Polytech Sorbonne

Apprentissage Statistique - MAIN5

Forest Cover Type Prediction

Participants :
Fatine BENTIRES ALJ
Alexia ZOUNIAS-SIRABELLA

Encadrant :
Olivier SCHWANDER
Patrick GALLINARI
Arthur PAJOT
Eloi ZABLOCKI

Année 2018 - 2019

Table des matières

1 Présentation du problème

1.1 Informations sur l'étude réalisée

Les managers des ressources naturelles responsables des stratégies de management d'écosystème ont besoin d'informations descriptives basiques comme un inventaire de données sur les forêts des pays pour aider leur processus de décision. Cependant, ils n'ont souvent pas accès à ces données pour des pays hors de leur juridiction. Une des solutions possibles est donc d'obtenir ces données via des modèles de prédiction.

La région étudiée inclus quatre "réserves sauvages" de la "Roosevelt National" forêt. Les données comprennent 12 mesures cartographiques considérées indépendantes tandis qu'il y a sept types de couverture majeurs des forêts utilisés comme des variables dépendantes. De nombreux sous-ensembles de ces variables peuvent être examinés pour déterminer le meilleur modèle.

1.2 Informations sur les données

Nombre d'instances/observations : 581 012
Type de classification : classification supervisée

Pour chaque ligne, la structure est la suivante (10 variables `quantitatives`, 4 colonnes `binaires`, 40 colonnes `binaires`, 1 variable supplémentaire) :

- `Elevation` (en mètre - quantitative)
- `Aspect` (en degré azimut - quantitative)
- `Slope` (en degré - quantitative)
- `Horizontal_Distance_To_Hydrology : Distance horizontale à la plus proche surface d'eau` (en mètre - quantitative)
- `Vertical_Distance_To_Hydrology : Distance verticale à la plus proche surface d'eau` (en mètre - quantitative)
- `Horizontal_Distance_To_Roadways : Distance horizontale à la plus proche route` (en mètre - quantitative)
- `Hillshade_9am` (de 0 à 255 - quantitative)
- `Hillshade_Noon` (de 0 à 255 - quantitative)
- `Hillshade_3pm` (de 0 à 255 - quantitative)
- `Horizontal_Distance_To_Fire_Points : Distance horizontale au plus proche feu de forêt` (en mètre - quantitative)
- `4 colonnes pour désigner la région concernée` (0 si absence et 1 sinon - qualitative)
- `40 colonnes pour décrire le type de sol` (0 si absence et 1 sinon - qualitative)
- `Type de couverture` (1 à 7)

2 Première analyse des données

2.1 Étude générale

2.1.1 Première manipulation des données

Dans un premier temps, nous avons calculé les informations de chaque variable quantitative :

Variable	Moyenne	Min	Max	Quartile 1	Médiane	Quartile 3
Elevation	2959,37	1859	3858	2809	2996	3163
Aspect	155,65	0	360	58	127	260
Slope	14,10	0	66	9	13	18
Horizontal_Distance_To_Hydrology	269,43	0	1397	108	218	384
Vertical_Distance_To_Hydrology	46,42	-173	601	7	30	69
Horizontal_Distance_To_Roadways	2350,15	0	7117	1106	1197	3328
Hillshade_9am	212,15	0	254	198	218	254
Hillshade_Noon	223,32	0	254	213	226	237
Hillshade_3pm	142,53	0	254	119	143	168
Horizontal_Distance_To_Fire_Points	1980,29	0	7173	1024	1710	2550

FIGURE 2 – *Informations relatives aux variables quantitatives*

Les conclusions sont les suivantes :

- `Elevation` : La plupart des arbres ont une altitude comprise entre 2850 et 3250 mètres environ
- `Aspect` : Au vu de la moyenne et des quartiles, on constate que les données sont plus hétérogènes pour cette variable
- `Slope` : Peu d'arbres sont fortement inclinés et la plupart ont une inclinaison de 10 degrés
- `Horizontal_Distance_To_Hydrology` : Au vu de ces informations, la plupart des arbres sont proches d'un point d'eau (entre 100 et 400 mètres)
- `Vertical_Distance_To_Hydrology` : De même, la plupart des arbres sont proches d'un point d'eau (moins de 100 mètres)
- `Horizontal_Distance_To_Roadways` : Pour cette variable, les données sont plutôt divergentes
- `Hillshade_9am` : La plupart des régions sont fortement ombragées et les données sont assez convergentes
- `Hillshade_Noon` : La plupart des régions sont fortement ombragées (plus qu'à 9h) et les données sont convergentes
- `Hillshade_3pm` : La plupart des régions sont ombragées et les données sont assez convergentes. Cependant, l'ombrage est moins important que le reste de la journée
- `Horizontal_Distance_To_Fire_Points` : On remarque que la plupart des arbres ont une distance au départ du feu le plus proche comprise entre 100 et 2500 mètres

