Zoidberg 2.0 : documentation

Définition apprentissage automatique

L'apprentissage automatique, également appelé "machine learning" en anglais, est un domaine de l'intelligence artificielle qui cherche à donner aux ordinateurs la capacité d'apprendre à partir de données en utilisant des modèles mathématiques. Il s'agit essentiellement d'un processus permettant d'extraire des informations pertinentes à partir d'un ensemble de données d'entraînement afin d'obtenir les paramètres optimaux pour un modèle donné.

L'objectif est de permettre au modèle d'atteindre les meilleures performances possibles lors de l'exécution de la tâche qui lui est assignée. Une fois l'apprentissage terminé, le modèle peut être déployé en production.[[1]](#footnote-1)

Train-validation-test procedure

Cette procédure se divise en 3 étapes : ensemble de données d’entrainement, ensemble de données de validation et ensemble de données de test.[[2]](#footnote-2)

Une image contenant table

Description générée automatiquement

1. Ensemble de données d'entraînement

L'ensemble d'entraînement est utilisé pour entraîner le modèle. Le modèle est ajusté aux données d'entraînement à l'aide de divers algorithmes d'apprentissage automatique, et l'objectif est de minimiser la différence entre la sortie prédite et la sortie réelle.

1. Ensemble de données de validation

L'ensemble de validation est utilisé pour évaluer les performances du modèle pendant l'entraînement. Il est utilisé pour sélectionner le meilleur modèle parmi un ensemble de modèles candidats, par exemple, des modèles avec différents hyperparamètres. Les performances du modèle sont évaluées sur l'ensemble de validation, et les hyperparamètres sont ajustés en fonction des performances jusqu'à ce que le meilleur modèle possible soit trouvé.

1. Ensemble de données de test

L'ensemble de test est utilisé pour évaluer les performances finales du modèle après qu'il a été entièrement entraîné et réglé à l'aide des ensembles d'entraînement et de validation. L'ensemble de test doit être complètement indépendant des ensembles d'entraînement et de validation et ne doit pas être utilisé pour entraîner ou régler le modèle. L'ensemble de test fournit une estimation impartiale des performances du modèle sur de nouvelles données.

Il est important de noter que la taille de l'ensemble de données et la proportion de données utilisées pour l'entraînement, la validation et les tests peuvent varier en fonction de l'application et des données disponibles. Une pratique courante consiste à utiliser une répartition 70-15-15, où 70% des données sont utilisées pour l'entraînement, 15% pour la validation et 15% pour les tests. D’autres pratiques consiste a fait 60-20-20 ou 80-10-10.

Validation procedure[[3]](#footnote-3)

La validation croisée (cross-validation en anglais) est une technique d'évaluation de la performance d'un modèle d'apprentissage automatique. Cette technique est utilisée pour estimer à quel point un modèle est capable de généraliser à de nouveaux ensembles de données.

La procédure de validation croisée se déroule en plusieurs étapes :[[4]](#footnote-4)

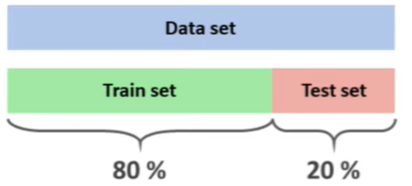
* Diviser les données en ensembles d'apprentissage et de test : Tout d'abord, les données sont divisées en deux ensembles distincts : l'ensemble d'apprentissage (pour entraîner le modèle) et l'ensemble de test (pour évaluer la performance du modèle).
* Diviser l'ensemble d'apprentissage en k : Ensuite, l'ensemble d'apprentissage est divisé en k de taille égale (par exemple, k=5 ou k=10).
* Entraîner le modèle sur k-1: Pour chaque itération de la validation croisée, le modèle est entraîné sur k-1 et évalué sur le pli restant.
* Évaluer la performance du modèle : Après avoir entraîné le modèle sur k-1, il est évalué sur le pli restant. Cette étape est répétée k fois, en utilisant chaque pli comme ensemble de test une fois.
* Calculer la performance moyenne : Enfin, la performance du modèle est calculée en moyennant les performances obtenues pour chaque pli de test.

Il existe différents types de techniques de validation croisée qui varient dans la façon dont l'ensemble de données est divisé et le nombre de plis utilisés. Certaines des techniques de validation croisée couramment utilisées comprennent :

* Validation croisée k-fold : L'ensemble de données est divisé en k plis de taille égale, et le modèle est entraîné et évalué k fois, chaque pli étant utilisé une fois comme ensemble de validation.
* Validation croisée k-fold stratifiée : Cela est similaire à la validation croisée k-fold, mais garantit que chaque pli a à peu près la même proportion d'échantillons de chaque classe pour éviter les déséquilibres de classe.
* Validation croisée leave-one-out (LOOCV) : Chaque pli consiste en un seul échantillon, et le modèle est entraîné sur tous les échantillons sauf un, puis évalué sur l'échantillon laissé de côté.
* Validation croisée pour séries temporelles : il est utilisé pour les données de séries temporelles où l'ordre des échantillons est important, et implique de diviser l'ensemble de données en périodes de temps consécutives et d'utiliser les périodes antérieures pour l'entraînement et les périodes ultérieures pour la validation.

