TP : Classification en Machine Learning

Algorithmes de classification utilisés

Avant de commencer le TP, nous allons présenter les algorithmes qui seront utilisés. Chaque méthode a ses forces, ses faiblesses et ses paramètres clés à optimiser. Des schémas et images sont fournis (liens ou générés automatiquement) pour aider à la compréhension.

1) Decision Tree (Arbre de décision)

Principe : un arbre de décision segmente l'espace des variables par une série de questions du type « la valeur de feature X est-elle > à un seuil ? ». Chaque nœud correspond à une décision, et les feuilles correspondent aux classes prédites.

- Avantages : interprétable, simple à visualiser, peu de preprocessing nécessaire.
- Inconvénients : tendance à sur-apprendre (overfitting) si l'arbre est trop profond.

Hyperparamètres importants :
- `max_depth` : limite la profondeur de l'arbre.
- `min_samples_split` : nb minimal d'échantillons pour diviser un nœud.
- `min_samples_leaf` : nb minimal d'échantillons dans une feuille.
- `criterion` : mesure d'impureté (entropy, gini).

https://www.youtube.com/watch?v=ZVR2Way4nwQ

2) Random Forest

Principe : une forêt aléatoire est composée de nombreux arbres de décision. Chaque arbre est construit sur un sous-échantillon des données et un sous-ensemble de variables. La prédiction finale est obtenue par vote majoritaire.

- Avantages : robuste, réduit l'overfitting par rapport à un seul arbre, performant sans tuning complexe.
- Inconvénients : moins interprétable qu'un arbre unique, plus coûteux en calcul.

Hyperparamètres importants :
- `n_estimators` : nombre d'arbres.
- `max_depth` : profondeur max des arbres.
- `max_features` : nb de variables prises en compte à chaque split.
- `min_samples_leaf` : nb minimal d'échantillons dans une feuille.

https://www.youtube.com/watch?v=v6VJ2RO66Ag

3) XGBoost (eXtreme Gradient Boosting)

Principe : le boosting construit les arbres séquentiellement. Chaque nouvel arbre corrige les erreurs des arbres précédents. XGBoost est une implémentation optimisée et très utilisée du gradient boosting.

- Avantages : souvent l'algorithme le plus performant sur données tabulaires.
- Inconvénients : tuning plus complexe, temps d'entraînement parfois élevé.

Hyperparamètres importants :
- `n_estimators` : nb d'arbres.
- `learning_rate` : taux d'apprentissage (impact de chaque arbre).
- `max_depth` : profondeur max.
- `subsample` : fraction d'échantillons utilisés par arbre.
- `colsample_bytree` : fraction de variables utilisées par arbre.

Principe du train_test_split

Lorsqu'on entraîne un modèle de machine learning, il est essentiel de pouvoir évaluer ses performances sur des données jamais vues. On sépare donc le dataset en deux parties :
- **train set** : utilisé pour l'apprentissage du modèle.
- **test set** : utilisé uniquement pour l'évaluation finale.

Exemple typique : 80% pour l'entraînement et 20% pour le test, en respectant la proportion des classes (stratification).

https://www.youtube.com/watch?v=SjOfbbfI2qY

Métriques d'évaluation en classification

Pour juger la qualité d'un modèle de classification, plusieurs métriques sont utilisées :

- **Accuracy** : proportion de prédictions correctes.
- **Precision** : parmi les prédictions positives, combien sont réellement positives ?
- **Recall (sensibilité)** : parmi les vrais positifs, combien sont retrouvés par le modèle ?
- **F1-score** : moyenne harmonique de la précision et du rappel, utile quand les classes sont déséquilibrées.

https://www.youtube.com/watch?v=Kdsp6soqA7o

Objectifs pédagogiques

- Comprendre et mettre en pratique plusieurs algorithmes de classification : Decision Tree, Random Forest, XGBoost.

- Savoir préparer un dataset, entraîner, évaluer et comparer des modèles.
- Analyser et interpréter des résultats (métriques, matrices de confusion, importances de variables).

Prérequis

Python, pandas, numpy, matplotlib, notions de scikit-learn.

Dataset utilisé

Le TP utilisera le dataset Breast Cancer Wisconsin, intégré à scikit-learn. Il s'agit de prédire si une tumeur est bénigne ou maligne à partir de mesures de cellules.

Déroulé du TP

1. Présentation des algorithmes.

2. Chargement et exploration du dataset.

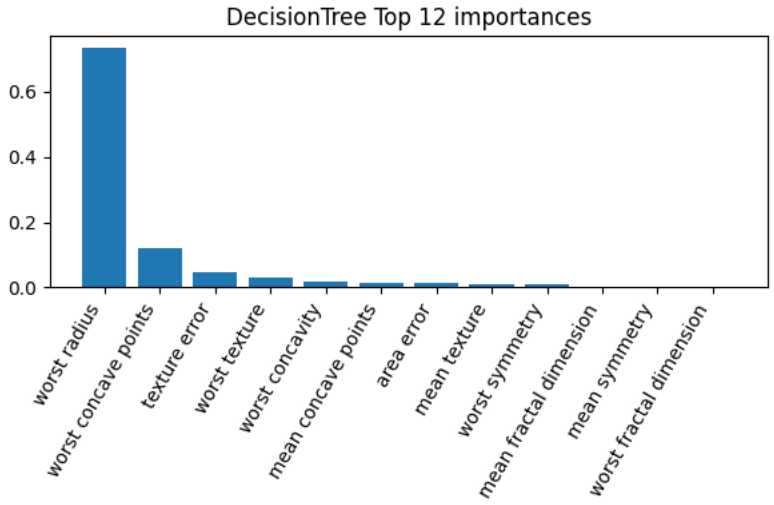
3. Séparation des données en train/test et preprocessing.

4. Entraînement des modèles.

5. Évaluation via métriques globales (accuracy, precision, recall, f1).

6. Visualisation (matrices de confusion).

7. Analyse des importances de variables.



8. Comparaison et discussion des résultats.

Livrables

- Un notebook Jupyter complété (code + réponses textuelles aux questions).
- Un court compte rendu (5–10 lignes) comparant les modèles et concluant sur celui qui semble le plus adapté.