2.1.2 Type de couverture

Nous nous sommes ensuite intéressées à la probabilité de d'appartenir à une des sept classes en créant une fonction `type_foret`. Nous observons une grande disparité d'appartenance aux différentes classes et obtenons le graphique suivant :

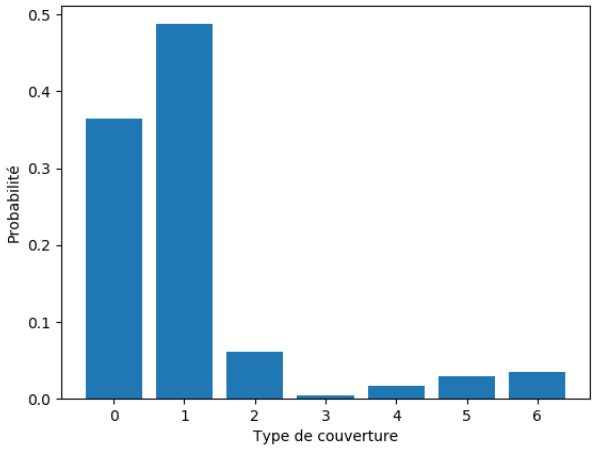

FIGURE 3 – *Probabilité d'appartenir à une classe*

Remarque : Il serait intéressant par la suite de modifier les données de manière à avoir une répartition selon les différentes classes qui soit plus homogène.

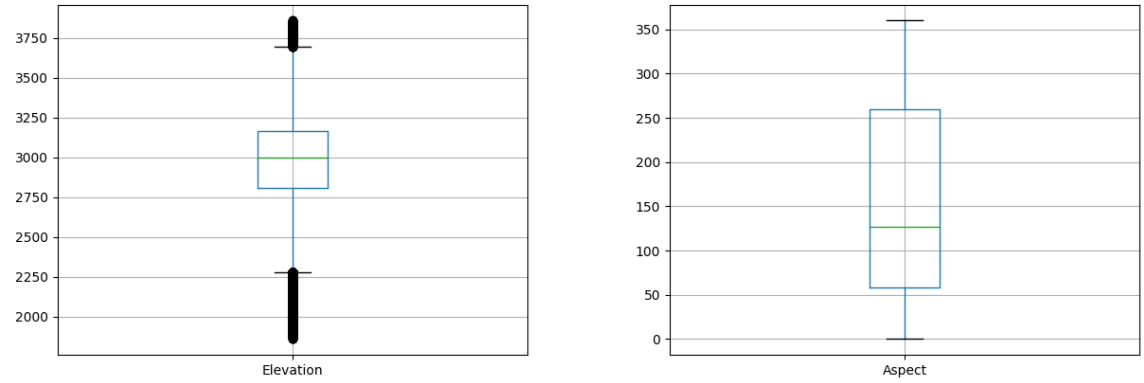

FIGURE 4 – *boxplots associés à la variable Elevation et Aspect*

2.1.3 Boxplot

L'étude des boxplot permet de donner une meilleure idée de l'étendue des valeurs et de leur position.

Nous pouvons observer ci-dessus deux exemples de boxplot associé aux variables "Elevation" et "Aspect" donnés par la fonction `boxplot`.

Elevation : On peut observer un boxplot ayant pour médiane une valeur aux alentours de 3000 (2996 d'après l'étude faite précédemment). De plus l'écart inter-quartile est faible et les données semblent symétriques ce qui montre que les valeurs des variables sont proches. Enfin, nous remarquons la présence de valeurs externes atypiques très éloignées des autres valeurs. De nombreux facteurs peuvent être à l'origine de ces valeurs. Nous déterminerons dans la suite leur origine.

Aspect : On remarque que 50% des arbres ont une orientation inférieure à 125 degrés. De plus l'écart inter-quartile est important ce qui traduit la présence de valeurs étendues. Les valeurs semblent également bien placées puisqu'aucun point ne dépasse du boxplot.

