**Analyse et explications Alexis Savva**

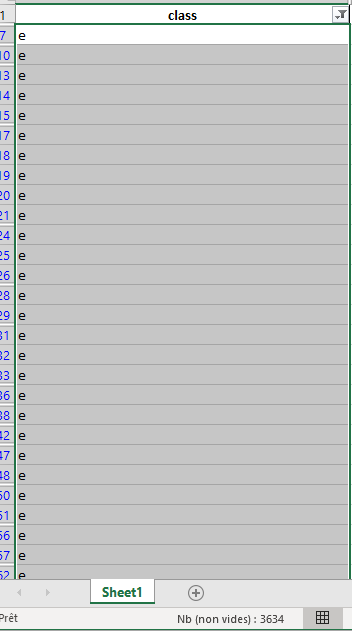
Avant de commencer à coder les modèles, j’ai dans un premier temps analysé le fichier train.csv.

Afin d’éviter d’avoir des résultats non représentatifs, j’ai étudié la répartition des champignons comestibles et vénéneux. Le but étant de voir si les deux classes étaient représentées équitablement.

Lors de mon analyse j’ai donc obtenu les résultats suivants :

Nombre d’entrées dans le fichier train.csv : 7006

Nombre de champignons comestibles : 3634



Nombre de champignons vénéneux : 3372



On constate donc un rapport de 48 % pour les champignons vénéneux et 52% pour les comestibles.

Grâce à cette rapide analyse, je m’assure que les deux classes sont bien représentées équitablement. S’il y avait eu un écart trop grand, j’aurais pu augmenter le nombre de ligne en dupliquant la classe minoritaire pour équilibrer le jeu de données.

Ensuite j’ai codé en Python un programme permettant de transformer les Strings en entier afin d’appliquer les modèles sur le jeu de données. Les fichiers permettant de faire cela se nomment :

* Excel.py
* Main.py
* File.py

Ces trois fichiers me permettent d’avoir un fichier Excel « propre » sur lequel je vais pouvoir tester mes modèles.

**Point d’amélioration**

J’aurais pu faire de l’embedding sur les données, aux lieux de faire un programme en python pour convertir les données string en int. On l’utilise lorsqu’on fait de l’analyse de texte, cela permet de ne pas faire de la corrélation entre les paramètres pour ne pas interférer sur les données.

**Modèle**

Pour répondre au mieux à la problématique, j’ai fait 3 modèles différents pour avoir les meilleurs résultats possibles.

Pour chaque modèle, j’ai entrainé les modèles sur 70% du jeu de données et testé sur 30%.

**Régression linéaire**

Pour ce premier modèle, j’ai testé toutes les valeurs afin d’avoir un premier résultat.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

df = pd.read\_excel('final.xlsx'); def predict(df):

train = df.drop(['class'], axis=1)

test = df['class']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train, test , test\_size=0.3, random\_state=2)

reg = LinearRegression()

reg.fit(X\_train,y\_train)

pred = reg.predict(X\_test)

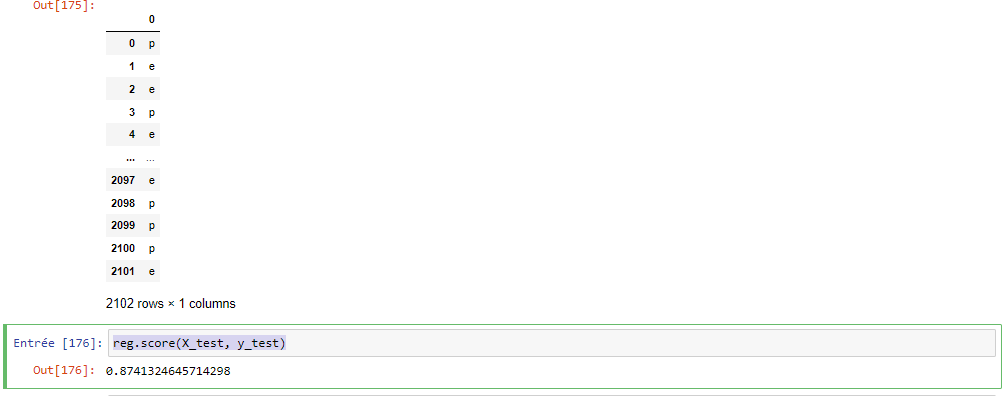
tab\_test = [ 'p' if x > 8 else 'e' for x in pred ]

result = pd.DataFrame(data=tab\_test)

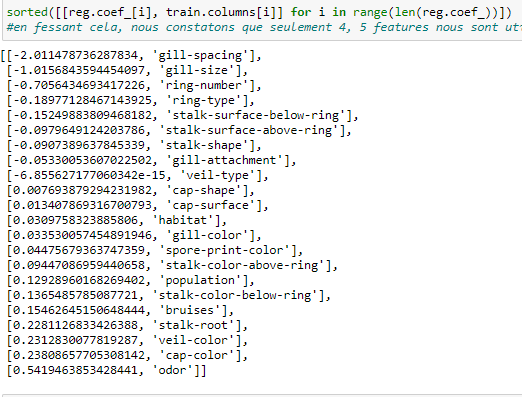
result

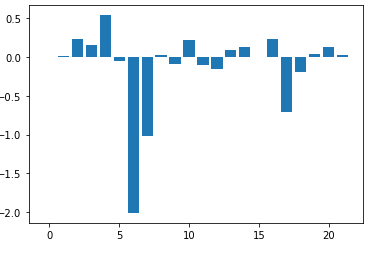
print(reg.score(X\_test, y\_test)

Avec ce modèle j’ai obtenu un score de 87 %



Après cela, il me semblait intéressant d’identifier les variables les plus pertinentes dans le modèle. J’ai donc trié les valeurs et dessiné un graphique.

