Forest Cover Type Prediction

Kevin Zagalo
kevin.zagalo@etu.upmc.fr

Ismail Benkirane
ismail.benkirane@etu.upmc.fr

Projet pour le cours *Apprentissage Statistique* du LIP6, Sorbonne Université — Janvier 2019

Ce projet a pour but de proposer et tester des modèles pour l'étude de la base de données *Covertype*[1], de 581 012 instances, avec 54 attributs et 7 classes à prédire, sans données manquantes.
Les attributs sont les suivants :

Nom	Unité	Description
`Elevation`	mètres	Altitude
`Aspect`	degrés	Orientation
`Slope`	degrés	Pente
`Horizontal_Distance_To_Hydrology`	mètres	Distance horizontale au point d'eau le plus proche
`Vertical_Distance_To_Hydrology`	mètres	Distance verticale au point d'eau le plus proche
`Horizontal_Distance_To_Roadways`	mètres	Distance horizontale à la route la plus proche
`Hillshade_9am`	entier entre 0 et 255	Ombrage à 9h au solstice d'été
`Hillshade_Noon`	entier entre 0 et 255	Ombrage à 12h au solstice d'été
`Hillshade_3pm`	entier entre 0 et 255	Ombrage à 15h au solstice d'été
`Horizontal_Distance_To_Fire_Points`	mètres	Distance horizontale au départ de feu le plus proche
`Wilderness_Area`	4 colonnes binaires	Wilderness area designation
`Soil_Type`	40 colonnes binaires	Type de sol
`Cover_Type`	entier entre 1 et 7	Classe

Il s'agit donc d'un problème de classification multi-classe avec 7 classes.

Table des matières

1. `https://archive.ics.uci.edu/ml/datasets/Covertype`

1 Chargement des données

On choisit d'utiliser la bibliothèque `pandas` pour charger les données, surtout pour l'analyse préliminaire. `pandas` fournit une panoplie de fonctions pour visualiser les données. `groupby`, `boxplot` et `hist` nous seront fort utiles pour choisir les données que nous exploiterons.

```
df_covtype = pd.read_csv('covtype/archives/covtype.data.gz', header=None,
                         parse_dates=dict_attributs, compression='gzip')
```

Le problème de ce chargement est qu'il stocke les données en type `str`, il nous faut donc convertir le type des données. C'est ce que font les fonctions `convert_to_listofbool`, `convert_to_int` et `convert_to_float`, que nous appliquons de la manière suivante :

```
for attribut in dict_attributs:
  if attribut is 'Wilderness_Area':
    wilderness = convert_to_listofbool(df_covtype[attribut])
    df_covtype[attribut] = [x.index(1)+1 for x in wilderness]
  elif attribut is 'Soil_Type':
    soil = convert_to_listofbool(df_covtype[attribut])
    df_covtype[attribut] = [x.index(1)+1 for x in soil]
  elif attribut in ['Cover_Type','Hillshade_9am','Hillshade_Noon','Hillshade_3pm']:
    df_covtype[attribut] = convert_to_int(df_covtype[attribut])
  else:
    df_covtype[attribut] = convert_to_float(df_covtype[attribut])
```

1.1 Variables et architecture des données

- `df_covtype` : *DataFrame* de toutes les données non traitées triées par attributs.
- `df_bycovtype` : *DataFrame.groupby* de toutes les données triées par attributs et regroupée par classes.
- `labels` :*np.array* des étiquettes des types de forêts.
- `dict_attributs` : *dictionnaire* qui associe les attributs aux bons index.
- `qualitative` : *liste* des attributs des données qualitatives.
- `wilderness` et `soil` : *listes* gardant en mémoire les vecteurs binaires pour les remplacer par des entiers.
- `Wilderness_Areas` :
  - 1 : Rawah Wilderness Area
  - 2 : Neota Wilderness Area
  - 3 : Comanche Peak Wilderness Area
  - 4 : Cache la Poudre Wilderness Area
- `Soil_Types` :
  - 1 to 40 : based on the USFS Ecological Landtype Units for this study area
- `forest_cover_types` :
  - 1 : Spruce/Firze
  - 2 : Lodgepole Pine
  - 3 : Ponderosa Pine
  - 4 : Cottonwood/Willow
  - 5 : Aspen
  - 6 : Douglas-fir
  - 7 : Krummholz