Train test split

Le test de train fractionné consiste à séparer la data :



Cela permet d'avoir un échantillon "TestSet" qui est aléatoirement choisi et qui n’est pas connue du train Set, le testSet est utilisé pour tester la fiabilité réelle de notre IA

**Voir le code en dessous**

Le train set, s'entraine qu'avec les mêmes données et par conséquent la fiabilité se voit diminué si on lui passe une data un peu différentes, c'est ce que le testSet permet de constater, grâce à cette nouvelle data apporter à notre IA, on peut voir comment elle gère les petites différences de data.

**Sources :**

* [Video explicative TestTrain](https://www.youtube.com/watch?v=w_bLGK4Pteo&ab_channel=MachineLearnia)
* [Explication concrete sur le TestTrain](https://learn.microsoft.com/fr-fr/dotnet/machine-learning/how-to-guides/train-machine-learning-model-ml-net)

**Code**

Import et visualisation basique de machineLearning

In [16]:

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** sklearn.datasets **import** load\_iris

In [17]:

iris **=** load\_iris()

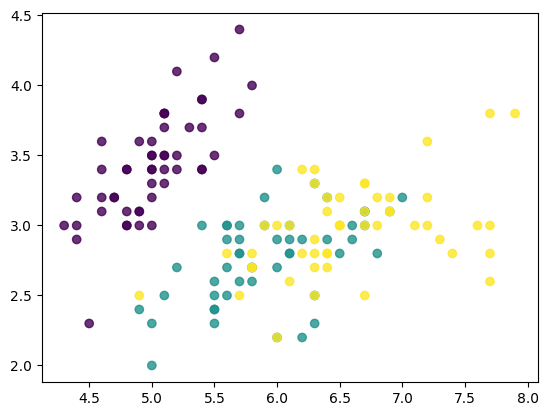
X **=** iris**.**data

y **=** iris**.**target

plt**.**scatter(X[:, 0], X[:, 1], c**=**y, alpha**=**0.8)

Out[17]:

<matplotlib.collections.PathCollection at 0x244656c82b0>



Utilisation du trainTest

In [18]:

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

In [19]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**5)

print('Train set:', X\_train**.**shape)

print('Test set:', X\_test**.**shape)

Train set: (120, 4)

Test set: (30, 4)

In [20]:

plt**.**figure(figsize**=**(12, 4))

plt**.**subplot(121)

plt**.**scatter(X\_train[:, 0], X\_train[:, 1], c**=**y\_train, alpha**=**0.8)

plt**.**title('Train set')

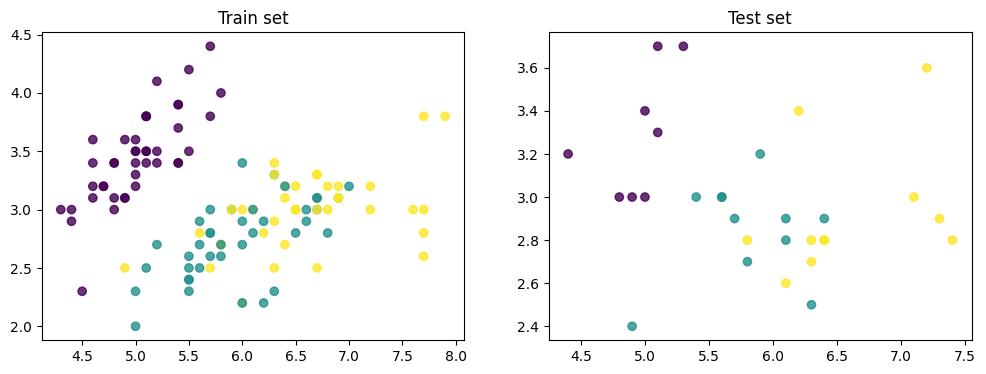
plt**.**subplot(122)

plt**.**scatter(X\_test[:, 0], X\_test[:, 1],c**=**y\_test, alpha**=**0.8)

plt**.**title('Test set')

Out[20]:

Text(0.5, 1.0, 'Test set')



In [21]:

**from** sklearn.neighbors **import** KNeighborsClassifier

In [22]:

model **=** KNeighborsClassifier(n\_neighbors**=**1)

model**.**fit(X\_train, y\_train)

print('train score:', model**.**score(X\_train, y\_train))

print('test score:', model**.**score(X\_test, y\_test))

train score: 1.0

test score: 0.9

Ici on peut voir que notre IA nous dit que la fiabilité était de 100%, alors qu'en réalité elle est de 90%

1. Source : https://www.cnil.fr/fr/definition/apprentissage-automatique [↑](#footnote-ref-1)
2. Source : https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 [↑](#footnote-ref-2)
3. Source : https://machinelearningmastery.com/k-fold-cross-validation/ [↑](#footnote-ref-3)
4. Source : https://datascientest.com/cross-validation [↑](#footnote-ref-4)