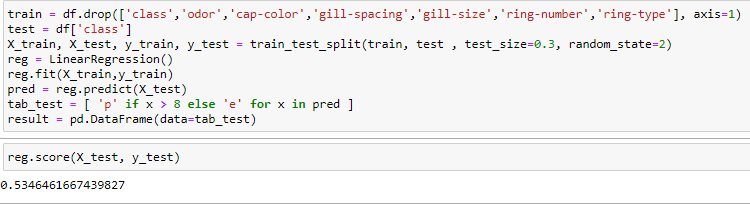




On constate des variables avec des valeurs très significatives. Ces variables sont :

* odor
* gill-spacing
* gill-size
* ring-number
* cap-color

Afin de tester mon hypothèse, j’ai créé le même modèle (régression linéaire) en excluant les variables les plus pertinentes.

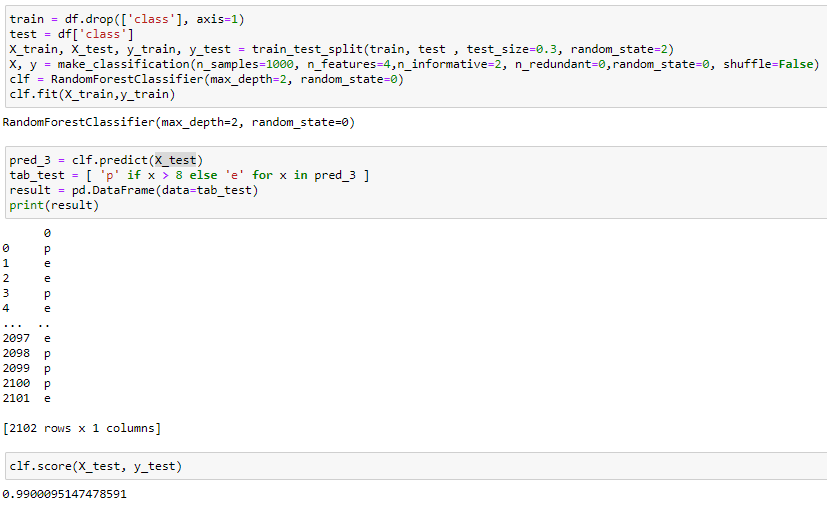


Avec ces nouveaux paramètres, j’ai un score de 53%. En utilisant le même modèle sans ces variables on constate que le modèle n’arrive pas à différencier les champignons comestibles des champignons vénéneux. On peut donc en conclure que ces valeurs sont pertinentes pour la prédiction de la classe d’un champignon.

Ce modèle avec une régression linéaire ne nous permet d’avoir un bon score pour déterminer la classe d’un champignon. Cependant nous avons déterminé les valeurs les plus pertinentes afin de prédire au mieux.

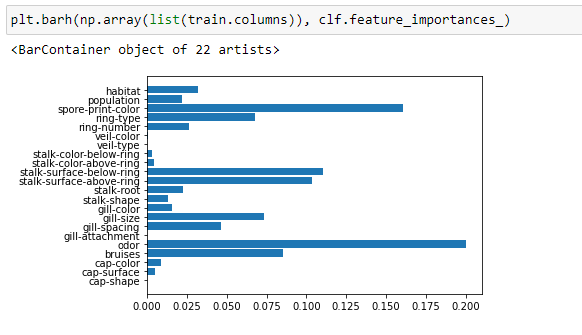
**Forêt d'arbres décisionnels**

J’ai utilisé toutes les variables pour ce modèle-ci.

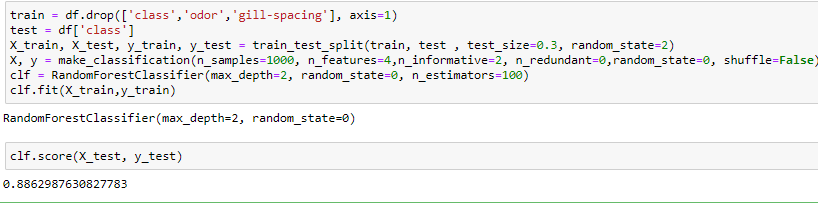


J’ai laissé les paramètres donnés dans la documentation (<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>) pour avoir une première vision des résultats.

Par default le nombre de n\_estimators est à 100. Ce paramètre sert à initier le nombre d’arbre dans la forêt.

