

TD 3

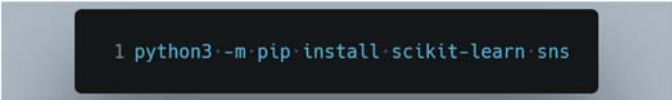
Machine Learning Pipeline

Les prérequis de TD sont les notions vues en CM et lors des précédents TD. En particulier, au TD1 et TD2, vous avez eu l'opportunité de vous (re)familiariser avec Python, et au second vous avez (re)découvert comment manipuler un jeu de données.

Bien que les premiers TD étaient davantage orientés Python qu'intelligence artificielle il est important de souligner que cet enseignement n'est pas un enseignement Python. Il vous appartient par conséquent de vous mettre à niveau sur les bases du langage si cela est nécessaire. Entre autres choses, vous devez être autonomes sur les questions d'installation de packages, de création et de lancement de fichiers *.py, de création d'algorithmes simples en Python comprenant les structures de contrôle de base (conditions, boucles, déclarations de variables, etc.).

Lors de ce TD, vous découvrirez **sklearn** et mettrez au point vos premiers classifieurs. Vous apprendrez à concevoir un *machine learning pipeline*, c'est à dire un processus de bout en bout d'apprentissage, de la récupération des données jusqu'à l'interprétation des résultats.

Thématique 1 (Installation de sklearn)



```
1 python3 -m pip install scikit-learn sns
```

1. Vérifiez que le package **sklearn** est installé sur votre machine
2. S'il n'est pas présent, installez-le. Son nom complet est **scikit-learn**, c'est ce dernier nom que vous devez utiliser (avec **pip***) pour l'installation, tandis que le nom **sklearn**, lui, est utilisé pour importer le package dans un projet.
3. Installez également le package **sns**

**Selon votre machine, vous pourriez avoir à utiliser python et non python3*

Thématique 2 (Import et aperçu des données)

1. Importez le jeu de données **iris** (à partir du fichier *iris.csv* du TD2) et vérifiez s'il contient des données manquantes. Si c'est le cas, vous appliquerez une méthode d'imputation par la moyenne. Les *features* doivent se trouver dans un tableau **numpy** X, et les *labels* dans une liste y.
2. Calculez la corrélation entre chaque paire de variable, et relevez vos réponses dans un fichier texte. Attention : vous devez chercher la fonction **numpy** qui permet de calculer une matrice de corrélation, et non coder "à la main" le calcul de la corrélation.
3. Sur la base des résultats obtenus, à quoi peut-on s'attendre pour la suite ?
4. Affichez les nuages de points suivants :
 - Longueur des pétales en fonction de leur largeur
 - Longueur des sépales en fonction de leur largeur
 - Largeur des pétales en fonction de la longueur des sépales
 - Longueur des pétales en fonction de la largeur des sépales

Proposez une interprétation de vos résultats.

Thématique 3 (Préparation des données)

Pour cette thématique, on part toujours des variables X (donnée) et y (labels) qui ne doivent jamais être altérées au cours du processus.

1. Normalisez les données vers un tableau **numpy** X_normalized
2. Standardisez les données vers un tableau **numpy** X_standardised
3. Appliquez un encodage catégoriel sur les labels (voir sklearn.preprocessing.LabelEncoder)

Thématique 4 (Compréhension de la documentation)

L'une des compétences fondamentales à acquérir, c'est de savoir utiliser la documentation de **sklearn**. Il n'est pas envisageable de tout retenir par cœur, ni de tout aborder. En revanche, il est bien plus accessible de retenir la méthodologie, le formalisme et les écueils à éviter. La documentation est justement là pour cela. La liste des classifieurs disponibles sur **sklearn** est disponible ici :

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning.

