Université de Rennes 1
UFR Sciences Economiques
Machine Learning
Année 2021-2022

Couvert Forestier : Classification

COLO BE Fonty

Etudiant en Master 2 Prédiction et Prévision Economiques, Univ. Rennes 1, Rennes, France

E-mail: fonty.fatimacolobe@etudiant.univ-rennes1.fr

ABSTRACT

En 1999, *Blackard, Jock A.* et *Denis J. Dean*, du département des Sciences Forestières de l'*Université d'État du Colorado*, ont publié un article scientifique intitulé : *Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables* [1], où ils ont comparé deux techniques différentes afin de prédire les types de couvert forestier situés dans quatre zones de nature sauvage dans le *Roosevelt National Forest*.

Keywords: Machine Learning, Data Science, Python, Scikit-Learn, Keras

1 INTRODUCTION

Le couvert forestier est défini comme étant la surface occupée par la projection horizontale des houppiers d'un arbre, d'une population d'arbres, d'un peuplement forestier dans son ensemble ou d'une ou plusieurs strates de végétation, dès lors, nous constatons que plusieurs types de couvert forestier peuvent exister au sein d'un même écosystème ou environnement.

Le but de cette étude est alors de prédire le type de couvert forestier à partir de données cartographiques, dont la collecte se fait généralement sur le terrain par des agents spécialisés, par opposition aux données de télédétection, dont l'extraction de l'information se fait généralement via des images de satellite.

L'approche que nous avons adoptée pour résoudre ce problème de classification en classes multiples consiste à explorer différents modèles d'apprentissage automatique, de sorte à en sélectionner le meilleur selon différents critères de performance que nous allons définir par la suite.

Ces modèles seront d'ordre paramétrique, par le biais d'estimation d'un modèle de *régression logistique* et non-paramétriques par le biais d'estimation d'un modèle des *k plus proches voisins*.

Les modèles d'agrégation qui se montrent beaucoup plus performants dans la résolution de problèmes de Machine Learning au cours de ces dernières années seront également estimés dans cette étude, en effet, nous allons également estimer un modèle de *forêts aléatoires*.

2 DESCRIPTION DES DONNÉES

2.1 Périmètre

La zone de récolte de ces données (voir Figure 1) concerne quatre régions sauvages du *Roosevelt National Forest*, où il n'y a quasiment eu aucune intervention humaine, de sorte à ce que les types de couvert forestier existants soient davantage dus aux résultats d'un processus écologique que d'une pratique de gestion forestière. Les régions concernées sont respectivement, la région sauvage du ***Rawah***, composée de 29 628 hectares en 1998, la région sauvage du ***Comanche Peak***, composée de 27 389 hectares en 1998, la région sauvage ***Neota***, composée de 3904 hectares en 1998 et de la région sauvage de ***Cache la Poudre***, composée de 3817 hectares en 1998.

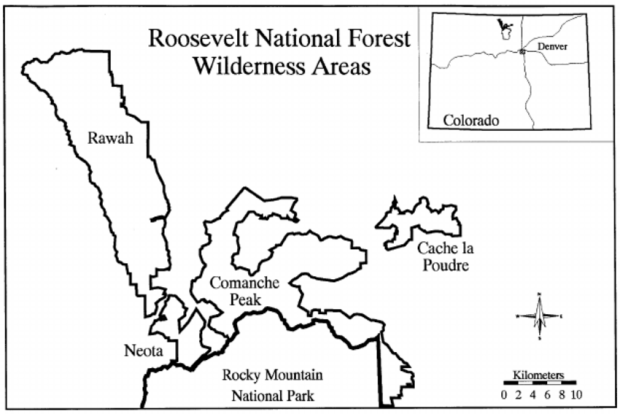

Figure 1. Zone de récolte

2.2 Description des variables

Les données que nous avons à disposition proviennent du site **UCI Machine Learning Repository** [2] et contiennent 12 variables explicatives dont 10 qui sont de type quantitatif et 2 de type qualitatif. Ci-dessous la description de ces variables exprimées dans leur unité respective.

