

TP : Initiation au Machine Learning avec Python

Réalisé par : Amirat Samy

Master EEA – Université de Lorraine

Année universitaire 2024–2025



Table des matières

1	Introduction	2
2	Environnement	2
3	bibliothèques	2
4	Notions fondamentales	2
5	Apprentissage supervisé	3
5.1	Régression simulée	3
5.2	Régression sur données réelles	4
5.3	Classification k-NN	5
6	Apprentissage non supervisé	5
6.1	Réduction de dimension (PCA, t-SNE)	5
6.2	Clustering (KMeans)	6
7	Études de cas	7
7.1	Régression sur données diabète	7
7.2	Classification avec matrice de confusion	8
8	Conclusion	9

1 Introduction

L'intelligence artificielle (IA) est aujourd'hui au cœur de nombreuses avancées technologiques. L'une de ses branches les plus actives est le **machine learning** (apprentissage automatique), qui consiste à permettre à une machine d'apprendre à partir de données, sans avoir été explicitement programmée. Le machine learning est utilisé dans la médecine, les transports, la finance, etc.

Le traitement des données est crucial : nettoyage, normalisation, choix des variables, séparation train/test... Tout cela détermine la performance des modèles.

Dans ce TP, nous avons appliqué des algorithmes fondamentaux de machine learning à l'aide de Python et de bibliothèques comme `scikit-learn`, `pandas`, et `matplotlib`. L'objectif : comprendre, implémenter, tester et évaluer des modèles supervisés et non supervisés.

2 Environnement

— **Python 3.9**

3 bibliothèques

- `numpy` : pour les opérations numériques
- `pandas` : pour la manipulation de données tabulaires
- `matplotlib` et `seaborn` : pour la visualisation
- `scikit-learn` : pour les algorithmes de machine learning

4 Notions fondamentales

Apprentissage supervisé : l'algorithme apprend à partir de données d'entraînement étiquetées, c'est-à-dire que chaque entrée est associée à une sortie connue. Exemple : prédire le prix d'une maison (régression) ou diagnostiquer une maladie (classification).

Apprentissage non supervisé : les données ne sont pas étiquetées. L'algorithme essaie de découvrir des structures ou des regroupements naturels dans les données. Exemple : regrouper des clients selon leurs comportements d'achat (clustering).

Algorithmes utilisés

- **Régression linéaire** : prédire une valeur continue
- **k-NN** : classer selon les k plus proches voisins
- **KMeans** : regrouper automatiquement les données
- **PCA** : réduire la dimension des données de manière linéaire
- **t-SNE** : réduction de dimension non linéaire

5 Apprentissage supervisé

5.1 Régression simulée

Objectif : Dans cette première partie, nous simulons une relation linéaire simple perturbée par du bruit. Le but est d'appliquer la méthode des moindres carrés pour ajuster une droite aux données et prédire de nouvelles valeurs. Ce type de modèle permet de modéliser des phénomènes continus (comme le prix d'un produit ou la température).

Exemple : Le programme suivant génère des points aléatoires autour d'une droite et applique une régression linéaire pour retrouver cette droite à partir des données bruitées.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics

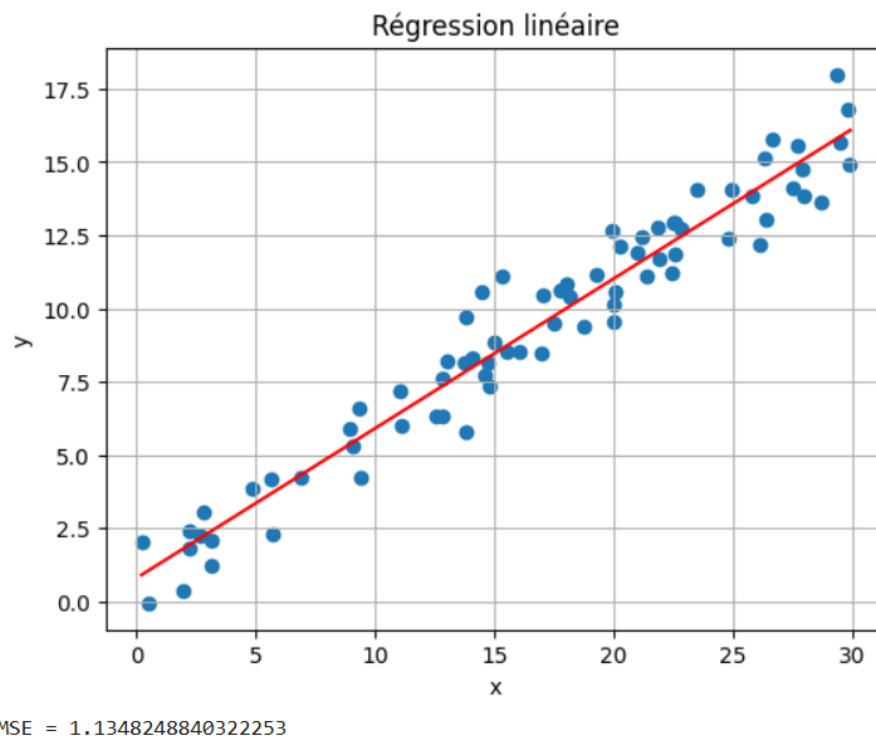
a_real = 0.5
b_real = 1.0
x = 30 * np.random.random(100)
y = a_real * x + b_real + np.random.normal(size=x.shape)

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size
=0.25)
A = np.array([[xj, 1] for xj in xtrain])
a, b = np.linalg.lstsq(A, ytrain, rcond=None)[0]

plt.scatter(xtrain, ytrain)
xline = np.linspace(min(xtrain), max(xtrain), 50)
plt.plot(xline, a * xline + b, color="red")
plt.show()

ypred = a * xtest + b
print("MSE =", metrics.mean_squared_error(ytest, ypred))
```

Illustration du résultat :



5.2 Régression sur données réelles

Objectif : prédire la température en fonction de l'humidité.

```
import pandas as pd
data = pd.read_csv("weatherHistory.csv")
x = data["Humidity"].values
y = data["Temperature (C)"].values
```

5.3 Classification k-NN

Objectif : prédire si une tumeur est bénigne ou maligne.

Le programme suivant charge les données, les découpe en ensembles d'entraînement et de test, puis entraîne le modèle et mesure la précision.

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
print("Accuracy:", knn.score(X_test, y_test))
```

Illustration du résultat :

Accuracy: 0.9649122807017544

6 Apprentissage non supervisé

6.1 Réduction de dimension (PCA, t-SNE)

Objectif : visualiser les chiffres manuscrits en 2D.

