KMeans distincts :
- Un premier en prenant l'ensemble d'apprentissage tel qu'il est et en formant donc 6 clusters,
- Un deuxième en renommant l'information de classe de tous les individus appartenant à "water" ou à "impervious" par "joined" et en formant cette fois 5 clusters.

Les résultats obtenus sont résumés dans le tableau 1.

Ainsi, on voit que la fusion des classes "water" et "impervious" dans le but de réduire l'erreur des KMeans n'augmente que de très peu la précision de cette méthode. En effet, la forme très agglomérée des classes ne permet pas de tracer des frontières claire entre celles-ci grâce aux KMeans.

TABLE 1 – Résultats des KMeans avec un nombre de clusters différents

Donnés	Nombre de clusters	Précision
5 classes	apprentissage	23,29%
	test	37,20%
6 classes	apprentissage	20,79%
	test	36,45%

2.2 Méthodes d'apprentissage non supervisées

Nos données se prêtent évidement à une analyse supervisée, puisqu'on nous fournit explicitement un ensemble de test et un ensemble d'entraînement. Cependant, nous avons voulu explorer les méthodes d'apprentissage non supervisées, car elle peuvent nous permettre d'éventuellement tirer de nouvelles informations sur notre jeu de données. Nous avons décidé de ne pas regrouper nos deux ensembles, afin de ne pas biaiser les résultats vis à vis des méthodes supervisées et de rester cohérent tout au long de notre étude.

2.2.1 Analyse en composantes principales

Prendre simultanément en compte toutes les variables de notre jeu de données constitue un problème complexe. L'analyse en composantes principales (ACP) consiste donc à remplacer les variables initiales par un nombre réduit de nouvelles variables, en essayant de li-

miter au maximum la perte d'informations. L'idée est de réussir à représenter les données pour distinguer visuellement les classes. La figure 6 représente d'une part l'inertie expliquée selon les 10 premiers axes factoriels (graphique en barres bleu) et d'autre part le cumul des inerties (courbe rouge). Nous pouvons constater que le premier axe explique presque 30% de l'inertie totale, tandis que tous les autres représentent moins de 10%. En prenant 10 axes d'inertie, plutôt que les 28 variables initiales, nous parvenons à expliquer environ 70% de l'inertie totale. Si nous appliquions la méthode du coude, nous pourrions conserver uniquement les 2 premiers axes d'inertie. Cependant, en raison du grand nombre de variables, conserver seulement les 2 premiers axes constituerait une perte d'information trop conséquente.

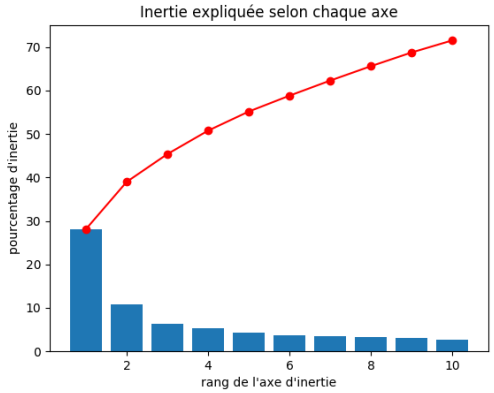

FIGURE 6 – Inertie expliquée par les 10 premiers axes de l'Analyse en Composantes Principales

La figure 7 représente les données selon les deux premiers axes d'inertie. Elle confirme qu'il n'est pas possible de distinguer convenablement nos classes seulement selon ces axes, puisque les points sont très regroupés, bien que nous puissions assez bien distinguer les classes *Water* et *Impervious* du reste des classes. Elle permet également de se rendre compte visuellement de l'omniprésence de la classe *Forest*.

Le cercle des corrélations 8 permet de se rendre compte que les variables sont toutes corrélées négativement avec le premier axe d'inertie, et distribuées de manière assez équilibrée selon le second axe. Aucune n'est cependant très corrélée, au premier ou au second axe (les flèches sont proches du centre du cercle).

