Université Paris 2 Panthéon-Assas
Session 2021–22

Machine Learning

Séance 10

L'objectif de cette session est de (i) rappeler le fonctionnement des arbres de décision et (ii) d'appliquer nos connaissances sur python.

1 Importation, description et visualisation des données

1. Importez la base de données
   — Première possibilité (la plus simple)

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target
```

   — Deuxième possibilité (moins simple)

```
import pandas
X = pandas.read_excel("/Users/data/data_X.xlsx")
y = pandas.read_excel("/Users/data/data_y.xlsx")
X = X.to_numpy()
y = y.to_numpy()
```

2. Décrivez la structure des données :
   — Pour la base X : `type(X)`, `X[:10]`, `X.shape)`

```
print(type(X))
print(X[:10])
print(X.shape)
```

   — Quelle est la structure de la base X ? Combien possède-t-elle de lignes ? Combien de colonnes ? Que représentent les colonnes ?

3. Visualisez les données :

```

import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Petal length")
plt.ylabel("Petal width")
```

4. Que représente Y ?

2 Arbre de décision

5. Implémentez un arbre de décision

```
from sklearn import tree
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
predictions = tree_clf.predict(X)
predictions
print(f'% Classification Correct : {tree_clf.score(X, y):.3f}')
```

6. Visualisez les deux arbres de décision suivants. Quelle est la différence principale entre ces deux arbres ?

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth=2)
clf.fit(X, y)
plot = tree.plot_tree(clf,feature_names=iris.feature_names[2:], class_names=iris.target_names,
proportion = True, rounded=True,filled=True)

```

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth=3)
clf.fit(X, y)
tree.plot_tree(clf,feature_names=iris.feature_names[2:], class_names=iris.target_names,
proportion = True, rounded=True,filled=True)
```

7. En utilisant un arbre de décision, prédisez le type de fleur ayant des pétales de 5cm de long et de 1,5cm de largeur :

```
tree_clf = tree.DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
tree_clf.predict_proba([[5, 1.5]])
tree_clf.predict([[5, 1.5]])
```

8. Questions sur les arbres de décision :
   (a) Vrai ou faux : les arbres de décision désignent une méthode linéaire et non-paramétrique
   (b) Vrai ou faux : le fonctionnement des arbres de décision revient à segmenter l'espace de décision
   (c) Vrai ou faux : pour obtenir les meilleures performances, il vaut toujours mieux utiliser des arbres plus profonds
   (d) Quel est le problème principal des arbres de décision simples ?
   (e) A quoi correspond le *Tree Pruning* ?

3 Préparation des données

9. Divisez la base de données en deux sous-bases : une base d'apprentissage et une base de test
   — Utilisez le code suivant :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

   — Combien d'observations contient la base de test ? Quelle proportion de la base de données cela représente ?

10. Vrai ou faux :
    (a) La base d'apprentissage est utilisée pour tester la performance du modèle
    (b) Il suffit de classer aléatoirement les observations avant la division de la base de données en deux pour obtenir toujours les mêmes performances pour un modèle donné
    (c) Il faut toujours mettre 20% des observations dans la base de test et 80% dans la base d'apprentissage

4 Comparaison de plusieurs modèles

4.1 Entraînement

11. Utilisez le code suivant :

```
from sklearn.ensemble import BaggingClassifier

clf = BaggingClassifier(base_estimator=tree.DecisionTreeClassifier(),n_estimators = 10, random_state=10)
clf.fit(X_train, y_train)
print(f'Test : {clf.score(X_test, y_test):.3f}')
print(f'Train : {clf.score(X_train, y_train):.3f}')

from sklearn.ensemble import RandomForestClassifier

RF_Model = RandomForestClassifier()
RF_Model = RandomForestClassifier(random_state=10,n_estimators = 10)
```

```
RF_Model.fit(X_train, y_train)
print(f'Test : {RF_Model.score(X_test, y_test):.3f}')
print(f'Train : {RF_Model.score(X_train, y_train):.3f}')


from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(n_estimators=10, learning_rate=0.1,
      max_depth=1, random_state=0).fit(X_train, y_train)

clf.fit(X_train, y_train)
print(f'Test : {clf.score(X_test, y_test):.3f}')
print(f'Train : {clf.score(X_train, y_train):.3f}')
```

12. Questions sur le code :
    (a) Quel(s) modèle(s) a-t-on utilisé ?
    (b) Quel modèle donne la meilleure performance ?
    (c) Quels sont les caractéristiques de chaque modèle ?
13. Questions sur le bagging :
    (a) Quelle est la différence entre un arbre de décision simple et le bagging ?
    (b) Vrai ou faux : le bagging n'utilise pas le bootstrap
    (c) Vrai ou faux : le bagging utilise un sous-ensemble des prédicteurs séléctionné aléatoirement
    (d) Vrai ou faux : le bagging fait "pousser" des arbres de manière séquentielle
14. Questions sur le random forest :
    (a) Quelle est la différence entre le bagging et le random forest ?
    (b) Vrai ou faux : le random forest n'utilise pas le bootstrap
    (c) Vrai ou faux : le random forest n'est jamais équivalent au bagging
    (d) Vrai ou faux : le random forest utilise un sous-ensemble des prédicteurs séléctionné aléatoirement
    (e) Vrai ou faux : le random forest fait "pousser" des arbres de manière séquentielle
15. Questions sur le boosting :
    (a) Quelle est la différence entre le boosting et le random forest ?
    (b) Vrai ou faux : le boosting n'utilise pas le bootstrap
    (c) Vrai ou faux : le boosting fait "pousser" des arbres de manière séquentielle
    (d) Vrai ou faux : le boosting utilise un sous-ensemble des prédicteurs séléctionné aléatoirement
    (e) Vrai ou faux : la performance du boosting est proportionnelle à la profondeur des arbres utilisés
    (f) Dans le cadre du boosting, peut-on dire que chaque arbre essaie de corriger les faiblesses du précédent ?

4.2 Hyperparameter Tuning

16. Que désigne l'expression *Hyerparameter Tuning* ?

4.2.1 Random Forest

17. Lancez le code suivant :