Cette liste vous indique la famille du classifieur (linéaire, quadratique, etc.). Chaque entrée vous donnera davantage d'indications sur le classifieur sélectionné. Pour la classification, on utilisera :

Les k-NN (from sklearn.neighbors import KNeighborsClassifier)

- <https://scikit-learn.org/stable/modules/neighbors.html>
- <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier>

Un SVM (from sklearn.svm import SVC)

- <https://scikit-learn.org/stable/modules/svm.html#classification>
- <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>

Et des arbres de décision (from sklearn.tree import DecisionTreeClassifier)

- <https://scikit-learn.org/stable/modules/tree.html#classification>
- <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

Cherchez et testez, en particulier :

1. Comment faire pour :
 - Entraîner un modèle sur des données (fonction fit)
 - Tester un modèle sur des données (fonctions predict et score)
2. Comment découper et stratifier un jeu de données en sous-ensembles pour avoir un jeu d'apprentissage et un jeu de validation
3. Evaluer les performances d'un modèle (module sklearn.metrics)

```
1 from sklearn.neighbors import KNeighborsClassifier~
2 from sklearn.metrics import mean_squared_error~
3 ~
4 # Import and preprocess data~
5 # ...~
6 ~
7 knn_clf = KNeighborsClassifier(n_neighbors=3)~
8 knn_clf.fit(X_train, y_train)~
9 ~
10 y_pred = knn_clf.predict(X_test)~
11 mae = knn_clf.score(X_test, y_test)~
12 mse = mean_squared_error(y_test_, y_pred, squared=False)~
13 ~
14 print(score)
```

Thématique 5 (Classification)

1. Découpez le jeu de données **iris** (brut) afin d'avoir environ 70% des exemples pour l'apprentissage et 30% pour la validation. Les sous-ensembles doivent être stratifiés et les exemples mélangés. Cette technique de validation/découpe s'appelle *hold-out*.
2. Entraînez le k-NN (en faisant varier k)
3. Évaluez ses performances en apprentissage et en validation.
4. Calculez également l'erreur quadratique (MSE)
5. Pour quelle valeur de k obtenez-vous les meilleures performances ?

```
1 from sklearn.model_selection import train_test_split~  
2 # ...~  
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=18)
```

Thématique 6 (Comparaison de modèles)

Dans la thématique précédente, vous avez obtenu des métriques qui vous permettent d'évaluer le classifieur. Dans cette thématique, vous devrez calculer ces mêmes métriques pour plusieurs classifieurs et plusieurs altérations des données.

- Classifieurs à essayer : les SVM et les DecisionTree
- Altérations des données à essayer : iris normalisé et iris standardisé

Soit en tout 4 essais à réaliser. Quelle combinaison s'en sort le mieux ?

Thématique 7 (Choix du meilleur classifieur)

Avant de retenir un classifieur, il est souhaitable de l'entraîner et de l'évaluer sur l'intégralité des données, et non sur (seulement) 70% ou 30%.

Pour ce faire, on applique une méthode appelée validation croisée, ou *cross-validation* (de son nom complet, *k-fold cross-validation*). La validation croisée consiste à diviser le jeu de données complet en *k folds* (en *k* échantillons), puis à entraîner un modèle sur *k-1* échantillons et à le valider sur l'échantillon restant. On répète l'opération *k* fois, jusqu'à ce que tous les *folds* soient passés à la fois en apprentissage et en validation. Au terme de l'opération, on dispose de métriques d'évaluation pour chacun des *k* modèles constitués, et on considère que les performances moyennes d'un modèle entraîné sur l'intégralité du jeu de données (ce qu'il n'est pas possible d'évaluer autrement que par l'estimation) est égale à la moyenne des performances sur les *k folds*.

Sur **sklearn**, le module `sklearn.model_selection` dispose d'une fonction `cross_validation` dédiée à cet effet.

```
1 from sklearn.model_selection import cross_val_score~  
2 # ...~  
3 svm_clf = svm.SVC(kernel='linear', C=1, random_state=42)~  
4 scores = cross_val_score(clf, X, y, cv=5)~  
5 scores
```

Lancez une validation croisée pour chacun des trois classifieurs avec chacune des altérations des données (soit neuf essais en tout). Pour le classifieur k-NN, on retiendra (pour simplifier), le meilleur *k* obtenu en hold-out.

Quelle combinaison (classifieur, altération de données) s'en sort le mieux, finalement ?