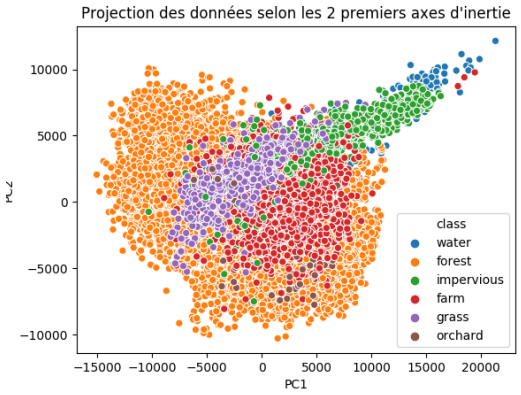

FIGURE 7 – Représentation de l'ACP selon les 2 premiers axes d'inertie

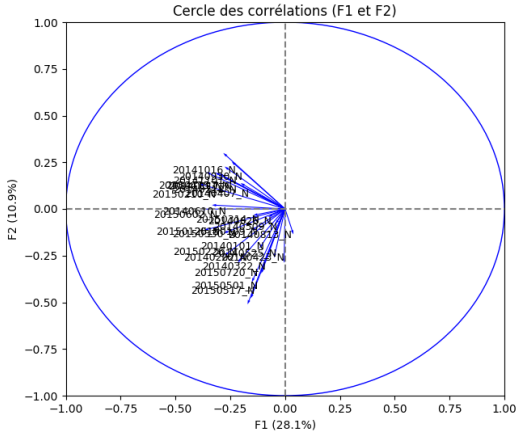

FIGURE 8 – Cercle de corrélation des variables aux 2 premiers axes d'inertie

2.2.2 Classification ascendante hiérarchique

La classification ascendante hiérarchique (CAH) permet de construire une hiérarchie indicée d'un ensemble, en partant à l'origine de singletons. Nous utilisons le critère d'agrégation de Ward muni de la métrique euclidienne sur les données d'entraînement, en restreignant à 6 clusters, correspondant au nombre de classes. Nous testons également pour 5 clusters, pour vérifier qu'aucune paire de classe ne se dégage. Les résultats sont consignés dans le tableau 2. Nous effectuons une prédiction sur les données de test, car bien qu'il s'agisse d'une méthode non supervisée, une prédiction est possible. L'indice de Rand valant 1 si les partitions sont identiques, nous constatons que la CAH est peu efficace

dans le cadre de notre jeu de données.

TABLE 2 – Résultat de la CAH selon le nombre de clusters

Clusters	Donnés	Indice de Rand ajusté
5 classes	apprentissage	0,08
	test	0,28
6 classes	apprentissage	0,13
	test	0,35

3 Méthodes supervisées

Nos données d'apprentissage étant fortement bruitées, contrairement aux données de test, nous pourrions être tentés de mélanger les 2 jeux de données, puis de reconstruire 2 ensembles plus homogènes. Nous avons décidé de ne pas prendre ce parti pris.

3.1 Méthodes utilisées

Dans cette partie, nous avons décidé de nous limiter aux méthodes vues en cours. Nous étudierons donc les méthodes suivantes :

- Analyse Discriminante Linéaire (ADL)
- Analyse Discriminante Quadratique (ADQ)
- Classificateur de Bayes
- *K-nearest neighbors (KNN)*
- Arbres de décision
- Régression logistique multinomiale

Nous pensons disposer de suffisamment de méthodes pour effectuer une bonne analyse et choisir un modèle de classification final. Parmi ces méthodes, nous en avons des plus simples comme le *KNN* et des plus complexes comme les arbres de décision. Nous ne prenons pas en compte la robustesse du modèle, mais nous recherchons un modèle qui convient le mieux à notre problème.

Nous savons que chaque méthode a ses propres caractéristiques et nous voulons d'abord explorer les méthodes avec leurs paramètres par défaut puis approfondir une méthode de notre choix. Aussi, nous partons du principe que les hypothèses de chaque modèle sont respectées. Pour l'analyse discriminante linéaire, par exemple, on fait l'hypothèse d'homoscédasticité. Nous savons que de manière générale, les modèles d'analyse discriminante ont pour but de réduire le nombre de paramètres à estimer. Dans ce cas, il est souhaitable d'avoir un modèle de complexité adaptée à la taille de l'ensemble.

Nous avons alors décidé qu'une méthode intéressante pour faire la sélection de modèle est la validation croisée. En général, nous trouvons intéressant d'appliquer cette méthode, car nous pouvons analyser les performances de chaque méthode de classification pendant la phase d'apprentissage. Donc, il est possible de rechercher des indications qui montrent comment le modèle se comporte dans différents sous-ensembles. Nous cherchons dans un premier temps à tester les méthodes avec leurs paramètres par défaut puis voir ensuite si l'une des méthodes aboutit à de bonnes performances.

