Prédiction du montant de pourboires (tips) pour une course en taxi et prédiction du type de végétation en forêt à partir de variables topographiques

Projet apprentissage supervisé, 12 novembre 2024

Pablo Hueso, Sam Vallet

Résumé

Ce compte-rendu a pour objectif de présenter et de comparer les performances de différents modèles de régression sur un jeu de données contenant les montants des pourboires de diverses courses de taxi à New York, afin de prédire ces montants. Une analyse similaire sera réalisée pour des modèles de classification, appliqués à un jeu de données décrivant des parcelles forestières, dans le but de prédire le type de forêt. Les modèles seront comparés en fonction de diverses métriques d'évaluation de la qualité de prédiction. Le code est disponible sur GitHub.

1 Prédiction du montant des pourboires de taxi

1.1 Introduction

Le premier des deux problèmes sur lequel on s'interesse est celui de la prédiction des pourboires dans les trajets en taxi dans la ville de New York. On dispose d'un jeu de données d'entraînement représentant 100 000 courses de taxis effectuées en janvier 2024 pour lesquelles le montant des pourboires est connu, avec 19 covariables et une variable réponse (le montant de pourboire pour le trajet). Nous disposons également d'un jeu de données de test avec 2 627 163 observations (sans la variable réponse).

1.2 Ensemble des covariables

Nous avons essentiellement trois types de variables. D'abord nous disposons de variables qui décrivent le trajet (date et heure de départ, d'arrivée, lieu de départ, lieu d'arrivée (divisés en latitude et longitude), nombre de passagers...), aussi on dispose d'un autre groupe de variables qui décrivent des informations monétaires du trajet (type de tarif, moyen de paiement, nombre de péages payés, taxes, suppléments...) et un troisième type de variables que l'on peut classer comme métavariables, qui indiquent des informations sur la manière dont les données du trajet ont été collectées (entreprise qui collecte les données et indication de si les données ont été stockées en mémoire dans le véhicule avant leur transmission). Pour trouver une description détaillée des variables, consultez l'annexe.

1.3 Nettoyage, exploration et préparation des données

La première étape fondamentale dans le développement d'une solution de machine learning est le traitement des données. Idéalement, nous souhaiterions travailler avec des données sans valeurs manquantes, ainsi que des données non dupliquées. Le jeu de données avec lequel nous commençons à travailler a déjà subi un prétraitement pour le rendre plus utilisable ; il ne présente donc ni lignes dupliquées ni valeurs manquantes (NaN). Nous nous concentrons donc principalement sur le traitement des valeurs aberrantes, en déterminant lesquelles sont des erreurs de mesure et lesquelles ne le sont pas, et comment les remplacer par des valeurs raisonnables.
Il convient de noter que, s'agissant d'une compétition Kaggle, nous n'avons pas la possibilité d'écarter les observations que nous considérons comme des erreurs de mesure dans le jeu de données de test, car c'est précisément sur ce jeu de données que nous devons faire nos prédictions. Étant donné que le jeu de test est beaucoup plus grand, la majorité des valeurs aberrantes s'y trouvera naturellement. En tout état de cause, dans cette section, sauf indication contraire, lorsque nous faisons référence au jeu de données, nous faisons référence aussi bien au jeu d'entraînement qu'au jeu de test.

Avant l'exploration, nous introduisons quelques petits mais importants changements dans les variables pour en faciliter la manipulation. Tout d'abord, nous introduisons une nouvelle variable, `trip_duration`, qui enregistre la durée de chaque trajet en minutes. Ensuite, nous appliquons un encodage binaire (0 ou 1) à la variable `store_and_fwd_flag` pour en faciliter la manipulation.

1.3.1 Valeurs extrêmes et valeurs aberrantes

Avant de commencer à explorer les données, précisons que, lorsque nous parlons de valeurs extrêmes, nous faisons référence aux valeurs situées dans les queues de la distribution, mais lorsque nous parlons de valeurs aberrantes, nous faisons référence à des valeurs qui (nous pensons) résultent d'une erreur de mesure ou de transmission des données. Nous nous concentrerons sur le remplacement des valeurs aberrantes par des valeurs raisonnables. Notons également que cette discussion concerne exclusivement les variables continues.

La procédure que nous avons suivie pour chaque variable est la suivante :

1. Identification des frontière d'erreur (valeurs qui ne peuvent pas être prises pour certaines variables)
2. Identification d'une variable hautement corrélée
3. Remplacement des valeurs aberrantes par une estimation linéaire à partir de la variable corrélée

Nous présentons ci-dessous les variables modifiées, les intervalles dans lesquels elles ont été modifiées, le nombre d'observations affectées et les variables utilisées pour faire l'estimation (ainsi que la corrélation avec la variable affectée calculée dans les limites acceptables). Dans chaque ligne du tableau, nous faisons référence à la variable à corriger par v, et à la variable corrélée par c.

Variable	**Frontière d'erreur**	**Variable corrélée**	**Corrélation**	**Nombre d' erreurs**
trip_duration	$v = 0$ ou $v \geq 300$	trip_distance	0.791	1786
fare_amount	$v \leq 4$ et $c \geq 5$	trip_distance	0.95	173
trip_distance	$v \leq 1$ et $c \geq 45$ [1]	fare_amount	0.95	1607

TABLE 1 – Tableau des variables avec frontières d'erreur, corrélations et nombre d'erreurs

Les valeurs des seuils ne sont pas arbitraires. En particulier, dans la deuxième ligne, nous avons choisi $v \leq 4$, car le tarif standard d'un trajet en taxi à New York est de 3 dollars + un montant calculé principalement en fonction de la distance. Ce coût initial de 3 dollars passe à 4 dollars pendant la nuit [1]. Nous avons ajouté une note en bas de page précisant que ces calculs ont été effectués en tenant compte du tarif JFK, qui est un tarif fixe de 70$ indépendamment de la distance [1].

Nous incluons ci-dessous une comparaison de `fare_amount` par rapport à `trip_distance` avant et après le traitement :

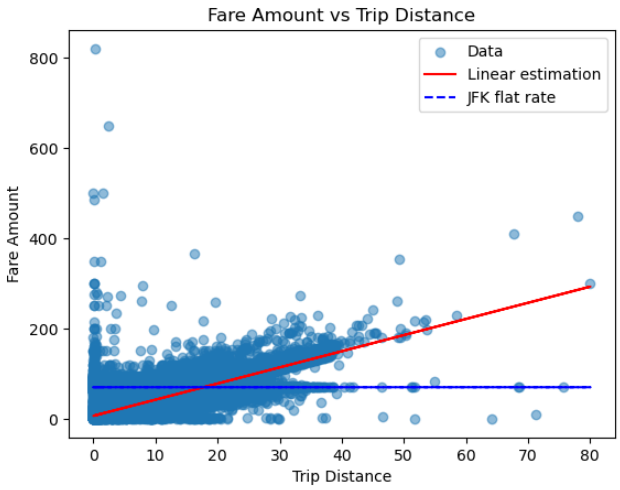

FIGURE 1 – Données avant traitement

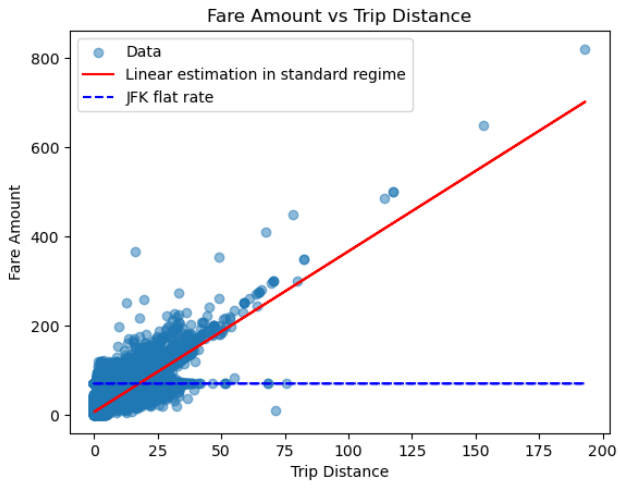

FIGURE 2 – Données après traitement

Nous pouvons observer deux populations clairement différenciées et une variance intra-population réduite dans les données post-traitées. Bien qu'il reste certains valeurs extrêmes qui sont probablement des erreurs, nous avons éliminé la grande majorité des données aberrantes.