Les conclusions des autres variables quantitatives sont les suivantes :

- `Slope` : Présence d'une médiane valant environ 12 et de beaucoup de valeurs atypiques
- `Horizontal_Distance_To_Hydrology` : Présence d'une médiane valant environ 200 et de beaucoup de valeurs atypiques
- `Vertical_Distance_To_Hydrology` : Présence d'une médiane valant environ 30 et de beaucoup de valeurs atypiques
- `Horizontal_Distance_To_Roadways` : Présence d'une médiane valant environ 1200 et de beaucoup de valeurs atypiques
- `Hillshade_9am` : Présence d'une médiane valant environ 220 et de beaucoup de valeurs atypiques
- `Hillshade_Noon` : Présence d'une médiane valant environ 200 et de beaucoup de valeurs atypiques
- `Hillshade_3pm` : Présence d'une médiane valant environ 150 et de beaucoup de valeurs atypiques
- `Horizontal_Distance_To_Fire_Points` : Présence d'une médiane valant environ 1800 et de beaucoup de valeurs atypiques

Remarque : En observant ces résultats nous pouvons conclure qu'il serait probablement judicieux de regrouper les variables `Hillshade_9am`, `Hillshade_Noon` et `Hillshade_3pm` puisqu'elles semblent avoir un caractère similaire. De plus, il serait également intéressant de regrouper les variables `Horizontal_Distance_To_Hydrology` et `Vertical_Distance_To_Hydrology` puisqu'elles font toutes les deux références à une distance vers un point d'eau et semblent avoir le même comportement.

2.2 Analyse en Composantes Principales

Données : n individus sur p variables quantitatives
But : Explorer les liaisons entre les variables (leurs corrélations) et les ressemblances entre individus (leurs distances). Représenter n individus dans un espace de taille k en perdant le moins d'information possible.

Les résultats peuvent être interprétés pour différentes raisons :

- Mesurer la qualité des représentations
- Expliquer la position des individus

Après avoir normalisé les données et sélectionné les données quantitatives, on essaie de déterminer la dimension k avec laquelle nous allons représenter nos données.

2.2.1 Détermination du nombre de facteurs

Méthode 1 : On utilise le ratio des variances. On obtient le graphique ci-dessous dans lequel on cherche la "cassure" :

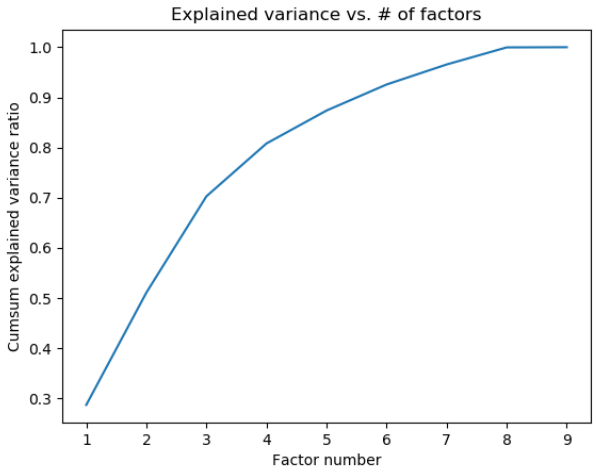

FIGURE 5 – *Variances cumulées*

On peut penser que l'on aura $k = 3$. Nous allons tout de même mettre en oeuvre une deuxième méthode pour valider ce résultat.

Méthode 2 : Il s'agit du test des bâtons brisés utilisant les valeurs propres et des valeurs seuils pour les 10 variables quantitatives.

Variable	Seuil	Valeurs propres
Elevation	2.92	2.58
Aspect	1.92	2.14
Slope	1.42	1.74
Horizontal_Distance_To_Hydrology	1.09	1.07
Vertical_Distance_To_Hydrology	0.84	0.77
Horizontal_Distance_To_Roadways	0.64	0.54
Hillshade_9am	0.47	0.46
...	...	...

FIGURE 6 – *Test des bâtons brisés*

Après la 3ème variable, la valeur propre est inférieure à la valeur seuil. Cela confirme le résultat $k = 3$.

