**Code Source Classification :**

**# =============== IMPORTATION DES BIBLIOTHÈQUES ===============**

**# pandas : pour la manipulation des données**

**# numpy : pour les calculs numériques**

**# sklearn : pour les modèles de machine learning et les métriques**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**# =============== CHARGEMENT ET PRÉPARATION DES DONNÉES ===============**

**# Chargement du dataset depuis GitHub**

**url\_donnees = "https://raw.githubusercontent.com/kagyvro48/tp01-ML-Classification-Heart-disease-dataset/main/heart\_classification.csv"**

**heart\_disease = pd.read\_csv(url\_donnees)**

**# =============== DIVISION DES DONNEES EN ENTRAINEMENT ET TEST ===============**

**# X contient toutes les colonnes sauf 'target'**

**# y contient uniquement la colonne 'target' (0: pas de maladie, 1: maladie)**

**X = heart\_disease.drop('target', axis=1)**

**y = heart\_disease['target']**

**# Division des données en ensembles d'entraînement (80%) et de test (20%)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y,**

**test\_size=0.2, # 20% pour le test**

**random\_state=42, # Pour la reproductibilité**

**stratify=y # Pour garder la même distribution de classes**

**)**

**# =============== ARBRE DE DÉCISION ===============**

**# Configuration des hyperparamètres à tester pour l'arbre de décision**

**params\_arbre = {**

**'max\_depth': [3, 5, 7, 10], # Profondeur maximale de l'arbre**

**'min\_samples\_split': [2, 5, 10], # Nombre minimum d'échantillons pour faire un split**

**'min\_samples\_leaf': [1, 2, 4] # Nombre minimum d'échantillons dans une feuille**

**}**

**# CLASSIFICATION HEART DISEASE DATASET**

**print("\n=== CLASSIFICATION HEART DISEASE DATASET TP01 ===")**

**# Création du modèle d'arbre avec GridSearch**

**print("\n=== Entraînement de l'Arbre de Décision ===")**

**arbre\_grille = GridSearchCV(**

**DecisionTreeClassifier(random\_state=42), # Le modèle de base**

**params\_arbre, # Les paramètres à tester**

**cv=5, # Validation croisée en 5 plis**

**scoring='accuracy', # Métrique à optimiser**

**n\_jobs=-1 # Utiliser tous les processeurs**

**)**

**# Entraînement du modèle**

**arbre\_grille.fit(X\_train, y\_train)**

**# Affichage des meilleurs paramètres trouvés**

**print("\nMeilleurs paramètres pour l'arbre :")**

**print(arbre\_grille.best\_params\_)**

**# Prédictions sur l'ensemble de test**

**pred\_arbre = arbre\_grille.predict(X\_test)**

**# =============== FORÊT ALÉATOIRE ===============**

**# Configuration des hyperparamètres pour la forêt aléatoire**

**params\_foret = {**

**'n\_estimators': [100, 200], # Nombre d'arbres dans la forêt**

**'max\_depth': [3, 5, 7], # Profondeur maximale des arbres**

**'min\_samples\_split': [2, 5], # Échantillons minimums pour split**

**'min\_samples\_leaf': [1, 2] # Échantillons minimums par feuille**

**}**

**# Création du modèle de forêt avec GridSearch**

**print("\n=== Entraînement de la Forêt Aléatoire ===")**

**foret\_grille = GridSearchCV(**

**RandomForestClassifier(random\_state=42), # Le modèle de base**

**params\_foret, # Les paramètres à tester**

**cv=5, # Validation croisée en 5 plis**

**scoring='accuracy', # Métrique à optimiser**

**n\_jobs=-1 # Utiliser tous les processeurs**

**)**

**# Entraînement du modèle**

**foret\_grille.fit(X\_train, y\_train)**

**# Affichage des meilleurs paramètres trouvés**

**print("\nMeilleurs paramètres pour la forêt :")**

**print(foret\_grille.best\_params\_)**

**# Prédictions sur l'ensemble de test**

**pred\_foret = foret\_grille.predict(X\_test)**

**# =============== ÉVALUATION DES MODÈLES ===============**

**# Fonction pour afficher les résultats de manière claire**

**def afficher\_resultats\_classification(y\_vrai, y\_pred, nom\_modele):**

**print(f"\n=== Résultats pour {nom\_modele} ===")**