1. Elevation (Mètre)
2. Aspect (Azimut)
3. Slope (Degré)
4. Horizontal distance to hydrology (Mètre)
5. Vertical distance to hydrology (Mètre)
6. Horizontal distance to Roadways (Mètre)
7. Hillshade 9am (Index à 255)
8. Hillshade Noon (Index 0 à 255)
9. Hillshade 3pm (Index 0 à 255)
10. Horizontal distance to fire points (Mètre)
11. Wilderness area (4 colonnes binaires)
12. Soil type (40 colonnes binaires)

La variable **Wilderness Area** correspond à la zone de récolte des données, elle a été encodée à chaud, c'est-à-dire que pour chaque modalité de cette variable, une variable binaire a été créée.
Ainsi nous obtenons ces variables supplémentaires :

- Wilderness area 1 = Rawah Wilderness Area
- Wilderness area 2 = Neota Wilderness Area
- Wilderness area 3 = Comanche Peak Wilderness Area
- Wilderness area 4 = Cache la Poudre Wilderness Area

La variable **Soil Type** quant à elle correspond au type de sol sur lequel les données ont été récoltées, elle a également été encodée à chaud, ce qui nous génère 40 variables binaires supplémentaires dans notre jeu de données pour chaque modalité.

2.3 Statistiques descriptives

Notre variable cible **Cover Type** qui désigne le type de couvert forestier est composée de 7 modalités. Il nous a paru important dans un premier temps d'observer la distribution des classes de cette variable afin de savoir si nous avons affaire à des données déséquilibrées (voir Figure 2).
Nous constatons que les types d'arbres *Spruce/Fir* et *Lodgepole Pine* sont surreprésentées dans ce jeu de données, il sera donc important de spécifier ce fait aux différents modèles afin que ceux-ci puissent accorder une plus grande importance aux classes ayant peu d'observations.

Au sein de la Figure 3, nous constatons qu'*a priori* le développement de certains types d'arbres se fait bien selon les caractéristiques de certaines zones.