2.2.2 Représentation des individus

L'étude du `cos2` permet d'estimer la qualité de la représentation. En effet, un `cos2` élevé indique une bonne représentation de la variable sur les axes principaux en considération. Dans ce cas, la variable est positionnée à proximité de la circonférence du cercle de corrélation. On observe dans notre cas que les individus dépendent généralement majoritairement du deuxième ou troisième axe.

Nous obtenons les résultats suivants :

Individu	max cos2_1	max cos2_2	max cos2_3
128516	0.98	x	x
358421	x	0.96	x
477649	x	x	0.95

FIGURE 7 – *Représentation des individus*

Les individus `128516`, `358421` et `477649` sont très bien positionnés puisqu'ils ont une valeur du `cos2` élevée. Si l'on s'intéresse à la probabilité d'être bien positionné, en considérant qu'un individu est bien positionné si il a un `cos2` supérieur à la moyenne des `cos2` sur l'axe, on trouve que `44%` des individus sont bien positionnées sur l'axe 1, `38%` sur l'axe 2 et `36%` sur le troisième axe. En règle générale les individus sont moyennement bien positionnés sur les axes, on comprend ainsi qu'il nous faudra effectuer des modifications sur les données.

2.2.3 Contribution aux axes

Les contributions permettent de déterminer les individus qui pèsent le plus dans la définition de chaque facteur.

Axe	Individu	Contribution maximale	Moyenne maximale
1	223888	3.78-05	1.72e-06
2	345358	8.04-05	1.72e-06
3	483577	6.44e-05	1.72e-06

FIGURE 8 – *Contribution aux axes*

En observant les valeurs, nous nous rendons compte qu'elles sont relativement basses. Ceci est la conséquence directe de la présence de nombreuses valeurs. De plus, la contribution maximale obtenue pour chaque axe est plus grande que la moyenne correspondante d'un facteur 30/40. Ceci montre que ces trois variables sont déterminantes pour le premier, second axe et troisième axe respectivement.

2.2.4 Cercle de corrélation

En observant le cercle on remarque quelques corrélations cependant l'étude de l'ACP seule n'apporte pas d'information concluante (probablement parce que nous avons pris k = 3). En couplant l'étude de l'ACP donnée par la fonction `ACP` et la matrice de corrélation donnée par `correlation_variables` nous observons les résultats suivants :

- `Elevation`, `Horizontal_Distance_To_Roadways` et `Horizontal_Distance_To_Fire_Points` semblent se comporter de la même façon
- `Aspect` et `Hillshade_9am` ont une correlation de `57%`, `Aspect` et `Hillshade_3pm` ont une correlation de `65%`
- `Vertical_Distance_To_Hydrology` et `Horizontal_Distance_To_Hydrology` ont une corrélation de `60%`
- `Hillshade_9am` et `Hillshade_3pm` ont une corrélation de `78%`
- `Hillshade_Noon` et `Hillshade_3pm` ont une corrélation de `59%`

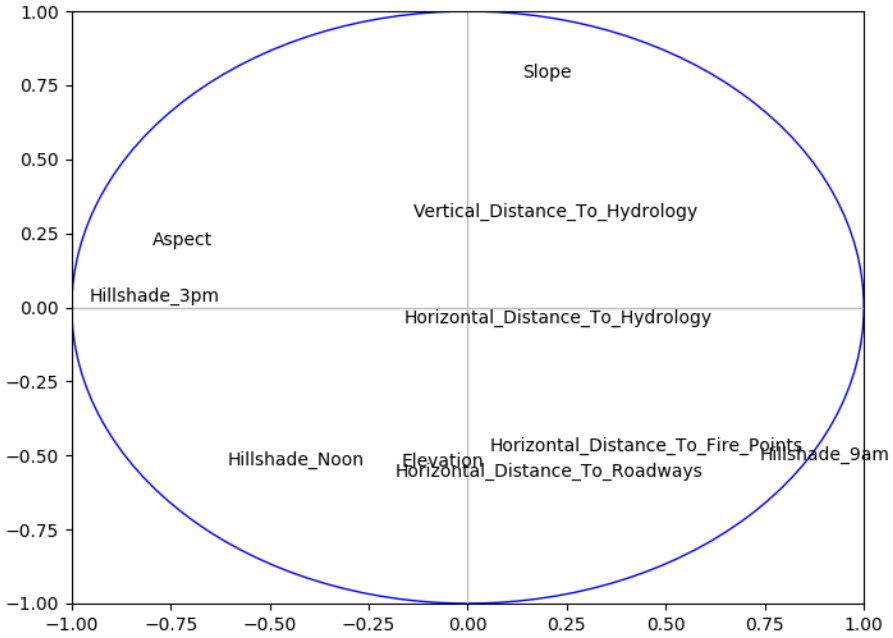

FIGURE 9 – *Cercle de corrélation*

En étudiant les données nous pouvons donc imaginer les modifications suivantes :