```
n_arbre = [1,2,3,4,5]
train_results = []
test_results = []
for i in n_arbre:
    rf =    RF_Model = RandomForestClassifier(bootstrap=True,
                                n_estimators=i,
                                random_state=100)
    rf.fit(X_train, y_train)
    train_pred = rf.predict(X_train)
    test_r = rf.score(X_test,y_test)
    test_results.append(test_r)
    train_r = rf.score(X_train,y_train)
    train_results.append(train_r)
from matplotlib.legend_handler import HandlerLine2D
line1, = plt.plot(n_arbre, train_results, label='Erreur␣d␣apprentissage')
line2, = plt.plot(n_arbre, test_results, label='Erreur␣de␣test')
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('Taux␣de␣Classification␣Correct')
plt.xlabel('Nombre␣d␣arbres')
plt.show()
```

(a) Quel hyperparamètre est étudié ici ?

(b) Comment varie la performance du modèle en fonction de cet hyperparamètre ?

(c) Si vous deviez choisir une valeur pour cet hyperparamètre afin d'obtenir la meilleure performance, quelle valeur choisiriez-vous ?

18. Lancez le code suivant :

```
max_depths = [1, 2, 3, 4,5]
train_results = []
test_results = []
for i in max_depths:
    rf =    RF_Model = RandomForestClassifier(bootstrap=True, max_depth=i, random_state=100)
    rf.fit(X_train, y_train)
    train_pred = rf.predict(X_train)
    test_r = rf.score(X_test,y_test)
    test_results.append(test_r)
    train_r = rf.score(X_train,y_train)
    train_results.append(train_r)
from matplotlib.legend_handler import HandlerLine2D
line1, = plt.plot(max_depths, train_results, label='Erreur␣d␣apprentissage')
line2, = plt.plot(max_depths, test_results, label='Erreur␣de␣test')
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('Taux␣de␣Classification␣Correct')
plt.xlabel('Profondeur␣d␣un␣arbre')
plt.show()
```

(a) Quel hyperparamètre est étudié ici ?

(b) Comment varie la performance du modèle en fonction de cet hyperparamètre ?

(c) Si vous deviez choisir une valeur pour cet hyperparamètre afin d'obtenir la meilleure performance, quelle valeur choisiriez-vous ?

4.2.2 Boosting

19. Lancez le code suivant :

```
#############################################################################
from sklearn.ensemble import GradientBoostingClassifier
#############################################################################
n_estimators_n = [1, 2, 5, 10, 20 ]
train_results = []
test_results = []
for i in n_estimators_n:
    rf =  GradientBoostingClassifier(n_estimators=i, learning_rate=0.1,
      max_depth=1, random_state=0).fit(X_train, y_train)
    rf.fit(X_train, y_train)
    train_pred = rf.predict(X_train)
    test_r = rf.score(X_test,y_test)
    test_results.append(test_r)
    train_r = rf.score(X_train,y_train)
    train_results.append(train_r)
from matplotlib.legend_handler import HandlerLine2D
```

```
line1 , = plt.plot(max_depths, train_results, label='Erreur d apprentissage')
line2 , = plt.plot(max_depths, test_results, label='Erreur de test')
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('Taux de Classification Correct')
plt.xlabel('Nombre d arbres')
plt.show()
```

(a) Quel hyperparamètre est étudié ici ?

(b) Comment varie la performance du modèle en fonction de cet hyperparamètre ?

(c) Si vous deviez choisir une valeur pour cet hyperparamètre afin d'obtenir la meilleure performance, quelle valeur choisiriez-vous ?

20. Lancez le code suivant :

```
max_depths = [1, 2,3,4]
train_results = []
test_results = []
for i in max_depths:
    rf =  GradientBoostingClassifier(n_estimators=20, learning_rate=0.1,
      max_depth=i, random_state=0).fit(X_train, y_train)
    rf.fit(X_train, y_train)
    train_pred = rf.predict(X_train)
    test_r = rf.score(X_test,y_test)
    test_results.append(test_r)
    train_r = rf.score(X_train,y_train)
    train_results.append(train_r)
from matplotlib.legend_handler import HandlerLine2D
line1 , = plt.plot(max_depths, train_results, label='Erreur d apprentissage')
line2 , = plt.plot(max_depths, test_results, label='Erreur de test')
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel('Taux de Classification Correct')
plt.xlabel('Profondeur d un arbre')
fig1 = plt.gcf()
fig1.savefig('/Users/maitre/Dropbox/Assas/cours/machine_learning/slides/lecture10/erreur_classification_profondeur_arbr
plt.show()
```

(a) Quel hyperparamètre est étudié ici ?

(b) Comment varie la performance du modèle en fonction de cet hyperparamètre ?

(c) Si vous deviez choisir une valeur pour cet hyperparamètre afin d'obtenir la meilleure performance, quelle valeur choisiriez-vous ?