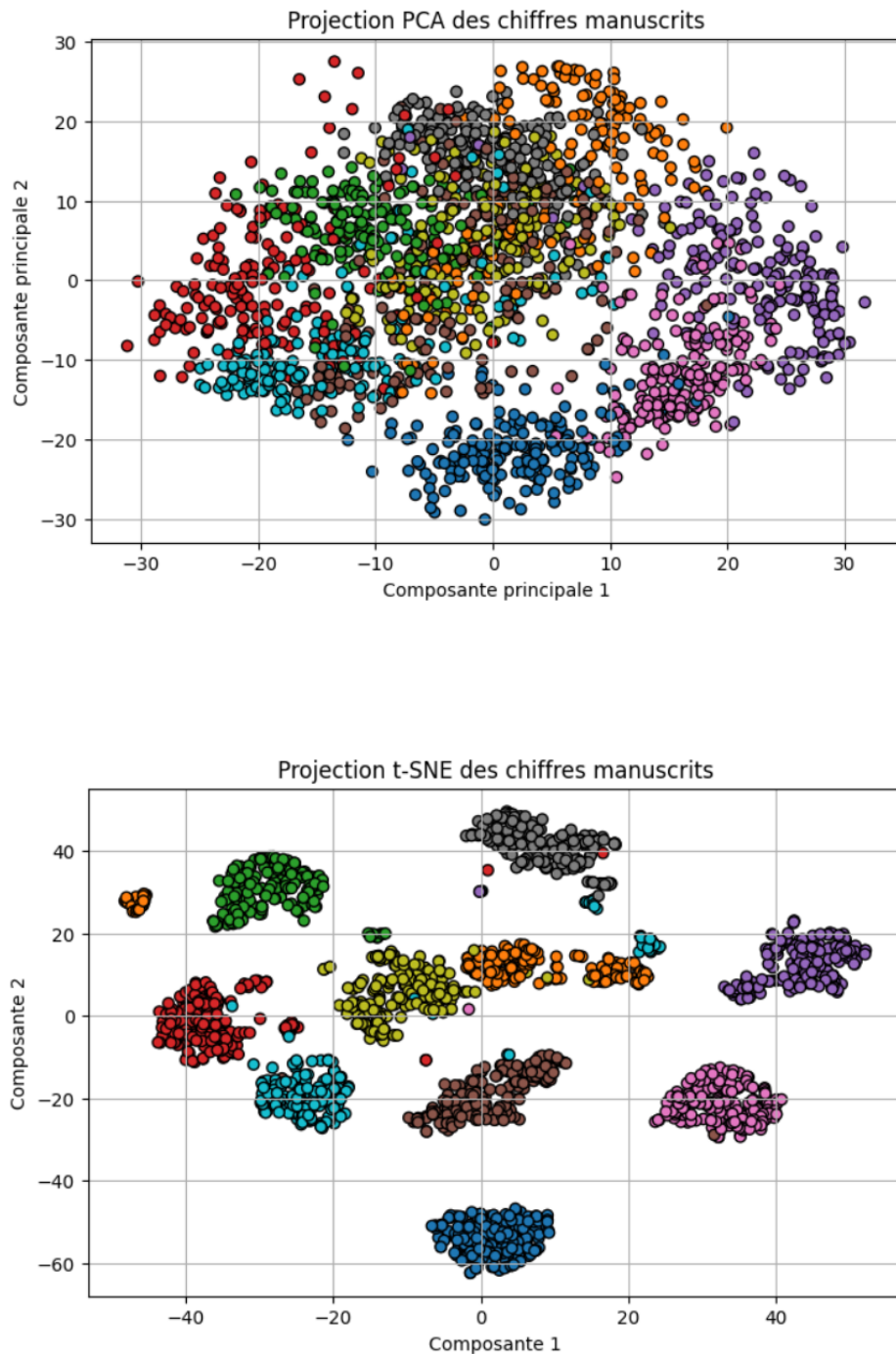
La réduction de dimension permet de simplifier des jeux de données complexes tout en conservant leur structure. Cela facilite la visualisation et parfois améliore les performances des algorithmes :

- **PCA** est une méthode linéaire qui projette les données sur les axes principaux.
- **t-SNE** est une méthode non linéaire souvent utilisée pour les images ou les données très denses.

```
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

X, y = load_digits(return_X_y=True)
pca = PCA(n_components=2).fit_transform(X)
tsne = TSNE(n_components=2).fit_transform(X)
```

Illustration du résultat :



6.2 Clustering (KMeans)

Objectif : regrouper les chiffres manuscrits en 10 clusters.

Le clustering (regroupement) permet de segmenter les données sans avoir besoin d'étiquettes. KMeans regroupe les points en k clusters en fonction de leur proximité. On applique ici KMeans sur les chiffres manuscrits pour voir s'il parvient à retrouver les

bonnes catégories.

Exemple : Le code ci-dessous entraîne un modèle KMeans sur les données, puis tente d'associer chaque cluster à une classe réelle par la méthode du mode.

```
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from scipy.stats import mode

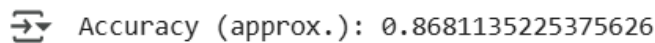
kmeans = KMeans(n_clusters=10)
clusters = kmeans.fit_predict(X)

labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(y[mask], keepdims=True)[0]

print("Accuracy (approx.):", accuracy_score(y, labels))
```

Illustration du résultat :

```
print("Accuracy (approx.):", accuracy_score(y, labels))
```



Bien que KMeans soit un algorithme non supervisé, nous avons comparé les clusters obtenus aux vraies classes pour estimer la justesse de la segmentation, ce qui donne ici une accuracy approximative de 86,8/100.

7 Études de cas

7.1 Régression sur données diabète

On utilise ici des données médicales réelles pour prédire la progression d'une maladie (diabète) en fonction de variables comme l'IMC (indice de masse corporelle).