- Fusionner `Vertical_Distance_To_Hydrology` et `Horizontal_Distance_To_Hydrology`
- Fusionner `Hillshade_9am` et `Hillshade_3pm` (et non pas `Hillshade_Noon` et `Hillshade_3pm` car la corrélation est plus importante)

3 Pré-traitements et construction des descripteurs

3.1 Méthode de séparation des données

Dans un premier temps nous commençons par séparer les données. Pour cela nous utilisons la fonction `train_test_split` de **sklearn** et nous attribuons 80% aux données d'apprentissage et 20% aux données de test.

3.2 Transformation des données

Cette section regroupe l'ensemble des transformations effectuées sur les données. Les résultats sont visibles dans la fonction `epuration_donnees`. Après mise en place de cette méthode, nous nous retrouvons avec `47565` lignes de données.

3.2.1 Amélioration de la proportion inter-classes

La première modification que nous avons effectué sur les données correspond à l'ajustement des proportions inter-classes de manière à avoir un jeu de données qui soit plus équilibré. Pour cela nous avons procédé en plusieurs étapes :

- Calcul des probabilités d'appartenance à chaque classe
- Récupération du nombre minimal dans toutes les classes confondues : fonction `min()`
- Tri des colonnes selon le `Cover_Type` de manière à pouvoir départager les lignes appartenant à chaque classe : fonction `data.sort_index(by='Cover_Type')` propre à un DataFrame
- Suppression de lignes de manière à avoir environ 2 fois le nombre minimal d'élément par classe (sauf pour la classe minimale) : fonction `data.drop()` propre à un DataFrame (donne `38000` lignes)
- Ajout de données équilibrées entre les classes : utilisation du SMOTE (permet de passer de 38000 à `47565` lignes)

Le SMOTE est une technique qui permet d'ajuster la distribution selon les classes d'un jeu de données. Comme le désequilibre entre les classes dans ces données est très important, nous avions deux choix d'implémentation : soit appliquer le SMOTE sur une fraction des données de manière à avoir un nombre de données raisonnable (pas trop grand) soit supprimer des lignes de manière à avoir un nombre d'éléments par classe à peu homogène puis appliquer le SMOTE de manière à combler les éventuelles erreurs de répartition selon les classes. Notre choix s'est porté sur le second cas car nous voulions traiter dans un premier temps les données "à la main". Après avoir réduit les données jusqu'à `38000` lignes, nous appliquons le SMOTE et obtenons `47565` lignes. En regardant les résultats, nous observons que les matrices de covariances d'apprentissage sont très bonnes et que les résultats sont bons en général c'est pourquoi nous avons estimé que ce nombre de données est suffisant.

3.2.2 Une nouvelle mesure de la distance

Au vu des observations de la partie `2.2.4 Cercle de corrélation`, nous avons décidé d'appliquer la distance euclidienne entre les variables `Vertical_Distance_To_Hydrology` et `Horizontal_Distance_To_Hydrology`. Nous avons commencé par mettre chaque variable au carré à l'aide de fonctions **numpy** puis nous avons sommé et appliqué la racine carrée sur cette somme pour créer la variable `VH_Distance_To_Hydrology`.

3.2.3 Regroupement des variables Hillshade

Dans section nous avons regroupé les variables `Hillshade_9am` et `Hillshade_3pm` en appliquant une soustraction entre les deux et en renommant le résultat `Hillshade_fusion`.

4 Mise en place des méthodes d'apprentissage

Dans cette partie nous nous sommes intéressées aux méthodes d'apprentissage. Chaque méthode présentée ci-dessus est ainsi dotée d'une matrice de confusion sur les données d'entraînement et de test ainsi que d'une méthode de validation croisée sur les qualités de prédiction `np.mean(cross_val_score())`. Un temps d'exécution est également fourni. Celui-ci prend en compte la création du classifieur ainsi que la prédiction mais ne prend pas en compte l'affichage des matrices de confusion. Le temps d'exécution présenté est celui après mise en place des hyperparamètres.