3.2 Sélection de méthodes

Pour choisir une famille de modèles, nous avons réalisé une validation-croisée afin d'examiner la performance moyenne des méthodes. À l'aide des fonctions *ShuffleSplit* et *learning_ curve* du module python *Scikit-learn* nous avons construit une fonction pour évaluer la performance des méthodes avec des ensembles de différents tailles. Notre évaluation consiste à effectuer une validation croisée avec 5 sous-ensembles de données de tailles variant entre 10% et 100% des données d'apprentissage. Nous réalisons une validation croisée avec 10 *splits* et une taille de test égale à 0.2, donc pour chaque sous-ensemble évalué, on obtient la moyenne des scores de train et de test. Pour résumer, notre processus d'évaluation consiste en la méthode suivante, également décrite dans la table 9.

- Pour chaque méthode M_i :
  - Pour chaque sous-ensemble S_i :
    - Faire la validation-croisée avec M_i et S_i
    - Prendre la moyenne des résultats et la variance

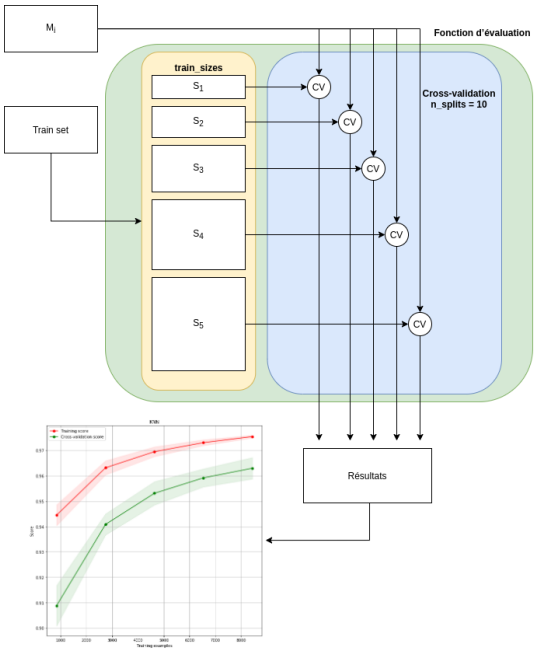

FIGURE 9 – Processus d'évaluation des méthodes

À la fin de l'évaluation, nous obtenons la moyenne finale des scores de validation croisée (3).

TABLE 3 – Résultats validation-croisée

Méthode	Score de test final
LDA	0.88
QDA	0.94
GaussianNB	0.84
KNN	0.96
DecisionTreeClassifier	0.89
LogisticRegression	0.85

Nous voyons qu'en moyenne les méthodes arrivent à bien prédire les données d'entraînement, mais le *k-nearest neighbors* se démarque. Cependant, ce processus donne une estimation biaisée du modèle, mais nous pouvons nous baser sur lui pour le choix de notre modèle, car cela nous permet de comprendre le comportement des classifieurs avec notre jeux de données. Afin d'explorer un peu plus les résultats d'entraînement, nous représentons les résultats de validation-croisée pour chaque méthode.

Ci-dessous, nous pouvons distinguer ces représentations (10, 11, 12, 13, 14, 15). En rouge nous avons le score d'entraînement et en vert le score de validation-croisée (ou score de test). La ligne centrale est la moyenne des scores et l'intervalle autour de cette ligne est la moyenne plus ou moins la variance de chaque validation.