1.3.2 Étude bivariée

Nous procédons ensuite à une rapide étude bivariée de nos variables en examinant la matrice de corrélation.

1. Aussi `RatecodeID` $\neq 2$. On distingue également $v \leq 8$, $c \geq 140$ et `RatecodeID` $\neq 2$

Dans l'image ci-dessous, plusieurs corrélations attirent l'attention. Nous avons déjà discuté de la relation entre `trip_distance` et `fare_amount`, ainsi que de la relation entre `trip_duration` et `trip_distance`. En conséquence, il est naturel que `fare_amount` et `trip_duration` soient également corrélés. Par ailleurs, les fortes valeurs de corrélation que nous observons sont faciles à expliquer :

La relation entre `PU_location_lon` et `Airport_fee` est particulièrement notable et s'explique par le fait que l'aéroport JFK est situé à l'est de Manhattan et à l'est de Brooklyn. D'autre part, nous observons une forte corrélation entre `tolls_amount` et `trip_distance`, ce qui est naturel, car plus un trajet est long, plus il est probable qu'il passe par un péage. En tout état de cause, les corrélations les plus intéressantes sont celles entre les covariables et notre variable cible. La variable présentant la plus forte corrélation avec `tip_amount` est `fare_amount`. Cela n'est pas surprenant : à New York, il est courant de laisser un pourboire de 15-20 %. De manière similaire, `trip_distance` présente une corrélation comparable mais légèrement inférieure avec `tip_amount`. Le reste des variables présente soit une faible corrélation, soit des explications simples. Cependant, la corrélation avec `payment_type`, une variable catégorielle à 6 niveaux, attire l'attention. Nous n'avons pas encore appliqué d'encodage pour cette variable, de sorte qu'à ce stade, un modèle pourrait inférer une relation d'ordre inexistante entre les modes de paiement. Néanmoins, la corrélation indique que le type de paiement influence le montant du pourboire, ce qui mérite une étude plus approfondie.

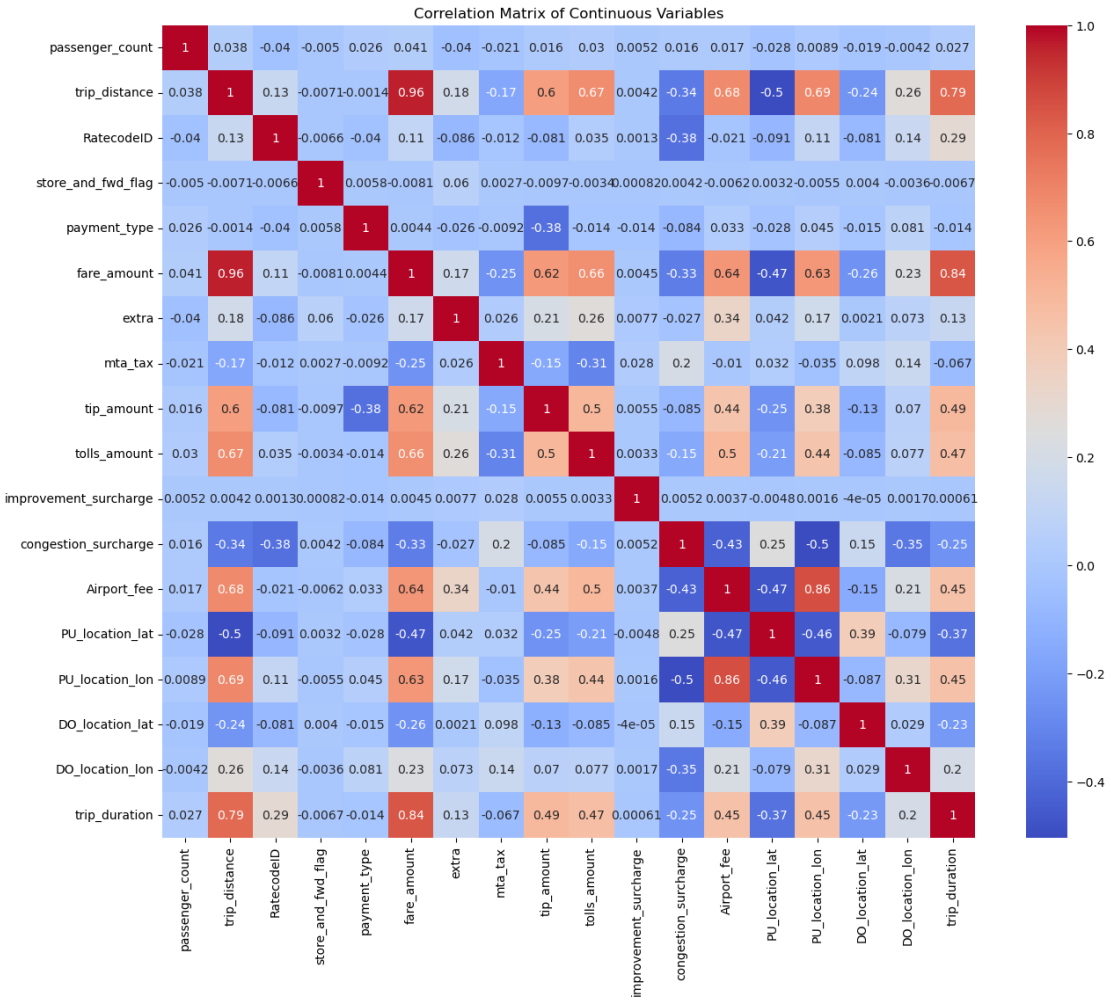

FIGURE 3 – Matrice de correlation

1.4 Transformation et création de variables

Dans cette section, nous incluons certaines des modifications que nous avons mises en œuvre ou explorées dans notre ensemble de variables. Nous présentons une motivation initiale pour explorer ces changements, et nous examinerons l'impact des variables introduites dans la section sur l'importance des variables, ainsi que dans le bilan de performance des modèles.

1.4.1 Encodage des variables catégorielles

Comme nous l'avons expliqué dans les sections précédentes, nous disposons de certaines variables catégorielles qui, bien que non ordinales, sont encodées par défaut avec des nombres entiers. Pour capturer leur caractère nominal et éviter que les modèles apprennent des relations inexistantes, il est préférable d'utiliser un encodage différent. Dans notre cas, nous avons appliqué un encodage one-hot à chacune des variables catégorielles.

1.4.2 Encodage des variables d'emplacement

Bien que les variables de localisation soient numériques, un encodage catégoriel pourrait mieux capturer l'information relative au lieu. Finalement, nous n'avons utilisé aucun encodage spécifique pour ces variables ; cependant, nous avons effectué quelques explorations que nous incluons en annexe.

1.4.3 Création de variables supplémentaires

Comme nous l'avons indiqué dans l'introduction, prédire les pourboires à partir des variables disponibles n'est pas une tâche simple. C'est pourquoi nous avons principalement cherché à introduire certaines variables qui, selon nous, pourraient être liées à la qualité du service.

Variables de vitesse Nous avons introduit une variable définie par :

$$\texttt{trip\_speed} = \frac{\texttt{trip\_distance}}{\texttt{trip\_duration}}$$

L'idée derrière cette variable est de mesurer si le trajet a duré plus ou moins longtemps que prévu. Par la suite, nous avons affiné cette approche en introduisant une variable `is_fast_trip` L'idée est de rendre la variable plus informative : dans les sections précédentes, nous avons observé que la relation entre trip_distance et trip_duration est presque linéaire (les non-linéarités étant provoquées par des facteurs tels que le trafic, les conditions météorologiques, différents conducteurs...). Par conséquent, nous pouvons (de manière approximative) transformer une variable en l'autre. Cela signifie qu'il existe une constante c pour laquelle (expression) sera généralement proche de 1 (supérieure à 1 pour les trajets plus rapides que "prévu"). Ainsi, nous introduisons la variable :

$$\texttt{is\_fast\_trip} = \mathbb{1}_{c\frac{\texttt{trip\_distance}}{\texttt{trip\_duration}}>1}$$

Où c est le coefficient d'un modèle linéaire qui prédit $trip\_duration$ à partir de $trip\_distance$.

