Compte Rendu
Analyse du Dataset Iris avec Random Forest

Chaïmaâ ABDIDA

10 décembre 2025

Table des matières

1 Introduction

1.1 Objectif du projet

Ce projet vise à entraîner et évaluer un modèle de classification basé sur l'algorithme **Random Forest** pour prédire l'espèce des fleurs d'Iris à partir de leurs caractéristiques morphologiques.

1.2 Dataset utilisé

— **Source** : Dataset Iris de l'UCI Machine Learning Repository (via Kaggle)
— **Nombre d'observations** : 150 entrées
— **Nombre de variables** : 6 colonnes

2 Description des données

2.1 Structure du dataset

Le dataset contient les colonnes présentées dans le tableau 1.

TABLE 1 – Structure du dataset Iris

Colonne	Type	Description
Id	int64	Identifiant unique de chaque observation
SepalLengthCm	float64	Longueur du sépale en centimètres
SepalWidthCm	float64	Largeur du sépale en centimètres
PetalLengthCm	float64	Longueur du pétale en centimètres
PetalWidthCm	float64	Largeur du pétale en centimètres
Species	object	Espèce de la fleur d'Iris (variable cible)

2.2 Caractéristiques des données

— **Valeurs manquantes** : 0 (aucune valeur manquante détectée)
— **Lignes dupliquées** : 0 (aucun doublon)
— **Conclusion** : Le dataset est déjà propre et ne nécessite aucun nettoyage

2.3 Code Python - Chargement et nettoyage

```
import kagglehub
import pandas as pd

# Telechargement du dataset depuis Kaggle
path = kagglehub.dataset_download("uciml/iris")
print("Path to dataset files:", path)

# Chargement du dataset
dataset = pd.read_csv(f'{path}/Iris.csv')

```

```python
# Affichage des informations du dataset
print("Dataset Info:")
dataset.info()

# Verification des valeurs manquantes
print("\nMissing Values:")
print(dataset.isnull().sum())

# Verification des lignes dupliquees
print("\nDuplicate Rows:")
duplicate_rows = dataset.duplicated().sum()
print(duplicate_rows)

# Validation de la proprete des donnees
if dataset.isnull().sum().sum() == 0 and duplicate_rows == 0:
    print("\nDataset is already clean.")
```

Listing 1 – Chargement et vérification du dataset

Résultat : Le dataset contient 150 entrées avec 6 colonnes, aucune valeur manquante et aucun doublon.

3 Préparation des données

3.1 Séparation des variables

Variables explicatives (X) :
— SepalLengthCm
— SepalWidthCm
— PetalLengthCm
— PetalWidthCm
Variable cible (y) :
— Species (3 classes : Iris-setosa, Iris-versicolor, Iris-virginica)

3.2 Division du dataset

Le dataset a été divisé en deux ensembles (voir tableau 2).

TABLE 2 – Division du dataset

Ensemble	Taille	Proportion
Entraînement (train)	120 observations	80%
Test	30 observations	20%

Paramètres utilisés :
— `test_size = 0.2` (20% pour le test)
— `random_state = 42` (pour la reproductibilité des résultats)

3.3 Code Python - Préparation

```python
from sklearn.model_selection import train_test_split

# Definition des variables explicatives (X) et de la cible (y)
X = dataset.drop(['Id', 'Species'], axis=1)
y = dataset['Species']

# Division des donnees en ensembles d'entrainement et de test
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

print("Shape of X_train:", X_train.shape)  # (120, 4)
print("Shape of X_test:", X_test.shape)    # (30, 4)
print("Shape of y_train:", y_train.shape)  # (120,)
print("Shape of y_test:", y_test.shape)    # (30,)
```

Listing 2 – Séparation et division des données

4 Algorithme de Machine Learning : Random Forest

4.1 Qu'est-ce que le Random Forest ?

Le **Random Forest** (Forêt Aléatoire) est un algorithme d'apprentissage supervisé basé sur un ensemble d'arbres de décision. Il fonctionne selon les principes suivants :

1. **Construction de multiples arbres** : Création de nombreux arbres de décision indépendants
2. **Échantillonnage aléatoire** : Chaque arbre est entraîné sur un sous-ensemble aléatoire des données
3. **Sélection aléatoire des variables** : À chaque nœud, un sous-ensemble aléatoire de variables est considéré
4. **Vote majoritaire** : Pour la classification, la prédiction finale est obtenue par vote majoritaire de tous les arbres

4.2 Avantages du Random Forest

— Résistance au surapprentissage (overfitting)
— Gestion efficace des données multidimensionnelles
— Robustesse face aux valeurs aberrantes
— Pas besoin de normalisation des données

4.3 Paramètres du modèle

Le modèle utilise les paramètres par défaut de scikit-learn, avec `random_state=42` pour assurer la reproductibilité.