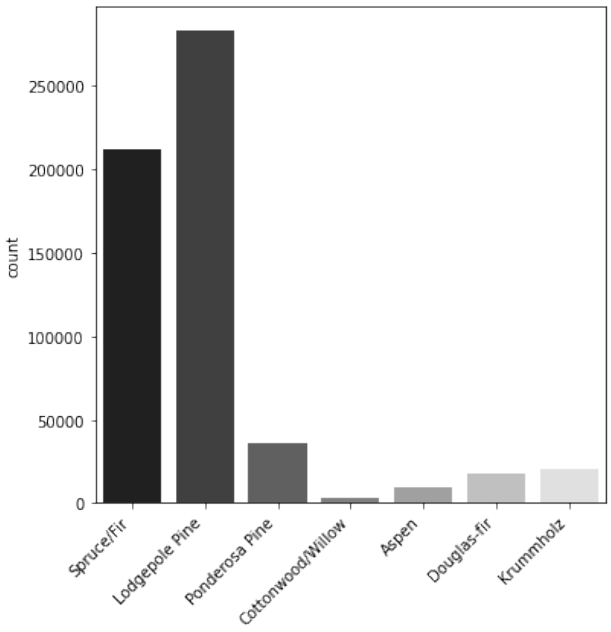

Figure 2. Distribution de classes

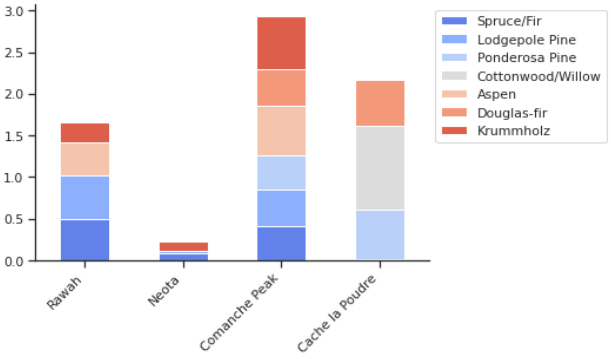

Figure 3. Types d'arbres par zone

3 MODÈLES / MÉTHODES

Les hyperparamètres de nos modèles ont été estimés par la fonction **RandomizedSearchCV** de la bibliothèque **Scikit-learn** [3] de Python, où nous avons spécifié le nombre de blocs pour la validation croisée KFold à 5, la fonction de score à optimiser comme étant la précision équilibrée afin de prendre en compte le déséquilibre des classes et le maximum d'itérations pour nos paramètres de recherche à 5.
La précision équilibrée est définie telle que :

$$\text{Précision équilibrée} = \frac{1}{K}\sum_{k=1}^{K}\frac{r_k}{n_k}$$

où K est le nombre de classes, r_k : le nombre de vrais positifs appartenant à la classe k, n_k : le nombre d'observations appartenant à la classe k.

3.1 Régression logistique

La régression logistique, malgré son nom, est un modèle linéaire de classification plutôt que de régression. Dans le cadre d'une régression multinomiale, nous allons être intéressées par modéliser la probabilité d'appartenance d'un individu à une catégorie "k" définie telle que :

$$\pi_k(\omega) = P[Y(\omega) = y_k / X(\omega)]$$

Avec la contrainte,

$$\sum_k \pi_k(\omega) = 1$$

La vraisemblance s'écrit,

$$L = \prod_\omega [\pi_1(\omega)]^{y_1(\omega)} \times \cdots \times [\pi_K(\omega)]^{y_K(\omega)}$$

avec,

$$y_k(\omega) = 1 \text{ ssi } Y(\omega) = y_k$$

Ainsi, la log-vraisemblance à maximiser est telle que :

$$LL = \sum_\omega y_1(\omega) \ln \pi_1(\omega) + \cdots + y_K(\omega) \ln \pi_K(\omega)$$

3.2 K plus proches voisins

Ce modèle est basé sur un type d'apprentissage basé sur des instances ou d'apprentissage non généralisant. La classification est calculée à partir d'un vote à la majorité simple des voisins les plus proches de chaque point : un point de requête se voit attribuer la classe de données qui a le plus de représentants parmi les voisins les plus proches du point.

3.3 Forêts aléatoires

Les méthodes d'agrégation d'arbres auquel appartient la méthode des *forêts aléatoires* consistent à construire un très grand nombre d'arbres "simples" : $g_1, \ldots, g_B$ et à les agréger en faisant la moyenne :

$$\frac{1}{B} \sum_{k=1}^{B} g_k(x)$$

L'algorithme des *forêts aléatoires* consiste à construire des arbres sur des échantillons bootstrap et à les agréger.

3.4 Mise en œuvre

Notre ensemble de test est composé de 1/3 de notre jeu de données et notre ensemble d'apprentissage est composé des 2/3 restants. Pour cette étude, nous avons décidé de ne pas créer un ensemble de validation puisque pour nos différents modèles, le meilleur que nous allons retenir ainsi que les hyperparamètres qui l'optimisent, sera obtenu par validation croisée 5Fold.

3.4.1 Les hyperparamètres

Pour la *régression logistique*, 3 hyperparamètres nous ont paru importants à optimiser :

1. La pénalité : ridge ou lasso
2. L'inverse de la force de régularisation C
3. Le solveur : newton-cg, sag ou saga

Certains hyperparamètres ont été spécifiés au modèle sans chercher à les optimiser, en effet, nous avons spécifié au modèle que nous voulons un poids différents selon le nombre d'observations de nos différentes classes afin que celui-ci pénalise d'avantage les classifications erronées des classes minoritaires pour à terme, leur accorder une plus grande importance. Cette spécificité est obtenue par **n_samples / (n_classes * np.bincount(y))** où n_samples correspond à la taille de notre échantillon, n_classes le nombre de classes et np.bincount(y) correspond à un vecteur contentant le nombre d'observations que possède chaque classe.