Variables dérivées de la date et de l'heure de prise en charge et de dépose Nous avons pensé que les personnes pourraient être plus ou moins enclines à laisser un pourboire en fonction de l'heure. Nous avons donc naturellement testé l'introduction d'une variable heure, catégorielle ordinale avec 24 niveaux. Cependant, cet encodage ne permet pas de capturer la nature cyclique du temps, nous avons donc décidé de compter le nombre de secondes depuis minuit et d'encoder cette valeur de manière cyclique, en définissant les variables `time_sin` et `time_cos` :

$$\texttt{time\_sin} = \sin(\frac{2\pi * \texttt{seconds\_from\_midnight}}{86400})$$

$$\texttt{time\_cos} = \cos(\frac{2\pi * \texttt{seconds\_from\_midnight}}{86400})$$

1.5 Importance des variables

Avant de tester différents modèles, nous incluons un graphique de l'importance des variables pour guider la sélection de celles-ci par la suite. Nous avons décidé d'utiliser l'importance des variables dans XGBoost, car il est bien connu que, dans la méthode Random Forest d'origine, les mesures d'importance des variables sont influencées par le nombre de catégories et l'échelle de mesure des variables prédictives, qui ne sont pas des indicateurs directs de la véritable importance de la variable. [2]
Par ailleurs, nous avons utilisé les résultats d'importance des variables pour guider notre intuition, mais nous ne nous sommes pas méthodiquement basés sur ces résultats.

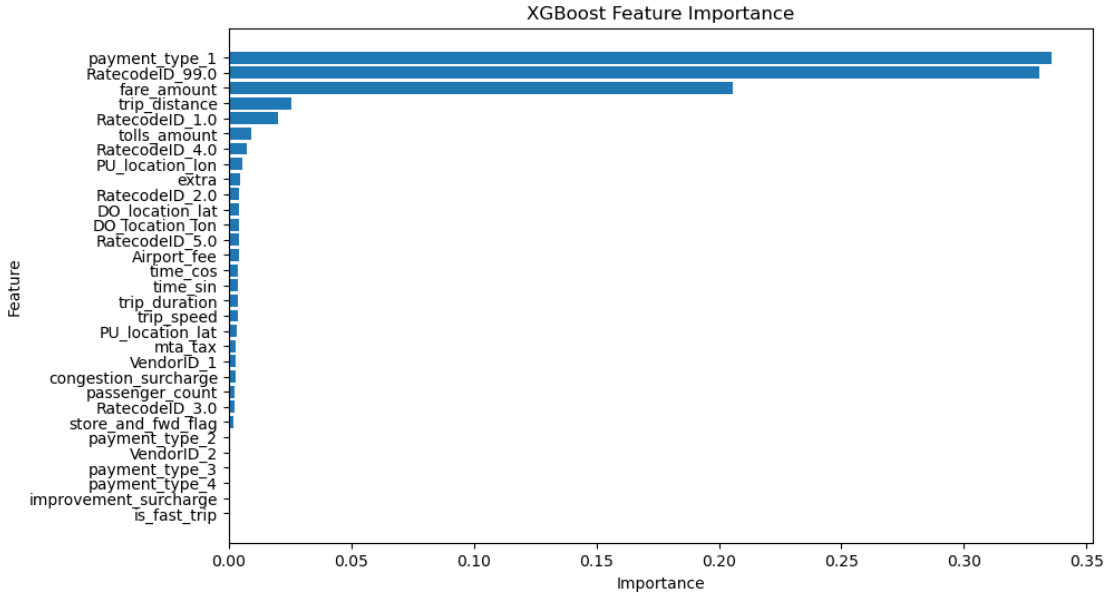

FIGURE 4 – Importance XGB

1.6 Modèles et performance

Nous avons principalement testé trois modèles : Random Forest, XGBoost et régression linéaire.

1.6.1 Régression linéaire

La régression linéaire a été le premier modèle que nous avons testé. Cependant, en raison de certaines interactions bivariées et du nombre de variables catégorielles, ce modèle n'est pas le plus adapté au problème. De plus, compte tenu de nos variables et du traitement des valeurs aberrantes que nous avons appliqué, les modèles de régression linéaire présentent des problèmes de multicolinéarité. [3] Pour ces raisons, nous nous sommes rapidement concentrés sur d'autres modèles.

1.6.2 Random Forest

Ce modèle est efficace pour obtenir des résultats de référence fiables et gérer différents types de données. Il est bon pour réduire la variance dans les prédictions, mais il peut parfois manquer certaines relations complexes entre les variables en raison de la structure indépendante de chaque arbre. En particulier, au fur et à mesure que nous développions

le projet, chaque fois que nous introduisions un changement dans les variables, nous testions généralement la performance avec un modèle de base de forêt aléatoire. D'autre part, bien que ce modèle ait une performance nettement supérieure aux régressions linéaires, ajuster un seul modèle de forêt aléatoire prenait plusieurs minutes, ce qui rendait le tuning des hyperparamètres nettement plus difficile. Par conséquent, nous n'avons pas investi de temps à essayer d'affiner la performance de nos modèles de forêt aléatoire en variant les hyperparamètres.

1.6.3 XGBoost

Ce modèle excelle souvent à capturer des motifs non linéaires et des interactions complexes, notamment avec des données de grande dimension. De plus, un modèle sans tuning des hyperparamètres présentait déjà une très bonne performance, prenant moins de deux secondes pour s'entraîner. La rapidité et la facilité d'ajustement de ces modèles nous ont permis de rechercher un ajustement plus fin à l'aide d'une RandomizedSearchCV. De plus, nous avons exploré différentes approches. En particulier, XGBoost supporte un encodage natif des variables catégorielles [4], que nous avons testé, mais sans obtenir de bons résultats. Nous sommes donc revenus à notre encodage one-hot.

1.6.4 Bilan de performance des modèles

Nous incluons ci-dessous un tableau résumant la performance des différents modèles testés. Pour évaluer la performance avec les données dont nous disposions, nous avons divisé notre ensemble d' entraînement en deux parties selon la règle du 80/20, servant ainsi de nouvel ensemble d' entraînement et de test, car nous n' avions pas accès à la variable cible dans l'ensemble de test. Dans la section XGB, tous les modèles ont été testés avec le prétraitement avancé (celui expliqué dans la section des valeurs extrêmes). Enfin, le modèle le plus prometteur que nous avons pu entraîner était un

Modèle	**MSE local**	**R2 local**	**Kaggle Private Score (R2)**
Régression linéaire naïve avec un prétraitement minimal	5.707	0.58	0.57510
Régression linéaire avec encodages de base	5.43	0.6	0.60191
RF encodages de base, sans variables d'emplacement	4.62	0.660	0.66282
RF encodages de base, avec variables d'emplacement	4.321	0.682	0.67854
RF+trip_speed+hour+prétraitement basique	4.294	0.684	0.68441
RF+trip_speed+encodage cyclicque+prétraitement basique	4.240	0.688	0.68799
RF précédente+sélection de variables	4.267	0.686	0.68788
RF+sans sélection + prétraitement avancé	4.226	0.689	0.68803
Base XGB	4.308	0.683	0.68235
Base XGB native categorical encoding	5.15	0.621	0.61907
XGB+RandomizedSearchCV	4.090	0.699	0.69736
XGB+Optuna library optimization	3.96	0.708	0.69694

Table 2 – Comparaison des modèles avec MSE, R2, and Kaggle Private Score

XGB dont les hyperparamètres ont été optimisés grâce à la librairie Optuna, qui a trouvé les hyperparamètres suivants :