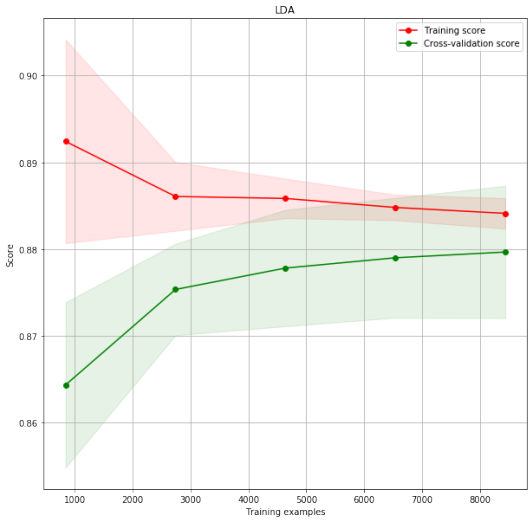

FIGURE 10 – Analyse Discriminant Linéaire

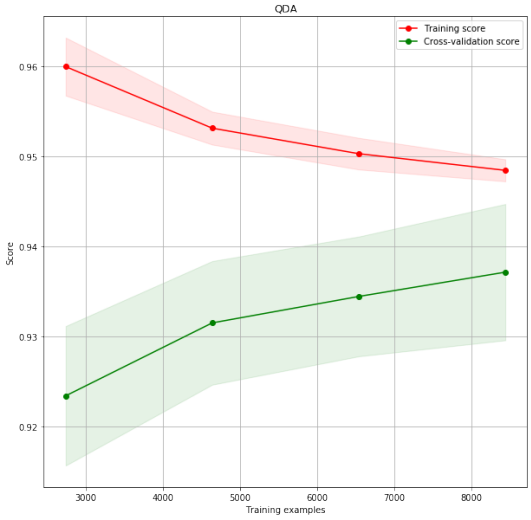

FIGURE 11 – Analyse Discriminant Quadratique

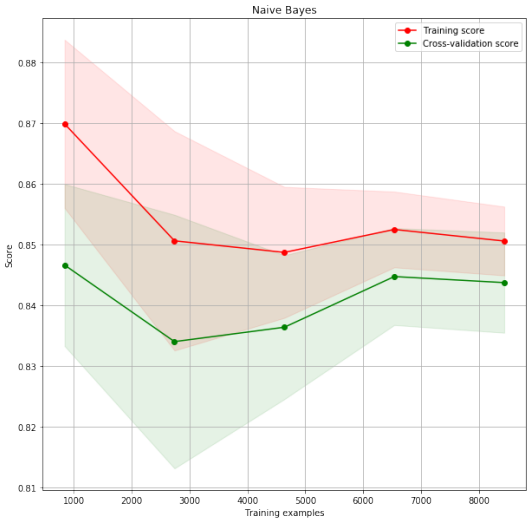

FIGURE 12 – Classifieur Bayes

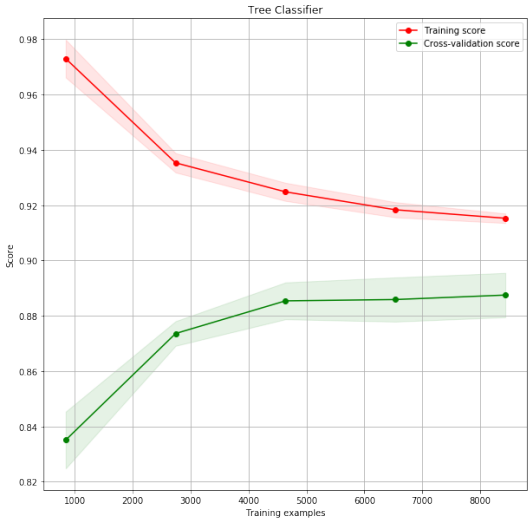

FIGURE 14 – Arbres de décision

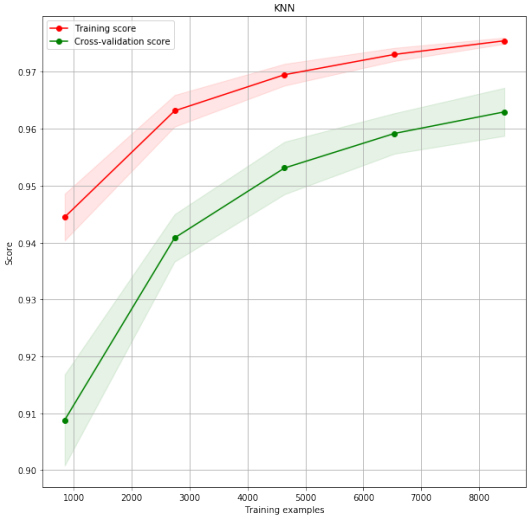

FIGURE 13 – *k-nearest neighbors*

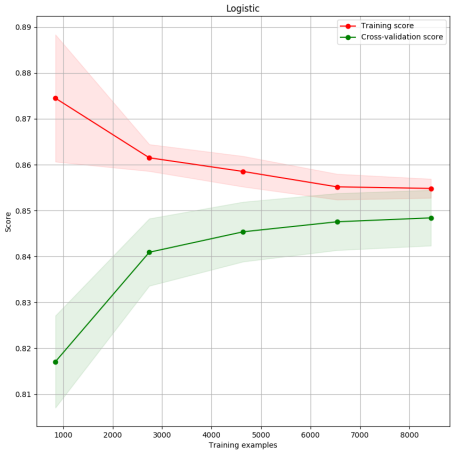

FIGURE 15 – Régression logistique

Nous pouvons voir que la méthode *KNN* (13) semble être un bon modèle pour le problème. En effet, les scores de test et de validation croisée approchent 1.0, la courbe de validation-croisé approche la courbe de test, et les deux courbes sont croissantes. Cela montre qu'en moyenne cette méthode est performante avec les données. La bande vert est réduite, preuve d'une variance assez faible. Ainsi, le *KNN* est peu sensible à la modification des données.

Les scores de test des autres méthodes sont globalement bons, mais elles présentent une grande variance, en particulier pour le classifieur de Bayes (12). Cela ne

permet pas de savoir si les méthodes peuvent bien généraliser les données. La méthode des arbres de décision (14) a elle une variance faible, mais une performance moins bonne que le *KNN*.

Finalement, nous avons décidé d'entraîner tous nos modèles avec l'ensemble de test complet et de vérifier leurs performances globales avec cet ensemble. Les résultats sont présentés dans le tableau 4.

TABLE 4 – Résultats de test

Méthode	Précision
LDA	0.59
QDA	0.55
GaussianNB	0.69
KNN	0.62
DecisionTreeClassifier	0.55
LogisticRegression	0.58

Nous observons que les méthodes ont presque toutes la même précision lorsqu'elles sont évaluées dans l'ensemble de test (compris entre 0.55 et 0.69). Cependant, cet ensemble ne comprend que 300 valeurs, ce qui peut ne pas être suffisant pour décider si un modèle est vraiment plus efficace que les autres. C'est pourquoi lors du choix de notre modèle, nous prenons en compte le comportement des modèles dans la phase d'apprentissage. Nous avons remarqué