Pour le modèle des *k plus proches voisins*, nous avons été intéressés par le fait d'optimiser 2 hyperparamètres :

1. Le nombre de voisins
2. La distance : euclidienne ou manhattan

L'hyperparamètre qui nous avons spécifié au modèle par défaut ici concerne le poids, en effet, nous lui avons spécifié que nous voulons qu'il considère le poids des points par l'inverse de leur distance, de sorte à ce que les voisins les plus proches d'un point de requête aient une plus grande influence que les voisins plus éloignés.

Pour le modèle des *forêts aléatoires*, nous avons été intéressés par le fait d'optimiser 6 hyperparamètres :

1. Le nombre d'arbres
2. Nombre de variables pour chaque fractionnement
3. La profondeur maximale de l'arbre
4. Le nombre minimum d'échantillons pour diviser un nœud
5. Le nombre minimum d'échantillons pour être un nœud
6. La méthode de sélection des échantillons par bootstrap ou non

L'hyperparamètre spécifié par défaut ici concerne le poids des classes de sorte à ce que le modèle ajuste automatiquement les poids inversement proportionnels aux fréquences de classe dans les données d'entrée sous la forme **n_samples / (n_classes * np.bincount(y))**.

4 RÉSULTATS ET DISCUSSION

Les critères de performance à maximiser que nous avons choisis pour comparer nos différents modèles sont : *la précision équilibrée* que nous avons déjà présentée dans la partie 3 et le *f1-score pondéré* pour un problème de classification en classes multiples.

$$F_1 := \sum_{k=1}^{K} w_k \frac{P_k \times R_k}{P_k + R_k}$$

où P_k et R_k correspondent respectivement à la précision et au rappel de la classe k et où $w_k = \frac{2n_k}{N}$ où n_k correspond au nombre d'observations de la classe k et où N correspond à la taille de notre échantillon.

4.1 Tableau des résultats

Dans la Table 1, nous pouvons constater que le modèle de *régression logistique* est celui qui possède la moins bonne capacité prédictive, en effet, nous constatons que la précision équilibrée pour ce modèle est de 0.641 tandis que sa performance est de 0.567 pour le f1-score pondéré, ce qui est moindre.

Le modèle des *k plus proches voisins* quant à lui semble être plus performant au niveau de la précision équilibrée et du f1-score pondéré qui sont respectivement de 0.836 et de 0.903.

Nous constatons en revanche, une nette différence pour le modèle des *forêts aléatoires* qui affiche un score de 0.922 pour la précision équilibrée et un score de 0.955 pour le f1-score pondéré, ce qui fait de lui le modèle le plus performant que nous avons réussi à obtenir pour cette étude.

Table 1. Tableau des résultats

Modèle	Bal Acc	F1-Score
LR	0.641	0.567
KNN	0.836	0.903
RF	0.922	0.955

4.1.1 Courbes d'apprentissage

Le modèle de *régression logistique* d'après ses courbes d'apprentissage (voir Figure 4) semble avoir convergé, en effet, nous pouvons constater que celui-ci affiche un score plutôt équivalent aussi bien lors de sa phase d'entrainement que lors de sa phase de validation, ce qui nous laisse penser que, même si nous augmentons le nombre d'observations que nous avons à disposition dans notre jeu de données d'entrainement, ce modèle aura du mal à avoir des résultats significativement supérieurs. Ainsi, *a priori* ce modèle semble ne pas être adapté pour répondre à ce problème de classification.

Le modèle des *k plus proches voisins* semble quant à lui être confronté au compromis de biais-variance, en effet, le modèle semble être très dépendant des données (voir Figure 5) et semble avoir du mal à comprendre la relation qui existe entre notre variable cible, et nos variables explicatives. Nous pouvons le constater par son score parfait (ce score correspond à la précision équilibrée) pendant toute sa phase d'entrainement et de par son score qui finit par atteindre environ les 0.84 lors de son dernier cycle de validation. Ce score peut *a priori* être amélioré en augmentant le nombre d'observations de nos données d'entrainement et en considérant un nombre de blocs pour la validation croisée KFold à 10.