5 Entraînement du modèle

Le modèle Random Forest a été entraîné sur les 120 observations de l'ensemble d'entraînement. L'algorithme a appris les relations entre les caractéristiques morphologiques des fleurs et leur espèce correspondante.

Étapes d'entraînement :

1. Instanciation du modèle RandomForestClassifier
2. Ajustement du modèle aux données d'entraînement (X_{train}, y_{train})
3. Construction de la forêt d'arbres de décision

5.1 Code Python - Entraînement

```python
from sklearn.ensemble import RandomForestClassifier

# Instanciation du modele Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Entrainement du modele sur les donnees d'entrainement
model.fit(X_train, y_train)

print("Random Forest Classifier model trained successfully.")
```

Listing 3 – Entraînement du modèle Random Forest

Explication du code :

— `RandomForestClassifier(random_state=42)` : Crée un modèle avec les paramètres par défaut

— `model.fit(X_train, y_train)` : Entraîne le modèle sur les données d'entraînement

— Le paramètre `random_state=42` assure la reproductibilité des résultats

6 Évaluation du modèle

6.1 Métriques de performance

Le modèle a été évalué sur l'ensemble de test (30 observations) avec les résultats suivants :

6.2 Code Python - Évaluation

```python
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

# Predictions sur l'ensemble de test
y_pred = model.predict(X_test)

# Calcul de la precision globale (Accuracy)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {accuracy:.4f}")
```

```
9
10 # Affichage du rapport de classification detaille
11 print("\nClassification Report:")
12 print(classification_report(y_test, y_pred))
13
14 # Affichage de la matrice de confusion
15 print("\nConfusion Matrix:")
16 print(confusion_matrix(y_test, y_pred))
```

Listing 4 – Évaluation du modèle

6.3 Accuracy (Précision globale)

Accuracy = 1.0000 (100%)
Le modèle a correctement classifié toutes les fleurs d'Iris de l'ensemble de test.

6.4 Rapport de classification détaillé

TABLE 3 – Rapport de classification par espèce

Espèce	**Precision**	**Recall**	**F1-Score**	**Support**
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11
Moyenne	**1.00**	**1.00**	**1.00**	**30**

Interprétation des métriques :
— **Precision** : Proportion de prédictions correctes parmi toutes les prédictions positives pour une classe
— **Recall** : Proportion d'instances correctement identifiées parmi toutes les instances réelles d'une classe
— **F1-Score** : Moyenne harmonique entre precision et recall

6.5 Matrice de confusion

TABLE 4 – Matrice de confusion

	Prédictions		
Réel	Setosa	Versicolor	Virginica
Setosa	10	0	0
Versicolor	0	9	0
Virginica	0	0	11

Analyse : La matrice de confusion montre une classification parfaite avec aucune erreur de classification (tous les éléments sont sur la diagonale).

7 Conclusions

7.1 Résultats obtenus

Le modèle Random Forest a démontré une **performance exceptionnelle** avec :
— Une précision de 100% sur l'ensemble de test
— Aucune erreur de classification
— Des scores parfaits pour toutes les espèces d'Iris

7.2 Points forts du projet

1. **Qualité des données** : Dataset propre sans valeurs manquantes ni doublons
2. **Choix de l'algorithme** : Random Forest adapté pour ce type de classification multi-classes
3. **Performance optimale** : Classification parfaite sur l'ensemble de test

7.3 Recommandations

1. **Validation croisée** : Bien que les résultats soient excellents, il serait recommandé d'effectuer une validation croisée (k-fold cross-validation) pour confirmer la robustesse du modèle
2. **Optimisation des hyperparamètres** : Explorer l'ajustement des paramètres du Random Forest (nombre d'arbres, profondeur maximale, etc.) via GridSearchCV
3. **Analyse de l'importance des variables** : Étudier quelles caractéristiques morphologiques contribuent le plus à la classification
4. **Test sur de nouvelles données** : Valider le modèle sur des données externes non vues pour confirmer sa capacité de généralisation

7.4 Contexte du dataset Iris

Le dataset Iris est un cas d'école classique en machine learning, connu pour être relativement simple à classifier. Les résultats parfaits obtenus sont cohérents avec la littérature scientifique sur ce dataset et confirment l'efficacité du Random Forest pour ce type de problème de classification.