- **Meilleurs hyperparamètres :**
  - `n_estimators` : 1000
  - `max_depth` : 5
  - `learning_rate` : 0.00498521321762481
  - `subsample` : 0.9998165540592395
  - `colsample_bytree` : 0.9890595259391189
  - `gamma` : 0.9419199390513634
  - `reg_alpha` : 0.011680385049163666

— `reg_lambda` : 0.7772467248168831
— `min_child_weight` : 3

1.7 Conclusion et ouverture

1.7.1 Conclusion

Pour la prédiction, nous avons eu accès à des variables générales qui décrivent le trajet et les tarifs associés ; cependant, celles-ci ne permettent pas d'expliquer entièrement la variable cible. En particulier, nous n'avons pas accès à des variables caractérisant la qualité du service (comme la courtoisie du conducteur, la douceur de la conduite, la propreté à l'intérieur du véhicule...), ce qui est essentiel pour prédire un pourboire. Par conséquent, il s'agit d'un problème difficile pour lequel il n'est pas possible d'obtenir des résultats très précis avec l'ensemble de covariables actuel. Cela s'est traduit par une sorte de barrière de performance que nous n'avons pas pu surmonter : même en essayant des approches plus fines ou des recherches d'hyperparamètres plus poussées, une fois un certain point atteint, ces méthodes n'avaient plus d'impact sur notre score R^2

1.7.2 Ouverture

Certaines idées n'ont pas pu être mises en œuvre par manque de temps. En particulier, nous pensons qu'un traitement différent des valeurs aberrantes pourrait améliorer le problème de multicolinéarité, ce qui permettrait d'envisager plus sérieusement l'application de modèles linéaires. Bien qu'une régression linéaire simple n'obtiendrait probablement pas de résultats exceptionnels, nous estimons qu'un modèle GAM avec des termes d'interaction appropriés pourrait offrir une bonne performance.

Par ailleurs, nous pensons qu'il serait possible d'obtenir un encodage plus informatif des variables de localisation. Regrouper les emplacements par quartiers semble être une idée naturelle à explorer. Une autre approche consisterait à faire en sorte que les groupes appropriés soient découverts automatiquement à l'aide d'un embedding. La motivation de cette idée est expliquée en annexe.

Une autre idée intéressante consisterait à revoir la manière dont les distances entre les points de prise en charge et de dépose sont calculées : évidemment, un taxi ne peut pas se déplacer librement dans l'espace, donc la distance euclidienne entre le point de prise en charge et le point de dépose n'est pas une mesure adéquate. La distance de Manhattan [5] fournirait peut-être des mesures plus réalistes.

Enfin, en ce qui concerne les modèles, il serait judicieux d'étudier les résidus de nos meilleurs modèles pour voir s'ils ressemblent à du bruit blanc, et, dans le cas contraire, essayer de déduire les effets des variables qu'ils n'ont pas réussi à capturer. Une fois cela fait, nous disposerions de modèles considérablement affinés, et il serait pertinent d'envisager une réunion d'experts.

2 Prédiction du type de végétation en forêt

2.1 Introduction

Les données utilisées dans cette étude concernent des parcelles forestières de 30 m x 30 m, chaque observation représentant une parcelle spécifique. Ces données ont été collectées dans la forêt nationale de Roosevelt, située dans le nord du Colorado. L'objectif de cette étude est de prédire le type de forêt de chaque parcelle en fonction de ces données. Le jeu de données d'entraînement comprend des informations sur 581 000 parcelles. Nous détaillerons les variables en annexe. Nous disposons également du type de foret, valeur que nous souhaitons predire : **1. Spruce/Fir, 2. Lodgepole Pine,**

3. Ponderosa Pine, 4. Cottonwood/Willow, 5. Aspen, 6. Douglas-fir, 7. Krummholz.

2.2 Analyse des données

Commençons d'abord par observer la matrice de correlation.

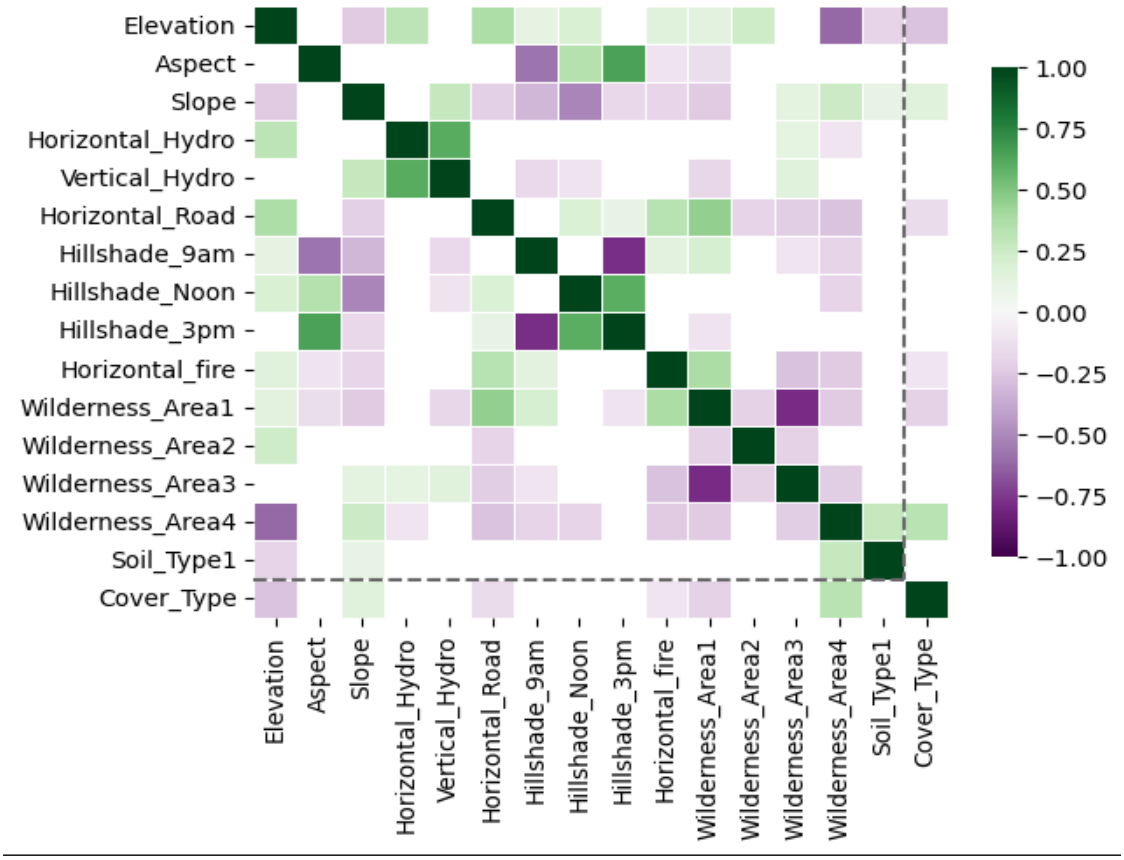

FIGURE 5 – Matrice de correlation

Variables corrélées négativement

— Elevation - Area4 : Cette corrélation négative pourrait indiquer que, dans la zone 4, on observe souvent des altitudes plus faibles.

— Aspect - Hillshade_9am : L'orientation (*Aspect*) est inversement liée à l'ombre de la colline à 9h, ce qui pourrait signifier que certaines orientations réduisent l'exposition à la lumière directe le matin.

— Hillshade_9am - Hillshade_3pm : Les indices d'ombre à 9h et 15h sont négativement corrélés, suggérant que des zones bien éclairées le matin tendent à être plus ombragées en après-midi.

— Area1 - Area3 : Les zones 1 et 3 présentent une relation inverse, peut-être en raison de différences topographiques ou environnementales marquées.

`Variables corrélées positivement`

— Aspect - Hillshade_3pm : L'orientation influence positivement l'ombre à 15h, montrant que certaines orientations reçoivent plus de lumière à cette heure.

— Horizontal_Distance_To_Hydrology - Vertical_Distance_To_Hydrology : Les distances horizontale et verticale par rapport aux points d' eau sont positivement liées, indiquant que les parcelles proches d'un cours d'eau horizontalement le sont aussi souvent verticalement.