Exemple : Le code suivant applique une régression simple sur la variable IMC du jeu de données 'load diabetes'.

```
from sklearn.datasets import load_diabetes
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X, y = load_diabetes(return_X_y=True)
X_bmi = X[:, np.newaxis, 2]
model = LinearRegression().fit(X_bmi, y)
print("MSE:", mean_squared_error(y, model.predict(X_bmi)))
```

Illustration du résultat :


```
model = LinearRegression().fit(X_bmi, y)
print("MSE:", mean_squared_error(y, model.predict(X_bmi)))
```

MSE: 3890.456585461273

7.2 Classification avec matrice de confusion

Ce cas approfondit la classification binaire en ajoutant une évaluation par matrice de confusion, qui détaille les vrais positifs, faux positifs, faux négatifs et vrais négatifs.

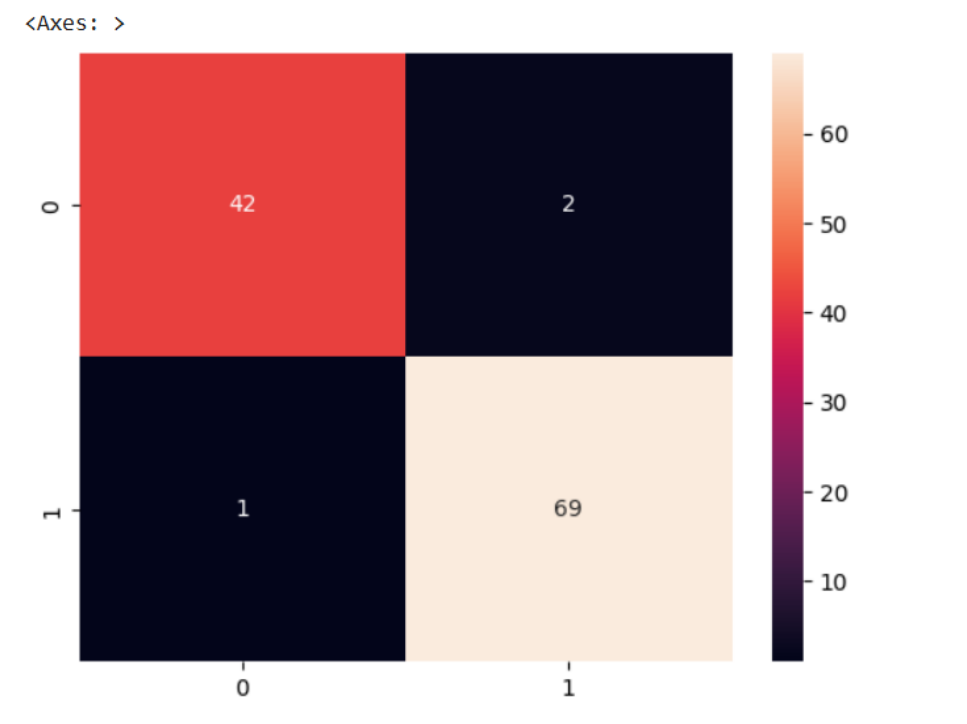
Exemple : On réutilise ici k-NN pour classer les tumeurs du sein, mais cette fois on affiche aussi la matrice de confusion pour mieux comprendre les erreurs du modèle.

```
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns

k_optimal = 7
knn = KNeighborsClassifier(n_neighbors=k_optimal)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
```

Illustration du résultat :



Dans ce cas, on obtient une matrice de confusion équilibrée, avec très peu d'erreurs. Cela signifie que le modèle k-NN a bien appris à distinguer les deux classes (tumeurs bénignes et malignes). Le nombre élevé de vrais positifs (en haut à gauche) et de vrais négatifs (en bas à droite) indique une bonne performance globale. On peut conclure que le modèle est fiable pour cette tâche de classification.

8 Conclusion

Ce TP m'a permis d'explorer différentes méthodes de machine learning, de la régression à la classification, en passant par la réduction de dimension et le clustering. J'ai appris à manipuler des jeux de données réels, visualiser les résultats et évaluer les performances des modèles.