4.1 Méthodes développées

4.1.1 Naive Bayes

La classifier naïve bayésienne est une classification probabiliste simple basée sur le théorème de Bayes. Il appartient à la classe des classifier linéaires. L'avantage de ce classifier est qu'il possède une forte indépendance vis à vis des hypothèses, est facile à mettre en oeuvre et est rapide. Cependant l'inconvénient est qu'il ne peut pas apprendre, il s'agit d'un algorithme probabiliste. En créant la fonction `Naive_Bayes` reposant sur l'utilisation du classifier `GaussianNB()` de **sklearn**, nous obtenons avant pré-traitement des données une qualité de prédiction de `45%` et après pré-traitement une qualité de prédiction de `66%`. En s'intéressant aux matrices de confusion nous observons que l'entraînement n'est pas efficace. Ceci explique les mauvais résultats obtenus.
Temps d'exécution : 0.2288799285888672 secondes

4.1.2 K plus proches voisins

La méthode des K plus proches voisins repose sur le principe suivant : pour chaque point x, on détermine les K plus proches voisins $V_K(x)$. En appliquant cette méthode à l'aide du classifier `KNeighborsClassifier()` de **sklearn** (fonction `KNN`) nous trouvons une qualité de prédiction avant pré-traitement des données de `96.8%` et après pré-traitement de `97.3%`. De plus les résultats obtenus avec la matrice de prédiction d'entraînement sont bons ce qui traduit un bon entraînement et le score basé sur la base de test (au vu de la matrice de confusion) ne semble pas très différent du score basé sur la base d'apprentissage. On peut donc conclure qu'il n'y a pas de sur-apprentissage.
Temps d'exécution : 1.0129117965698242 secondes

4.1.3 Perceptron multi-couches

Le perceptron multi-couches est un type de réseau neuronal utilisé en classification supervisée formé de plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. En appliquant le classifier `MLPClassifier()` de **sklearn** à travers la fonction `perceptron_multi_couches`, nous obtenons une qualité de prédiction avant pré-traitement des données de `48%` et après pré-traitement de `74%`. Ici aussi le score basé sur la base de test (au vu de la matrice de confusion) ne semble pas très différent du score basé sur la base d'apprentissage. On peut donc conclure qu'il n'y a pas de sur-apprentissage.
Temps d'exécution : 22.2500638961792 secondes

4.1.4 Arbre de décision et forêts aléatoires

Ces méthodes sont adaptées à notre cas car elles peuvent s'appliquer non seulement aux variables `quantitatives` mais également aux variables `qualitatives`. Les avantages sont également nombreux : elles sont très rapides, faciles à implémenter et peuvent être utilisées sur des données linéairement non séparables, non équilibrées. En appliquant le classifier `DecisionTreeClassifier()` de **sklearn** (fonction `arbre_de_decision`) nous trouvons une qualité de prédiction avant pré-traitement des données de `88%` et après pré-traitement de `96%`. En appliquant le classifier `RandomForestClassifier()` de **sklearn** (fonction `random_forest`) nous trouvons une qualité de prédiction avant pré-traitement de `93%` et après pré-traitement de `97%`. Il est normal d'obtenir de meilleurs résultats avec la seconde méthode puisqu'elle correspond à un ensemble d'arbres tandis que la première est donné par un unique arbre. En comparant les score basés sur la base de test et d'apprentissage nous remarquons qu'il n'y a pas de grande différence de résultats. On peut donc conclure qu'il n'y a pas de sur-apprentissage.
Temps d'exécution pour la méthode de l'arbre de décision : 0.5589790344238281 secondes
Temps d'exécution pour la méthode de forêts aléatoires : 9.57901906967163 secondes

4.1.5 Régression logistique

La méthode de régression logistique est un modèle de régression linéaire. Le principe est d'associer à un vecteur de variables aléatoire une variable binomiale aléatoire y. Il s'agit d'un cas particulier du modèle linéaire. En créant la fonction `regression_logistique` reposant sur l'utilisation du classifier `LogisticRegression()` de **sklearn**, nous obtenons avant pré-traitement des données une qualité de prédiction de `70%` et après pré-traitement une qualité de prédiction de `78%`.
Temps d'exécution : 4.417582750320435 secondes

4.2 Amélioration des méthodes : mise en place des hyperparamètres

Dans les étapes précédentes nous avons utilisé les hyperparamètres par défaut des classifier. Cependant la modification de ces derniers est susceptible d'améliorer les résultats considérablement. Dans cette partie nous nous sommes donc intéressées à la modification de ces hyperparamètres et particulièrement à ceux de KNN, des arbres de décisions et forêts aléatoires puisqu'il s'agit des classifiers qui fournissent les meilleurs résultats.

