

Cahier thématique

Évaluation de performance  
Fonctions pour le scoring

**Manipulation 1**: Fonctions pour la classification  
**Manipulation 2**: Fonctions pour la régression

**Préparé par :** Hafed Benteftifa  
  
© Hafed Benteftifa et Nesrine Zemirli 2015-2016

Ce document ne peut être utilisé dans le cadre d’une formation, publication papier, site internet ou tout support sans mon accord express.

Aucune reproduction, même partielle, ne peut être faite de ce document et de l'ensemble de son contenu : textes, images, etc. sans mon autorisation express. Pour toutes informations, communiquer avec moi sur [info@degenio.com](mailto:info@degenio.com).

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Changement** |
| 20 Mai 2016 | 1.0 | Version initiale |
|  |  |  |
|  |  |  |

**Approche**

Afin d’évaluer la performance d’un système d’apprentissage et de savoir si l’on est proche du résultat escompté, on développe une fonction qui donne le score indiquant la qualité du résultat.

On doit considérer les fonctions pour les cas de classification binaire, classification multi-label, régression, clustering ou tout problème en relation avec l’apprentissage.

**Manipulation 1**: Fonctions pour la classification

**Objectif**

Évaluer les fonctions de scoring dans le cas de la classification

**Préliminaire**

* Anaconda ou pycharm est disponible.

**Démarche**

# Partie 1: Classification multilabel

1. L'objectif ici est la prédiction de plus d’un label comme dans le cas de savoir la nature d’une chaussure parmi plusieurs chaussures, un rôle ou catégorie parmi plusieurs catégories. On utilisera pour la demonstration le Dataset iris.

from sklearn import datasets

iris = datasets.load\_iris()

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris.data, iris.target, test\_size=0.50, random\_state=4)

# Utilisation de multiclass classifier (prob) /Decision Tree

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(max\_depth=2)

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

#classes prédites, utilisation de echantillon test

y\_pred = classifier.predict(X\_test)

print iris.target\_names

**Mesure 1 : Matrice de confusion**

C’est une table qui donne une représentation sur les mauvaises classifications pour chaque classe.

Les éléments qui ne sont pas sur la diagonale devraient être idéalement à 0

#Obtention de la matrice de confusion

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print cm

En sortie, on aura:

**[[30 0 0]**

**[ 0 19 3]**

**[ 0 2 21]]**

**Mesure 2 : Accuracy**

Elle donne le nombre de labels prédits qui sont exactement les mêmes que les vrais labels. Cela donne le pourcentage des labels classifiés correctement.

print "Accuracy:", metrics.accuracy\_score(y\_test, y\_pred)

Accuracy: 0.933333333333

**Mesure 3 : Précision**

Elle donne le nombre de résultats corrects dans l’ensemble des résultats. Par exemple, si l’on faisait une classification, elle donne la mesure des labels corrects dans chacun des ensembles de labels

print "Precision:", metrics.precision\_score(y\_test, y\_pred)

Precision: 0.933333333333

**Mesure 4 : Recall**

Elle donne la mesure des résultats appropriés ou corrects dans l’ensemble comparativement à l’ensemble des labels du dataset. En termes de classification, c’est le rapport entre des labels correctement classifiés et le nombre total des labels du dataset

print "Recall:", metrics.recall\_score(y\_test, y\_pred)

Recall: 0.933333333333

**Mesure 5 : F1 score**

Elle donne une mesure moyenne (harmonique) de la précision et du Recall

print "F1 score:", metrics.f1\_score(y\_test, y\_pred)

F1 score: 0.933267359393

Note : l'intervalle de valeurs pour Accuracy, Precision, Recall, et F1 est dans [0,1]

**Manipulation 2**: Fonctions pour la régression

**Objectif**

Évaluer les fonctions de scoring dans le cas de la régression.

**Préliminaire**

* Anaconda ou pycharm est disponible.

**Approche**

Les mesures possibles sont les suivantes:

**Mean absolute error ou MAE** : C’est la moyenne L1 norme du vecteur différence entre les valeurs prédites et les vraies valeurs

from sklearn.metrics import mean\_absolute\_error mean\_absolute\_error([1.0, 0.0, 0.0], [0.0, 0.0, -1.0])

Out: 0.66666666666666663

**Mean squared error ou MSE :** C’est la moyenne L2 norme du vecteur différence entre les valeurs prédites et les vraies valeurs

from sklearn.metrics import mean\_squared\_error

mean\_squared\_error([-10.0, 0.0, 0.0], [0.0, 0.0, 0.0])

Out: 33.333333333333

**Score R2** : coefficient de détermination. Elle prend des valeurs entre 0 et 1 avec les valeurs vers 1 indiquant un modèle parfait.

From sklearn.metrics import r2\_score

r2\_score([3.0, 0.0, 2.0], [3.1, 0.3, 1.9])