— Hillshade_Noon - Hillshade_3pm : Les ombrages à midi et 15h sont liés, montrant une continuité dans l' exposition solaire l'après-midi.

On observe qu'il n'y a pas de corrélation très forte. Aucune variable n'a été supprimée (le choix esthétique est de ne pas inclure toutes les variables `soil_type` dans la matrice de corrélation, bien que celle-ci soit présente dans le code, sans grands changements).

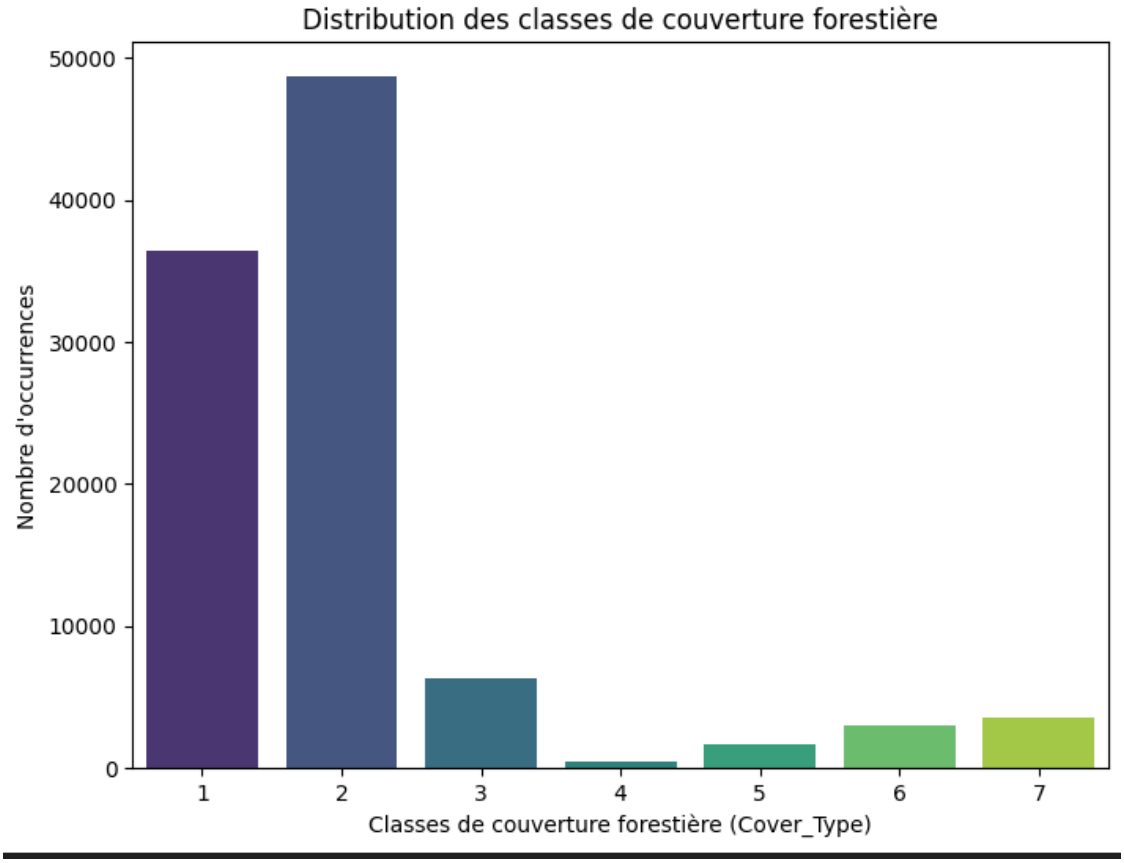

FIGURE 6 – Distribution des classes

Les classes 3, 4, 5, 6 et 7 sont sous-représentées par rapport aux classes 1 et 2. Cela peut entraîner plusieurs problèmes, notamment un biais de prédiction, car les modèles de machine learning auront tendance à privilégier les classes majoritaires. De plus, les performances du modèle peuvent être réduites pour les classes sous-représentées. Des solutions possibles à explorer incluent l'oversampling et l'undersampling.

2.2.1 Modification des variables

Après une première analyse de notre jeu de données, nous constatons que la variable Elevation prend des valeurs variées. Nous choisissons donc de rajouter une variable, l'élévation standardisée pour mieux normaliser les données. Nous appliquons le même traitement à la variable Horizontal_Distance_To_Roadways, qui prend également des valeurs élevées.

Ensuite, les variables Aspect et Slope sont exprimées en degrés, ce qui entraîne une discontinuité entre les directions proches (0° et 360°), éloignées numériquement. Pour corriger cela, nous les transformons en leurs valeurs cosinus et sinus afin de représenter la direction de manière plus continue et linéaire.

Enfin, pour les distances à l'hydrologie (horizontale et verticale), nous les élevons au carré. Étant donné que ces distances sont faibles, cette transformation nous permet de mieux différencier les distances faibles et moyennes, et de capturer ainsi plus finement les relations entre ces distances et la variable cible. Nous gardons pour l'instant en plus de ces

nouvelles variables les variables initiales, nous ne savons pas si c'est transformations augmente ou non les performances. Nous étudions également les variables binaires de notre dataset.

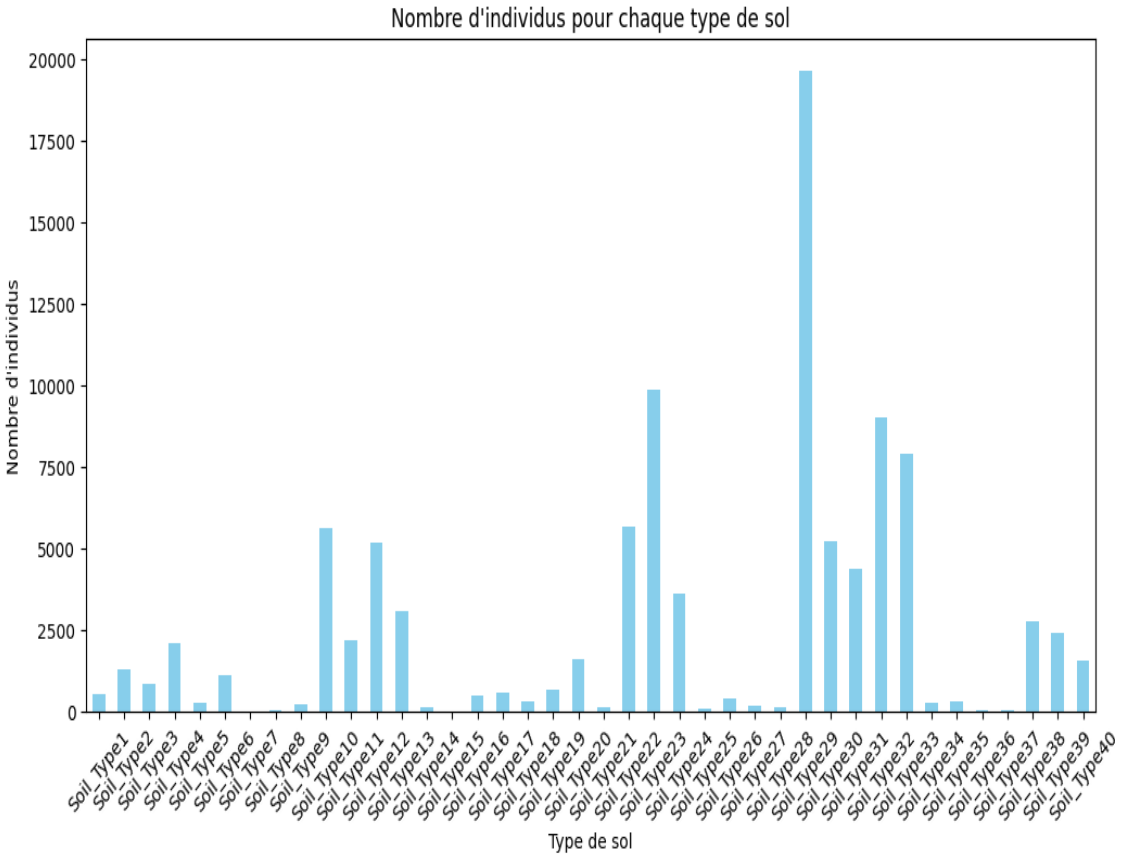

FIGURE 7 – Distribution des classes

On observe que certains types de sol contiennent peu d'individus. On peut se demander si il peut etre pertinent de retirer ces variables. Nous avons donc testé de retirer les variables "Soil" ayant moins de 200 occurrences. En effet, les variables binaires avec peu d'exemples apportent peu d'information et peuvent augmenter le risque de surajustement.