1.2 Transformation des données qualitatives

On préfèrera garder dans le DataFrame `df_covtype` des entiers plutôt que des vecteurs binaires, quitte à les y remettre dans les données de train et de test ensuite. Cela facilitera grandement l'analyse préliminaire. On utilisera donc `wilderness` et `soil` uniquement pour la partie test des modèles.

```
In [10]: df_covtype[['Elevation','Aspect','Soil_Type','Wilderness_Area','Cover_Type']].head()
```

Out[10]:

	Elevation	Aspect	Soil_Type	Wilderness_Area	Cover_Type
0	2596.0	51.0	29	1	5
1	2590.0	56.0	29	1	5
2	2804.0	139.0	12	1	2
3	2785.0	155.0	30	1	2
4	2595.0	45.0	29	1	5

Figure 1 – Quelques attributs du DataFrame

2 Analyse préliminaire et pré-traitement des données

Tout d'abord on constate que les données sont inégalement réparties selon les classes. Cela peut vouloir dire plusieurs choses : soit nos données sont mal échantillonnées, soit les types `1` et `2` sont effectivement largement plus répandues. C'est quelque chose dont nous n'avons pas la maitrise, une discussion avec un expert sur le sujet serait préférable. Nous continuerons l'étude sans experts et en supposant que les données sont raisonnablement échantillonnées.

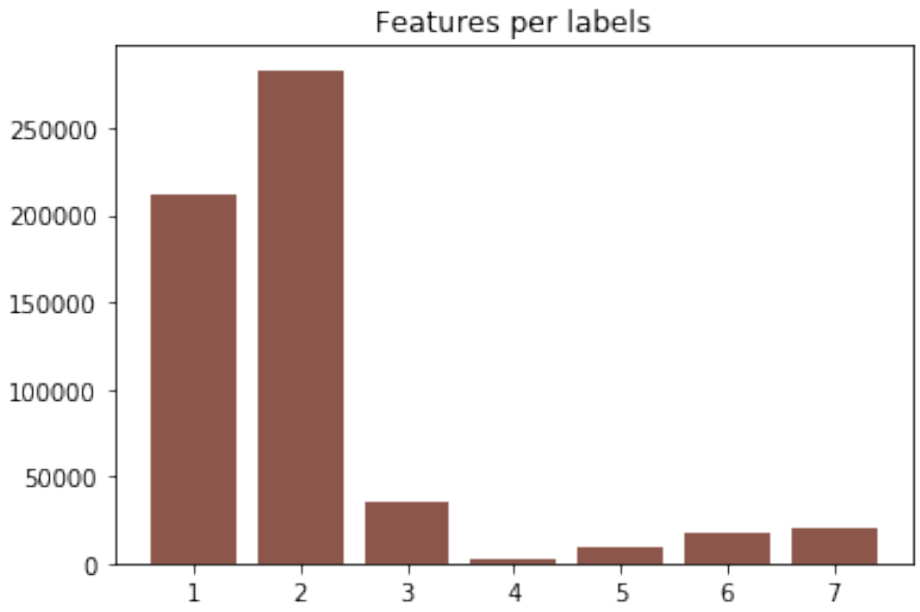
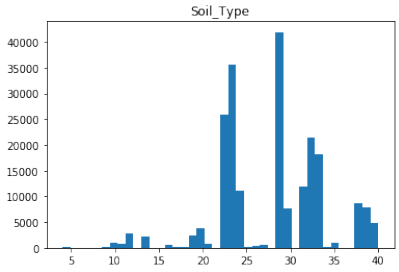
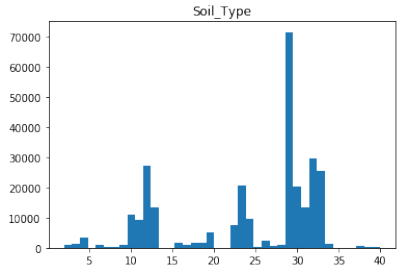


