

VISUALISATION D'UN MODELE D'ARBRE DE DECISION

Jean marc Dongo KOUADIO

1. Charger des données

Les données utilisées dans le cadre de ce projet, proviennent de kaggle lien de data : <https://www.kaggle.com/datasets/erdemtaha/cancer-data> Il s'agit de la base de données diagnostic du cancer du sein

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Charger les données
data_cancer = pd.read_csv('Cancer_Data.csv')

# Aperçu des données
print(data_cancer.head())
```

la structure des dataframes

	id	diagnosis	...	fractal_dimension_worst	Unnamed: 32
0	842302	M	...	0.11890	NaN
1	842517	M	...	0.08902	NaN
2	84300903	M	...	0.08758	NaN
3	84348301	M	...	0.17300	NaN
4	84358402	M	...	0.07678	NaN

[5 rows x 33 columns]

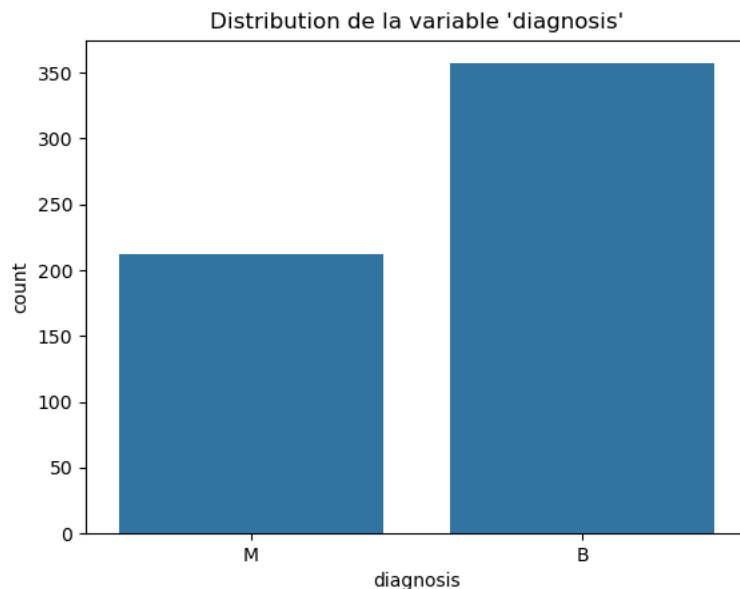
2. Préparation des données

Je supprime les variables 'id' et 'Unnamed: 32' qui ne nous fournissent pas d'informations importantes

```
# Suppression des variables 'id' et 'Unnamed: 32'
data_cancer.drop(columns=['id', 'Unnamed: 32'], inplace=True)
```

```
# Conversion de la variable cible en binaire (M = 1, B = 0)
data_cancer['diagnosis'] = data_cancer['diagnosis'].map({'M': 1, 'B': 0})
#
```

3. Analyse des données



La conversion de la variable cible diagnosis de 'M' (malin) et 'B' (bénin) à des valeurs binaires 0 & 1 est cruciale pour que le modèle puisse apprendre correctement.

```
# Conversion de la variable cible en binaire (M = 1, B = 0)
data_cancer['diagnosis'] = data_cancer['diagnosis'].map({'M': 1, 'B': 0})
```

4. Modélisation

```
# Variable cible
y = data_cancer['diagnosis']

# Variables indépendantes
X = data_cancer.drop('diagnosis', axis=1)

# Séparation des données en données d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=1111)

# Affichage des tailles des jeux de données
print("Shape des données d'entraînement X:", X_train.shape)
print("Shape des données d'entraînement y:", y_train.shape)
print("Shape des données de test X:", X_test.shape)
print("Shape des données de test y:", y_test.shape)

# Création du modèle d'arbre de décision
tree_model = DecisionTreeClassifier(random_state=42)

# Entraînement du modèle
tree_model.fit(X_train, y_train)
```

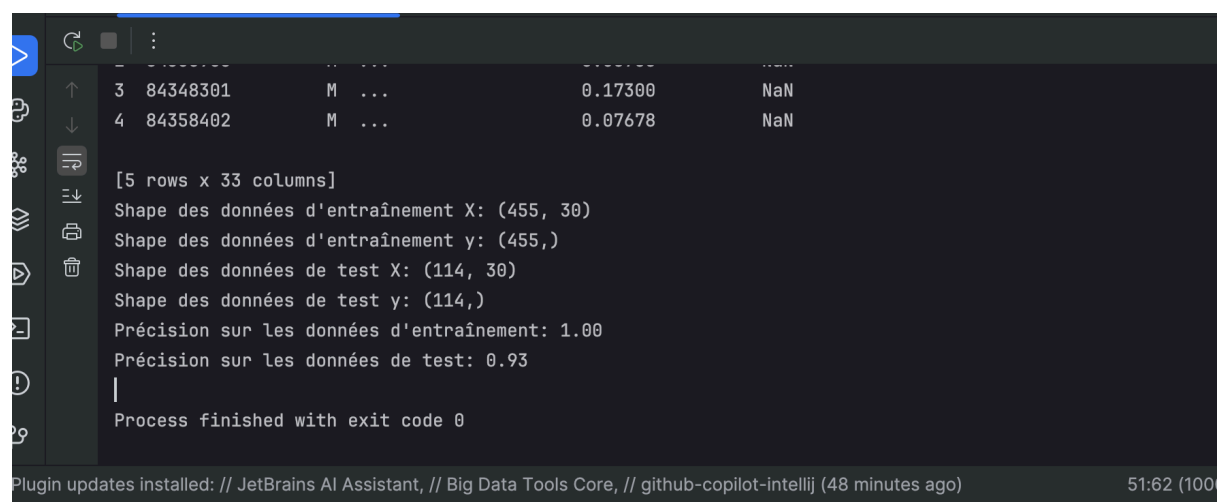
```
# Précision sur les données d'entraînement
train_score = tree_model.score(X_train, y_train)
print(f"Précision sur les données d'entraînement: {train_score:.2f}")

# Précision sur les données de test
test_score = tree_model.score(X_test, y_test)
print(f"Précision sur les données de test: {test_score:.2f}")
```

Données d'entraînement (80%) et données de test (20%)

Création d'un modèle d'arbre de décision

Précision globale sur les données d'entraînement



The screenshot shows a Jupyter Notebook interface with a terminal window. The terminal output displays the following information:

```
[5 rows x 33 columns]
Shape des données d'entraînement X: (455, 30)
Shape des données d'entraînement y: (455,)
Shape des données de test X: (114, 30)
Shape des données de test y: (114,)
Précision sur les données d'entraînement: 1.00
Précision sur les données de test: 0.93
|
Process finished with exit code 0
```

At the top of the terminal, two rows of data are visible:

3	84348301	M	...	0.17300	NaN
4	84358402	M	...	0.07678	NaN

The bottom status bar indicates: "Plugin updates installed: // JetBrains AI Assistant, // Big Data Tools Core, // github-copilot-intellij (48 minutes ago)" and "51:62 (100)".

Visualisation du modèle d'arbre de décision

```
plot_tree(tree_model, feature_names=X.columns, class_names=['B', 'M'],
          filled=True, fontsize=10)
```