J’ai affiché comme pour le modèle d’avant un graphique afin d’analyser les valeurs.

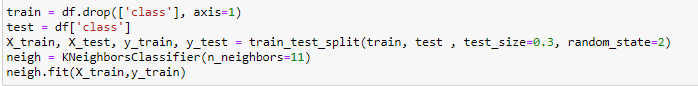
Je test le même modèle sans la valeur la plus pertinente « odor » et sans « gill-spacing » une des valeurs les plus pertinentes de la régression linéaire.



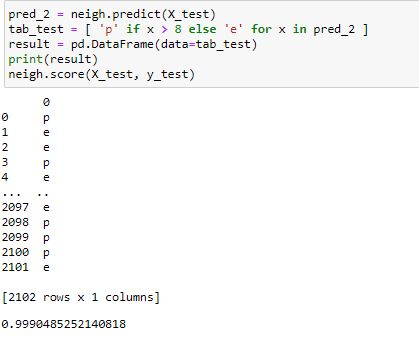
On remarque que nous obtenons un score de 88%. C’est moins bon que notre premier test mais cela reste tout de même un bon score. De plus, on remarque que ces valeurs sont significatives pour un RandomForestClassifier.

Ce modèle comportant un RandomForestClassifier est intéressant car nous obtenons un score qui permet de déterminer la classe d’un champignon et en même temps nous pouvons remarquer la pertinence des valeurs pour le modèle.

**K plus proches voisins**

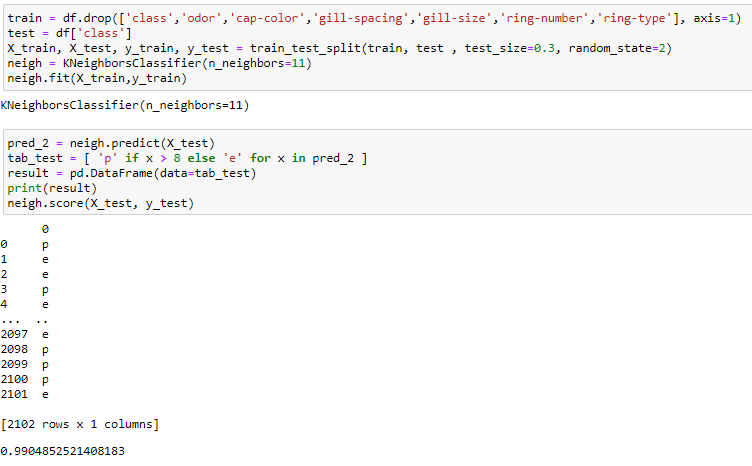
Comme pour les deux premiers modèles, j’ai utilisé toutes les valeurs dans mon premier test.

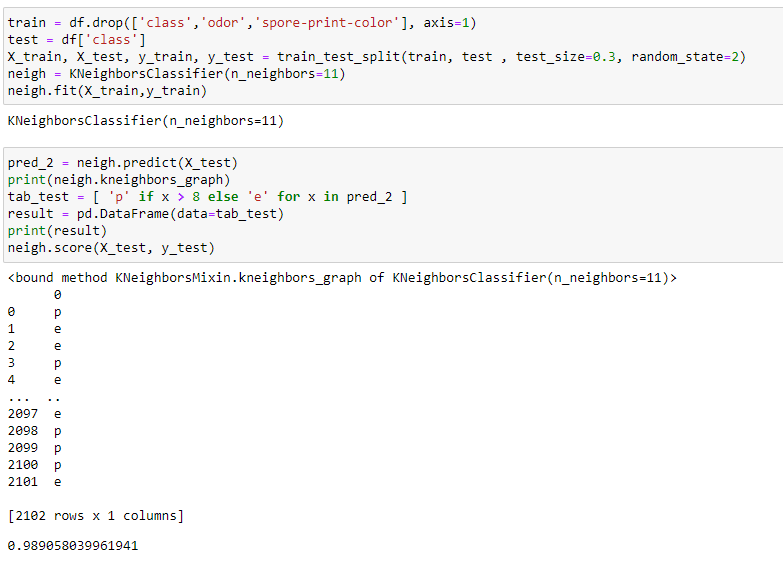
J’ai testé toutes les valeurs pour avoir un premier résultat. J’ai initialisé la variable n\_neighbors à 11 pour avoir des résultats représentatifs car il est impair et assez grand.



Avec ce premier test, j’obtiens un score de 99.9% pour la classification des champignons.

Nous allons maintenant utiliser le même modèle sans les variables les plus pertinentes de la régression linéaire ainsi que les deux variables les plus pertinentes du RandomForestClassifier.





On remarque que nous obtenons un score de 99% et 98%. C’est moins bon que notre premier test mais c’est très satisfaisant. On peut aussi remarquer que les valeurs de la régression linéaire sont moins pertinentes que celle du RandomForestClassifier pour le KNN.

Ce modèle comportant KNN nous permet d’avoir un bon score pour déterminer la classe d’un champignon. Cependant la détermination des valeurs les plus pertinentes est plus compliquée à cerner que pour le modèle de régression linéaire.