TP1 - Fondamentaux de l'apprentissage automatique 8INF867

Ce projet implémente l'algorithme ID3 pour la construction d'arbres de décision personnalisées pour les données catégorielles. Il inclut également des tests unitaires pour vérifier le bon fonctionnement de l'implémentation.

Membres du groupe projet

- **Mathis AULAGNIER** - *AULM12040200*
- **Héléna BARBILLON** - *BARH30530200*
- **Moïse Tosmanya KUMAVI** - *KUMM06100000*

Structure du Workspace

```
.
├── Data # Dossier contenant les données utilisées pour les tests
│   ├── MBA.csv
│   ├── iris.csv
│   └── titanic.csv
├── README.md
├── Travail Pratique.pdf #Sujet du projet
├── Rapport du projet.pdf #Rapport du projet
├── __init__.py
├── ID3.py # Implémentation de l'algorithme ID3
├── DecisionTreeVisulizer.py # Pour afficher l'arbre graphiquement
├── main.py
```

Détails des Fichiers

- **ID3 .py** : Ce fichier contient l'implémentation de l'algorithme ID3 pour la construction d'arbres de décision.
- **DecisionTreeVisualizer .py** : Ce fichier contient un classe permettant d'afficher l'arbre dans un graphique après sa construction.
- **data/** : Ce répertoire contient les jeux de données utilisés pour entraîner et tester l'algorithme ID3.
- **main .py** : Ce fichier contient l'implémentation de chargement des données, l'entraînement, la prédiction, l'élagage et la visualisation de l'arbre.
- **requirements.txt** : Liste des dépendances Python nécessaires pour exécuter le projet.

Librairies nécessaires

Pour exécuter ce projet, les bibliothèques suivantes sont requises :

> *numpy : Manipulation des tableaux et des calculs numériques.*
> *pandas : Gestion des données sous forme de tableaux.*
> *scikit-learn : Utilisation des pipelines pour la discrétisation des données et l'imputation des valeurs manquantes.*
> *matplotlib : Visualisation de l'arbre de décision.*
> *networkx : Construction et visualisation des graphes.*

Installation

Pour installer les dépendances nécessaires, exécutez la commande suivante :

```
pip install -r requirements.txt
```

Alternativement, créez et activez un environnement virtuel pour gérer les dépendances :

Créer un environnement virtuel
`python -m venv env`
Activer l'environnement virtuel (Windows)
`env\Scripts\activate`
Activer l'environnement virtuel (macOS/Linux)
`source env/bin/activate`
Installer les dépendances
`pip install pandas numpy scikit-learn`

Initialisation de la Classe

```
from id3 import ID3
# Exemple d'initialisation
model = ID3(depth_limit=5, nom_colonne_classe='admission', seuil_gini=0.05, seuil_discre
tisation=10, number_bins=10)
```

Paramètres

- `depth_limit` : Profondeur maximale de l'arbre.

- `nom_colonne_classe` : Nom de la colonne cible dans le dataset.
- `seuil_gini` : Seuil d'impureté de Gini pour la division des nœuds.
- `seuil_discretisation` : Seuil pour la discrétisation des données.
- `number_bins` : Nombre de bins (spécifie que les valeurs numériques seront divisées en 10 intervalles égaux pour réduire le nombre de valeurs différentes dans la colonne) pour discrétiser les caractéristiques continues.

Méthodes

`fit(df)`

Construit l'arbre de décision à partir des données d'apprentissage. Paramètres : df : DataFrame Pandas contenant les données d'apprentissage, y compris la colonne cible. Utilisation :

```
model.fit(training_data)
predict(X)
```

Prédit les étiquettes de classe pour de nouvelles instances en utilisant l'arbre construit.

Paramètres :

X : DataFrame Pandas contenant les données de test.

Retourne :

`predictions` : Liste des étiquettes de classe prédites.

`chemin` : Liste des chemins de décision.

Utilisation :

```
predictions, chemin = model.predict(test_data)
```

I. Structure générale du Code :

La fonction principale (main.py) :

Chargement des données : Le fichier `MBA.csv` est chargé et séparé en un ensemble d'entraînement et un ensemble de validation avec `train_test_split`.

Entraînement et Prédiction : L'arbre ID3 est créé et entraîné, puis des prédictions sont effectuées sur les ensembles d'entraînement et de validation.

Élagage : L'élagage est effectué après l'entraînement pour éviter le sur-apprentissage, et l'arbre après élagage est affiché.

Visualisation : L'arbre est visualisé avec DecisionTreeVisualizer.

Classe ID3 (ID3.py):

- Méthodes principales : `fit`, `build_tree`, `predict`, `discretize`, `prune`, `gini`, `information_gain`, `best_split`.

- Discrétisation : Les attributs numériques sont discrétisés avant la construction de l'arbre avec `KBinsDiscretizer`.

- Gestion des valeurs manquantes : Les valeurs manquantes sont imputées par la moyenne ou une valeur constante.

Création des pipelines pour chaque type de donnée (`numeric_pipeline`, `categorical_pipeline`, `boolean_pipeline`)

Convertion des valeur `NaN` de la classe **cible** `y` en valeur manquante ('Etiquette manquante')

```
y = y.fillna('Etiquette manquante')
```

- Élagage : Le modèle applique un élagage postérieur basé sur l'ensemble de validation.

Classe DecisionTreeVisualizer (DecisionTreeVisualizer.py) :

Permet de visualiser l'arbre sous forme de graphique en utilisant `networkx` et `matplotlib`.

Cet affichage est conseillé pour les petits arbres.

II. Explication du projet:

Vue d'ensemble des principales fonctionnalités du projet:

1. Préparation des données

Le jeu de données est d'abord chargé et divisé en un ensemble d'entraînement et un ensemble de validation.

Les attributs numériques sont discrétisés à l'aide de la méthode `KBinsDiscretizer` de scikit-learn, tandis que les valeurs manquantes sont gérées via l'imputation.

2. Construction de l'arbre de décision

L'algorithme ID3 utilise le critère de Gini pour mesurer l'homogénéité des classes à chaque nœud de l'arbre.

L'arbre est construit récursivement en choisissant, à chaque étape, le meilleur attribut à partir du gain d'information.

3. Élagage postérieur

Une fois l'arbre construit, un processus d'élagage est appliqué en utilisant les données de validation.
Cela permet de réduire la taille de l'arbre en supprimant les branches qui ne contribuent pas significativement à la précision de la prédiction.

4. Prédiction

Le modèle construit est ensuite utilisé pour prédire les classes des instances dans les ensembles d'entraînement et de validation.
Le taux de précision est calculé avant et après l'élagage pour évaluer l'efficacité de la réduction de la complexité de l'arbre.

5. Visualisation

Une fonction `print_tree` de la calsse `ID3` permet d'afficher les arbres dans le terminal, en affichant les données lignes par lignes.
La classe `DecisionTreeVisualizer` permet d'afficher dans un graphique les arbres de petite taille.
Le modèle final est visualisé sous la forme d'un graphe grâce à matplotlib et networkx, ce qui permet de voir les différentes décisions prises par l'arbre.

Conclusion

La classe ID3 offre une implémentation personnalisable de l'algorithme d'arbre de décision ID3, capable de gérer les valeurs manquantes et de discrétiser les caractéristiques continues.