Nous testons une variété de modèles de prédiction que nous détaillerons plus tard dans la partie "Modèles et Performance". Deux modèles ressortent cependant comme ayant les meilleurs résultats : Random Forest et l'algorithme de boosting XGBoost. Nous nous intéressons à leurs variables explicatives.

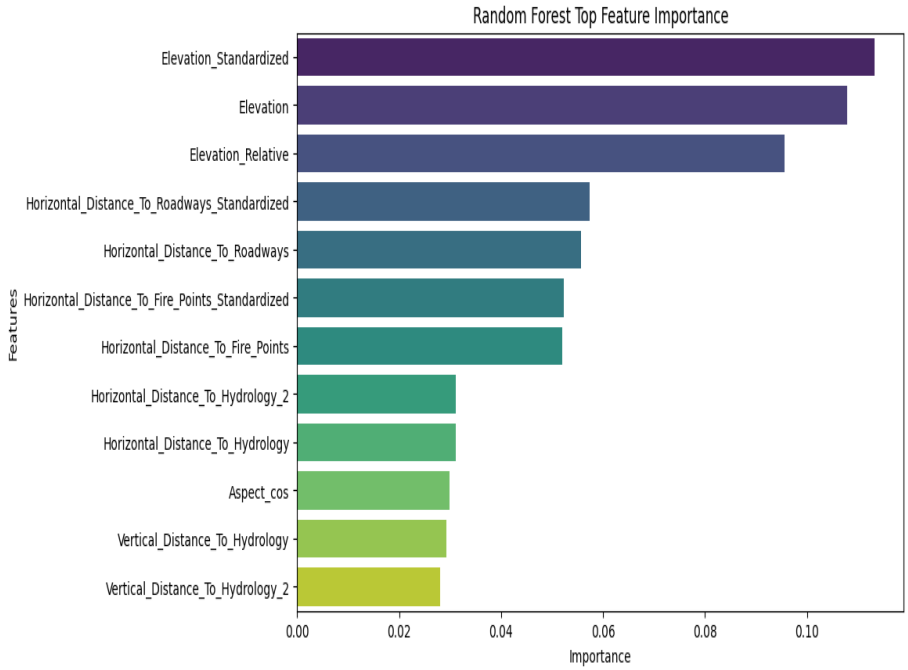

FIGURE 8 – Importance des features pour Random Forest

On remarque que les variables importantes sont celles de distance majoritairement.

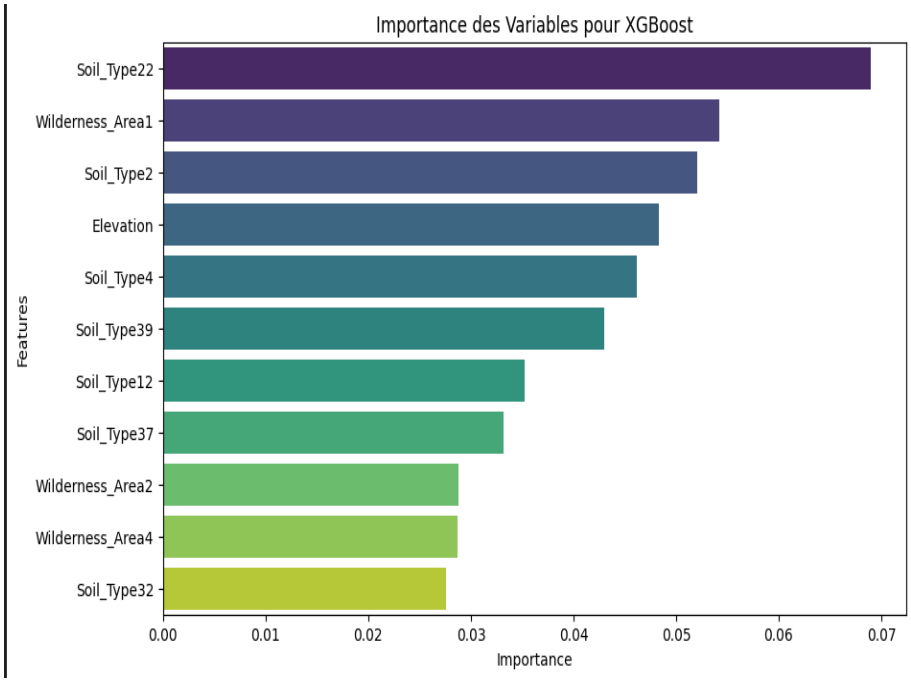

FIGURE 9 – Importance des Features pour XGBoost

Les variables importantes sont principalement celle de sol et d'arene.

On constate que les deux modeles n'ont pas les memes variables, XGB donne plus d'importances aux types de sol et random forest les distances. La difference peut s'explique par la difference du role des variables pour ces deux algorithmes. Cela nous informe cependant que il existe une relation complexe entre les types de sol et les cover types. Il serait donc interessant de chercher a regrouper les types de soles en moins de variables. Les deux modèles ont cependant une variable en commun dans celles qui ne sont pas representatives que l'on enleve de notre jeu de donnée(Soil_Type15).

2.3 Modèles et Performances

Dans cette section nous parlerons des modèles que nous avons utilisé et des differents resultats obtenues.

2.3.1 Modèles

Pour notre étude, nous avons examiné plusieurs modèles de classification : régression logistique, Bagging, Gradient Boosting, XGBoost, Random Forest, ainsi que le SVM (avec noyaux gaussien et polynômial). Les hyperparamètres utilisés initialement étaient ceux par défaut.Nous avons entrainé les modèles sur les données non modifié et modifié pour ameliorer les performances.

Ensuite, nous avons cherché à optimiser les hyperparamètres des modèles les plus performants (XGBoost, Bagging et Random Forest) en utilisant l'algorithme `RandomizedSearchCV`. Cet algorithme fonctionne de manière similaire à une recherche par grille (`GridSearchCV`), mais sélectionne de manière aléatoire les valeurs des hyperparamètres à tester à partir de la grille définie. Le suffixe `CV` signifie que les résultats sont validés par validation croisée. Les hyperparamètres optimisés sont les suivants :

- `XGBoost` :

  `{subsample :0.9,  n_estimators: 500, max_depth: 8,learning_rate: 0.267,`
  `colsample_bytree: 0.7, random_state: 42`
  `}`

- `Random Forest` :

  `{n_estimators: 192, max_depth: 38, min_samples_split: 2, min_samples_leaf: 1}`
  `cit`
- `Bagging` :

  `{n_estimators: 190, max_samples: 1.0, max_features: 0.875}`

2.3.2 Métriques utilisées

La métrique principale utilisée dans le challenge Kaggle est le score F1, qui a servi comme point central d'évaluation des modèles. Trois méthodes d'évaluation ont été considérées :

F1 sur le set de test : La première métrique est l'erreur de test. Pour l'ensemble de test, nous avons prélevé 20% du jeu d'entraînement, en veillant à équilibrer la sélection avec 20% d'individus de chaque classe de la variable 'Cover_Type'. Comme dans notre cas on dispose de plusieurs classe, le score que l'on utilise est celui de la moyenne des valeurs de F1 pour les differentes classes pondéré par le nombre d'individu present dans la classe.

F1 avec validation croisée : Nous avons également évalué les modèles par validation croisée en divisant l'ensemble d'entraînement en cinq parties. La validation croisée permet de limiter le surapprentissage (overfitting) et d' obtenir une évaluation plus fiable. Comme pour le set test on utilise F1 en ponderant en fonction du nombre d'individu des classes.