qu'en variant le volume des données d'apprentissage, certains modèles ont tendance à améliorer leurs performances et d'autres à empirer, de même certains modèles ne varient pas beaucoup avec le changement de données. Ces points sont importants pour déterminer notre modèle final, car cela nous indique une certaine capacité du modèle à généraliser nos données.

D'après notre analyse exploratoire nous avons aussi trouvé un autre point intéressant à explorer qui concerne la distribution des variables du jeu de données. Nous avons vu que plus de 70% des données d'entraînement sont étiquetées comme *Forest* est seulement 0,5% des données sont du type *Orchard*, ce qui compte pour 53 individus sur plus de 10,000. Au contraire, les classes sont réparties de manière très homogènes dans l'ensemble de test (16). Nous avons donc décidé d'explorer ce point et vérifier si avec un jeu de données avec le même nombre d'individus donne de meilleurs résultats. D'abord nous effectuons plusieurs échantillonnages, car nous avons beaucoup plus de données du type *Forest*. En général, à la fin, dans la partie apprentissage, les méthodes ont presque la même performance qu'avec l'ensemble complet d'entraînement. De plus, lorsque nous prédisons avec les données de test, ils approchent également les modèles entraînées avec l'ensemble complet d'entraînement. Néanmoins, la fonction *classification_report* permet de visualiser les performances de la classification pour chaque classe et nous remarquons que les méthodes n'arrivent pas à prédire correctement la classe *Orchard*. Pour cette raison, nous restons sur la première approche avec toutes les données d'entraînement.

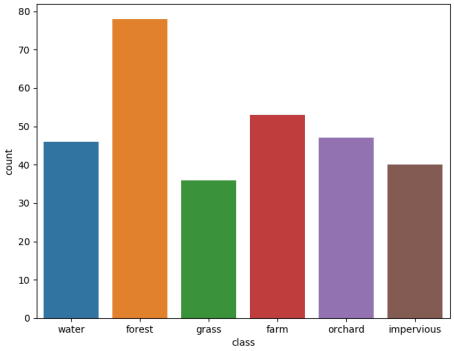

FIGURE 16 – Distribution des valeurs dans l'ensemble de test

Après nos analyses, nous décidons donc de choisir le *K-Nearest Neighbors* comme méthode finale. Les justifications d'après la phase d'évaluation sont plausibles et elle fournit une bonne performance finale dans les données de test. En conclusion, nous pensons que cette méthode généralise bien nos données.