KNN : l'hyperparamètre à considérer est le nombre de voisins `n_neighbors`. En le modifiant, nous remarquons que la meilleure valeur est obtenue pour `n_neighbors` = 1

n_neighbors	prédiction
par défaut	0.973
1	0.984
2	0.9805
3	0.9803
4	0.975
10	0.959

FIGURE 10 – *Influence de l'hyperparamètre n_neighbors*

Arbre de décision : les hyperparamètres à considérer sont multiples : le critère `criterion` (par défaut sur l'indice de Gini), la profondeur de l'arbre `max_depth`, le nombre minimal d'exemples nécessaires pour développer un noeud interne `min_samples_split`, le nombre maximal de feuilles de l'arbre `max_leaf_nodes` ...
Nous nous sommes intéressées au hyperparamètres suivants : `criterion`, `max_depth` et `min_samples_split`.

criterion	prédiction
par défaut (indice de Gini)	0.95
entropy	0.96

FIGURE 11 – *Influence de l'hyperparamètre criterion*

max_depth	prédiction
par défaut	0.95
10	0.90
100	0.95
200	0.95

FIGURE 12 – *Influence de l'hyperparamètre max_depth*

min_samples_split	prédiction
par défaut	0.95
2 - 3 - 10	0.95

FIGURE 13 – *Influence de l'hyperparamètre min_samples_split*

Cependant les résultats obtenus avec la plupart des hyperparamètres ne semblent pas concluants. Dans la suite nous ne considérerons uniquement l'hyperparamètre `criterion`.

Forêts aléatoires : là aussi les hyperparamètres à considérer sont multiples. Nous nous sommes intéressées à deux hyperparamètres : le nombre d'itérations `n_estimators` (c'est exactement le nombre d'arbres de décisions qui feront partie de la forêt) et le critère `criterion`.

criterion	prédiction
par défaut	0.968
entropy	0.972

FIGURE 14 – *Influence de l'hyperparamètre criterion*

n_estimator	prédiction
par défaut	0.968
10	0.9684
30	0.974
50	0.976
100	0.977

FIGURE 15 – *Influence de l'hyperparamètre n_estimator*

Nous remarquons ici que le meilleur résultat est donné par `n_estimator` = 100 et `criterion` = 'entropy' c'est pourquoi c'est celui que nous considérons dans notre code.

5 Conclusion

Afin d'étudier ce projet en profondeur nous avons développé/utilisé de nombreuses méthodes de Machine Learning en se basant principalement sur les bibliothèques **Pandas** et **Sklearn**. Ainsi, dans un premier temps, notre intérêt s'est porté sur l'analyse des données afin d'avoir une meilleure compréhension des variables. Pour cela nous avons mis en place des méthodes de visualisation telles que le boxplot ou histogramme et avons développé une analyse autour de l'ACP. Dans un deuxième temps nous avons cherché à appliquer certaines méthodes de classification supervisées (arbres de décision, plus proches voisins,...) afin de se faire une première idée des qualités de prédictions. Finalement, au vu des déductions précédentes, nous avons traité les données (SMOTE, regroupement de variables) ce qui nous a conduit à l'obtention de bons résultats. En considérant notre traitement de données et les modifications faites, les meilleurs résultats, en terme de performance et de temps d'exécution, sont obtenus pour la méthode du `k plus proches voisins` et `forêts aléatoires`.

Bibliographie

[1] Polytech Sorbonne, "Logo Polytech Sorbonne." (https://www.facebook.com/pg/PolytechParisUPMC/photos/?tab=album&album_id=156421134394814).

[2] Sorbonne Université, "Logo Sorbonne." (http://www.sorbonne-universite.fr).

[3] F. Villers, "Analyse de données : Classification supervisée ou Analyse discriminante," 2017. [Notes de cours pour MAIN4 - Polytech Sorbonne].

[4] P. Gallinari, "Apprentissage statistique," 2018. [Notes de cours pour MAIN5 - Polytech Sorbonne].

[5] Eric, "Tanagra Data Mining." (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_ACP_Python.pdf).