Score public Kaggle : Enfin, la dernière métrique est le score public de Kaggle, qui correspond au F1 score calculé sur les données de test du challenge.

2.3.3 Résultats

Les résultats obtenus avec les différents modèles ont été compilés dans le tableau suivant :

Modèle	**Test set accuracy**	**Test set Weighted avg**	**CV f1_weighted**	**Kaggle Private Score**
Régression logistique	0.68	0.66	NA	0.713
Bagging	0.92	0.92	0.906	0.916
Gradient Boosting	0.77	0.77	0.769	NA
XGBoost	0.92	0.92	0.911	0.919
Random Forest	0.90	0.90	0.89	0.900
SVM gaussien	0.70	0.68	NA	NA
SVM polynôme	0.70	0.67	NA	NA
Bagging (données modifiées)	0.92	0.92	0.904	0.914
Random Forest (données modifiées)	0.91	0.91	0.901	0.909
XGBoost (données modifiées)	0.92	0.92	0.909	0.919
Bagging (Hyperparamètres)	0.92	0.92	0.912	0.920
Random Forest (Hyperparamètres)	0.90	0.90	0.892	0.901
XGBoost (Hyperparamètres)	0.92	0.92	**0.914**	0.921
XGBoost (Hyperparamètres) (données modifiées)	**0.93**	**0.93**	**0.914**	**0.923**

2.3.4 Conclusion

Les modèles utilisant des méthodes d'ensembling, comme Bagging, Random Forest et XGBoost, se révèlent être les plus performants selon les différentes métriques employées. Parmi eux, XGBoost se distingue comme le meilleur modèle. Bien que la modification des variables et l'optimisation des hyperparamètres n'aient pas entraîné de changements drastiques dans les performances, elles ont permis d'obtenir une légère amélioration. Notamment l'optimisation des Hyperparamètres permet a notre modèle de mieux interpreter les variables ajoutés. Notre modèle le plus performant sur toutes les métriques d'utilisation est XGBoost avec les Hyperparametres optimisé et les données modifié.

Les résultats montrent que les modèles étudiés offrent des performances prometteuses. Cependant, certaines pistes d'amélioration n'ont pas encore été explorées et mériteraient une attention particulière. Tout d'abord, dans l'optimisation des variables, il serait intéressant d'examiner le regroupement des types de sol, notamment en utilisant des méthodes non supervisées pour les classifier. Une première approche à ce sujet est proposée en annexe. L'over-sampling et l'under-sampling constituent également des pistes à approfondir ; une première approche sera également présentée en annexe, mais elle nécessite un développement plus poussé.

L'agrégation des résultats des différents modèles testés pourrait également être une stratégie prometteuse. En combinant les prédictions par un vote pondéré favorisant les modèles les plus performants, nous pourrions améliorer la capacité de généralisation. Enfin, l' utilisation de réseaux de neurones pourrait également s' avérer intéressante, en effet les réseaux de neurones sont éfficace pour capturer des relations complexes et non linéaires entre les variables.

3 Annexes

3.1 Prédiction des pourboires en taxi

3.1.1 Description des variables

Variable	Description
VendorID	Code représentant le fournisseur des données correspondant au trajet (1 : Creative Mobile Technologies, 2 : VeriFone Inc.)
tpep_pickup_datetime	Date et heure de début de la course
tpep_dropoff_datetime	Date et heure de fin de la course
passenger_count	Nombre de passagers (indiqué par le conducteur)
trip_distance	Distance correspondant à la course
PU_location_lat	Latitude (départ de la course)
PU_location_lon	Longitude (départ de la course)
DO_location_lat	Latitude (arrivée de la course)
DO_location_lon	Longitude (arrivée de la course)
RateCodeID	Type de tarif (1= Standard rate, 2=JFK, 3=Newark, 4=Nassau or Westchester, 5=Negotiated fare, 6=Group ride)
store_and_fwd_flag	Variable binaire indiquant si les informations ont été stockées en mémoire dans le véhicule avant d' être transmises (Y=yes, N=no
payment_type	Type de paiement (1= Credit card, 2= Cash, 3= No charge, 4= Dispute, 5= Unknown, 6= Voided trip)
fare_amount	Tarif de la course sur la base de la distance et du temps
Extra	Montant des extras
MTA_tax	Taxe MTA
Improvement_surcharge	Taxe additionnelle
tolls_amount	Montant des péages payés durant le trajet
Congestion_Surcharge	Taxe additionnelle
airport_fee	Frais d' aéroport

3.1.2 Encodage des variables d'emplacement

Une observation que nous avons faite en explorant diverses transformations est que, malgré le fait de travailler avec des centaines de milliers de trajets en taxi, en regroupant la latitude et la longitude en une seule variable de coordonnées, nous avons constaté qu'il n'y avait qu'environ 240 lieux où les chauffeurs de taxi prenaient ou déposaient leurs clients. Nous soupçonnons que la personne ayant construit le jeu de données a appliqué un certain type de clustering aux coordonnées et a remplacé les coordonnées de chaque trajet par le barycentre du cluster auquel elles appartenaient.

Cela suggère l'idée de transformer les coordonnées en variables catégorielles avec 240 modalités. Cependant, après un encodage one-hot, cela créerait trop de variables, c'est pourquoi nous n'avons pas poursuivi cette idée. Une possibilité serait de transformer cet espace de 240 variables orthogonales en un espace de dimension nettement inférieure à l'aide d'un embedding. Nous suggérons cette approche dans la section d'ouverture.