**print(f"Précision (Accuracy) : {accuracy\_score(y\_vrai, y\_pred):.3f}")**

**print("\nRapport de classification détaillé :")**

**print(classification\_report(y\_vrai, y\_pred))**

**print("\nMatrice de confusion :")**

**print(confusion\_matrix(y\_vrai, y\_pred))**

**# Affichage des résultats pour les deux modèles**

**afficher\_resultats\_classification(y\_test, pred\_arbre, "l'Arbre de Décision")**

**afficher\_resultats\_classification(y\_test, pred\_foret, "la Forêt Aléatoire")**

**# =============== IMPORTANCE DES FEATURES ===============**

**# Récupération et affichage de l'importance des features pour la forêt aléatoire**

**importance\_features = pd.DataFrame({**

**'Feature': X.columns,**

**'Importance': foret\_grille.best\_estimator\_.feature\_importances\_**

**})**

**importance\_features = importance\_features.sort\_values('Importance', ascending=False)**

**print("\n=== Importance des caractéristiques (Forêt Aléatoire) ===")**

**print(importance\_features)**

**Résultat du Code Source Classification en console :**

**=== CLASSIFICATION HEART DISEASE DATASET TP01 ===**

**=== Entraînement de l'Arbre de Décision ===**

**Meilleurs paramètres pour l'arbre :**

**{'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2}**

**=== Entraînement de la Forêt Aléatoire ===**

**Meilleurs paramètres pour la forêt :**

**{'max\_depth': 7, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100}**

**=== Résultats pour l'Arbre de Décision ===**

**Précision (Accuracy) : 0.985**

**Rapport de classification détaillé :**

**precision recall f1-score support**

**0 0.97 1.00 0.99 100**

**1 1.00 0.97 0.99 105**

**accuracy 0.99 205**

**macro avg 0.99 0.99 0.99 205**

**weighted avg 0.99 0.99 0.99 205**

**Matrice de confusion :**

**[[100 0]**

**[ 3 102]]**

**=== Résultats pour la Forêt Aléatoire ===**

**Précision (Accuracy) : 0.990**

**Rapport de classification détaillé :**

**precision recall f1-score support**

**0 1.00 0.98 0.99 100**

**1 0.98 1.00 0.99 105**

**accuracy 0.99 205**

**macro avg 0.99 0.99 0.99 205**

**weighted avg 0.99 0.99 0.99 205**

**Matrice de confusion :**

**[[ 98 2]**

**[ 0 105]]**

**=== Importance des caractéristiques (Forêt Aléatoire) ===**

**Feature Importance**

**2 cp 0.145340**

**11 ca 0.123912**

**7 thalach 0.108256**

**12 thal 0.106809**

**9 oldpeak 0.103420**

**0 age 0.090238**

**8 exang 0.083978**

**4 chol 0.067971**

**3 trestbps 0.064271**

**10 slope 0.052095**

**1 sex 0.027167**

**6 restecg 0.016421**

**5 fbs 0.010121**

**Code Source Régression :**

**# =============== IMPORTATION DES BIBLIOTHÈQUES ===============**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.tree import DecisionTreeRegressor**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.compose import ColumnTransformer**

**from sklearn.pipeline import Pipeline**

**# =============== CHARGEMENT ET PRÉPARATION DES DONNÉES ===============**

**# Chargement du dataset**

**url\_donnees = "https://raw.githubusercontent.com/kagyvro48/tp01-ML-Regression-California-Houses-Prices-dataset/main/housing.csv"**

**california\_housing = pd.read\_csv(url\_donnees)**

**# =============== PRÉTRAITEMENT DES DONNÉES ===============**

**# Séparation des caractéristiques numériques et catégorielles**

**colonnes\_numeriques = california\_housing.select\_dtypes(include=['int64', 'float64']).columns.tolist()**

**colonnes\_numeriques.remove('median\_house\_value') # Retirer la variable cible**

**# Créer des pipelines de prétraitement**

**numeric\_transformer = Pipeline(steps=[**

**('scaler', StandardScaler())**

**])**

**# Encoder la variable catégorielle ocean\_proximity**

**label\_encoder = LabelEncoder()**

**california\_housing['ocean\_proximity\_encoded'] = label\_encoder.fit\_transform(california\_housing['ocean\_proximity'])**

