Bootcamp IA — Atelier Guidé : Classification supervisée (Scikit-learn)

Département IA & Ingénierie des Données
Institut Supérieur d'Informatique (ISI Dakar)

Samedi 20 Septembre 2025
14h15 – 16h00

Objectif de l'atelier

Cet atelier vise à pratiquer **trois algorithmes de classification** avec Scikit-learn : **KNN**, **Régression Logistique** et **Arbre de Décision**. Les participants apprendront à :

- Charger et explorer rapidement des jeux de données standards.
- Construire, entraîner et évaluer des modèles de classification.
- Visualiser des matrices de confusion et (pour les tâches binaires) la courbe ROC.

Contexte & Données

Nous utiliserons trois datasets **intégrés à Scikit-learn** :

- **Iris** (3 classes) — pour KNN.
- **Breast Cancer** (binaire) — pour la Régression Logistique.
- **Wine** (3 classes) — pour l'Arbre de Décision.

Mission : entraîner, évaluer, comparer les trois approches et discuter des avantages/limites.

Organisation des participants

- Les 70 participants sont répartis en **14 groupes de 5**.
- Environnement : Python (Anaconda) ou Google Colab.
- Temps imparti : **1h15**.
- Livrables par groupe : code + graphes + mini-interprétation.

Plan de travail

1. **Préparer l'environnement** (imports, fonctions utilitaires).
2. **Partie A (KNN & Iris)** : split, entraînement, évaluation + étude de k.
3. **Partie B (Régression Logistique & Breast Cancer)** : pipeline avec standardisation, ROC/AUC.
4. **Partie C (Arbre de Décision & Wine)** : profondeur, critère (gini/entropy), visualisation.
5. **Synthèse** : tableau récapitulatif & discussion.

Code de départ (commun)

```python
# ================== Imports communs ==================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import (
    accuracy_score, confusion_matrix, ConfusionMatrixDisplay,
    classification_report, roc_curve, auc
)

# Pour des graphiques plus nets
plt.rcParams["figure.dpi"] = 120

def plot_conf_mat(y_true, y_pred, labels=None, title="Matrice de confusion"):
    cm = confusion_matrix(y_true, y_pred, labels=labels)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
    disp.plot(values_format='d')
    plt.title(title)
    plt.tight_layout()
    plt.show()
```

Partie A — KNN sur *Iris*

```python
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

# Charger les données
iris = load_iris(as_frame=True)
X, y = iris.data, iris.target
labels = iris.target_names

# Split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)

# Entraînement KNN avec k=3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Évaluation
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy (k=3): {acc:.3f}")
plot_conf_mat(y_test, y_pred, labels=labels, title="Iris - KNN (k=3)")

# Étude de k (1 à 15)
ks = list(range(1, 16))
accs = []
for k in ks:
    clf = KNeighborsClassifier(n_neighbors=k)
```

```
    clf.fit(X_train, y_train)
    accs.append(accuracy_score(y_test, clf.predict(X_test)))

plt.plot(ks, accs, marker="o")
plt.xlabel("k (nombre de voisins)")
plt.ylabel("Accuracy test")
plt.title("Iris - Impact de k sur la performance KNN")
plt.grid(True, alpha=0.3)
plt.show()
```

Partie B — Régression Logistique sur *Breast Cancer*

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# Charger les données
bc = load_breast_cancer(as_frame=True)
X, y = bc.data, bc.target
target_names = bc.target_names

# Split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)

# Pipeline : StandardScaler + LogisticRegression
pipe_lr = Pipeline(steps=[
    ("scaler", StandardScaler()),
    ("lr", LogisticRegression(max_iter=500))
])
pipe_lr.fit(X_train, y_train)

# Évaluation
y_pred = pipe_lr.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification report:\n", classification_report(y_test, y_pred,
    target_names=target_names))
plot_conf_mat(y_test, y_pred, labels=target_names, title="Breast Cancer
    - Régression Logistique")

# Courbe ROC / AUC
y_proba = pipe_lr.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, lw=2, label=f"AUC = {roc_auc:.3f}")
plt.plot([0, 1], [0, 1], linestyle="--", lw=1)
plt.xlabel("Faux positifs (FPR)")
plt.ylabel("Vrais positifs (TPR)")
plt.title("Breast Cancer - Courbe ROC (Régression Logistique)")
plt.legend(loc="lower right")
plt.show()
```

Partie C — Arbre de Décision sur *Wine*

```python
from sklearn.datasets import load_wine
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Charger les données
wine = load_wine(as_frame=True)
X, y = wine.data, wine.target
labels = wine.target_names

# Split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)

# Arbre profondeur max = 3
tree_clf = DecisionTreeClassifier(max_depth=3, criterion="gini",
    random_state=42)
tree_clf.fit(X_train, y_train)

y_pred = tree_clf.predict(X_test)
print(f"Accuracy (depth=3, gini): {accuracy_score(y_test, y_pred):.3f}")
plot_conf_mat(y_test, y_pred, labels=labels, title="Wine - Arbre de Dé
    cision (gini, depth=3)")

# Visualisation
plt.figure(figsize=(10, 6))
plot_tree(tree_clf, feature_names=X.columns, class_names=labels, filled=
    True, rounded=True)
plt.title("Wine - Arbre de Décision (gini, depth=3)")
plt.show()
```

À retenir

Points clés

- **KNN** : simple, efficace, mais sensible au choix de k et à l'échelle.
- **Régression Logistique** : interprétable, robuste sur des problèmes binaires.
- **Arbre de Décision** : lisible, puissant mais sujet au surapprentissage.