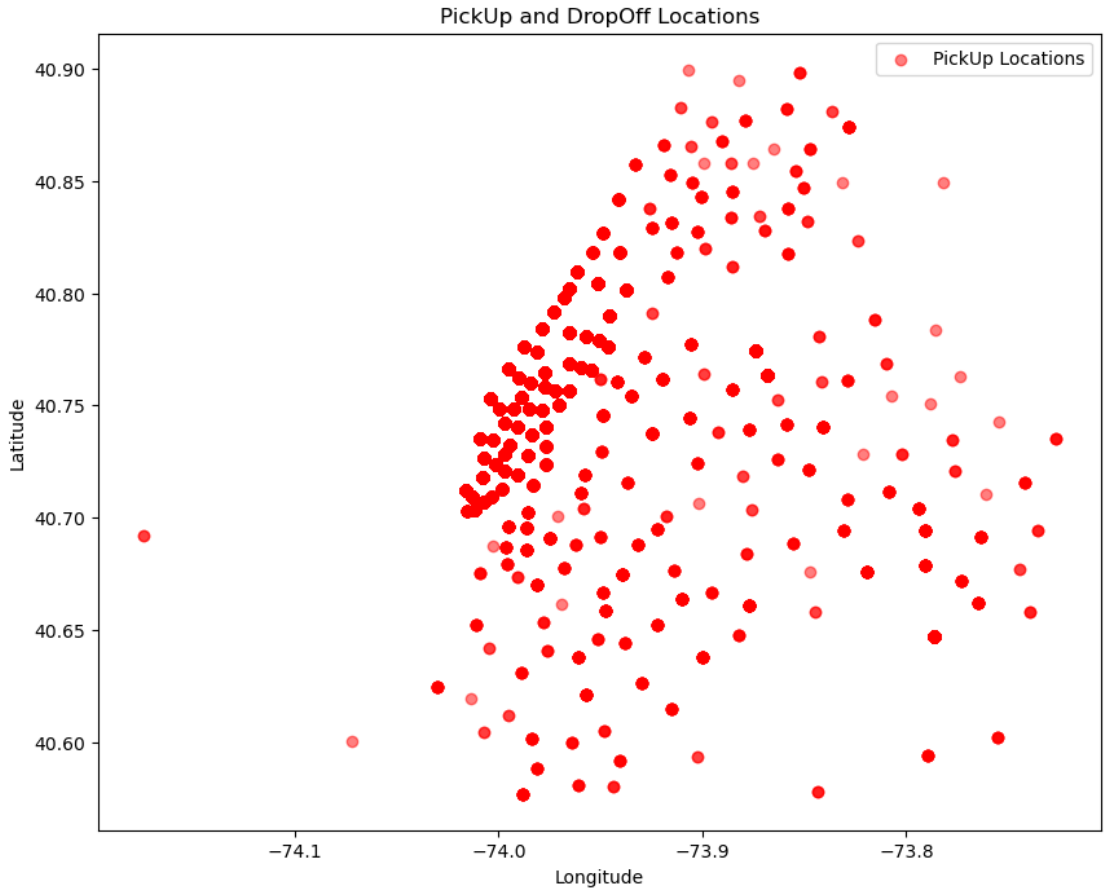

FIGURE 10 – Lieux de prise en charge

3.2 Prédiction du type de végétation

3.2.1 Description des variables

Variable	Description
Elevation	Altitude en mètres.
Aspect	Azimut en degrés.
Slope	Pente en degrés.
Horizontal_Distance_To_Hydrology	Distance à la source d'eau (m).
Vertical_Distance_To_Hydrology	Distance verticale à l'eau (m).
Horizontal_Distance_To_Roadways	Distance à la route (m).
Hillshade_9am, Hillshade_Noon, Hillshade_3pm	Indice d'ombre (0-255).
Horizontal_Distance_To_Fire_Points	Distance aux points d'incendie (m).
Wilderness_Area	Zone sauvage (4 colonnes binaires).
Soil_Type	Type de sol (40 colonnes binaires).
Cover_Type	Type de couverture forestière (1 à 7).

TABLE 3 – Extrait de la documentation des données

Nous pouvons également commencer par avoir quelques informations supplémentaire sur ces types de couverture forestiere.

Les forêts de **Spruce/Fir** se trouvent dans des zones froides et humides, souvent en altitude, dans des régions montagneuses. Les forêts de **Lodgepole Pine** sont principalement composées de pins de montagne, dont les cônes ne s'ouvrent qu'après un incendie, rendant ces forêts régulièrement exposées aux feux. Les forêts de **Ponderosa Pine** se caractérisent par de grands arbres à l'écorce épaisse, résistants aux incendies de faible intensité. Les forêts de **Cottonwood/Willow** sont dominées par des peupliers et des saules, et se trouvent souvent près des rivières et des zones humides. Les forêts d'**Aspen** se régénèrent rapidement après un incendie ou toute perturbation, grâce à la capacité de l'aspen à repousser à partir de ses racines. Les forêts de **Douglas-fir** sont dominées par de grands arbres, capables de vivre très longtemps. Enfin, les forêts de **Krummholz** sont des forêts de conifères naines qui croissent dans des conditions difficiles, généralement à la limite de l'arbre, dans des zones montagneuses subalpines ou alpines.

3.2.2 Over Sampling

Nous avons commencé à réaliser de l'over-sampling en utilisant la bibliothèque `SMOTE` pour générer de nouvelles instances de manière synthétique. En regroupant les catégories de `cover_type` les moins représentées, nous avons augmenté le nombre d'individus pour les types 3,6,7 et 4,5. Cette approche permet de conserver a peu près l'ordre de grandeur des catégories tout en rééquilibrant les classes. Nous avons ainsi ajusté les proportions pour obtenir la distribution suivante :

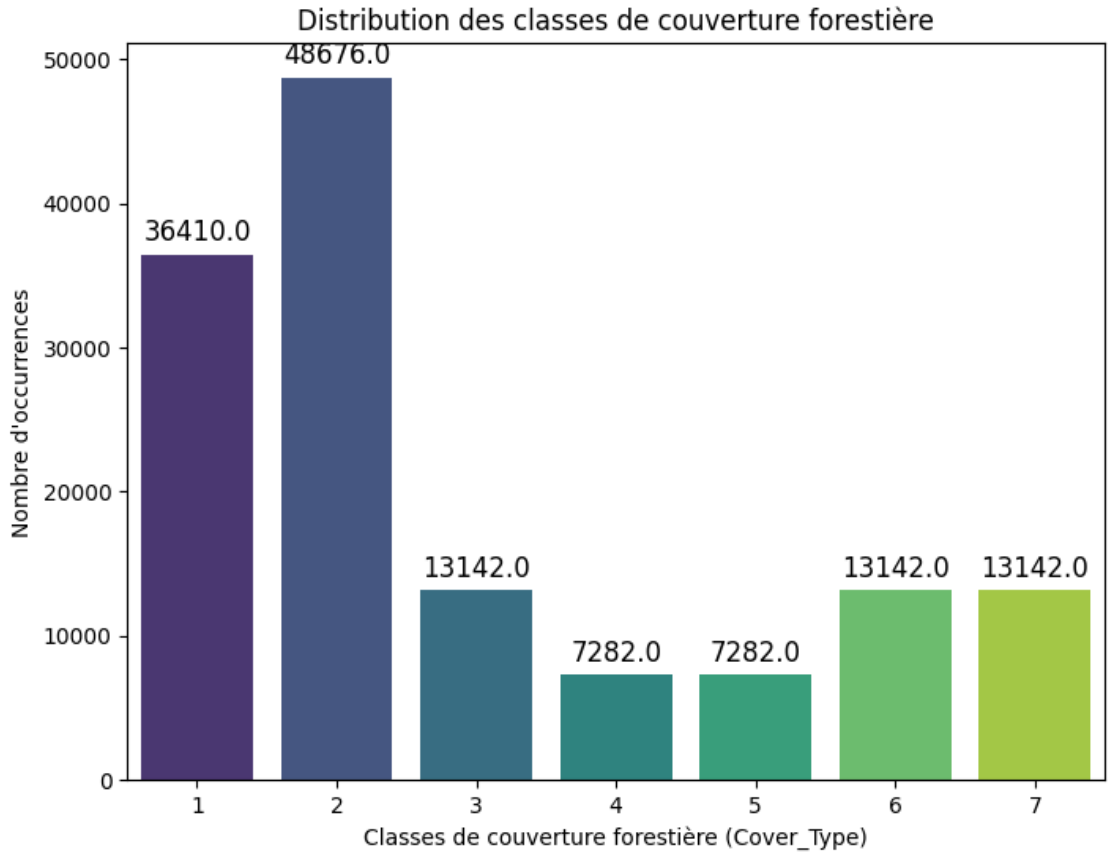

FIGURE 11 – histogramme oversampling

Cela ne fait pas changer les performances des modèles, il pourrait étre interessant de réaliser une e grille search pour avoir une idée de la quantité d'individu a rajouter pour chaque variable.

3.2.3 Regroupement des variables pour le type de sol

ONous nous basons sur la description fournie pour chaque type de sol afin de regrouper certains types en cinq catégories : *Affleurement rocheux*, *Famille Leighcan*, *Pierreux Caillouteux*, *Complexes Cry* et *Sols hydriques humides*.

L'ajout de ces variables, ainsi que l'utilisation de l'oversampling, nous a permis d'obtenir un modèle XGBoost atteignant un score de 0.926 sur Kaggle.

Références

[1] NYC.GOV. *NYC Taxi Fares*. 2022. URL : http://healthit.hhs.gov/portal/server.pt/community/healthit_hhs_gov__university_training__facts_at_a_glance/1430.

[2] Carolin STROBL et al. "Bias in random forest variable importance measures : Illustrations, sources and a solution". In : *BMC Bioinformatics* 8 (2007), p. 25. DOI : 10.1186/1471-2105-8-25.

[3] Christine KERIBIN. *Modélisation Statistique*. Relevant section : 6.5.5. 2024. URL : https://www.imo.universite-paris-saclay.fr/~christine.keribin/STA201/ENSTA-STA201-2024.pdf.

[4] Chris JARRETT. *Categorical Features in XGBoost Without Manual Encoding*. 2023. URL : https://developer.nvidia.com/blog/categorical-features-in-xgboost-without-manual-encoding/.

[5] WIKIPEDIA. *Distance de Manhattan*. 2024. URL : https://fr.wikipedia.org/wiki/Distance_de_Manhattan.