4 Sélection de variables

Pour aller plus loin dans la comparaison des méthodes de classification, nous nous sommes intéressés aux méthodes de sélection de variables. En effet, notre ensemble d'apprentissage étant assez bruité, nous voulions explorer la possibilité de sélectionner seulement les variables pertinentes pour l'apprentissage supervisé. Nous développerons ici les méthodes de sélection par un test ANOVA, à l'aide du critère BIC[4] et à l'aide de forêts aléatoires.

4.1 Test ANOVA

Dans le cas de cette étude, nous avions en entrée des données quantitatives et en sortie une information qualitative (la décision quant à la classe d'appartenance d'un individu). C'est donc le test ANOVA qui a été retenu pour réaliser cette sélection. En croisant ces résultats

avec ceux du test de Student effectué lors de la phase d'analyse exploratoire (voir paragraphe 2.1), nous retrouvons le même classement des variables par rapport à leur corrélation avec *max_ ndvi*.

Pour comparer aux performances du modèle avec le nombre complet de variables, nous avons réappris notre jeu de données réduits aux 10 meilleures variables selon ANOVA et obtenu les résultats suivants (5) :

TABLE 5 – Précision des tests de prédiction suite à la sélection de variables par ANOVA test

Méthode	Modèle complet	Modèle réduit
LDA	0.59	0.56
QDA	0.55	0.60
GaussianNB	0.69	0.84
KNN	0.62	0.92
Decision Tree	0.55	0.88

Ainsi, à l'exception de l'analyse discriminante linéaire, toutes les autres méthodes répondent plutôt bien à la sélection de variable par ANOVA. Notons que les méthodes d'analyse discriminantes sont celles qui présentent la plus grande dispersion en précision d'après les figures 10 et 11.

4.2 Utilisation du critère BIC pour l'analyse discriminante

D'après les travaux de Q. Zhang et H. Wang [4], nous pouvons utiliser le critère de sélection de modèle BIC pour comparer des modèles d'analyse discriminante pour chacune des combinaisons de variables possibles. Ainsi, il s'agit de choisir le modèle (donc la combinaison de variables) qui minimise ce critère. La formule permettant de calculer le critère BIC est la suivante :

$$BIC = -2 \ln L(\theta) + dl \times \log n \qquad (1)$$

où L(θ) est la fonction de vraisemblance de la fonction de densité de la loi normal multidimensionnelle, *dl* est le degré de liberté du modèle et n le nombre d'individus.

Voici l'algorithme rétrogressif mis en oeuvre pour choisir le modèle optimal du point de vue du critère BIC : [1]

1. Initialisation : Soit $S_{(0)}$ le modèle complet (avec l'ensemble des variables) de départ. Exécuter une analyse discriminante quadratique sur $S_{(0)}$ et calculer $BIC(S_0)$.
2. Evaluation : A la t-ème étape (t>0), disposant de $S_{(t-1)}$, calculer $d_{(t)}$=$argmin_{j \in S_{(t-1)}} BIC(S_{(t-1)} \backslash \{j\})$ et mettre à jour $S_{(t)} = S_{(t-1)} \backslash \{d_{(t)}\}$.
3. Sélection : Répéter p fois l'étape de sélection, générant une séquence de modèles M=$\{S_{(t)} : 0 \leq t \leq p\}$. D'après M, le meilleur modèle est S=$argmin_{S \in M} BIC(S)$.

La combinaison optimale obtenue grâce à cette procédure est le singleton {max_ndvi}. Le précision de ce modèle est de 40%, contre 55% avec $S_{(0)}$.