**# Préparer X et y**

**X = california\_housing.drop(['median\_house\_value', 'ocean\_proximity'], axis=1)**

**y = california\_housing['median\_house\_value']**

**# Division des données**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y,**

**test\_size=0.2,**

**random\_state=42**

**)**

**# =============== ARBRE DE DÉCISION ===============**

**# Configuration des hyperparamètres**

**params\_arbre = {**

**'max\_depth': [5, 10, 15],**

**'min\_samples\_split': [2, 5],**

**'min\_samples\_leaf': [1, 2]**

**}**

**# REGRESSION CALIFORNIA HOUSES PRICES DATASET**

**print("\n=== REGRESSION CALIFORNIA HOUSES PRICES DATASET TP01 ===")**

**# Création et entraînement du modèle**

**print("\n=== Entraînement de l'Arbre de Décision ===")**

**arbre\_grille = GridSearchCV(**

**DecisionTreeRegressor(random\_state=42),**

**params\_arbre,**

**cv=5,**

**scoring='neg\_mean\_squared\_error',**

**n\_jobs=-1**

**)**

**arbre\_grille.fit(X\_train, y\_train)**

**print("\nMeilleurs paramètres pour l'arbre :")**

**print(arbre\_grille.best\_params\_)**

**# Prédictions**

**pred\_arbre = arbre\_grille.predict(X\_test)**

**# =============== FORÊT ALÉATOIRE ===============**

**# Configuration des hyperparamètres**

**params\_foret = {**

**'n\_estimators': [100, 200],**

**'max\_depth': [5, 10, 15],**

**'min\_samples\_split': [2, 5],**

**'min\_samples\_leaf': [1, 2]**

**}**

**# Création et entraînement du modèle**

**print("\n=== Entraînement de la Forêt Aléatoire ===")**

**foret\_grille = GridSearchCV(**

**RandomForestRegressor(random\_state=42),**

**params\_foret,**

**cv=5,**

**scoring='neg\_mean\_squared\_error',**

**n\_jobs=-1**

**)**

**foret\_grille.fit(X\_train, y\_train)**

**print("\nMeilleurs paramètres pour la forêt :")**

**print(foret\_grille.best\_params\_)**

**# Prédictions**

**pred\_foret = foret\_grille.predict(X\_test)**

**# =============== ÉVALUATION DES MODÈLES ===============**

**def afficher\_resultats\_regression(y\_vrai, y\_pred, nom\_modele):**

**print(f"\n=== Résultats pour {nom\_modele} ===")**

**mse = mean\_squared\_error(y\_vrai, y\_pred)**

**rmse = np.sqrt(mse)**

**mae = mean\_absolute\_error(y\_vrai, y\_pred)**

**r2 = r2\_score(y\_vrai, y\_pred)**

**print(f"Erreur quadratique moyenne (MSE) : {mse:,.2f}")**

**print(f"Racine de l'erreur quadratique moyenne (RMSE) : {rmse:,.2f}")**

**print(f"Erreur absolue moyenne (MAE) : {mae:,.2f}")**

**print(f"Coefficient de détermination (R²) : {r2:.3f}")**

**# Affichage des résultats pour les deux modèles**

**afficher\_resultats\_regression(y\_test, pred\_arbre, "l'Arbre de Décision")**

**afficher\_resultats\_regression(y\_test, pred\_foret, "la Forêt Aléatoire")**

**# =============== IMPORTANCE DES FEATURES ===============**

**importance\_features = pd.DataFrame({**

**'Feature': X.columns,**

**'Importance': foret\_grille.best\_estimator\_.feature\_importances\_**

**})**

**importance\_features = importance\_features.sort\_values('Importance', ascending=False)**

**print("\n=== Importance des caractéristiques (Forêt Aléatoire) ===")**

**print(importance\_features)**

**Résultat du Code Source Régression en console :**

**=== REGRESSION CALIFORNIA HOUSES PRICES DATASET TP01 ===**

**=== Entraînement de l'Arbre de Décision ===**

**Meilleurs paramètres pour l'arbre :**

**{'max\_depth': 10, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5}**

**=== Entraînement de la Forêt Aléatoire ===**

**Meilleurs paramètres pour la forêt :**

**{'max\_depth': 15, 'min\_samples\_leaf': 2, 'min\_samples\_split': 2, 'n\_estimators': 200}**

**=== Résultats pour l'Arbre de Décision ===**

**Erreur quadratique moyenne (MSE) : 3,696,459,815.69**

**Racine de l'erreur quadratique moyenne (RMSE) : 60,798.52**

**Erreur absolue moyenne (MAE) : 39,736.55**

**Coefficient de détermination (R²) : 0.718**

**=== Résultats pour la Forêt Aléatoire ===**

**Erreur quadratique moyenne (MSE) : 2,516,579,711.06**

**Racine de l'erreur quadratique moyenne (RMSE) : 50,165.52**

**Erreur absolue moyenne (MAE) : 32,328.95**

**Coefficient de détermination (R²) : 0.808**

**=== Importance des caractéristiques (Forêt Aléatoire) ===**

**Feature Importance**

**7 median\_income 0.537840**

**8 ocean\_proximity\_encoded 0.114057**

**1 latitude 0.107024**

**0 longitude 0.104426**

**2 housing\_median\_age 0.052139**

**5 population 0.029443**

**4 total\_bedrooms 0.020492**

**3 total\_rooms 0.020158**

**6 households 0.014421**