Figure 2 – Histogramme des données par types de forêts

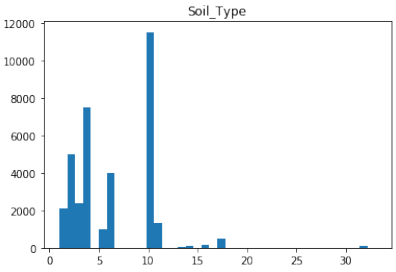
Au vu du nombres de paramètres que comportent les attributs `Soil_Type` et `Wilderness_Area`, on peut se demander si les garder est vraiment utile.



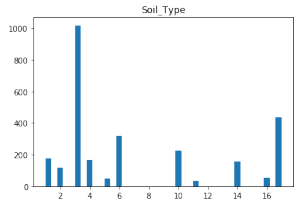
(a) Classe 1



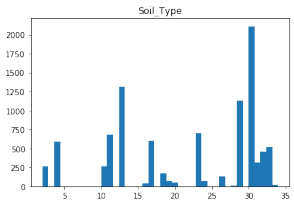
(b) Classe 2



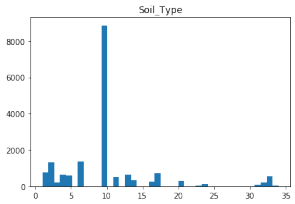
(c) Classe 3



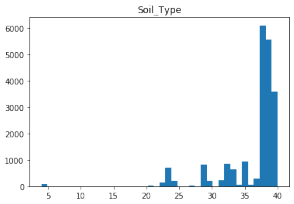
(d) Classe 4



(e) Classe 5

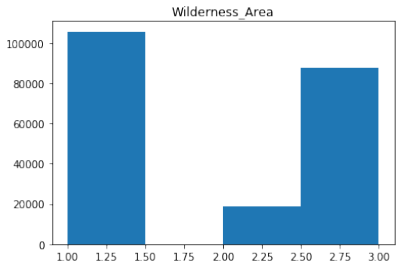


(f) Classe 6

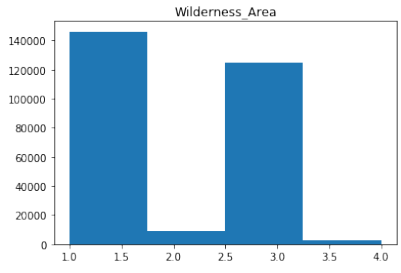


(g) Classe 7

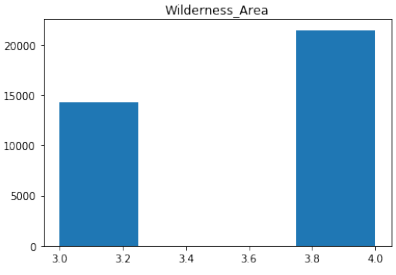
FIGURE 3 – Distribution des types de sols par classe de forêts



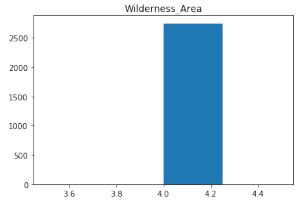
(a) Classe 1



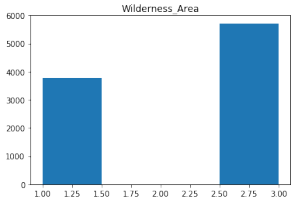
(b) Classe 2



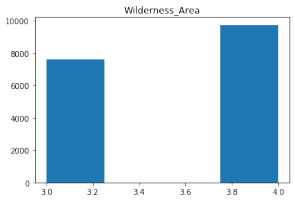
(c) Classe 3



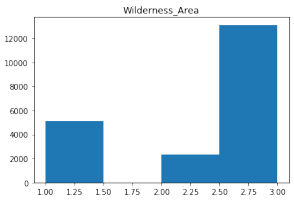
(d) Classe 4



(e) Classe 5



(f) Classe 6



(g) Classe 7

FIGURE 4 – Distribution de l'attribut Wilderness par classe de forêts

3 Test de différents modèles