Dans une moindre mesure, le même compromis de biais-variance semble se poser pour le modèle des *forêts aléatoires* (voir Figure 6), en effet, celui-ci affiche un score parfait pendant toute sa phase d'apprentissage pour avoir des résultats moindres lors de ses premières phases de validation, pour à terme, atteindre un score d'envrion 0.92 lors de son dernier cycle. De même ce modèle peut *a priori* être amélioré en augmentant le nombre d'observations de nos données d'entrainement, en considérant d'autres hyperparamètres et une recherche exhaustive pour les optimiser et en considérant un nombre de blocs pour la validation croisée KFold à 10.

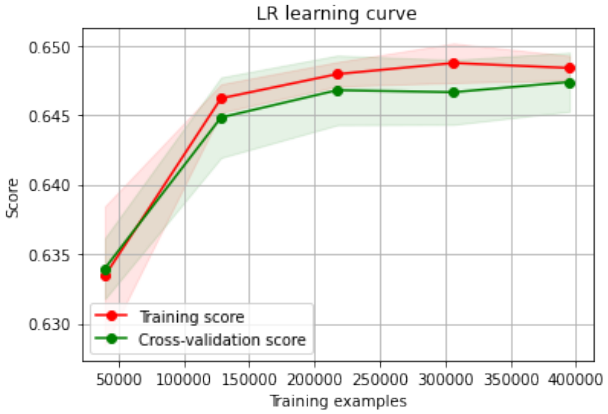

Figure 4. Courbes d'apprentissage LR

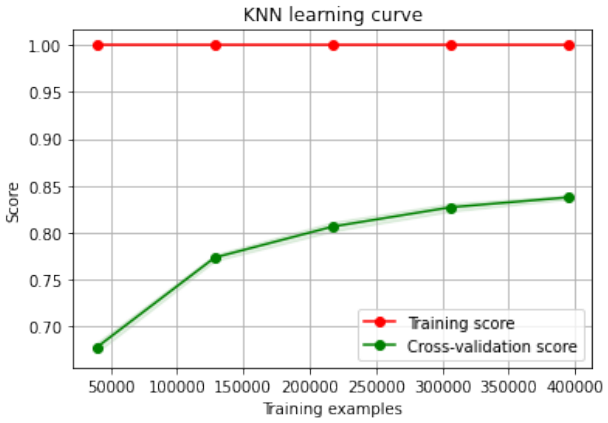

Figure 5. Courbes d'apprentissage KNN

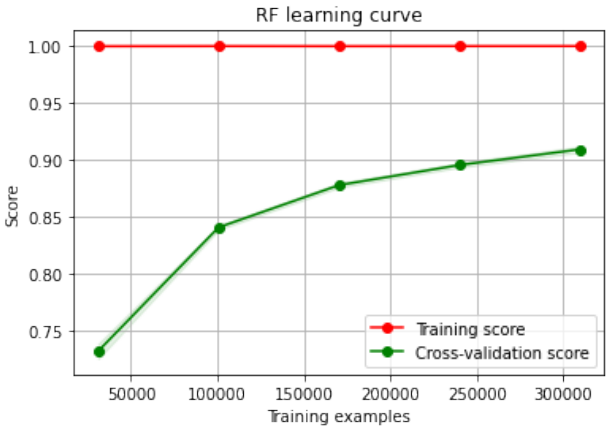

Figure 6. Courbes d'apprentissage RF

4.1.2 Rapport de classification

Au sein de la Figure 7 nous pouvons constater que notre modèle de *régression logistique* a du mal à prédire les classes 2, 3, 4, 5 et 6 qui comportent très peu d'observations contrairement aux deux premières classes, ce qui peut, en partie, expliquer les résultats moindres de ce modèle.