Cette précision est moindre par rapport au modèle complet. On en conclut que même si les autres variables apportent moins d'information au modèle, elle permettent quand même d'en améliorer la prédiction. De plus, en se basant sur l'analyse discriminante quadratique ici, on part déjà avec de faibles chances d'obtenir une précision satisfaisante au vue de ses résultats comparativement à d'autres méthodes de classification. Nous pouvons pousser un peu plus loin l'analyse grâce à la fonction *classification_ report()* du module *sklearn* de python et qui nous renvoie ces informations (6) :

TABLE 6 – Précisions par classes de l'ADQ

Modèle	Classe	Précision
Optimal selon BIC	farm	0.35
	forest	0.39
	grass	1.00
	impervious	0.45
	orchard	1.00
	water	1.00
Complet	farm	0.52
	forest	0.51
	grass	0.31
	impervious	0.65
	orchard	1.00
	water	1.00

Ainsi, la variable *max_ ndvi* suffit à elle seule à classer complètement les classes *grass*, *orchard* et *water* mais ne fournit pas assez d'informations pour permettre de bien classer le reste des classes qui enregistrent toutes une bien moindre précision avec le modèle réduit par rapport au modèle complet.

4.3 Forêts aléatoires

Une forêt aléatoire est un ensemble de d'arbres de décision. Chaque arbre est créé en partitionnant le jeu de données global. La stratégie des forêts aléatoires (*random forest*), permet d'effectuer un tri sur la pureté des

1. Il a été simplifié par rapport à celui de Q. Zhang et H. Wang en calculant la fonction de vraisemblance sans passer par le nouveau modèle proposé dans leur article.

noeuds, c'est à dire faire diminuer l'impureté de Gini sur tous les arbres. C'est donc une méthode qui peut s'avérer efficace pour sélectionner des variables, et ainsi éviter le sur-apprentissage. La figure suivante 17 affiche l'importance de chaque variable dans la foret, du point de vue de l'impureté de Gini, déterminé grâce à l'attribut *feature_importances_* de la fonction *sklearn RandomForestClassifier*.

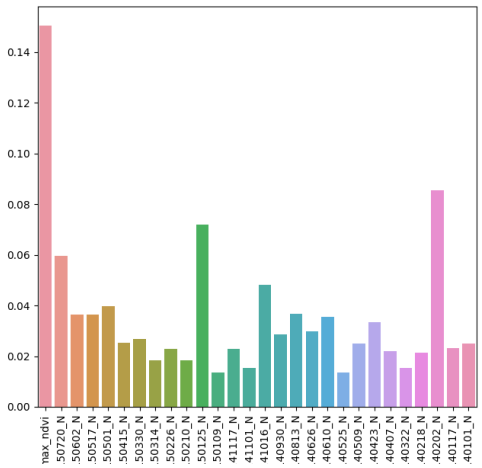

FIGURE 17 – Importance des variables dans la forêt aléatoire

Ainsi, 4 variables ont une importance supérieure à 5%, et nous pouvons remarquer que la variable "max_ndvi" concentre à elle seule plus de 15% de l'importance. Nous pourrions donc restreindre les méthodes non supervisées précédemment décrites à ces 4 variables, ou bien choisir celles dont l'importance est significative. Automatiquement, 9 variables sont choisies par la fonction *SelectFromModel*.

TABLE 7 – Précision des tests de prédiction suite à la sélection de variables par Random Forest

Méthode	Modèle complet	Modèle réduit
LDA	0.59	0.55
QDA	0.55	0.62
GaussianNB	0.69	0.60
KNN	0.62	0.61
Decision Tree	0.55	0.60
LogisticRegression	0.58	0.45

La tableau 7 présente les résultats pour les méthodes décrites précédemment. Nous voyons que cette sélection par les forets aléatoires s'avère peu efficace, puisque la précision est peu améliorée, et diminue même parfois. L'ANOVA semble donc être une méthode beaucoup plus adaptée dans notre situation.

5 Conclusion

Les méthodes d'apprentissage supervisé se sont avérées les plus adaptées pour résoudre le problème de classification lié à notre jeu de données. Parmi elles, nous avons pu retenir le modèle des K-plus-proches voisins comme étant celui qui était le plus généralisable tout en offrant des niveaux de précision satisfaisants. L'application de certaines méthodes de sélection de variables ont pu améliorer dans une certaine mesure la performance des classifieurs mis à l'épreuve mais ne change rien quant à notre choix de classifieur pour ce jeu de données.

Références

[1] Projet openstreetmap url.

[2] Site de l'uci, dépôt des datasets : Crowdsourced Mapping.

[3] DronesImaging. Indice de végétation ndvi url.

[4] Qiong Zhang and Hansheng Wang. On bic's selection consistency for discriminant analysis, disponible à cette url. Technical report, 2009.