COMPTE RENDU 2 : CLASSIFICATION

I ETUDE DE CAS :

Le dataset **WeatherAUS** de Kaggle contient des données météorologiques collectées dans différentes stations météorologiques en Australie. Il est souvent utilisé pour des tâches d'analyse exploratoire, de prédiction et de classification, en particulier pour prédire si la pluie tombera le jour suivant (**RainTomorrow**).

Source : Kaggle (WeatherAUS)

Objectif principal : Prédire la variable cible RainTomorrow (binaire : "Yes" ou "No").

Période des données : De 2007 à 2017.

II ÉTAPES POUR IMPLÉMENTER DES ALGORITHMES DE CLASSIFICATION

1. Comprendre le problème et préparer les données

- **Définir l'objectif :** Comprendre la nature du problème, par exemple, ici, prédire si la pluie tombera demain (RainTomorrow), ce qui est une tâche de classification binaire.
- **Analyser le dataset :** Examiner les colonnes et les types de variables pour comprendre les données disponibles (numériques, catégoriques, etc.) et détecter les variables importantes.
- **Vérifier les valeurs manquantes et les traiter :** Utiliser des méthodes comme la suppression ou l'imputation des valeurs manquantes.
- **Analyser les outliers :** Identifier et gérer les valeurs aberrantes qui peuvent affecter l'apprentissage des modèles.
- **Encodage des variables catégorielles :** Convertir les variables catégorielles (comme Location ou RainToday) en variables numériques via l'encodage (par exemple, encodage one-hot ou label encoding).

; et pr

- **Visualisation des relations :** Utiliser des outils comme les **pair plots** ou la **matrice de corrélation** pour identifier les relations entre les variables.
- **Réduction de la dimensionnalité :** Si nécessaire, réduire le nombre de variables (par exemple, avec PCA) ou utiliser la sélection de caractéristiques pour ne garder que les variables les plus pertinentes.
- **Normalisation/Standardisation :** Si vous utilisez des modèles sensibles à l'échelle des données (comme K-NN ou SVM), il est essentiel de standardiser les données (par exemple, avec StandardScaler ou MinMaxScaler).

3. Séparer les données en ensembles d'entraînement et de test

- **Train/Test Split :** Diviser les données en deux ensembles, généralement 70-80 % pour l'entraînement et 20-30 % pour le test. Utilisez la fonction train_test_split de bibliothèques comme **scikit-learn**.
- **Cross-validation :** Pour une meilleure évaluation, utilisez la **validation croisée** (par exemple, avec cross_val_score) pour entraîner le modèle sur différents sous-ensembles des données et éviter le surapprentissage.

4. Choisir un modèle de classification

En fonction des caractéristiques du dataset, vous choisirez l'un des modèles suivants :

- **K-NN (K-Nearest Neighbors) :** Utilisé pour des données ayant des relations non linéaires et où les clusters sont visibles. Convient aux petites tailles de données.
- **Arbres de décision (Decision Trees) :** Utile pour des modèles simples et interprétables. Fonctionne bien avec des données mixtes et des relations complexes.

ensei

décision qui réduit le surapprentissage en combinant plusieurs arbres.

- **SVM (Support Vector Machine) :** Utile pour des données avec une frontière de décision nette. Pratique lorsque les classes sont séparées par une ligne ou un hyperplan.
- **Régression logistique :** Bien adaptée aux problèmes où la relation entre les caractéristiques et la classe cible est approximativement linéaire.

5. Entraîner le modèle

- **Fit le modèle sur les données d'entraînement :** Utilisez la méthode fit() du modèle choisi pour entraîner l'algorithme sur l'ensemble d'entraînement. Assurez-vous d'utiliser les données prétraitées.
- **Optimisation des hyperparamètres :** Certains modèles nécessitent des ajustements d'hyperparamètres (par exemple, le nombre de voisins dans K-NN ou la profondeur des arbres dans les arbres de décision). Utilisez des techniques comme **Grid Search** ou **Random Search** pour trouver les meilleurs paramètres.

6. Évaluer le modèle

- **Prédictions :** Utilisez la méthode predict() pour prédire les résultats sur l'ensemble de test.
- **Métriques d'évaluation :** Évaluez la performance du modèle à l'aide de différentes métriques de classification comme :
  - **Exactitude (Accuracy)**
  - **Précision (Precision)**
  - **Rappel (Recall)**

entre

rappel)

- **Courbe ROC** et **AUC** (pour les problèmes de classification binaire)

- **Matrice de confusion :** Affiche la performance du modèle en comparant les classes réelles et prédites.

7. Améliorer le modèle

- **Équilibrer les classes :** Si les classes sont déséquilibrées (par exemple, beaucoup plus de "Non" que de "Oui" pour RainTomorrow), vous pouvez appliquer des techniques comme l'**oversampling** (SMOTE) ou l'**undersampling** pour améliorer la performance du modèle.

- **Fine-tuning :** Réajuster les hyperparamètres en fonction des résultats des tests et de la validation croisée pour améliorer le modèle.

- **Ensembles de modèles :** Vous pouvez aussi essayer des techniques comme **le stacking** ou **le boosting** (ex : XGBoost) pour combiner plusieurs modèles et améliorer la performance globale.