Le modèle des *k plus proches voisins* semble quant à lui (voir Figure 8) prendre en compte le déséquilibre de nos classes et affiche de meilleurs scores au niveau de la précision, le rappel et le f1-score qui ne descendent jamais en-dessous des 0.72 et ce, bien même que celui-ci possède une plus ou moins grande variance par rapport aux scores sur les données d'apprentissage et de validation.

En revanche, au sein de la Figure 9 pour le modèle des *forêts aléatoires*, nous pouvons constater que celui-ci affiche de bien meilleurs résultats même lorsque nos classes comportent peu d'observations, en effet, la précision, le rappel et le f1-score pour toutes les classes sont au minimun à 0.86, ce qui fait de lui notre meilleur modèle, il semble donc ainsi être le plus adapté pour répondre à notre problème de classification en classes multiples à condition d'optimiser le plus possible nos hyperparamètres.

Classification Report:

	precision	recall	f1-score	support
0	0.65	0.61	0.63	31946
1	0.78	0.46	0.58	42414
2	0.66	0.39	0.49	5336
3	0.18	0.81	0.29	388
4	0.09	0.72	0.16	1458
5	0.27	0.63	0.38	2578
6	0.28	0.87	0.42	3032
accuracy			0.53	87152
macro avg	0.41	0.64	0.42	87152
weighted avg	0.68	0.53	0.57	87152

Figure 7. Rapport de classification LR

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.91	0.92	31946
1	0.92	0.93	0.93	42414
2	0.88	0.88	0.88	5336
3	0.74	0.72	0.73	388
4	0.76	0.78	0.77	1458
5	0.77	0.78	0.77	2578
6	0.93	0.93	0.93	3032
accuracy			0.91	87152
macro avg	0.85	0.85	0.85	87152
weighted avg	0.91	0.91	0.91	87152

Figure 8. Rapport de classification KNN

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	31946
1	0.96	0.97	0.97	42414
2	0.94	0.96	0.95	5336
3	0.88	0.86	0.87	388
4	0.91	0.86	0.89	1458
5	0.90	0.91	0.91	2578
6	0.97	0.97	0.97	3032
accuracy			0.96	87152
macro avg	0.93	0.93	0.93	87152
weighted avg	0.96	0.96	0.96	87152

Figure 9. Rapport de classification RF

5 CONCLUSION

5.1 Effets de conclusion

Ainsi, nous avons vu selon les trois modèles que nous avons estimés, que le modèle de *régression logistique multinomiale* n'est pas adapté pour répondre à ce problème de classification, ce modèle semble avoir convergé et affiche des résultats moindres, ce qui nous laisse penser qu'il sera difficile de significativement pouvoir l'améliorer. Le modèle des *k plus proches voisins* quant à lui semble pouvoir encore être amélioré en augmentant par exemple le nombre d'observations dans le jeu de données entrainement et en augementant le nombre de blocs pour la validation croisée KFold ou la StratifiedKFold. Le modèle des *forêts aléatoires* semble être assez prometteur et peut être encore sensiblement amélioré en jouant sur la valeur des hyperparamètres, en augmentant le nombre d'observations du jeu données d'entrainement et/ou en considérant d'autres types de validation croisée.

5.2 Perspectives

Un modèle d'Extreme Gradient Boosting (XGBoost) peut également être envisagé pour répondre à ce problème de classification, en effet, la grande panoplie de ses hyperparamètres offre à ce modèle une grande flexibilité et son estimateur de base qui est basé sur les arbres de décision font de lui un modèle assez prometteur.

REFERENCES

[1] Blackard, J. A. & Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Comput. Electron. Agric.* **24**, 131–151 (1999). URL https://www.sciencedirect.com/science/article/pii/S0168169999000460. DOI https://doi.org/10.1016/S0168-1699(99)00046-0.

[2] Dua, C., Dheeru et Graff. UCI machine learning repository (2017). URL http://archive.ics.uci.edu/ml.

[3] Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).